UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Facultad de Ciencias

Coloreando Gráficas con Tres Metaheurísticas Basadas en Colonia de Hormigas

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Licenciado en Ciencias de la Computación

PRESENTA:

José Luis Vázquez Lázaro

TUTORA:

Dr. María de Luz Gasca Soto

Ciudad Universitaria, Ciudad de México, 2021

A mi esfuerzo, a mi madre (Florentina) y a ti, Horacio.

Agradecimientos

A la UNAM, por permitirme cumplir una promesa.

A la Facultad de Ciencias, por brindarme su conocimiento.

A la Doctora Lucy por su confianza, guía y consejo.

A la familia que encontré: ¡Gracias Laura!

A mi hermana (Claudia) y a mis sobrinos (Astrid, Iván y Mateo), por su amor y su admiración a mi dedicación.

A mi madre, por todo lo que soy.

A Horacio, por estar en mi vida y apoyarme en este largo camino.

Índice General

Introducción

El **Problema de la Coloración de Gráficas** es de suma importancia. Lamentablemente, este problema no se ha podido solucionar de manera eficiente, es decir, no existe hasta ahora un algoritmo de tiempo polinomial que lo resuelva; ya que el Problema de la Coloración de Gráficas es un problema **NP-Completo** y, como sabemos, no se ha dado respuesta a la pregunta ¿**P** = **NP**? A pesar de que no existen métodos exactos y eficientes para resolver este problema, no se ha abandonado, ya que se ha optado por atacarlo a través de métodos de aproximación: ya sea con algoritmos de aproximación, con desempeño computacional garantizado; como con heurísticas, las cuales, en general, construyen una buena solución de forma rápida [1].

Dada una gráfica simple $G = (V, E)$, de tamaño n, el mejor algoritmo de aproximación que se conoce para el Problema de la Coloración de Gráficas devuelve, en tiempo polinomial, una coloración válida, con a lo más

$$\frac{n(\log(\log(n)))^2}{(\log(n))^3}\chi(G)$$

colores. Sin embargo, se sabe que, a menos que P = NP, no puede existir un algoritmo de aproximación que devuelva, en tiempo polinomial, una coloración válida para G con a lo más $n^{(1/7)-\varepsilon}\chi(G)$, para todo $\varepsilon > 0$. Dado que para el Problema de la Coloración de Gráficas los algoritmos de aproximación no son muy prometedores, muchos de los trabajos actuales se han concentrado en el diseño de heurísticas para atacar el problema, como lo son las **Metaheurísticas Basadas en Colonia de Hormigas**.

Este trabajo tiene por objetivo principal analizar y comparar la calidad de las soluciones producidas por tres diferentes Metaheurísticas Basadas en Colonia de Hormigas para el Problema de la Coloración de Gráficas. Para llevar a cabo este objetivo, se realizaron las siguientes tareas:

1. Exponer el marco teórico de las Metaheurísticas Basadas en Colonia de Hormigas, formalmente llamado Optimización por Colonia de Hormigas.

2. Explicar e implementar tres diferentes versiones de Metaheurísticas Basadas en Colonia de Hormigas, bajo una misma representación para las gráficas.

[1] Ver Apéndice B **Complejidad Computacional**

3. Analizar el desempeño computacional de cada una de las heurísticas descritas sobre un conjunto dado de gráficas a través de una serie de ejecuciones. Así como analizar el impacto del uso de feromonas en la calidad de la solución construida por cada una de éstas.
4. Comparar los resultados obtenidos.

Estructura de la tesis

- Capítulo 1 **Optimización por Colonia de Hormigas**. Se da una breve descripción del marco teórico de este esquema metaheurístico y mencionan algunas de sus aplicaciones más importantes.
- Capítulo 2 **El Problema de la Coloración de Gráficas**. Se describe formalmente este problema de optimización, su versión de decisión y su naturaleza como problema NP-Completo. Además, se proporcionan algunas cotas del número cromático de una gráfica.
- Capítulo 3 **La Metaheurística *ANTCOL***. Se dan los detalles de esta metaheurística; esto es, su estrategia general, sus parámetros, así como las funciones complementarias y auxiliares que utiliza.
- Capítulo 4 **La Metaheurística *D-ANTCOL***. Se dan las características de esta metaheurística; es decir, su estrategia general, sus parámetros y las funciones complementarias y auxiliares que utiliza.
- Capítulo 5 **La Metaheurística *ABAC***. Se dan los detalles de esta metaheurística; esto es, su estrategia general, sus parámetros, además de las funciones complementarias y auxiliares que utiliza.
- Capítulo 6 **Experimentos y Resultados**. Se describen los detalles de los experimento realizados: los ejemplares y su codificación, los detalles de la implementación de las metaheurísticas y los resultados obtenidos.
- Apéndice A **Teoría de Gráficas**. Se exponen las definiciones básicas sobre la Teoría de Gráficas, para establecer la notación usada en este trabajo.
- Apéndice B **Complejidad Computacional**. Se presentan los conceptos y resultados básicos de la Complejidad Computacional que le dan sustento a este trabajo.
- Apéndice C **Métodos Constructivos para Coloración de Gráficas**. Se describen las principales características de los algoritmo polinomiales conocidos que construyen una q-coloración válida, para una gráfica dada.
- Apéndice D **Aplicaciones de la *ACO***. Se proporcionan las referencias más relevantes de las principales aplicaciones de la metaheurística ACO a lo largo del tiempo.

- Apéndice E **Construcción de ejemplares**. Se dan los detalles de la construcción de cada uno de los tipos de ejemplares utilizados en los experimentos.

Capítulo 1

Optimización por Colonia de Hormigas

A metaheuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality.
—*Tabu Search*, Fred Glover & Manuel Laguna, 1998

De acuerdo con Coello [1], la abundancia de problemas de optimización de alto grado de dificultad, en el mundo real, ha sido una de las principales causas por la que en los últimos años se ha desarrollado un número considerable de **técnicas heurísticas**[2] (técnicas de aproximación adaptables), que permiten obtener al menos un resultado subóptimo en un período de tiempo relativamente corto en problemas que resultaría impráctico resolver mediante fuerza bruta; es decir, enumerando todas las posibles soluciones para escoger de entre ellas a la mejor. Las heurísticas más conocidas hoy en día no hacen más que adaptar ideas conocidas desde hace mucho tiempo en otras disciplinas. Por ejemplo, los Algoritmos Genéticos emulan los mecanismos de la evolución; los métodos de Flujo en Redes se fundamentan en ideas de la electricidad y la hidráulica; por otro lado, el Recocido (o Templado) Simulado se basa en un proceso físico de la industria metalúrgica. Similarmente, la técnica conocida como **Optimización por Colonia de Hormigas**, *ACO*[3], se basa en el comportamiento de las colonias de hormigas para buscar alimento y se utiliza como una **metaheurística** (técnica heurística de alto nivel) para resolver problemas de optimización combinatoria. Esto significa que la técnica tiene que combinarse con un mecanismo de búsqueda y lo que hace, básicamente, es evitar que dicho mecanismo quede atrapado en un óptimo local.

El material expuesto en las siguientes secciones de este capítulo, está basado en el libro *"Ant Colony Optimization"* de Dorigo y Stützle [6].

[2] Comúnmente llamadas simplemente *heurísticas*.

[3] Siglas en inglés de *Ant Colony Optimization*. En adelante se utilizará el acrónimo *ACO* para referirse a esta metaheurística.

1.1. Antecedentes

La primera implementación de la *ACO* fue presentada por Marco Dorigo en su tesis doctoral [3], él la llamó **Sistema de Hormigas**, *AS*[4], y fue el resultado de una investigación acerca del enfoque que se le podía dar a la **Optimización Combinatoria** a través de la Inteligencia Artificial que Dorigo realizó en el Politécnico de Milano en colaboración con Alberto Colorni y Vittorio Maniezzo. El *AS* se aplicó inicialmente al Problema del Agente Viajero y al Problema de la Asignación Cuadrática.

En 1999, la *ACO* fue definida formalmente por Dorigo, Di Caro y Gambardella, [4, 5], lo que derivó en el estudio formal de las metaheurísticas basadas en colonia de hormigas, siendo la *ACO* su marco teórico. Desde entonces, este campo se enfoca en el estudio de modelos derivados de la observación del comportamiento de las hormigas reales y utiliza estos modelos como fuente de inspiración para el diseño de nuevas metaheurísticas para la solución de problemas de optimización combinatoria y problemas de control distribuido. La idea principal es que los principios de autoorganización que permiten el comportamiento altamente coordinado de las hormigas reales pueden explotarse para coordinar poblaciones de agentes artificiales (hormigas artificiales) que colaboren para resolver problemas computacionales.

Varios de los diferentes aspectos del comportamiento de las colonias de hormigas han inspirado diferentes tipos de metaheurísticas. Por ejemplo: la búsqueda de alimento, la división del trabajo en el nido, la clasificación de las crías y el transporte cooperativo. En todos estos ejemplos, las hormigas coordinan sus actividades a través de **estigmergia**[5]. Por ejemplo, una hormiga buscadora de alimento deposita una sustancia química en el suelo que aumenta la probabilidad de que otras hormigas sigan el mismo camino que ésta. Los biólogos han demostrado que el comportamiento de muchas sociedades de insectos se puede explicar a través de modelos bastante simples en los que sólo está presente la comunicación de tipo estigmergia. En otras palabras, los biólogos han demostrado que a menudo es suficiente considerar la comunicación de tipo estigmergia para explicar cómo los insectos sociales pueden lograr la autoorganización. La idea detrás de las metaheurísticas basadas en colonia de hormigas es utilizar una forma de estigmergia artificial para coordinar sociedades de agentes artificiales.

1.2. Comportamiento alimentario de las hormigas

Las colonias de hormigas, y en general los insectos que forman sociedades, son sistemas distribuidos que, a pesar de la simplicidad de sus individuos, presentan una organización social altamente estructurada. Como resultado de esta organización, las colonias de hormigas pueden realizar tareas complejas que, en algunos casos, superan

[4]Siglas en inglés de *Ant System*. En adelante se utilizará el acrónimo *AS* para referirse a esta implementación.

[5]Una forma de comunicación indirecta entre agentes, donde el rastro dejado en el ambiente por una acción de un agente estimula el funcionamiento de una acción subsecuente por parte de otro agente igual o uno diferente.

con creces las capacidades individuales de una sola hormiga.

La facultad de percepción visual de muchas especies de hormigas es muy rudimentaria e, incluso, hay especies de hormigas que están completamente ciegas. Por lo que un hallazgo de suma importancia en las primeras investigación sobre el comportamiento de las hormigas fue que la mayor parte de la comunicación entre individuos de la colonia, o entre la colonia y su medio ambiente, está basada en el uso de sustancias químicas producidas por las hormigas. Estas sustancias son llamadas **feromonas**. Existe un tipo particular de feromona que es de gran importancia para la vida social de algunas especies de hormigas, la **feromona de rastro**. La feromona de rastro es utilizada por algunas especies de hormigas, como la *Lasius niger* o la hormiga argentina *Iridomyrmex humilis*, para marcar caminos en el suelo, por ejemplo, caminos desde las fuentes de alimento al nido. Al detectar estas feromonas, las hormigas recolectoras pueden seguir el camino hacia las fuentes de alimientos descubiertas por otras hormigas. Este comportamiento colectivo de tendimiento de rastros y seguimiento de rastros por el cual una hormiga se ve influenciada por el rastro químico dejado por otras hormigas corresponde a una comunicación de tipo estigmergia (la Figura 1.1 muestra este hecho) y es la fuente de inspiración de la *ACO*.

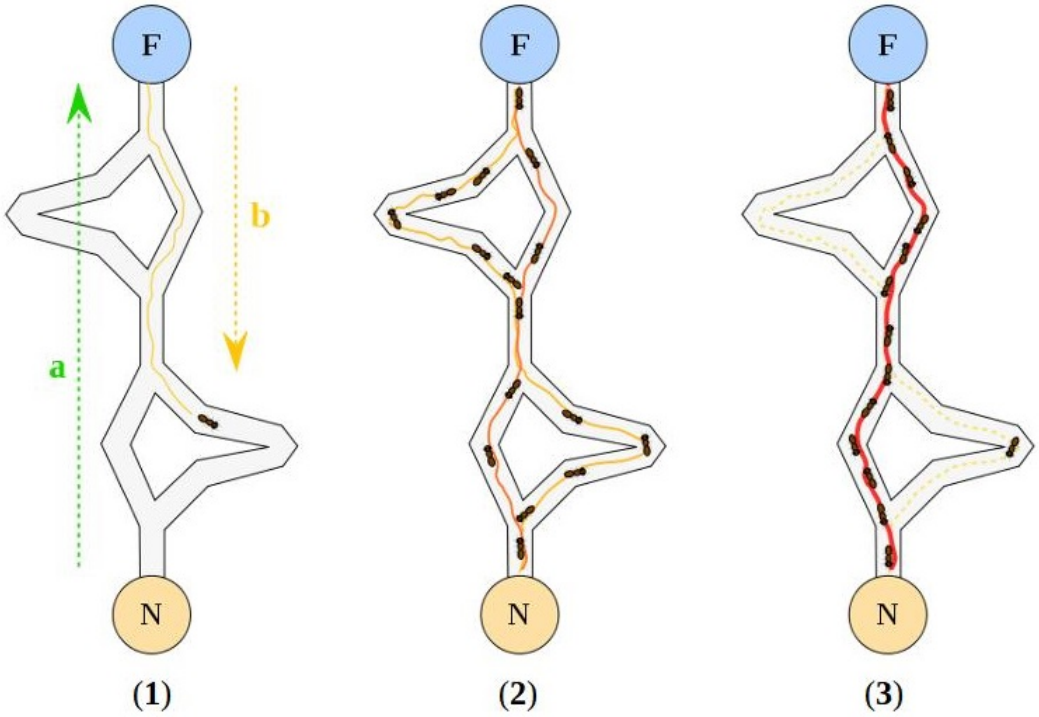

Figura 1.1: Proceso de estigmergia en las hormigas: **(1)** Inicio de la búsqueda de alimento. **(2)** Tendido de la feromona de rastro por parte de las hormigas buscadoras. **(3)** Seguimiento del rastro de feromonas por parte de la hormigas recolectoras.

1.3. La metaheurística *ACO*

Los problemas de optimización combinatoria son intrigantes porque a menudo son fáciles de definir pero muy difíciles de resolver. Esto se debe principalmente a que muchos de estos problemas son NP-Completos; esto es, se cree firmemente que no se pueden resolver de manera óptima en tiempo polinomial. Por lo que, para resolver ejemplares complejos, se usan a menudo técnicas de aproximación que devuelvan soluciones subóptimas en un tiempo relativamente corto, utilizando algún conocimiento específico del problema para construir o mejorar soluciones (adaptación). A este tipo de técnicas se les llama **heurísticas**.

Desde la última década del siglo XX, muchos investigadores centraron su atención en una nueva clase de heurísticas, las **metaheurísticas** o **heurísticas de alto nivel**. Una metaheurística es una heurística que utiliza conceptos algorítmicos para guiar y modificar a otras heurísticas con el objetivo de evitar que dichas heurísticas queden atrapadas en óptimos locales. El uso de metaheurísticas ha aumentado significativamente la capacidad de encontrar soluciones de muy alta calidad para problemas de optimización combinatoria NP-Completos de gran relevancia en un tiempo razonable.

Una metaheurística particularmente exitosa es la *ACO*. Comenzando con *AS* (su primera implementación), se desarrollaron varias metaheurísticas basadas en la *ACO* que se aplicaron con un éxito considerable a una gran variedad de problemas de optimización combinatoria, tanto académicos como reales. La *ACO* es una heurística de alto nivel en la que una colonia de hormigas artificiales es encaminada a buscar y encontrar buenas soluciones a problemas de optimización combinatoria que se sabe son NP-Completos. La cooperación es un componente de diseño clave en la *ACO*: Lo conveniente es asignar los recursos computacionales a un conjunto de agentes relativamente simples (hormigas artificiales) dotados de una comunicación de tipo estigmergia, donde se espera que las buenas soluciones encontradas sean una propiedad emergente de la interacción cooperativa de los agentes.

Lo anterior implicó dos hechos importantes:

1. La *ACO* se convirtió en el marco teórico (o marco metaheurístico) de todas las Metaheurísticas Basadas en Colonia de Hormigas.

2. Todas las Metaheurísticas Basadas en Colonia de Hormigas pasaron a ser implementaciones particulares de la *ACO*. De hecho, abusando del lenguaje, fueron bautizadas como **Algoritmos *ACO***[6].

1.4. ¿Dónde se ha aplicado la *ACO*?

La Tabla 1.1 es un resumen de la Tabla C.1 del Apéndice D, presentada por Dorigo y Stützle en [6], que muestra la cronología de las aplicaciones más importantes de la *ACO*.

[6] Un nombre contradictorio, pues, en estricto sentido, ninguna heurística es un algoritmo.

Tabla 1.1: Aplicaciones de la *ACO* (resumen) [6]

Tipo de problema	**Nombre del problema**
Enrutamiento	Agente Viajero
	Enrutamiento Vehicular
	Ordenamiento Secuencial
Asignación	Asignación Cuadrática
	Coloración de Gráficas
	Asignación Generalizada
	Asignación de Canales
	Asignación de Horarios
Calendarización	Sistemas de producción *Job Shop*
	Sistemas de producción *Open Shop*
	Sistemas de producción *Flow Shop*
	Tardanza Total
	Tardanza Total Ponderada
	Calendarización de Proyectos
	Sistemas de producción *Group Shop*
Subconjunto	Mochila Múltiple
	Conjunto Independiente
	Asignación de Redundancia
	Cubierta de Conjuntos
	Clan
Otro	Subsecuencia Común Más Larga
	Satisfacción de Restricciones
	Plegado de Proteínas 2D-HP
	Empaquetamiento en cajas
Aprendizaje Automático	Reglas de clasificación
	Redes Bayesianas
	Sistemas Difusos
Enrutamiento en Redes	Enrutamiento en Redes Orientadas a la Conexión
	Enrutamiento en Redes No Orientadas a la Conexión
	Enrutamiento en Redes Ópticas

Capítulo 2

El Problema de la Coloración de Gráficas

The Four Color Conjecture. *The regions of every map can be colored with four or fewer colors in such a way that every two regions sharing a common boundary are colored differently.*
—Francis Guthrie, 1852

En ese capítulo se expondrá el Problema de la Coloración de Gráficas y se demostrará su naturaleza como un problema NP-Completo.

2.1. Terminología básica

En la presente sección damos las definiciones básicas sobre Coloración de Gráficas. El marco teórico que da sustento a estas definiciones está expuesto en el Apéndice A.

En el resto de esta sección, en las siguientes secciones de este capítulo y en el resto de capítulos, trabajaremos únicamente con gráficas simples. Por lo que usaremos los términos **gráfica simple** y **gráfica** indistintamente.

Definición 2.1 Sea $G = (V, E)$ una gráfica. Una **coloración** de G es una función $\mathscr{C} : G.V \to C \subseteq \mathbb{Z}^+$. En este caso, el conjunto C es llamado **gama de colores** y sus elementos son llamados **colores**.

Definición 2.2 Sean $G = (V, E)$ una gráfica y W un subconjunto propio de $G.V$. Una **coloración parcial** de G es una función $\mathscr{C} : W \to C \subseteq \mathbb{Z}^+$. De manera análoga, a la Definición 2.1, el conjunto C es llamado **gama de colores** y sus elementos son llamados **colores**. Para cada $v \in G.V$, diremos que v es un **vértice coloreado** por $\mathscr{C}$, si $v \in W$. En caso contrario, diremos que v es un **vértice no coloreado** por $\mathscr{C}$.

Definición 2.3 Sean $G = (V, E)$ una gráfica, $\mathscr{C}$ una coloración parcial de G y v un vértice de G no coloreado por $\mathscr{C}$. El **grado de saturación** de v bajo $\mathscr{C}$, denotado por

$dsat_{\mathscr{C}}(v)$, es el número de colores distintos asignados a los vértices de N, donde N es el conjunto $\{ u \in Adj_G(v) \mid u$ es un vértice coloreado por $\mathscr{C} \}$.

Definición 2.4 Sean $G = (V, E)$ una gráfica y q un entero positivo tal que $q \leqslant |G.V|$. Una ***q*-coloración** de G es una función $\mathscr{C} : G.V \to \{1, 2, \ldots, q\}$ suprayectiva. El entero q es llamado **número de colores** de $\mathscr{C}$.

Definición 2.5 Sean $G = (V, E)$ una gráfica, $\mathscr{C}$ una q-coloración de G e $i \in \{1, 2, \ldots, q\}$ un color. El conjunto $\{ v \in G.V \mid \mathscr{C}(v) = i \}$ es llamado **clase de color** y se denota como $V(i)$, para cada $i \in \{1, 2, \ldots, q\}$.

Definición 2.6 Sean $G = (V, E)$ una gráfica y $\mathscr{C}$ una q-coloración de G. Decimos que $\mathscr{C}$ es **válida** (o **propia**) si y sólo si, para cada $uv \in G.E$, $\mathscr{C}(u) \neq \mathscr{C}(v)$.

Definición 2.7 Sea $G = (V, E)$ una gráfica. Diremos que G es ***q*-coloreable** si admite una q-coloración válida.

Definición 2.8 Sea $G = (V, E)$ una gráfica. El mínimo entero $q \in \mathbb{Z}^+$ para el cuál G es q-coloreable es llamado **número cromático** de G y se denota por $\chi(G)$.

Definición 2.9 Sea $G = (V, E)$ una gráfica. Cualquier q-coloración propia de G tal que $q = \chi(G)$, es llamada **coloración óptima** de G.

Los siguientes enunciados son consecuencia inmediata de las definiciones:

Resultado 2.1 Para toda gráfica $G = (V, E)$ de orden n, se tiene que:

- Toda q-coloración de G es una coloración de G.
- Toda coloración parcial de G puede extenderse a una coloración de G. En particular, si G tiene una coloración parcial suprayectiva y con gama de colores $\{1, 2, \ldots, p\}$, ésta puede extenderse a una q-coloración de G, con $p \leqslant q$.
- Toda coloración de G que utilice exactamente q colores, puede transformarse en una q-coloración de G al renombrar sus colores con la gama de colores $\{1, 2, \ldots, q\}$.
- $\chi(G)$ siempre existe y está acotado, trivialmente, de la siguiente manera:

  $$1 \leqslant \chi(G) \leqslant n$$

  pues si $G = K_n$, entonces $\chi(G) = n$; y si $G = (K_n)^c$, entonces $\chi(G) = 1$.
- Si G es q-coloreable, entonces $\chi(G) \leqslant q \leqslant n$. Es decir, el número de colores de cualquier coloración válida de G es una cota superior de $\chi(G)$.
- Para todo $W \subseteq G.V$, W es un conjunto independiente si y sólo si W es un conjunto estable.

- Si G es q-coloreable, entonces para todo $i \in \{1, 2, \ldots, q\}$, $V(i)$ es un conjunto independiente. Lo que implica, para todo $i \in \{1, 2, \ldots, q\}$, $V(i)$ es un conjunto estable.

- Si G es q-coloreable, entonces $\{\ V(i) \mid i \in \{1, 2, \ldots, q\}\ \}$ es una partición de $G.V$ en q conjuntos estables. Por otro lado, si existe una partición de $G.V$ en q conjuntos estables, entonces G es q-coloreable y los q conjuntos de la partición son las q clases de color correspondientes. Por lo tanto, G es q-coloreable si y sólo si existe una partición de $G.V$ en q conjuntos estables.

- Si $\mathscr{C} : W \subset G.V \to \{1, 2, \ldots, p\}$ es una coloración parcial y suprayectiva de G tal que ningún par de vértices adyacentes en W tienen el mismo color, $\mathscr{C}$ induce una partición de W en p conjuntos estables. Esta partición de W en p conjuntos independientes se puede extender a una partición de $G.V$ en q conjuntos estables, con $p \leqslant q$. A este tipo de coloración parcial la llamaremos **coloración parcial válida**.

2.2. El Problema del Número Cromático

En esta sección se describe el Problema del Número Cromático, la versión de decisión del Problema de la Coloración de Gráficas, y se demuestra que es un problema NP-Completo.

El **Problema de la Coloración de Gráficas** es el problema de optimización combinatoria definido por:

1. Ejemplar Genérico: Una gráfica $G = (V, E)$ de orden n.

2. Enunciado de Optimización: Determinar el mínimo $q \in \mathbb{Z}^+$ para el cual G tiene una q-coloración válida, es decir, determinar $\chi(G)$.

La versión de decisión de este problema es llamado **Problema del Número Cromático**, *CN*[7], y se define de la siguiente manera:

1. Ejemplar Genérico: Una gráfica $G = (V, E)$ de orden n y un entero positivo B tal que $2 < B \leqslant n$.

2. Pregunta de Decisión: ¿Existe una coloración válida de G con a lo más B colores?

A continuación, mostraremos que el problema *CN* es NP-Completo mediante una reducción (transformación) del problema *3SAT* a éste. La definición formal del problema *3SAT* es la siguiente:

[7]Siglas en inglés de *Cromatic Number*. En adelante se utilizará el acrónimo *CN* para referirse a este problema.

1. Ejemplar Genérico: Un conjunto $U = \{x_1, x_2, \ldots, x_r\}$ de variables y una colección $C = \{c_1, c_2, \ldots, c_s\}$ de cláusulas, sobre U, tal que para cada $i \in \{1, 2, \ldots, s\}$, $|c_i| = 3$.

2. Pregunta de Decisión: ¿Existe una asignación de verdad $\tau : U \to \{0, 1\}$ tal que para cada $i \in \{1, 2, \ldots, s\}$, $\tau(c_i) = 1$?

Teorema 2.1 El problema *CN* es NP-Completo.

Demostración.

Por demostrar: *(i)* *CN*$\in$ NP.
(ii) Existe un problema Π NP-Completo tal que $\Pi \propto_p$ *CN*.

(i) Por demostrar que existe un algoritmo no-determinístico polinomial para *CN*.

El algoritmo **Check-CN** mostrado en el Pseudocódico 2.1 es un algoritmo no-determinístico para *CN*. La fase adivinadora toma tiempo $O(n)$, pues las líneas 2, 4 y 5 requieren tiempo constante. Por otro lado, la fase verificadora, en el peor caso, toma tiempo $O(n^2)$. Lo que implica, **Check-CN** toma tiempo $O(n^2)$. Lo que indica que **Check-CN** es polinomial.

Por lo tanto, **Check-CN** es un algoritmo no-determinístico polinomial para *CN*.

Por lo tanto, *CN*$\in$ NP.

Pseudocódigo 2.1 Algoritmo **Check-CN**

Entrada: Un ejemplar I de *CN*, es decir, una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$, y un entero positivo B tal que $2 < B \leqslant n$.
Salida: *YES*, si G tiene una coloración válida con a lo más B colores. *NO*, en otro caso.

```
1:  /* Fase adivinadora */
2:  Sea colouring un arreglo de tamaño n
3:  for i ← 1 to n do
4:      Sea face el resultado de lanzar un dado equilibrado de B caras
5:      colouring[i] ← face
6:  end for
7:
8:  /* Fase verificadora */
9:  for each e ∈ G.E do
10:     Sean i, j ∈ {1, 2, ..., n} tales que e = v_i v_j
11:     if ( colouring[i] == colouring[j] ) then
12:         return NO
13:     end if
14: end for
15: return YES
```

(ii) Proponemos Π = *3SAT*

Por demostrar: Existe $\mathscr{F}$: *3SAT* $\to$ *CN* tal que:
(a) $\mathscr{F}$ puede ser computada en tiempo polinomial.
(b) Para cada $I = (U, C) \in$ *3SAT*, existe una asignación de verdad $\tau : U \to \{0, 1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1, 2, \ldots, s\}$, si y sólo si en $\mathscr{F}(I) = (G, B)$, G es B-coloreable.

Proponemos $\mathscr{F}$: *3SAT* $\to$ *CN* definida de la siguiente manera:

Sea $I = (U, C) \in$ *3SAT*, entonces $\mathscr{F}(I) = (G, B)$ es el ejemplar de *CN* donde:

- $G.V = \{x_1, x_2, \ldots, x_r\} \cup \{\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_r\} \cup \{y_1, y_2, \ldots, y_r\} \cup \{c_1, c_2, \ldots, c_s\}$.

- $G.E = \{\ x_i\overline{x}_i \mid i \in \{1,2,\ldots,r\}\ \} \cup$
  $\{\ y_iy_j \mid i,j \in \{1,2,\ldots,r\} \ \wedge\ i \neq j\ \} \cup$
  $\{\ y_ix_j \mid i,j \in \{1,2,\ldots,r\} \ \wedge\ i \neq j\ \} \cup$
  $\{\ y_i\overline{x}_j \mid i,j \in \{1,2,\ldots,r\} \ \wedge\ i \neq j\ \} \cup$
  $\{\ x_iC_j \mid i \in \{1,2,\ldots,r\} \ \wedge\ j \in \{1,2,\ldots,s\} \ \wedge\ x_i \notin C_j\} \cup$
  $\{\ \overline{x}_iC_j \mid i \in \{1,2,\ldots,r\} \ \wedge\ j \in \{1,2,\ldots,s\} \ \wedge\ \overline{x}_i \notin C_j\}.$
- $B = r + 1.$

La Figura 2.1 muestra un ejemplo de esta transformación.

(a) De la definición de $\mathscr{F}$ se sigue que: la gráfica G tiene $3r + s$ vértices y a lo más

$$\binom{r}{2} + 2r^2 - r + s(2r - 1)$$

aristas, y B es construido en tiempo constante. Lo que implica que tanto G como B pueden ser construidos en tiempo polinomial. Entonces, $\mathscr{F}(I)$ puede ser construida en tiempo polinomial, para toda $I \in$ *3SAT*.

Por lo tanto, $\mathscr{F}$ puede ser computada en tiempo polinomial.

(b) Sea $I \in$ *3SAT*.

$\Rightarrow$] Supongamos que existe una asignación de verdad $\tau : U \to \{0,1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1,2,\ldots,s\}$.

Por demostrar: en $\mathscr{F}(I) = (G,B)$, G es B-coloreable.

Como τ es una asignación de verdad que satisface todas las claúsulas en C, para cada $i \in \{1,2,\ldots,s\}$ existe $i_j \in \{1,2,\ldots,r\}$ tal que la literal z_{i_j} (donde z_{i_j} es x_{i_j} o $\overline{x}_{i_j}$) de la claúsula c_i tiene valor de verdad T. Lo que implica, dada la construcción de G, que la coloración obtenida de la siguiente manera:

- Para toda $i \in \{1,2,\ldots,s\}$, colorear los vértices c_i y z_{i_j} con el color i_j, el vértice $\overline{z}_{i_j}$ con el color $r+1$ y cualquier par de vértices $x_i, \overline{x}_i$, que no han sido coloreados, con los colores i y $r+1$ respectivamente.
- Para toda $j \in \{1,2,\ldots,r\}$, colorear el vértice y_j con el color j.

es una $(r+1)$-coloración válida de G. Lo que implica que G es $(r+1)$-coloreable.

Por lo tanto, como $B = r + 1$, en $\mathscr{F}(I) = (G,B)$, G es B-coloreable.

$\Leftarrow$] Supongamos que en $\mathscr{F}(I) = (G,B)$, G es B-coloreable.

Por demostrar: existe una asignación de verdad $\tau : U \to \{0,1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1,2,\ldots,s\}$.

Nótese que si $r \leqslant 4$, la afirmación es trivialmente cierta, pues sólo será necesario analizar las 16 posibles asignaciones de verdad para U para determinar si todas las claúsulas en C son satisfacibles. Por lo tanto, podemos suponer que $r > 4$.

Como G es B-coloreable, entonces G tiene una B-coloración válida. Lo que implica que G tiene una $(r+1)$-coloración válida, pues $B = r + 1$. Por otro lado, podemos suponer que bajo esta coloración, el vértice y_i ha sido coloreado con el color i, para toda $i \in \{1,2,\ldots,r\}$, pues la subgráfica inducida por $\{y_1, y_2, \ldots, y_r\}$ es completa. Además, dado que el vértice y_i es adyacente al par de vértices $\{x_j, \overline{x}_j\}$, para toda $i,j \in \{1,2,\ldots,r\}$ tal que $i \neq j$, se sigue que, bajo esta coloración, uno y sólo uno de los vértices del par $\{x_j, \overline{x}_j\}$ debe recibir el color $r+1$.

Ahora, de la construcción de G y dado que cada claúsula en C tiene exactamente tres literales y $r > 4$, se sigue que el vértice c_j es adyacente a los vértices x_l y $\overline{x}_l$, para cada

$j \in \{1, 2, \ldots, s\}$ y para algún $l \in \{1, 2, \ldots, r\}$. Entonces, dado que uno y sólo uno de los vértices del par $\{x_l, \overline{x}_l\}$ debe recibir el color $r+1$, el vértice c_j no puede ser coloreado con el color $r+1$. Luego, cada vértice c_j debe ser coloreado con uno de los colores $\{1, 2, \ldots, r\}$. Lo que implica que la asignación de verdad $\tau : U \to \{0, 1\}$ tal que,

- $\tau(x_i) = F$, si el vértice x_i del par $\{x_i, \overline{x}_i\}$ está coloreado con el color $r+1$; y
- $\tau(x_i) = T$, si el vértice $\overline{x}_i$ del par $\{x_i, \overline{x}_i\}$ está coloreado con el color $r+1$,

satisface a todas las claúsulas en C, para toda i con $1 \leqslant i \leqslant r$. Esto es cierto, ya que cada vértice c_j debe estar coloreado con el mismo color de algún vértice correspondiente a una literal contenida en la claúsula c_j y esa literal tiene valor de verdad T.

De esta manera, existe una asignación de verdad $\tau : U \to \{0, 1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1, 2, \ldots, s\}$.

Entonces, para cada $I = (U, C) \in$ *3SAT*, existe una asignación de verdad $\tau : U \to \{0, 1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1, 2, \ldots, s\}$, si y sólo si en $\mathscr{F}(I) = (G, B)$, G es B-coloreable.

Por lo tanto, existe $\mathscr{F} :$ *3SAT* $\to$ *CN* tal que:

(a) $\mathscr{F}$ puede ser computada en tiempo polinomial.

(b) Para cada $I = (U, C) \in$ *3SAT*, existe una asignación de verdad $\tau : U \to \{0, 1\}$ tal que $\tau(c_i) = 1$, para toda $i \in \{1, 2, \ldots, s\}$, si y sólo si en $\mathscr{F}(I) = (G, B)$, G es B-coloreable.

Por lo tanto, existe un problema Π NP-Completo tal que $\Pi \propto_p$ *CN*.

Ergo, por *(i)* y *(ii)*, *CN* es NP-Completo.

Q. E. D.

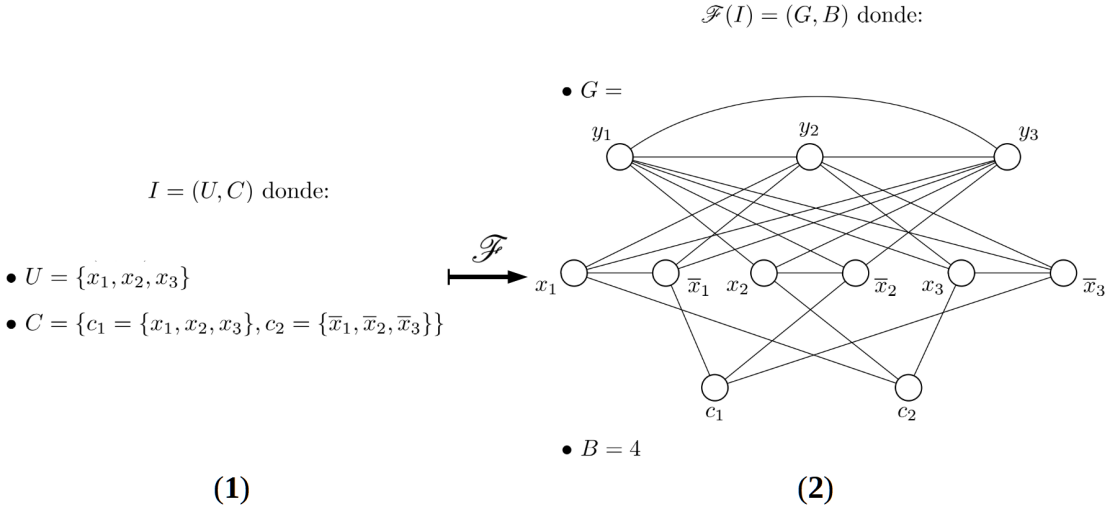

Figura 2.1: Transformación de un ejemplar de *3SAT* en un ejemplar de *CN*: **(1)** Ejemplar de *3SAT*. **(2)** Ejemplar de *CN* correspondiente bajo $\mathscr{F}$.

2.3. Cotas no triviales del Número Cromático

Dado que determinar el número cromático de cualquier gráfica $G = (V, E)$ dada resulta una tarea bastante difícil, es conveniente conocer, al menos, algunas cotas para este número.

Teorema 2.2 Para cualesquiera gráficas $G = (V, E)$ y $H = (W, F)$, si H es subgráfica de G, entonces

$$\chi(H) \leqslant \chi(G).$$

Demostración.

Sean $G = (V, E)$ y $H = (W, F)$ dos gráficas. Supongamos que H es subgráfica de G.

Por demostrar: $\chi(H) \leqslant \chi(G)$.

Supongamos que $\chi(G) = q$, entonces existe una q-coloración válida $\mathscr{C}$ de G. Dado que $\mathscr{C}$ asigna colores distintos a cada par de vértices adyacentes de G, también asigna colores distintos a cada par de vértices adyacentes de H, pues H es subgráfica de G. Lo que implica que H es p-coloreable, con $p \leqslant q$. Lo que implica que $\chi(H) \leqslant p \leqslant q = \chi(G)$.

Por lo tanto, $\chi(H) \leqslant \chi(G)$.

Ergo, para cualesquiera gráficas G y H, si H es subgráfica de G, entonces

$$\chi(H) \leqslant \chi(G).$$

Q. E. D.

Corolario 2.1 Para cada gráfica $G = (V, E)$,

$$\omega(G) \leqslant \chi(G).$$

Teorema 2.3 Para toda gráfica $G = (V, E)$ de orden n,

$$\frac{n}{\alpha(G)} \leqslant \chi(G) \leqslant n - \alpha(G) + 1.$$

Demostración.

Sea $G = (V, E)$ una gráfica de orden n.

Por demostrar: $n/\alpha(G) \leqslant \chi(G) \leqslant n - \alpha(G) + 1$.

Supongamos que $\chi(G) = q$, entonces G tiene una q-coloración válida con clases de color $V(1), V(2), \ldots, V(q)$. Luego, dado que

$$n = |G.V| = \sum_{i=1}^{k} |V(i)| \leqslant k \cdot \alpha(G),$$

se sigue que

$$\frac{n}{\alpha(G)} \leqslant \chi(G).$$

Ahora, sea W un conjunto independiente máximo de G (es decir, un conjunto independiente de cardinalidad $\alpha(G)$) y supongamos que $G.V \setminus W = \{u_1, u_2, \ldots, u_l\}$, donde $l = |G.V \setminus W|$. Si se asigna el color 1 a cada vértice de W y el color $i + 1$ al vértice u_i, para todo $i \in \{1, 2, \ldots, l\}$, se produce una $(l + 1)$-coloración válida de G. Lo que implica que

$$\chi(G) \leqslant l + 1 = |G.V \setminus W| + 1 = n - \alpha(G) + 1.$$

Por lo tanto, $n/\alpha(G) \leqslant \chi(G) \leqslant n - \alpha(G) + 1$.
Ergo, para toda gráfica $G = (V, E)$ de orden n,

$$\frac{n}{\alpha(G)} \leqslant \chi(G) \leqslant n - \alpha(G) + 1.$$

Q. E. D.

Teorema 2.4 Para toda gráfica $G = (V, E)$ de orden n,

$$\chi(G) \leqslant 1 + \Delta(G).$$

Demostración.
Sea $G = (V, E)$ una gráfica de orden n.
Por demostrar: $\chi(G) \leqslant 1 + \Delta(G)$.
Supongamos que $G.V = \{v_1, v_2, \ldots, v_n\}$ y coloreamos los vértices de la siguiente manera:

1. Asignar el color 1 al vértice v_1.
2. Una vez que los vértices $v_1, v_2, \ldots, v_j$ han sido coloreados, asignar al vértice v_{j+1} el color más pequeño que no haya sido utilizado para colorear a ningún vecino de v_{j+1} que pertenezca al conjunto $\{v_1, v_2, \ldots, v_j\}$, para toda $j \in \{1, 2, \ldots, n-1\}$.

Entonces, por construcción, ésta es una coloración válida de G tal que: al vértice v_1 le asigna el color 1 y, para cada $i \in \{2, 3, \ldots, n\}$, asigna al vértice v_i el color 1 o el color $k + 1$, donde k es el entero más grande para el cual todos los colores de la gama $\{1, 2, \ldots, k\}$ son usados para colorear a los vecinos de v_i en el conjunto $S = \{v_1, v_2, \ldots, v_{i-1}\}$. Dado que a los más $deg_G(v_i)$ vecinos de v_i pertenecen a S, el valor más grande que puede tomar k es $deg_G(v_i)$. Lo que implica que el color asignado a v_i es a lo más $1 + deg_G(v_i)$.
Por lo tanto,

$$\chi(G) \leqslant \text{máx}\{\ 1 + deg_G(v_i) \mid i \in \{1, 2, \ldots, n\}\ \} = 1 + \Delta(G).$$

Ergo, para toda gráfica $G = (V, E)$ de orden n,

$$\chi(G) \leqslant 1 + \Delta(G).$$

Q. E. D.

Teorema 2.5 Para cualquier gráfica $G = (V, E)$ de orden $n \geq 2$ se cumple que, G es k-partita si y sólo si G es k-coloreable.

Demostración.
Sea $G = (V, E)$ una gráfica de orden $n \geq 2$.
$\Rightarrow$] Supongamos que G es k-partita.
Por demostrar: G es k-coloreable.
Como G es k-partita, existe una partición de $G.V$ en k conjuntos partitos $V_1, V_2, \ldots, V_k$ tal que para toda $e \in G.E$ exiten $i, j \in \{1, 2, \ldots, k\}$, con $i \neq j$, tales que e une un vértice de V_i y un vértice de V_j. Entonces, para toda $i \in \{1, 2, \ldots, k\}$, $G[V_i]$ no tiene aristas. Lo que implica, $\{V_1, V_2, \ldots, V_k\}$ es una partición de $G.V$ en k conjuntos independientes. Por tanto, existe una partición de $G.V$ en k conjuntos estables.
Por lo tanto, como existe una partición de $G.V$ en k conjuntos estables, G es k-coloreable.

$\Leftarrow$] Supongamos que G es k-coloreable.

Por demostrar: G es k-partita.

Como G es k-coloreable, existe una partición de $G.V$ en k conjuntos estables $V_1, V_2, \ldots, V_k$. Lo que implica, $\{V_1, V_2, \ldots, V_k\}$ es una partición de $G.V$ en k conjuntos independientes. Por tanto, $\{V_1, V_2, \ldots, V_k\}$ es una partición de $G.V$ tal que para toda $e \in G.E$ exiten $i, j \in \{1, 2, \ldots, k\}$, con $i \neq j$, tales que e une un vértice de V_i y un vértice de V_j.

Por lo tanto, G es k-partita.

Ergo, para cualquier gráfica $G = (V, E)$ de orden $n \geq 2$ se cumple que, G es k-partita si y sólo si G es k-coloreable.

Q. E. D.

Los siguientes corolarios se siguen del teorema anterior:

Corolario 2.2 G es k-partita si y sólo si $\chi(G) \leqslant k$.

Corolario 2.3 G es bipartita si y sólo si $\chi(G) = 2$.

Corolario 2.4 Para toda $k > 2$, $G = K_{n_1,n_2,\ldots,n_k}$ si y sólo si $\chi(G) = k$.

Capítulo 3

La Metaheurística *ANTCOL*

I am lost! Where is the line?!
—*A Bug's Life*, Walt Disney, 1998

En este capítulo se expone el primer algoritmo *ACO* creado para atacar el Problema de la Coloración de Gráficas. Se trata de la metaheurística, llamada ***ANTCOL***, presentada por Costa y Hertz en el artículo *"Ants Can Colour Graphs"* [9].

La mayoría de los problemas de optimización combinatoria que han surgido en el mundo real, y que resultan ser NP-Completos, son problemas que pueden formularse como **Problemas de Asignación**, *ATPs*[8]. Cualquier *ATP* consiste de n objetos (o artículos) y m recursos, y tiene por objetivo determinar una asignación de los n objetos a los m recursos que minimice una función objetivo dada y, al mismo tiempo, satisfaga un conjunto de restricciones impuestas. Formalmente, un *ATP* se define como sigue:

1. Ejemplar Genérico: Un conjunto $O = \{o_1, o_2, \ldots, o_n\}$ de objetos y un conjunto $R = \{r_1, r_2, \ldots, r_m\}$ de recursos.
2. Enunciado de Optimización: Minimizar la función $f : \mathscr{S} \to \mathbb{R}$, sujeta a las restricciones:
   (1) $\sum_{j \in J_i} x_{ij} = 1$, para toda $i \in \{1, 2, \ldots, n\}$.
   (2) $R_k(X) \leqslant 0$, para toda $k \in \{1, 2, \ldots, K\}$.
   (3) $x_{ij} = 0$ o 1, para toda $i \in \{1, 2, \ldots, n\}$ y para toda $j \in J_i$.
   (4) Para cada $i \in \{1, 2, \ldots, n\}$, $J_i \subseteq \{1, 2, \ldots, m\}$ es el conjunto de índices de los recursos admisibles para el objeto o_i.

   Donde:
   - f es la **función objetivo** dada. También llamada **función de costo**.

[8] Siglas en inglés de *Assignment Type Problems*. En adelante se utilizará el acrónimo *ATPs* para referirse a este tipo de problemas, usando *ATP* para el singular.

- $\mathscr{S}$ es el **espacio de soluciones candidatas** y es el conjunto de todas las matrices Booleanas $X = (x_{ij})$ de tamaño $n \times m$ tales que:

$$x_{ij} = \begin{cases} 1 & \text{si el objeto } o_i \text{ es asignado al recurso } r_j \\ 0 & \text{en otro caso,} \end{cases}$$

  para toda $i \in \{1, 2, \ldots, n\}$ y para toda $j \in \{1, 2, \ldots, m\}$.

Las restricciones (1), (3) y (4) fuerzan a que el objeto o_i sea asignado a exactamente un recurso r_j. Una solución $X \in \mathscr{S}$ que satisface las restricciones (1)-(4) es llamada **solución factible**. El conjunto $\mathscr{F} = \{ X \in \mathscr{S} \mid X \text{ es solución factible } \}$ es llamado **conjunto de soluciones factibles**. Un solución $X^* \in \mathscr{F}$ tal que $f(X^*) \leqslant f(X)$, para toda $X \in \mathscr{F}$, es llamada **solución óptima global** o simplemente **solución óptima**. Sea $\{\theta_1, \theta_2, \ldots, \theta_{k-1}\} \subseteq \{1, 2, \ldots, n\}$, con $1 \leqslant k \leqslant n$. Una solución $X \in \mathscr{S}$ en la que cada objeto de $\{o_{\theta_1}, o_{\theta_2}, \ldots, o_{\theta_{k-1}}\} \subseteq O$ ya ha sido asignado a un recurso y que satisface todas la restricciones, es llamada **solución parcial** y se denota como $X[\![k-1]\!]$. La función objetivo f, así como las restricciones de tipo (2), son particulares para cada problema de estudio. Varios problemas bien conocidos son ejemplares de los $ATPs$. Por ejemplo, El Problema de la Asignación de Recursos, El Problema de la Asignación Cuadrática, El Problema del Agente Viajero, El Problema de la Coloración de Gráficas y El Problema de Cubierta de Conjuntos. La complejidad de estos problemas está relacionada con la naturaleza de su función objetivo y las restricciones que se le impongan al espacio de soluciones. Con excepción del Problema de Asignación de Recursos, todos los problemas $ATPs$ mencionados anteriormente son NP-Completos.

De aquí en adelante, y como es habitual en Computación, al describir tanto algoritmos como heurísticas todas las matrices serán codificadas como arreglos bidimensionales. En particular, cada elemento de $\mathscr{S}$ es codificado a través de un arreglo bidimensional de tamaño $n \times m$. De manera que, para cada $i \in \{1, 2, \ldots, n\}$ y para toda $j \in \{1, 2, \ldots, m\}$, la componente x_{ij} de $X \in \mathscr{S}$ corresponde al elemento $s[i][j]$ del arreglo bidimensional s que codifica a X. Por otro lado, si X es una solución parcial y s el arreglo bidimensional que la codifica, denotaremos a X por $s[\![k-1]\!]$.

3.1. Metaheurística ANT-Generic

Sea Π un problema de tipo ATP. Una implementación de la metaheurística ACO para atacar Π es un proceso constructivo en donde:

- La colonia de hormigas siempre construye soluciones factibles. Cada solución factible es creada por la colonia de manera probabilista tomando como base el conocimiento obtenido de la construcción de la solución anterior.
- La noción de rastro de feromonas se utiliza para medir el grado de conveniencia de asignar un objeto a un recurso.
- En cada etapa k del proceso constructivo, $1 \leqslant k \leqslant n$, una hormiga elige un objeto no asignado o_i con la máxima probabilidad $\rho_{it}(k, i)$ y un recurso r_j, factible para o_i, con la máxima probabilidad $\rho_{re}(k, i, j)$. Dada una solución parcial $s[\![k-1]\!]$ y dados un objeto no asignado o_i y un recurso admisible r_j para o_i, las probabilidades $\rho_{it}(k, i)$ y $\rho_{re}(k, i, j)$

se definen de la siguiente manera:

$$\rho_{it}(k,i) = \frac{\tau_1(s[\![k-1]\!], o_i)^\alpha \cdot \eta_1(s[\![k-1]\!], o_i)^\beta}{\sum_{o \in O \setminus \{o_{\theta_1}, o_{\theta_2}, \ldots, o_{\theta_{k-1}}\}} \tau_1(s[\![k-1]\!], o)^\alpha \cdot \eta_1(s[\![k-1]\!], o)^\beta}$$

$$\rho_{re}(k,i,j) = \frac{\tau_2(s[\![k-1]\!], o_i, r_j)^\alpha \cdot \eta_2(s[\![k-1]\!], o_i, r_j)^\beta}{\sum_{l \in J_i} \tau_2(s[\![k-1]\!], o_i, r_l)^\alpha \cdot \eta_2(s[\![k-1]\!], o_i, r_l)^\beta}$$

donde $\tau_1(s[\![k-1]\!], o_i)$ y $\tau_2(s[\![k-1]\!], o_i, r_j)$ miden, respectivamente, la intensidad del rastro de feromonas dejado en el objeto o_i y en el recurso admisible r_j para el objeto o_i; $\eta_1(s[\![k-1]\!], o_i)$ y $\eta_2(s[\![k-1]\!], o_i, r_j)$ miden, respectivamente, la deseabilidad del objeto o_i y del recurso admisible r_j para el objeto o_i; α y β son exponentes positivos que controlan la importancia entre el rastro de feromonas y la deseabilidad.

El Pseudocódico 3.1 presenta una aplicación genérica de la metaheurística *ACO*, llamada **ANT-Generic**, para atacar los *ATPs*.

3.1.1. Parámetros

A continuación se dará una breve descripción de cada uno de los parámetros utilizados por **ANT-Generic**.

- n: es el número de objetos.
- m: es el número de recursos.
- $nCycles$: es el número total de ciclos llevados a cabo por la metaheurística.
- $nAnts$: es el número de hormigas en la colonia.
- α y β: exponentes positivos para controlar la importancia entre el rastro de feromonas y la deseabilidad en ρ_{it} y ρ_{re}.
- ε: es el factor de evaporación del rastro de feromonas.
- $pheromoneItem$: guarda la información del rastro de feromonas τ_1. Para cualesquiera $i, k \in \{1, 2, \ldots, n\}$, el valor de $pheromoneItem[k][i]$ es el valor de $\tau_1(s[\![k-1]\!], o_i)$.
- $pheromoneResource$: guarda la información del rastro de feromonas τ_2. Para cualesquiera $i, k \in \{1, 2, \ldots, n\}$ y para cualquier $j \in J_i$, el valor de $pheromoneResource[k][i][j]$ es el valor de $\tau_2(s[\![k-1]\!], o_i, r_j)$.
- $desirabilityItem$: guarda la información de la deseabilidad η_1. Para cada $i, k \in \{1, 2, \ldots, n\}$, el valor de $desirabilityItem[k][i]$ es el valor de $\eta_1(s[\![k-1]\!], o_i)$.
- $desirabilityResource$: guarda la información de la deseabilidad η_2. Para cada $i, k \in \{1, 2, \ldots, n\}$ y para toda $j \in J_i$, el valor de $desirabilityResource[k][i][j]$ es el valor de $\eta_2(s[\![k-1]\!], o_i, r_j)$.
- $bestSolution$: es la mejor solución factible construida por la colonia de hormigas.
- $bestCost$: es el costo de $bestSolution$.
- $assignedItems$: guardará, para cada hormiga, los objetos ya asignados a un recurso.

Pseudocódigo 3.1 Metaheurística **ANT-Generic**

Entrada: Un ejemplar I de un problema Π, de tipo ATP; es decir, un conjunto $O = \{o_1, o_2, \ldots, o_n\}$ de objetos y un conjunto $R = \{r_1, r_2, \ldots, r_m\}$ de recursos.

Salida: Un arreglo bidimensional de tamaño $n \times m$ que contiene una solución factible para Π muy cercana a cualquier solución óptima de Π.

```
1: /* Parámetros */
2: n ← |O|
3: m ← |R|
4: nCycles ← 0
5: nAnts ← 0
6: α ← 0; β ← 0
7: ε ← 0.0
8: Sean pheromoneItem y desirabilityItem dos arreglos bidimensionales de tamaño n × n
9: Sean pheromoneResource y desirabilityResource dos arreglos tridimensionales de tamaño n×n×m
10: Sea bestSolution un arreglo bidimensional de tamaño n × m
11: bestCost ← ∞
12: assignedItems ← ∅
13: candidates ← ∅
14: stoppingCriterion ← false
15: InicializarColonia()
16:
17: /* Rastro de feromonas y deseabilidad */
18: InicializarRastro1()
19: InicializarRastro2()
20: InicializarDeseabilidad1()
21: InicializarDeseabilidad2()
22:
23: /* Proceso constructivo */
24: while ( stoppingCriterion != false ) do
25:     nCycles ← nCycles + 1
26:     for id ← 1 to nAnts do
27:         Sea s_id un arreglo bidimensional de tamaño n × m
28:         for k ← 1 to n do
29:             i ← ElegirObjeto(k)
30:             j ← ElegirRecurso(k, i)
31:             AsignarObjeto(s_id, i, j)
32:             o_θk ← o_i
33:             assignedItems ← assignedItems ∪ {o_θk}
34:         end for
35:         assignedItems ← ∅
36:         candidates ← candidates ∪ {s_id}
37:     end for
38:     bestCandidate ← arg min { CalcularCosto(s) | s ∈ candidates }
39:     if ( CalcularCosto(bestCandidate) < bestCost ) then
40:         bestSolution ← bestCandidate
41:         bestCost ← CalcularCosto(bestSolution)
42:     end if
43:     candidates ← ∅
44:     ActualizarRastro1()
45:     ActualizarRastro2()
46:     ActualizarDeseabilidad1()
47:     ActualizarDeseabilidad2()
48: end while
49: return bestSolution
```

- *candidates*: guardará, en cada iteración, las soluciones factibles construidas por la colonia.
- *stoppingCriterion*: es una condición Booleana que establece la terminación de la ejecución de la metaheurística.

3.1.2. Manipulación de objetos y recursos

La metaheurística **ANT-Generic** cuenta con las siguientes funciones para manipular los objetos y los recursos de Π:

- ElegirObjeto(k): selecciona un índice $i \in \{1, 2, \ldots, n\} \setminus \{\theta_1, \theta_2, \ldots, \theta_{k-1}\}$ tal que el objeto $o_i \in O$ tiene la máxima probabilidad $\rho_{it}(k, i)$ cuando la solución parcial $s[\![k-1]\!]$ ya ha sido construida. Devuelve el índice i. La especificación de esta función se muestra en el Pseudocódigo 3.2.
- ElegirRecursoAdmisible(k, i): selecciona un índice $j \in J_i$ tal que el recurso $r_j \in R$, admisible para el objeto o_i, tiene la máxima probabilidad $\rho_{re}(k, i, j)$ cuando la solución parcial $s[\![k-1]\!]$ ya ha sido construida. Devuelve el índice j. La especificación de esta función se muestra en el Pseudocódigo 3.3.
- AsignarObjeto(s, i, j): asigna el objeto o_i al recurso r_j en la solución s. La especificación de esta función se muestra en el Pseudocódigo 3.4.
- CalcularCosto(s): calcula y devuelve el costo de la solución s.

Pseudocódigo 3.2 Función ElegirObjeto

Entrada: Un entero positivo $k \leqslant n$ tal que la solución parcial $s[\![k-1]\!]$ ya fue construida.
Salida: Un índice $i \in \{1, 2, \ldots, n\} \setminus \{\theta_1, \theta_2, \ldots, \theta_{k-1}\}$ tal que el objeto $o_i \in O$ tiene la máxima probabilidad $\rho_{it}(k, i)$.

```
1: I ← {1, 2, ..., n} \ {θ_1, θ_2, ..., θ_{k-1}}
2: i ← arg max { ρ_it(k, l) | l ∈ I }
3: return i
```

Pseudocódigo 3.3 Función ElegirRecursoAdmisible

Entrada: Un entero positivo $i \leqslant n$ y un entero positivo $k \leqslant n$ tal que la solución parcial $s[\![k-1]\!]$ ya fue construida.
Salida: Un índice $j \in J_i$ tal que el recurso $r_j \in R$ tiene la máxima probabilidad $\rho_{re}(k, i, j)$.

```
1: j ← arg max { ρ_re(k, i, l) | l ∈ J_i }
2: return j
```

Pseudocódigo 3.4 Función AsignarObjeto

Entrada: Un arreglo bidimensional s de tamaño $n \times m$, un entero positivo $i \leqslant n$ y un entero positivo $j \leqslant m$.
Salida: La asignación del objeto o_i al recurso r_j en la solución s.

```
1: s[i][j] ← 1
2: for l ← 1 to m do
3:     if ( l != j ) then
4:         s[i][l] ← 0
5:     end if
6: end for
```

3.1.3. Funciones complementarias

En esta subsección se da una breve descripción de cada una de las funciones complementarias utilizadas por la metaheurística **ANT-Generic**.

- InicializarColonia(): establece el valor más adecuado para $nAnts$, α, β y ε según la definición particular del problema Π.
- InicializarRastro1(): establece el valor inicial del rastro de feromonas τ_1 a 1, pues las hormigas no tienen ningún conocimiento del problema a resolver.
- InicializarRastro2(): asigna al rastro de feromonas τ_2 el valor inicial 1, pues las hormigas no tienen ningún conocimiento del problema a resolver.
- InicializarDeseabilidad1(): establece el valor inicial de la deseabilidad η_1 a 1, pues las hormigas no tienen ningún conocimiento del problema a resolver.
- InicializarDeseabilidad2(): asigna a la deseabilidad η_2 el valor inicial 1, pues las hormigas no tienen ningún conocimiento del problema a resolver.
- ActualizarRastro1(): actualiza el rastro de feromonas τ_1 según la definición particular del problema Π y el factor de evaporación ε.
- ActualizarRastro2(): modifica el rastro de feromonas τ_2 según la definición particular de Π y ε.
- ActualizarDeseabilidad1(): actualiza la deseabilidad η_1 según la definición particular del problema Π y el factor de evaporación ε.
- ActualizarDeseabilidad2(): modifica la deseabilidad η_2 según la definición particular de Π y ε.

3.2. Metaheurística ANTCOL

El Problema de la Coloración de Gráficas es un ejemplar de ATP porque se puede tomar al conjunto de vértices, de la gráfica de entrada, como el conjunto de objetos y a la gama de colores como el conjunto de recursos. Además, dado que siempre es posible colorear cualquier gráfica de orden n con la gama de colores $\{1, 2, \ldots, n\}$, se establece $m = n$. La definición formal del Problema de la Coloración de Gráficas como un ATP es la siguiente:

1. Ejemplar Genérico: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$, y la gama de colores $\{1, 2, \ldots, n\}$.
2. Enunciado de Optimización: Minimizar la función $f : \mathscr{S} \to \mathbb{Z}$ definida por:

$$f(X) = \sum_{j=1}^{n} j \cdot \delta(\sum_{i=1}^{n} x_{ij}),$$

sujeta a las restricciones:

(1) $\sum_{j \in J_i} x_{ij} = 1$, para toda $i \in \{1, 2, \ldots, n\}$.

(2) $R_j(X) = \sum_{v_i v_k \in E} x_{ij} \cdot x_{kj} \leqslant 0$, para toda $j \in \{1, 2, \ldots, n\}$.

(3) $x_{ij} = 0$ o 1, para toda $i \in \{1, 2, \ldots, n\}$ y para cada $j \in J_i$.

(4) Para cada $i \in \{1, 2, \ldots, n\}$, $J_i = \{1, 2, \ldots, n\}$ es la gama de colores disponible para colorear el vértice v_i.

Donde:

- $\mathscr{S}$ es el conjunto de todas las matrices Booleanas cuadradas $X = (x_{ij})$ de tamaño n tales que:
$$x_{ij} = \begin{cases} 1 & \text{si } v_i \text{ tiene asignado el color } j \\ 0 & \text{en otro caso,} \end{cases}$$
para cada $i, j \in \{1, 2, \ldots, n\}$.
- $\delta : \mathbb{Z}^+ \cup \{0\} \to \mathbb{Z}$ definida por:
$$\delta(z) = \begin{cases} 1 & \text{si z} \geqslant 1 \\ 0 & \text{en otro caso.} \end{cases}$$

Cada elemento $X \in \mathscr{S}$ es una posible coloración de G. Las restricciones (1), (3) y (4) garantizan que a cada vértice se le asigna exactamente un color y la restricción (2) evita que las aristas tengan puntos extremos con el mismo color. De manera que, bajo las restriciones impuestas, cualquier elemento de $\mathscr{F}$ es una q-coloración válida de G. La función objetivo f suma los colores utilizados en la coloración X. Por lo que si X es una q-coloración propia de G, $f(X) = \sum_{i=1}^{q} i$; y por tanto, si es una solución óptima, utiliza necesariamente todos los colores de la gama de colores $\{1, 2, \ldots, \chi(G)\}$. Además, nótese que una solución parcial, bajo este problema, corresponde a una coloración parcial válida de G.

La metaheurística **ANTCOL** es un caso particular de **ANT-Generic** para atacar el Problema de la Coloración de Gráficas. En esta implementación, cada solución factible construida por la colonia de hormigas es una partición $s = \{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, para alguna $q \in \mathbb{Z}^+$. Recordemos que G es q-coloreable si y sólo si existe una partición de $G.V$ en q-conjuntos estables. El esquema completo de **ANTCOL** se muestra en el Pseudocódigo 3.5. En las siguientes subsecciones se darán los detalles particulares de **ANTCOL**.

3.2.1. Estrategia general

Sea $G = (V, E)$ la gráfica de entrada, con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$. La metaheurística procede a ejecutar una serie de ciclos. En cada iteración, el rol de cada hormiga consiste en colorear G, con la gama de colores $\{1, 2, \ldots, n\}$, utilizando uno de los dos **MCCG**[9] que tiene disponibles para obtener una q-coloración válida de G. La elección del método la determina el parámetro de entrada auxiliar *method*. El rastro de feromonas y la deseabilidad juegan un papel fundamental en estos métodos constructivos de coloración utilizados por las hormigas como veremos más adelante. A continuación, se elije la coloración que tenga el menor número de colores y se establece como la mejor coloración propia encontrada hasta el momento. Este proceso se repite hasta alcanzar el número límite de iteraciones establecido. La experiencia adquirida por la colonia en cada etapa de la construcción es almacenada en una matriz cuadrada de tamaño n, llamada $trailMatrix$, que es actualizada periódicamente. $trialMatrix$ simula el terreno donde las hormigas buscan su alimento tendiendo y siguiendo rastros de feromonas.

[9] Ver Apéndice C **Métodos Constructivos para la Coloración de Gráficas**

Pseudocódigo 3.5 Metaheurística **ANTCOL**

Entrada: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$, y un entero $method \in \{1, 2\}$.

Salida: Una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, la cual corresponde a una q-coloración propia de G, donde para cada $i \in \{1, 2, \ldots, q\}$, $V_i = V(i)$.

1: **/* Parámetros */**
2: $n \leftarrow |G.V|$
3: $nCycles \leftarrow 0$; $nAnts \leftarrow 0$
4: $\alpha \leftarrow 0$; $\beta \leftarrow 0$
5: $\varepsilon \leftarrow 0.0$
6: Sean $pheromoneVertex$ y $desirabilityVertex$ dos arreglos bidimensionales de tamaño $n \times n$
7: Sean $pheromoneColor$ y $desirabilityColor$ dos arreglos tridimensionales de tamaño $n \times n \times n$
8: $bestColouring \leftarrow \varnothing$
9: $bestNumColours \leftarrow \infty$
10: Sean $trialMatrix$ y $updateMatrix$ dos arreglos bidimensionales de tamaño $n \times n$
11: InicializarColonia()
12:
13: **/* Rastro de feromonas */**
14: InicializarRastros()
15:
16: **/* Deseabilidad */**
17: InicializarDeseabilidades()
18:
19: **/* Coloración construida por las hormigas */**
20: **for** $cycle \leftarrow 1$ **to** $nCycle$ **do**
21: **for** $i \leftarrow 1$ **to** n **do**
22: **for** $j \leftarrow 1$ **to** n **do**
23: $updateMatrix[i][j] \leftarrow 0$
24: **end for**
25: **end for**
26: **for** $id \leftarrow 1$ **to** $nAnts$ **do**
27: $s_{id} \leftarrow \varnothing$
28: $T_1 \leftarrow pheromoneVertex$
29: $T_2 \leftarrow pheromoneColor$
30: $D_1 \leftarrow desirabilityVertex$
31: $D_2 \leftarrow desirabilityColor$
32: **if** ($method == 1$) **then**
33: $\alpha \leftarrow 2$
34: $s_{id} \leftarrow$ **ANT-RLF**$(G, trialMatrix, T_1, T_2, D_1, \alpha, \beta, 2, 1)$
35: **end if**
36: **if** ($method == 2$) **then**
37: $\alpha \leftarrow 1$
38: $s_{id} \leftarrow$ **ANT-DSATUR**$(G, trialMatrix, T_1, T_2, D_1, D_2, \alpha, \beta, 1)$
39: **end if**
40: $q \leftarrow |s_{id}|$
41: **if** ($q < bestNumColours$) **then**
42: $bestColouring \leftarrow s_{id}$
43: $bestNumColours \leftarrow q$
44: **end if**
45: DejarRastro(s_{id}, q)
46: **end for**
47: ActualizarMatrizRastros()
48: **end for**
49: **return** $bestColouring$

3.2.2. Parámetros

A continuación se describen las particularidades de cada uno de los parámetros utilizados por la metaheurística **ANTCOL**, así como los valores recomendados para éstos con base en los experimentos de los autores.

- n: es el número de vértices de G y la cardinalidad de la gama de colores $\{1, 2, \ldots, n\}$.
- $nCycles$: es el número de ciclos en todo el proceso de coloración y su valor se establece en 50.
- $nAnts$: es el número de hormigas en la colonia y se establece en 100.
- α y β: exponentes positivos para controlar la importancia entre el rastro de feromonas y la deseabilidad en ρ_{it} y ρ_{re}. El valor de α se estable en 2, si el valor del parámetro de entrada *method* es 1; en caso contrario, el valor de α se establece en 1. El valor de β se establece en 4.
- ε: es el factor de evaporación del rastro de feromonas y su valor se establece en 0.5.
- $pheromoneVertex$: guarda la información del rastro de feromonas τ_1. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, el valor de $pheromoneVertex[k][i]$ es el valor de $\tau_1(s[\![k-1]\!], v_i)$.
- $pheromoneColor$: guarda la información del rastro de feromonas τ_2. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, y para todo color j, con $j \in J_i$, el valor de $pheromoneColor[k][i][j]$ es el valor de $\tau_2(s[\![k-1]\!], v_i, j)$.
- $desirabilityVertex$: guarda la información de la deseabilidad η_1. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, el valor de $desirabilityVertex[k][i]$ es el valor de $\eta_1(s[\![k-1]\!], v_i)$.
- $desirabilityColor$: guarda la información de la deseabilidad η_2. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, y para todo color j, con $j \in J_i$, el valor de $desirabilityColor[k][i][j]$ es el valor de $\eta_2(s[\![k-1]\!], v_i, j)$.
- $bestColouring$: es la mejor q-coloración propia de G construida por la colonia de hormigas. Se trata de una partición de $G.V$ en q conjuntos estables.
- $bestNumColours$: es el número de colores de $bestColouring$.
- $trialMatrix$: dados cualesquiera dos vértices no adyacentes v_i y v_j, el valor de la entrada $trailMatrix[i][j]$ corresponde al rastro de feromonas existente entre v_i y v_j. Este valor es proporcional a la calidad de la coloración obtenida al colorear con el mismo color a los vértices v_i y v_j. Por lo tanto, $trialMatriz$ ayuda a mantener información acerca de lo conveniente que resulta agregar un vértice no coloreado a la clase de color actual para que siga siendo un conjunto independiente. Inicialmente, todas las entradas de $trialMatrix$ quedan establecidas en 1, para todo par de vértices no adyacentes. El factor de evaporación ε se utiliza para que el rastro de feromonas almacenado en $trialMatrix$ se evapore progresivamente en cada ciclo realizado por la metaheurística. Sean $s_1, s_2, \ldots, s_{nAnts}$ las coloraciones válidas construidas por la colonia al final de una iteración cualquiera y sea S_{ij} el conjunto de todas las coloraciones de $\{s_1, s_2, \ldots, s_{nAnts}\}$ en las cuales v_i y v_j tienen el mismo color. Finalmente, sea q_a el número de colores de s_a, para toda $a \in \{1, 2, \ldots, nAnts\}$. El valor de $trialMatrix[i][j]$ se actualiza de la siguiente manera:

$$trialMatrix[i][j] = \varepsilon \cdot trialMatrix[i][j] + \sum_{s_a \in S_{ij}} \frac{1}{q_a}$$

para todas $i, j \in \{1, 2, \ldots, n\}$ tales que $v_i v_j \notin G.E$.

- $updateMatrix$: es una matriz cuadrada de tamaño n que ayuda a actualizar los valores de $trialMatrix$. Inicialmente, todos las entradas de $updateMatrix$ se establecen en 0.

3.2.3. Funciones complementarias

Las particularidades de las funciones complementarias utilizadas por la metaheurística **ANTCOL** se describen a continuación:

- InicializarColonia(): establece el valor indicado para los parámetros $nCycles$, $nAnts$, β y ε. Ademas, inicializa los valores de las entradas de $trialMatrix$. La especificación de esta función se muestra en el Pseudocódigo 3.6.
- InicializarRastro1(): asigna el valor inicial 1 a las entradas de $pheromoneVertex$. La especificación de esta función se muestra en el Pseudocódigo 3.7.
- InicializarRastro2(): inicializa el valor de las entradas de $pheromoneColor$ a 1. La especificación de esta función se muestra en el Pseudocódigo 3.8.
- InicializarDeseabilidad1(): da el valor inicial 1 a las entradas de $desirabilityVertex$ a 1. La especificación de esta función se muestra en el Pseudocódigo 3.9.
- InicializarDeseabilidad2(): inicializa el valor de las entradas de $desirabilityColor$ a 1. La especificación de esta función se muestra en el Pseudocódigo 3.10.
- InicializarRastros(): invoca a las funciones que inicializan los rastros de feromonas. La especificación de esta función se muestra en el Pseudocódigo 3.11.
- InicializarDeseabilidades(): manda llamar a las funciones que inicializan las deseabilidades. La especificación de esta función se muestra en el Pseudocódigo 3.12.
- DejarRastro(s_{id}, q): regista en $updateMatrix$ el rastro de feromonas dejado por la hormiga actual, entre cada par de vértices no adyacentes, en cada una de las q clases de color de la q-coloración válida s_{id} de G que construyó. La especificación de esta función se muestra en el Pseudocódigo 3.13.
- ActualizarMatrizRastros(): Actualiza los valores de $trialMatriz$ con ayuda de los valores de $updateMatrix$. La especificación de esta función se muestra en el Pseudocódigo 3.14.

Pseudocódigo 3.6 Función InicializarColonia

Entrada: Los valores recomendados para los parámetros $nCycles$, $nAnts$, β y ε.
Salida: El establecimiento de los valores de los parámetros $nCycles$, $nAnts$, β y ε. Además, la inicialización de los valores de la matriz de rastros de feromonas.

```
1: nCycle ← 50
2: nAnts ← 100
3: β ← 4
4: ε ← 0.5
5: for i ← 1 to n do
6:     for j ← 1 to n do
7:         if ( v_i v_j ∉ G.E ) then
8:             trialMatrix[i][j] ← 1
9:         end if
10:    end for
11: end for
```

Pseudocódigo 3.7 Función InicializarRastro1

Entrada: El valor inicial para el rastro de feromonas τ_1.
Salida: La inicialización del valor de las entradas de $pheromoneVertex$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         pheromoneVertex[i][j] ← 1
4:     end for
5: end for
```

Pseudocódigo 3.8 Función InicializarRastro2

Entrada: El valor inicial para el rastro de feromonas τ_2.
Salida: La inicialización del valor de las entradas de $pheromoneColor$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         for k ← 1 to n do
4:             pheromoneColor[i][j][k] ← 1
5:         end for
6:     end for
7: end for
```

Pseudocódigo 3.9 Función InicializarDeseabilidad1

Entrada: El valor inicial para la deseabilidad η_1.
Salida: La inicialización del valor de las entradas de $desirabilityVertex$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         desirabilityVertex[i][j] ← 1
4:     end for
5: end for
```

Pseudocódigo 3.10 Función InicializarDeseabilidad2

Entrada: El valor inicial para la deseabilidad η_2.
Salida: La inicialización del valor de las entradas de $desirabilityColor$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         for k ← 1 to n do
4:             desirabilityColor[i][j][k] ← 1
5:         end for
6:     end for
7: end for
```

Pseudocódigo 3.11 Función InicializarRastros

Entrada: Las funciones que inicializan los rastros de feromonas.
Salida: La inicialización de los rastros de feromonas

```
1: InicializarRastro1()
2: InicializarRastro2()
```

Pseudocódigo 3.12 Función InicializarDeseabilidades

Entrada: Las funciones que inicializan las deseabilidades.
Salida: La inicialización de las deseabilidades.

```
1: InicializarDeseabilidad1()
2: InicializarDeseabilidad2()
```

Pseudocódigo 3.13 Función DejarRastro

Entrada: La q-coloración válida s_{id} de G construida por la hormiga actual id.
Salida: El rastro de feromonas dejado por la hormiga actual entre cada par de vértices no adyacentes en cada una de las q clases de color de esta q-coloración válida de G que construyó.

```
1: for i ← 1 to q do
2:     for each u ∈ s_id.V_i do
3:         Sea j ∈ {1, 2, ..., n} el índice de u en G.V
4:         for each v ∈ s_id.V_i do
5:             Sea k ∈ {1, 2, ..., n} el índice de v en G.V
6:             updateMatrix[j][k] ← updateMatrix[j][k] + 1/q
7:         end for
8:     end for
9: end for
```

Pseudocódigo 3.14 Función ActualizarMatrizRastros

Entrada: La matriz de rastros de feromonas.
Salida: La actualización de la matriz de rastros de feromonas.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         if ( v_i v_j ∉ G.E ) then
4:             trialMatrix[i][j] ← ε * trialMatrix[i][j] + updateMatrix[i][j]
5:         end if
6:     end for
7: end for
```

3.2.4. Métodos constructivos para ANTCOL

La columna vertebral de la metaheurística **ANTCOL** está conformada por dos MCCG: **ANT-RLF** y **ANT-DSATUR**. Evidentementeme, la estrategia utilizada por el algoritmo **ANT-RLF** es *RLF* y la utilizada por el algoritmo **ANT-DSATUR** es *DSATUR*. La razón por la cual **ANTCOL** utiliza estas dos estrategias se debe a que *RLF* y *DSATUR* tienen un desempeño significativamente mejor que las estrategias *RANDOM*, *LF* y *SL*.

El algoritmo **ANT-RLF** tiene dos características particulares:

1. Su parámetro de entrada $firstVertex$ define dos maneras de elegir el primer vértice que será agregado al conjunto estable actual.
2. Su parámetro de entrada *desirability* define tres formas de actualizar el valor de la deseabilidad $\eta_1(s[\![k-1]\!], v_i)$ para terminar de construir el conjunto estable actual.

La especificación de **ANT-RLF** se muestra en el Pseudocódigo 3.15. Para evitar la pérdida de legibilidad, debido a las múltiples opciones del algoritmo, se utilizaron las funciones auxiliares presentadas en los Pseudocódigos 3.16-3.22.

Pseudocódigo 3.15 Algoritmo **ANT-RLF**

Entrada: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$; la matriz de rastros $trialMatrix$ asociada a G; un arreglo bidimensional $pheromoneVertex$, de tamaño $n \times n$, con la información del rastro de feromonas τ_1; un arreglo tridimensional $pheromoneColor$, de tamaño $n \times n \times n$, con la información del rastro de feromonas τ_2; un arreglo bidimensional $desirabilityVertex$, de tamaño $n \times n$, con la información de la deseabilidad η_1; los exponentes positivos α y β, que controlan la importancia entre el rastro de feromonas y la deseabilidad en ρ_{it}; un entero $firstVertex \in \{1, 2\}$; y finalmente, un entero $desirability \in \{1, 2, 3\}$.

Salida: Una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, la cual corresponde a una q-coloración propia de G, donde para cada $i \in \{1, 2, \ldots, q\}$, $V_i = V(i)$.

1: **/* Parámetros */**
2: $partition \leftarrow \varnothing$; $uncolouredVertices \leftarrow G.V$; $colouredVertices \leftarrow \varnothing$
3: $numColouredVertices \leftarrow 0$; $q \leftarrow 0$
4: $currentVertex \leftarrow$ **NULL**
5:
6: **/* Proceso constructivo de la coloración */**
7: **while** ($uncolouredVertices \; != \varnothing$) **do**
8: **if** ($firstVertex == 1$) **then**
9: $currentVertex \leftarrow$ EjecutarSeleccion1()
10: **end if**
11: **if** ($firstVertex == 2$) **then**
12: $currentVertex \leftarrow$ EjecutarSeleccion2()
13: **end if**
14: Sea $i \in \{1, 2, \ldots, n\}$ el índice de $currentVertex$ en $G.V$
15: $q \leftarrow q + 1$
16: $V_q \leftarrow \{v_i\}$
17: Insertar V_q en $partition$, como elemento
18: $colouredVertices \leftarrow colouredVertices \cup \{v_i\}$
19: $numColouredVertices \leftarrow numColouredVertices + 1$
20: $W \leftarrow \{w \in uncolouredVertices \mid w \notin Adj_{G[uncolouredVertices]}(v_i)\} \setminus \{v_i\}$
21: $B \leftarrow \{w \in uncolouredVertices \mid w \in Adj_{G[uncolouredVertices]}(v_i)\}$
22: **while** ($W \; != \varnothing$) **do**
23: $numColouredVertices \leftarrow numColouredVertices + 1$
24: $I \leftarrow \{ \; i \in \{1, 2, \ldots, n\} \mid v_i \notin colouredVertices\}$
25: ActualizarRastros(I)
26: **if** ($desirability == 1$) **then**
27: ActualizarDeseabilidad1-Opcion1(I)
28: **end if**
29: **if** ($desirability == 2$) **then**
30: ActualizarDeseabilidad1-Opcion2(I)
31: **end if**
32: **if** ($desirability == 3$) **then**
33: ActualizarDeseabilidad1-Opcion3(I)
34: **end if**
35: $j \leftarrow$ ElegirVertice(I, W)
36: $partition.V_q \leftarrow partition.V_q \cup \{v_j\}$
37: $colouredVertices \leftarrow colouredVertices \cup \{v_j\}$
38: $B \leftarrow B \cup Adj_{G[W]}(v_j)$
39: $W \leftarrow W \setminus (Adj_{G[W]}(v_j) \cup \{v_j\})$
40: **end while**
41: $uncolouredVertices \leftarrow uncolouredVertices \setminus partition.V_q$
42: **end while**
43: **return** $partition$

Pseudocódigo 3.16 Función EjecutarSeleccion1

Entrada: El conjunto de vértices no coloreados $uncolouredVertices$.
Salida: Un vértice $v \in uncoloredVertices$.
1: $v \leftarrow$ **arg max** $\{ deg_{G[uncolouredVertices]}(u) \mid u \in uncolouredVertices \}$
2: **return** v

Pseudocódigo 3.17 Función EjecutarSeleccion2

Entrada: El conjunto de vértices no coloreados $uncolouredVertices$.
Salida: Un vértice $v \in uncoloredVertices$.
1: $\rho \leftarrow 1/|uncolouredVertices|$
2: Seleccionar de manera aleatoria, con probabilidad $1/\rho$, un vértice $v \in uncolouredVertices$
3: **return** v

Pseudocódigo 3.18 Función ActualizarRastros

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor de los rastros de feromonas τ_1 y τ_2 para los vértices correspondientes a I.
1: **for each** $i \in I$ **do**
2: **for** $j \leftarrow 1$ **to** q **do**
3: $trialSum \leftarrow 0$
4: **for each** $v \in partition.V_j$ **do**
5: Sea $k \in \{1, 2, \ldots, n\}$ el índice de v en $G.V$
6: $trialSum \leftarrow trialSum + trialMatriz[k][i]$
7: **end for**
8: $pheromoneColor[numColouredVertices][i][j] \leftarrow trialSum/|partition.V_j|$
9: **end for**
10: **end for**
11: **for each** $i \in I$ **do**
12: $pheromoneVertex[numColouredVertices][i] \leftarrow pheromoneColor[numColouredVertices][i][q]$
13: **end for**

Pseudocódigo 3.19 Función ActualizarDeseabilidad1-Opcion1

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor de la deseabilidad η_1 para los vértices correspondientes a I.
1: **for each** $i \in I$ **do**
2: $desirabilityVertex[numColouredVertices][i] \leftarrow deg_{G[B]}(v_i)$
3: **end for**

Pseudocódigo 3.20 Función ActualizarDeseabilidad1-Opcion2

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor de la deseabilidad η_1 para los vértices correspondientes a I.
1: **for each** $i \in I$ **do**
2: $desirabilityVertex[numColouredVertices][i] \leftarrow |W| - deg_{G[W]}(v_i)$
3: **end for**

Pseudocódigo 3.21 Función ActualizarDeseabilidad1-Opcion3

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor de la deseabilidad η_1 para los vértices correspondientes a I.

```
1: for each i ∈ I do
2:     desirabilityVertex[numColouredVertices][i] ← deg_{G[W∪B]}(v_i)
3: end for
```

Pseudocódigo 3.22 Función ElegirVertice

Entrada: El conjunto de índices I de los vértices no coloreados y el conjunto W de vértices no coloreados que pueden ser agregados al conjunto estable actual.
Salida: Un índice $j \in I$ tal que el vértice $v_j \in W$ tiene la máxima probabilidad $\rho_{it}(numColouredVertices, j)$.

```
1: J ← { i ∈ I | v_i ∈ W}
2: j ← arg max { ρ_it(numColouredVertices, l) | l ∈ J }
3: return j
```

Por su parte, el algoritmo **ANT-DSATUR** tiene las siguientes particularidades:

1. El valor de la deseabilidad $\eta_1(s[\![k-1]\!], v_i)$ de un vértice no coloreado v_i, corresponde a su grado de saturación.
2. Su parámetro de entrada $nextVertex$ define dos formas de elegir el siguiente vértice a ser coloreado, así como el color que se le asignará. En la segunda forma, cuando el valor de $nextVertex$ vale 2, el vértice v_i a ser coloreado no recibe necesariamente el color $c_{min}(v_i)$; donde $c_{min}(v_i)$ es el color más pequeño, de la gama de colores $\{1, 2, \ldots, n\}$, que se le puede asignar a v_i para mantener la factibilidad de la coloración parcial construida hasta el momento. El color $c \in \{1, 2, \ldots, n\}$ que se le asignará a v_i será elegido tomando en cuenta el rastro $\tau_2(s[\![k-1]\!], v_i, c)$ tal que $c_{min}(v_i) \leqslant c \leqslant q$, donde q es el número de colores ya utilizados.

La especificación de **ANT-DSATUR** se muestra en el Pseudocódigo 3.23. Nuevamente, para evitar la pérdida de legibilidad, debido a las múltiples opciones del algoritmo, se utilizaron las funciones auxiliares presentadas en los Pseudocódigos 3.24-3.27.

Dada una coloración parcial válida $s[\![k-1]\!] = \{V_1, V_2, \ldots, V_q\}$, un vértice no coloreado v_i y un color j con el que es factible colorear v_i, el rastro de feromonas $\tau_2(s[\![k-1]\!], v_i, j)$ es actualizado, tanto por **ANT-RLF** como por **ANT-DSATUR**, de la siguiente manera:

$$\tau_2(s[\![k-1]\!], v_i, j) = \begin{cases} 1 & \text{si } V_j = \varnothing \\ \dfrac{\sum_{v_l \in V_j} trialMatrix[l][i]}{|V_j|} & \text{en otro caso.} \end{cases}$$

Este rastro indica la calidad de la coloración parcial válida obtenida si se colorea el vértice v_i con el color j.

Por otro lado, los valores recomendados para los parámetros α, β, $firstVertex$ y $desirability$ con los que **ANTCOL** debe invocar a **ANT-RLF** son: 2, 4, 2 y 1 respectivamente. Mientras que los valores recomendados para los parámetros α, β y $nextVertex$ con los que **ANTCOL** debe invocar a **ANT-DSATUR** son: 1, 4 y 1 respectivamente. Estos valores recomendados para llamar a los algoritmos **ANT-RLF** y **ANT-DSATUR** son resultado de los experimentos realizados por los autores y corresponden a los que produjeron los mejores resultados.

Pseudocódigo 3.23 Algoritmo **ANT-DSATUR**

Entrada: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$; la matriz de rastros $trialMatrix$ asociada a G; un arreglo bidimensional $pheromoneVertex$, de tamaño $n \times n$, con la información del rastro de feromonas τ_1; un arreglo tridimensional $pheromoneColor$, de tamaño $n \times n \times n$, con la información del rastro de feromonas τ_2; un arreglo bidimensional $desirabilityVertex$, de tamaño $n \times n$, con la información de la deseabilidad η_1; un arreglo tridimensional $desirabilityColor$, de tamaño $n \times n \times n$, con la información de la deseabilidad η_2; los exponentes positivos α y β, que controlan la importancia entre el rastro de feromonas y la deseabilidad en ρ_{it} y ρ_{re}; y finalmente, un entero $nextVertex \in \{1, 2\}$.

Salida: Una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, la cual corresponde a una q-coloración propia de G, donde para cada $i \in \{1, 2, \ldots, q\}$, $V_i = V(i)$.

1: **/* Parámetros */**
2: $partition \leftarrow \varnothing$; $partial \leftarrow \varnothing$; $uncolouredVertices \leftarrow G.V$; $colouredVertices \leftarrow \varnothing$
3: $numColouredVertices \leftarrow 1$; $q \leftarrow 1$
4: $index \leftarrow 0$; $currentColor \leftarrow 0$
5:
6: **/* Proceso constructivo de la coloración */**
7: **for** $i \leftarrow 1$ **to** n **do**
8: $c_{min}(v_i) \leftarrow 1$; $dsat_{colouredVertices}(v_i) \leftarrow 0$; $V_i \leftarrow \varnothing$
9: Insertar V_i en $partial$, como elemento
10: **end for**
11: $v \leftarrow$ **arg max** $\{ deg_{G[uncolouredVertices]}(u) \mid u \in uncolouredVertices \}$
12: $partial.V_1 \leftarrow partial.V_1 \cup \{v\}$
13: $colouredVertices \leftarrow colouredVertices \cup \{v\}$
14: **for** $j \leftarrow 2$ **to** n **do**
15: $numColouredVertices \leftarrow numColouredVertices + 1$
16: **for each** $u \in Adj_{G[uncolouredVertices]}(v)$ **do**
17: Actualizar el valor de $c_{min}(u)$ y $dsat_{colouredVertices}(u)$
18: **end for**
19: $uncolouredVertices \leftarrow uncolouredVertices \setminus \{v\}$
20: $I \leftarrow \{ i \in \{1, 2, \ldots, n\} \mid v_i \in uncolouredVertices\}$
21: ActualizarDeseabilidad1(I)
22: **if** ($nextVertex == 1$) **then**
23: ActualizarRastro2(I)
24: ActualizarRastro1-Opcion1(I)
25: $index \leftarrow$ **arg max** $\{ \rho_{it}(numColouredVertices, l) \mid l \in I \}$
26: $currentColor \leftarrow c_{min}(v_{index})$
27: **end if**
28: **if** ($nextVertex == 2$) **then**
29: ActualizarRastro1-Opcion2(I)
30: $index \leftarrow$ **arg max** $\{ \rho_{it}(numColouredVertices, l) \mid l \in I \}$
31: $C \leftarrow \{c_{min}(v_{index}), \ldots,$ **max**$(c_{min}(v_{index}), q) \}$
32: $currentColor \leftarrow$ **arg max** $\{ \rho_{re}(numColouredVertices, index, c) \mid c \in C \}$
33: **end if**
34: $partial.V_{currentColor} \leftarrow partial.V_{currentColor} \cup \{v_{index}\}$
35: $colouredVertices \leftarrow colouredVertices \cup \{v_{index}\}$
36: $v \leftarrow v_{index}$
37: **if** ($currentColor == q + 1$) **then**
38: $q \leftarrow q + 1$
39: **end if**
40: **end for**
41: **for** $i \leftarrow 1$ **to** q **do**
42: Insertar $partial.V_i$ en $partition$, como elemento
43: **end for**
44: **return** $partition$

Pseudocódigo 3.24 Función ActualizarRastro1-Opcion1

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor del rastro de feromonas τ_1 para los vértices correspondientes a I.

for each $i \in I$ **do**
  $t \leftarrow pheromoneColor[numColouredVertices][i][c_{min}(v_i)]$
  $pheromoneVertex[numColouredVertices][i] \leftarrow t$
end for

Pseudocódigo 3.25 Función ActualizarRastro1-Opcion2

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor del rastro de feromonas τ_1 para los vértices correspondientes a I.

for each $i \in I$ **do**
  $pheromoneVertex[numColouredVertices][i] \leftarrow 1$
end for

Pseudocódigo 3.26 Función ActualizarRastro2

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor del rastro de feromonas τ_2 para los vértices correspondientes a I.

for each $i \in I$ **do**
  for $j \leftarrow 1$ **to** q **do**
    $trialSum \leftarrow 0$
    for each $v \in partial.V_j$ **do**
      Sea $k \in \{1, 2, \ldots, n\}$ el índice de v en $G.V$
      $trialSum \leftarrow trialSum + trialMatriz[k][i]$
    end for
    $pheromoneColor[numColouredVertices][i][j] \leftarrow trialSum/|partial.V_j|$
  end for
end for

Pseudocódigo 3.27 Función ActualizarDeseabilidad1

Entrada: El conjunto de índices I de los vértices no coloreados.
Salida: Actualización del valor de la deseabilidad η_1 para los vértices correspondientes a I.

for each $i \in I$ **do**
  $desirabilityVertex[numColouredVertices][i] \leftarrow dsat_{colouredVertices}(v_i)$
end for

Nótese que **ANTCOL** no conoce, en absoluto, el funcionamiento de sus métodos constructivos (MCCG). Además, es de suma importancia mencionar que uno de los resultados más interesantes obtenidos de los experimentos realizados por el equipo de trabajo de Costa y Hertz (y que tendrá gran relevancia en el siguiente capítulo), es el hecho de que la metaheurística **ANTCOL** bajo el MCCG **ANT-RLF** es mucho más eficiente que bajo el MCCG **ANT-DSATUR**. Por lo que para los fines de este trabajo, los experimentos que se realizarán con la metaheurística **ANTCOL** serán utilizando exclusivamente el algoritmo **ANT-RLF** con los valores recomendados.

Capítulo 4

La Metaheurística *D-ANTCOL*

Ants have the most complicated social organization on earth next to humans.
—*The Ants*, E. O. Wilson, 1991

En este capítulo se expone una revisión de la metaheurística ***ANTCOL***, llamada ***D-ANTCOL***, presentada por Dowsland y Thompson en el artículo *"An improved ant colony optimisation heuristic for graph colouring"* [10].

La metaheurística **D-ANTCOL** es una mejora de **ANTCOL** que deja a un lado la utilización del algoritmo **ANT-DSATUR** y se concentra en las características particulares del MCCG **ANT-RLF** para atacar el Problema da la Coloración de Gráficas. ¿Por qué es una mejora de **ANTCOL**? Como se mencionó en el capítulo anterior, los experimentos de Costa y Hertz mostraron que **ANTCOL** basada exclusivamente en el algoritmo **ANT-RLF** es superior a la basada en el algoritmo **ANT-DSATUR**.

Lo que hace **D-ANTCOL** es integrar la definición de **ANT-RLF** a la de **ANTCOL**, con ligeras modificaciones, pero importantes, eliminando por completo la utilización del algoritmo **ANT-DSATUR**.

4.1. Metaheurística D-ANTCOL

Recordemos que el Problema de la Coloración de Gráficas es un problema de tipo *ATP* definido de la siguiente manera:

1. Ejemplar Genérico: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$, y la gama de colores $\{1, 2, \ldots, n\}$.
2. Enunciado de Optimización: Minimizar la función $f : \mathscr{S} \to \mathbb{Z}$ definida por:

$$f(X) = \sum_{j=1}^{n} j \cdot \delta(\sum_{i=1}^{n} x_{ij}),$$

sujeta a las restricciones:

(1) $\sum_{j \in J_i} x_{ij} = 1$, para toda $i \in \{1, 2, \ldots, n\}$.

(2) $R_j(X) = \sum_{v_i v_k \in E} x_{ij} \cdot x_{kj} \leqslant 0$, para toda $j \in \{1, 2, \ldots, n\}$.

(3) $x_{ij} = 0$ o 1, para toda $i \in \{1, 2, \ldots, n\}$ y para cada $j \in J_i$.

(4) Para cada $i \in \{1, 2, \ldots, n\}$, $J_i = \{1, 2, \ldots, n\}$ es la gama de colores disponible para colorear el vértice v_i.

Donde:

- $\mathscr{S}$ es el conjunto de todas las matrices lógicas cuadradas $X = (x_{ij})$ de tamaño n tales que:
$$x_{ij} = \begin{cases} 1 & \text{si } v_i \text{ tiene asignado el color } j \\ 0 & \text{en otro caso,} \end{cases}$$
para cada $i, j \in \{1, 2, \ldots, n\}$.
- $\delta : \mathbb{Z}^+ \cup \{0\} \to \mathbb{Z}$ definida por:
$$\delta(z) = \begin{cases} 1 & \text{si z} \geqslant 1 \\ 0 & \text{en otro caso.} \end{cases}$$

Por otro lado, la metaheurística **ANTCOL**, bajo el uso de los métodos constructivos **ANT-RLF** y **ANT-DSATUR**, define un proceso constructivo que genera una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, con q probablemente muy cerca de $\chi(G)$. Cada hormiga de la colonia ejecuta este proceso constructivo, donde en cada etapa k, con $1 \leqslant k \leqslant n$, la hormiga actual elige un vértice no coloreado v_i que tenga la máxima probabilidad $\rho_{it}(k, i)$ y lo colorea con un color j, factible para v_i, que tenga la máxima probabilidad $\rho_{re}(k, i, j)$. En este caso, dada una coloración parcial válida de G, $s[\![k-1]\!]$, en la que los vértices $v_{\theta_1}, v_{\theta_2}, \ldots, v_{\theta_{k-1}}$ ya han sido coloreados; y dado un vértice no coloreado v_i y un color j con el que es factible colorear v_i, las probabilidades $\rho_{it}(k, i)$ y $\rho_{re}(k, i, j)$ se definen de la siguiente manera:

$$\rho_{it}(k, i) = \frac{\tau_1(s[\![k-1]\!], v_i)^\alpha \cdot \eta_1(s[\![k-1]\!], v_i)^\beta}{\sum_{v \in G.V \setminus \{v_{\theta_1}, v_{\theta_2}, \ldots, v_{\theta_{k-1}}\}} \tau_1(s[\![k-1]\!], v)^\alpha \cdot \eta_1(s[\![k-1]\!], v)^\beta},$$

$$\rho_{re}(k, i, j) = \frac{\tau_2(s[\![k-1]\!], v_i, j)^\alpha \cdot \eta_2(s[\![k-1]\!], v_i, j)^\beta}{\sum_{l \in J_i} \tau_2(s[\![k-1]\!], v_i, l)^\alpha \cdot \eta_2(s[\![k-1]\!], v_i, l)^\beta},$$

donde $\tau_1(s[\![k-1]\!], v_i)$ y $\tau_2(s[\![k-1]\!], v_i, j)$ miden, respectivamente, la intensidad del rastro de feromonas dejado en el vértice v_i y en el color j, con el que es factible colorear al vértice v_i; $\eta_1(s[\![k-1]\!], v_i)$ y $\eta_2(s[\![k-1]\!], v_i, j)$ miden, respectivamente, la deseabilidad del vértice v_i y del color j, con el que es factible colorear al vértice v_i; y finalmente, α y β son exponentes positivos que controlan la importancia entre el rastro de feromonas y la deseabilidad. En particular, cuando la metaheurística **ANTCOL** utiliza el algoritmo **ANT-RLF**, las clases de color son construidas de manera secuencial como conjuntos independientes maximales. En todo el proceso de coloración llevado a cabo por **ANT-RLF**, las únicas decisiones que debe tomar son:

1. Cuál de los vértices no coloreados es el primero que debe ser insertado en la clase de color actual.
2. Cuál de los vértices que siguen sin ser coloreados debe ser insertado en la clase de color actual.

La primera decisión es tomada por **ANT-RLF** a través de su parámetro de entrada $firstVertex$, donde:

- si el valor de $firstVertex$ es 1, el primer vértice en ser insertado en la clase de color actual, es un vértice $v \in uncolouredVertices$ que tenga el máximo grado en la gráfica $G[uncolouredVertices]$; y
- si el valor de $firstVertex$ es 2, el primer vértice en ser insertado en la clase de color actual, es un vértice $v \in uncolouredVertices$ elegido de manera aleatoria con probabilidad uniforme.

D-ANTCOL optará por integrar la segunda forma de elegir el primer vértice que será agregado a la clase de color actual.

En el caso de la segunda decisión, ésta depende exclusivamente de la probabilidad ρ_{it}. Al mismo tiempo, ρ_{it} depende exclusivamente de los valores de τ_1 y de η_1 para los vértices en $G.V \setminus \{v_{\theta_1}, v_{\theta_2}, \ldots, v_{\theta_{k-1}}\}$, el conjunto de vértices no coloreados; pues los valores de α y β son fijados previamente. El valor de $\tau_1(s[\![k-1]\!], v_i)$ es actualizado por **ANT-RLF** de la siguiente manera:

$$\tau_1(s[\![k-1]\!], v_i) = \tau_2(s[\![k-1]\!], v_i, q),$$

donde q es el número de colores ya utilizados en la coloración parcial válida $s[\![k-1]\!]$ y el valor de $\tau_2(s[\![k-1]\!], v_i, j)$ está dado por:

$$\tau_2(s[\![k-1]\!], v_i, j) = \begin{cases} 1 & \text{si } V_j = \varnothing, \\ \dfrac{\sum\limits_{v_r \in V_j} trialMatrix[l][i]}{|V_j|} & \text{en otro caso,} \end{cases}$$

D-ANTCOL mantendrá esta manera de actualizar los valores de los rastros de feromonas $\tau_1(s[\![k-1]\!], v_i)$ y $\tau_2(s[\![k-1]\!], v_i, j)$. Por otro lado, a través de su parámetro de entrada *desirability*, el algoritmo **ANT-RLF** define tres formas de actualizar el valor de la deseabilidad $\eta_1(s[\![k-1]\!], v_i)$. Éstas son:

- $\eta_1(s[\![k-1]\!], v_i) = deg_{G[B]}(v_i)$, si el valor de *desirability* es igual a 1;
- $\eta_1(s[\![k-1]\!], v_i) = |W| \setminus deg_{G[W]}(v_i)$, si el valor de *desirability* es igual a 2; y
- $\eta_1(s[\![k-1]\!], v_i) = deg_{G[B \cup W]}(v_i)$, si el valor de *desirability* es igual a 3,

donde W es el conjunto de todos los vértices no coloreados que pueden ser insertados en la clase de color actual y B es el conjunto de todos los vértices no coloreados que no pueden ser agregados a la clase de color actual. **D-ANTCOL** utilizará estas tres formas de actualizar $\eta_1(s[\![k-1]\!], v_i)$, pero modificando la manera de seleccionarlas: en lugar de que la elección sea fija, será aleatoria. Finalmente, **D-ANTCOL** calculará el valor de la probabilidad ρ_{it} considerando el conjunto de vértices W en lugar del conjunto de vértices $G.V \setminus \{v_{\theta_1}, v_{\theta_2}, \ldots, v_{\theta_{k-1}}\}$, es decir,

$$\rho_{it}(k,i) = \frac{\tau_1(s[\![k-1]\!], v_i)^\alpha \cdot \eta_1(s[\![k-1]\!], v_i)^\beta}{\sum\limits_{v \in W} \tau_1(s[\![k-1]\!], v)^\alpha \cdot \eta_1(s[\![k-1]\!], v)^\beta}.$$

El esquema completo de la metaheurística **D-ANTCOL** se muestra en el Pseudocódigo 4.1. En las siguientes subsecciones se darán los detalles particulares de **D-ANTCOL**.

Pseudocódigo 4.1 Metaheurística **D-ANTCOL**

Entrada: Una gráfica $G = (V, E)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$.
Salida: Una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables, la cual corresponde a una q-coloración propia de G, donde para cada $i \in \{1, 2, \ldots, q\}$, $V_i = V(i)$.

```
1: /* Parámetros */
2: n ← |G.V|
3: nCycles ← 0
4: nAnts ← 0
5: α ← 0
6: β ← 0
7: ε ← 0.0
8: Sean pheromoneVertex y desirabilityVertex dos arreglos bidimensionales de tamaño n × n
9: Sea pheromoneColor un arreglo tridimensional de tamaño n × n × n
10: bestColouring ← ∅
11: bestNumColours ← ∞
12: Sean trialMatrix y updateMatrix dos arreglos bidimensionales de tamaño n × n
13: uncolouredVertices ← G.V
14: colouredVertices ← ∅
15: numColouredVertices ← 0
16: InicializarColonia()
17:
18: /* Rastros de feromonas */
19: InicializarRastro1()
20: InicializarRastro2()
21:
22: /* Deseabilidad */
23: InicializarDeseabilidad1()
24:
25: /* Coloración construida por las hormigas */
26: for cycle ← 1 to nCycle do
27:     for i ← 1 to n do
28:         for j ← 1 to n do
29:             updateMatrix[i][j] ← 0
30:         end for
31:     end for
32:     for id ← 1 to nAnts do
33:         s_id ← ∅
34:         q ← 0
35:         T_1 ← pheromoneVertex
36:         T_2 ← pheromoneColor
37:         D_1 ← desirabilityVertex
38:         Colorear(s_id, q)
39:         if ( q < bestNumColours ) then
40:             bestColouring ← s_id
41:             bestNumColours ← q
42:         end if
43:         DejarRastro(s_id, q)
44:         uncolouredVertices ← G.V
45:         colouredVertices ← ∅
46:         numColouredVertices ← 0
47:     end for
48:     ActualizarMatrizRastros()
49: end for
50: return bestColouring
```

4.1.1. Estrategia general

Sea $G = (V, E)$ la gráfica de entrada, con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$. La estrategia de **D-ANTCOL** es, esencialmente, la misma que la de **ANTCOL**: procede a ejecutar una serie de ciclos. En cada iteración, el rol de cada hormiga de la colonia consiste en crear una q-coloración propia de G. En seguida, se elije la coloración que tenga el menor número de colores y se establece como la mejor coloración válida encontrada hasta el momento. Este proceso se repite hasta alcanzar el número máximo de iteraciones establecido. Análogamente a **ANTCOL**, la experiencia adquirida por la colonia en cada etapa de la construcción es almacenada en una matriz cuadrada de tamaño n, llamada $trailMatrix$, que es actualizada periódicamente. **D-ANTCOL** actualiza $trialMatrix$ de la misma manera que **ANTCOL**. Recordemos que $trialMatrix$ simula el terreno donde las hormigas buscan su alimento tendiendo y siguiendo rastros de feromonas.

4.1.2. Parámetros

A continuación se describen los detalles de cada uno de los parámetros utilizados por la metaheurística **D-ANTCOL**, así como los valores recomendados para éstos en base a los experimentos reportados por los autores.

- n: es el número de vértices de G y la cardinalidad de la gama de colores $\{1, 2, \ldots, n\}$.
- $nCycles$: es el número total de ciclos que llevará a cabo la metaheurística y su valor se establece en 50.
- $nAnts$: es el número total de hormigas en la colonia y se establece en 100.
- α y β: exponentes positivos para controlar la importancia entre el rastro de feromonas y la deseabilidad en ρ_{it}. El valor de α se establece en 2 y el valor de β se establece en 4.
- ε: es el factor de evaporación del rastro de feromonas y su valor se establece en 0.5.
- $pheromoneVertex$: guarda la información del rastro de feromonas τ_1. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, el valor de $pheromoneVertex[k][i]$ es el valor de $\tau_1(s[\![k-1]\!], v_i)$.
- $pheromoneColor$: guarda la información del rastro de feromonas τ_2. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, y para todo color j, con $j \in J_i$, el valor de $pheromoneColor[k][i][j]$ es el valor de $\tau_2(s[\![k-1]\!], v_i, j)$.
- $desirabilityVertex$: guarda la información de la deseabilidad η_1. Para cada vértice v_i y solución parcial $s[\![k-1]\!]$, con $i, k \in \{1, 2, \ldots, n\}$, el valor de $desirabilityVertex[k][i]$ es el valor de $\eta_1(s[\![k-1]\!], v_i)$.
- $bestColouring$: es la mejor q-coloración propia de G construida por la colonia de hormigas. Se trata de una partición de $G.V$ en q conjuntos estables.
- $bestNumColours$: es el número de colores de $bestColouring$
- $trialMatrix$: dados cualesquiera dos vértices no adyacentes v_i y v_j, el valor de la entrada $trailMatrix[i][j]$ corresponde al rastro de feromonas existente entre v_i y v_j. Este valor es proporcional a la calidad de la coloración obtenida de colorear con el mismo color a los vértices v_i y v_j. Por lo tanto, $trialMatrix$ ayuda a mantener información acerca de lo conveniente que resulta agregar un vértice no coloreado a la clase de color actual para que siga siendo un conjunto independiente. Inicialmente, todos las entradas de $trialMatrix$,

correspondientes a vértices no adyacentes, quedan establecidas en 1. El factor de evaporación ε se utiliza para que el rastro de feromonas almacenado en $trialMatrix$ se evapore progresivamente en cada ciclo realizado por la metaheurística. Sean $s_1, s_2, \ldots, s_{nAnts}$ las coloraciones válidas construidas por la colonia al final de una iteración cualquiera y sea S_{ij} el conjunto de todas las coloraciones de $\{s_1, s_2, \ldots, s_{nAnts}\}$ en las cuales v_i y v_j tienen el mismo color. Finalmente, sea q_a el número de colores de s_a, para toda $a \in \{1, 2, \ldots, nAnts\}$. El valor de $trialMatrix[i][j]$ se actualiza de la siguiente manera:

$$trialMatrix[i][j] = \varepsilon \cdot trialMatrix[i][j] + \sum_{s_a \in S_{ij}} \frac{1}{q_a},$$

para todas $i, j \in \{1, 2, \ldots, n\}$.

- $updateMatrix$: es una matriz cuadrada de tamaño n que ayuda a actualizar los valores de $trialMatrix$. Inicialmente, todos las entradas de $updateMatrix$ se establecen en 0.
- $uncolouredVertices$: guarda el conjunto de todos los vértices no coloreados para la hormiga actual. Inicialmente $uncolouredVertices$ es igual a $G.V$.
- $colouredVertices$: guarda el conjunto de vértices ya coloreados por la hormiga actual. Inicialmente es vacío.
- $numColouredVertices$: guarda el número de vértices ya coloreados por la hormiga actual. Su valor inicial es 0.

4.1.3. Funciones complementarias

Las funciones complementarias utilizadas por la metaheurística **D-ANTCOL** se describen a continuación:

- InicializarColonia(): establece el valor indicado para los parámetros $nCycles$, $nAnts$, α, β y ε. Ademas, inicializa los valores de las entradas de $trialMatrix$. La especificación de esta función se muestra en el Pseudocódigo 4.2.
- InicializarRastro1(): asigna el valor inicial 1 a las entradas de $pheromoneVertex$. La especificación de esta función se muestra en el Pseudocódigo 4.3.
- InicializarRastro2(): inicializa el valor de las entradas de $pheromoneColor$ a 1. La especificación de esta función se muestra en el Pseudocódigo 4.4.
- InicializarDeseabilidad1(): da el valor inicial 1 a las entradas de $desirabilityVertex$ a 1. La especificación de esta función se muestra en el Pseudocódigo 4.5.
- Colorear(s_{id}, q): hace que la hormiga actual id cree una partición $\{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables y la guarde en s_{id}. La especificación de esta función se muestra en el Pseudocódigo 4.6.
- DejarRastro(s_{id}, q): regista en $updateMatrix$ el rastro de feromonas dejado por la hormiga actual, entre cada par de vértices no adyacentes, en cada una de las q clases de color de la q-coloración válida s_{id} de G que construyó. La especificación de esta función se muestra en el Pseudocódigo 4.7.
- ActualizarMatrizRastros(): Actualiza los valores de $trialMatriz$ con ayuda de los valores de $updateMatrix$. La especificación de esta función se muestra en el Pseudocódigo 4.8.

Pseudocódigo 4.2 Función InicializarColonia

Entrada: Los valores recomendados para los parámetros $nCycles$, $nAnts$, α, β y ε.
Salida: El establecimiento de los valores de los parámetros $nCycles$, $nAnts$, α, β y ε. Además, la inicialización de los valores de la matriz de rastros de feromonas.

```
1: nCycle ← 50
2: nAnts ← 100
3: α ← 2; β ← 4; ε ← 0.5
4: for i ← 1 to n do
5:     for j ← 1 to n do
6:         if ( v_i v_j ∉ G.E ) then
7:             trialMatrix[i][j] ← 1
8:         end if
9:     end for
10: end for
```

Pseudocódigo 4.3 Función InicializarRastro1

Entrada: El valor inicial para el rastro de feromonas τ_1.
Salida: La inicialización del valor de las entradas de $pheromoneVertex$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         pheromoneVertex[i][j] ← 1
4:     end for
5: end for
```

Pseudocódigo 4.4 Función InicializarRastro2

Entrada: El valor inicial para el rastro de feromonas τ_2.
Salida: La inicialización del valor de las entradas de $pheromoneColor$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         for k ← 1 to n do
4:             pheromoneColor[i][j][k] ← 1
5:         end for
6:     end for
7: end for
```

Pseudocódigo 4.5 Función InicializarDeseabilidad1

Entrada: El valor inicial para la deseabilidad η_1.
Salida: La inicialización del valor de las entradas de $desirabilityVertex$.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         desirabilityVertex[i][j] ← 1
4:     end for
5: end for
```

Pseudocódigo 4.6 Función COLOREAR

Entrada: El conjunto s_{id} y el entero q.
Salida: Una partición $s_{id} = \{V_1, V_2, \ldots, V_q\}$ de $G.V$ en q conjuntos estables.

while ($uncolouredVertices\ != \varnothing$) **do**
  $\rho \leftarrow 1/|uncolouredVertices|$
  Seleccionar de manera aleatoria, con probabilidad $1/\rho$, un vértice $v \in uncolouredVertices$
  Sea $i \in \{1, 2, \ldots, n\}$ el índice de v en $G.V$
  $q \leftarrow q + 1$; $V_q \leftarrow \{v_i\}$; Insertar V_q en s_{id}, como elemento
  $colouredVertices \leftarrow colouredVertices \cup \{v_i\}$
  $numColouredVertices \leftarrow numColouredVertices + 1$
  $W \leftarrow \{w \in uncolouredVertices \mid w \notin Adj_{G[uncolouredVertices]}(v_i)\} \setminus \{v_i\}$
  $B \leftarrow \{w \in uncolouredVertices \mid w \in Adj_{G[uncolouredVertices]}(v_i)\}$
  while ($W\ != \varnothing$) **do**
    $numColouredVertices \leftarrow numColouredVertices + 1$
    $I \leftarrow \{\ i \in \{1, 2, \ldots, n\} \mid v_i \notin colouredVertices\}$
    for each $i \in I$ **do**
      for $j \leftarrow 1$ **to** q **do**
        $trialSum \leftarrow 0$
        for each $v \in s_{id}.V_j$ **do**
          Sea $k \in \{1, 2, \ldots, n\}$ el índice de v en $G.V$
          $trialSum \leftarrow trialSum + trialMatriz[k][i]$
        end for
        $pheromoneColor[numColouredVertices][i][j] \leftarrow trialSum/|s_{id}.V_j|$
      end for
    end for
    for each $i \in I$ **do**
      $t \leftarrow pheromoneColor[numColouredVertices][i][q]$
      $pheromoneVertex[numColouredVertices][i] \leftarrow t$
    end for
    Seleccionar de manera aleatoria, con probabilidad $1/3$, un elemento $desirability \in \{1, 2, 3\}$
    if ($desirability == 1$) **then**
      for each $i \in I$ **do**
        $desirabilityVertex[numColouredVertices][i] \leftarrow deg_{G[B]}(v_i)$
      end for
    end if
    if ($desirability == 2$) **then**
      for each $i \in I$ **do**
        $desirabilityVertex[numColouredVertices][i] \leftarrow |W| \setminus deg_{G[W]}(v_i)$
      end for
    end if
    if ($desirability == 3$) **then**
      for each $i \in I$ **do**
        $desirabilityVertex[numColouredVertices][i] \leftarrow deg_{G[W \cup B]}(v_i)$
      end for
    end if
    $J \leftarrow \{\ i \in I \mid v_i \in W\}$
    $j \leftarrow$ **arg max** $\{\ \rho_{it}(numColouredVertices, l) \mid l \in J\ \}$
    $s_{id}.V_q \leftarrow s_{id}.V_q \cup \{v_j\}$
    $colouredVertices \leftarrow colouredVertices \cup \{v_j\}$
    $B \leftarrow B \cup Adj_{G[W]}(v_j)$
    $W \leftarrow W \setminus (Adj_{G[W]}(v_j) \cup \{v_j\})$
  end while
  $uncolouredVertices \leftarrow uncolouredVertices \setminus s_{id}.V_q$
end while

Pseudocódigo 4.7 Función DejarRastro

Entrada: La q-coloración válida s_{id} de G construida por la hormiga actual id.
Salida: El rastro de feromonas dejado por la hormiga actual entre cada par de vértices no adyacentes en cada una de las q clases de color de esta q-coloración válida de G que construyó.

```
1: for i ← 1 to q do
2:     for each u ∈ s_id.V_i do
3:         Sea j ∈ {1, 2, ..., n} el índice de u en G.V
4:         for each v ∈ s_id.V_i do
5:             Sea k ∈ {1, 2, ..., n} el índice de v en G.V
6:             updateMatrix[j][k] ← updateMatrix[j][k] + 1/q
7:         end for
8:     end for
9: end for
```

Pseudocódigo 4.8 Función ActualizarMatrizRastros

Entrada: La matriz de rastros de feromonas.
Salida: La actualización de la matriz de rastros de feromonas.

```
1: for i ← 1 to n do
2:     for j ← 1 to n do
3:         if ( v_i v_j ∉ G.E ) then
4:             trialMatrix[i][j] ← ε * trialMatrix[i][j] + updateMatrix[i][j]
5:         end if
6:     end for
7: end for
```

Capítulo 5

La Metaheurística *ABAC*

Ants are good citizens, they place group interests first.
—*This Simian World*, Clarence Day, 1920

En este capítulo se expondrá la metaheurística, llamada ***ABAC***[10], presentada por Bui et al. en el artículo *"An ant-based algorithm for coloring graphs"* para el Problema de la Coloración de Gráficas. Aunque los autores refieren que *ABAC* es una metaheurística basada en colonia de hormigas, no es un algoritmo *ACO*, pues:

- A diferencia de los verdaderos algoritmos *ACO* para colorear gráficas, donde cada hormiga colorea la gráfica completa, cada hormiga en *ABAC* colorea sólo una parte de la gráfica usando únicamente información local. Estas coloraciones locales hechas por las hormigas de *ABAC* forman una coloración completa para la gráfica.
- Sus hormigas no poseen estigmergia (a través de feromonas), en su lugar, utilizan una estructura tabú para evitar quedar atrapadas en la misma parte de la gráfica.

A pesar de esto, los autores indican que la metaheurística tuvo un buen desempeño en el conjunto de 119 gráficas que tomaron para sus pruebas, indicando que la metaheurística produjo resultados muy consistentes, teniendo desviaciones estándar muy pequeñas en más de 50 ejecuciones para cada gráfica del conjunto [11].

5.1. Metaheurística ABAC

La idea principal de **ABAC** consiste en distribuir aleatoriamente un conjunto de hormigas en los vérices de una gráfica simple, tomada como entrada, para que la coloreen. Cada hormiga sigue el mismo conjunto de reglas para colorear los vértices de la gráfica. A diferencia de las hormigas de los algoritmos *ACO* para el Problema de la Coloración de Gráficas, las hormigas de **ABAC** no encuentra una solución completa al problema. En su lugar, cada hormiga de **ABAC** colorea sólo una parte de la gráfica. De esta manera, **ABAC** es más susceptible a una implementación distribuida. Sin embargo, no se presentará tal implementación en este trabajo. Otra diferencia que distingue a **ABAC** de un auténtico algoritmo *ACO* es que sus hormigas

[10]Siglas en inglés de *Ant-Based Algorithm for Coloring Graphs*. En adelante se utilizará el acrónimo *ABAC* para referirse a esta metaheurística.

no tienen capacidad de tendido y seguimiento de rastro de feromonas. Los autores indican que esto es muy útil para **ABAC**, pues ayuda a reducir su tiempo de ejecución y, en experimentos limitados, esta ausencia de estigmergia no mostró un efecto visible o significativo en la calidad de las soluciones producidas. El esquema completo de **ABAC** se muestra en el Pseudocódico 5.1. En las siguientes subsecciones se describirá a detalle las diversas partes de la metaheurística.

5.1.1. Estrategia general

Sea $G = (V, E)$ la gráfica de entrada, con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$. Primero se obtiene una k-coloración válida para G, con conjunto de colores $\{1, 2, \ldots, k\}$. Nótese que k es una cota superior del número cromático de G, es decir, $k \geqslant \chi(G)$. En seguida, una k_1-coloración para G, con $k_1 \leqslant k$ y que puede no ser una coloración válida, se deriva de esta k-coloración válida; esta será la coloración inicial. A continuación, una colonia de hormigas es distribuida de manera aleatoria en los vértices de G. La metaheurística procede entonces a ejecutar una serie de ciclos. En cada iteración, cada hormiga de la colonia intenta colorear la parte de la gráfica más cercana a su ubicación usando únicamente el conjunto actual de colores que se tiene disponible. Al final de un ciclo, si el conflicto total de G es 0 (esto es, si la coloración actual es válida), el número actual de colores disponibles se reduce en uno y comienza otro ciclo. En otro caso, el número actual de colores disponibles se aumenta en uno antes de iniciar otra iteración. La metaheurística también podría tomar otras acciones para sacar a la colonia de un potencial óptimo local antes de que comience otro ciclo. Las condiciones de paro se describen con detalle en la sección 5.1.5.

5.1.2. Parámetros

A continuación se dará una breve descripción de cada uno de los parámetros utilizados por la metaheurística **ABAC**, así como los valores recomendados para estos por los autores con base en sus experimentos.

- n: es el número de vértices de G.
- *antColony*: es la colonia de hormigas que será distribuida en G. Cada hormiga tiene los siguientes atributos:
  - $antId$: su identificador.
  - $currentVertex$: el vértice actual donde se encuentra.
  - $recentlyVisited$: la lista tabú que contiene aquellos vértices recientemente visitados por la hormiga que se volverán prohibidos, esto es, vértices que la hormiga evitará volver a visitar en próximas iteraciones. Esta lista tabú ayudará a evitar que la hormiga se quede atrapada en la misma área de la gráfica coloreando los mismos vértices una y otra vez, permitiendo una exploración más diversa de la gráfica.
  - $sizeLimit$: es la longitud máxima de $recentlyVisited$ y su valor será establecido en $\lceil nMoves/3 \rceil$.

  Por otro lado, **ABAC** cuenta con las siguientes funciones para manipular a la colonia:

  - CrearHormiga(id, s): crea una hormiga con identificador id y con longitud de lista tabú s. La especificación de esta función se muestra en el Pseudocódigo 5.2.
  - AsignarVertice(id, v): asigna a la hormiga con identificador id el vértice v. La especificación de esta función se muestra en el Pseudocódigo 5.3.

Pseudocódigo 5.1 Metaheurística **ABAC**

Entrada: Una gráfica $G = (V, E)$ con $G.V = \{v_1, v_2, \ldots, v_n\}$ y $G.E = \{e_1, e_2, \ldots, e_m\}$.
Salida: Un arreglo de tamaño n que contiene una q-coloración para G, tal que para cada $i \in \{1, 2, \ldots, n\}$, el valor de su i-ésimo elemento es el color de v_i.

```
1: /* Parámetros */
2: n ← |G.V|
3: antColony ← ∅
4: nCycles ← 0; nAnts ← 0; nMoves ← 0
5: α ← 0.8; β ← 0.5; γ ← 0.7
6: Sean currentColouring y bestColouring dos arreglos de tamaño n
7: availableColours ← 0; bestNumColours ← 0
8: MXRLFclassLimit ← 0
9: Sea vertexConflict un arreglo de tamaño n
10: totalConflict ← 0
11: nChangeCycles ← 0
12: nJoltCycles ← 0
13: nBreakCycles ← 0
14: CalcularParametros()
15:
16: /* Coloración inicial */
17: CrearColoracionInicial()
18: CalcularConflictos()
19: CalcularConflictoTotal()
20:
21: /* Coloración construida por las hormigas */
22: DistribuirHormigas()
23: for cycle ← 1 to nCycles do
24:     for id ← 1 to nAnts do
25:         for move ← 1 to nMoves do
26:             ColorearVertice(id)
27:             ActualizarConflictoLocal(id)
28:             ActualizarListaTabu(id)
29:             MoverALongitudDos(id)
30:         end for
31:     end for
32:     calcularConflictoTotal()
33:     if ( totalConflict == 0 and bestNumColours > availableColours ) then
34:         bestColouring ← currentColouring
35:         bestNumColours ← availableColours
36:         availableColours ← availableColours − 1
37:     end if
38:
39:     /* Perturbación y condición de paro */
40:     if ( availableColours no ha mejorado en nChangeCycles ) then
41:         availableColours ← availableColours + 1
42:     end if
43:     if ( availableColours no ha mejorado en nJoltCycles ) then
44:         RealizarOperacionJolt()
45:     end if
46:     if ( bestNumColours no ha mejorado en nBreakCycles ) then
47:         break
48:     end if
49: end for
50: return bestColouring
```

- ColorearVertice(id): la hormiga con identificador id, que se encuentra en G, colorea su vértice actual con un color del conjunto de colores $\{1, 2, \ldots, availableColours\}$ que reduce al máximo su conflicto actual bajo $currentColouring$. La especificación de esta función se muestra en el Pseudocódigo 5.4.
- ActualizarConflictoLocal(id): la hormiga con identificador id, que se encuentra en G, actualiza el conflicto de su vértice actual y el de sus vecinos bajo $currentColouring$. La especificación de esta función se muestra en el Pseudocódigo 5.5.
- ActualizarListaTabu(id): la hormiga, que se encuentra en G, con identificador id agrega su vértice actual a su lista tabú, siempre que la longitud máxima no sea excedida. La especificación de esta función se muestra en el Pseudocódigo 5.6.
- MoverALongitudDos(id): la hormiga con identificador id, que se encuentra en G, se mueve de su vértice actual a otro vértice usando un camino de longitud 2. La primera arista del camino es seleccionada de manera aleatoria por la hormiga de entre todas las aristas en G que tienen a su vértice actual como vértice inicial. La segunda arista seleccionada por la hormiga tiene como vértice inicial al vértice final de la primera arista seleccionada y como vértice final aquel vértice de entre todos los vértices adyacentes a su vértice inicial, y que no pertenecen a la lista tabú de la hormiga, con el mayor conflicto según $vertexConflict$. La especificación de esta función se muestra en el Pseudocódigo 5.7. Es importante mencionar que, la selección de u en la línea 2 puede realizarse de manera aleatoria bajo una distribución uniforme considerando el número de vecinos del vértice actual de la hormiga o, simplemente, puede ser tomando cualquiera, en particular el primero. Además. otro punto a resaltar es que el vértice w, de la línea 5, es el vértice a distancia dos del vértice actual de la hormiga en cuestión.

Pseudocódigo 5.2 Función CrearHormiga

Entrada: Dos enteros positivos id y s.
Salida: Una hormiga con identificador id y longitud de lista tabú s.
1: Sea ant una hormiga
2: $ant.antId \leftarrow id$
3: $ant.recentlyVisited \leftarrow \varnothing$
4: $ant.sizeLimit \leftarrow s$
5: **return** ant

Pseudocódigo 5.3 Función AsignarVertice

Entrada: Un entero positivo id y un vértice v.
Salida: El vértice actual de la hormiga con identificador id es v.
1: Sea ant la hormiga con identificador id
2: $ant.currentVertex \leftarrow v$

- $nCycles$: es el número de ciclos en todo el proceso de coloración y su valor será establecido en $\text{mín}\{6n, 4000\}$.
- $nAnts$: es el número de hormigas en la colonia y será establecido en 20 % del número de vértices de G. Por razones de eficiencia, $nAnts$ no deberá exceder 100.

Pseudocódigo 5.4 Función ColorearVertice

Entrada: Un entero positivo id
Salida: La hormiga con identificador id coloreó su vértice actual con un color que minimiza su conflicto.

Sea ant, la hormiga en G, con identificador id.
Sea $i \in \{1, 2, \ldots, n\}$ el índice de $ant.currentVertex$ en $G.V$
$A \leftarrow \{1, 2, \ldots, availableColours\}$
$M \leftarrow \{a \in A \mid a$ minimiza el conflicto de $ant.currentVertex\}$
Seleccionar de manera aleatoria, con probabilida $1/|M|$, $c \in M$
$currentColouring[i] \leftarrow c$

Pseudocódigo 5.5 Función ActualizarConflictoLocal

Entrada: Un entero positivo id.
Salida: La hormiga con identificador id actualizó el conflicto de su vértice actual.

Sea ant, la hormiga en G, con identificador id
Sea $i \in \{1, 2, \ldots, n\}$ el índice de $ant.currentVertex$ en $G.V$
Sea c el número de vértices en $Adj_G(ant.currentVertex)$ con el mismo color que $ant.currentVertex$ bajo $currentColouring$
$vertexConflict[i] \leftarrow c$
for each $v \in Adj_G(ant.currentVertex)$ **do**
    Sea $j \in \{1, 2, \ldots, n\}$ el índice de v en $G.V$
    Sea d el número de vértices en $Adj_G(v)$ con el mismo color que v bajo $currentColouring$
    $vertexConflict[j] \leftarrow d$
end for

Pseudocódigo 5.6 Función ActualizarListaTabu

Entrada: Un entero positivo id.
Salida: La hormiga con identificador id agregó a su lista tabú su vértice actual.

Sea ant, la hormiga en G, con identificador id
if ($|ant.recentlyVisited| + 1 \leqslant ant.sizeLimit$) **then**
    $ant.recentlyVisited \leftarrow recentlyVisited \cup \{ant.currentVertex\}$
end if

Pseudocódigo 5.7 Función MoverALongitudDos

Entrada: Un entero positivo id.
Salida: La hormiga con identificador id se movió de su vértice actual a otro vértice usando un camino de longitud 2.

Sea ant, la hormiga en G, con identificador id
$t \leftarrow |Adj_G(ant.currentVertex)|$
if (t ! = 0) **then**
    Seleccionar de manera aleatoria, con probabilidad $1/t$, $u \in Adj_G(ant.currentVertex)$
    $F \leftarrow \{v \in Adj_G(u) \mid v \notin ant.recentlyVisited\}$
    if (F ! = $\varnothing$) **then**
        $w \leftarrow$ **arg max** { conflicto de v en $vertexConflict \mid v \in F$ }
        $ant.currentVertex \leftarrow w$
    end if
end if

- $nMoves$: es el número de vértices que una hormiga puede visitar antes de que se detenga. $nMoves$ será establecido de la siguiente manera:

$$nMoves = \begin{cases} \lceil n/4 \rceil & \text{si } nAnts < 100, \\ 20 + \lceil n/nAnts \rceil & \text{en otro caso.} \end{cases}$$

- α: es el porcentaje de colores producidos por **MXRLF** que estará disponible para que las hormigas lo utilicen inicialmente. Su valor se establece en 0.8.
- β: es el porcentaje de colores producidos por **MXRLF** que se ocupará para crear una coloración inicial para G antes de distribuir la colonia en ésta. Su valor se establece en 0.5.
- γ: es el porcentaje de colores producidos por **MXRLF** que se usará para colorear aquellos vértices cuyo color no pertenezca al conjunto de colores $\{1, 2, \ldots, \lceil \beta k \rceil\}$ cuando se cree la coloración inicial para G. Su valor se establece en 0.7.
- $currentColouring$: es la coloración actual de G, construida por la colonia de hormigas. Para cada $i \in \{1, 2, \ldots, n\}$, $currentColouring[i]$ es el color del vértice v_i. El valor inicial de $currentColouring$ será establecido por **MXRLF**, por lo que el valor inicial de $currentColouring$ será una k-coloración válida de G.
- $bestColouring$: es la mejor l-coloración actual de G, construida por la colonia de hormigas. Para cada $i \in \{1, 2, \ldots, n\}$, $bestColouring[i]$ es el color del vértice v_i. El valor inicial de $bestColouring$ será el valor inicial de $currentColouring$.
- $availableColours$: es el número de colores del conjunto actual de colores disponibles que utiliza la colonia de hormigas para construir la coloración $currentColouring$. El valor inicial de $availableColours$ será $\lceil \alpha k \rceil$ y el conjunto actual de colores disponibles inicial será $\{1, 2, \ldots, \lceil \alpha k \rceil\}$. En cada iteración de **ABAC**, el conjunto actual de colores disponibles será $\{1, 2, \ldots, availableColours\}$.
- $bestNumColours$: es el número de colores (de clases de color) de la coloración $bestColouring$. La gama de colores de $bestColouring$ siempre será $\{1, 2, \ldots, bestNumColours\}$. El valor inicial de $bestNumColour$ será k, esto es, el número de clases de color de $bestColouring$ en su estado inicial.
- $MXRLFclassLimit$: es el tamaño máximo que pueden tener las clases de color producidas por **MXRLF**. Su valor será establecido en $\lceil 0.7n \rceil$.
- $vertexConflict$: contiene el conflicto actual, bajo $currentColouring$, de cada vértice de G. Para cada $i \in \{1, 2, \ldots, n\}$, $vertexConflict[i]$ es el conflicto del vértice v_i.
- $totalConflict$: es el conflicto total de G bajo $currentColouring$.
- $nChangeCycles$: es el número de ciclos consecutivos permitidos por **ABAC**, durante los cuales la colonia de hormigas no ha mejorado el número de colores del conjunto actual de colores disponibles, antes de que incremente en uno el valor de $availableColours$. El valor de $nChangeCycles$ se establecerá en 20.
- $nJoltCycles$: es el número de ciclos consecutivos permitidos por **ABAC**, durante los cuales la colonia de hormigas no ha mejorado el número de colores del conjunto actual de colores disponibles, antes de que le aplique una operación $jolt$ a la coloración actual para perturbarla. Una operación $jolt$ selecciona todos aquellos vértices en $G.V$ con conflicto actual, bajo $currentColouring$, mayor al 10 % de $totalConflict$ y vuelve a colorear, de manera aleatoria, a los vecinos de estos vértices utilizando el 80 % del conjunto

actual de colores disponibles. Para aplicar una operación *jolt* a *currentColouring*, la metaheurística **ABAC** utiliza la función RealizarOperacionJolt(); la especificación de ésta se muestra en el Pseudocódigo 5.8. El valor de *nChangeCycles* se establecerá en máx$\{\lceil n/2 \rceil, 600\}$.

- *nBreakCycles*: es el número de ciclos consecutivos permitidos por **ABAC**, durante los cuales la colonia de hormigas no ha mejorado el valor de *bestNumColours* y, por tanto, el de la coloración *bestNumColours*, antes de terminar su ejecución. El valor de *nChangeCycles* se establecerá en máx$\{\lceil 5n/2 \rceil, 1600\}$.

Pseudocódigo 5.8 Función RealizarOpercionJolt

Entrada: La coloración actual de G, *currentColouring*.
Salida: La coloración *currentColouring* fue perturbada a través de una operación *jolt*.

```
1: P ← ∅
2: for i ← 1 to n do
3:     if ( vertexConflict[i] > totalConflict/10 ) then
4:         P ← P ∪ {v_i}
5:     end if
6: end for
7: d ← ⌈0.8 * availableColours⌉
8: D ← {1, 2, ..., d}
9: for each u ∈ P do
10:    for each w ∈ Adj_G(u) do
11:        Sea i ∈ {1, 2, ..., n} el índice de w en G.V
12:        Seleccionar de manera aleatoria, con probabilidad 1/d, c ∈ D
13:        currentColouring[i] ← c
14:    end for
15: end for
```

El cálculo y establecimiento del valor (inicial o fijo) de cada parámetro descrito anteriormente (excepto los parámetros *currentColouring*, *bestColouring*, *bestNumColours* y *availableColours*), será realizado por la función CalcularParametros(), cuya especificación se muestra en el Pseudocódigo 5.9.

5.1.3. Coloración inicial

El primer objetivo de **ABAC** es encontrar una cota superior para el número cromático de G. Para este propósito, **ABAC** utiliza el algoritmo **MXRLF**, cuya definición se detalla en el Apéndice C, haciendo una invocación de éste con los siguientes parámetros: **MXRLF**(G, *currentColouring*, 1, *MXRLFclassLimit*). Lo que implica que *currentColouring* contiene una k-coloración válida para G, con gama de colores $\{1, 2, \ldots, k\}$. Lo que implica que esta k-coloración válida es la mejor coloración válida para G encontrada inicialmente. Por lo tanto, el valor inicial de la coloración *bestColouring* es esta k-coloración valida y el valor inicial de *bestNumColours* es k.

El segundo objetivo de **ABAC** es derivar una coloración inicial para la colonia de hormigas, a partir de la coloración *currentColouring*. Para este propósito, **ABAC** procede de la siguiente manera: borra el color de los vértices cuyo color actual no pertenezca a la gama de colores $\{1, 2, \ldots, \lceil \beta k \rceil\}$. Luego, colorea estos vértices usando, de manera aleatoria, los colores del conjunto $\{1, 2, \ldots, \lceil \gamma k \rceil\}$. De esta manera, **ABAC** ha creado una $\lceil \gamma k \rceil$-coloración, no necesariamente válida, para G a partir de *currentColouring*. Esta coloración será usada, entonces, por la

Pseudocódigo 5.9 Función CalcularParametros

Entrada: El número de vértices de G, n.
Salida: El establecimiento de los valores de los parámetros.

```
1: nCycles ← min {6 * n, 400}
2: nAnts ← ⌈0.2 * n⌉
3: if ( nAnts < 100 ) then
4:     nMoves ← ⌈n/4⌉
5: else
6:     nMoves ← 20 + ⌈n/nAnts⌉
7: end if
8: MXRLFclassLimit ← ⌈0.7 * n⌉
9: for i ← 1 to nAnts do
10:    ant ← CrearHormiga(i, ⌈nMoves/3⌉)
11:    antColony ← antColony ∪ {ant}
12: end for
13: currentColouring ← MXRLF(G, G.V, currentColouring, 1, MXRLFclassLimit)
14: Sea k el número de colores de currentColouring
15: bestColouring ← currentColouring
16: bestNumColours ← k
17: availableColours ← ⌈α * k⌉
18: nChangesCycles ← 20
19: nJoltCycles ← max {⌈n/2⌉, 600}
20: nBreakCycles ← max {⌈(5 * n)/2⌉, 1600}
```

colonia de hormigas como punto de partida. Nótese que, $\{1, 2, \ldots, \lceil \beta k \rceil\} \subseteq \{1, 2, \ldots, \lceil \gamma k \rceil\} \subseteq \{1, 2, \ldots, \lceil \alpha k \rceil\}$, pues $0 < \beta < \gamma < \alpha < 1$. Además, recordemos que $\{1, 2, \ldots, \lceil \alpha k \rceil\}$ es el conjunto actual de colores disponibles inicial.

La metaheurística **ABAC** utilizará la función CrearColoracionInicial() para realizar estos dos objetivos, la especificación de esta función se muestra en el Pseudocódigo 5.10

Finalmente, **ABAC** utilizará la función CalcularConflictos() y la función CalcularConflictoTotal() para calcular, respectivamente, el conflicto de cada vértice de G y el conflicto total de G bajo esta coloración inicial. Los Pseudocódigos 5.11 y 5.12 presentan una implementación de las mismas, respectivamente.

5.1.4. ¿Cómo colorean las hormigas?

La metaheurística usa la función DistribuirHormigas(), presentada en el Pseudocódigo 5.13, para distribuir, de manera aleatoria, las $nAnts$ hormigas de la colonia en los vértices de G. A continuación, **ABAC** ejecuta una serie de ciclos. En cada una de estas iteraciones, las hormigas son activadas por **ABAC** una a la vez. Cuando una hormiga es activada, colorea únicamente un área local de G, pues sólo conoce la información de su vértice actual y la de sus vecinos, usando el conjunto actual de colores disponibles, es decir, el conjunto $\{1, 2, \ldots, availableColours\}$. El objetivo de la hormiga activada es colorear (o recolorear) su vértice actual para que su conflicto sea cero. Si esto no es posible, la hormiga seleccionará, de manera aleatoria, un color en $\{1, 2, \ldots, availableColours\}$ que minimice el conflicto de su vértice actual. Para llevar a cabo esta tarea de coloración por parte de la hormiga activada, **ABAC** utiliza la función ColorearVertice(). En seguida, **ABAC** utiliza la función ActualizarConflictoLocal() para que la hormiga activada actualice el conflicto de su vértice actual.

Después de que una hormiga activada termina de colorear su vértice actual y actualizar el

Pseudocódigo 5.10 Función CrearColoracionInicial

Entrada: Los parámetros necesarios para crear la coloración inicial.
Salida: La coloración inicial para la colonia de hormigas.

```
1: B ← {1, 2, ..., ⌈β * k⌉}
2: C ← {1, 2, ..., ⌈γ * k⌉}
3: P ← ∅
4: for i ← 1 to n do
5:     if ( currentColouring[i] ∉ B ) then
6:         P ← P ∪ {v_i}
7:     end if
8: end for
9: for each v ∈ P do
10:    Sea j ∈ {1, 2, ..., n} el índice de v en G.V
11:    Seleccionar de manera aleatoria, con probabilidad 1/⌈γ * k⌉, c ∈ C
12:    currentColouring[j] ← c
13: end for
```

Pseudocódigo 5.11 Función CalcularConflictos

Entrada: La coloración actual de G, *currentColouring*.
Salida: El cálculo del conflicto de cada vértice de G.

```
1: for i ← 1 to n do
2:     Sea c el número de vértices en Adj_G(v_i) con el mismo color que v_i bajo currentColouring
3:     vertexConflict[i] ← c
4: end for
```

Pseudocódigo 5.12 Función CalcularConflictoTotal

Entrada: La coloración actual de G, *currentColouring*.
Salida: El cálculo del conflicto total de G.

```
1: t ← 0
2: for i ← 1 to n do
3:     t ← t + vertexConflict[i]
4: end for
5: totalConflict ← t/2
```

Pseudocódigo 5.13 Función DistribuirHormigas

Entrada: El número de hormigas de la colonia, *nAnts*.
Salida: Las hormigas de la colonia han sido distribuidas, de manera aleatoria, en los vértices de G.

```
1: I ← {1, 2, ..., n}
2: for i ← 1 to nAnts do
3:     Seleccionar de manera aleatoria, con probabilidad 1/|I|, j ∈ I
4:     AsignarVertice(i, v_j)
5:     I ← I \ {j}
6: end for
```

conflicto de este vértice, **ABAC** le ordena a la hormiga, usando la función ActualizarListaTabu(), que agregue a su lista tabú su vértice actual. Después, le ordena, usando la función MoverADistanciaDos(), que se mueve a otro vértice, utilizando un camino de longitud dos, e intente colorearlo bajo las mismas reglas antes mencionadas. Cada hormiga hará $nMoves$ de estos movimientos antes de detenerse.

5.1.5. Perturbación y condición de paro

Cuando todas las hormigas han terminado su tarea de colorear, bajo las reglas impuestas por **ABAC**, al final de un ciclo, tenemos una coloración de G. Entonces, **ABAC** procede a calcular el conflicto total de G bajo esta coloración, llamada $currentColouring$. Si el conflicto total de G bajo $currentColouring$ es cero, se ha encontrado una mejor coloración válida que la almacenada en $bestColouring$ y, por tanto, **ABAC** procede a realizar las actualizaciones pertinentes sobre los parámetros $bestColouring$, $bestNumColours$ y $availableColours$; en particular, reduce en 1 el valor de $availableColours$. Por otro lado, **ABAC** incrementará en 1 el valor de $availableColours$, si el valor de $availableColours$ no ha mejorado en los últimos $nChangeCycles$. Además, **ABAC** mantiene la mejor coloración encontrada hasta el momento, es decir, los valores de $bestColouring$ y $bestNumColours$ no cambian.

Para ayudar a las hormigas a escapar de óptimos locales, **ABAC** perturbará el valor de la coloración actual, $currentColouring$, aplicándole una operación $jolt$, utilizando la función RealizarOperacionJolt(). Específicamente, **ABAC** aplicará una operación $jolt$ a la coloración actual creada por las hormigas, si el número de colores del conjunto actual de colores disponibles no se ha reducido en los últimos $nJoltCycles$ ciclos. La idea de la operación $jolt$ es inyectar la suficiente perturbación a la coloración actual para moverla fuera del óptimo local en el que se encuentra atrapada, pero no la suficiente como para destruir, totalmente, la coloración que ha creado la colonia de hormigas hasta el momento.

La metaheurística terminará su ejecución después de que se hayan efectuado el número de ciclos preestablecido, $nCycles$, o bien, si no se ha mejorado en los últimos $nBreakCycles$ el número de clases de color de la mejor coloración encontrada, es decir, el valor de $bestNumColours$.

Capítulo 6

Experimentos y Resultados

It is only when you watch the dense mass of thousands of ants, crowded together around the Hill, blackening the ground, that you begin to see the whole beast, and now you observe it thinking, planning, calculating. It is an intelligence, a kind of live computer, with crawling bits for its wits.
—Lewis Thomas, 1974

En este capítulo se presentan los detalles de los experimento realizados: el objetivo, los ejemplares y su codificación, los detalles de la implementación de las metaheurísticas; así como los resultados obtenidos.

6.1. Objetivo

Analizar el desempeño de cada una de las metaheurísticas descritas sobre un conjunto dado de gráficas (conjunto de ejemplares) a través de una serie de ejecuciones. Así como analizar el impacto del uso de feromonas en la calidad de la solución construida por cada una de éstas, y comparar los resultados obtenidos.

6.2. Detalles de la implementación

Cabe mencionar que cada una de las tres metaheurísticas expuestas en este trabajo fue implementada en el lenguaje de programación **Java 11.0.10**. Las tres implementaciones se realizaron bajo una misma representación para las gráficas de entrada, a saber, matriz de adyacencias.

6.3. Conjunto de ejemplares

Dado que la manera elegida para representar gráficas en la implementación de las metaheurísticas es matriz de adyacencias, cada uno de los ejemplares considerados para nuestros experimentos fueron codificados a través de su matriz de adyacencias correspondiente. A con-

tinuación, se describe brevemente el tipo de gráficas utilizadas (los detalles de la construcción se dan en el Apéndice E):

- Se trabajó con mallas bidimensionales ($D_{r\times s,\rho}$) y tridimensionales ($D_{r\times s\times t,\rho}$) de órdenes 50, 70, 100, 120 150, 170, 200, 220, 250, 270 y 300, a las que se agregaron diagonales con probabilidad $\rho = 0.25$, evitando el mayor número de cruces de aristas posible. En el caso de las mallas bidimensionales, nunca hubo cruces. Las Figuras 6.1 y 6.2 muestran un ejemplo sencillo de este tipo de mallas. Además, por construcción, estas dos clases de gráficas son, en el peor caso, 4-coloreables.

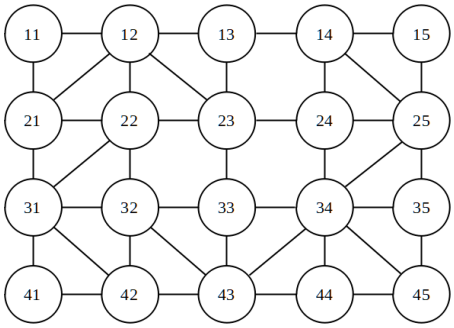

Figura 6.1: Malla bidimensional de orden 4×5 con diagonales.

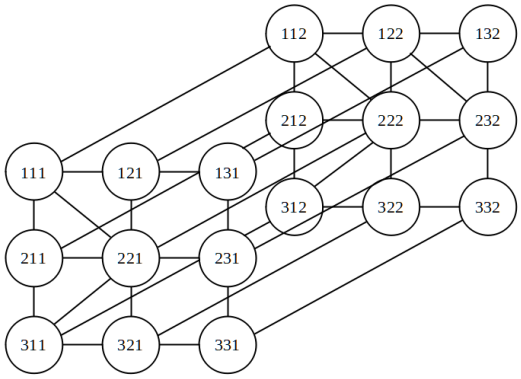

Figura 6.2: Malla tridimensional de orden $3 \times 3 \times 2$ con diagonales.

- Se trabajó con gráficas bipartitas r-regulares de orden $2t$ y saltos de s unidades ($B_{r,s,t}$), con $r = 5, 7, 11, 13, 17, 19, 23, 31, 37, 41$; $s = 7, 11, 13, 17, 19, 23, 31, 37, 41, 43$; y $t = 25, 35, 50, 60, 75, 85, 100, 110, 125, 135, 150$. La Figura 6.3 muestra un ejemplo sencillo de tipo de gráficas. Dado que todos estos ejemplares son gráficas bipartitas, por el Corolario 2.3, tienen número cromático igual a 2.
- Se trabajó con gráficas k-partitas completas de orden n y tamaño máximo ($K_{max}(n,k)$), con $n = 50, 70, 100, 120, 150, 170, 200, 220, 250, 270, 300$; y $k = 5, 7, 11, 13, 17, 19, 23, 31, 37, 41, 43$. La Figura 6.4 muestra un ejemplo simple de clase de gráficas. Como se trata de gráficas k-partitas completas, con $k > 2$, se sigue del Corolario 2.4 que todos estos ejemplares tienen número cromático igual a k.

- Se trabajó con gráficas aleatorias de orden n y probabilidad ρ (denotadas $G(n, \rho)$), con $n = 50, 70, 100, 120, 150, 170, 200, 220, 250, 270\ 300$; y $\rho = 0.2, 0.4, 0.6, 0.8$. La Figura 6.5 muestra un ejemplo simple de clase de gráficas.

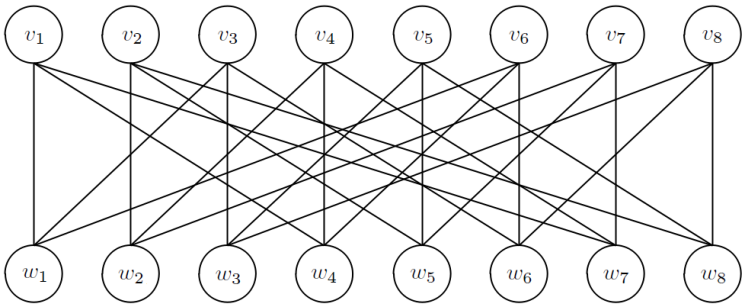

Figura 6.3: Gráfica bipartita 3-regular de orden 16 y saltos de 3 unidades.

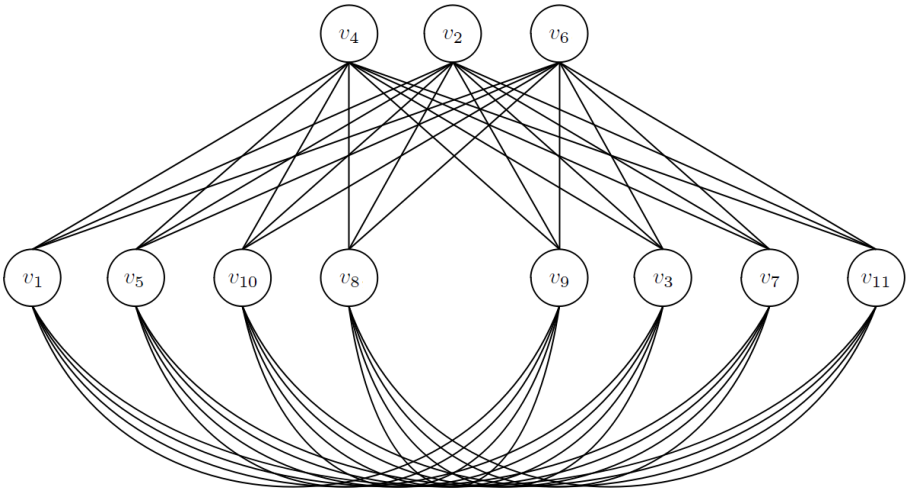

Figura 6.4: Ejemplo de gráfica 3-partita de orden 11 y tamaño máximo.

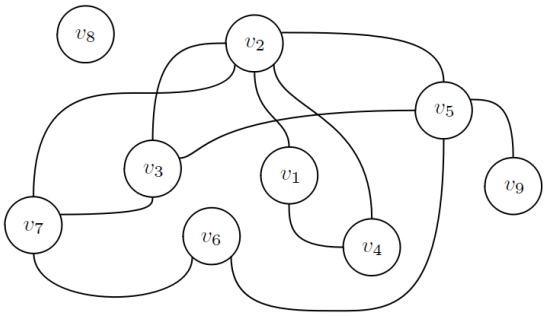

Figura 6.5: Ejemplo de gráfica aleatoria de orden 9 y probabilidad 0.5.

La Tabla 6.1 presenta el conjunto completo de ejemplares construidos. En ésta se describen los principales datos de cada gráfica de entrada: su número de vértices, es decir, su orden; su tamaño, o sea, su número de aristas; su densidad; su número cromático, si es que se conoce; el valor de k para el cual se satisface el Corolario A.4, siempre que tal valor exista; y la mejor cota p que se conoce de su número cromático. El valor D significa "Desconocido" y NE "No Existe".

Tabla 6.1: Conjunto de ejemplares construidos

Ejemplar	**Orden** n	**Tamaño** m	**Densidad** $\rho(G)$	**No. Cromático** $\chi(G)$	**Corolario A.4** $\chi(G) > k$	**Mejor cota** $\chi(G) \leqslant p$
$D_{2\times25,0.25}$	50	79	0.064	D	NE	4
$D_{5\times10,0.25}$	50	96	0.078	D	NE	4
$D_{2\times35,0.25}$	70	108	0.045	D	NE	4
$D_{7\times10,0.25}$	70	133	0.055	D	NE	4
$D_{4\times25,0.25}$	100	207	0.042	D	NE	4
$D_{10\times10,0.25}$	100	227	0.046	D	NE	4
$D_{4\times30,0.25}$	120	231	0.032	D	NE	4
$D_{10\times12,0.25}$	120	242	0.034	D	NE	4
$D_{3\times50,0.25}$	150	309	0.028	D	NE	4
$D_{10\times15,0.25}$	150	337	0.030	D	NE	4
$D_{2\times85,0.25}$	170	281	0.019	D	NE	4
$D_{10\times17,0.25}$	170	348	0.024	D	NE	4
$D_{4\times50,0.25}$	200	419	0.021	D	NE	4
$D_{10\times20,0.25}$	200	449	0.023	D	NE	4
$D_{4\times55,0.25}$	220	412	0.017	D	NE	4
$D_{10\times22,0.25}$	220	470	0.019	D	NE	4
$D_{2\times125,0.25}$	250	433	0.014	D	NE	4
$D_{10\times25,0.25}$	250	577	0.019	D	NE	4
$D_{3\times90,0.25}$	270	496	0.014	D	NE	4
$D_{10\times27,0.25}$	270	558	0.015	D	NE	4
$D_{6\times50,0.25}$	300	670	0.015	D	NE	4
$D_{10\times30,0.25}$	300	694	0.015	D	NE	4
$D_{2\times5\times5,0.25}$	50	114	0.093	D	NE	4
$D_{5\times5\times2,0.25}$	50	121	0.099	D	NE	4
$D_{2\times5\times7,0.25}$	70	172	0.071	D	NE	4
$D_{7\times5\times2,0.25}$	70	175	0.072	D	NE	4
$D_{2\times5\times10,0.25}$	100	240	0.048	D	NE	4
$D_{4\times5\times5,0.25}$	100	272	0.055	D	NE	4
$D_{5\times6\times4,0.25}$	120	329	0.046	D	NE	4
$D_{5\times2\times12,0.25}$	120	289	0.040	D	NE	4
$D_{3\times2\times25,0.25}$	150	348	0.031	D	NE	4
$D_{6\times5\times5,0.25}$	150	421	0.038	D	NE	4

Tabla 6.1 (Continuación)

Ejemplar	**Orden** n	**Tamaño** m	**Densidad** $\rho(G)$	**No. Cromático** $\chi(G)$	**Corolario A.4** $\chi(G) > k$	**Mejor cota** $\chi(G) \leqslant p$
$D_{2\times5\times17,0.25}$	170	419	0.029	D	NE	4
$D_{17\times5\times2,0.25}$	170	450	0.031	D	NE	4
$D_{2\times4\times25,0.25}$	200	478	0.024	D	NE	4
$D_{8\times5\times5,0.25}$	200	566	0.028	D	NE	4
$D_{2\times5\times22,0.25}$	220	540	0.022	D	NE	4
$D_{4\times5\times11,0.25}$	220	606	0.025	D	NE	4
$D_{2\times5\times25,0.25}$	250	620	0.020	D	NE	4
$D_{10\times5\times5,0.25}$	250	716	0.023	D	NE	4
$D_{3\times3\times30,0.25}$	270	682	0.019	D	NE	4
$D_{3\times9\times10,0.25}$	270	740	0.020	D	NE	4
$D_{2\times6\times25,0.25}$	300	754	0.017	D	NE	4
$D_{4\times15\times5,0.25}$	300	847	0.019	D	NE	4
$B_{5,7,25}$	50	125	0.102	2	NE	2
$B_{7,11,35}$	70	245	0.101	2	NE	2
$B_{11,13,50}$	100	550	0.111	2	NE	2
$B_{13,17,60}$	120	780	0.109	2	NE	2
$B_{17,19,75}$	150	1275	0.114	2	NE	2
$B_{19,23,85}$	170	1615	0.112	2	NE	2
$B_{23,29,100}$	200	2300	0.116	2	NE	2
$B_{29,31,110}$	220	3190	0.132	2	NE	2
$B_{31,37,125}$	250	3875	0.124	2	NE	2
$B_{37,41,135}$	270	4995	0.138	2	NE	2
$B_{41,43,150}$	300	6150	0.137	2	NE	2
$K_{max}(50,5)$	50	1000	0.816	5	4	5
$K_{max}(50,47)$	50	1222	0.998	47	22	47
$K_{max}(70,7)$	70	2100	0.869	7	5	7
$K_{max}(70,43)$	70	2388	0.989	43	22	43
$K_{max}(100,11)$	100	4545	0.918	11	9	11
$K_{max}(100,41)$	100	4873	0.984	41	24	41
$K_{max}(120,13)$	120	6645	0.931	13	10	13
$K_{max}(120,37)$	120	7002	0.981	37	24	37
$K_{max}(150,17)$	150	10587	0.947	17	14	17
$K_{max}(150,31)$	150	10885	0.974	31	22	31
$K_{max}(170,19)$	170	13689	0.953	19	15	19
$K_{max}(170,23)$	170	13819	0.962	23	19	23
$K_{max}(200,23)$	200	19128	0.961	23	18	23
$K_{max}(200,19)$	200	18945	0.952	19	17	19
$K_{max}(220,31)$	220	23418	0.972	31	24	31
$K_{max}(220,17)$	220	22776	0.945	17	14	17
$K_{max}(250,37)$	250	30402	0.977	37	29	37
$K_{max}(250,13)$	250	28845	0.927	13	11	13
$K_{max}(270,41)$	270	35556	0.979	41	32	41
$K_{max}(270,11)$	270	33135	0.912	11	10	11
$K_{max}(300,43)$	300	43953	0.980	43	34	43
$K_{max}(300,7)$	300	38571	0.860	7	6	7

Tabla 6.1 (Continuación)

Ejemplar	**Orden** n	**Tamaño** m	**Densidad** $\rho(G)$	**No. Cromático** $\chi(G)$	**Corolario A.4** $\chi(G) > k$	**Mejor cota** $\chi(G) \leqslant p$
$G_1(50, 0.3)$	50	351	0.287	D	NE	23
$G_2(50, 0.5)$	50	597	0.487	D	NE	33
$G_3(50, 0.7)$	50	882	0.720	D	3	43
$G_4(50, 0.9)$	50	1094	0.893	D	6	48
$G_1(70, 0.3)$	70	762	0.316	D	NE	32
$G_2(70, 0.5)$	70	1224	0.507	D	NE	44
$G_3(70, 0.7)$	70	1701	0.704	D	3	57
$G_4(70, 0.9)$	70	2169	0.898	D	7	68
$G_1(100, 0.3)$	100	1482	0.299	D	NE	46
$G_2(100, 0.5)$	100	2518	0.509	D	NE	63
$G_3(100, 0.7)$	100	3471	0.701	D	3	79
$G_4(100, 0.9)$	100	4435	0.896	D	7	95
$G_1(120, 0.3)$	120	2225	0.321	D	NE	50
$G_2(120, 0.5)$	120	3577	0.501	D	NE	74
$G_3(120, 0.7)$	120	4998	0.700	D	3	98
$G_4(120, 0.9)$	120	6404	0.897	D	7	116
$G_1(150, 0.3)$	150	3333	0.298	D	NE	62
$G_2(150, 0.5)$	150	5583	0.499	D	NE	92
$G_3(150, 0.7)$	150	7882	0.705	D	3	123
$G_4(150, 0.9)$	150	10097	0.904	D	8	143
$G_1(170, 0.3)$	170	4391	0.306	D	NE	73
$G_2(170, 0.5)$	170	7104	0.494	D	NE	99
$G_3(170, 0.7)$	170	9937	0.692	D	3	135
$G_4(170, 0.9)$	170	12938	0.901	D	8	162
$G_1(200, 0.3)$	200	6024	0.303	D	NE	75
$G_2(200, 0.5)$	200	10000	0.503	D	NE	118
$G_3(200, 0.7)$	200	13922	0.699	D	3	154
$G_4(200, 0.9)$	200	17891	0.899	D	9	192
$G_1(220, 0.3)$	220	7256	0.301	D	NE	84
$G_2(220, 0.5)$	220	12091	0.502	D	NE	126
$G_3(220, 0.7)$	220	16910	0.702	D	3	174
$G_4(220, 0.9)$	220	21800	0.905	D	9	209
$G_1(250, 0.3)$	250	9396	0.302	D	NE	93
$G_2(250, 0.5)$	250	15493	0.498	D	NE	143
$G_3(250, 0.7)$	250	21664	0.696	D	3	193
$G_4(250, 0.9)$	250	28022	0.900	D	9	237
$G_1(270, 0.3)$	270	11239	0.309	D	NE	104
$G_2(270, 0.5)$	270	18615	0.513	D	NE	166
$G_3(270, 0.7)$	270	25868	0.712	D	3	213
$G_4(270, 0.9)$	270	33100	0.911	D	10	260
$G_1(300, 0.3)$	300	13943	0.311	D	NE	125
$G_2(300, 0.5)$	300	22999	0.512	D	NE	175
$G_3(300, 0.7)$	300	31816	0.709	D	3	240
$G_4(300, 0.9)$	300	40734	0.908	D	9	285

6.4. Detalles de la ejecución

Cada metaheurística fue ejecutada, sobre cada uno de los ejemplares construido, 25 veces. Las ejecuciones se llevaron a cabo en un equipo portátil con las siguientes características:

- **Procesador**: Intel Core i5 2.4 Ghz
- **Memoria RAM**: 8 GB
- **HDD**: 500 GB
- **Sistema Operativo**: Ubuntu 20.04.2.0

6.5. Resultados

En esta sección se presentan los resultados obtenidos de aplicar los tres métodos a las gráficas antes descritas.

6.5.1. Mallas con diagonales

La Tabla 6.2 muestra el desempeño computacional de las tres metaheurísticas sobre el conjunto de mallas bidimensionales y tridimensionales con diagonales. En cada una de las columnas se indica el número de colores de la mejor q-coloración válida encontrada por la colonia de hormigas, así como el tiempo de ejecución t, en segundos, correspondiente.

Tabla 6.2: Desempeño de las metaheurísticas I

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$D_{2\times25,0.25}$	3	25.578	3	16.255	6	0.437
$D_{5\times10,0.25}$	3	23.309	3	15.812	6	0.326
$D_{2\times35,0.25}$	3	62.591	3	48.749	6	0.933
$D_{7\times10,0.25}$	4	65.100	3	38.981	6	1.005
$D_{4\times25,0.25}$	4	157.956	4	109.496	7	2.218
$D_{10\times10,0.25}$	4	168.668	4	116.289	5	1.920
$D_{3\times40,0.25}$	4	287.698	4	200.151	8	3.391
$D_{10\times12,0.25}$	4	299.431	4	237.243	7	3.853
$D_{3\times50,0.25}$	4	470.373	4	376.213	9	6.373
$D_{10\times15,0.25}$	4	555.257	4	350.270	8	6.336
$D_{2\times85,0.25}$	4	907.756	4	623.332	11	8.611
$D_{10\times17,0.25}$	4	845.384	4	529.922	8	9.316
$D_{4\times50,0.25}$	4	1386.105	4	1020.378	9	13.087
$D_{10\times20,0.25}$	4	1491.974	4	1045.222	11	12.763
$D_{4\times55,0.25}$	4	1952.104	4	1398.283	11	19.606
$D_{10\times22,0.25}$	4	1985.015	4	1342.028	11	15.807

Tabla 6.2 (Continuación)

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$D_{2\times125,0.25}$	4	2630.414	4	2135.469	15	25.939
$D_{10\times25,0.25}$	4	3247.108	4	2262.143	12	27.866
$D_{3\times90,0.25}$	4	3272.097	4	2351.431	15	28.703
$D_{10\times27,0.25}$	4	4269.282	4	3103.420	13	32.675
$D_{6\times50,0.25}$	4	4929.838	4	3463.669	15	46.332
$D_{10\times30,0.25}$	4	5008.165	4	3412.272	14	42.982
$D_{2\times5\times5,0.25}$	3	21.948	3	13.608	5	0.343
$D_{5\times5\times2,0.25}$	4	19.599	4	13.444	5	0.325
$D_{2\times5\times7,0.25}$	4	59.719	4	39.212	6	0.806
$D_{7\times5\times2,0.25}$	4	60.510	4	45.599	6	0.993
$D_{2\times5\times10,0.25}$	4	191.418	4	126.799	6	1.783
$D_{4\times5\times5,0.25}$	4	194.422	4	127.546	7	1.730
$D_{5\times6\times4,0.25}$	4	354.316	4	236.096	7	3.401
$D_{5\times2\times12,0.25}$	4	387.559	4	264.174	7	4.536
$D_{3\times2\times25,0.25}$	4	713.788	4	468.144	8	6.065
$D_{6\times5\times5,0.25}$	4	641.614	4	451.494	10	9.293
$D_{2\times5\times17,0.25}$	4	965.714	4	663.840	8	14.670
$D_{17\times5\times2,0.25}$	4	857.697	4	645.504	9	16.692
$D_{2\times4\times25,0.25}$	4	1568.099	4	1122.586	9	14.993
$D_{8\times5\times5,0.25}$	4	1571.040	4	1107.324	11	17.227
$D_{2\times5\times22,0.25}$	4	2128.577	4	1540.469	11	16.507
$D_{4\times5\times11,0.25}$	4	2026.727	4	1363.675	13	17.669
$D_{2\times5\times25,0.25}$	4	3185.786	4	2247.128	12	30.028
$D_{10\times5\times5,0.25}$	4	2706.984	4	1904.681	10	29.128
$D_{3\times3\times30,0.25}$	4	3945.495	4	2665.117	13	31.487
$D_{3\times9\times10,0.25}$	4	3982.463	4	2670.117	12	34.956
$D_{2\times6\times25,0.25}$	4	5786.360	4	4087.751	11	48.312
$D_{4\times15\times5,0.25}$	4	4942.943	4	3561.361	15	48.025

Podemos observar, de la Tabla 6.2, que para cada ejemplar $D_{r\times s,\rho}$ y $D_{r\times s\times t,\rho}$ considerados, el número de colores de la mejor q-coloración válida encontrada tanto por **ANTCOL** como por **D-ANTCOL** nunca excede la mejor cota conocida del número cromático del ejemplar; es decir, $q \leqslant p$. Mientras que el número de colores de la mejor q-coloración válida construida por **ABAC** siempre es menor que $4p$; esto es, $q < 4p$. En todos los casos, el tiempo de ejecución es directamente proporcional al número de vértices. Sin embargo, los desempeños computacionales de **ANTCOL** y **D-ANTCOL** comparados con el de **ABAC**, son ineficientes; pues en general, el tiempo de ejecución de **ABAC** corresponde a lo sumo al 1.3 % del tiempo de ejecución de **ANTCOL** y a lo más al 1.7 % del tiempo de ejecución de **D-ANTCOL**.

Notemos el marcado contraste entre la calidad de las coloraciones válidas construidas por las hormigas de **ANTCOL** y **D-ANTCOL**, y lo eficiente que resulta el desempeño computacional de **ABAC**. Por otro lado, aunque los comportamientos de **ANTCOL** y **D-ANTCOL** son muy similares, **D-ANTCOL** es superior. Además, se puede observar que ninguno de estos ejemplares tiene densidad superior a 0.1.

6.5.2. Gráficas bipartitas

La Tabla 6.3 corresponde al desempeño computacional de las tres metaheurísticas sobre el conjunto de ejemplares $B_{r,s,t}$. Bajo este escenario, las metaheurísticas **ANTCOL** y **D-ANTCOL** construyeron, en todos los casos, una coloración óptima; es decir, una 2-coloración válida. Por su parte, el número de colores de la mejor q-coloración válida construida por **ABAC**, en cada caso, es a lo más 3.5 el óptimo; esto es, $q \leqslant 3.5\chi(B_{r,s,t})$. Por otro lado, nuevamente ocurre que tanto el desempeño computacional de **ANTCOL** como el de **D-ANTCOL** es ineficiente comparado con el de **ABAC**. En este caso, tenemos que el tiempo de ejecución de **ABAC** corresponde a lo más al 8.1 % del tiempo de ejecución de **ANTCOL** y a lo sumo al 10.1 % del tiempo de ejecución de **D-ANTCOL**. Puede notarse que este comportamiento por parte de las tres metaheurísticas sobre el conjunto de ejemplares $B_{r,s,t}$, considerados, es similar al escenario de las mallas con diagonales. Más aún, se observa nuevamente que el tiempo de ejecución es directamente proporcional al número de vértice y se tiene gráficas poco densas, pues cada gráfica $B_{r,s,t}$ construida tiene densidad $0.1 < \rho(B_{r,s,t}) < 0.139$. Otra vez, es evidente la superioridad de **D-ANTCOL** sobre **ANTCOL**.

Tabla 6.3: Desempeño de las metaheurísticas II

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$B_{5,7,25}$	2	18.224	2	13.923	5	0.268
$B_{7,11,35}$	2	42.804	2	36.389	5	0.845
$B_{11,13,50}$	2	102.919	2	86.152	6	2.725
$B_{13,17,60}$	2	170.784	2	141.468	6	4.827
$B_{17,19,75}$	2	329.372	2	275.302	7	13.432
$B_{19,23,85}$	2	481.381	2	362.527	7	27.310
$B_{23,29,100}$	2	704.773	2	603.368	6	36.901
$B_{29,31,110}$	2	835.126	2	716.876	6	54.364
$B_{31,37,125}$	2	1197.671	2	1008.846	7	87.629
$B_{37,41,135}$	2	1505.320	2	1214.277	6	122.324
$B_{41,43,150}$	2	2167.775	2	1825.859	7	172.823

6.5.3. Gráficas k-partitas

Cada ejemplar del conjunto de gráficas k-partitas completas de tamaño máximo construidas, tiene una alta densidad, a saber, $0.860 \leqslant \rho(K_{max}(n,k)) \leqslant 0.998$. Además, sabemos que el número cromático de este tipo de gráficas es igual a k. Los resultados mostrados en la Tabla 6.4 pertenecen a este conjunto de ejemplares altamente densos. De manera análoga al escenario anterior, las metaheurísticas **ANTCOL** y **D-ANTCOL** construyeron una coloración óptima en todos los casos, o sea, una k-coloración válida. En el caso de la metaheurística **ABAC**, el número de colores de la mejor q-coloración válida construida, para cada ejemplar, está acotada de la siguiente manera: $k \leqslant q \leqslant k+1$. Esto significa que **ABAC** mejoró notablemente la calidad de las coloraciones válidas generadas por su colonia de hormigas en comparación con los dos escenarios anteriores. No obstante, este incremento en la calidad, implicó sacrificar tiempo

de ejecución; pues tanto **ANTCOL** como **D-ANTCOL** tienen mejores tiempos de ejecución que **ABAC**, excepto para el ejemplar $K_{max}(50, 5)$. Por otro lado, es evidente, de nuevo, el comportamiento sobresaliente de la metaheurística **D-ANTCOL**.

Tabla 6.4: Desempeño de las metaheurísticas III

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$K_{max}(50, 5)$	5	7.190	5	5.371	5	1.717
$K_{max}(50, 47)$	47	1.814	47	1.772	47	3.682
$K_{max}(70, 7)$	7	11.980	7	9.912	8	13.170
$K_{max}(70, 43)$	43	5.517	43	5.272	44	13.815
$K_{max}(100, 11)$	11	25.778	11	24.119	11	30.093
$K_{max}(100, 41)$	41	19.488	41	17.646	42	44.754
$K_{max}(120, 13)$	13	42.431	13	42.261	14	59.576
$K_{max}(120, 37)$	37	34.278	37	31.985	37	80.596
$K_{max}(150, 17)$	17	90.353	17	80.204	18	159.947
$K_{max}(150, 31)$	31	73.218	31	93.697	31	182.653
$K_{max}(170, 19)$	19	115.606	19	103.054	19	273.453
$K_{max}(170, 23)$	23	116.483	23	113.734	24	289.904
$K_{max}(200, 23)$	23	269.093	23	248.485	24	647.570
$K_{max}(200, 19)$	19	236.236	19	223.954	20	671.535
$K_{max}(220, 31)$	31	273.949	31	255.958	32	913.309
$K_{max}(220, 17)$	17	258.893	17	226.755	18	745.117
$K_{max}(250, 37)$	37	404.865	37	389.444	38	1435.719
$K_{max}(250, 13)$	13	454.361	13	404.513	14	1747.516
$K_{max}(270, 41)$	41	490.054	41	450.571	42	2041.441
$K_{max}(270, 11)$	11	573.600	11	512.611	12	1795.856
$K_{max}(300, 43)$	43	754.043	43	659.574	44	3049.174
$K_{max}(300, 7)$	7	1499.309	7	1256.147	7	2467.395

6.5.4. Gráficas aleatorias

Hasta este punto hemos analizado el comportamiento de **ANTCOL**, **D-ANTCOL** y **ABAC** en ejemplares que tienen una estructura regular. Ahora, analicemos el comportamiento de las tres metaheurísticas sobre el conjunto de ejemplares $G(n, k)$ construidos, donde es altamente probable tener vértices aislados y una estructura irregular.

La densidad de cada ejemplar del conjunto de gráficas $G_{n,\rho}$ generadas para los experimentos, está acotada por abajo por 0.287 y por arriba por 0.911. Los resultados de los experimentos realizados sobre este conjunto de ejemplares se muestran en la Tabla 6.5. Para cada gráfica, **ANTCOL** mejoró, en promedio, la mejor cota conocida p[11] un 68.2 %; **D-ANTCOL** la mejoró, en promedio, un 70.1 %; y **ABAC** la mejoró, en promedio, un 69.9 %. Por otra parte, aunque el

[11] En este caso, el valor de p corresponde a $\Delta(G_i(n, k)) + 1$.

tiempo de ejecución de las tres metaheurísticas crece en relación al número de vértices, para toda $n \in \{50, 70, 100, 120, 150, 170, 200, 220, 250, 270, 300\}$, los tiempos de ejecución de **ANTCOL** y **D-ANTCOL** sobre el conjunto $\{G_1(n, 0.3), G_2(n, 0.5), G_3(n, 0.7), G_4(n, 0.9)\}$ son inversamente proporcionales a la densidad de los elementos; mientras que el tiempo de ejecución de **ABAC** es directamente proporcional. Esto significa, una mejora continua de las complejidades de las metaheurísticas **ANTCOL** y **D-ANTCOL**; y el empeoramiento de la complejidad de **ABAC**. Sin embargo, el número de colores de la mejor q-coloración válida construida por **ABAC** siempre es menor que el número de colores de la mejor q-coloración válida construida por **ANTCOL** y siempre es menor o igual que número de colores de la mejor q-coloración válida construida por **D-ANTCOL**, para todo ejemplar $G_i(n, k)$ tal que $n \geqslant 170$.

Tabla 6.5: Desempeño de las metaheurísticas IV

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$G_1(50, 0.3)$	7	10.332	7	8.704	8	0.641
$G_2(50, 0.5)$	10	8.356	10	6.468	11	1.129
$G_3(50, 0.7)$	15	5.584	15	4.932	16	1.801
$G_4(50, 0.9)$	22	4.062	22	3.951	23	1.783
$G_1(70, 0.3)$	9	26.915	9	22.353	10	3.532
$G_2(70, 0.5)$	14	17.824	13	14.837	14	5.760
$G_3(70, 0.7)$	20	14.252	19	12.650	21	7.645
$G_4(70, 0.9)$	30	11.234	28	9.179	30	10.602
$G_1(100, 0.3)$	12	63.191	11	52.125	12	11.084
$G_2(100, 0.5)$	19	51.963	17	42.722	18	25.159
$G_3(100, 0.7)$	26	36.126	24	40.752	26	34.758
$G_4(100, 0.9)$	41	25.619	37	22.355	37	30.292
$G_1(120, 0.3)$	14	98.187	13	82.647	16	23.801
$G_2(120, 0.5)$	21	80.151	20	69.010	21	47.452
$G_3(120, 0.7)$	31	64.286	28	50.583	29	58.106
$G_4(120, 0.9)$	48	34.418	45	32.724	45	93.487
$G_1(150, 0.3)$	16	173.619	15	143.350	16	57.923
$G_2(150, 0.5)$	25	144.093	23	119.734	24	92.235
$G_3(150, 0.7)$	37	103.365	34	90.959	35	144.421
$G_4(150, 0.9)$	59	65.722	54	63.485	54	205.339
$G_1(170, 0.3)$	18	309.688	17	256.486	17	93.089
$G_2(170, 0.5)$	28	221.524	26	196.955	26	162.189
$G_3(170, 0.7)$	42	194.816	37	179.104	36	305.767
$G_4(170, 0.9)$	66	147.733	62	135.951	61	453.876
$G_1(200, 0.3)$	21	528.981	19	372.191	19	142.990
$G_2(200, 0.5)$	32	409.209	30	301.432	28	317.602
$G_3(200, 0.7)$	47	251.511	43	211.777	42	467.007

Tabla 6.5 (Continuación)

Ejemplar	**ANTCOL**		**D-ANTCOL**		**ABAC**	
	q	t (seg.)	q	t (seg.)	q	t (seg.)
$G_4(200, 0.9)$	76	158.287	68	147.959	67	637.374
$G_1(220, 0.3)$	22	535.624	21	433.963	21	236.359
$G_2(220, 0.5)$	35	481.087	32	378.996	30	447.924
$G_3(220, 0.7)$	52	385.278	47	302.838	45	751.582
$G_4(220, 0.9)$	82	283.642	76	262.288	74	1019.933
$G_1(250, 0.3)$	24	734.884	23	691.795	21	452.765
$G_2(250, 0.5)$	38	682.628	36	621.346	33	1100.105
$G_3(250, 0.7)$	57	575.908	52	552.357	50	1369.663
$G_4(250, 0.9)$	90	339.53	86	301.239	83	1571.800
$G_1(270, 0.3)$	26	917.002	25	748.179	23	531.431
$G_2(270, 0.5)$	42	830.085	39	758.54	36	996.937
$G_3(270, 0.7)$	63	575.052	58	552.630	55	1803.897
$G_4(270, 0.9)$	104	529.856	96	445.921	93	2530.265
$G_1(300, 0.3)$	29	1235.937	27	1045.028	25	810.982
$G_2(300, 0.5)$	47	1346.789	45	1141.012	43	1907.870
$G_3(300, 0.7)$	68	842.113	63	748.380	58	2309.632
$G_4(300, 0.9)$	110	642.544	103	529.180	100	3552.869

Conclusiones

A pesar de las impresionantes capacidades de las computadoras actuales, todavía existen muchos problemas de optimización combinatoria que requieren un enfoque heurístico para atacar el caso general. El Problema de la Coloración de Gráficas es uno de estos problemas.

En este trabajo mostramos cómo la forma en que las hormigas coordinan sus actividades de búsqueda de alimento se puede utilizar para abordar la coloración de gráficas. Nuestra experiencia con las tres metaheurísticas descritas en este trabajo nos lleva a los siguientes comentarios generales:

1. Los resultados obtenidos de las metaheurísticas **ANTCOL** y **D-ANTCOL** son uniformes; es decir, en todos los casos, el número de colores de la mejor coloración construida por la colonia nunca excedió la mejor cota conocida del número cromático correspondiente. Por otro lado, aunque los resultados obtenidos de la metaheurística **ABAC** son irregulares, son estables, esto es: tanto para el conjunto de las mallas con diagonales como para el conjunto de gráficas bipartitas, el número de colores de la mejor coloración construida por las hormigas es a lo más cuatro veces la mejor cota conocida del número cromático correspondiente; para el conjunto de gráficas k- partitas de tamaño máximo, el número de colores de la mejor coloración construida por la colonia es a lo más el valor del número cromático correspondiente más una unidad; y para el conjunto de gráficas aleatorias, el número de colores de la mejor coloración construida por la colonia nunca excedió la mejor cota conocida del número cromático correspondiente.

2. Cada hormiga de la colonia utilizada por la metaheurística **ANTCOL**, así como la utilizada por la metaheurística **D-ANTCOL**, es un agente simple en el sentido de que no necesita ir descubriendo su entorno, pues conoce toda la información de la gráfica de entrada desde el inicio. Aunque esencialmente el mecanismo usado por cualquier hormiga de **ANTCOL** y **D-ANTCOL** consiste en elegir un vértice no coloreado con el mayor grado para agregarlo a la clase de color actual y después descartar los vecinos de este vértice para repetir este proceso en la gráfica generada por los vértices no coloreados restantes, este mecanismo no es estático. Las feromonas y la deseabilidad con las que **ANTCOL** y **D-ANTCOL** dotan a sus hormigas, permiten elegir no sólo un vértice sin colorear de mayor grado, sino aquel que permita a las hormigas agregar la mayor cantidad de vértices sin colorear a la clase de color actual, logrando así una alta calidad en la coloración válida final construida por la colonia. Mientras que **ANTCOL** tiene una única manera de usar el rastro de feromonas y la deseabilidad para que la hormiga actual seleccione y agregue a la clase de color actual el primer vértice, **D-ANTCOL** tiene tres opciones distintas que elige de manera aleatoria. Lo que resulta en una mayor diversificación en la búsqueda por parte de la colonia de hormigas de **D-ANTCOL**. Esto es consistente con los resultados obtenidos, en donde en todo los casos, **D-ANTCOL** es

superior a **ANTCOL**.

A pesar de estas cualidades y ventajas dadas por las feromonas y la deseabilidad a las hormigas de **ANTCOL** y **D-ANTCOL**, su poder no es absoluto, pues cuando la gráfica de entrada es poco densa, el número de vértices no coloreados que pueden ser agregados a la clase de color actual es muy grande. Lo que implica un incremento notable en el número de iteraciones que debe hacer cada hormiga cuando sea su turno de construir una coloración válida para la gráfica de entrada. Lo que implica un tiempo de ejecución altamente ineficiente. De aquí que se haya observado en los resultados obtenido que tanto el tiempo de ejecución de **ANTCOL** como el de **D-ANTCOL** son inversamente proporcionales a la densidad de la gráfica de entrada.

Otra desventaja que se observa de los resultados obtenidos sobre **ANTCOL** y **D-ANTCOL** ocurre cuando la gráfica de entrada es aleatoria, ya que por construcción es altamente probable que existan vértices aislados que serán coloreados con el mismo color y al ir eliminado vértices en el proceso de coloración llevado a cabo por la colonia, la gráfica generada resultante tiende a tener una estructura "regular" que hace que **ANTCOL** y **D-ANTCOL** alcancen el mejor resultado obtenido mucho antes de terminar su ejecución.

3. Cada una de las hormigas distribuidas por **ABAC** en los vértices de la gráfica de entrada es un agente con la capacidad de ir descubriendo su entorno a través de los vecinos de su vértice actual, construir con alta probabilidad una coloración válida para la región descubierta y memorizar, a corto plazo, cada uno de los vértices que ha visitado; pues el objetivo de la colonia de hormigas de **ABAC** es generar una coloración válida, con un menor número de colores, a partir de una coloración válida inicial, usando búsqueda local. Aunque estas cualidades de las hormigas las convierten en agentes complejos, se observa un comportamiento contrastante. Cuando la gráfica de entrada tiene una densidad muy baja, el número de vecinos del vértice actual de cada hormiga es bastante pequeño. Lo que implica un tiempo de ejecución altamente eficiente. Sin embargo, la extensión de la región de la gráfica de entrada descubierta será también pequeña. Lo que provocará que en cierto punto de la ejecución, las hormigas visiten una y otra vez los mismos vértices sin ninguna mejora. Lo que implica una mala calidad de la coloración final producida por la colonia. Esto es consistente con los resultados obtenidos, donde se observa una relación de proporcionalidad directa entre la calidad de la coloración válida producida por la colonia de **ABAC** y la densidad de la gráfica de entrada, y una relación de proporcionalidad inversa entre la densidad de la gráfica de entrada y el tiempo de ejecución de **ABAC**.

   Por otro lado, este comportamiento distribuido de las hormigas de **ABAC**, le permiten a la colonia encontrar peores soluciones que le permitan salir de óptimos locales, sobre todo cuando la gráfica de entrada es altamente densa. De aquí que **ABAC** produzca mejores coloraciones válidas para gráficas aleatorias que **ANTCOL** y **D-ANTCOL**.

4. El análisis y comparación de las tres metaheurísticas muestran que el poder de las feromonas está supeditado a la regularidad y densidad de la gráfica de entrada, y que no usarlas tampoco trae enormes desventajas. Por lo que pensar en una estrategia que se enfoque en revisar la densidad y la estructura de las aristas de la gráfica de entrada en primera instancia, para a continuación decidir cuál de estas metaheurísticas aplicar, permitiría explotar sus ventajas y minimizar las desventajas de aplicar una de éstas directamente.

Apéndice A

Teoría de Gráficas

En el presente apéndice se exponen las definiciones básicas sobre la Teoría de Gráficas, así como de los términos, resultados y notación usados en los capítulos anteriores.

A.1. Conceptos básicos

Definición A.1 Una **gráfica no dirigida** G es un conjunto no vacío V de objetos llamados **vértices** junto con un conjunto E de subconjuntos de cardinalidad 2 de V llamados **aristas**. Cada arista $\{u, v\} \in E$ es denotada como uv o vu. Si $e = uv$, decimos que e une a u y v. En una gráfica no dirigida G, el número de vértices es el **orden** de G y el número de aristas es el **tamaño** de G. Para indicar que una gráfica no dirigida G tiene conjunto de vértices V y conjunto de aristas E, escribimos $G = (V, E)$. Para enfatizar que V es el conjunto de vértices de una gráfica no dirigida G, escribimos $G.V$. Análogamente, para el conjunto de aristas escribimos $G.E$. En este documento sólo se consideran conjuntos finitos tanto de vértices como de aristas.

Definición A.2 Sea $G = (V, E)$ una gráfica no dirigida. Decimos que una gráfica no dirigida $H = (W, F)$ es una **subgráfica** de G si y solo si $H.W \subseteq G.V$ y $H.F \subseteq G.E$.

Definición A.3 Sea $G = (V, E)$ una gráfica no dirigida y S un subconjunto no vacío de $G.V$. La **gráfica inducida** por S, denotada por $G[S]$, es la subgráfica de G que tiene a S como conjunto de vértices y para cualesquiera $u, v \in S$, uv es una arista de $G[S]$ si y solo si uv es una arista de G.

Definición A.4 Sea $G = (V, E)$ una gráfica no dirigida. Si existen $u, v \in V$ tales que están unidos por un número finito de aristas, decimos que G tiene una **multiarista**.

Definición A.5 Sea $G = (V, E)$ una gráfica no dirigida. Si existe $v \in V$ tal que $vv \in E$; es decir, si existe una arista en E que una a un vértice consigo mismo, decimos que G tiene una **lazo**.

Definición A.6 Sea $G = (V, E)$ una gráfica no dirigida. Decimos que G es una **gráfica simple** si y sólo si G no tiene lazos ni multiaristas.

Definición A.7 Sea $G = (V, E)$ una gráfica. Si $e = uv$ es una arista de G, diremos que u y v son **vértices adyacentes** y que e **incide** en u y en v. Dos vértices adyacentes se denominan **vecinos** uno del otro.

Definición A.8 Sean $G = (V, E)$ una gráfica y v un vértice de G. El conjunto $\{\ u \in G.V \mid u$ es vecino de $v\ \}$ es llamado **vecindad** de v en G y se denota por $Adj_G(v)$.

Definición A.9 Sean $G = (V, E)$ una gráfica y v un vértice de G. El **grado** de v en G es el número de vértices en $Adj_G(v)$ y se denota por $deg_G(v)$.

Definición A.10 Sea $G = (V, E)$ una gráfica. El **grado mínimo** de G, denotado por $\delta(G)$, se define como:

$$\delta(G) = \text{mín}\{\ deg_G(v) \mid v \in G.V\ \}$$

Definición A.11 Sea $G = (V, E)$ una gráfica. El **grado máximo** de G, denotado por $\Delta(G)$, se define como:

$$\Delta(G) = \text{máx}\{\ deg_G(v) \mid v \in G.V\ \}$$

Definición A.12 Sea $G = (V, E)$ una gráfica. Un **camino** en G es una sucesión de sus vértices y aristas $C = (v_0, e_0, v_1, e_1, \ldots, e_{n-1}, v_n)$, donde para cada $i \in \{1, 2, \ldots, n-1\}$, $e_i = v_i v_{i+1}$.

Definición A.13 Sean $G = (V, E)$ una gráfica y u y v dos vértices de G. Un ***uv*-camino** es un camino en G de u a v.

Definición A.14 Sean $G = (V, E)$ una gráfica y C un uv-camino en G. La **longitud** de C en G es igual al número de aristas que posee tal camino y se denota por $l_G(C)$.

Definición A.15 Sea $G = (V, E)$ una gráfica. Diremos que G es **conexa** si y solo si para cualquier par de vértices $u, v \in G.V$, existe un uv-camino en G.

Definición A.16 Sea $G = (V, E)$ una gráfica de orden n. Diremos que G es **completa** si y solo si para todo par de vértices $u, v \in G.V$, con $u \neq v$, $uv \in G.E$. Tal gráfica se denota por K_n y su tamaño es $\binom{n}{2} = \frac{n(n-1)}{2}$.

Definición A.17 Sea $G = (V, E)$ una gráfica. El **complemento** de G, denotado por G^c, es la gráfica tal que: $G^c.V = G.V$ y para todo par de vértices $u, v \in G^c.V$, con $u \neq v$, uv es una arista de G^c si y solo si uv no es una arista de G.

Definición A.18 Sea $G = (V, E)$ una gráfica de orden $n \geqslant 2$ y sea k un entero positivo tal que $2 \leqslant k \leqslant n$. Decimos que G es ***k*-partita** si y sólo si existe una partición de $G.V$ en k conjuntos $V_1, V_2, \ldots, V_k$, con $|V_i| = n_i$, tal que para toda $e \in G.E$ existen $i, j \in \{1, 2, \ldots, k\}$, con $i \neq j$, tales que e une un vértice de V_i y un vértice de V_j. Cada conjunto V_i de la partición es llamado **conjunto partito** y $\sum_{i=1}^{k} n_i = n$. Además, si $k = 2$, decimos que G es una gráfica **bipartita**.

Definición A.19 Sea $G = (V, E)$ una gráfica k-partita. Decimos que G es ***k*-partita completa**, y escribimos $K_{n_1, n_2, \ldots, n_k}$, si y sólo si para todas $i, j \in \{1, 2, \ldots, k\}$, con $i \neq j$, cada vértice de V_i es adyacente a todo vértice de V_j. El tamaño de $K_{n_1, n_2, \ldots, n_k}$ es igual a $\frac{n(n-1)}{2} - \sum_{i=1}^{k} \frac{n_i(n_i-1)}{2}$.

Definición A.20 Sea $G = (V, E)$ una gráfica de orden n y r un entero tal que $0 \leqslant r \leqslant n-1$. Decimos que G es ***r*-regular** si y solo si $deg_G(v) = r$, para todo $v \in G.V$.

Definición A.21 Sea $G = (V, E)$ una gráfica de orden $n \geqslant 4$. Decimos que G es una **malla bidimensional** si y solo si existen enteros $r, s \geqslant 2$ tales que $n = r \times s$ y G tiene la siguiente estructura:

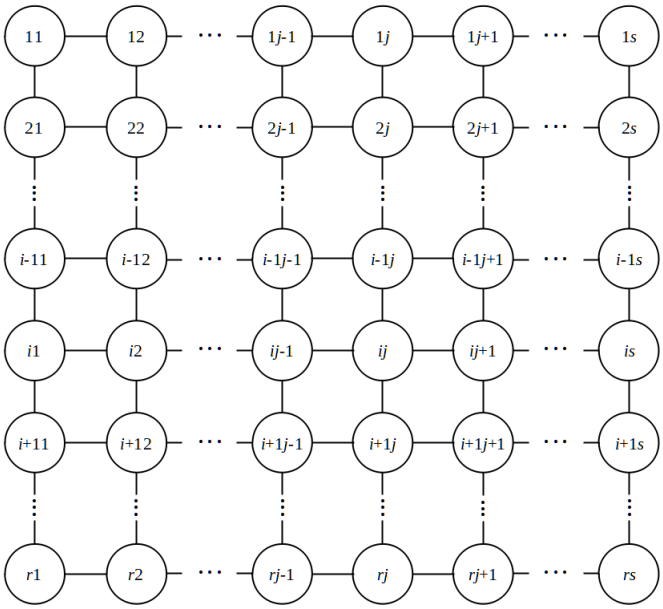

Figura A.1: Estructura de una malla bidimensional: etiquetado de los vértices.

Definición A.22 Sea $G = (V, E)$ una gráfica de orden $n \geqslant 8$. Decimos que G es una **malla tridimensional** si y solo si existen enteros $r, s, t \geqslant 2$ tales que $n = r \times s \times t$ y G está conformada por t mallas bidimensionales de orden $r \times s$ tales que para toda $k \in \{1, 2, \ldots, t\}$, el vértice ijk es adyacente al vértice $ijk + 1$, para toda $i \in \{1, 2, \ldots, r\}$ y para toda $j \in \{1, 2, \ldots, s\}$. La figura A.2 muestra la estructura de una malla tridimensional.

Definición A.23 Sean $G = (V, E)$ una gráfica y W un subconjunto no vacío de $G.V$. Decimos que W es un **conjunto estable** si y sólo si sus elementos son no adyacentes por pares.

Definición A.24 Sean $G = (V, E)$ una gráfica y W un subconjunto no vacío de $G.V$. Decimos que W es un **conjunto independiente** si y solo si para todos $u, v \in W$, $uv \notin G.E$. El máximo $l \in \mathbb{Z}^+$ para el cuál G tiene un conjunto independiente de cardinalidad l es llamado **número de independencia** de G y se denota por $\alpha(G)$.

Definición A.25 Sea $G = (V, E)$ una gráfica y $H = (W, F)$ una subgráfica de G. Decimos que H es un **clan** de G si y solo si H es por sí misma una gráfica completa. El máximo $l \in \mathbb{Z}^+$ para el cuál G tiene un clan de orden l es llamado **número de clan** de G y se denota por $\omega(G)$.

Definición A.26 Sea $G = (V, E)$ una gráfica de orden n y tamaño m. La **densidad** de G, denotado por $\rho(G)$, se define como el cociente entre el número de aristas de G y el número de aristas de la gráfica completa K_n, esto es:

$$\rho(G) = \frac{m}{\frac{n(n-1)}{2}} = \frac{2m}{n(n-1)}.$$

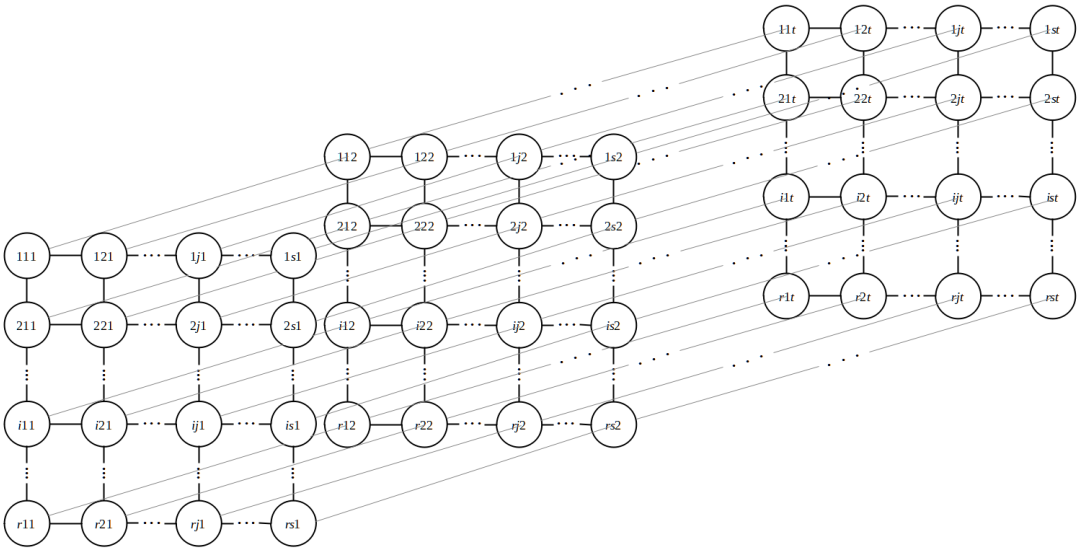

Figura A.2: Estructura de una malla tridimensional: etiquetado de los vértices.

A.2. Relevancia de las gráficas *k*-partitas

Sea $G = (V, E)$ una gráfica de orden $n \geqslant 2$. Si G es bipartita, sabemos que existe una partición de $G.V$ en dos conjuntos partitos V_1 y V_2 tales que cada una de las aristas de $G.E$ une un vértice de V_1 con un vértice de V_2. Sean n_1 y n_2 las cardinalidades de V_1 y V_2 respectivamente, entonces $n_1, n_2 \geqslant 1$ y $n = n_1 + n_2$. Por tanto, una gráfica bipartita completa tiene un número de aristas igual a $n_1(n-n_1)$, donde $1 \leqslant n_1 \leqslant n-1$. De manera que, el valor de n_1 que máximiza $n_1(n - n_1)$ es: $n_1 = n/2$, si n es par; y $n_1 = (n-1)/2$, si n es impar. Por lo tanto, si se define $m_{max}(n, 2)$ como el número máximo de aristas que puede tener una gráfica bipartita de orden $n \geqslant 2$, se tiene que:

$$m_{max}(n,2) = \begin{cases} \left(\frac{n}{2}\right)^2 & \text{si } n \text{ es par} \\ \left(\frac{n-1}{2}\right)\left(\frac{n+1}{2}\right) & \text{si } n \text{ es impar} \end{cases} \tag{A.1}$$

Por otro lado, dado que

$$\left(\frac{n-1}{2}\right)\left(\frac{n+1}{2}\right) < \left(\frac{n}{2}\right)^2,$$

se deduce el siguiente:

Resultado A.1 Toda gráfica $G = (V, E)$ de orden $n \geqslant 2$ y tamaño $m > (n/2)^2$ no es bipartita.

El resultado anterior puede traducirse en términos de la densidad de G. Según la definición de $\rho(G)$, el tamaño de G puede escribirse como:

$$m = \rho(G)\frac{n(n-1)}{2} \tag{A.2}$$

De modo que, toda gráfica $G = (V, E)$ de orden $n \geqslant 2$ y densidad $\rho(G)$ tales que

$$\rho(G)\frac{n(n-1)}{2} > \left(\frac{n}{2}\right)^2$$

no es bipartita. Por tanto, si despejamos $\rho(G)$, obtenemos el siguiente:

Resultado A.2 Toda gráfica $G = (V, E)$ de orden $n \geqslant 2$ y densidad $\rho(G)$ tales que

$$\rho(G) > \frac{n}{2(n-1)}$$

no es bipartita.

Ahora, generalicemos la ecuación A.1 para gráficas k-partitas. Definamos $m_{max}(n, k)$ como el número máximo de aristas que puede tener una gráfica k-partita de orden $n \geqslant 2$. Consideremos así una gráfica $G = (V, E)$ k-partita de orden $n \geqslant 2$. Entonces existe una partición de $G.V$ en k conjuntos partitos $V_1, V_2, \ldots, V_k$, con $|V_i| = n_i$, tal que para toda arista $e \in G.E$ existen $i, j \in \{1, 2, \ldots, k\}$, con $i \neq j$, tales que e une un vértice de V_i y un vértice de V_j. Además, sabemos que

$$\sum_{i=1}^{k} n_i = n$$

y que para toda $i \in \{1, 2, \ldots, k\}$, $n_i \geqslant 1$.

Notemos que si G es k-partita completa, contiene las $n(n-1)/2$ aristas de la gráfica K_n menos todas las arístas correspondientes a vértices de un mismo conjunto partito V_i, para toda $i \in \{1, 2, \ldots, k\}$. De esta forma, el número total de aristas de una gráfica k-partita completa de orden $n \geqslant 2$ es:

$$\frac{n(n-1)}{2} - \sum_{i=1}^{k} \frac{n_i(n_i-1)}{2}.$$

A continuación, debemos determinar los valores de $n_1, n_2, \ldots, n_k$ que maximizan esta función. Dado que el primer término de la función solo depende de n, minimizaremos el segundo término. Lo que da como resultado el siguiente problema de programación lineal:

$$\begin{aligned}
\text{Minimizar} \quad & \sum_{i=1}^{k} \frac{n_i(n_i-1)}{2} & \\
\text{Sujeto a} \quad & \sum_{i=1}^{k} n_i = n & \\
& n_i \in \mathbb{Z} & i = 1, 2, \ldots, k \\
& n_i \geqslant 1 & i = 1, 2, \ldots, k
\end{aligned}$$

Es fácil verificar que la partición $\{V_1, V_2, \ldots, V_k\}$ que optimiza el programa anterior debe ser tal que todos los conjuntos partitos V_i tengan el mismo orden o a lo sumo órdenes separados por una unidad. De esta forma, para cada $i \in \{1, 2, \ldots, k\}$, el valor óptimo de n_i es:

$$n_i = \begin{cases} \left\lfloor \frac{n}{k} \right\rfloor & \text{si } 1 \leqslant i \leqslant k - (n \bmod k) \\ \left\lfloor \frac{n}{k} \right\rfloor + 1 & \text{si } k + 1 - (n \bmod k) \leqslant i \leqslant k \end{cases}$$

Ejemplo A.1 Si quisieramos definir la gráfica $G = (V, E)$ 6-partita de orden 100 y tamaño máximo, deberiamos particionar $G.V$ en dos conjuntos partitos de cardinalidad 16 y cuatro conjuntos partitos de cardinalidad 17; pues en este caso tenemos que: $\lfloor n/k \rfloor = \lfloor 100/6 \rfloor = 16$ y $n \bmod k = 100 \bmod 6 = 4$. Además, esta es la únca partición de $G.V$ tal que ningún par de conjuntos partitos tienen cardinalidades de diferencia mayor a uno.

Una vez conocida la partición óptima de $G.V$, puede calacularse el valor de $m_{max}(n, k)$. Con el objetivo de simplificar las operaciones hacemos $q = \lfloor n/k \rfloor$ y $r = n \bmod k$. Asi, tenemos:

$$\begin{aligned}
m_{max}(n,k) &= \frac{n(n-1)}{2} - \sum_{i=1}^{k} \frac{n_i(n_i-1)}{2}\bigg|_{n_i \ opt} \\
&= \frac{n(n-1)}{2} - \frac{1}{2}\left(\sum_{i=1}^{k-r} q(q-1) + \sum_{i=k-r+1}^{k} q(q+1)\right) \\
&= \frac{n(n-1)}{2} - \frac{q}{2}\left(\sum_{i=1}^{k-r} (q-1) + \sum_{i=k-r+1}^{k} (q+1)\right) \\
&= \frac{n(n-1)}{2} - \frac{q}{2}\left((k-r)(q-1) + r(q+1)\right) \\
&= \frac{n(n-1)}{2} - \frac{q}{2}\left(kq - k - rq + r + rq + r\right) \\
&= \frac{n(n-1)}{2} - \frac{q}{2}\left(kq - k + 2r\right) \\
&= \frac{n(n-1)}{2} - \frac{q}{2}\left(k(q-1) + 2r\right).
\end{aligned}$$

Finalmente, considerando los valores de q y r, se obtiene la expresión:

$$m_{max}(n,k) = \frac{n(n-1)}{2} - \frac{1}{2}\left\lfloor \frac{n}{k} \right\rfloor \left(k\left(\left\lfloor \frac{n}{k} \right\rfloor - 1\right) + 2(n \bmod k)\right) \tag{A.3}$$

Además, se puede observar que si sustituimos en esta ecuación $k = 2$, obtenemos los valores calculados en la ecuación A.1.

El análisis anterior nos permite establecer los siguientes teorema y corolario:

Teorema A.1 Para cualquier gráfica $G = (V, E)$ de orden $n \geqslant 2$ y tamaño m, si existe un entero positivo $k \geqslant 2$ tal que m cumple la desigualdad

$$m > m_{max}(n, k)$$

entonces G no es k-partita.

Corolario A.1 Toda gráfica $G = (V, E)$ k-partita de orden $n \geqslant 2$ y tamaño m tal que

$$m = m_{max}(n, k)$$

es una gráfica k-partita completa. Más aun, dada cualquier partición de G en k conjuntos estables, ningún conjunto partito puede superar en más de una unidad la cardinalidad de otro conjunto partito.

Los corolarios siguientes son consecuencia del Teorema 2.5 y sus corolarios, así como de los dos resultados anteriores.

Corolario A.2 Para cualquier gráfica $G = (V, E)$ de orden $n \geqslant 2$ y tamaño m, si existe un entero positivo $k \geqslant 2$ tal que m cumple la desigualdad

$$m > m_{max}(n, k)$$

entonces $\chi(G) > k$, es decir, G no es k-coloreable.

Corolario A.3 Toda gráfica $G = (V, E)$ k-partita de orden $n \geqslant 2$ y tamaño m tal que

$$m = m_{max}(n, k)$$

tiene número cromático $\chi(G) = k$. Más aun, dada cualquier k-coloración de G, ninguna clase de color puede superar en más de una unidad la cardinalidad de otra clase de color.

Ejemplo A.2 Sea $G = (V, E)$ una gráfica de orden 100 tal que quiere colorearse con un número de colores igual a 5. Dado que $m_{max}(100, 5) = 4000$, si el tamaño de G es mayor a 4000, entonces G no es 5-coloreable.

De manera análoga al Resultado A.2, el Colorario A.2 puede expresarse en términos de la densidad de G. De modo que se pueda obtener el valor de $\rho(G)$ a partir del cual G no puede ser k-coloreable. Para lograr esto, sutituyamos la ecuación A.2 en la desigualdad del Colorario A.2:

$$\rho(G)\frac{n(n-1)}{2} > \frac{n(n-1)}{2} - \frac{1}{2}\left\lfloor \frac{n}{k} \right\rfloor \left(k\left(\left\lfloor \frac{n}{k} \right\rfloor - 1\right) + 2(n \bmod k)\right).$$

Lo que implica,

$$\left\lfloor \frac{n}{k} \right\rfloor \left(k\left(\left\lfloor \frac{n}{k} \right\rfloor - 1\right) + 2(n \bmod k)\right) > n(n-1)(1-\rho(G)) \tag{A.4}$$

Por otro lado, dado que

$$\frac{n}{k} - 1 < \left\lfloor \frac{n}{k} \right\rfloor \leqslant \frac{n}{k}$$

y

$$0 \leqslant n \bmod k \leqslant k - 1,$$

entonces

$$\left\lfloor \frac{n}{k} \right\rfloor \left(k\left(\left\lfloor \frac{n}{k} \right\rfloor - 1\right) + 2(n \bmod k)\right) > \left(\frac{n}{k} - 1\right) k \left(\frac{n}{k} - 2\right) \tag{A.5}$$

Luego, de las desigualdades A.4 y A.5, se tiene que

$$\left(\frac{n}{k} - 1\right) k \left(\frac{n}{k} - 2\right) \geqslant n(n-1)(1-\rho(G)).$$

De esta manera, al retomar lo dicho por el Corolario A.2, tendremos que si G satisface la desigualdad

$$\left(\frac{n}{k} - 1\right) k \left(\frac{n}{k} - 2\right) \geqslant n(n-1)(1-\rho(G)),$$

entonces tendrá un número cromático superior a k. Por lo tanto, despejando $\rho(G)$ se obtiene el siguiente corolario:

Corolario A.4 Para toda gráfica $G = (V, E)$ de orden $n \geqslant 2$, si existe un entero positivo $k \geqslant 2$ tal que

$$\rho(G) \geqslant 1 - \frac{k}{n(n-1)} \left(\frac{n}{k} - 1\right) \left(\frac{n}{k} - 2\right)$$

entonces $\chi(G) > k$, es decir, G no es k-coloreable.

Ejemplo A.3 Toda gráfica $G = (V, E)$ de orden 200 y densidad superior a 0.68 tiene número cromático superior a $k = 3$.

A.3. Notación para gráficas

A continuación, se presenta una tabla con la notación que usada durante el desarrollo de esta tesis.

Tabla A.1: Notación para graficas

Notación	**Significado**
$G.V$	Conjunto de vértices de G
$G.E$	Conjunto de aristas de G
$G[S]$	Subgráfica de G inducida por S
$Adj_G(v)$	Vecindad del vértice v en G
$deg_G(v)$	Grado del vértice v en G
$\delta(G)$	Grado mínimo de G
$\Delta(G)$	Grado máximo de G
$l_G(C)$	Longitud de C en G
K_n	Gráfica completa de orden n
G^c	Complemento de G
$K_{n_1,n_2,\ldots,n_k}$	Gráfica k-partita completa
$\alpha(G)$	Número de independencia de G
$\omega(G)$	Número de clan de G
$\rho(G)$	Densidad de G
$dsat_{\mathscr{C}}(v)$	Grado de saturación de v bajo $\mathscr{C}$
$V(i)$	Clase de color i
$\chi(G)$	Número cromático de G

Apéndice B

Complejidad Computacional

En el presente apéndice se exponen las definiciones básicas sobre Complejidad Computacional.

Definición B.1 Un **problema** es una pregunta general a responder y está descrito por los siguientes componentes:

1. Una descripción de cada uno de sus **parámetros** y las restricciones de éstos.
2. Un enunciado que especifique qué propiedades debe satisfacer la **respuesta** o **solución**.

Definición B.2 Sea Π un problema. Un **ejemplar** de Π se obtiene especificando valores particulares para todos los parámetros de Π. El conjunto de todos los ejemplares de un problema Π se denota D_Π.

Definición B.3 Sea Π un problema. Decimos que Π es un **problema de decisión** si y solo si el conjunto de respuestas de Π es $\{YES, NO\}$. El conjunto de todos los ejemplares de Π para los cuales la solución es YES, se denota por Y_Π.

Definición B.4 Sea Π un problema de decisión. Un **algoritmo no determinista** para Π consiste de dos fases separadas, la primera llamada **fase adivinadora** y la segunda **fase verificadora**. Dado un ejemplar E de Π, la fase adivinadora se encarga de "adivinar" un candidato a solución S para E. A continuación, tanto E como S son pasados como entradas a la fase verificadora, que procede a computar de manera determinista si S es o no solución de E; es decir, determina si E satisface las condiciones del problema Π. En caso afirmativo, el algoritmo termina su ejecución y la salida es YES. En caso contrario, el algoritmo termina su ejecución y la salida es NO, o bien, el algoritmo nunca termina su ejecución.

Definición B.5 Sea $\mathscr{A}$ un algoritmo no determinista. $\mathscr{A}$ es de **tiempo polinomial** si y sólo si tanto su fase adivinadora como su fase verificadora están acotadas por un polinomia con respecto al tamaño de la entrada $E \in D_\Pi$.

Definición B.6 Sean Π_1 y Π_2 dos problemas de decisión. Una **transformación polinomial** del problema Π_1 al problema Π_2 es una función $f : D_{\Pi_1} \to D_{\Pi_2}$ que satisface las siguientes dos condiciones:

1. f es computable por un algortimo de tiempo polinomial.

2. Para todo ejemplar $E \in D_{\Pi_1}$, $E \in Y_{\Pi_1}$ si y solo si $f(E) \in Y_{\Pi_2}$.

Si existe una transformación polinomial de Π_1 a Π_2, escribimos $\Pi_1 \propto_p \Pi_2$ y decimos que Π_1 es polinomialmente reducible a Π_2.

Lema B.1 Sean Π_1, Π_2 y Π_3 tres problemas de decisión. Si $\Pi_1 \propto_p \Pi_2$ y $\Pi_2 \propto_p \Pi_3$, entonces $\Pi_1 \propto_p \Pi_3$.

Demostración.

Sean Π_1, Π_2 y Π_3 tres problemas de decisión.

Supongamos que $\Pi_1 \propto_p \Pi_2$ y $\Pi_2 \propto_p \Pi_3$.

Por demostrar: $\Pi_1 \propto_p \Pi_3$; esto es, existe una transformación polinomial de Π_1 a Π_3.

Como $\Pi_1 \propto_p \Pi_2$, existe una transformación polinomial f_1 de Π_1 a Π_2. Análogamente, existe una transformación polinomial f_2 de Π_2 a Π_3, dado que $\Pi_2 \propto_p \Pi_3$.

Notemos que $f_2 \circ f_1$ es una transformacion polinomial de Π_1 a Π_3. Así, $\Pi_1 \propto_p \Pi_3$.

Ergo, si $\Pi_1 \propto_p \Pi_2$ y $\Pi_2 \propto_p \Pi_3$, entonces $\Pi_1 \propto_p \Pi_3$, para cualesquiera problemas de decisión Π_1, Π_2 y Π_3.

Q. E. D.

Definición B.7 Sea Π un problema de decisión. Π pertenece a la clase de complejidad **P** si y sólo si existe un algoritno determinista de tiempo polinomial que resuelve Π.

Definición B.8 Sea Π un problema de decisión. Π pertenece a la clase de complejidad **NP** si y solo si existe un algoritmo no determinista de tiempo polinomial para Π.

Definición B.9 Sea Π un problema de decisión. Decimos que Π es **NP-Completo** si y sólo si $\Pi \in$ **NP**; y para todo $\Pi' \in$ **NP**, $\Pi' \propto_p \Pi$.

Lema B.2 Sea Π_1 un problema de decisión. Si $\Pi_1 \in$ **NP** y existe un problema de decisión Π_2 **NP-Completo** tal que $\Pi_2 \propto_p \Pi_1$, entonces Π_1 es **NP-Completo**.

Demostración.

Sea Π_1 un problema de decisión.

Supongamos que $\Pi_1 \in$ **NP** y que existe un problema de decisión Π_2 **NP-Completo** tal que $\Pi_2 \propto_p \Pi_1$.

Por demostrar: Π_1 es **NP-Completo**.

$\Pi_1 \in$ **NP** por hipótesis.

Ahora, dado que Π_2 es **NP-Completo**, se tiene que $\Pi' \propto_p \Pi_2$, para todo $\Pi' \in$ **NP**. Lo que implica, para todo $\Pi' \in$ **NP**, $\Pi' \propto_p \Pi_2$ y $\Pi_2 \propto_p \Pi_1$. Entonces, para todo $\Pi' \in$ **NP**, $\Pi' \propto_p \Pi_1$; por Lema B.1.

Por lo tanto, Π_1 es **NP-Completo**.

Ergo, para todo problema de decisión Π_1, si ocurre que $\Pi_1 \in$ **NP** y existe un problema de decisión Π_2 **NP-Completo** tal que $\Pi_2 \propto_p \Pi_1$, entonces Π_1 es **NP-Completo**.

Q. E. D.

Apéndice C

Métodos Constructivos para la Coloración de Gráficas

Dada una gráfica $G = (V, A)$ de orden n, sabemos que no existe, hasta el momento, un algoritmo eficiente que determine $\chi(G)$. Sin embargo, determinar cotas superiores de $\chi(G)$ de manera eficiente a partir de la creación de coloraciones válidas de G sí es posible. Dentro de los muchos algoritmos para obtener una q-coloración válida de G hay un conjunto de algoritmos polinomiales llamados **Métodos Constructivos para la Coloración de Gráficas**[12].

La estrategia general de estos algoritmos consiste en construir una q-coloración válida de G a partir de extender una coloración parcial válida de G al ir coloreando un vértice a la vez con el color más pequeño posible. La calidad de la coloración resultante depende fuertemente del orden en que fueron tomados los vértices para ser coloreados. Esta última, es la característica más importante de los MCCG.

Hay cinco estrategias distintas para ordenar y colorear los vértices de una gráfica G por parte de los MCCG: *RANDOM*, *LF* (*Largest First*), *SL* (*Smallest Last*), *DSATUR* (*Degree of Saturation*) y *RLF* (*Recursive Largest First*). Estas cinco estrategias corresponden a las cinco clases en que se divide este conjunto de algoritmos. El ordenamiento de los vértices puede ser **estático** o **dinámico**. Un ordenamiento de los vértices es llamado **estático** si puede ser completamente determinado antes de asignar cualquier color. Un ordenamiento de los vértices es llamado **dinámico** si la elección del próximo vértice a colorear se basa en las asignaciones de color anteriores.

La Tabla C.1 describe la mecánica de cada estrategia, donde U es el conjunto de todos los vértices no coloreados. La complejidad de todas las estrategias es $O(n^2)$ excepto la de *RLF*, la cual tiene complejidad $O(n^3)$. En contraste, en la práctica, el *ranking* de las cinco estrategias con respecto a la calidad de la coloración construida es: *RLF*, *DSATUR*, *LF*, *SL* y *RANDOM*. Por lo que ambos criterios deben ser considerados a la hora de seleccionar uno de estos métodos. Aunque en general, la elección de un MCCG debe hacerse de acuerdo con el tiempo disponible para construir una solución y los recursos computacionales con los que se cuenta.

Los MCCG ofrecen una forma rápida de producir cotas superiores de $\chi(G)$. En particular, los MCCG son útiles para inicializar heurísticas cuyo objetivo es encontrar:

- una q-coloración de G, con alta probabilidad de ser válida y q muy cercana a $\chi(G)$, cuando no se conoce $\chi(G)$; o bien

[12] Usaremos las siglas MCCG para el singular y el plural.

- una q-coloración válida de G, con $\chi(G) \leqslant q$, cuando ya se conoce el valor de $\chi(G)$ o se conoce una cota no trivial de $\chi(G)$.

Tabla C.1: Métodos Constructivos para la Coloración de Gráficas

Tipo de ordenamiento	**Estrategia**	**Descripción**
Estático	*RANDOM*	Los vértices son organizados de manera aleatoria. Después, son elegidos en el orden establecido para ser coloreados.
	LF	Los vértices son organizados de manera decreciente con respecto a sus grados en G. Si hay más de un vértice con el mismo grado, estos se organizan de manera aleatoria. Los vértices son elegidos en el orden establecido para ser coloreados.
	SL	Los vértices son organizados de la siguiente manera: $v_1, v_2, \ldots, v_n$ tal que para cada $i \in \{1, 2, \ldots, n\}$, el vértice v_i tiene el grado más pequeño en la subgráfica de G inducida por $v_1, v_2, \ldots, v_i$. Si hay más de un vértice con el mismo grado, estos se ordenan de manera arbitraria. Los vértices serán coloreados en este orden.
Dinámico	*DSATUR*	Los vértices son ordenados eligiendo, en cada paso de la coloración, el vértice $v \in U$ con el máximo grado de saturación bajo la coloración parcial actual. Si v no es único, se da preferencia al vértice v para el cual $deg_{G[U]}(v)$ es mínimo.
	RLF	Las clases de color son construidas de manera secuencial. Para construir la primera clase de color, los vértices de U son ordenados de manera decreciente con respecto a sus grados en $G[U]$ y, en seguida, son coloreados en el orden establecido con el primer color disponible, siempre que el vértice actual no sea adyacente a alguno anterior ya coloreado. Después, los vértices coloreados, y sus respectivas aristas, son eliminados de G y se procede a construir la siguiente clase de color, de manera recursiva, usando la gráfica resultante de esta eliminación y el siguiente color disponible. Este proceso recursivo se lleva a cabo hasta que todos los vértices sean coloreados. Las clases de color producidas tienen la particularidad de ser conjuntos independientes maximales.

En la siguiente sección se presenta un ejemplo de una implementación específica de un Método Constructivo para la Coloración de Gráficas, se trata del algoritmo **MXRLF** [14, 15] utilizado por la metaheurística **ABAC**. El algoritmo **MXRLF** es usado por **ABAC** para obtener una cota inicial del número cromático de la gráfica de entrada.

C.1. Algoritmo MXRLF

El algoritmo **MXRLF** es un MCCG que utiliza la estrategia *RLF* y tiene la particularidad de limitar el tamaño de las clases de color construidas a través de su parámetro de entrada $MXRLFclassLimit$. La especificación de este algoritmo se muestra en el Pseudocódigo C.1.

Pseudocódigo C.1 Algoritmo **MXRLF**

Entrada: Una gráfica $G = (V, A)$, con $G.V = \{v_1, v_2, \ldots, v_n\}$; el conjunto de todos los vértices no coloreados de G, $uncolouredVertices$; un arreglo $colouring$ de tamaño n; un entero positivo $colour$; y finalmente, un entero positivo $MXRLFclassLimit$.

Salida: Un arreglo de tamaño n que contiene una q-coloración válida para G, tal que para cada $i \in \{1, 2, \ldots, n\}$, el valor de su i-ésimo elemento es el color de v_i.

```
/* Parámetros */
colouredVertices ← ∅

/* Proceso constructivo de la coloración */
if ( uncolouredVertices != ∅ ) then
    v ← arg max { deg_{G[uncolouredVertices]}(u) | u ∈ uncolouredVertices }
    Sea i ∈ {1, 2, ..., n} el índice de v en G.V
    colouring[i] ← colour
    colouredVertices ← colouredVertices ∪ {v}
    P ← {u ∈ uncolouredVertices | u ∉ Adj_{G[uncolouredVertices]}(v)} \ {v}
    R ← {u ∈ uncolouredVertices | u ∈ Adj_{G[uncolouredVertices]}(v)}
    if ( P != ∅ ) then
        while ( P != ∅ ) do
            u ← 0
            U ← { arg max { deg_{G[R∪{u}]}(w) | w ∈ R ∪ {u} } | u ∈ P }
            W ← P ∩ U
            if ( W == ∅ ) then
                u ← arg min { deg_{G[P]}(w) | w ∈ P }
            else
                u ← arg min { deg_{G[P]}(w) | w ∈ W }
            end if
            Sea j ∈ {1, 2, ..., n} el índice de u en G.V
            colouring[j] ← colour
            R ← R ∪ Adj_{G[P]}(u)
            P ← P \ (Adj_{G[P]}(u) ∪ {u})
            colouredVertices ← colouredVertices ∪ {u}
            if ( |colouredVertices| + 1 > MXRLFclassLimit ) then
                P ← ∅
            end if
        end while
    end if
    uncolouredVertices ← uncolouredVertices \ colouredVertices
    if ( uncolouredVertices != ∅ ) then
        colour ← colour + 1
        MXRLF(G, uncolouredVertices, colouring, colour, MXRLFclassLimit)
    end if
else
    return colouring
end if
```

Apéndice D

Aplicaciones de la *ACO*

La Tabla D.1, presentada por Dorigo y Stützle en [6], muestra las referencias más relevantes de las principales aplicaciones de la metaheurítica *ACO* a lo largo del tiempo.

Tabla D.1: Aplicaciones de la *ACO*

Tipo de problema	**Nombre del problema**	**Referencias principales**
Enrutamiento	Agente Viajero	Dorigo, Maniezzo & Colorni (1991, 1996) Dorigo (1992) Gambardella & Dorigo (1995) Dorigo & Gambardella (1997) Stützle & Hoos (1997, 2000) Bullnheimer, Hartl & Strauss (1999) Cordón, de Viana, Herrera & Morena (2000)
	Enrutamiento Vehicular	Bullnheimer, Hartl & Strauss (1999) Gambardella, Taillard & Agazzi (1999) Reimann, Stummer & Doerner (2002)
	Ordenamiento Secuencial	Gambardella & Dorigo (1997, 2000)
Asignación	Asignación Cuadrática	Maniezzo, Colorni, & Dorigo (1994) Stützle (1997) Maniezzo & Colorni (1999) Maniezzo (1999) Stützle & Hoos (2000)
	Coloración de Gráficas	Costa & Hertz (1997)

Tabla D.1 (Continuación)

Tipo de problema	**Nombre del problema**	**Referencias principales**
Asignación	Asignación Generalizada	Lourenço & Serra (1998, 2002)
	Asignación de Canales	Maniezzo & Carbonaro (2000)
	Asignación de Horarios	Socha, Knowles & Sampels (2002) Socha, Sampels & Manfrin (2003)
Calendarización	Sistema de Producción *Job Shop*	Colorni, Dorigo, Maniezzo & Trubian (1994)
	Sistema de Producción *Open Shop*	Pfahringer (1996)
	Sistema de Producción *Flow Shop*	Stützle (1998)
	Tardanza Total	Bauer, Bullnheimer, Hartl & Strauss (2000)
	Tardanza Total Ponderada	den Besten, Stützle & Dorigo (2000) Merkle & Middendorf (2000, 2003) Gagné, Price & Gravel (2002)
	Calendarización de Proyectos	Merkle, Middendorf & Schmeck (2000, 2002)
	Sistema de Producción *Group Shop*	Blum (2002, 2003)
Subconjunto	Mochila Multiple	Leguizamón & Michalewicz (1999)
	Conjunto Independiente	Leguizamón & Michalewicz (2000)
	Asignación de Redundancia	Liang & Smith (1999)
	Cubierta	Leguizamón & Michalewicz (2000) Hadji, Rahoual, Talbi & Bachelet (2000)
	Clan	Fenet & Solnon (2003)
Otro	Subsecuencia Común Más Larga	Michel & Middendorf (1998, 1999)
	Satisfacción de Restricciones	Solnon (2000, 2002)
	Plegado de Proteínas 2D-HP	Shmygelska, Aguirre-Hernández & Hoos (2002)
	Empaquetamiento en Cajas	Levine & Ducatelle (2003)

Tabla D.1 (Continuación)

Tipo de problema	**Nombre del problema**	**Referencias principales**
Aprendizaje Automático	Reglas de Clasificación	Parpinelli, Lopes & Freitas (2002)
	Redes Bayesianas	de Campos, Gámez, & Puerta (2002)
	Sistemas Difusos	Casillas, Cordón & Herrera (2000)
Enrutamiento en Redes (ER)	ER Orientadas a la Conexión	Schoonderwoerd, Holland, Bruten & Rothkrantz (1996) Schoonderwoerd, Holland & Bruten (1997) White, Pagurek & Oppacher (1998) Di Caro & Dorigo (1998) Bonabeau, Henavy, Guérin, Snyers, Kuntz & Theraulaz (1998)
	ER No Orientadas a la Conexión	Di Caro & Dorigo (1997, 1998) Subramanian, Druschel & Chen (1997) Heusse, Snyers, Guérin & Kuntz (1998) van der Put (1998)
	ER Ópticas	Navarro Varela & Sinclair (1999)

Apéndice E

Construcción de Ejemplares

En el presente apéndice se describen los detalles de la construcción de las gráficas que conforman el conjunto de ejemplares utilizado en los experimentos.

E.1. Malla bidimensional con probabilidad ρ de tener diagonales

Este tipo de gráfica es una malla bidimensional (ver Apéndice A) con la siguiente modificación es su estructura: la probabilidad de que exista la arista que une los vértices $i-1j-1$ e ij o la arista que une los vértices $i-1j$ e $ij-1$, pero no ambas, es ρ; para toda $i \in \{2,3,\dots r\}$ y para toda $j \in \{2,3,\dots s\}$. Se decidió denotar a esta clase de gráfica por $D_{r\times s,\rho}$. El algoritmo presentado en el Pseudocódigo E.1 permite construir este tipo de gráficas.

E.2. Malla tridimensional con probabilidad ρ de tener diagonales

Este tipo de gráfica es una malla tridimensional (ver Apéndice A) con la siguiente modificación en su estructura: la probabilidad de que contengan la arista que une los vértices $i-1j-1k$ e ijk o la arista que une los vértices $i-1jk$ e $ij-1k$, pero no ambas, es ρ; para toda $i \in \{2,3,\dots r\}$, para toda $j \in \{2,3,\dots s\}$ y para toda $s \in \{1,2,\dots,t\}$. Se decidió denotar a esta clase de gráfica por $D_{r\times s\times t,\rho}$. El algoritmo presentado en el Pseudocódigo E.2 fue usado en la construcción de mallas trimensionales con diagonales.

E.3. Gráfica bipartita r-regular de orden $2t$ y saltos de s unidades

Este tipo de ejemplar corresponde a una gráfica $G = (V, E)$ bipartita de orden $2t$, con $t \geqslant 1$, que tiene las siguientes caracteristicas: conjuntos partitos $V_1 = \{v_1, v_2, \dots, v_t\}$ y $V_2 = \{w_1, w_2, \dots, w_t\}$ tales que para toda $i \in \{1, 2, \dots, t\}$,

$$Adj_G(v_i) = \{w_{(((i-1)+js) \bmod t)+1} \mid j \in \{0, 1, \dots, r-1\}\},$$

Pseudocódigo E.1 Algoritmo BIDIMESH-DIAGONAL

Entrada: Dos enteros r y s tales que $r, s \geqslant 2$; y un real positivo ρ tal que $0 \leqslant \rho < 1$.
Salida: La matriz de adyacencias de una malla bidimensional con diagonales.

```
 1: n ← r * s
 2: Sea adjacenciesMatrix un arreglo bidimensional de tamaño n × n
 3: for i ← 1 to n do
 4:     for j ← 1 to n do
 5:         adjacenciesMatrix[i][j] ← 0
 6:     end for
 7: end for
 8:
 9: /* Malla bidimensional */
10: for i ← 1 to s do
11:     for j ← 0 to r − 1 do
12:         if ( j − 1 ⩾ 0 ) then
13:             adjacenciesMatrix[j * s + i][(j − 1) * s + i] ← 1
14:         end if
15:         if ( i − 1 > 0 ) then
16:             adjacenciesMatrix[j * s + i][j * s + (i − 1)] ← 1
17:         end if
18:         if ( j + 1 < r ) then
19:             adjacenciesMatrix[j * s + i][(j + 1) * s + i] ← 1
20:         end if
21:         if ( i + 1 ⩽ s ) then
22:             adjacenciesMatrix[j * s + i][j * s + (i + 1)] ← 1
23:         end if
24:     end for
25: end for
26:
27: /* Diagonales */
28: for i ← 2 to s do
29:     for j ← 1 to r − 1 do
30:         α ← random(0, 1)
31:         if ( α ⩽ ρ ) then
32:             Seleccionar de manera aleatoria, con probabilidad 0.5, β ∈ {0, 1}
33:             if ( β == 0 ) then
34:                 adjacenciesMatrix[(j − 1) * s + (i − 1)][j * s + i] ← 1
35:                 adjacenciesMatrix[j * s + i][(j − 1) * s + (i − 1)] ← 1
36:             end if
37:             if ( β == 1 ) then
38:                 adjacenciesMatrix[(j − 1) * s + i][j * s + (i − 1)] ← 1
39:                 adjacenciesMatrix[j * s + (i − 1)][(j − 1) * s + i] ← 1
40:             end if
41:         end if
42:     end for
43: end for
```

Pseudocódigo E.2 Algoritmo TRIDIMESH-DIAGONAL

Entrada: Tres enteros r, s y t tales que $r, s, t \geqslant 2$; ; y un real positivo ρ tal que $0 \leqslant \rho < 1$.
Salida: La matriz de adyacencias de una malla tridimensional con diagonales.

```
 1: n ← r * s * t
 2: q ← r * s
 3: Sea adjacenciesMatrix un arreglo bidimensional de tamaño n × n
 4: for i ← 1 to n do
 5:     for j ← 1 to n do
 6:         adjacenciesMatrix[i][j] ← 0
 7:     end for
 8: end for
 9: for  k ← 0 to t − 1 do
10:
11:     /* Malla tridimensional */
12:     for i ← 1 to s do
13:         for j ← 0 to r − 1 do
14:             if ( j − 1 ⩾ 0 ) then
15:                 adjacenciesMatrix[j ⋆ s + i + k * q][(j − 1) * s + i + k * q] ← 1
16:             end if
17:             if ( i − 1 > 0 ) then
18:                 adjacenciesMatrix[j * s + i + k * q][j * s + (i − 1) + k * q] ← 1
19:             end if
20:             if ( j + 1 < r ) then
21:                 adjacenciesMatrix[j * s + i + k * q][(j + 1) * s + i + k * q] ← 1
22:             end if
23:             if ( i + 1 ⩽ s ) then
24:                 adjacenciesMatrix[j * s + i + k * q][j * s + (i + 1) + k * q] ← 1
25:             end if
26:             if ( k + 1 < t ) then
27:                 adjacenciesMatrix[j * s + i + k * q][j * s + i + (k + 1) * q] ← 1
28:                 adjacenciesMatrix[j * s + i + (k + 1) * q][j * s + i + k * q] ← 1
29:             end if
30:         end for
31:     end for
32:
33:     /* Diagonales */
34:     for i ← 2 to s do
35:         for j ← 1 to r − 1 do
36:             α ← random(0, 1)
37:             if ( α ⩽ ρ ) then
38:                 Seleccionar de manera aleatoria, con probabilidad 0.5, β ∈ {0, 1}
39:                 if ( β == 0 ) then
40:                     adjacenciesMatrix[(j − 1) * s + (i − 1) + k * q][j * s + i + k * q] ← 1
41:                     adjacenciesMatrix[j * s + i + k * q][(j − 1) * s + (i − 1) + k * q] ← 1
42:                 end if
43:                 if ( β == 1 ) then
44:                     adjacenciesMatrix[(j − 1) * s + i + k * q][j * s + (i − 1) + k * q] ← 1
45:                     adjacenciesMatrix[j * s + (i − 1) + k * q][(j − 1) * s + i + k * q] ← 1
46:                 end if
47:             end if
48:         end for
49:     end for
50: end for
```

donde r y s son dos enteros positivos, fijados previamente, tales que $r \leqslant t$ y $deg_G(v_i) = r$. Se decidió denotar a esta clase de gráfica por $B_{r,s,t}$. El algoritmo mostrado en el Pseudocódigo E.3 permite construir la matriz de adyacencias de este tipo de ejemplar.

Pseudocódigo E.3 Algoritmo BIPARTITE-REGULAR

Entrada: Tres enteros positivos r, s y t tales que $r \leqslant t$ y para toda $i \in \{1, 2, \ldots, t\}$, $|A| = r$; donde $A = \{(((i-1) + js) \bmod t) + 1 \mid j \in \{0, 1, \ldots, r-1\}\}$.
Salida: La matriz de adyacencias de una gráfica bipartita r-regular de orden $2t$ y saltos de s unidades.

```
1: n ← 2 * t
2: Sea adjacenciesMatrix un arreglo bidimensional de tamaño n × n
3: for i ← 1 to n do
4:     for j ← 1 to n do
5:         adjacenciesMatrix[i][j] ← 0
6:     end for
7: end for
8:
9: /* Bipartición */
10: for i ← 1 to t do
11:     for j ← 0 to r − 1 do
12:         adjacenciesMatrix[i][(((i − 1) + j * s) mod t) + 1 + t] ← 1
13:         adjacenciesMatrix[(((i − 1) + j * s) mod t) + 1 + t][i] ← 1
14:     end for
15: end for
```

E.4. Gráfica k-partita completa de tamaño máximo

Por el Corolario A.1, una gráfica k-partita completa, de orden $n \geqslant 2$, tiene tamaño máximo si y solo si la partición de $G.V$ está conformada por $k - r$ conjuntos partitos de cardinalidad q y r conjuntos partitos de cardinalidad $q + 1$; donde q, y r son el cociente y residuo de n/k respectivamente. A este tipo de ejemplares se les denotó $K_{max}(n, k)$. El algoritmo presentado en el Pseudocódigo E.4 construye la matriz de adyacencias de esta clase de gráficas k-partitas completas.

E.5. Gráfica aleatoria de orden n y probabilidad ρ

Una gráfica aleatorio de orden n y probabilidad ρ, denotada como $G(n, \rho)$, es una gráfica de orden n tal que existe una arista con probabilidad ρ entre cualquier par de vértices, independientemente de la existencia o no existencia de cualquier otra arista. Esta familia de gráficas ha sido de profundamente estudiada con respecto al Problema de la Coloración de Gráficas, especialmente para $\rho = 0.5$.

Pseudocódigo E.4 Algoritmo k-Partite-Complete-Max

Entrada: Un entero positivo $n \geqslant 2$ y un entero positivo k tal que $2 \leqslant k \leqslant n$.
Salida: La matriz de adyacencias de una gráfica k-partita completa de orden n y tamaño máximo.

```
1: q ← ⌊n/k⌋
2: r ← n mod k
3: partition ← ∅
4: Sea adjacenciesMatrix un arreglo bidimensional de tamaño n × n
5: for i ← 1 to n do
6:     for j ← 1 to n do
7:         adjacenciesMatrix[i][j] ← 0
8:     end for
9: end for
10:
11: /* Construcción de conjuntos partitos */
12: I ← {1, 2, ..., n}
13: for s ← 1 to k − r do
14:     V_s ← ∅
15:     for i ← 1 to q do
16:         Seleccionar de manera aleatoria, con probabilidad 1/|I|, index ∈ I
17:         V_s ← V_s ∪ {index}
18:         I ← I \ {index}
19:     end for
20:     Insertar V_s en partition como elemento
21: end for
22: for t ← 1 to r do
23:     V_{k−r+t} ← ∅
24:     for i ← 1 to q + 1 do
25:         Seleccionar de manera aleatoria, con probabilidad 1/|I|, index ∈ I
26:         V_{k−r+t} ← V_{k−r+t} ∪ {index}
27:         I ← I \ {index}
28:     end for
29:     Insertar V_{k−r+t} en partition como elemento
30: end for
31:
32: /* Adyacencias */
33: for each U ∈ partition do
34:     for each W ∈ partition do
35:         if ( U != W ) then
36:             for each i ∈ U do
37:                 for each j ∈ W do
38:                     adjacenciesMatrix[i][j] ← 1
39:                 end if
40:             end for
41:         end if
42:     end for
43: end for
```

Pseudocódigo E.5 Algoritmo RANDOM-GRAPH

Entrada: Un entero positivo n y un real positivo ρ tal que $0 \leqslant \rho < 1$.
Salida: La matriz de adyacencias de una gráfica aleatoria de orden n y probabilidad ρ.

```
 1: Sea adjacenciesMatrix un arreglo bidimensional de tamaño n × n
 2: for i ← 1 to n do
 3:     for j ← 1 to n do
 4:         adjacenciesMatrix[i][j] ← 0
 5:     end for
 6: end for
 7:
 8: /* Adyacencias aleatorias */
 9: for i ← 1 to n do
10:     for j ← i + 1 to n do
11:         α ← random(0, 1)
12:         if ( α ⩽ ρ ) then
13:             adjacenciesMatrix[i][j] ← 1
14:             adjacenciesMatrix[j][i] ← 1
15:         end if
16:     end for
17: end for
```

Índice de figuras

Índice de tablas

Índice de pseudocódigos

ÍNDICE DE PSEUDOCÓDIGOS

Bibliografía

[1] C. A. Coello Coello. *Búsqueda Tabú: Evitando lo Prohibido*. Soluciones Avanzadas, Año 5, Número 49, páginas 72-80, México, 1997. 5

[2] Ant Colony Optimization. http://www.aco-metaheuristic.org/

[3] M. Dorigo. *Optimization, Learning and Natural Algorithms*. Ph.D. Thesis, Politecnico di Milano, Italy, 1992. 6

[4] M. Dorigo. & G. Di Caro. *The Ant Colony Optimization Meta-Heuristic*. In: D. Corne, M. Dorigo & F. Glover (Eds.). *New Ideas in Optimization*. McGraw-Hill, pages 11-32, UK, 1999. 6

[5] M. Dorigo, G. Di Caro & L. M. Gambardella. *Ant Algorithms for Discrete Optimization*. Artificial Life, Volume 5, Number 2, pages 137-172, USA, 1999. 6

[6] M. Dorigo & T. Stützle. *Ant Colony Optimization*. The MIT Press, USA, 2004. 5, 8, 9, 87

[7] G. Chartrand & O. R. Oellerman. *Applied and Algorithmic Graph Theory*. Mc Graw Hill, USA, 1993.

[8] G. Chartrand & P. Zhang. *Chromatic Graph Theory*. CRC Press, USA, 2009.

[9] D. Costa & A. Hertz. *Ants Can Colour Graphs*. The Journal of the Operational Research Society, Volume 48, Issue 3, pages 295-305, USA, 1997. 21

[10] K. A. Dowsland & J. M. Thompson. *An improved ant colony optimisation heuristic for graph colouring*. Discrete Applied Mathematics, Volume 156, Issue 3, pages 313-324, Netherlands, 2008. 39

[11] T. N. Bui, T. V. H. Nguyen, C.M. Patel & K.-A.T. Phan. *An ant-based algorithm for coloring graphs*. Discrete Applied Mathematics, Volume 156, Issue 2, pages 190-200, Netherlands, 2008. 49

[12] M. M. Halldórson. *A still better performance guarantee for approximate graph coloring*. Information Processing Letters, Volume 45, Issue 1, pages 19-23, Netherlands, 1993.

[13] M. Bellare, O. Goldreich & M. Sudan. *Free bits, PCPs and non-approximability—towards tight results*. SIAM Journal on Computing, Volume 27, Issue 3, pages 804-915, USA, 1998.

[14] F.T. Leighton. *A graph coloring algorithm for large scheduling problems*. Journal of Research of the National Bureau of Standards, Volume 84, Issue 6, pages 489-506, USA, 1979. 84

[15] D.S. Johnson, C. Aragon, L. McGeoch & C. Schevon. *Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning.* Operations Research, Volume 39, Issue 3, pages 378-406, USA, 1991. 84

Índice alfabético