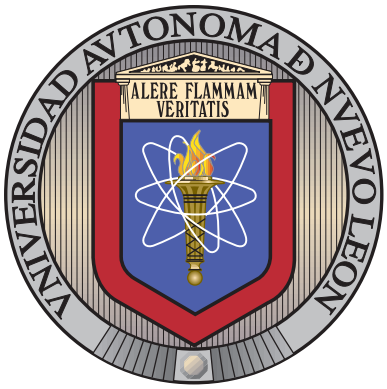
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO



COMPLEJIDAD COMPUTACIONAL ESTRUCTURAL EN REDES COMPLEJAS

POR

TANIA TURRUBIATES LÓPEZ

EN OPCIÓN AL GRADO DE

DOCTOR EN INGENIERÍA

CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN JULIO 2012

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

DIVISIÓN DE ESTUDIOS DE POSGRADO

COMPLEJIDAD COMPUTACIONAL ESTRUCTURAL EN REDES COMPLEJAS

POR

TANIA TURRUBIATES LÓPEZ

EN OPCIÓN AL GRADO DE

DOCTOR EN INGENIERÍA

CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

SAN NICOLÁS DE LOS GARZA, NUEVO LEÓN JULIO 2012

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

División de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Complejidad computacional estructural en redes complejas», realizada por la alumna Tania Turrubiates López, con número de matrícula 1508354, sea aceptada para su defensa como opción al grado de Doctor en Ingeniería con especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dra. Satu Elisa Schaeffer
Asesor

Dra. Laura Cruz Reyes
Revisor

Dr. José Arturo Berrones Santos
Revisor

Dr. Hugo Jair Escalante Balderas
Revisor

Dr. Valmir Carneiro Barbosa
Revisor

Vo. Bo.

Dr. Moisés Hinojosa Rivera
División de Estudios de Posgrado

San Nicolás de los Garza, Nuevo León, julio 2012

A mis padres,
por siempre ser mi mayor soporte.

A mis hermanos,
por sus palabras de aliento.

A Moyito,
que tanto quiero y que me hizo falta.

Gracias Dios,
por poner en mi camino a la ciencia
para conocerte mejor.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SIMBOLOGÍA

$\mathcal{G}$ *grafo* $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$

$V_\mathcal{G}$ conjunto de *nodos* en el grafo

$E_\mathcal{G}$ conjunto de *aristas* en el grafo

n la cardinalidad del conjunto $V_\mathcal{G}$ (orden del grafo), la cantidad de nodos

e la cardinalidad del conjunto $E_\mathcal{G}$ (tamaño del grafo), la cantidad de aristas

$\mathsf{den}(\mathcal{G})$ *densidad* del grafo

$\mathbf{A}$ matriz de *adyacencia*

$\Gamma(i)$ conjunto *vecino* del nodo i

$\langle k \rangle$ *grado* promedio del grafo

$\Delta(\mathcal{G})$ *grado máximo*

$\delta(\mathcal{G})$ *grado mínimo*

$\mathsf{ex}(i)$ *excentricidad* de un nodo

$\mathsf{diam}(\mathcal{G})$ diámetro

$\mathsf{rad}(\mathcal{G})$ radio

$\mathcal{S}^\mathcal{G}$ subgrafo de $\mathcal{G}$

$\Pr(k)$ probabilidad de que un nodo tenga grado k

$C(i)$ coeficiente de agrupamiento del nodo i

$C(\mathcal{G})$ coeficiente de agrupamiento del grafo

$L(\mathcal{G})$ longitud de ruta más corta

$D(\mathcal{G})$ diámetro del grafo

$E(\mathcal{G})$ eficiencia global del grafo

E_ℓ eficiencia local del grafo

$\mathit{ddc}(i)$ coeficiente de dispersión del grado

$\mathit{ddc}(\mathcal{G})$ coeficiente de dispersión del grado global

ER grafos generados con el modelo Erdős-Rényi

WS grafos generados con el modelo Watts-Strogatz

BA grafos generados con el modelo Barabási-Albert

KL grafos generados con el modelo de Kleinberg

RGG grafos generados con el modelo de grafos geométricos aleatorios

SAT problema de satisfactibilidad booleana

$\mathcal{O}$ cota superior

Ω cota inferior

Θ cota estrecha

L lenguaje

$\mathcal{S}$ espacio de soluciones

$\mathcal{N}(s)$ vecindario en el espacio de búsqueda

AGRADECIMIENTOS

Mi profundo agradecimiento a los miembros del comité de esta tesis: Dra. Satu Elisa Schaeffer, Dra. Laura Cruz Reyes, Dr. Valmir Carneiro Barbosa, Dr. José Arturo Berrones Santos y Dr. Hugo Jair Escalante Balderas por las sugerencias dadas.

Mi sincero aprecio a la Dra. Elisa Schaeffer por haber dirigido esta tesis; gracias a ella aprendí muchas cosas que no solo están reflejadas y escritas en esta tesis. Al Dr. Morten Foss por recibirme en la Universidad de Aarhus, Dinamarca, y darme la oportunidad de trabajar en un área de aplicación totalmente diferente a la computacional.

Reciban mi reconocimiento las instituciones de las cuales recibí apoyo: Universidad Autónoma de Nuevo León y el Consejo Nacional de Ciencia y Tecnología que proporcionaron todas las facilidades necesarias de infraestructura y económicas para el desarrollo de esta investigación, particularmente a la Facultad de Ingeniería Mecánica y Eléctrica por los apoyos economicos complementarios para la asistencia a congresos y estancias en el extranjero.

A los profesores del Posgrado de Ingeniería de Sistemas de los cuales aprendí mucho, así también a mis compañeros de los programas de maestría y doctorado por su compañerismo y la amistad que me brindaron.

A mi familia, padres y hermanos, por siempre estar apoyando en todos los aspectos, por siempre estar ahí a pesar de todo; han sido mi gran soporte. A mis amigos del Instituto Tecnológico de Ciudad Madero que coincidimos en esta ciudad y me brindaron su soporte moral cuando más lo necesitaba, a mi amiga casi hermana Karime por tenerme siempre en sus oraciones, a mis amigos del Instituto Tecnológico Superior de Álamo Temapache.

RESUMEN

Tania Turrubiates López.

Candidato para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.
Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio:

COMPLEJIDAD COMPUTACIONAL ESTRUCTURAL
EN REDES COMPLEJAS

Número de páginas: 112.

OBJETIVOS Y MÉTODO DE ESTUDIO: El desarrollo del presente trabajo se enfoca en estudiar el efecto de la estructura de instancias en la complejidad computacional al resolver un problema. Para realizar este trabajo, se acota la población de estudio a instancias de problemas computacionales relacionados con grafos, específicamente al problema de k-coloreo. Esto no es impedimento para que el estudio se extienda a otros problemas computacionales, gracias a las reducciones que se pueden realizar entre problemas.

Con el fin de observar efectos estructurales de las instancias en la solución del problema, se utilizan modelos de generación de redes complejas, los cuales proporcionan una herramienta para generar instancias de grafos de igual tamaño y orden pero con diferente estructura; también se utilizan funciones de caracterización de redes complejas para obtener medidas de propiedades estructurales de las instancias generadas.

Se toman algoritmos del estado del arte para la solución del problema de k-coloreo. Estos algoritmos formulan de diferente manera la función objetivo y la vecindad de soluciones, lo cual permite — independientemente de estos dos aspectos — observar la influencia de la estructura de la instancia en el espacio de soluciones generado.

Para cuantificar los efectos estructurales en el desempeño algorítmico, se propone una medida de desempeño basada en la convergencia de los algoritmos hacia la solución final. Se hace uso de diseños experimentales, regresiones y algoritmos de agrupamiento para establecer relaciones entre la medida de desempeño algorítmico propuesta y las propiedades estructurales de las instancias y cómo es esa relación.

CONTRIBUCIONES Y CONCLUSIONES: La contribución del presente trabajo se centra en establecer un marco de trabajo efectivo y práctico para el estudio del efecto de la estructura de instancias de grafos en la complejidad computacional de un problema. Para ello se propone una medida que refleja el efecto de la estructura de la instancia en el desempeño algorítmico independientemente del tamaño de la instancia y parámetros del problema. Esta medida puede ser aplicada en el estudio de los efectos estructurales de instancias de otros problemas computacionales diferentes al abordado en esta tesis y usando otros tipos de algoritmos.

El marco de trabajo propuesto permite establecer relaciones entre características estructurales de instancias y el efecto que tiene en la dificultad asociada a la solución del problema, lo cual proporciona las bases para desarrollar algoritmos especializados que tomen en cuenta la estructura de la instancia de forma que sean más eficientes, desarrollar algoritmos que en tiempo de ejecución se adapten a la estructura de la instancia y optimizar la estructura de instancias de redes reales para facilitar o dificultar un proceso según las necesidades del usuario.

Firma del asesor: _____

Dra. Satu Elisa Schaeffer

CAPÍTULO 1

INTRODUCCIÓN

Los problemas computacionales no son solo cosas que tienen que resolverse,
también son objetos que valen la pena estudiarse.
- Christos H. Papadimitriou.

El ser humano desde sus orígenes se ha enfrentado a situaciones en diferentes contextos que debe *resolver* tomando en cuenta los *recursos* disponibles, evaluando *alternativas* de solución y eligiendo la mejor o más adecuada en ese momento. Conforme el hombre ha ido evolucionado, también han evolucionado los problemas que debe resolver. La cantidad de factores que intervienen en los problemas ha crecido y la manera en cómo los factores interactuan entre sí hace que los problemas sean cada vez más complejos [Heylighen, 1999]. Por ello la necesidad de contar con *mecanismos* que lo auxilien en el proceso de solución de problemas. Ante esta situación se ha puesto de manifiesto la necesidad de definir claramente procedimientos a seguir para resolver un problema, lo que se conoce como *algoritmo*; también se han creado máquinas que realizan grandes cálculos para solucionar problemas. Hoy en día, cada vez más, se hace uso de las computadoras para solucionar problemas, tomar decisiones, facilitar actividades del día a día y hasta tener un rato de esparcimiento.

De cierta manera, ahora las computadoras se encuentran resolviendo problemas, evaluando alternativas y presentando la mejor o las mejores posibles, saber que un problema se puede resolver por una computadora no basta para decir si resulta práctico hacerlo o no. Se tiene que tener en cuenta que es posible resolver una gran cantidad de problemas,

es decir, se puede diseñar un algoritmo y escribir un programa, pero los recursos de *tiempo* de procesamiento y el *espacio* en memoria de las computadoras son importantes en la práctica. Actualmente la cantidad de memoria de una computadora es un recurso limitado y generalmente se desea un tiempo de respuesta corto con respecto al tamaño del problema que se está resolviendo, también es deseable saber si el caso del problema a resolver es fácil o difícil lo cual se aborda en este proyecto.

En este capítulo se abordarán de manera general, en la primera sección, los temas que se tratarán en la tesis, los cuales se abordarán con mayor detalle en el Capítulo 2. Posteriormente se describirá el problema que se está abordando, lo que motivó a estudiarlo, la hipótesis principal de este trabajo, los objetivos planteados y la contribución científica.

1.1 ANTECEDENTES

El ser humano interactúa constantemente con sistemas, resolviendo problemas y tomando decisiones. Un *sistema* es un conjunto de elementos que se relacionan entre sí para lograr una tarea específica. Muchos sistemas naturales y hechos por el hombre son considerados como *sistemas complejos*. Estos sistemas complejos están caracterizados por tener una gran cantidad de elementos que interactúan entre sí de acuerdo a reglas que cambian con el tiempo, lo que provoca una dinámica en las conexiones de los elementos, generando comportamientos y fenómenos que no son predecibles ni explicables a partir del estudio aislado de los elementos [Amaral y Ottino, 2004]. Los sistemas complejos se presentan a diferentes niveles de organización como las redes metabólicas de proteínas y redes de neuronas a nivel celular; redes de transporte, redes de telecomunicaciones y redes sociales a nivel tecnológico [Barabási, 2007].

Los sistemas complejos pueden ser modelados a través de *grafos*, donde los elementos son representados por *nodos* y las relaciones existentes entre los elementos se representan mediante *aristas*. Estos grafos conocidos como *redes complejas* proporcionan una herramienta flexible y general para representar de manera abstracta los sistemas complejos y los cambios dinámicos en su patrón de conexión llamado *topología* [da F.

Costa *et al.*, 2007]. Las redes complejas poseen una estructura topológica no trivial y un comportamiento dinámico, lo que ha motivado el estudio de las *características estructurales* de estas redes. En este proyecto se busca establecer una metodología que permita entender cómo estas propiedades influyen en los procesos que se llevan a cabo en ellas.

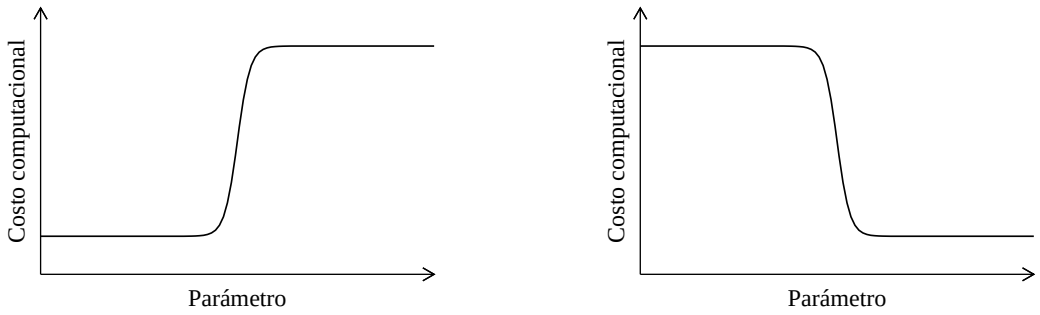
Según la manera en que se relacionan entre sí los elementos en la red compleja, se pueden clasificar como redes *uniformes* [Erdős y Renyí, 1960], redes de *pequeño mundo* [Watts y Strogatz, 1998], redes de *libre escala* o *exponenciales* [Albert y Barabási, 2002]. Cada una de estas redes posee características estructurales que las hacen vulnerables o resistentes a ataques intencionales o fallas aleatorias en el sistema que representan. Debido a la gran cantidad de sistemas que pueden ser modelados como redes complejas, diversas investigaciones se han enfocado realizar análisis de *propiedades topológicas* en términos de múltiples *métricas* [da F. Costa *et al.*, 2007] y en el estudio de los *procesos* que se llevan a cabo en las redes complejas y que pueden verse afectados por la estructura de la red compleja [Alderson, 2008; López, 2007; Schaeffer, 2006].

Un fenómeno de interés en el área de los sistemas complejos es el fenómeno de *percolación* que describe las transiciones abruptas del sistema de un estado a otro, llamdas *transiciones de fase*. Este fenómeno está relacionado con la aparición de un grupo de elementos de tamaño infinito que tiene como consecuencia cambios significativos en propiedades físicas del sistema [Albert y Barabási, 2002]. Este fenómeno llevado al área de los sistemas complejos estudia la probabilidad con la cual, a partir de grupos aislados de nodos se forma un grupo de gran tamaño, que se expande por toda la red o viceversa, en términos más sencillos, estudia qué pasa con la red si se eliminan o agregan nodos o aristas aleatoriamente. Una de las principales aplicaciones del estudio del fenómeno de percolación es la robustez de las redes de comunicación y biológicas [Holme, 2004].

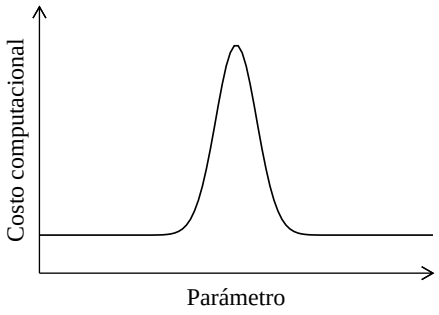
En investigaciones recientes se ha descubierto que problemas computacionales clasificados como **NP**-completos[1] presentan *transiciones de fase* tal como los sistemas complejos. En los problemas **NP**-completos las transiciones de fase se refieren a cambios en la

[1]La clase **NP** contiene aquellos problemas que son decididos por una máquina de Turing no determinista en tiempo polinomial. Los problemas **NP**-completos son los más difíciles de la clase **NP**; este tema se trata en la Sección 2.3.

dificultad de resolver el problema [Achlioptas *et al.*, 2005; Fu y Anderson, 1986; Martin *et al.*, 2001]. Se ha detectado que de acuerdo a ciertos parámetros referentes al problema se pueden presentar diferentes transiciones de fase en la solución del problema [Achlioptas *et al.*, 2005; Mammen y Hogg, 1997]; ejemplos de estas transiciones se presentan en la Figura 1.1. Es importante recordar que el uso del término "dificultad" está relacionado con el costo computacional asociado al uso de recursos como tiempo de procesamiento y espacio en memoria.



(a) *Transición fácil-difícil.* (b) *Transición difícil-fácil.*



(c) *Transición fácil-difícil-fácil.*

Figura 1.1: *En problemas computacionales, al variar un parámetro del problema en cierto rango, se puede observar un cambio drástico en la dificultad asociada al costo computacional, en la Figura 1.1(a) se muestran cambios de un régimen fácil a uno difícil, también se pueden tener cambios de régimen difícil a uno fácil como en la Figura 1.1(b). Además se pueden tener cambios de régimen fácil-difícil-fácil (ver Figura 1.1(c)) cuando ciertos parámetros asociados a la instancia del problema cambian.*

Para estudiar la facilidad o dificultad con la cual se resuelve un problema, se ha recurrido a analizar las propiedades del espacio de soluciones generados por los algorítmos de búsqueda. Este espacio de soluciones es representado como una *superficie de aptitudes*

[Rosas, 2007; Yossi y Poli, 2005] y su caracterización ha sido orientada a describir las trayectorias que siguen los algorítmos en la búsqueda de mejores soluciones.

1.2 DESCRIPCIÓN DEL PROBLEMA

La *teoría de complejidad computacional* estudia la cantidad de recursos computacionales — tiempo de procesamiento y espacio en memoria — necesarios para resolver un problema en su ***peor caso***. Entre mayor sea la cantidad de recursos usados se dice que el problema es más difícil de solucionar [Papadimitriou, 1994].

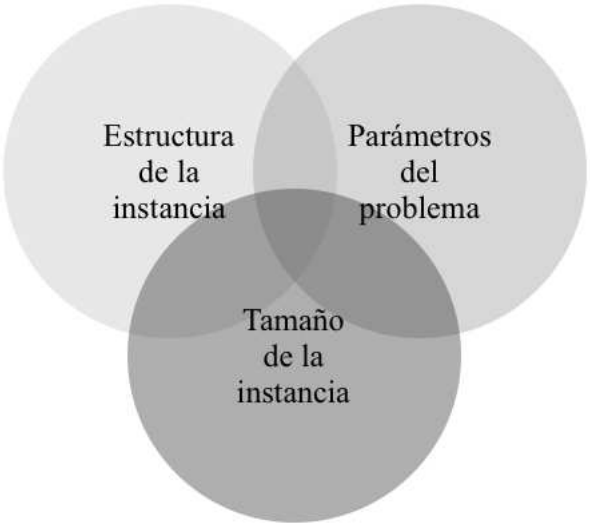


Figura 1.2: *Factores que influyen en la solución de un problema en su peor caso.*

Para estimar esta cantidad de recursos y por ende la dificultad del problema se ha tomado como referencia el *tamaño de la instancia* del problema que se desea resolver; en términos simples, la cantidad de datos que se procesarán para solucionar el problema. En general es de interés realizar esta estimación en un sentido *asintótico*, es decir, para un tamaño de instancia suficientemente grande. Estudios recientes indican que considerar solo el tamaño de la instancia no es suficiente para determinar si será fácil o difícil de resolver [Flum y Grohe, 2006; Orponen *et al.*, 1994]. La *teoría de complejidad parametrizada* explica cómo parámetros de un problema influyen de manera importante en el uso de estos recursos para su resolución [Flum y Grohe, 2006]. Aún así en la práctica, para instancias

de igual tamaño y con los mismos parámetros de un problema, se observa que unas instancias se resuelven más fácilmente que otras por diferentes algoritmos [Porumbel *et al.*, 2010; Smith-Miles *et al.*, 2009; Watson *et al.*, 1999]. Esto da indicios que la estructura de la instancia es otro factor (véase Figura 1.2) que influye de manera significativa en la facilidad o dificultad para resolverla y que es lo que se desea estudiar en este proyecto.

Es en este punto donde surgen cuestionamientos importantes que motivan el desarrollo del presente proyecto de investigación: ¿Qué ***propiedades estructurales*** de las instancias afectan en el ***desempeño de los algoritmos*** usados para resolver el problema? ¿Qué propiedades estructurales de las instancias hacen que el problema a resolver sea más fácil o más difícil? En esta tesis se buscan las respuestas a estas preguntas.

1.3 JUSTIFICACIÓN

En este proyecto se busca ***caracterizar la dificultad*** de una instancia en términos más descriptivos que el tamaño de la misma, que es la medida tradicional de complejidad computacional. El principal objetivo es ***identificar características estructurales*** en las instancias que permitan determinar si se está ante una instancia fácil o difícil. Con esta información se podrán establecer las bases para la construcción de instancias fáciles o difíciles de resolver lo que lleva también a diseñar y seleccionar algoritmos que obtengan ventajas del conocimiento de la estructura de la instancia.

De interés particular es la ***optimización estructural*** donde una instancia dada se modifica para que la solución de ciertos problemas sea más fácil o difícil. Existen situaciones donde se requiere que la infraestructura facilite ciertos procesos como el transporte de mercancía o mensajes, evacuaciones en caso de desastres o la fluidez vial, mientras en otras situaciones es deseable que ciertos procesos sean difíciles de realizar como la propagación de virus y ataques a un sistema de transporte, entre otros [Kicinger *et al.*, 2005; Surry *et al.*, 1995; Xu, 2010].

1.4 HIPÓTESIS

Además del tamaño de la instancia y los parámetros del problema a resolver, la complejidad de resolución de un problema dada una instancia se ve también afectada por la estructura de la misma; por ello es necesario *estudiar, analizar y caracterizar sus propiedades estructurales*.

1.5 OBJETIVOS

El objetivo principal es ***establecer un marco de trabajo efectivo y práctico*** para problemas de grafos, extensible a otros problemas tratados por la complejidad computacional, que permita *caracterizar la dificultad* inherente en la estructura de instancias iguales en tamaño pero distintas en términos estructurales. Para lograr este objetivo es necesario:

- ***Generar instancias de grafos*** *de igual tamaño pero diferente estructura*.
- ***Caracterizar instancias*** a través de *medidas* de propiedades estructurales.
- ***Caracterizar el desempeño algorítmico*** tomando en cuenta la *convergencia* de los algorítmos de búsqueda hacia una solución, de tal forma que la caracterización permita distinguir efectos de la estructura de las instancias en el desempeño algorítmico.
- ***Diseñar experimentos*** que permitan establecer qué propiedades estructurales de las instancias hacen que sean fáciles o difíciles de resolver y qué relación guardan estas propiedades estructurales con el desempeño algorítmico.
- Realizar y documentar ***experimentación*** de los efectos de propiedades estructurales en el desempeño algorítmico.

1.6 CONTRIBUCIÓN CIENTÍFICA

El presente trabajo de tesis aporta dos contribuciones principales. La primera es la formulación de una *medida* de desempeño algorítmico que permite observar el efecto de la estructura de la instancia, independientemente del tamaño de la instancia y los parámetros del problema. La segunda es el desarrollo de un *marco de trabajo* que permite establecer relaciones entre el desempeño algorítmico y las propiedades estructurales de las instancias de grafos; esto con el objetivo de lograr un entendimiento profundo de la manera en que la estructura del grafo impacta en la solución de un problema.

El conocimiento de la estructura de la instancia se puede aprovechar en el *desarrollo de algoritmos especializados* para ciertos tipos de instancias que ocurren en campos de aplicación particulares para alcanzar mayor eficiencia computacional y mayor calidad en aproximaciones[2] rápidas. También permite el desarrollo de métodos de solución *adaptativos* que caractericen y aprovechen la estructura de la instancia en tiempo de ejecución.

La meta es contribuir en la *caracterización de la complejidad de una instanci*a, tomando en cuenta no solo el tamaño de la instancia o parámetros del problema, sino la estructura de la instancia.

1.7 ESTRUCTURA DE LA TESIS

El restante del trabajo se encuentra estructurado de la siguiente manera: en el Capítulo 2 se presentan los fundamentos teóricos y conceptos necesarios de las diferentes áreas que convergen en el desarrollo de este trabajo como lo es la teoría de complejidad computacional, teoría de grafos y redes complejas. En el Capítulo 3 se presenta una revisión de la literatura existente donde se ha trabajado para determinar qué estructura de la instancia[3] es más fácil o más difícil, cómo se ha abordado y qué se ha logrado.

[2]Se entiende por aproximaciones a las soluciones de un problema que se encuentran dentro de cierto rango alrededor de la mejor solución posible que se puede encontrar de un problema [Vazirani, 2001].

[3]Muchos de los trabajos relacionados usan indistintamente los términos *problema* e *instancia*; en este trabajo hacemos enfásis en la diferencia entre estos dos términos.

En el Capítulo 4 se describe el método de estudio llevado a cabo para la realización de este trabajo. En el Capítulo 5 se presenta la propuesta de un marco de trabajo para abordar el problema planteado así como la propuesta de una medida de desempeño algorítmico que permite detectar la influencia de la estructura de las instancias en la solución del problema. En el Capítulo 6 se describen los experimentos realizados y se analizan los resultados obtenidos. En el Capítulo 7 se describen las conclusiones del trabajo y se comenta brevemente el trabajo que queda por hacer como consecuencia de los resultados obtenidos y nuevos cuestionamientos surgidos.

CAPÍTULO 2

FUNDAMENTOS TEÓRICOS

Los que se enamoran de la práctica sin la teoría
son como los pilotos sin timón ni brújula,
que nunca podrán saber a dónde van.
- Leonardo Da Vinci.

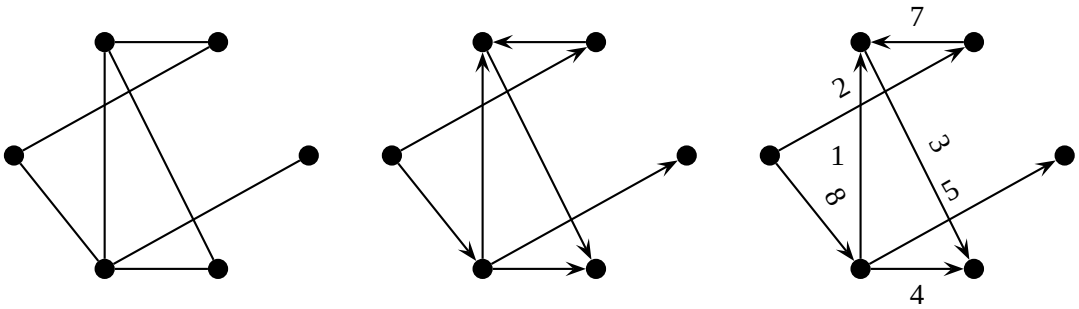
En este capítulo se presentan definiciones y conceptos básicos de las áreas de conocimiento necesarias para abordar el problema planteado y a los cuales se hace referencia en el presente documento. En la Sección 2.1 se presentan conceptos de teoría de grafos los cuales permiten una mejor comprensión de las demás secciones. En la Sección 2.2 se abordan temas de teoría de redes complejas, los cuales son necesarios para la generación y caracterización de instancias. Es importante hablar de la teoría de la complejidad computacional, por lo cual en la Sección 2.3 se aborda este tema, seguida de la Sección 2.4 donde se toca el tema de complejidad parametrizada. En la Sección 2.5 se presentan conceptos básicos referentes a los problemas de optimización y posteriormente en la Sección 2.6 se aborda el tema de algoritmos metaheurísticos.

2.1 TEORÍA DE GRAFOS

El estudio de la teoría de grafos comenzó formalmente cuando Euler [1741] resolvió el problema de los puentes de Königsberg, observando que la solución del problema dependía solamente de las propiedades de sus conexiones. A partir de entonces los gra-

fos se han convertido en una herramienta importante en diversas áreas del conocimiento, ya que permiten representar de manera abstracta situaciones en las cuales elementos se relacionan entre sí. A continuación se describe formalmente la terminología de teoría de grafos utilizada en el presente trabajo.

Un *grafo* $\mathcal{G}$ está definido por dos conjuntos $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$, donde $V_\mathcal{G}$ es el conjunto de *nodos o vértices*, los elementos que se desean representar; y $E_\mathcal{G}$ es el conjunto de *aristas o conexiones* que son las relaciones existentes entre los nodos. El conjunto $E_\mathcal{G}$ está formado por pares de nodos de la forma (i, j). Si tenemos una arista (i, i) se dice que ésta es una arista *reflexiva* llamada *lazo*. Cuando estos pares son no ordenados se dice que el grafo es *no dirigido*; cuando los pares son ordenados se tiene un *grafo dirigido*. En un *grafo ponderado* las aristas, ya sean dirigidas o no, tienen asociado un *peso* $w(i, j)$. En la Figura 2.1 se muestran ejemplos de estos tipos de grafos. Si los nodos o las aristas tienen asignadas *etiquetas* que los haga distinguibles unos de otros se tiene un *grafo etiquetado*. En este trabajo se utilizan grafos no dirigidos y no ponderados, por lo cual las definiciones mencionadas a continuación se formularán en términos de este tipo de grafo.



(a) *Grafo no dirigido.* (b) *Grafo dirigido.* (c) *Grafo dirigido ponderado.*

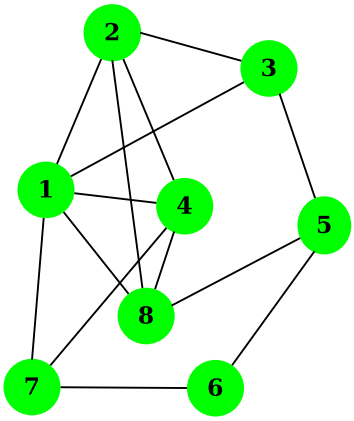
Figura 2.1: *A la izquierda se tiene un grafo no dirigido con seis nodos y siete aristas. En la figura del centro se presenta el mismo grafo con la diferencia que se ha dado dirección a las aristas. A la derecha se tiene un grafo dirigido en el cual las aristas tienen pesos.*

El número de nodos $n = |V_\mathcal{G}|$ es denominado *orden del grafo* y el número total de aristas $e = |E_\mathcal{G}|$ se denomina *tamaño del grafo*. El tamaño del grafo puede ser como mínimo cero y como máximo $n(n-1)/2$. La *densidad* de un grafo $\mathsf{den}(\mathcal{G})$ es la proporción

de aristas presentes del máximo posible:

$$den(\mathcal{G}) = 2e/(n(n-1)). \tag{2.1}$$

Dos nodos son *adyacentes* si una arista los une. Matemáticamente un grafo se puede representar mediante una matriz cuadrada de dimensión $n \times n$, llamada *matriz de adyancencia* $\mathbf{A}$; cada elemento de la matriz de adyacencia $\mathbf{A}(i,j)$ representa la presencia o ausencia de una arista entre los nodos i y j. Si existe una arista entre los nodos i y j, entonces $a_{i,j} = 1$, en caso contrario $a_{i,j} = 0$. El conjunto de nodos adyacentes a un nodo i del grafo $\mathcal{G}$ se denomina *conjunto vecino* del nodo i y se denota por $\Gamma(i)$.



$$\begin{aligned}
\mathcal{G} &= (V, E) \\
V_{\mathcal{G}} &= \{1,2,3,4,5,6,7,8\} \\
E_{\mathcal{G}} &= \{(1,2),(1,3),(1,4),(1,7),(1,8), \\
&\quad (2,3),(2,4),(2,8),(3,5),(4,7) \\
&\quad (4,8),(5,6),(5,8),(6,7)\} \\
n &= |V_{\mathcal{G}}| = 8 \\
e &= |E_{\mathcal{G}}| = 14 \\
\Gamma(1) &= \{2,3,4,7,8\} \\
k_1 &= |\Gamma(1)| = 5 \\
\langle k \rangle &= 2e/n = (2 \times 14)/8 = 3.5 \\
\delta(\mathcal{G}) &= k_6 = 2 \\
\Delta(\mathcal{G}) &= k_1 = 5 \\
den(\mathcal{G}) &= 0.5
\end{aligned}$$

Figura 2.2: *Grafo y su descripción mediante la notación descrita.*

El número de nodos adyacentes a un nodo i es el *grado* del nodo i y se define como

$$k_i = \sum_{j=1}^{n} a_{i,j} = |\Gamma(i)|. \tag{2.2}$$

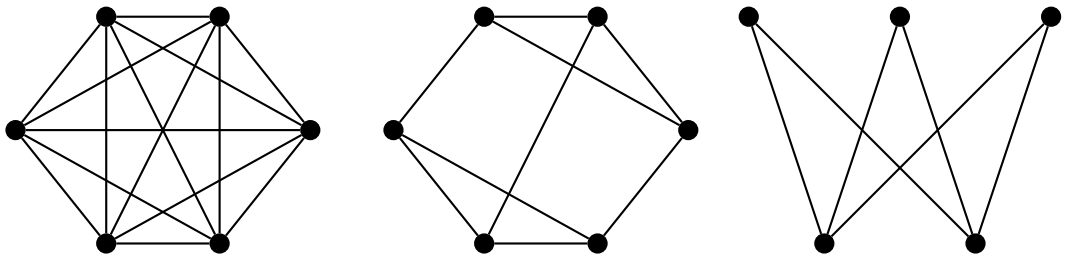
El *grado máximo* del grafo $\mathcal{G}$ se expresa como $\Delta(\mathcal{G})$ y el *grado mínimo* como $\delta(\mathcal{G})$. El *grado promedio* del grafo $\mathcal{G}$ se representa como $\langle k \rangle = 2e/n$. En la Figura 2.2 se presenta un grafo con la notación de los conceptos descritos hasta este momento.

Cuando se tiene un grafo de tamaño máximo se dice que el grafo es *completo* y se denota como K_n. Si todos los nodos de un grafo $\mathcal{G}$ tienen el mismo grado k, el grafo

es *regular* y se denota como k-regular. Un grafo *bipartito* es aquel en el cual los nodos puedes separarse en dos conjuntos disjuntos V_1 y V_2, $V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$ tales que el conjunto de aristas sea

$$\forall u, w \in E \Rightarrow (u \in U \wedge w \in W) \vee (u \in W \wedge w \in U), \tag{2.3}$$

es decir, las aristas únicamente unen nodos de un conjunto con nodos de otro conjunto. En la Figura 2.3 se muestran ejemplos de estos tres tipos de grafos.



(a) *Grafo completo.* (b) *Grafo 3-regular.* (c) *Grafo bipartito.*

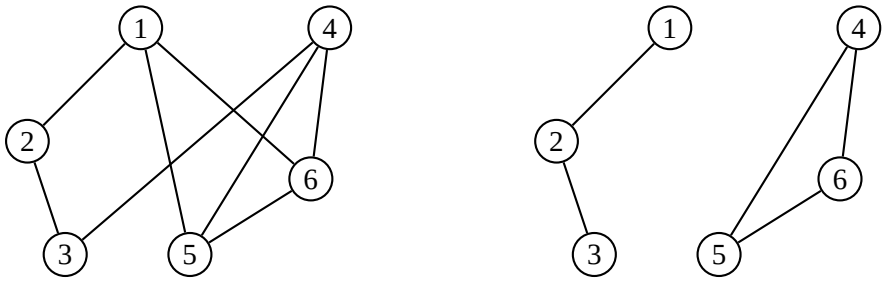
Figura 2.3: *En la Figura 2.3(a) se muestra un grafo completo de seis nodos y quince aristas; en la Figura 2.3(b) se tiene un grafo 3-regular lo que quiere decir que cada nodo tiene grado tres. En la Figura 2.3(c), se presenta un grafo bipartito: los tres nodos de la parte superior forman un conjunto y los dos de la parte inferior forman otro conjunto; como todos los nodos de un conjunto están relacionados con todos los nodos del otro conjunto este es un grafo bipartito completo.*

Si dos nodos i, j no son adyacentes, pueden estar conectados por una secuencia de m aristas, las cuales forman la *ruta* entre i y j, siendo m la longitud de la ruta. A la longitud mínima posible entre todas la rutas existentes entre el nodo i y el nodo j se le conoce como *distancia geodésica* $d_{i,j}$.

Un grafo es *conexo* si para cualquier par de nodos $i, j \in \mathcal{G}$, existe al menos una ruta entre i y j. Si G es conexo, $d_{i,j}$ es positivo y finito, lo que implica que el número de aristas para ir de i a j es finito. Por otro lado, si $\mathcal{G}$ no es conexo, para por lo menos un par i y j no existe una ruta entre ellos, por lo cual se marca que $d_{i,j} = \infty$; para estos grafos no conexos medidas basadas en distancias no tienen sentido al no existir rutas entre todos los nodos.

La *excentricidad* $\mathsf{ex}(i)$ de un nodo i en un grafo conexo $\mathcal{G}$ es la distancia geodésica entre i y el nodo más alejado de i en $\mathcal{G}$. El *diametro* $\mathsf{diam}(\mathcal{G})$ de $\mathcal{G}$ es la mayor de las excentricidades de entre todos los nodos de $\mathcal{G}$, mientras que el *radio* $\mathsf{rad}(\mathcal{G})$ es la menor de ellas.

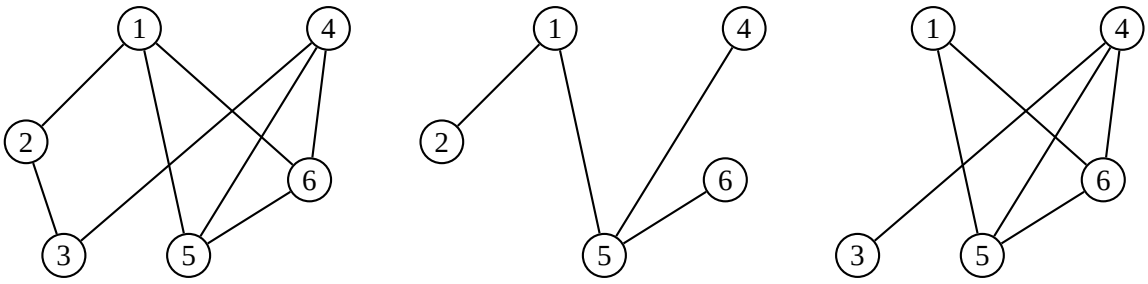
Una *ruta simple* no visita el mismo nodo más de una vez. Un *ciclo* es una ruta simple que empieza y termina en el mismo nodo. Un grafo que no contiene ciclos se dice que es *acíclico*; un grafo acíclico conexo es llamado *árbol*. Un *subárbol* de un grafo es un subconjunto de aristas conexas que no forman un ciclo; al subárbol que contiene todos los nodos de un grafo se le llama *árbol abarcante* del grafo.



Grafo conexo. *Grafo no conexo.*

Figura 2.4: *Ejemplo de grafo conexo y no conexo.*

Un grafo $\mathcal{S}^{\mathcal{G}}$ se le denomina *subgrafo* del grafo $\mathcal{G}$ si $V_{\mathcal{S}^{\mathcal{G}}} \subseteq V_{\mathcal{G}}$ y $E_{\mathcal{S}^{\mathcal{G}}} \subseteq E_{\mathcal{G}}$, donde $(i,j) \in E_{\mathcal{S}^{\mathcal{G}}} \Rightarrow (i \in V_{\mathcal{S}^{\mathcal{G}}} \wedge j \in V_{\mathcal{S}^{\mathcal{G}}})$. El subgrafo $\mathcal{S}^{\mathcal{G}}$ de $\mathcal{G}$ se dice que es *inducido* por $V_{\mathcal{S}^{\mathcal{G}}}$ si $V_{\mathcal{S}^{\mathcal{G}}} \subseteq V_{\mathcal{G}}$ y nodos i, j en $\mathcal{G}$ son adyacentes en $\mathcal{S}^{\mathcal{G}}$ si y solo si i, j son adyacentes en $\mathcal{G}$.



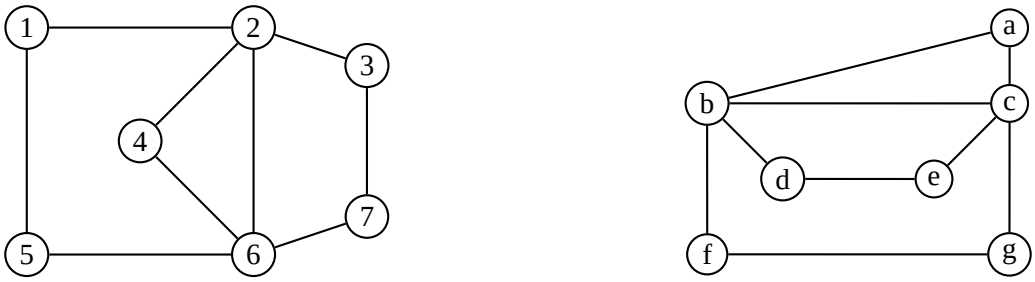
Grafo $\mathcal{G}$. *Un subgrafo de $\mathcal{G}$.* *Un subgrafo inducido de $\mathcal{G}$.*

Figura 2.5: *Ejemplos de subgrafos.*

Un subgrafo completo de $\mathcal{G}$ es llamado *clique* de $\mathcal{G}$. Un clique de orden k es un *k-clique*. El orden máximo de un clique de $\mathcal{G}$ es llamado el *número clique* de $\mathcal{G}$ y es denotado por $\omega(\mathcal{G})$.

Dos grafos $\mathcal{G}$ y $\mathcal{H}$ son *isomorfos*, es decir tienen la misma estructura, si existe una función biyectiva[1] $\phi : V(\mathcal{G}) \to V(\mathcal{H})$ tal que dos nodos i y j son adyacentes en $\mathcal{G}$ si y solo si $\phi(i)$ y $\phi(j)$ son adyacentes en $\mathcal{H}$. La función ϕ es llamada *isomorfismo*. Si para $\mathcal{G}$ y $\mathcal{H}$ existe por lo menos una ϕ, se denota como $\mathcal{G} \cong \mathcal{H}$. En la Figura 2.6 los grafos $\mathcal{G}$ y $\mathcal{H}$ son isomorfos; un isomorfismo $\phi : V(\mathcal{G}) \to V(\mathcal{H})$ está definida por

$$\begin{aligned} &\phi(1) = d, \quad \phi(2) = b, \quad \phi(3) = f, \quad \phi(4) = a, \\ &\phi(5) = e, \quad \phi(6) = c, \quad \phi(7) = g. \end{aligned} \tag{2.4}$$



Grafo $\mathcal{G}$ *Grafo $\mathcal{H}$*

Figura 2.6: *Grafos isomorfos.*

2.2 SISTEMAS Y REDES COMPLEJAS

Un *sistema* es un conjunto de componentes relacionados entre sí para lograr un fin común. Los diferentes sistemas naturales o artificiales pueden ser simples, complicados o complejos [Amaral y Ottino, 2004]. Los *sistemas simples* tienen un número pequeño de componentes los cuales actúan de acuerdo a leyes bien comprendidas. Los *sistemas complicados* tienen un gran número de componentes los cuales tienen roles bien definidos y están gobernados por reglas bien comprendidas. Los *sistemas complejos* tienen típicamente un gran número de componentes los cuales pueden actuar de acuerdo a reglas que

[1]Una función $f\colon X \to Y$ es biyectiva si cada uno de los elementos del conjunto X tiene una imagen distinta en el conjunto Y, y a cada elemento de Y le corresponde un elemento del conjunto X.

pueden cambiar a través del tiempo y que pueden no ser bien comprendidas, la conectividad de los componentes y sus roles varían.

La *física estadística*[2] trata con sistemas complejos, en donde predecir el comportamiento exacto de los componentes individuales podría ser intratable, por tanto, se limita a realizar *predicciones estadísticas* acerca del comportamiento colectivo de los componentes. En la última década, se ha percibido que muchos sistemas formados por número muy grande de elementos pequeños obedecen leyes universales independientemente de los detalles microscópicos [Amaral *et al.*, 2001]. Diversos estudios utilizan grafos para modelar estos sistemas complejos dando lugar a las *redes complejas* [Bollobás y Riordan, 2002]. Es en este punto donde la teoría de grafos y la física estadística se combinan para proveer las bases necesarias que permitan el estudio de las redes complejas [da F. Costa *et al.*, 2007] enfocándose en tres puntos importantes [Kaufmann y Zweig, 2009; Novaes De Santana, 2005]:

1. ***Analizar*** redes complejas: encontrando y destacando propiedades estadísticas que *caractericen la estructura y el comportamiento* de la red compleja, ofreciendo una forma adecuada de medir esas propiedades.

2. Crear ***modelos*** de redes que capturen *propiedades estructurales* esenciales de las redes del mundo real y que ayuden a entender el significado de estas propiedades.

3. ***Procesos*** sobres redes complejas: analizando los resultados de los procesos o algoritmos que se llevan acabo en ellas

Los *métodos de análisis de redes* pueden ser clasificados con respecto al nivel de granularidad de los objetos analizados [Baur *et al.*, 2009]:

1. Métodos a ***nivel de elementos***: analizan propiedades *individuales* de nodos y aristas como lo es su importancia o *centralidad*.

2. Análisis a ***nivel de grupo***: incluye métodos que implican el cómputo de *grupos densamente conectados* y el cálculo de *roles estructurales y posiciones*.

[2]Para introducirse en esta área del conocimiento se recomienda el libro de Huang [2001].

3. Análisis a ***nivel de red***: se centra en analizar las *propiedades globales* de la red, tal como la densidad, la distribución del grado, transitividad o reciprocidad, así como el desarrollo *modelos de grafos aleatorios*.

2.2.1 FUNCIONES DE CARACTERIZACIÓN

Las *funciones de caracterización* son medidas que tienen como objetivo proveer en términos cuantitativos las propiedades estructurales de la grafo. Al aplicar las funciones de caracterización sobre un grafo $\mathcal{G}$ se obtiene un vector de características $f_{\mathcal{G}}$ [da F. Costa *et al.*, 2007] que nos permiten caracterizar y analizar las redes complejas.

Las funciones de caracterización pueden clasificarse en dos tipos:

- Basadas en ***información global***: requieren información de todo el grafo $\mathcal{G}$ para poder calcularlas.
- Basadas en ***información local*** [Dehmer, 2008]: requieren información de cierto nodo del grafo. Para ello se definen subgrafos en los cuales los nodos pueden ser aquellos que se encuentran a cierta distancia o el subgrafo inducido $\mathcal{S}_i^{\mathcal{G}}$ por el nodo i de interés y su vecindad $\Gamma(i)$, $V_{\mathcal{S}_i^{\mathcal{G}}} = \{i\} \cup \Gamma(i)$.

A continuación se enuncian las funciones de caracterización utilizadas en el presente trabajo.

DISTRIBUCIÓN DEL GRADO

El *grado* k_i de un nodo i es una medida basada en información local. Se define $\Pr(k)$ como la fracción de nodos en el grafo que tienen grado k, dicho de otro modo, $\Pr(k)$ es la probabilidad de que un nodo seleccionado aleatoriamente tenga grado k. Una gráfica de $\Pr(k)$ para cualquier grafo dado puede formarse haciendo un histograma de los grados de los nodos. Este histograma representa distribución del grado de un grafo [Newman, 2003]. Con base en lo anterior, se define a $X_k^{\mathcal{G}}$ como el número de nodos en $\mathcal{G}$ con grado k. Así,

la probabilidad de que un nodo en $\mathcal{G}$ tenga grado k se expresa por:

$$\Pr(k) = \frac{1}{n} X_k^{\mathcal{G}}, \tag{2.5}$$

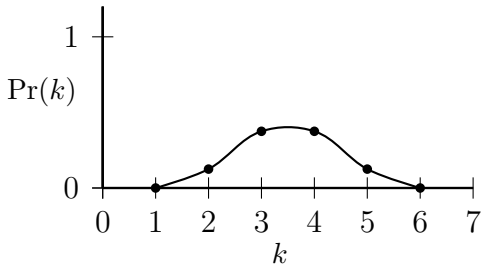
en la Figura 2.7 se muestra como obtener la distribución de grado para un grafo determinado.

i	1	2	3	4	5	6	7	8
k_i	5	4	3	4	3	2	3	4

(a) Grado de los nodos.

k_i	$X_k^{\mathcal{G}}$	$\Pr(k)$
5	1	0.125
4	3	0.375
3	3	0.375
2	1	0.125

(b) Datos para calcular $\Pr(k)$.



(c) Distribución del grado.

Figura 2.7: *Distribución del grado para el grafo de la Figura 2.2.*

COEFICIENTE DE AGRUPAMIENTO

El coeficiente de agrupamiento mide la tendencia a formar grupos [Albert y Barabási, 2002; Newman, 2003] en el grafo. Para calcularlo se define el coeficiente de agrupamiento del nodo i como la densidad del grafo inducido $\mathcal{S}_i^{\mathcal{G}}$ por la vecindad $\Gamma(i)$:

$$C(i) = \mathit{den}(\mathcal{S}_i^{\mathcal{G}}). \tag{2.6}$$

El coeficiente de agrupamiento del grafo $C(\mathcal{G})$ es el promedio del coeficiente de agrupamiento de los nodos de $\mathcal{G}$:

$$C(\mathcal{G}) = \frac{1}{n} \sum_{i \in V_{\mathcal{G}}} C(i). \tag{2.7}$$

LONGITUD DE RUTA MÁS CORTA

Una característica de interés en el estudio de redes complejas es la *longitud de ruta más corta* que caracteriza al grafo que representa a determinada red compleja [Albert y Barabási, 2002; Latora y Marchiori, 2001], la cual se define como el promedio de las distancias geodésicas:

$$L(\mathcal{G}) = \frac{1}{n(n-1)} \sum_{\substack{\{i,j\} \subseteq V_{\mathcal{G}} \\ i \neq j}} d_{i,j}. \tag{2.8}$$

DIÁMETRO

El diámetro del grafo $D(\mathcal{G})$, se refiere a la longitud más grande de las distancias geodésicas:

$$D(\mathcal{G}) = \max_{\substack{\{i,j\} \subseteq V_{\mathcal{G}} \\ i \neq j}} d_{i,j}. \tag{2.9}$$

EFICIENCIA

La *eficiencia del grafo* $E(\mathcal{G})$ es una medida de cómo la información es eficientemente *intercambiada* en todo el grafo [Latora y Marchiori, 2001]. La eficiencia ϵ_{ij} de comunicación entre los nodos i y j se define como $\epsilon_{i,j} = 1/d_{i,j}\ \forall i, j \in V_{\mathcal{G}}$. Cuando no existe una ruta entre i y j, $d_{i,j} = \infty$, la eficiencia se define como $\epsilon_{i,j} = 0$. La *eficiencia global* del grafo $E(\mathcal{G})$ se calcula promediando la eficiencia de comunicación entre cada par de nodos en $\mathcal{G}$:

$$E(\mathcal{G}) = \frac{1}{n(n-1)} \sum_{\substack{\{i,j\} \subseteq V_{\mathcal{G}} \\ i \neq j}} \epsilon_{ij}. \tag{2.10}$$

Se puede caracterizar las propiedades locales de $\mathcal{G}$ evaluando para cada subgrafo inducido $\mathcal{S}_i^{\mathcal{G}}$ la eficiencia global. Se define la *eficiencia local* como el promedio de la eficiencia de los subgrafos locales:

$$E_\ell = \frac{1}{n} \sum_{i \in V_{\mathcal{G}}} E(\mathcal{S}_i^{\mathcal{G}}). \tag{2.11}$$

La eficiencia local es similar al coeficiente de agrupamiento, debido a que manifiesta la tolerancia a fallas del sistema, mostrando que tan eficiente es la comunicación entre los vecinos del nodo i cuando éste es removido. Por otro lado la eficiencia global es una medida similar a la longitud de ruta ya que muestra la eficiencia con la cual se comunican entre sí dos nodos cualesquiera del grafo. La ventaja de utilizar estas dos medidas radica en que el estudio de la eficiencia en el intercambio de información cuando se presentan ataques o fallos se puede extender a grafos no conexos y/o ponderados, esto es analizado y discutido ampliamente por Latora y Marchiori [2001].

COEFICIENTE DE DISPERSIÓN DEL GRADO

El *coeficiente de dispersión del grado* (*ddc*) [Ortega-Izaguirre, 2008; Santillán *et al.*, 2010] se define como una función basada en información local, la cual tiene como objetivo medir la dispersión entre el grado de un nodo i y $\Gamma(i)$. El coeficiente de dispersión del grado para el nodo i se define como:

$$\mathit{ddc}(i) = \frac{\sigma(i)}{\mu(i)}, \tag{2.12}$$

y representa la variación del grado entre el nodo i y la suma de los grados de su conjunto vecino $\Gamma(i)$ donde,

$$\sigma(i) = \sqrt{\frac{1}{|\{i\} \cup \Gamma(i)|} \sum_{j \in \Gamma(i)} [k_j - \mu(i)]^2 + [k_i - \mu(i)]^2} \tag{2.13}$$

y

$$\mu(i) = \frac{1}{|\{i\} \cup \Gamma(i)|} \sum_{j \in \Gamma(i)} [k_j] + k_i, \tag{2.14}$$

representa la variación del grado entre el nodo i y $\Gamma(i)$, $\mu(i)$ es el grado promedio encontrado en el conjunto $\{i\} \cup \Gamma(i)$, k_i y k_j representan el grado de i y j respectivamente. Para calcular el coeficiente de dispersión del grado *global*, es decir para toda el grafo, se utiliza la siguiente fórmula:

$$\mathit{ddc}(\mathcal{G}) = \frac{1}{n} \sum_{i \in V_{\mathcal{G}}} \mathit{ddc}(i), \tag{2.15}$$

donde n es el orden del grafo.

2.2.2 TIPOS DE REDES COMPLEJAS

Las redes complejas pueden representar diversos sistemas complejos ya sea biológicos, tecnológicos y sociales; sin embargo comparten ciertas propiedades además de su gran tamaño. Estas propiedades son:

- ***Tienden a ser esparcidas:*** tienen baja densidad, considerando el número de nodos. En grafos con n nodos, el máximo número aristas es $n(n-1)/2$ [Hayes, 2000]. En las redes del mundo real el número de aristas es generalmente más cercano a n que a $n(n-1)/2$.

- ***Tienden a ser agrupadas:*** las aristas en la red no están distribuidas uniformemente sino tienden a formar grupos [Hayes, 2000]. Por ejemplo en las redes sociales si dos personas están relacionadas con un tercero, es muy posible que esas dos personas se conozcan mutuamente.

- ***Tienden a tener diámetro pequeño:*** un grafo conexo debe tener al menos $n-1$ aristas y el diámetro más largo posible es $n-1$. Por otra parte un grafo completo con $n(n-1)/2$ aristas tiene un diámetro de 1. Sería de esperarse que grafos con un número de aristas cercano al mínimo número de aristas en la red tengan un gran diámetro. Sin embargo en las redes complejas el diámetro suele estar alrededor de $\log n$, valor que es mucho más pequeño que n [Albert *et al.*, 2000]. Considerando la definición de diámetro dada anteriormente, es claro que en estas redes la distancias entre los nodos tienden a ser cortas; mientras la red crece en algunos casos el díametro disminuye [Leskovec *et al.*, 2005].

En resúmen, las redes complejas que representan sistemas del mundo real presentan un alto coeficiente de agrupamiento y una longitud característica pequeña [Watts y Strogatz, 1998]. La presencia de estas dos propiedades en las redes complejas es conocida como efecto de *pequeño mundo* (en inglés: *small world*). La manifestación más popular de *pequeño mundo* es el concepto de "seis grados de separación" reportado por Milgram [1967] quien realizó un experimento para examinar la longitud de ruta de conocidos entre pares de personas en los Estados Unidos.

Independientemente de estas propiedades comunes, las redes complejas se pueden diferenciar a través de la distribución del grado, la cual proporciona una idea acerca de la conectividad más no de cómo está estructurada. La idea de tener redes de diferente estructura con la misma distribución del grado es analizada por Alderson [2008] y se

ejemplifica en la Figura 2.8. Sin embargo aún con esta desventaja evidente la clasificación de las redes complejas en la literatura existente [López, 2007], se realiza de acuerdo con la distribución del grado, clasificándolas en *redes aleatorias, redes de libre escala* y *redes exponenciales*.

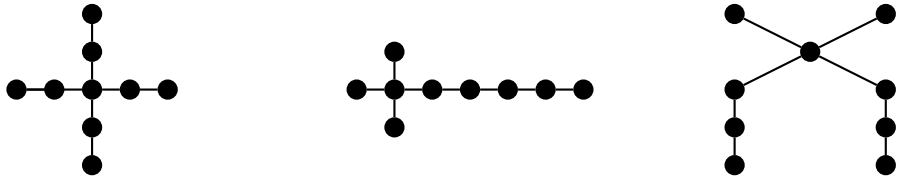


Figura 2.8: *En estos tres grafos se puede observar que la distribución del grado es la misma; en los tres se tiene uno nodo con grado cuatro, cuatro nodos con grado dos y cuatro nodos con grado uno, pero la estructura en estos tres grafos es diferente.*

REDES ALEATORIAS

Como se mencionó al inicio de este capítulo, la teoría de grafos se originó en el siglo XVIII con los trabajos de Euler [1741] acerca de la solución del problema de los puentes de Königsberg. En el siglo XX la teoría de grafos se volvió más estadística y algorítmica. Una particular fuente de ideas es el estudio de *grafos aleatorios* en los cuales las aristas se distribuyen aleatoriamente.

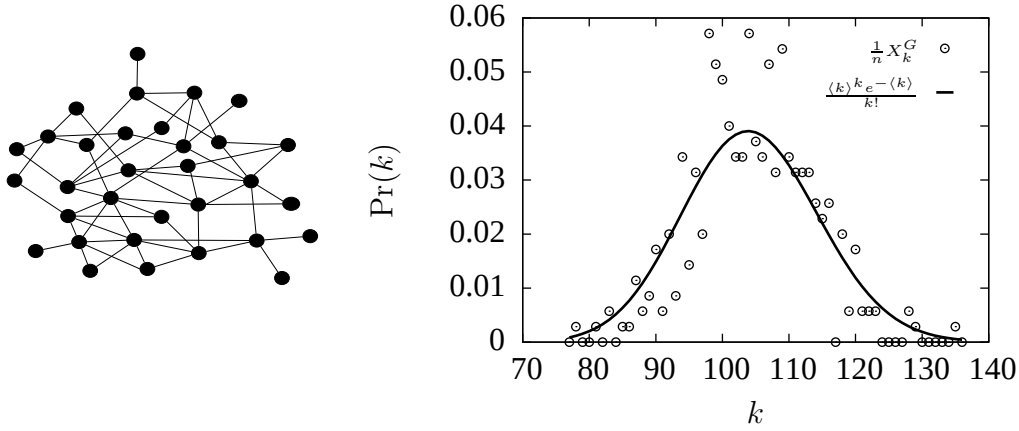
La teoría de grafos aleatorios comenzó su desarrollo a finales de los 50's y comienzos de los 60's. Gilbert [1959] introdujo su *modelo de grafos aleatorios* $\mathcal{G}_{n,p}$ donde $\{X_{i,j} : 1 \leq i \leq j \leq n\}$ es un arreglo de variables aleatorias de Bernoulli idéntica e independientemente distribuidas con $\Pr(X_{ij} = 1) = p$ y $\Pr(X_{ij} = 0) = 1 - p$, por tanto $\mathcal{G}_{n,p}$ es un grafo con n nodos en el cual los nodos i y j son adyacentes, si $X_{ij} = 1$ [Bollobás, 2001]. En otras palabras, se fija el número de nodos n y se añaden aristas con una probabilidad p independientemente una de otra. Los grafos generados con este enfoque son llamados *grafos aleatorios uniformes*, ya que la probabilidad de que dos nodos i, j estén conectados es uniforme.

En estos grafos aleatorios, la probabilidad $\Pr(k)$ de que un nodo tenga grado k sigue una *distribución binomial* con parámetros $n - 1$ y p:

$$\Pr(k) = C_{n-1}^{k} p^k (1-p)^{n-1-k} = \binom{n-1}{k} p^k (1-p)^{n-1-k}. \tag{2.16}$$

Para valores grandes de n, la distribución del grado sigue aproximadamente una *distribución de Poisson*, de ahí que la probabilidad de que un nodo tenga grado k es:

$$\Pr(k) \sim \frac{\langle k \rangle^k e^{-\langle k \rangle}}{k!}. \tag{2.17}$$



(a) *Estructura de una red aleatoria.*

(b) *Gráfica de la distribución del grado.*

Figura 2.9: *En la Figura 2.9(a) se aprecia la estructura de una red aleatoria donde se puede observar como cada nodo se relaciona aproximadamente con la misma cantidad de nodos. En la Figura 2.9(b) se muestra la distribución del grado obtenida para una red aleatoria con* 350 *nodos,* $9{,}157$ *aristas,* $\delta(\mathcal{G}) = 83$*,* $\Delta(\mathcal{G}) = 131$*,* $\langle k \rangle = 104$*. Los círculos representan la distribución del grado real, la línea continua representa la distribución del grado calculada con la Ecuación 2.17.*

Dado que una red aleatoria las aristas entre los nodos son establecidas aleatoriamente, la mayoría de los nodos tiene aproximadamente el mismo grado, cercano al grado promedio de la red $\langle k \rangle$. En la Figura 2.9 se muestra un ejemplo de la estructura de una red aleatoria y la distribución del grado.

REDES DE LIBRE ESCALA

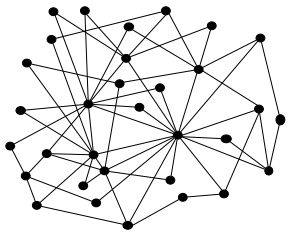
En la década de los 90's diversos investigadores como Albert *et al.* [2000] y Faloutsos *et al.* [1999] descubrieron que la distribución del grado en las redes del mundo real como el World Wide Web, Internet, redes de proteínas y metabolismo, las redes de lenguaje[3] y sociales, difieren a la distribución de Poisson, exhibiendo una distribución del

[3]Red formada por palabras conectadas por relaciones sintácticas.

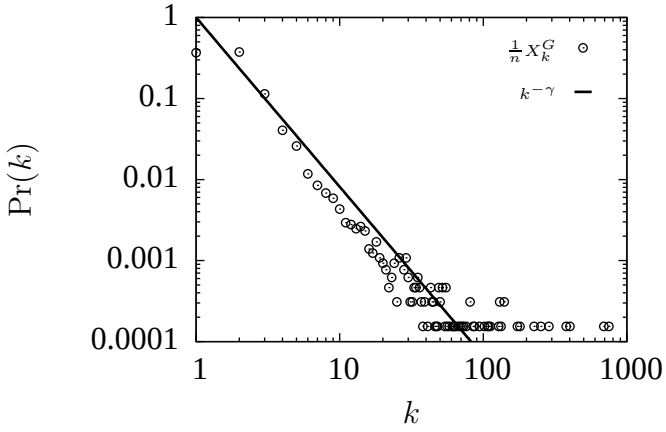
grado de *ley de potencias* (en inglés: *power-law*):

$$P(k) \sim k^{-\gamma}. \tag{2.18}$$

Las redes este tipo de distribución son llamadas de *libre escala* (en inglés: *scale-free*), debido a que independientemente de la escala, es decir del número de nodos, la distribución del grado no cambia. La característica típica de estas redes es que un conjunto reducido de nodos que conforman la red tienen un gran numero de enlaces (un grado muy alto) y resto de los nodos tienen pocos enlaces (un grado pequeño) [Barabási, 2005; Newman, 2003]. Esta característica permite que la tolerancia a fallas aleatorias sea grande, pero si los nodos con grado muy alto llamados nodos centrales son atacados intencionalmente esta red es muy vulnerable [Albert *et al.*, 2000].



(a) *Estructura de una red de libre escala.*



(b) *Gráfica de la distribución del grado.*

Figura 2.10: *En la Figura 2.10(a) se aprecia la estructura de una red de libre escala; en este tipo de red pocos nodos tienen un alto grado y la mayoría de los nodos tiene grado pequeño. En la Figura 2.10(b) se muestra la distribución del grado obtenida para una red de libre escala con* $6{,}474$ *nodos,* $12{,}572$ *aristas,* $\delta(\mathcal{G}) = 1$*,* $\Delta(\mathcal{G}) = 1{,}458$*,* $\langle k \rangle = 3.8$*. Los círculos representan la distribución del grado real, la línea continua representa la distribución del grado obtenida mediante mínimos cuadrados para la Ecuación 2.18 con* $\gamma = 2.09$*.*

En esta distribución el parámetro γ decrece desde ∞ hasta 0, siendo un parámetro de control que describe que tan rápido decae la frecuencia de aparición del grado k, de manera que el grado promedio de la red se incrementa a medida que γ se decrementa. En este caso $P(k)$ no tiene un pico definido, y para una k grande decae aproximadamente como una serie de potencias, apareciendo como una línea recta en una gráfica en escala

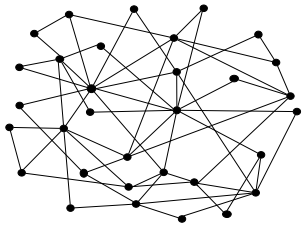
logarítmica en ambos ejes, con poco ruido al inicio y mucho ruido al final. Esto se ejemplifica en la Figura 2.10(b) donde se hace un ajuste por medio de mínimos cuadrados, omitiendo ese ruido para obtener el valor de γ; si el ruido no se omitiera al hacer el ajuste, la línea mostraría una menor pendiente.

EXPONENCIALES

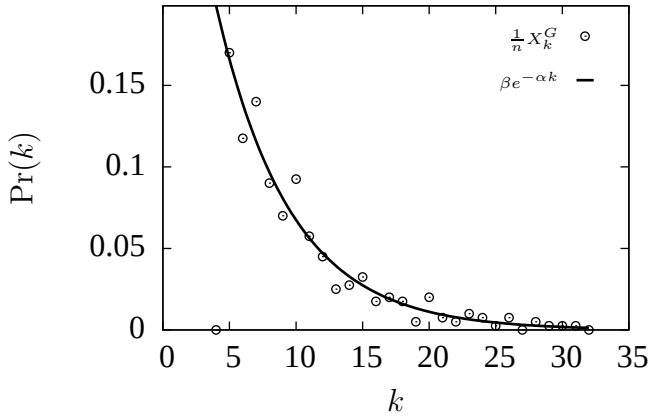
Aunque las redes de libre escala son comunes, también existen casos de redes que exhiben una distribución del grado exponencial como la red de energía del sur de California [Amaral *et al.*, 2000] y la red de vías de ferrocarril de la India [Sen *et al.*, 2003]; a estas redes se les llama redes *exponenciales*.

En estas redes la distribución del grado $P(k)$ muestra un pico en $\langle k \rangle$ y después cae exponencialmente para valores grandes de k:

$$P(k) \approx \beta e^{-\alpha k}. \tag{2.19}$$



(a) *Estructura de una red exponencial.*



(b) *Gráfica de la distribución del grado.*

Figura 2.11: *En la Figura 2.11(a) se aprecia la estructura de una red exponencial. En la Figura 2.11(b) se muestra la distribución del grado obtenida para una red aleatoria con* 400 *nodos,* $1{,}982$ *aristas,* $\delta(\mathcal{G}) = 5$*,* $\Delta(\mathcal{G}) = 31$*,* $\langle k \rangle = 9.9$*. Los círculos representan la distribución del grado real, la línea continua representa la distribución del grado obtenida realizando un ajuste de mínimos cuadrados para la Ecuación 2.19 donde* $\beta = 0.4$ *y* $\alpha = 0.18$

Este tipo de red no es muy tolerante a fallas aleatorias; al eliminar cualquier nodo el daño es similar, debido a que cada nodo en la red se relaciona aproximadamente con la misma cantidad de nodos [Albert *et al.*, 2000]. En la Figura 2.11(b) se hace un

ajuste de mínimos cuadrados para obtener la curva que describe el comportamiento de la distribución exponencial, ignorando cualquier ruido que se pueda tener.

2.2.3 MODELOS DE GENERACIÓN DE REDES COMPLEJAS

En un principio los modelos de generación de grafos aleatorios no tenían como objetivo capturar propiedades de las redes del mundo real; más bien se enfocaban en analizar la existencia de ciertas estructuras y comportamiento de los algoritmos aplicados a la problemas de grafos. Sin embargo hoy en día los modelos de generación son una herramienta importante que buscan reproducir grafos que comparten ciertas propiedades de las redes del mundo real para analizar y comprender procesos que se llevan a cabo en estas redes [Barabási y Albert, 1999]. A continuación se describen cinco modelos de generación de grafos que buscan capturar ciertas características de las redes del mundo real y que son de interés para el desarrollo del presente trabajo.

MODELO DE ERDŐS-RÉNYI (ER)

El modelo de Erdős y Renyí [1960] comienza fijando el número de nodos n y el número de aristas e que tendrá el grafo. Al principio se tienen los n nodos desconectados. En el siguiente paso se seleccionan uniformente al azar e aristas de las $\binom{n}{2}$ posibles. El modelo formal es conocido como el modelo $\mathcal{G}_{n,e}$; en el presente trabajo se hace referencia a los grafos generados con este modelo como grafos ER.

MODELO DE WATTS-STROGATZ (WS)

Watts y Strogatz [1998] reevaluaron el empleo de grafos aleatorios como modelos de redes del mundo real, ya que hasta entonces se asumía que la conexión entre los nodos era regular o aleatoria. Sin embargo observaron que muchas redes biológicas, tecnológicas y sociales se encontraban en algún lugar entre estos dos extremos, véase Figura 2.12. Estas redes tenían dos características particulares un *alto coeficiente de agrupamiento*, como una red regular, y una *longitud de ruta pequeña*, como una red aleatoria. Para lograr reproducir este tipo de redes, propusieron un modelo donde existe un proceso de reconexión sobre una red regular y nombraron a estas redes *pequeño mundo*.

El modelo comienza fijando el número de nodos n los cuales son colocados en un círculo. Cada nodo es conectado con los k vecinos cercanos a su izquierda y derecha

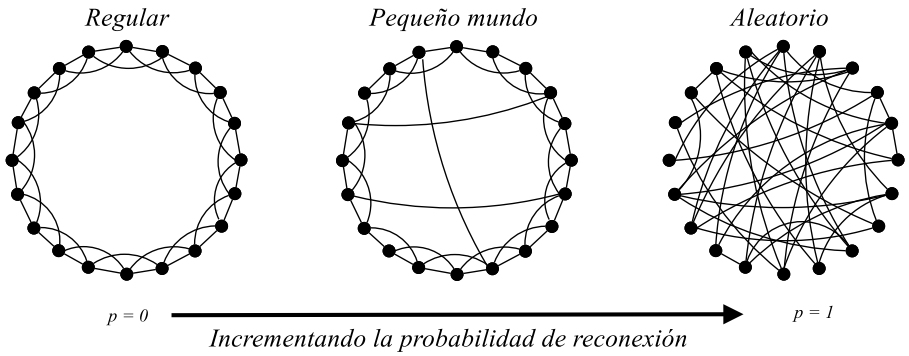


Figura 2.12: *El proceso de reconexión tiene como objetivo interpolar entre una red regular y una red aleatoria, sin alterar el número de nodos y aristas.*

formando un anillo; esto produce un alto coeficiente de agrupamiento. Para introducir una longitud de ruta pequeña se selecciona una probabilidad de *reconexión* p_r muy pequeña y para cado nodo v, con una probabilidad p_r, se reconecta uniformemente al azar una arista de v con un nodo del conjunto $V_{\mathcal{G}}$. En el presente trabajo a los grafos generados con este modelo se denotan como grafos WS.

MODELO BARABÁSI-ALBERT (BA)

En 1999 un grupo de investigadores [Barabási y Albert, 1999] observaron una característica que no era reproducida por los grafos aleatorios ni los grafos de pequeño mundo esta característica tiene que ver con la distribución del grado. En los grafos aleatorios y *pequeño mundo* la distribución del grado siguen una distribución de Poisson, pero las redes del mundo real presentan una distribución que se aproxima a la *ley de potencias*. Para reproducir grafos con esta característica, Barabási y Albert [1999] introducen un modelo basado en *crecimiento* y *enlace preferencial*.

El modelo supone la existencia de un grafo inicial con n_0 nodos y e_0 aristas. A cada paso t se añade un nuevo nodo que se conectará con nodos ya existentes en el grafo: cada nuevo nodo se conectará con m aristas al grafo donde $m \leq n_0$. La manera en que se agregan las aristas es otorgando cierta preferencia a los nodos existentes en el grafo dependiendo de sus grados. Se asume que la probabilidad Π de que un nuevo nodo sea enlazado al nodo i depende del grado k_i, tal que:

$$\Pi(k_i) = \frac{k_i}{\sum_{j \in V_{\mathcal{G}}} k_j}. \tag{2.20}$$

Después de t lapsos de tiempo, el procedimiento resulta en una red con $n = t + n_0$ nodos y con $e = m \times t + e_0$ aristas. En el presente trabajo a los grafos generados con este modelo se denotan como grafos BA.

MODELO DE KLEINBERG (KL)

El modelo propuesto por Kleinberg [2000] pretende capturar el aspecto algorítmico de la investigación de Milgram [1967]; sigue el paradigma de Watts y Strogatz [1998] de tener muchas conexiones locales y unas cuantas conexiones de largo alcance. A diferencia del modelo de Watts y Strogatz, en lugar de usar un anillo como estructura básica, se tiene una rejilla de dos dimensiones. Se comienza con un n nodos colocados en una rejilla de $s \times s$ puntos. Cada nodo se identifica por su posición en la rejilla $v = (i, j), i, j \in \{1, 2, \ldots, s\}$. Las *conexiones locales* de un nodo v se establecen con todos los nodos ubicados en un radio $p > 1$, usando la *distancia de Manhattan*:

$$\text{dist}(v, u) = \text{dist}((i, j), (k, l)) = |k - i| + |l - j|. \tag{2.21}$$

Para la conexiones de largo alcance se fija un número $q > 0$ aristas que serán asignadas a cada nodo $v \in V_G$ aleatoria e independientemente, cada arista se elige con una probabilidad proporcional a $\text{dist}(v, w)^{-r}$, donde $r \geq 0$; no se permiten aristas duplicadas. En este trabajo a los grafos generado con este modelo se denotan como grafos KL .

MODELO DE GRAFOS GEOMÉTRICOS ALEATORIOS (RGG)

Los grafos *geométricos aleatorios* o RGG por sus siglas (en inglés: *random geometric graphs*) — estudiados por Penrose [2003] — tienen sus orígenes en los trabajos de Gilbert. Recientemente han sido de gran relevancia para modelar redes sensoras e inalámbricas [Avin, 2006].

El modelo para generar este tipo de grafos comienza fijando el número de nodos n y un radio de conexión $r < 1$. A cada nodo se le asigna uniformemente al azar un punto P de coordenada (x, y) dentro de un cuadro unitario. Para establecer las conexiones entre los nodos, se toman las coordenadas de los nodos para calcular la distancia euclidiana:

$$\text{dist}(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}. \tag{2.22}$$

Dos nodos se conectan si su distancia euclidiana es menor o igual al radio de conexión r. Por esta razón se dice que un grafo RGG es un grafo aleatorio con una métrica [Avin, 2006].

2.3 COMPLEJIDAD COMPUTACIONAL

La *complejidad computacional* es una rama de la teoría de la computación que se encarga de clasificar problemas computacionales de acuerdo a su dificultad. Para estudiar la dificultad de los problemas es necesario tener claros conceptos como problema, instancia y algoritmo. Un *problema* es un pregunta que requiere respuesta y posee uno o más parámetros. Un problema está definido por una descripción detallada de sus parámetros y un enunciado acerca de qué propiedades se deben satisfacer con la respuesta que se dé. Una *instancia* del problema se obtiene al especificar valores particulares a los parámetros del problema, de ahí que un problema tiene asociado un conjunto de instancias que posiblemente sea infinito. Un *algoritmo* es un procedimiento formal para encontrar la respuesta correcta a la pregunta de un problema para cierta instancia del problema.

Por ejemplo el problema de *satisfactibilidad booleana* (SAT), donde dada una expresión booleana, se desea saber si existe o no una asignación de verdad para las variables que haga la expresión verdadera. Una instancia de este problema es una expresión booleana específica. Otro problema computacional interesante es el ***coloreo de grafos***. El objetivo de este problema es asignar distintos "colores" (generalmente representado por números enteros) a los nodos de un grafo, de manera que dos nodos unidos por una arista tengan colores diferentes [Chartrand y Zhang, 2008]. Una asignación de colores de este tipo se denomina un *coloreo legal*.

En la teoría de complejidad computacional [Papadimitriou, 1994], los problemas computacionales que se abordan son problemas llamados *problemas de decisión* que requieren como respuesta *"sí"* si existe solución al problema o *"no"* si no existen soluciones. También aborda *problemas de optimización* de los cuales se hablará en la Sección 2.5. Para definir términos relacionados con la teoría complejidad computacional en esta sección se trabajará con problemas de decisión; el problema SAT mencionado anteriormente es un ejemplo de un problema de decisión.

La dificultad de un problema está asociada a la cantidad de recursos computacionales ya sea el tiempo de cómputo o el espacio en memoria que necesita un algoritmo para

decidir el problema, es decir, que lo resuelva. Generalmente se mide el *tiempo de cómputo* o el *espacio en memoria* en función del tamaño de la instancia.

Una *clase de complejidad* está conformada por un conjunto de problemas decisión que comparten características acerca de cómo son resueltos. Una clase de complejidad se define de acuerdo a ciertos parámetros, los cuales se detallan en subsecuentes secciones. Estos son:

- Un ***modelo de cómputo***, tal como una máquina de Turing.
- El ***modo*** de llevar a cabo el cómputo: determinista o no determinista.
- El ***recurso*** que se desea acotar, ya sea tiempo o espacio.
- Una ***función*** cota que toma como parámetro el tamaño de la instancia n.

Por lo regular es de interés que la *tasa de crecimiento* de las funciones que definen la cota en función del tamaño de la instancia sea *polinomial*; esto nos indica requerimientos de *tiempo de cómputo aceptables*. En contraste, *tasas de crecimiento exponencial* causan preocupación y podríamos llegar a decir que el problema es intratable, es decir, *no tener una solución práctica* incluso en instancias del peor caso de un tamaño pequeño.

Otra rama de la teoría de la computación es la teoría de *complejidad parametrizada* [Flum y Grohe, 2006], en la cual se define una cota en función del tamaño de la instancia y un parámetro del problema.

A continuación se abordan ciertos temas fundamentales para comprender cómo se realiza la clasificación de problemas en clases de complejidad como **P** o **NP** y además comprender conceptos de la teoría de complejidad parametrizada.

2.3.1 ANÁLISIS DE ALGORITMOS

Cuando se tiene un problema es importante estimar la cantidad de tiempo de procesamiento y espacio en memoria que un algoritmo necesitará para resolverlo. Para ello es necesario estimar teóricamente estos recursos, principalmente el tiempo de cómputo, el cual está asociado con el tiempo que lleva a un algoritmo realizar una operación básica.

La *eficiencia computacional* de un algoritmo se mide como el *número de operaciones básicas* que realiza en función del tamaño de la instancia n. Generalmente para

analizar un algoritmo se aisla una operación específica que sea fundamental para el problema que se estudia. Por ejemplo, para el problema de ordenar un arreglo, las operaciones básicas son las comparaciones y asignaciones.

Además de las operaciones básicas que se tomen en cuenta y del tamaño de la instancia, se debe considerar la "naturaleza de la instancia". Por ejemplo un algoritmo para ordenar números podría efectuar poco trabajo si esos números están parcialmente ordenados en comparación a que están totalmente desordenados que sería el *peor caso*. Generalmente se estima los recursos a utilizar considerando el comportamiento del algoritmo en el *peor caso* de una instancia.

Al asociar una unidad de tiempo a cada operación básica, se tendrá el tiempo de cómputo $T(n)$ y es de interés particular calcular su "complejidad" en *sentido asintótico*, es decir, para un tamaño de instancia suficientemente grande, cuando n tiende a infinito.

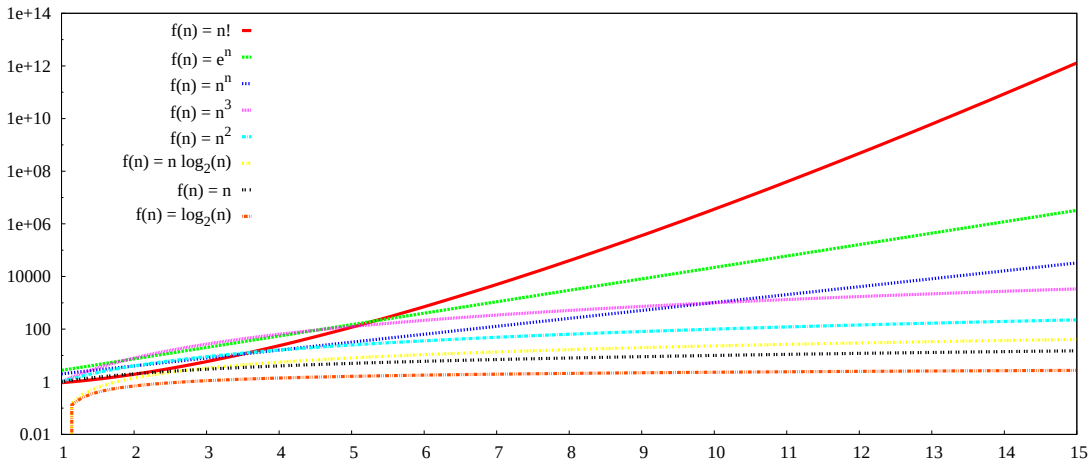


Figura 2.13: *Crecimiento de funciones comúnmente encontradas en el análisis de algoritmos.*

Cuando se estudia el comportamiento asintótico, se estudia la *velocidad de crecimiento* de $T(n)$ conforme el tamaño de la instancia aumenta asociando a $T(n)$ funciones $f(n)$ que caracterizan su comportamiento. La importancia de realizar este análisis en sentido asintótico se ilustra en la Figura 2.13 en donde se muestran algunas funciones comúnmente encontradas en el análisis de algoritmos. Se puede observar que para valores pequeños de x el orden de crecimiento de las funciones no es apreciable, por lo cual es importante apreciar el orden de crecimiento para valores suficientemente grandes de x. Para hacer referenciar al comportamiento asintótico de los algoritmos se definen las siguientes notaciones:

Definición 2.1 *Sean $f(n)$ y $g(n)$ funciones de $\mathbb{N} \to \mathbb{R}$:*

(a) $f(n) \in \mathcal{O}(g(n))$ *si* $\exists c > 0, n_0 \geq 0$ *tal que* $\forall n \geq n_0, |f(n)| \leq c \cdot |g(n)|$. *Se tiene entonces que $g(n)$ es una cota* ***superior****.*

(b) $f(n) \in \Omega(g(n))$ *si* $\exists c > 0, n_0 \geq 0$ *tal que* $\forall n \geq n_0, |f(n)| \geq c \cdot |g(n)|$. *Se tiene entonces que $g(n)$ es una cota* ***inferior****.*

(c) $f(n) \in \Theta(g(n))$ *si* $\exists c, c' > 0, n_0 \geq 0$ *tal que* $\forall n \geq n_0, c \cdot |f(n)| \leq |g(n)| \leq c' \cdot |f(n)|$. *Se tiene entonces que $g(n)$ es una cota* ***estrecha****.*

La notación $\mathcal{O}(f(n))$ indica que el tiempo de cómputo de un algoritmo está a lo más caracterizado por $f(n)$, siendo ésta la cota que nos interesa estudiar cuando se habla de complejidad computacional. Al seleccionar o diseñar algoritmos para solucionar un problema es importante considerar tasas de crecimiento del tiempo de cómputo sean polinomiales o de orden inferior en lugar de tasas de crecimiento exponenciales o factoriales, sin embargo aún con tasas de crecimiento polinomial para instancias del problema muy grandes el tiempo de procesamiento tiende a ser mayor. Por ejemplo en la Tabla 2.1 se muestra para diferentes valores de n el tiempo de ejecución para un algoritmo que necesita $f(n)$ operaciones básicas si cada operación básica toma un microsegundo de ejecución; es importante notar que para tamaños de entrada pequeñas el tiempo de ejecución para algoritmos con comportamiento $\mathcal{O}(2^n)$ es de segundos pero conforme va creciendo el tamaño de entrada los tiempos de ejecución — en el peor de los casos — llevaría siglos incluso más tiempo que la edad calculada del universo, lo cual se marca en la tabla como ∞, aunque no es en sí infinito sino un valor extremadamente grande.

Tabla 2.1: *Tiempo que se llevará un algoritmo con comportamiento $f(n)$ para entradas de tamaño n, si cada operación básica toma un microsegundo de ejecución.*

$f(n)$ \ n	10	100	1,000	10,000
$\log n$	4×10^{-6} seg	2×10^{-6} seg	3×10^{-6} seg	4×10^{-6} seg
n	1×10^{-5} seg	1×10^{-4} seg	1×10^{-3} seg	1×10^{-2} seg
$n \log n$	1×10^{-5} seg	2×10^{-4} seg	3×10^{-3} seg	4×10^{-2} seg
n^3	1×10^{-3} seg	1 seg	16.67 min	11.57 días
2^n	1×10^{-3} seg	4×10^{14} siglos	3×10^{285} siglos	"∞"

Definir las operaciones básicas de un algoritmo es adecuado cuando se requiere analizar algoritmos específicos o comparar algoritmos. En la teoría de complejidad compu-

tacional las operaciones básicas son definidas por un modelo matemático; uno de estos modelos es la máquina de Turing.

2.3.2 MÁQUINAS DE TURING

La *máquina de Turing* es un modelo formal simple pero suficiente para estudiar muchos aspectos de la computación como su eficiencia y es capaz de expresar cualquier algoritmo. La máquina de Turing usa una sola estructura de datos, una cadena de símbolos en una cinta en la cual un cursor realiza operaciones como leer un símbolo de la cadena, desplazarse sobre la cinta y escribir un nuevo símbolo en la posición actual, las operaciones a realizar se basan en un conjunto de "reglas".

Formalmente una máquina de Turing se define como una tupla M = (Q, Σ, δ, q_0) donde:

- Q: un conjunto finito de *estados*.
- Σ: un conjunto finito de *símbolos* que siempre contiene el símbolo en *blanco* '$\sqcup$' y el símbolo de *comienzo* '$\triangleright$'; a este conjunto también se le conoce como *alfabeto*.
- δ: es una *función de transición* que mapea $Q \times \Sigma \rightarrow (Q \cup \{h,$ "si", "no"$\}) \times \sigma \times \{\leftarrow$ (izquierda), $\rightarrow$ (derecha), $-$(no moverse)$\}$.
- q_o: es el *estado inicial* $q_o \in Q$.

Se asume que el estado h es *estado de paro* (en inglés: *halt*), "sí" es el *estado de aceptación* y "no" el *estado de rechazo*. En pocas palabras δ es una regla que especifíca para cada combinación de un estado actual $q \in Q$ y un símbolo actual $\sigma \in \Sigma$ una transición a un estado p, un símbolo ρ con el cual se sobreescribe σ y el cursor se mueve hacia una dirección o se queda en la posición actual; esto es $\delta(q, \sigma) \rightarrow (p, \rho, D)$, donde $D = \{\leftarrow, \rightarrow, -\}$ son las direcciones a las cuales se puede mover el cursor que lee los símbolos. La máquina siempre realiza una transición hasta que se alcanza algún estado de paro $\{h,$ "sí", "no"$\}$; si es así se dice que la máquina M ha parado.

Denotamos la cadena de entrada para la máquina de Turing como $x \in \Sigma^*$. El símbolo $*$ es la cerradura de Kleene; A^* es el conjunto formado por todas las cadenas posibles que pueden formarse con los símbolos en A, incluida la cadena de longitud cero, es decir, la cadena vacía. Dado x, si M llega al estado "sí", se escribe $M(x) =$ "sí" y se dice que M *acepta* la entrada x. Si se llega al estado "no", $M(x)$ = "no", se dice que M *rechaza* la

entrada x. Si h se ha alcanzado la salida es la cadena de M al tiempo del paro y se denota como y, $M(x) = y$. Si M nunca llega a un estado de paro con la cadena x, se escribe $M(x) =$"↗".

Como ejemplo considerese una máquina M donde:

$$\begin{aligned} Q &= \{s, q, q_0, q_1\}, \\ \Sigma &= \{0, 1, \sqcup, \triangleright\}, \\ q_o &= s, \\ x &= 010, \end{aligned} \tag{2.23}$$

y δ está definida en la Figura 2.14 donde también se muestra la computación de la cadena x. Esta máquina M simplemente inserta un espacio en blanco entre el símbolo de inicio y la cadena de entrada x. Cada combinación de estado, posición del cursor y símbolo en la cinta de la máquina de Turing se le conoce como *configuración*. Formalmente una configuración de M es una tripleta (q, w, v), donde $q \in Q$ es un estado y $w, v \in \Sigma^*$, w es la cadena del lado izquierdo del cursor y v es la cadena del lado derecho del cursor incluyendo la posición donde se encuentra el cursor. Cuando se va de una configuración a otra se dice que se ha dado un *paso*; en este ejemplo, de la configuración inicial a la final se dieron 15 pasos.

$q \in Q$	$\sigma \in \Sigma$	$\delta(q, \sigma)$
s	0	$(s, 0, \rightarrow)$
s	1	$(s, 1, \rightarrow)$
s	$\sqcup$	$(q, \sqcup, \leftarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
q	0	$(q_0, \sqcup, \rightarrow)$
q	1	$(q_1, \sqcup, \rightarrow)$
q	$\sqcup$	$(q, \sqcup, -)$
q	$\triangleright$	$(h, \triangleright, \rightarrow)$
q_0	0	$(s, 0, \leftarrow)$
q_0	1	$(s, 0, \leftarrow)$
q_0	$\sqcup$	$(s, 0, \leftarrow)$
q_0	$\triangleright$	$(h, \triangleright, \rightarrow)$
q_1	0	$(s, 1, \leftarrow)$
q_1	1	$(s, 1, \leftarrow)$
q_1	$\sqcup$	$(s, 1, \leftarrow)$
q_1	$\triangleright$	$(h, \triangleright, \rightarrow)$

0.	s,	$\underline{\triangleright}010$
1.	s,	$\triangleright\underline{0}10$
2.	s,	$\triangleright0\underline{1}0$
3.	s,	$\triangleright01\underline{0}$
4.	s,	$\triangleright010\underline{\sqcup}$
5.	q,	$\triangleright01\underline{0}\sqcup$
6.	q_0,	$\triangleright01 \sqcup \underline{\sqcup}$
7.	s,	$\triangleright01\underline{\sqcup}0$
8.	q,	$\triangleright0\underline{1} \sqcup 0$
9.	q_1,	$\triangleright0 \sqcup \underline{\sqcup}0$
10.	s,	$\triangleright0\underline{\sqcup}10$
11.	q,	$\triangleright\underline{0} \sqcup 10$
12.	q_0,	$\triangleright \sqcup \underline{\sqcup}10$
13.	s,	$\triangleright\underline{\sqcup}010$
14.	q,	$\underline{\triangleright} \sqcup 010$
15.	h,	$\triangleright\underline{\sqcup}010$

Figura 2.14: *En la parte izquierda tenemos la defincón de una máquina de Turing y a la derecha su computación el símbolo "_" indica la posición del cursor en la cadena que se está procesando.*

Las máquinas de Turing son ideales para resolver cierto tipos de problemas especializados sobre cadenas, como aceptar o decidir *lenguajes*. De hecho, los problemas de decisión pueden describirse como lenguajes sombre alfabetos finitos. A continuación se listan unas definiciones de utilidad.

Sea $L \subset (\Sigma \setminus \{\sqcup\})^*$ un *lenguaje*, es decir, un conjunto de cadenas de símbolos. Sea M una máquina de Turing tal que para cada cadena $x \in (\Sigma \setminus \{\sqcup\})^*$. Decimos que M *decide* L si y solo si $x \in L$, entonces $M(x) =$ "sí"; si $x \notin L$ entonces $M(x) =$ "no". Si M decide L entonces L es un *lenguaje recursivo*.

Una máquina Turing *acepta* un lenguaje L si para toda sucesión $x \in (\Sigma \setminus \{\sqcup\})$ aplica que si $x \in L, M(x) =$ "sí", pero si $x \notin L, M(x) =$ " $\nearrow$". Los lenguajes aceptados por alguna máquina Turing son *recursivamente numerables*.

Las *funciones computables* son el objeto básico de estudio de la teoría de la computabilidad y consisten en las funciones que pueden ser calculadas por una máquina de Turing. Suponga que $f : (\Sigma \setminus \{\sqcup\})^* \to \Sigma^*$ y sea M una máquina de Turing con alfabeto Σ, se dice que M *computa* a f si $\forall x \in (\Sigma \setminus \{\sqcup\})^*$, $M(x) = f(x)$. Si tal M existe, f es llamada *función recursiva*.

Para definir formalmente el tiempo y el espacio requerido por una máquina de Turing para hacer computaciones, se debe introducir una generalización de esta. La máquina de Turing con *múltiples cadenas* o más bien máquina de Turing con k cadenas, donde $k \geq 1$. La máquina se define como anteriormente se hizo, la función de transición decide el siguiente estado igual que antes, solo que ahora decide para cada cadena el símbolo a escribir y la dirección del cursor viendo el estado y símbolo actual de cada cadena.

El resultado de la computación de una máquina de Turing de k cadenas con entrada x es como en la descrita anteriormente ($k = 1$) con la sola diferencia que la salida se lee de la última cadena cuando la máquina se detiene. La configuración y el número de pasos se define análogamente a como se definió anteriormente. Si para una máquina con k cadenas se llega de una configuración inicial a una configuración de paro en t pasos, entonces el tiempo requerido por M sobre la entrada x es t. Es decir, el tiempo requerido es simplemente el número de pasos para parar. Si $M(x) =$ " $\nearrow$" entonces el tiempo requerido por M sobre x es infinito.

Sea $f : \mathbb{N} \to \mathbb{N}$ y $n = |x|$ la longitud de la cadena X. Se dice que la máquina M opera en tiempo $f(n)$ para cualquier entrada x, el tiempo requerido por M sobre x es $\mathcal{O}(f(n))$. Supongamos que un lenguaje $L \subset (\Sigma \setminus \{\sqcup\})$ es decidido por una máquina de

Turing con múltiples cadenas en tiempo $f(n)$. Se dice que $L \in \mathbf{TIME}(f(n))$. Esto es $\mathbf{TIME}(f(n))$ un conjunto de lenguajes que pueden ser decididos por máquina de Turing con múltiples cadenas que operan en un tiempo acotado por $f(n)$. $\mathbf{TIME}(f(n))$ es una *clase de complejidad* de tiempo, pero también se tienen clases de complejidad de espacio. Cuando la máquina de Turing para, el cursor se detiene en cierta posición de la última cinta; denotemos w como la parte izquierda de la cadena tomando en cuenta el símbolo sobre el cual está el cursor y u la parte de la cadena a la derecha del cursor incluyendo la posición donde se encuentra el cursor. Entonces el *espacio requerido* por M es $\sum_{i=1} |w_i u_i|$. Supongase ahora que $f : \mathbb{N} \to \mathbb{N}$. Entonces decimos que una máquina de Turing M opera en un espacio acotado por $f(n)$. Sea L un lenguaje, decimos que L están en la clase de complejidad de espacio $\mathbf{SPACE}(f(n))$ si existe una máquina de Turing que decida L y opere en un espacio acotado por $f(n)$.

Una máquina no determinista es una tupla N = (Q, Σ, Δ, q_0) donde Q, Σ y q_0 se definen de igual manera que lo hicimos anteriormente. En una máquina no determinista no tiene una sola acción siguiente definida, sino una opción de acciones: una *relación*

$$\Delta \subset (Q \times \Sigma) \times [(Q \cup \{h, \text{“sí”}, \text{“no”}\}) \times \Sigma \times D]. \tag{2.24}$$

Esto es para cada combinación de estado símbolo hay una o más acciones posibles a realizar. Una entrada es *aceptada* si hay más de una secuencia de opciones no determinadas que resulten en “sí”. Otras opciones podrían resultar en rechazo, una computación aceptable es suficiente. La cadena es *rechazada* si y solo si no hay secuencias de opciones que conduzcan a aceptar la cadena.

Se dice que una máquina N decide un lenguaje L en tiempo $\mathcal{O}(f(n))$, si N decide L y hay k cantidad de pasos de una configuración inicial a una configuracion de paro; entonces $k \leq f(|x|)$. Esto es, se requiere que N, además de aceptar a L, no tenga rutas de computación más grandes que $f(n)$. El conjunto de lenguajes decididos por una máquina de Turing no determinista en tiempo $\mathcal{O}(f(n))$ es la clase de complejidad $\mathbf{NTIME}(f(n))$.

2.3.3 CLASES DE COMPLEJIDAD

Con las definiciones anteriores, podemos definir las *clases de complejidad*. Dada una función $f(n)$, se consideran las siguientes clases:

- Clase $\mathbf{TIME}(f(n))$ es el conjunto de lenguajes L tales que una *máquina de Turing determinista* decide en tiempo $\mathcal{O}(f(n))$.

- Clase **NTIME**$(f(n))$ es el conjunto de lenguajes L tales que una *máquina de Turing no determinista* decide en tiempo $\mathcal{O}(f(n))$.
- Clase **SPACE**$(f(n))$ es el conjunto de lenguajes L tales que una *máquina de Turing determinista* decide usando un espacio $\mathcal{O}(f(n))$.
- Clase **NSPACE**$(f(n))$ es el conjunto de lenguajes L tales que una *máquina de Turing no determinista* decide usando un espacio $\mathcal{O}(f(n))$.

Ahora bien en base a la familia de funciones parametrizadas por un entero $k > 0$ que acotan el recurso, en este caso el tiempo, se definen las siguientes clases:

- Clase **P** es el conjunto de lenguajes L tales que una máquina de Turing determinista decide en tiempo polinomial:

$$\mathbf{P} = \bigcup_{k>0} \mathbf{TIME}(n^k). \tag{2.25}$$

- Clase **NP** es el conjunto de lenguajes L tales que una máquina de Turing no determinista decide en tiempo polinomial:

$$\mathbf{NP} = \bigcup_{k>0} \mathbf{NTIME}(n^k). \tag{2.26}$$

- Clase **EXP** es el conjunto de lenguajes L tales que una máquina de Turing determinista decide en tiempo **TIME**(2^{n^k}).
- Clase **NEXP** es el conjunto de lenguajes L tales que una máquina de Turing no determinista decide en tiempo **NTIME**(2^{n^k}).

A continuación se definen las clases en base a la familia de funciones parametrizadas por un entero $k > 0$ sobre el espacio:

- Clase **PSPACE** es el conjunto de lenguajes L tales que una máquina de Turing determinista decide usando un espacio polinomial:

$$\mathbf{PSPACE} = \bigcup_{k>0} \mathbf{SPACE}(n^k). \tag{2.27}$$

- Clase **NPSPACE** es el conjunto de lenguajes L tales que una máquina de Turing no determinista decide usando un espacio polinomial:

$$\textbf{NPSPACE} = \bigcup_{k>0} \textbf{NSPACE}(n^k). \tag{2.28}$$

- Clase **L** es el conjunto de lenguajes L tales que una máquina de Turing determinista decide en tiempo polinomial $\textbf{SPACE}(\log n)$.

- Clase **NL** es el conjunto de lenguajes L tales que una máquina de Turing no determinista decide en tiempo polinomial $\textbf{NSPACE}(\log n)$.

Hay una serie de inclusiones entre las distintas clases de complejidad:

$$\textbf{SPACE}(f(n)) \subseteq \textbf{NSPACE}(f(n)) \tag{2.29}$$

y

$$\textbf{TIME}(f(n)) \subseteq \textbf{NTIME}(f(n)). \tag{2.30}$$

Se han dedicado muchos esfuerzos a estudiar las relaciones entre las diferentes clases de complejidad y todavía quedan problemas abiertos como ¿**P** = **NP** o **P** $\neq$ **NP**? [Fortnow, 2009] Se tiene claro que **P** $\subseteq$ **NP**, pero no se ha conseguido aún demostrar que o **P** = **NP** o que **P** $\neq$ **NP**. Para demostrar la primera, **P** = **NP**, sería necesario demostrar que todo problema en **NP** puede resolverse con una máquina de Turing determinista en tiempo polinomial. Para demostrar la segunda, **P** $\neq$ **NP**, habría que encontrar un problema **NP** que no esté en **P**, es decir, un problema para el que exista una máquina de Turing no determinista de complejidad en tiempo polinomial para el que nunca pueda encontrarse una determinista de la misma complejidad.

Se ha demostrado que **PSPACE** = **NPSPACE**, un resultado que suguiere fuertemente que el no determinismo es menos poderoso con respecto al espacio que con respecto al tiempo [Papadimitriou, 1994]. Con esta información se pueden hacer las siguientes inclusiones:

$$\textbf{L} \subseteq \textbf{P} \subseteq \textbf{NP} \subseteq \textbf{PSPACE} = \textbf{NPSPACE}. \tag{2.31}$$

2.3.4 REDUCCIONES

Las clases de complejidad contienen un conjunto infinito de lenguajes. Por ejemplo la clase **NP** contiene problemas como TSP, SAT, CLIQUE, entre otros. No todos los problemas en la misma clase son igualmente difíciles. Para establecer un orden de problemas de acuerdo a su dificultad, se proponen las reducciones. Un problema $\mathcal{A}$ es por lo menos tan difícil como otro problema $\mathcal{B}$ si existe una *reducción* del problema $\mathcal{B}$ al problema $\mathcal{A}$.

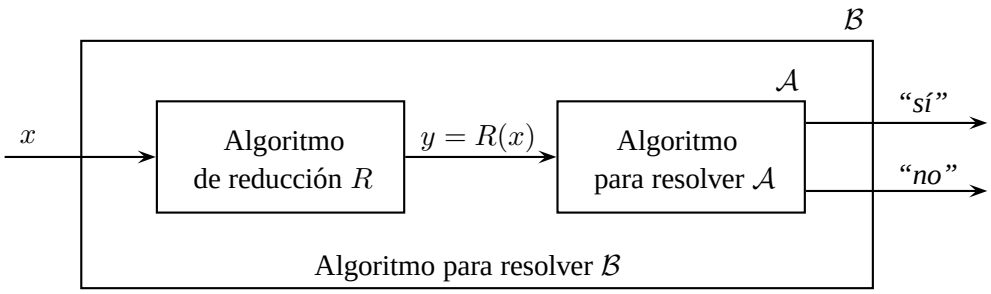


Figura 2.15: *Diagrama de una reducción de un problema $\mathcal{B}$ a un problema $\mathcal{A}$.*

Es así que el problema $\mathcal{B}$ se reduce al problema $\mathcal{A}$ si existe una *transformación eficiente* R tal que, por cada entrada x de $\mathcal{B}$, produce una entrada *equivalente* y de $\mathcal{A}$, es decir $y = R(x)$, esto se ilustra en la Figura 2.15. El término *equivalente* quiere decir que la respuesta a $R(x)$ considerada como entrada para $\mathcal{B}$ es una respuesta correcta a y considerada como entrada para $\mathcal{A}$. En otras palabras, para resolver $\mathcal{B}$ por la entrada x solo hay que construir $R(x)$ y resolver $\mathcal{A}$. Si contamos entonces con un algoritmo para el problema $\mathcal{A}$ y un algoritmo para construir $R(x)$ la combinacion de ambos da un algoritmo para resolver $\mathcal{B}$.

Aquí es importante que la transformación R sea *fácil* de realizar, es decir, las reducciones también se clasifican según los recursos que se usen, por ejemplo las reducciones de Cook permite que $R(x)$ sean computadas por una ***máquina de Turing*** en ***tiempo polinomial*** y en ***espacio logarítmico***. Poniéndolo en términos de lenguajes, se tiene:

Un lenguaje L es *reducible* a un lenguaje L', denotado por $L \leq L'$, si y solo si existe una función R de cadenas a cadenas que está computada por una máquina Turing determinista en espacio $\mathcal{O}(\log n)$ tal que $\forall x, x \in L$ si y solo si $R(x) \in L'$. La función R se llama una *reducción* de L a L'.

2.3.5 COMPLETEZ

La *transitividad* de la reducción polinomial nos dice que es una relación de equivalencia, y la relación $\leq$ impone un orden parcial en las clases de equivalencia de problemas resultantes, es decir, nos ordena los problemas segun su complejidad. De hecho, la clase **P** forma la "menor" clase de equivalencia con este orden parcial y, por tanto, puede verse como la clase que contiene los problemas de decisión computacionalmente "más fáciles".

Sea **C** una clase de complejidad, y sea L un lenguaje que pertenece a **C**. Diremos que L es **C**-*completo* si todo lenguaje $L' \in$ **C** se puede reducir a L, $L' \leq L$. Un lenguaje es **C**-*duro* si se puede reducir cada lenguaje $L' \in$ **C** a L, pero no se sabe si es válido que $L \in$ **C**. Si el lenguaje que corresponde a un problema es completo, entonces, el problema computacional es completo.

El primer problema de decisión que se demostró ser completo en la clase **NP** fue el problema SAT [Cook, 1971]. Esta clase de problemas **NP**-*completos* son de interés práctico; un listado clásico de algunos problemas de este tipo es realizado por Garey y Johnson [1979].

2.4 COMPLEJIDAD PARAMETRIZADA

La *teoría de complejidad parametrizada* [Flum y Grohe, 2006] es una rama de la teoría de complejidad computacional la cual se enfoca en clasificar problemas de acuerdo a su dificultad con respecto a ciertos parámetros de los problemas, de manera que la complejidad de un problema es medida en función de un parámetro además del tamaño de la instancia del problema. La idea central de esta teoría es la noción de la tratabilidad de parámetro fijo, la cual relaja la noción de tratabilidad — tiempo de cómputo polinomial — admitiendo algoritmos para los cuales el tiempo de cómputo es exponencial en términos del tamaño de la instancia pero en términos de algún parámetro del problema se tienen tiempos de cómputo polinomiales. Es importante decir que esta teoría trata con problemas parametrizados en donde se debe indicar para qué parámetro específico del problema se analizará la complejidad.

Definición 2.2 *Sea Σ un alfabeto finito.*

*(1) Una **parametrización** de Σ^* es un mapeo $\kappa : \Sigma^* \to \mathbb{N}$ que es computable en tiempo polinomial.*

*(2) Un **problema parametrizado** es un par (Q, κ) que consiste en un conjunto $Q \subseteq \Sigma^*$ y una parametrización κ de Σ^*.*

Es importante recordar que la motivación para la noción de tratabilidad de parámetro fijo *ftp* (en inglés: *fixed-parameter tractability*) es que para ciertos valores de un parámetro del problema, el problema es tratable en términos de tipos de cómputo.

Definición 2.3 *Sea Σ un alfabeto finito y $\kappa : \Sigma^* \to \mathbb{N}$ una parametrización:*

*(1) Un algoritmo $\mathcal{A}$ con una entrada Σ es un **algoritmo-ftp** con respecto a κ si existe una función computable $f : \mathbb{N} \to \mathbb{N}$ y un polinomio p tal que para cada $x \in \Sigma^*$, el tiempo de cómputo de un algoritmo $\mathcal{A}$ sobre la entrada x sea a lo más*

$$f(\kappa(x)) \cdot p(|x|). \tag{2.32}$$

(2) Un problema parametrizado (Q, κ) es tratable de parámetro fijo si existe un algoritmo-ftp con respecto a κ que decide Q.

FTP denota a la clase de todos los problemas que son tratables de parámetro fijo, de manera que si $Q \in$ **PTIME** entonces $(Q, \kappa) \in$ **FTP** para cada parametrización κ.

Si bien la clase **FTP** puede ser vista como la clase análoga de **P** en la teoría de la complejidad parametrizada, la clase parametrizada análoga a **NP** es la clase *para*-**NP** en donde se reemplaza el término "algoritmo" de la definición de **FTP** por el de "algoritmo no determinista".

Definición 2.4 *Un problema parametrizado (Q, κ) está en para-**NP**, si existe una función computable $f : \mathbb{N} \to \mathbb{N}$, un polinomio p y un algoritmo no determinista que dado $x \in \Sigma^*$, decide si $x \in Q$ en a lo más $f(\kappa(x)) \cdot p(|x|)$ tiempo de cómputo.*

Un ejemplo de problema parametrizado es el problema p-SAT donde el parámetro p es el número de variables en cada cláusula de la expresión booleana. Se sabe que cuando $p \geq 3$ el problema es intratable, pero para $p = 2$ el problema se vuelve tratable. Para más ejemplos de parametrizaciones de problemas y adentrarse en este tópico se recomienda el libro de Flum y Grohe [2006].

2.5 PROBLEMAS DE OPTIMIZACIÓN

Como se mencionó en la Sección 2.3, además de los problemas de decisión, la complejidad computacional también trata con *problemas de optimización*; mientras que en los primeros la respuesta es siempre *"sí"* o *"no"*, en los segundos se requiere además encontrar la mejor de todas las soluciones posibles de acuerdo a una criterio de evaluación llamado *función objetivo*. Un ejemplo de este tipo de problema es el de *máxima satisfactibilidad* (MAX-SAT) en el cual el problema es determinar el máximo número de cláusulas en una fórmula booleana que pueden ser satisfechas por una asignación de verdad. En los problemas de optimización, la instancia está compuesta por un conjunto de variables de decisión, un conjunto de restricciones sobre los valores de las variables de decisión y una función objetivo que de acuerdo a un criterio asigna un valor real a cada instancia. Si las variables de decisión son discretas el problema se dice que es de *optimización combinatoria*.

Definición 2.5 *Un problema de optimización $\mathcal{P} = (\mathcal{S}, f)$ se define por:*

1. *un conjunto de variables $X = x_1, \ldots, x_n$,*
2. *el dominio de las variables $D_1, \ldots, D_n$,*
3. *un conjunto de restricciones sobre las variables,*
4. *un función objetivo a ser minimizada*[4]*, donde $f : D_1 \times \cdots \times D_n \to \mathbb{R}$.*

El conjunto de todas las posibles asignaciones factibles es:

$$\mathcal{S} = \Big\{ s = \{(x_1, v_1), \ldots, (x_n, v_n)\} \mid v_i \in D_i, s \text{ satisface todas las restricciones} \Big\},$$

el cual es conocido como *espacio de soluciones*.

Cuando las variables de decisión cumplen con todas las restricciones se dice que es una *configuración factible*. En este tipo de problemas el objetivo es identificar cual de las configuraciones factibles tiene el mejor valor de la función objetivo. Si el problema de optimización es de *maximización*, el mejor valor es el mayor de los valores asociados a las configuraciones factibles que se obtienen, si es de *minimización* el mejor valor es el

[4]Maximizar una función f es equivalente a minimizar $-f$; tomando esto en cuenta, se trabajará con problemas de minimización sin perder generalidad en el tema.

menor de los valores. Una configuración factible que corresponda con el mejor valor es llamada *solución óptima* $\bar{s}$ de la instancia. Se utiliza el término de *optimización global* para referirse al proceso de encontrar una solución $\bar{s}$ del conjunto $\mathcal{S}$.

Definición 2.6 *Una solución* $\bar{s} \in \mathcal{S}$ *es un* ***óptimo global*** *de* $\mathcal{P}$ *si y solo si* $f(s) \geq f(\bar{s})\ \forall s \in \mathcal{S}$.

Se puede transformar cualquier problema de optimización en un problema de decisión incluyendo una cota para el valor objetivo como dato adicional y preguntar si existe una solución que su valor objetivo que no exceda la cota definida. El punto importante a resaltar es que si se encuentra una solución óptima fácilmente, el problema de decisión no puede ser más difícil que el correspondiente problema de optimización, es decir, si se puede encontrar una respuesta al problema de optimización en tiempo polinomial, también se puede resolver el problema de decisión en tiempo polinomial. De esta manera, si se demuestra que un problema de decisión es **NP**-*completo* se puede decir que el correspondiente problema de optimización es **NP**-*duro*, es decir, por lo menos tan díficil como el problema de decisión.

2.6 METAHEURÍSTICOS

Debido a la importancia práctica de los problemas de optimización, muchos algoritmos han sido desarrollados para resolver estos problemas. Estos algoritmos pueden dividirse en *exactos* o *aproximados*. Los algoritmos exactos garantizan encontrar para cada tamaño de instancia de un problema de optimización una solución óptima en un tiempo acotado. Sin embargo para problemas **NP**-*duros* no se conoce un algoritmo exacto en tiempo polinomial que lo resuelva, en el peor de los casos necesitaría un tiempo de cómputo exponencial; por lo general es necesario enumerar todas las posibles soluciones asociadas al problema. Esto resulta impráctico por el trabajo que requiere, por lo cual en muchas ocasiones es preferible usar algoritmos aproximados en donde se sacrifica la garantía de encontrar una solución óptima por obtener muy buenas soluciones en un tiempo de cómputo aceptable [Blum y Roli, 2003].

Dentro de los algoritmos aproximados encontramos algoritmos de búsqueda conocidos como *heurísticos* los cuales son un conjunto de reglas que permiten encontrar soluciones de buena calidad. Básicamente dentro de los algoritmos heurísticos se pueden distinguir entre algoritmos *constructivos* y algoritmos de *búsqueda local*. Los algoritmos

constructivos generan soluciones añadiendo componentes hasta que la solución está completa. Los algoritmos de búsqueda local empiezan de una solución inicial e iterativamente tratan de reemplazar la solución actual por una solución mejor, en un vecindario definido apartir de la solución actual. Por esta razón al espacio de soluciones también se le conoce como *espacio de búsqueda*. Típicamente se combinan estas dos estrategías en un proceso iterativo donde en cada repetición se construye una solución inicial y después empieza fase de mejora a través de una búsqueda local hasta llegar a una solución mejor que las demás o llegar a la solución óptima [Blum y Roli, 2003].

Definición 2.7 *Un **vecindario** $\mathcal{N}(s)$ de s es una función $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ que asigna a cada $s \in \mathcal{S}$ un conjunto de vecinos $\mathcal{N}(s) \subseteq \mathcal{S}$* [5].

Definición 2.8 *Una solución $\bar{s} \in \mathcal{S}$ es un **óptimo local** de $\mathcal{P}$ si y solo si $\exists\, \varepsilon > 0$ tal que $\forall s \in \mathcal{S}, f(s) \geq f(\bar{s})$, con $\|s - \bar{s}\| < \varepsilon$.*

Una desventaja de los problemas de optimización es que muchos de ellos tienen múltiples óptimos locales. Por ello han emergido *metaheurísticas* que combinan varios métodos heurísticos con el objetivo de explorar el espacio de búsqueda efectiva y eficientemente para no quedar atrapado en óptimos locales en las cuales se pueden emplear técnicas de intensificación y diversificación. El término *diversificación* se refiere a la exploración del espacio de búsqueda para identificar regiones prometedoras y el término *intensificación* se refiere a la explotación de la experiencia acumulada durante el proceso de búsqueda. Una descripción amplia de diversos metaheurísticos es realizada por Blum y Roli [2003]. A continuación se describirán brevemente algunos:

GRASP Procedimiento de búsqueda voraz adaptativo aleatorizado (en inglés: *Greedy Randomized Adaptive Search Procedure*), introducido por Feo y Resende [1995], es un procedimiento en donde cada paso consiste primeramente de una fase de *construcción* para obtener una buena solución inicial y finalmente una fase de *mejora* a través de una búsqueda local. Resende y Veiga [2003] hicieron una amplia compilación de trabajos relacionados con este procedimiento.

En la fase constructiva se considera que una solución s es un subconjunto de elementos y se va construyendo añadiendo paso a paso un nuevo elemento. El elemento es elegido de una lista restringida de candidatos compuesta por lo mejores elementos. Para seleccionar los mejores los elementos se ranquean a través de una función

[5] El conjunto $2^{\mathcal{S}}$ es el conjunto de todos los posibles subconjuntos de $\mathcal{S}$.

voraz que se basa en el beneficio que aporta el elemento a la solución parcial actual. En la segunda fase como se mencionó anteriormente consiste en tomar la solución construida como solución inicial de un algoritmo de búsqueda local, por lo cual el algoritmo GRASP no usa datos históricos del proceso de búsqueda.

Recocido simulado Los orígenes de este algoritmo (en inglés: *simulated-annealing*) están en la mecánica estadística, con el algoritmo de Metropolis *et al.* [1953]. Fué propuesto por Kirkpatrick *et al.* [1983] e independientemente por Černý [1985]. La idea fundamental de recocido simulado es que permite moverse con cierta probabilidad a soluciones peores que la solución actual, con el objetivo de escapar de óptimos locales. El algoritmo comienza generando una solución inicial e inicializando un parámetro llamado *temperatura* T. En cada iteración una solución $s' \in \mathcal{N}(s)$ es generada y aceptada como nueva solución actual en dependencia de $f(s)$, $f(s')$ y T. La solución s' remplaza a s si $f(s') < f(s)$, en el caso $f(s') \geq f(s)$ el reemplazo se realiza en función de T y $f(s') - f(s)$. Esta probabilidad se calcula con la *distribución de Boltzmann* $\exp(-(f(s') - f(s))/T)$. El parámetro de la temperatura se va decrementando durante el proceso de búsqueda, por lo cual al inicio la probabilidad de aceptar soluciones peores a la solución actual es alta y va decreciendo hasta que se converge a un algoritmo iterativo simple de mejora.

Búsqueda tabú Este procedimiento (en inglés: *tabu search*) propuesto por Glover [1989] aplica una mejor búsqueda local y utiliza una memoria a corto plazo para escapar de los óptimos locales y evitar ciclos. La memoria es implementada como una *lista tabú* que mantiene un seguimiento de las soluciones visitadas recientemente y prohibe movimientos en la búsqueda que lleven a los elementos de la lista tabú. De esta manera el vecindario de la solución actual se restringe a soluciones que no pertenecen a la lista tabú, de manera que en cada iteración se elige la mejor solución del vecindario como solución actual. Esta solución se añade a la lista tabú al mismo tiempo que se elimina una de las soluciones en la lista tabú.

La búsqueda tabú es considerada como una técnica de búsqueda dinámica en el vecindario. La longitud de esta lista tabú es conocida como *tenencia tabú* y controla la memoria del proceso de búsqueda. Con tenencias pequeñas la búsqueda se concentra en áreas pequeñas del espacio de búsqueda; por el contrario tenencias grandes forza al proceso de búsqueda a explorar regiones más grandes. La tenencia tabú puede variar durante el proceso de búsqueda lo que da lugar a un procedimiento más robusto. Por ende se propone guardar información acerca de ciertas propiedades de las soluciones, que nos permitan determinar si una solución cumple con cier-

tas *condiciones tabú*; estas condiciones permite generar el conjunto de soluciones permitidas. Esto puede ser más eficiente pero tener condiciones tabú significa asignar probablemente un estado tabú a más de una solución. Para evitar este problema se define un *criterio de aspiración* el cual permite incluir soluciones aunque sea prohibida por las *condiciones tabú*.

El uso de memoria a largo plazo añade a este procedimiento cuatro dimensiones: recencia, frecuencia, calidad e influencia. La *memoria básada en recencia* registra en cual de las iteraciones más recientes se encontró esa solución; *la memoria básada en frecuencia* registra cuantas veces se ha visitado cada solución, lo que ayuda a diversificar la búsqueda. La *calidad* se refiere a la acumulación y extracción de información de la historia de búsqueda para identificar buenas soluciones. Por último la *influencia* tiene que ver en cómo la información guardada ha impactado en el proceso de búsqueda.

Computación evolutiva El término computación evolutiva refiere a los algoritmos inspirados en la capacidad natural que tienen los seres vivos para evulucionar y adaptarse a su ambiente. En cada iteración el algoritmo trabaja con un conjunto de soluciones llamada *población* sobre la cual se aplican *operadores* a cada solución o individuo, para generar individuos de la siguiente iteración llamada *generación*. Generalmente se utilizan dos operadores: el *cruzamiento* para combinar dos o más individuos para producir nuevos y la *mutación* que causa una autoadaptación del individuo. Los individuos producidos en una generación pasan por un proceso de selección en base a su *aptitud*, por ejemplo el valor de la función objetivo, en el cual individuos con mayor aptitud tienen mayor probabilidad de ser elegidos como parte de la siguiente generación. Hay una gran variedad de algoritmos propuestos como la programación evolutiva [Fogel y Fogel, 1995], estrategías evolutivas [Beyer y Schwefel, 2002] y los algoritmos géneticos [Goldberg, 1989; Holland, 1992].

2.6.1 DESEMPEÑO DE METAHEURÍSTICOS

Como se mencionó en la sección anterior, los algoritmos aproximados — heurísticos y metaheuríticos — sacrifican la calidad de la solución obtenida debido a que en muchas ocasiones resulta impráctico buscar la mejor de las soluciones posibles. Sin embargo se puede hablar de un algoritmo *superior* que obtenga soluciones de muy alta calidad, que obtenga soluciones rápidamente y que sea robusto, es decir, que tenga buen desempeño en un amplio rango de problemas y valores de parámetros propios del algoritmo. Para

determinar que algoritmo es superior a otro se debe demostrar superioridad en uno o más de estos aspectos: calidad, tiempo y robustez. Existen dos maneras para demostrar esta superioridad ya sea a través de un análisis teórico — véase Sección 2.3.1 — o de pruebas experimentales.

Las pruebas experimentales consisten en resolver un problema computacional usando una implementación del algoritmo que se desea demostrar su eficiencia o efectividadd en ciertas instancias estándar (en inglés: *benchmark*) del problema que son usadas para probar algoritmos. Para ello se ejecuta varias veces el algoritmo con esa instancia en particular y se comparan los resultados obtenidos con los de otros algoritmos existentes.

Es importante señalar que existen diferentes factores que pueden afectar significativamente la información obtenida con el experimiento, entre ellos: el problema seleccionado, las instancias del problema usadas, la implementación del algoritmo, parámetros de algoritmo, las características del equipo en donde se llevan a cabo las experimentaciones, las medidas de desempeño consideradas [Greenberg, 1990]. Un aspecto importante es asegurarse que la información reportada sea significativa. Barr *et al.* [1995], Johnson [1996] y McGeoch [2001] presentan guías para diseñar experimentos y reportar los resultados de la contribución del algoritmo propuesto, los valores de los parámetros para los cuales el algoritmo se desempeño bien y en general el proceso llevado a cabo para determinarlos; esto con el objetivo de identificar ideas innovadoras que pueden ser usadas en otros problemas y evaluar la robustez del algoritmo.

La medición del desempeño del algoritmo es un aspecto crítico, es importante tener claro qué medir, cómo medir y cómo asegurarnos que la medición no interfiere con el experimento llevado a cabo [Fleming y Wallace, 1986; McGeoch, 1996; Moret, 2002]. En general se deben tener medidas que puedan resolver las siguientes preguntas:

- ¿Cuál es la mejor solución encontrada por el algoritmo?
- ¿Cuánto tiempo lleva encontrar la mejor solución?
- ¿Qué tan rápido es el algoritmo para encontrar buenas soluciones?
- ¿Qué tan robusto es el algoritmo?

Las medidas de desempeño usadas en pruebas experimentales tienen que ver con el valor de la solución, el espacio de memoria usado, la velocidad y tasa de convergencia hacia la solución óptima, o qué tan cercana es la solución encontrada con la solución óptima

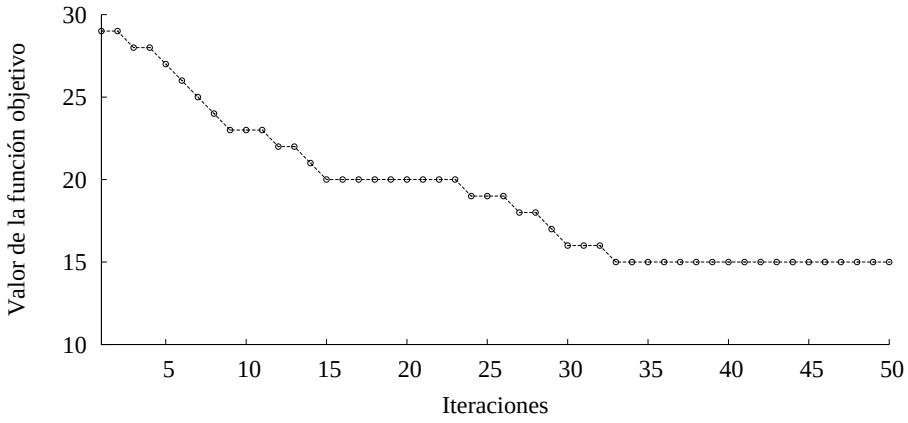


Figura 2.16: *En esta figura se muestra el comportamiento de un algoritmo donde se minimiza el valor de la función objetivo. Se puede ver como el algoritmo no encontró soluciones mejores que el mejor conocido de la iteración* 15 *a la* 24*, después el algoritmo empezo a obtener mejores resultados y apartir de la iteración* 33 *volvió a tener un comportamiento similar. Se consideró detener el algoritmo al llegar a la iteración* 50*. La pregunta aquí es: de haber permitido realizar más iteraciones, ¿el algoritmo habría obtenido mejores soluciones?*

o con la solución mejor conocida hasta el momento, cantidad de iteraciones, número de llamadas a una rutina especial del algoritmo, cantidad de comparaciones y asignaciones.

Medir el tiempo de cómputo puede llegar a ser problemático por la variedad de equipos usados por diferentes investigadores para realizar las pruebas y las estretegias de programación que se consideraron para implementar al algoritmo. Por ello es importante que siempre que se pueda analizar la complejidad del algoritmo aproximado en términos de la notación $\mathcal{O}$ [Klein y Young, 2010]. Además generalmente en estos algoritmos es común establecer un criterio de parada como el tiempo de ejecución, número de iteraciones máximas, parar después de x iteraciones sin cambios en la solución obtenida, lo cual afecta la calidad de la solución. Por ejemplo, en la Figura 2.16 si se toma la decisión de parar tomando en cuenta ciertas iteraciones sin cambio en la solución, podríamos cortar la oportunidad de obtener mejores soluciones.

2.6.2 ANÁLISIS DEL PAISAJE DE APTITUDES

Ante la necesidad de algoritmos aproximados que sean en la práctica efectivos y eficientes, se ha propuesto estudiar el espacio de búsqueda del problema, ya que el desempeño de los algoritmos está relacionado con la dificultad o facilidad de desplazamiento por el espacio de búsqueda. Una manera de estudiar el espacio de búsqueda, es a través del

concepto *paisaje de aptitudes* — originado en la biología evolutiva por Wright [1932] y adoptado en la computación evolutiva [Fogel y Fogel, 1995] — donde la función objetivo generalmente es considerada como una *función de aptitud*, el vecindario $\mathcal{N}(s)$ es llamado *configuraciones* y la elección de la siguiente solución es llamada *movimiento*.

La dificultad asociada al espacio de búsqueda está relacionado con la *rugosidad* del paisaje de aptitudes, es decir, que tenga una gran cantidad de óptimos locales vistos como obstáculos para encontrar el óptimo global; por lo cual encontrar buenas soluciones depende de de la función de aptitud y la geometría del espacio de búsqueda la cual es inducida por el proceso de búsqueda [Jones, 1995]. También la *neutralidad* del espacio de búsqueda — configuraciones con la misma aptitud — impacta el desempeño de los algoritmos. De aquí que la geometría del espacio de búsqueda sea referida como paisaje de aptitudes con montañas, picos, valles y mesetas, como se ilustra en la Figura 2.17; aunque un poco diferente desde nuestra perspectiva tridimensional. Una revisión del tema y las propiedades geométricas de los paisajes es hecha por Stadler [2002].

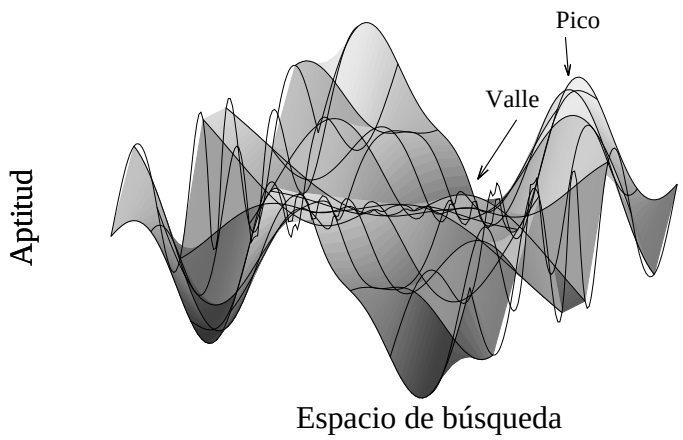


Figura 2.17: *Representación tridimensional de una superficie de aptitudes.*

Por lo regular, el estudio del paisaje de búsqueda se realiza sobre las trayectorias de búsqueda que realizan los algoritmos, referidas como caminatas aleatorias[6] [Weinberger, 1990]. Algunas medidas que indican la rugosidad o neutralidad de la superficie de aptitudes son:

Auto-correlación El objetivo de esta medida es cuantificar a la rugosidad de la superficie a través de la correlación existente entre configuraciones adyacentes. La idea pro-

[6]Una caminata aleatoria es una trayectoria que resulta de hacer pasos sucesivos aleatorios entre vecinos.

puesta por Weinberger [1990] es generar una caminata aleatoria en la superficie y en cada paso guardar la aptitud encontrada. De esta manera se genera una serie de tiempo de aptitudes $F = \{f_1, f_2, \ldots, f_n\}$. Para calcular esta medida se debe especificar una distancia k entre las aptitudes. La autocorrelación es estimada como:

$$\rho_k = \frac{\sum_{i=1}^{n-k}(f_i - \bar{f})(f_{i+k} - \bar{f})}{\sum_{i=1}^{n}(f_i - \bar{f})^2},\ n > k, \tag{2.33}$$

donde:

$$\bar{f} = \frac{1}{n}\sum_{i=1}^{n} f,\ n > 0. \tag{2.34}$$

Si $|\rho_k| \approx 1$, las diferencias entre pares de aptitudes vecinas son similares, es decir, la superficie es como un valle. Si $|\rho_k| \approx 0$, los valores aptitud son casi independientes; por tanto la superficie es muy rugosa [Weinberger, 1990].

Longitud de auto-correlación Esta medida también propuesta por Weinberger [1990] indica cuál es el valor mayor de distancia a la que el conjunto de soluciones se vuelve no correlacionado. Se calcula mediante:

$$\|\rho_k\| = \frac{1}{1 - \rho_k}. \tag{2.35}$$

Un valor alto de $\|\rho_k\|$ indica una superficie muy plana, mientras que un valor pequeño indica una superficie más rugosa.

Coeficiente de engaño Esta medida propuesta por Jelasity *et al.* [1999a] se aplica sobre las aptitudes finales de m ejecuciones del algoritmo, obteniéndose el conjunto $F = \{f_1, f_2, \ldots, f_m\}$ sobre el cual se calcula la aptitud final esperada E, el valor máximo de las aptitudes finales $F_{\text{máx}}$ y el valor mínimo de las aptitudes finales $F_{\text{mín}}$. El coeficiente de engaño se obtiene como:

$$s_K = 1 - \frac{E - F_{\text{mín}}}{F_{\text{máx}} - F_{\text{mín}}}. \tag{2.36}$$

Se define que sí $F_{\text{máx}} - F_{\text{mín}} = 0$, entonces $s_K = 0$. Valores cercanos a uno indican que la estructura del paisaje de aptitudes es engañoso lo que hace que el algoritmo de búsqueda obtenga aptitudes finales muy diferentes y encontrar una buena solución

siempre que se ejecute el algoritmo es díficil. Por otro lado valores cercanos a cero indican que la superficie de aptitudes es amigable porque lleva siempre a soluciones similares.

Si bien estas medidas provienen del análisis estadístico, se han propuesta medidas basadas en el análisis de información que se obtiene a través de las caminatas de los algoritmos. Para esto se codifica la serie de tiempo de aptitudes en una cadena $S = s_1 s_2 \dots s_n$ de símbolos donde $s_i \in \{\bar{1}, 0, 1\}$. Esta codificación se realiza de dos maneras dependiendo de la medida de información que se desea obtener de la superficie. Detalles de cómo se realiza la codificación de las cadenas y cómo se hacen los cálculos para obtener medidas basadas en información se encuentran en el trabajo realizado por Vassilev *et al.* [2000], en el trabajo realizado por Rosas [2007] se muestra un ejemplo de cómo realizar el cálculo de estas medidas estadísticas y de información para un problema de empaquetado de objetos. A continuación se describen algunas medidas basadas en el análisis de información:

Contenido de información Es una medida basada en la entropía de la superficie; es usada para caracterizar la rugosidad con la cual se topa el algoritmo en su trayectoria por el espacio de búsqueda.

Contenido de información parcial Esta medida está relacionada con la cantidad de óptimos locales que aparecen en la trayectoria del algoritmo.

Información de densidad de valles Es otra medida basada en la entropía de la superficie; es usada para caracterizar las partes planas visitadas por el algoritmo en su trayectoria por el espacio de búsqueda.

Existen otras medidas basadas en las distancias entre dos soluciones s_i y s_j en el espacio de búsqueda que tienen que ver con el número de movimientos que el algoritmo hace para llegar de la solución s_i a la solución s_j. El cálculo de esta distancia entre soluciones depende del problema que se está estudiando y cómo se representa la solución [Bouziri *et al.*, 2011].

Básicamente el estudio del paisaje de aptitudes se ha orientado a mejorar el desempeño de los algoritmos, ya que la información obtenida se ha utilizado para diseñar estrategías de búsqueda que permitan a los algoritmos navegar en el espacio de búsqueda de una manera inteligente.

CAPÍTULO 3

TRABAJOS RELACIONADOS

Todos los objetos de la razón e investigación humana
pueden, naturalmente, dividirse en dos grupos, a saber:
relaciones de ideas y cuestiones de hecho.
- David Hume.

En este capítulo se describen y analizan trabajos relacionados con el tema de estudio, trabajos que inciden en uno o más aspectos de la investigación. Principalmente los trabajos aquí descritos se han enfocado en determinar qué propiedades del espacio de búsqueda afectan el desempeño de los algoritmos usados e identificar qué instancias son más difíciles o más fáciles de resolver. En este trabajo se quiere determinar *qué propiedades estructurales de las instancias efectan la complejidad de solución del problema*.

3.1 DESCRIPCIÓN

Existe una gran variedad de trabajos que proponen algoritmos para obtener mejores soluciones que las reportadas en el estado del arte haciendo uso eficiente de los recursos computacionales [Khajehzadeh *et al.*, 2011; Zanakis *et al.*, 1989]; trabajos que estudian el espacio de búsqueda ya sea para explicar el desempeño de los algoritmos o para diseñar estrategias de búsqueda que aprovechen la estructura del espacio de soluciones [Baluja y Davies, 1997; Jelasity *et al.*, 1999b; Popovici y De Jong, 2005; Rosas, 2007; Yossi y Poli, 2005]. La mayoría de estos trabajos se limitan a mostrar la superioridad de los algoritmos propuestos y/o cómo fueron incorporadas ciertas estrategias en la búsqueda de soluciones.

Cuando se estudia espacio de soluciones se debe tomar en cuenta que la percepción de la estructura — en términos de rugosidad o planaridad — depende de la formulación de la función de aptitud y las configuraciones de los movimientos; cuando se reportan

resultados de los algoritmos hay que tomar en cuenta las características del equipo usado, técnicas de programación y las medidas de desempeño usadas. En ambos casos se presta mayor atención a los algoritmos y no a las instancias sobre las cuales se ejecutan los algoritmos, señalando solalmente cuales instancias son díficiles o fáciles de resolver en términos del resultado obtenido en un determinado tiempo, sin mencionar porque son difíciles o fáciles, o que características poseen que las hacen difíciles o fáciles.

La *teoría de la complejidad parametrizada* ha provisto un marco de trabajo para estudiar la complejidad computacional de un problema dependiendo de parámetros propios del problema [Flum y Grohe, 2004; Marx y Schlotter, 2010, 2011], desde el punto de vista de la *física estadística* se ha estudiado valores de parámetros en problemas **NP**-completos en los cuales se presenta una transición de fase, es decir, cambios en la dificultad de solución del problema [Achlioptas *et al.*, 2005; Fu y Anderson, 1986; Mammen y Hogg, 1997; Martin *et al.*, 2001; Monasson *et al.*, 1999], como los ejemplificados en la Sección 1.1. En ambos casos, se ha concluido que para ciertos parámetros de problemas **NP**-completos estos son tratables, es decir, son fáciles de resolver, sin embargo esto es tomando en cuenta parámetros propios del problema, no de la instancia del problema.

En los últimos años, diversos trabajos han mostrado que la estructura de las redes del mundo real influyen en el comportamiento de varios procesos que toman lugar en ellas. En este trabajo es de principal interés estudiar qué propiedades estructurales en las instancias de grafos facilitán o dificultán la solución de un problema. A continuación se describen trabajos en los cuales se ha señalado que la estructura de la instancia puede ser un factor que influye en la dificultad de solución de un problema, planetando como problema abierto el entendimiento de qué propiedades influyen en la solución de un problema y cómo lo hacen.

Coudert [1997] abordó el problema de coloreo de grafos. Estudia heurísticos y una técnica exacta (coloreo secuencial) sobre grafos “artificiales” y grafos del “mundo real”. Concluyó que los grafos del mundo real son instancias fáciles debido a su número cromático puede ser igual al tamaño del clique más grande. Para determinar la facilidad de solución, toma en cuenta el número de heurísticos que llegaron al número cromático o qué tan alejados quedaron de dicho número. Este trabajo recibió una fuerte crítica por parte de Kirovski y Potkonjak [1998], en particular en dos aspectos: el primero es que Coudert trata de resolver un problema difícil usando otro problema difícil — encontrar el clique de tamaño máximo — y el segundo es que no provee resultados experimentales en una diversidad grande de grafos. Kirovski y Potkonjak encontraron ejemplos de la vida real difíciles de colorear que resultaron ser grafos muy densos, por lo cual descartaron la con-

clusión dada por Coudert, recalcando que la ***evaluación del rendimiento y variabilidad de un algoritmo de coloreo de grafos es fuertemente dependiente de la estructura de los grafos y que esta dependencia debe ser estudiada con mayor rigor***.

Mulet *et al.* [2002] estudiaron el problema de coloreo sobre grafos aleatorios con cierta conectividad media para un número k de colores. Encontraron que grafos con baja conectividad aceptan coloreos legales, mientras que grafos con alta conectividad son no coloreables. Se enfocaron en determinar la conectividad crítica tras la cual la probabilidad de que sean no coloreables tiende a uno cuando el tamaño del grafo crece infinitamente. A esta transición se le llama q-COL/UNCOL, donde el valor de la conectividad crítica depende del número de colores permitidos; los grafos generados cerca de la conectividad crítica son difíciles de colorear.

Hamiez y Hao [2004] señalaron que las propiedades de las soluciones pueden ayudar a explicar el comportamiento de algunos algoritmos sobre un grafo particular. Estudiaron una propiedad llamada *conjunto respresentativo* concluyendo que se puede suponer cierta relación entre la existencia, número y tamaño de estos conjuntos y la estructura de los grafos, pero que se necesita mayor evidencia para confirmar o refutar esta hipótesis. Recalcaron la ***importancia de investigar las relaciones entre la estructura de los grafos y las propiedades de las soluciones y también las relaciones entre las propiedades de las soluciones y la dificultad para resolver el problema en una instancia en particular***.

Los grafos del mundo real exhiben una distribución del grado de *ley de potencias*. Ante la evidencia práctica que la optimización combinatoria en estos grafos es más fácil que los grafos generales; Ferrante *et al.* [2008] enfocaron su estudio en probar que muchos problemas **NP**-duros de optimización en grafos permanecen **NP**-duros en grafos de libre escala. Sus resultados teóricos muestran que — en el peor caso — los grafos de libre escala son duros con respecto a muchos problemas importantes de optimización en grafos. Sin embargo, evidencia experimental mostró que la optimización es considerablemente más fácil que en instancias de grafos del mundo real, por lo cual concluyeron que estos grafos no caracterizan a instancias del peor caso.

Smith-Miles *et al.* [2009] señalaron que el ***conocimiento de la estructura del problema y las características de las instancias pueden asistir a la selección de algoritmos para resolver el problema***. Recalcaron que a pesar de que el desempeño algorítmico se ha estudiado mediante el espacio de búsqueda o las características de las instancias del problema, *se tiene un deficiente entendimiento de cómo exactamente el rendimiento de los algoritmos depende de las características de las instancias*.

En este estudio Smith-Miles *et al.* enfatizan que un aspecto importante en ***cualquier estrategia orientada a establecer relaciones entre el desempeño algorítmico y estructura de la instancia es la selección de caracteristicas con la cual se realizará la caracterización de instancias***. Posteriormente Smith-Miles *et al.* [2010] señalaron que las características que se consideren correlacionar con la dificultad de solución, pueden incluir métricas basadas en identificar los retos que impone el espacio de búsqueda.

Porumbel *et al.* [2010] presentaron un análisis del espacio de búsqueda con el objetivo de mejorar algoritmos de búsqueda local para el problema de coloreo de grafos; para realizar este análisis hicieron uso de la distancia entre coloreos. Su hipótesis radica en que las soluciones de alta calidad no están distribuidas al azar en el espacio de búsqueda, sino que están agrupadas en esferas de determinado tamaño. Basado en esto construyen algoritmos que explotan el agrupamiento de las soluciones. Encontraron diferencias en los espacios de búsqueda con respecto a los coloreos legales y número de conflictos, en varias instancias. Llegaron a la conclusión que estas ***diferencias en los espacios de búsqueda recaen en la estructura de los grafos a ser coloreados, sin llegar a establecer una relación entre las características del espacio de búsqueda y la estructura de los grafos.***

Cocco *et al.* [2006] hacen una revisión de métodos de física estadística aplicada al análisis de algorítmos para problemas de optimización con distribuciones de entradas aleatorias, señalaron que si bien los físicos estadísticos y los teóricos de las ciencias computacionales buscan entender el comportamiento asintótico de los procesos actuando en espacio de configuraciones exponencialmente largos, existen diferencias en la manera que abordan este problema. Señalan que desde el punto de vista de la física estadística dos aspectos son importantes: el primero es la complejidad de las transiciones de fase, que son abruptos cambios en la complejidad cuando un parámetro es variado; el segundo aspecto es el rol que juegan las propiedades de las instancias bajo estudio, ya que algunas propiedades de las instancias pueden implicar la ineficiencia de toda una clase de algoritmos. Esta última es una de las principales preguntas abiertas: ***cómo el desempeño está relacionado con propiedades de las instancias del problema y no de los detalles de los algoritmos de búsqueda.***

Juhos y van Hemert [2008] colorean grafos contrayendo nodos, haciendo hípernodos. La secuencia en la cual los nodos se contraen define el espacio de búsqueda en términos de permutaciones. El problema de coloreo de grafos es conocido por exhibir un incremento en su dificultad a medida que la densidad se incrementa para un determinado número de colores se vuelve no-soluble. El mezclado se basa en propiedades locales, lo que hace cambiar las propiedades del nodo, el orden en que se mezclan definen el espa-

cio de búsqueda, por lo tanto *la dificultad está dada por dos factores: las propiedades topológicas y el orden de mezclado*.

En trabajos anteriores, van Hemert [2006] hace hincapié que ejecutar algoritmos sobre *instancias estándar de prueba* (en inglés: *benchmark*) tiene una reducida contribución, debido a que sólo se analizan resultados del desempeño sobre esas instancias. Propone una metodología basada en algoritmos evolutivos, en donde apartir de instancias difíciles se generan más instancias difíciles. La manera en la cual evalúa la dificultad de las instancias es con el número de operaciones realizadas por los algoritmos para resolver el problema sobre esas instancias. van Hemert [2006] analiza las propiedades estructurales para explicar la dificultad de estas instancias en términos de estas propiedades. Trabaja con tres problemas, para los cuales la propiedades estructurales de las instancias son diferentes.

3.2 DISCUSIÓN

Diversos investigadores han abordado el problema de la complejidad inherente de las instancias del problema. Sin embargo para problemas relacionados con grafos no se ha llegado a una conclusión acerca de qué propiedades estructurales — para grafos del mismo tamaño y orden — influyen en la dificultad o facilidad para resolver una instancia dada del problema.

Si bien se sabe que ciertos parámetros del problema así como el tamaño de la instancia influyen en la dificultad de solución, los estudios realizados tratan con instancias que no son comparables en términos de tamaño, lo que dificulta la identificación de propiedades estructurales. Ésta es una de las principales inquietudes de este trabajo: usar instancias del mismo tamaño pero que tengan diferente estructura para observar y analizar la manera en que ésta influye en la solución del problema.

Para lograr responder la pregunta de qué propiedades estructurales en grafos afectan la manera en que se resuelve un problema es necesario establer un marco de trabajo efectivo y práctico que ayude a identificar estas propiedades.

CAPÍTULO 4

MÉTODO DE ESTUDIO

Si supiese qué es lo que estoy haciendo
no le llamaría investigación ¿verdad?
- Albert Einstein.

En este capítulo se presenta el método de estudio llevado acabo para desarrollar esta tesis. El objetivo de este capítulo es *proponer un marco de trabajo* que sea aplicable a cualquier problema en el que se desee estudiar el *efecto de la estructura* de las instancias sobre el *desempeño de los algoritmos* que resuelven el problema, de manera que se puedan establecer relaciones entre propiedades estructurales y complejidad de solución.

Las actividades llevadas a cabo para lograr el objetivo planteado se pueden dividir en seis etapas las cuales se muestran en la Figura 4.1 y se discuten a detalle en las siguientes secciones. La primera etapa (Sección 4.1) es ***elegir el problema computacional*** para realizar el estudio. La segunda (Sección 4.2) y tercera (Sección 4.3) etapa consisten en ***recopilar y caracterizar instancias*** de igual tamaño pero diferente estructura. La cuarta etapa (Sección 4.4) corresponde a la ***elección de algoritmos*** para solucionar el problema elegido y estudiar el desempeño algorítmico sobre las instancias generadas. En la quinta etapa (Sección 4.5) se *caracteriza el desempeño algorítmico*; para esto se desarrolla una ***medida de desempeño*** basada en cómo convergen los algoritmos hacia una solución y que sea independiente del algoritmo y del problema. Por último, la sexta etapa (Sección 4.6) consiste en ***establecer relaciones*** entre las propiedades estructurales y el desempeño de los algoritmos.

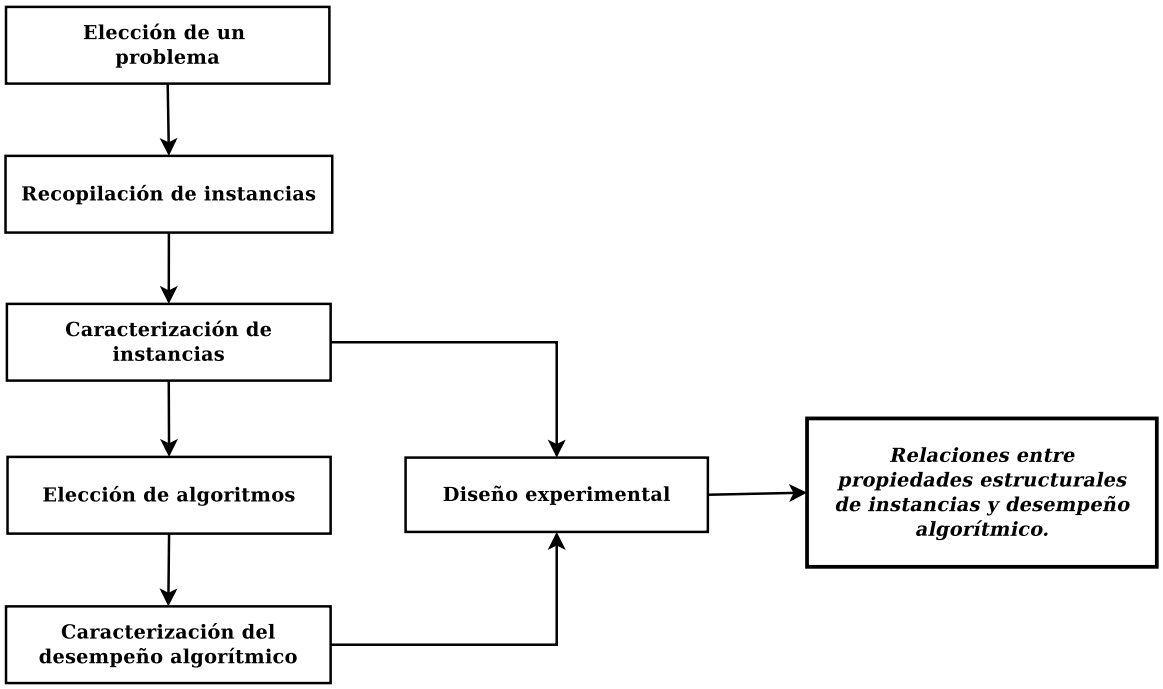


Figura 4.1: *Pasos a seguir para establecer relaciones entre propiedades estructurales de instacias propuestas de un problema y el desempeño de los algoritmos que solucionan el problema.*

4.1 ELECCIÓN DEL PROBLEMA COMPUTACIONAL

La elección del problema computacional es un paso fundamental debido a que el problema computacional elegido determina las instancias que se usarán para el estudio y su caracterización y los algoritmos usados para estudiar el desempeño algorítmico. Por esta razón se recomienda elegir un problema del cual se esté familiarizado o en su defecto que sea un problema ampliamente estudiado en la literatura existente con abundantes resultados teóricos y prácticos y para los cuales existan instancias de prueba; este último punto es discutido en la siguiente sección.

También es recomendable que otros problemas computacionales puedan reducirse a este problema. Esto será informativo ya que los resultados obtenidos se puedan extender, al menos en algún grado, a otros problemas; esto gracias a la teoría de la **NP**-completez (véase Sección 2.3.5), la cual ha demostrado que muchos problemas aparentemente diferentes pueden ser mapeados a otro de tal manera que las soluciones se preservan bajo el mapeo. Cheeseman *et al.* [1991] reportan que la dificultad de las instancias de un problema se mapean a instancias difíciles en otro problema cuando se realiza una reducción y que

las propiedades de estas instancias se preservan en la transformación. Selman *et al.* [1992] muestran resultados en los cuales al reducir una instancia difícil de un problema a otro la dificultad se mapea. Teniendo una instancia difícil para un problema **NP**-completo se puede obtener una instancia difícil para cualquier problema completo a través de reducciones [Gutfreund *et al.*, 2007; Walsh, 2009].

4.2 RECOPILACIÓN DE INSTANCIAS

Una de las principales líneas de investigación dentro del área de las ciencias computacionales es la tarea de diseñar e implementar algoritmos para solucionar problemas computacionales de decisión y de optimización. Una manera de poner a prueba los algoritmos diseñados para resolver el problema o mostrar superioridad sobre otros algoritmos es utilizar instancias estándar de prueba que se encuentran disponibles en línea.

Las instancias proveen una herramienta con la cual la comunidad científica, pueden hacer comparaciones de desempeño entre algoritmos, ya que (como se habló en la Sección 2.6.1) la diversidad de equipos en los cuales se realizan las experimentaciones computacionales y detalles de implementación son factores no proveen un ambiente de igualdad de condiciones. Sin embargo con instancias de prueba los resultados referentes a las soluciones obtenidas pueden ser comprables en términos de su calidad.

En general se usan instancias de diferentes tamaños clasificándolas como pequeñas, medianas o grandes e identificando algunas como instancias retadoras, es común estudiar el desempeño del los algoritmos sobre grandes instancias ya que (como se aclaró en la Sección 2.3.1) el tamaño de la instancia tiene un impacto en el tiempo de cómputo requerido para resolver el problema. Sin embargo para algunas instancias de gran tamaño en las cuales se espera que el tiempo de cómputo requerido sea mayor, diversos algoritmos obtienen soluciones de buena calidad en poco tiempo, es decir, las instancias son fáciles de resolver, lo que sugiere que la estructura de la instancia de alguna manera influye en la facilidad o dificultad de resolverla.

El objetivo de este trabajo es identificar qué propiedades estructurales en las instancias hacen que sean fáciles o difíciles de resolver. Para ello es importante contar con ***instancias de igual tamaño pero diferente estructura***. En este trabajo no se pretende mostrar superioridad entre algoritmos sino la afectación de las propiedades estructurales de las instancias en el comportamiento de uno o varios algoritmos. En caso que las instan-

cias de prueba disponibles no cumplan con estas características, es importante desarrollar herramientas que nos permitan generar instancias con las características deseadas.

4.3 CARACTERIZACIÓN DE INSTANCIAS

Una vez seleccionado el problema computacional que se desea abordar y recopilar las instancias con las características deseadas — mencionadas en la sección anterior — es importante capturar cuantitativamente propiedades estructurales de las instancias. La caracterización de instancias es un paso importante en este trabajo, ya que el objetivo perseguido es identificar qué propiedades referentes a la estructura de la instancia afectan el desempeño de los algoritmos ya sea facilitando o dificultando la solución del problema. De aquí que si se detectan diferencias en el desempeño de los algoritmos, éstas pueden relacionarse con determinada estructura de la instancia, no únicamente con el tamaño.

Esto también lo señalan Smith-Miles *et al.* [2009] y Pérez *et al.* [2008] quienes buscan establecer relaciones entre estructura y desempeño algoritmico con el objetivo de seleccionar algoritmos. Como se mencionó en la Sección 4.1 la elección del problema es la que marcará la pauta referente a las propiedades que se desean caracterizar de las instancias.

4.4 ELECCIÓN DE ALGORITMOS

El objetivo principal de este trabajo es *identificar aspectos presentes en la estructura de las instancias que influyan en la facilidad o dificultad para resolver el problema*. Cabe señalar que ***no*** se pretende evaluar el desempeño de los algoritmos para establecer que algoritmo es mejor que otro. Los algoritmos en sí no son el objeto de estudio de este trabajo; son el medio con el cual se realiza el estudio y por tanto deben poseer ciertas caracterísstica que se mencionan a continuación.

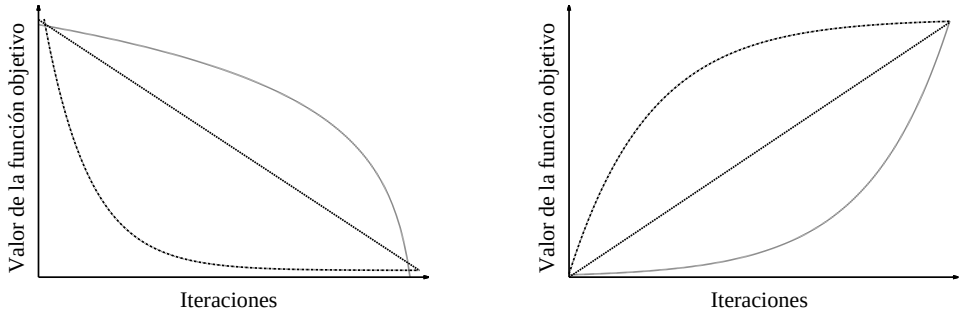
Es importante trabajar con algoritmos que aborden de diferente manera el problema elegido; nos interesa trabajar con algoritmos del estado del arte que formulen de diferente manera la función objetivo y la vecindad de soluciones. Como se mencionó en la Sección 2.6.2, tanto la formulación de la función objetivo y la manera de generar vecindades de alguna manera inducen ciertas características del espacio de búsqueda. Si la estructura de la instancia tiene un efecto significativo en el desempeño algorítmico, se podrá establacer también si determinada propiedad estructural hace que nuestro espacio de búsqueda sea

más rugoso o más plano, de manera tal que las propiedades estructurales guíen el diseño de estrategías de búsqueda que saquen el mayor provecho de este conocimiento.

4.5 CARACTERIZACIÓN DEL DESEMPEÑO

Para poder identificar el efecto de la estructura de la instancia, es importante caracterizar el desempeño del algoritmo en base a cómo converge hacia una solución, esto dependiendo del objetivo que se desea alcanzar minimizar o maximizar. Para ello se espera que durante la ejecución vaya hacia la dirección deseada.

Por ejemplo, en la Figura 4.2(a) se muestra el comportamiento deseable en problemas de optimización donde el objetivo es minimizar. El eje de las x representa las iteraciones y el eje de las y representa el valor de la función objetivo. Si al algoritmo se le dificulta resolver la instancia, tardará más para que el valor de la función objetivo disminuya (línea superior). Si le es relativamente fácil, gradualmente va disminuyendo el valor de la función objetivo (línea media). Si le es muy fácil, rápidamente disminuirá el valor de la función objetivo (línea inferior). Al utilizar algoritmos heurísticos se busca mejorar rápido para una terminación oportuna.



(a) *Convergencia en problemas de minimización.*

(b) *Convergencia en problemas de maximización.*

Figura 4.2: *Comportamiento algorítmico que se desea caracterizar.*

El comportamiento deseable e indeseable de algoritmos en problemas de maximización se muestra en la Figura 4.2(b). Si al algoritmo se le dificulta resolver la instancia tardará más para que el valor de la función objetivo aumente (línea inferior). Si le es relativamente fácil, gradualmente va aumentando el valor de la función objetivo (línea media). Si le es muy fácil, rápidamente aumentará hacia el valor de la función objetivo (línea superior).

4.6 DISEÑO EXPERIMENTAL

El *diseño y análisis de experimentos* es una herramienta estadística importante que nos permite planear, ejecutar e interpretar un experimento para obtener conclusiones válidas y objetivas acerca del problema que se está estudiando. Un *experimento* se define como una prueba planeada donde se introducen cambios controlados en las variables de entrada de un proceso o sistema para observar el efecto de esos cambios sobre la salida o *variable respuesta* del proceso o sistema. Esto se ejemplifica en la Figura 4.3.

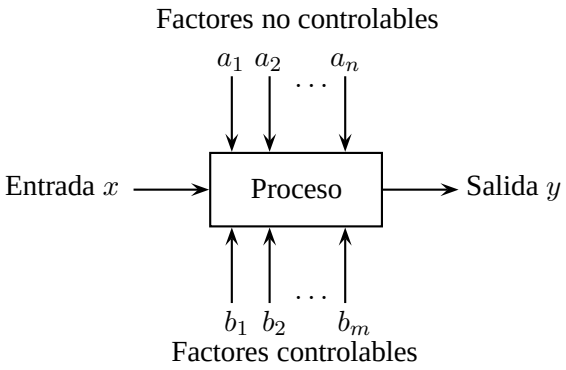


Figura 4.3: *Representación del proceso.*

La realización de un experimento tiene como objetivo determinar qué variables de entrada pueden tener mayor influencia en la respuesta o determinar valores de las variables de entrada de manera que se produzca una respuesta deseada o que la variabilidad de la respuesta sea muy pequeña.

Para llevar a cabo un buen experimento se recomienda seguir un esquema general de procedimientos; para llevar a cabo este trabajo de investigación se sigue el esquema presentado por Montgomery [2006] el cual se enlista a continuación:

1. Identificar y enunciar el problema.
2. Elegir factores, niveles y rangos.
3. Seleccionar de variables respuesta.
4. Elegir el diseño experimental.
5. Realizar del experimento.

6. Analizar estadísticamente los datos.

7. Dar conclusiones y recomendaciones.

En muchos experimentos se tiene principal interés en estudiar los *efectos* o la influencia de uno, dos o más *factores* sobre una o varias *variables respuesta*, estos factores pueden ser controlables o no, cualitativos o cuantitativos. Cada factor tiene *niveles*, es decir, valores específicos dentro de un rango definido por el experimentador. Las variables respuestas deben ser cuantificables de manera que provean información útil acerca del proceso que se desea estudiar. Es importante determinar el número de *réplicas* del experimento, es decir, la cantidad de repeticiones independientes que se harán de experimento con cada combinación de niveles de los factores.

CAPÍTULO 5

SOLUCIÓN PROPUESTA

La formulación de un problema,
es más importante que su solución.
- Albert Einstein.

En este capítulo se presenta el *marco de trabajo* propuesto para abordar el estudio de propiedades estructurales en instancias que afectan el desempeño algorítmico. En la Sección 5.1 se define el tipo de problemas que abordaremos; estos estan relacionados con grafos. Posteriormente en la Sección 5.2 se enuncian las diversas consideraciones que se deben tomar en cuenta al generar instancias de grafos de igual tamaño y orden pero diferente estructura y la Sección 5.3 describe qué considerar al caracterizarlas. En la Sección 5.4 se discuten las características a tomar en cuenta cuando se eligen algoritmos para abordar el estudio. En la Sección 5.5 se discute como las medidas tradicionalmente usadas para medir el desempeño de los algoritmos no son adecuadas para el estudio que se realiza en este trabajo y se propone una medida de caracterización basada en la convergencia de los algoritmos hacia una determinada solución. Finalmente en la Sección 5.6 se detallan aspectos que se deben considerar para establecer relaciones entre las propiedades estructurales de las instancias y el desempeño algorítmico. Este marco de trabajo se aplica para un problema en particular en el Capítulo 6.

5.1 ELECCIÓN DEL PROBLEMA

Para realizar este trabajo, se acotó la población de estudio a instancias de problemas computacionales relacionados con grafos, introducidos en Sección 2.1. Los grafos son una de las herramientas más importantes en las ciencias computacionales, matemáticas, ingeniería y muchas otras disciplinas ya que proporcionan una representación abstracta

de relaciones entre elementos que conforman una amplia variedad de sistemas [Skiena, 2008]. Otro aspecto que jugó un importante papel en la decisión de trabajar con grafos fue que en literatura existen una amplia cantidad modelos de generación de grafos con diferente estructura y funciones que nos permiten caracterizar propiedades estructurales de los grafos, discutidos en la Sección 2.2.

5.2 GENERACIÓN DE INSTANCIAS

Para aplicar el marco de trabajo propuesto y lograr el objetivo planteado en este trabajo, es necesario que las instancias de prueba cumplan con ciertas caracteristicas que es tener grafos con el mismo orden n y tamaño e pero diferente estructura. Para el problema computacional elegido existen instancias de prueba que no cumplen con estas caracteristicas, si bien se cuenta con grafos de diferentes estructuras, entre los grafos que comparten la misma estructura no tienen el mismo tamaño o se cuenta con pocas instancias para realizar comparaciones significativas entre ellas. Con el fin de observar efectos estructurales en la solución del problema y poder realizar comparaciones, se utilizan modelos de generación de redes complejas, los cuales proporcionan una herramienta para *generar instancias de grafos controlando el orden y tamaño pero con diferente estructura*.

Los modelos utilizados para generar las instancias de grafos son: el modelo de Erdős-Rényi (ER), el de Watts-Strogatz (WS), el de Barabási-Albert (BA), el de Kleinberg (KL) y el modelo de grafos geométricos aleatorios (RGG). Los modelos fueron implementados en lenguaje Java [Gosling *et al.*, 2005]. Se desarrolló una herramienta que para grafos menores a 100 nodos pueden visualizarse a través de su interfaz gráfica. A través de la terminal o consola pueden generarse grafos de hasta $4{,}900$ nodos. Las instancias de grafos tienen el formato DIMACS [2010].

Es importante trabajar con conjuntos de grafos de diferente orden, a medida que el orden de los grafos va creciendo también crece el número de aristas que se pueden tener en los grafos. Para fines de comparación, en lugar de manejar el número de aristas e, se trabajará con la densidad $\mathit{den}(\mathcal{G})$ de los grafos, es decir, con la proporción de aristas presentes de las máximas posibles, dependiendo del número de nodos con el que se generen los grafos.

Una parte importante en la generación de instancias es el ajuste de parámetros de los modelos usados de manera que se generen instancias con el mismo número de nodos y de aristas. Algunos parámetros de entrada imponen ciertas restricciones para establecer

el orden y densidad de los grafos. Por una parte el modelo KL utiliza una malla de $s \times s$ nodos, por lo cual para cualquier modelo el orden de los grafos es restringido por el parámetro s del modelo KL, es decir, $n = s^2$. Restricciones en cuanto al número mínimo de aristas que pueden tener los grafos son impuestas por el modelo WS y KL al crear el anillo y la malla respectivamente. A partir de las restricciones dadas por estos modelos se obtuvieron los parámetros para cada uno de los modelos mencionados.

Un aspecto importante es trabajar con grafos conexos: si se generaran grafos no conexos se tendrían instancias fáciles, ya que cada componente no conexo equivaldría a resolver el problema para cada componente que se tenga. Los modelos KL, WS y BA construyen grafos conexos, pero los modelos ER y RRG al generar grafos de baja densidad pudiesen resultar grafos no conexos. Debido a que verificar la conexidad de los grafos al momento de generarlos es costoso computacionalmente, se hicieron adecuaciones a los modelos ER y RGG para asegurar que grafos de baja densidad sean conexos. Para el modelo ER se generó primeramente un árbol abarcante (en inglés: *spanning tree*) con lo cual se tienen $n - 1$ aristas en el grafo y después se aplicó el modelo tal cual para agregar las aristas faltantes para completar la densidad deseada. En el caso del modelo RGG primero se generó un árbol mínimo abarcante (en inglés: *minimum spanning tree*) y después se agregaron aristas entre nodos más cercanos hasta obtener la densidad deseada. El modelo de BA no especifica el número de nodos iniciales ni como deben estar conectados para empezar a construir los grafos; con el fin de no inducir una estructura en particular a los grafos generados con este modelo, se inicia la generación de grafos con dos nodos conectados, lo cual limita la cantidad de aristas que se pueden agregar en los grafos.

5.3 CARACTERIZACIÓN DE INSTANCIAS

Generalmente cuando se trabaja con instancias de grafos se presta principal atención al número de nodos n y el número de aristas e, el grado mínimo $\delta(\mathcal{G})$, el grado máximo $\Delta(\mathcal{G})$ y el grado promedio $\langle k \rangle$. Esto proporciona información acerca de la distribución del grado, pero esta información no es suficiente; como discute [Alderson, 2008], se pueden tener grafos con la misma distribución del grado pero diferente estructura, por lo que es importante contar con funciones de caracterización que nos permita tener una idea de *cómo es la estructura del grafo que estamos estudiando*. Las funciones de caracterización usadas son las descritas en la Sección 2.2.1.

5.4 ELECCIÓN DE ALGORITMOS

Los algoritmos seleccionados para estudiar el efecto de la estructura de las instancias en el desempeño algorítmico deben obtener buenas soluciones para el problema que se desea resolver. Además es deseable que formulen de diferente manera la función objetivo o en que generen de manera diferente las vecindades. También es importante seleccionar algoritmos que no tengan demasiados parámetros a ajustar que estén relacionados con el funcionamiento del algoritmo o en su defecto seleccionar aquellos en los cuales se proveean información suficiente acerca de los valores de parámetros con los cuales se ha observado que los algoritmos tengan el mejor desempeño [1]. Lo importante en este punto es tener cierta diversidad en cuanto a estrategias de solución con el fin de observar el efecto de la estructura de la instancia independientemente del los aspectos mencionados.

5.5 CARACTERIZACIÓN DE DESEMPEÑO ALGORÍTMICO

Cuando se evalúa el desempeño de un algoritmo no se puede evitar realizar comparaciones con otros algoritmos que resuelven el mismo problema utilizando instancias de prueba. Para realizar estas comparaciones se realizan un gran número de repeticiones midiendo el tiempo de cómputo promedio, el número de iteraciones promedio realizadas por el algoritmo, la media y desviación estándar de las soluciones obtenidas y la mejor solución obtenida.

Diversos investigadores se han dado a la tarea de generar guías para el experimentador acerca de cómo presentar los resultados obtenidos cuando se evalúa el comportamiento de algoritmos de manera que no se lleguen a malas interpretaciones de estas medidas, por qué no usar ciertas medidas o qué tipo de media — aritmética o geométrica — utilizar [Barr *et al.*, 1995; Birattari y Dorigo, 2007; Fleming y Wallace, 1986].

En nuestro caso estas medidas de desempeño no son útiles para lograr el objetivo perseguido en este trabajo. La medida de desempeño que se desea utilizar no debe verse afectada por el equipo en el cual se ejecutan los algoritmos ni por el lenguaje de programación utilizado. Estos dos aspectos impactan en mediciones relacionadas con el tiempo de cómputo. Si bien el tiempo de cómputo indica la velocidad de convergencia a una solución, no indica cómo fue esa convergencia. Por otro lado, el desempeño del algoritmo

[1] En caso de tener un algoritmo que tenga como entrada varios parámetros y no se tenga información acerca de los valores con los cuales ha tenido buen desempeño, se deberá hacer un estudio orientado al ajuste de parámetros.

también se ve afectado por factores como la *formulación de la función objetivo* y la generación de *vecindades*. Estas soluciones obtenidas al final de la ejecución si bien sirven para medir la robustez del algoritmo, no indican cómo fue el desempeño del algoritmo durante la ejecución [Barr *et al.*, 1995].

Como se discutió en la Sección 4.5, para estudiar el efecto de la estructura de una instancia es necesario usar una medida que permita abstraer cómo se llego a una solución. Recordemos que el objetivo principal de este trabajo es determinar propiedades estructurales en instancias de grafos que influyen en la complejidad de resolución de la instancia.

En primer lugar se desea identificar estructuras de grafos para las cuales la complejidad de solución es mayor. Para ello es importante generar instancias del mismo tamaño que posean diferente estructura, tal como se discutió en las Secciones 4.2 y 5.2. Además es importante fijar parámetos de manera que sea posible observar el efecto de la estructura de la instancia en la manera en que el valor objetivo disminuye o aumenta según sea el caso.

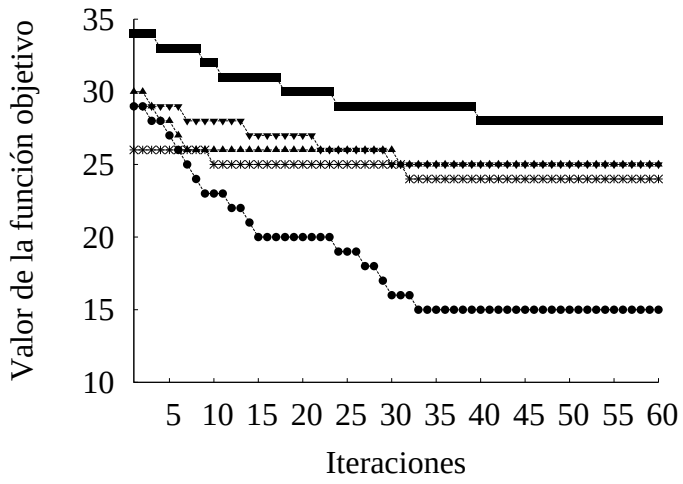


Figura 5.1: *Comportamiento de un algoritmo que resuelve un problema con cinco instancias de igual tamaño pero diferente estructura.*

Al trabajar con diferentes algoritmos se desea determinar si una estructura de grafo influye de manera similar independientemente de la formulación de la función objetivo y generación de vecindades. Con estas consideraciones lo que se busca es aislar el efecto que la estructura tuviese en el desempeño. En la Figura 5.1 se muestra el comportamiento de un algoritmo configurado con los mismos parámetros para resolver un problema para instancias de igual tamaño pero diferente estructura. Las soluciones finales pueden

ser distintas o similares, pero el cómo se llegó a esas soluciones es diferente. Para lograr identificar efectos de la estructura se propone obtener medidas de desempeño caracterizando el *perfil de desempeño*, es decir, el valor de la función objetivo en cada iteración del algoritmo.

Al momento de estar trabajando con algunas instancias de prueba y los algoritmos seleccionados para obtener el valor de la función objetivo en cada iteración, se considero hacer una regresión cuadrática con los valores obtenidos en cada ejecución del algoritmo, como se muestra en la Figura 5.2, como puede observarse al hacer el ajuste a una fucnión cuadrática se pierde información, por ejemplo, de las veces que el algoritmo obtuvo la misma solución durante varias iteraciones.

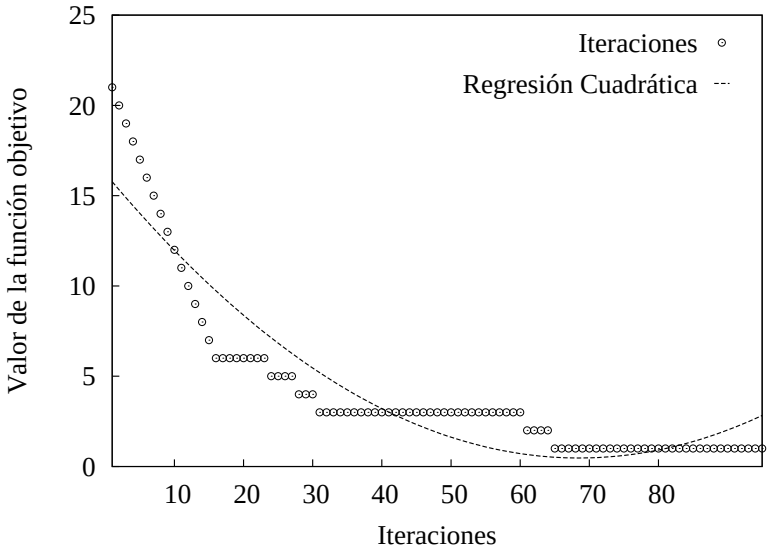


Figura 5.2: *Caracterización del perfil de desempeño mediante regresión cuadrática.*

En segundo lugar se consideró utilizar el promedio de la *pendientes*[2] en el perfil de desempeño, pero se puede tener dos perfiles de desempeño cuya convergencia es totalmente distinta y tener la misma pendiente promedio, como se observa en la Figura 5.3. Ante este caso se consideró otorgar cierta *ponderación* a las pendientes que fuera disminuyendo cierta λ a medida que cambios en la pendiente fueran ocurriendo, el principal inconveniente de esto es determinar el valor de λ ante la incertidumbre de cuantos cam-

[2]Las pendientes en el perfiles de desempeño son las diferentes inclinaciones que se presentaron en el perfil de desempeño. Interpretada geométricamente, la pendiente es la razón de cambio en el valor de la función objetivo al ir iterando el algoritmo, es deseable que para problemas de minimización la pendiente sea negativa y para problemas de maximazación la pendiente sea positiva.

bios de pendiente en el perfil de desempeño se pueden tener. En los perfiles ilustrados en la Figura 5.3 ambos tienen tres cambios de pendiente que equivalen a tres segmentos de recta, en la Tabla 5.1 se listan las pendientes de cada segmento según el orden de occurrencia.

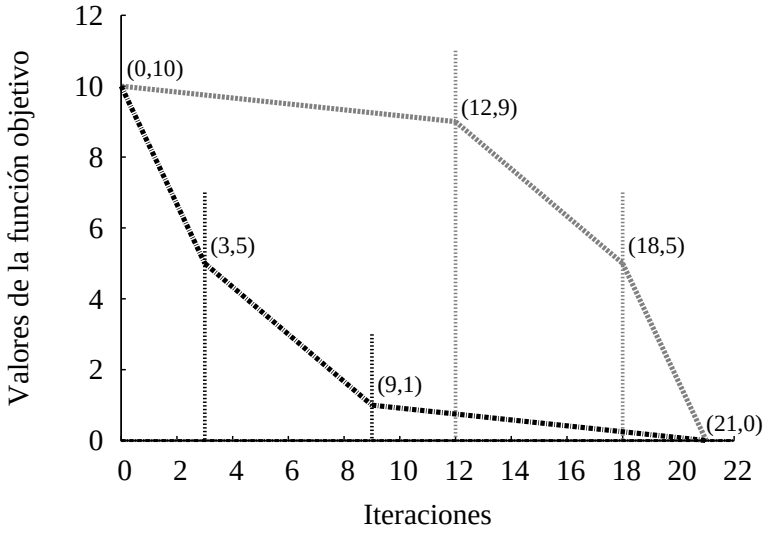


Figura 5.3: *Perfiles de desempeño.*

Tabla 5.1: *Pendientes de los perfiles mostrados en la Figura 5.3.*

Perfil superior		Perfil inferior	
Segmento	Pendiente	Segmento	Pendiente
1	-1/12	1	-5/3
2	-2/3	2	-2/3
3	-5/3	3	-1/12

La caracterización del perfil de desempeño se realiza sumando el área bajo cada uno de los m segmentos de recta que forma el perfil de desempeño; esta área se calcula mediante la regla de Simpson (Ecuación 5.1) usando como límite inferior y superior los puntos donde la pendiente cambia. Ante la posibilidad de tener dos perfiles de desempeño diferentes con la misma área, se realiza una *regresión cuadrática y exponencial* para comparar entre perfiles los coeficientes de la ecuación cuadrática y también el coeficiente y exponente de la ecuación exponencial, sin embargo estas diferencias son tan pequeñas que estadísticamente son significativamente iguales. En la Tabla 5.2 se muestran las medidas de los perfiles: promedio, promedio ponderado ($\lambda = 0.02$) y el área bajo el perfil de desempeño.

$$\texttt{area\_total} = \sum_{k=1}^{m} \int_{a_k}^{b_k} f_k(x)\,\mathrm{d}x \simeq \sum_{k=1}^{m} \frac{b_k - a_k}{6}\left[f_k(a_k) + f_k\left(\frac{a_k + b_k}{2}\right) + f(b_k)\right] \quad (5.1)$$

Tabla 5.2: *Medidas de desempeño para los perfiles de la Figura 5.3.*

Medida	Perfil 1	Perfil 2
Pendiente promedio	-0.85	-0.85
Pendiente ponderada	-0.77	-0.8
Área bajo perfil	46.5	136.5

Los datos necesarios para caracterizar el desempeño algorítmico son el vector con los valores de la función objetivo en cada iteración de los algoritmos usados X y el número total de iteraciones `iter_totales`. El procedimiento para obtener el área bajo el perfil de desempeño se muestra en el Algoritmo 5.1. Esta medida puede calcularse fuera de línea y en línea sin que esto consuma demasiados recursos computacionales.

Algoritmo 5.1: Área bajo el perfil de desempeño.

Entrada: Un vector X de tamaño `iter_totales`
Salida: El área bajo el perfil de desempeño

Inicio
  `aux` $\leftarrow 0.0$;
  `area_total` $\leftarrow 0.0$;
  `lim_inf` $\leftarrow$ `lim_sup` $\leftarrow 0$;
  Desde `i` $\leftarrow 1$ **to** `total_iter`-1 **hacer**
    `pendiente` $\leftarrow (X[$`i`$+1] - X[$`i`$])/($`i`$+1-$`i`$)$;
    Si `aux` $\neq$ `pendiente` **entonces**
      `lim_sup` $\leftarrow$ `i`;
      `const` $\leftarrow$ (`pendiente`$*-$`i`$) + X[$`i`$]$;
      `area_total` $\leftarrow$ `area_total` $+$ ((`lim_sup`$-$`lim_inf`$)/6)$ $*$ (`pendiente` $*$ `lim_inf` $+$ `const`) $+$ $4 *$ ((`pendiente`$*($`lim_inf`$-$`lim_sup`$)/2)+$`const`) $+$ (`pendiente`$*$`lim_sup`$+$`const`);
    fin-si
    `lim_inf` $\leftarrow$ `lim_sup`;
    `aux` $\leftarrow$ `pendiente`;
  fin-desde
  Devuelve `area_total`
Fin

La idea de caracterizar el perfil de desempeño mediante el área debajo del perfil, pudiera asociarse al área bajo la curva ROC[3] la cual se emplea como un índice de la exactitud de un sistema clasificador binario [Mason y Graham, 2002]. Sin embargo el área debajo del perfil de desempeño, lo se ve como una probabilidad si no como una cantidad que denota que tan difícil fue para un algoritmo resolver una instancia. En el caso de problemas de minimización a mayor área más dificil fue resolver el problema, a menor área más fácil fue resolver el problema; en el caso de problemas de maximización, a mayor área más fácil fue resolver el problema, a menor area más difícil fue resolver el problema.

5.6 DISEÑO EXPERIMENTAL

Contando con las propiedades estructurales y los indicadores de desempeño, es importante determinar si la estructura de la instancia tiene un efecto significativo en el desempeño algorítmico, determinar cuales propiedades estructurales son las que influyen de manera importante y poder establecer relaciones entre estas dos variables de estudio. El diseño experimental deberá proporcionar información de cómo diversos factores principalmente la estructura de la instancia afectan la complejidad de solución independientemente de los parámetros del problema y particularidades del algoritmo.

Una vez elegido el diseño experimental, es importante analizar los datos que se generan a partir de la caracterización de instancias y desempeño algorítmico para determinar el tipo de pruebas estadísticas a utilizar ya sean paramétricas o no paramétricas. Debido a la gran cantidad de datos que se generan, es importante considerar la utilización de técnicas multivariadas tanto para reducir de dimensionalidad de las variables si es requerido, así como para explicar las relaciones existentes entre propiedades estructurales de las instancias y desempeño algorítmico.

[3]Una curva ROC — Característica Operativa del Receptor (en ingles: Receiver Operating Characteristic) — es una representación de *sensibilidad* frente a la *especificidad* para un sistema clasificador binario (positivo - negativo) según varia el umbral de discriminación. La sensibilidad es la probablidad de clasificar correctamente a un individuo cuya clase real sea positiva y la especificidad es la probabilidad de clasificar correctamente a un individuo cuya clase real sea negativa [Zou *et al.*, 2007].

CAPÍTULO 6

CASO DE ESTUDIO

La mejor estructura no garantizará
los resultados ni el rendimiento.
Pero la estructura equivocada
es una garantía de fracaso.

- Peter Drucker.

En esta capítulo se evalúa tanto la medida de desempeño algorítmico propuesta, como la aplicación de la metodología en un caso de estudio. Se presentan resultados de cómo la medida de desempeño sirve para identificar estructuras de grafos que tienen mayor impacto en el desempeño algorítmico.

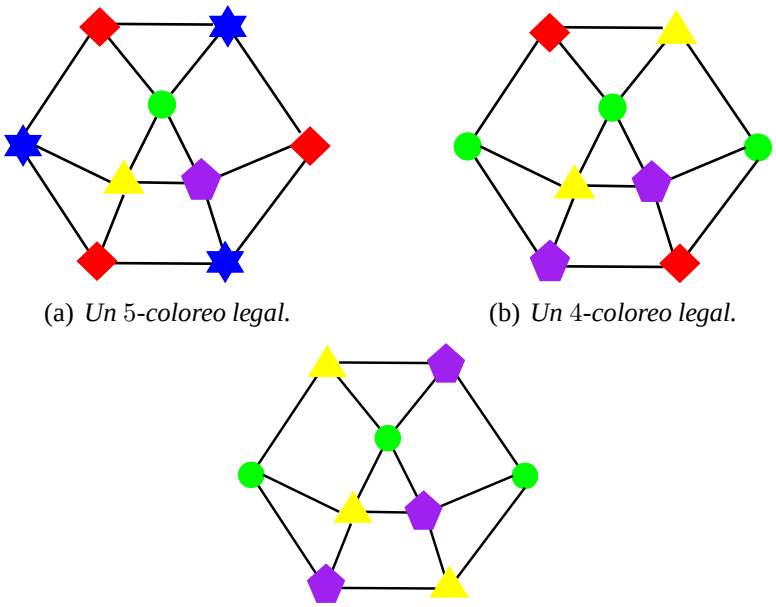
En la Sección 6.1 se presenta el problema seleccionado para aplicar el marco de trabajo propuesta en el capítulo anterior, así como la medida de desempeño propuesta. Primeramente se describe el problema a estudiar. En la Sección 6.2 se presenta información acerca del tamaño y orden de las instancias de grafos generadas y su caracterización. En la Sección 6.3 se describen los algoritmos seleccionados para el estudio y las razones por las cuales se seleccionaron. Además se describe cómo se efectuaron las experimentaciones y cómo se obtuvieron las medidas de caracterización del desempeño algorítmico. Finalmente en la Sección 6.5 se presentan los resultados obtenidos al caracterizar el perfil de desempeño con la medida propuesta

6.1 ELECCIÓN DEL PROBLEMA: COLOREO DE GRAFOS

Formalmente el problema de coloreo de grafos se plantea de la siguiente manera: dado un grafo $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ y un entero k, un k-coloreo de $\mathcal{G}$ es una función

$$c : V_\mathcal{G} \rightarrow \{1, \ldots, k\} \mid c(u) \neq c(v), (v, u) \in V_\mathcal{G}. \tag{6.1}$$

Los nodos con color $i(1 \leq i \leq k)$ definen una *clase de color*. Si dos nodos adyacentes tienen el mismo color se dice que se tiene un *conflicto*. Un k-coloreo sin conflictos se dice que es *legal*, ejemplos de coloreos legales se ilustran en la Figura 6.1.



(a) *Un* 5*-coloreo legal.* (b) *Un* 4*-coloreo legal.*

(c) *Un* 3*-coloreo legal.*

Figura 6.1: *Diferentes coloreos legales para un grafo con* 9 *nodos y* 15 *aristas.*

Al número mínimo de colores con el que se puede colorear un grafo sin tener conflictos se denota como $\chi(\mathcal{G})$ el *número cromático* de $\mathcal{G}$; un ejemplo es presentado en la Figura 6.1(c). Este problema es aplicado a la asignación de radiofrecuencias [Marx, 2004] y a la de calendarización y asignación de actividades [Trick, 2010]. Es importante señalar que cada clase de color es un conjunto independiente[1] de nodos, por lo que una k-coloración es lo mismo que una partición del conjunto de nodos en k conjuntos independientes y los términos k-partición y k-coloreo tienen el mismo significado.

6.2 GENERACIÓN Y CARACTERIZACIÓN DE INSTANCIAS

En una primera etapa se generaron instancias de grafos de media y alta densidad. Información acerca del número de nodos y densidades de las instancias generadas con los modelos de ER, BA, KL, RGG y WS, se muestra en la Tabla 6.1, la cantidad de aristas puede calcularse de la siguiente manera $(n \cdot (n-1)/2) \cdot \mathit{den}(\mathcal{G})$; por cada modelo para

[1]Un conjunto independiente es un conjunto de nodos en un grafo tal que ninguno es adyacente a otro.

cada número de nodos y densidades se generaron 30 instancias dando un total de $1{,}800$ instancias de alta densidad.

Tabla 6.1: *Información acerca de los grafos con alta densidad.*

n	*den*$(\mathcal{G})$	n	*den*$\mathcal{G}$	n	*den*$(\mathcal{G})$
64	0.2	256	0.2	1,024	0.2
	0.4		0.4		0.4
	0.6		0.6		0.6
	0.8		0.8		0.8

También se generaron 30 instancias de grafos con un mayor número de nodos y de baja densidad por cada uno de los modelos de ER, BA, KL, RGG y WS, debido a que las redes del mundo real cuentan con estas características [Albert, 2001; Newman, 2003]. Información referente a estas instancias se presenta en la Tabla 6.2, en total fueron generadas $1{,}050$ instancias de baja densidad.

Tabla 6.2: *Información acerca de los grafos con baja densidad.*

n	*den*$(\mathcal{G})$	n	*den*$(\mathcal{G})$
100	~ 0.059	2,500	0.002
400	0.015	3,600	0.001
900	0.007	4,900	~ 0.001
1,600	0.004		

Se utilizaron funciones de caracterización de redes complejas para obtener medidas de propiedades estructurales de las instancias generadas; detalles de estas funciones fueron descritas en la Sección 2.2.1. Fueron implementadas en lenguaje Java [Gosling *et al.*, 2005].

Además del número de nodos y de aristas de los grafos, los datos recabados fueron el grado promedio y la desviación estándar de los grados. Para obtener información adecuada acerca de la distribución del grado, se obtuvieron medidas estadísticas como la curtosis, asimetría, varianza, coeficiente de variación, valor máximo y mínimo y rango [Ross, 2009].

Se calcularon funciones de caracterización que proveen información global de los grafos, como coeficiente de agrupamiento, coeficiente de dispersión del grado, longitud de ruta más corta, diámetro, radio, eficiencia local y global. Estas funciones son promedios sobre información local por lo cual también se calcularon las medidas estadísticas antes mencionadas para la distribución del grado para cada una de las funciones globales. Todas

estas funciones proporcionan información cuantitativa que permite buscar un patrón en la estructura que impacte en el desempeño algorítmico.

6.3 ELECCIÓN DE ALGORITMOS Y CARACTERIZACIÓN DE DESEMPEÑO

Se tomaron algoritmos del estado del arte para la solución del problema de k-coloreo [Blöchliger y Zufferey, 2008]. Estos algoritmos formulan de diferente manera la función objetivo y la vecindad de soluciones, lo cual permite independientemente de estos dos aspectos observar la influencia de la estructura de la instancia en el espacio de soluciones generado. Además que proporcionan valores de parámetros propios de los algoritmos con los cuales se obtuvieron mejores resultados. El código de los cuatro algoritmos está disponible en `http://www.bloechligair.ch/science/`. Otro punto más por el cual se seleccionaron estos algoritmos es que permiten variar el parámetro del problema, es decir, la cantidad de colores k con la cual se desea colorear el grafo. A continuación se describen brevemente los algoritmos seleccionados:

TabuCol En este algoritmo, una solución consiste en k conjuntos $C_1 \dots C_k$. Si $v \in C_i$ entonces v tiene asignado el color i. La solución inicial es generada con un algoritmo voraz donde cada nodo v se asigna a C_i donde i es el menor valor posible sin crear un conflicto. Si esto es imposible, se selecciona aleatoriamente un conjunto C. La función objetivo es minimizar el número de aristas en conflicto. Una solución vecina se obtiene cambiando el color de uno de los nodo que introduce el conflicto. La tenencia tabú se calcula como $\text{UNIFORM}(0, 9) + \alpha * \text{NCV}(s)$ donde α la fijan en 0.6 y $\text{NCV}(s)$ representa el número de nodos que están en conflicto, por lo que la tenencia tabú es *dinámica* [Blöchliger y Zufferey, 2008].

Reac-TabuCol La función objetivo, la solución inicial y la obtención de las soluciones vecinas se definen igual que en el algoritmo TabuCol. Lo que cambia es la manera que se calcula la tenencia tabú; es ajustada de manera *reactiva* dependiendo del valor de la función objetivo. Si la función objetivo fluctúa mucho, la tenencia disminuye lentamente; en caso contrario, la tenencia se incrementa. Cuando la tenencia tabú es muy pequeña y la búsqueda local actúa como un "descenso acelerado", por otra parte cuando la tenencia tabú es grande la búsqueda se diversifica. Blöchliger y Zufferey [2008] ofrecen detalles acerca de cómo medir la fluctuación del valor objetivo.

PartialCol En este algoritmo una solución consiste en k conjuntos disjuntos estables (no contienen nodos adyacentes) $S_1, \ldots S_k$ y un conjunto $O = V_{\mathcal{G}} \setminus \bigcup_{i=1}^{k} S_i$. Si $v \in S_i$, entonces v tiene asignado el color i. Si $v \in O$, entonces v no está coloreado. Blöchliger y Zufferey [2008] discuten cómo el espacio de solucion inducido por esta solución y la obtenida mendiante el algoritmo TabuCol es diferente. Para generar la solución inicial usan un algoritmo voraz: cada nodo v se asigna a S_i donde i es el menor valor posible de manera que S_i permanezca estable. Si esto no es posible, se asigna al conjunto O. La función objetivo es minimizar la cantidad de nodos en el conjunto O. Una solución vecina se obtiene coloreando un nodo $u \in O$ con el color c, por lo que es posible que S_c no sea estable. Si esto ocurre nodos adyacentes a u se remueven de S_c de manera que se vuelta estable. La tenencia tabú es dínamica y se calcula de la misma manera que en TabuCol.

React-PartialCol La función objetivo, la solución inicial y la obtención de las soluciones vecinas se definen igual que en el algoritmo PartialCol; lo que cambia es la manera que se calcula la tenencia tabú. La tenencia tabú es ajustada de manera *reactiva* como se describe en Reac-TabuCol.

Para cada instancia se ejecutaron los cuatro algoritmos treinta veces para cada valor k elegido, esta cantidad de veces asegura contar con la cantidad de muestras suficiente para que el experimento tenga potencia alta, es decir, que el experimento pueda detectar diferencias pequeñas entre las medias que se estarán comparando. En cada ejecución se obtuvieron los perfiles de desempeño (el valor de la función objetivo en cada iteracion del algoritmo). La caracterización se realizó fuera de línea, es decir, al terminar cada ejecución.

La información obtenida en cada ejecución fueron: el área bajo el perfil de desempeño, pendiente promedio del perfil, así como pendiente promedio ponderada, resultados de regresiones cuadráticas y exponenciales, esto para fines de evaluación de la medida de desempeño propuesta. Además para fines de evaluación y comparación se obtuvo información acerca del número de iteraciones, solución final de cada ejecución, distancias entre soluciones e información acerca del espacio de soluciones obtenida con las medidas descritas en la Sección 2.6.2. Se promediaron los resultados de las treinta ejecuciones.

Se observó el desempeño de los algoritmos para diferentes parámetros del problema, k. El rango de valores para k fue determinado considerando cotas inferiores y superiores para el número cromático $\chi(\mathcal{G})$. Para obtener cotas inferiores se calculó a través de CLIQUER [Niskanen y Östergård, 2003] los tamaños de clique; para las cotas superiores se

ejecuto el algoritmo DSATUR [Brélaz, 1979] de manera que se tienen los mejores valores de $\chi(\mathcal{G})$ conocidos, para los grafos generados.

6.4 DISEÑO EXPERIMENTAL

Una vez caracterizadas las instancias mediante las propiedades estructurales y el desempeño algorítmico, se procede a establecer relaciones entre estos dos aspectos. Primeramente se clasifican los grafos en clases de acuerdo a sus propiedades estructurales y se comparan las clases resultantes con los modelos de generación empleados con el objetivo de confirmar que cada modelo generó grafos diferentes en términos estructurales.

Una vez confirmado esto, el diseño experimental se enfoca en determinar cuál o cuáles propiedades estructurales afectan significativamente el desempeño algorítmico. Primero, se examina si existen diferencias estadísticas entre la medida de desempeño sobre diferentes estructuras de instancias. Esto se logra a través de pruebas ANOVA [Miller, 1997], siempre controlando el tamaño y orden del grafo. Después se utilizan pruebas de Tukey [1991] para identificar qué clases de grafos tienen un desempeño algorítmico similar y cuáles difieren, identificando también para qué clases de grafos fué más facil o más difícil resolver el problema, los resultados obtenidos con las pruebas de Tukey se verifican con un algoritmo de agrupamiento, el cual ayuda a identificar que propiedades estructurales difieren entre las instancias fáciles y difíciles. Por último se utilizan modelos de regresión para establecer relaciones entre propiedades estructurales y la dificultad de solución.

6.5 RESULTADOS

En esta sección se presentan los resultados obtenidos al caracterizar el perfil de desempeño con la medida propuesta: *área bajo el perfil de desempeño*, sobre los dos conjuntos de instancias generadas. Se reportan resultados por grupos primero para grafos con alta densidad y después para grafos con baja densidad. En ambos casos podemos formular la siguiente hipótesis: ***La estructura de grafos de igual tamaño y orden impacta en el desempeño algorítmico***. Se presentan resultados en forma gráfica para una mejor interpretación.

6.5.1 GRAFOS CON DENSIDADES ALTAS

Por cada instancia de los modelos se ejecuto 30 veces cada algoritmo, y se trabajo con el promedio de la medida de desempeño propuesta; en la Figura 6.2 se muestran los resultados obtenidos para grafos de alta densidad con 64 nodos.

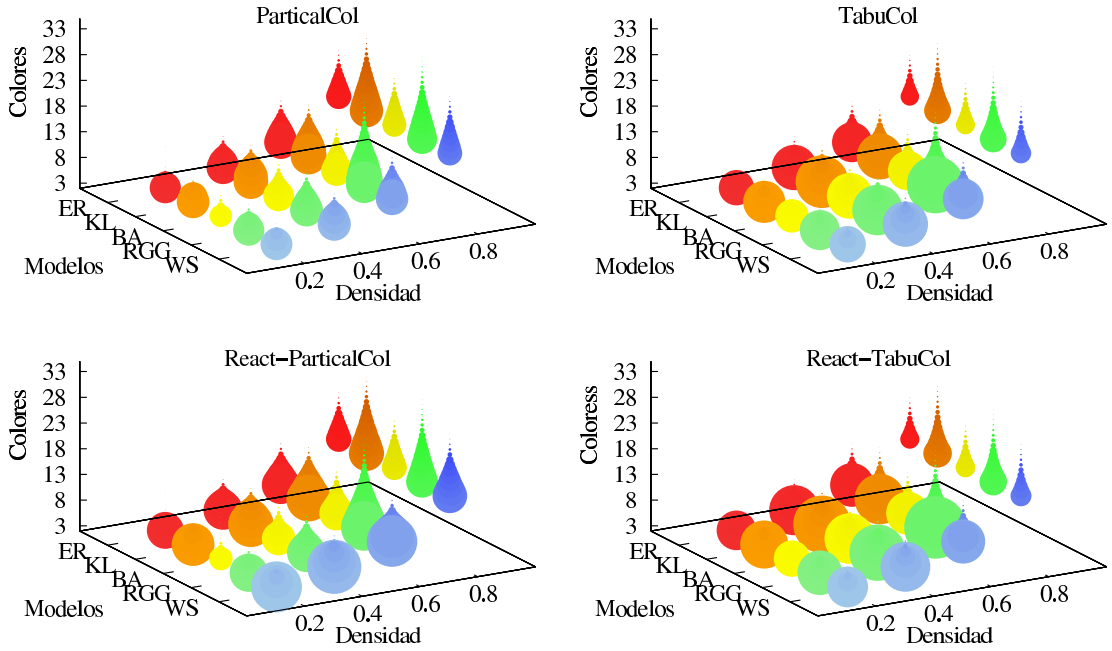


Figura 6.2: *Área bajo el perfil de desempeño para grafos con 64 nodos de mediana y alta densidad. En el eje x se representa los grafos generados con cierto modelo, en el eje y se representan las diferentes densidades de los grafos generados, en el eje z representa el número de colores con los cuales fueron coloreados los grafos, en cada coordenada (x, y, z) se dibuja un punto cuyo diámetro es proporcional al área promediada sobre las instancias generadas con el modelo x de densidad y y coloreadas con $k = z$.*

Si se pone atención a las filas (eje x) que representan cada uno de los modelos, puede observarse que la medida de desempeño refleja el efecto del tamaño de la instancia, es decir, la densidad (eje y) afecta la convergencia de los algoritmos como señalan Mulet *et al.* [2002], conforme cambia la densidad se necesitan más colores y el diámetro del punto que representa el área aumenta. También permite observar el efecto del parámetro del problema (eje z): a mayor número de colores es más fácil colorear el grafo y los algoritmos tienen una rápida convergencia; esto es estudiado por Flum y Grohe [2006], conforme el parametro aumenta el diámetro del punto que representa el área disminuye, por eso la forma de gota que se observa en la gráfica. Es interesante observar que cada modelo de generación tiene efecto similar sobre el desempeño algorítmico de los cuatro algoritmos

usados para el estudio, lo cual indica que la estructura del grafo afecta la dificultad de solución independientemente de las formulaciones de la función objetivo y la manera en la cual se definen las vecindades en el espacio de soluciones.

Las conclusiones iniciales para este primer conjunto de instancias son las siguientes: los grafos generados con el modelo BA y ER resultan más faciles de resolver que los generados con el modelo KL y el modelo RGG. Un detalle importante que es necesario mencionar en este punto es que para el modelo WS se esperaba un comportamiento similar a los grafos generados modelo de KL o RGG, porque estos modelos se caracterizan por generar grafos con un alto coeficiente de agrupamiento, pero al observar los resultados el comportamiento es similar a los grafos generados con el modelo ER. Al realizar una revisión de los parámetros usados para la generación de estos grafos, se detecto que el parámetro de la probabilidad de reconexión era un valor muy alto lo que hace que la estructura de pequeño mundo se pierda. Esto fue corregido para generar grafos con baja densidad. Es importante señalar dos aspectos: el primero es que este error al momento de fijar el parámetro permitió observar que efectivamente la medida captura el efecto de la estructura en el desempeño algorítmico y el otro asepecto es resaltar que la selección del modelo de generación no fija por si solo las propiedades estructurales, los valores de los parámetros empleados para la generación afectan la estructura del grafo.

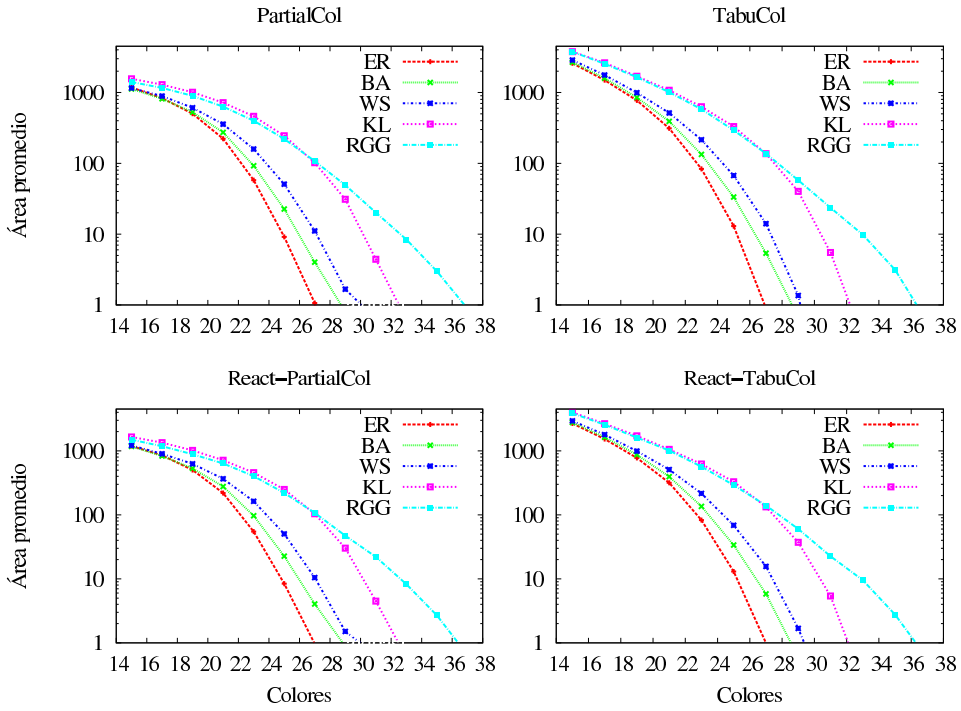


Figura 6.3: *Área bajo el perfil de desempeño en escala logarítmica para grafos con* 64 *nodos y densidad de* 0.8*.*

En la Figura 6.3 se muestra los resultados obtenidos del área bajo el pefil de desempeño para grafos con 64 nodos y densidad de 0.8, el eje y está en escala logarítmica y representa el área promedio, en el eje x se representa el parámetro k del problema. Se puede observar que el comportamiento de los algoritmos y la convergencia hacia una solución es similar, esto se debe a que *la densidad es muy alta lo que provoca que los modelos generen grafos casi completos, por lo cual diferencias estructurales se pierden para grafos de muy alta densidad independientemente del modelo usado para generarlos.* Por ello ***se concluye que trabajar con grafos de muy alta densidad no proporciona información útil para este estudio***.

6.5.2 GRAFOS CON DENSIDADES BAJAS

Para grafos con baja densidad, el efecto de la estructura en el desempeño es más evidente; esto es ilustrado en la Figura 6.4. Sin embargo, el efecto del parámetro k del problema de coloreo es dominante y no es trivial determinar que modelo de generación provee las instancias más fáciles o más difíciles apesar del parámetro k que dicta la complejidad del problema de coloreo, también en la Figura 6.4 hay que considerar que se está graficando el área bajo el perfil de desempeño — promediado sobre las instancias y las ejecuciones de los algoritmos — para grafos de diferente orden, siendo el orden del grafo otro factor dominante.

En un análisis más detallado — para grafos con determinado orden y $k = 3$ — se observa que en general los modelos BA y ER producen grafos de baja densidad más fácil de colorear; esto se observa en la Figura 6.5. Particularmente, incrementar k es menos beneficioso para los grafos BA que para los otros: su dificultad no se decrementa tan rápido como otros modelos, inclusive para grafos BA con mayor densidad. Para los modelos RGG, KL y WS tienden a ser más difíciles de resolver para los valores k empleados en este estudio.

Los cuatro algoritmos reaccionan de manera diferente ante los cambios en el parámetro k: PartialCol y React-PartialCol tiene un comportamiento mientras que los algoritmos TabuCol y React-Tabucol presentan otros, por ejemplo se observa que para estos dos últimos algoritmos es más fácil resolver las instancias generadas con el modelo ER, así también las instancias KL son aún más difíciles de resolver por los algoritmos TabuCol y React-Tabucol.

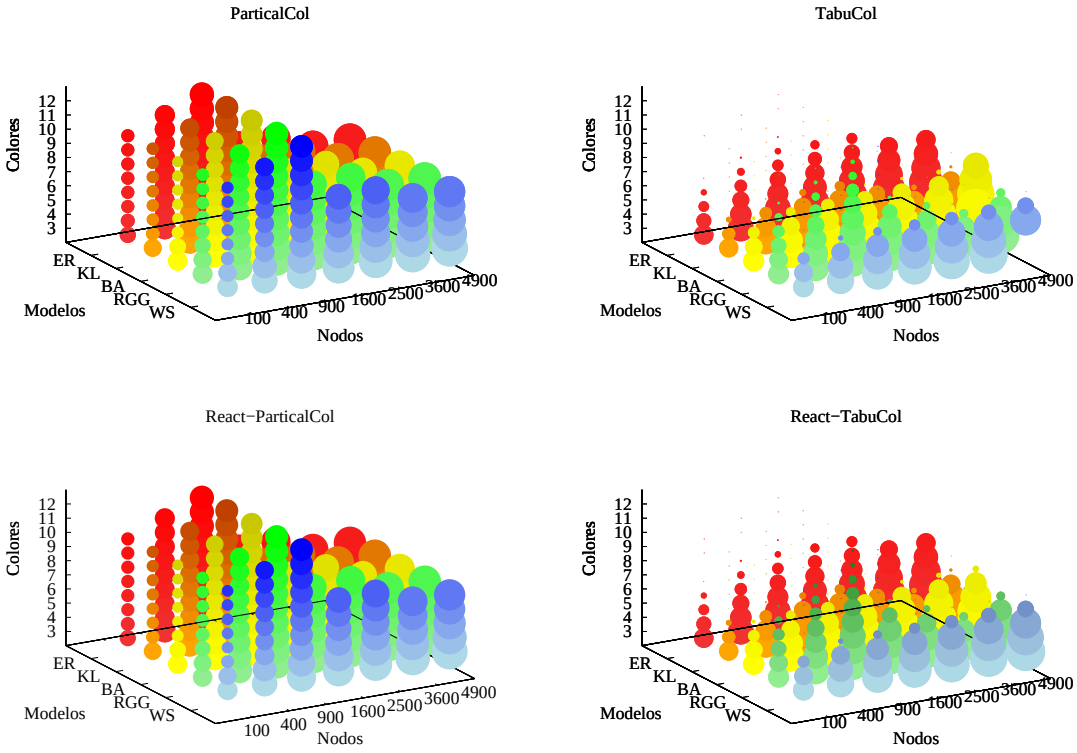


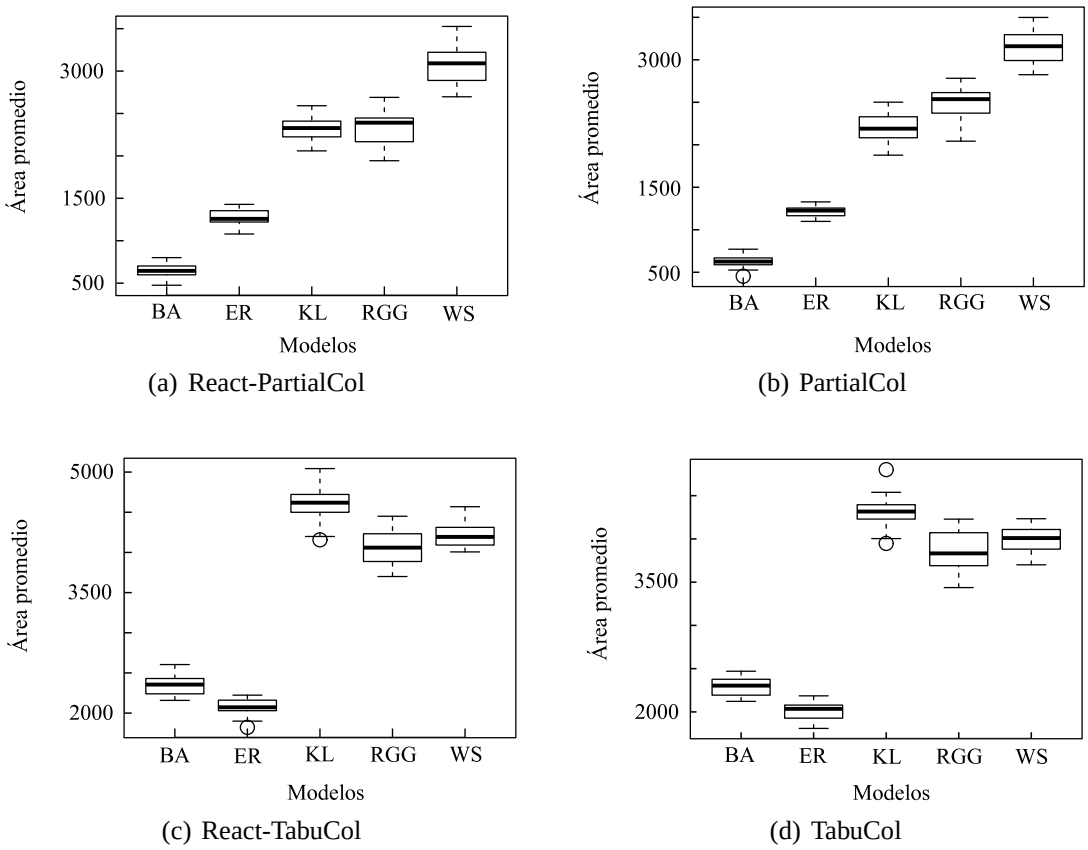
Figura 6.4: *Área bajo el perfil de desempeño para grafos con* $\mathsf{den}(\mathcal{G}) < 0.06$*. En el eje* x *se representa los grafos generados con cierto modelo, en el eje* y *se representan diferentes ordenes de grafos, en el eje* z *representa los colores con los cuales fueron coloreados los grafos, en cada coordenada* (x, y, z) *se gráfica un punto cuyo diámetro es logarítmicamente proporcional al área promediada sobre las instancias generadas con el modelo* x *de densidad* y *y coloreadas con* $k = z$

6.5.3 RESULTADOS ESTADÍSTICOS

Para confirmar las conclusiones obtenidas a través del análisis realizado sobre las gráficas obtenidas para la medida de desempeño propuesta, se realizaron pruebas estadísticas en los datos obtenidos. Usamos el modelo de los efectos:

$$y_{ij} = \mu + \tau_i + \epsilon_{ij}, \tag{6.2}$$

donde i identifica la instancia y j el modelo con la cual fue generada, y_{ij} es el área bajo el perfil de desempeño, μ es el promedio global, τ_i es el efecto de la estructura del grafo — identificado por cada modelo de generación — y ϵ_{ij} es un error aleatorio; donde $N = 30$ réplicas por observación. Se estudia la hipótesis que el efecto de la estructura del grafo es nula, es decir, que el desempeño algorítmico fuese igual para todos los grafos independientemente del modelo de generación usado. Para aceptar o rechar la hipótesis se



(a) React-PartialCol (b) PartialCol

(c) React-TabuCol (d) TabuCol

Figura 6.5: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 100 *vértices con densidad* $den(\mathcal{G}) < 0.06$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

realizó una prueba ANOVA por cada algoritmo;

$$\begin{aligned} H_0 :&\ \ \tau_1 = \tau_2 = \cdots = \tau_a = 0, \\ H_1 :&\ \ \text{al menos una } \tau_i \neq 0, \end{aligned} \tag{6.3}$$

donde $a = 5$. El criterio de rechazo se fijó como $F_0 > F_{\alpha,\, a-1,\, N-a}$, con un nivel de significancia de $\alpha = 0.05$ y una potencia de 0.916.

El análisis de la varianza para grafos con 100 nodos con $den(\mathcal{G}) < 0.06$ y $k = 3$ es mostrada en la Tabla 6.3. Para cada algoritmo se tiene que $F_{0.05,\, 4,\, 25} = 2.76$, con criterio de rechazo $F_0 > F_{0.05,\, 4,\, 25}$, la hipótesis nula (el efecto de la estructura no introduce variabilidad, veasé Ecuación 6.3) es rechazada y la hipótesis alternativa (al menos una estructura de los grafos introduce variabilidad en la variable respuesta) es aceptada. Esto permite concluir que hay un efecto significativo en el desempeño algorítmico causado por

Tabla 6.3: *Resultados de las pruebas ANOVA para las instancias de grafos con* 100 *nodos y* $\mathsf{den}(\mathcal{G}) \approx 0.059$

Algoritmo	F_0
PartialCol	1,340.99
React-PartialCol	1,146.09
TabuCol	1,307.60
React-TabuCol	1,647.60

la estructura de la instancia, independientemente del algoritmo utilizado en el experimento.

Para validar esta conclusión se verificaron que se cumplieran los supuestos de normalidad a través de las gráficas de los residuales resultando NID$(0,\, \sigma^2)$ y sin patrones, como es requerido para que el análisis de la varianza sea válido; un ejemplo de estas gráficas se muestra en la Figura 6.6.

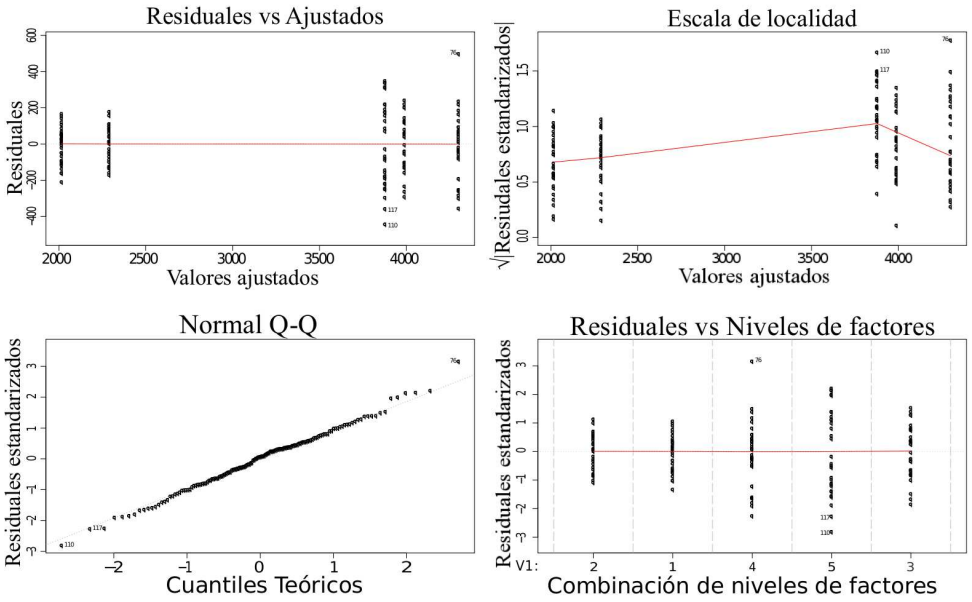


Figura 6.6: *Gráficas de los residuales para comprobar los supuestos de normalidad para validar el análisis de la varianza.*

Para los cuatro algoritmos, se encontró — a través de los resultados de la prueba ANOVA y pruebas de Tukey — que las instancias de grafos con 100 nodos generadas con los modelos KL, WS, y RGG son más difíciles de resolver que aquellos generados con los modelos BA y ER, para mayor claridad observe la Figura 6.5.

Este mismo análisis se realizo para los demás grafos de baja densidad — información de los grafos en la Tabla 6.2 — coloreados con $k = 3$ colores, los resultados se

muestran en la Tabla 6.4. Para cada algoritmo ejecutado sobre el conjunto de grafos de baja densidad se tiene que $F_{0.05,\,4,\,25} = 2.76$, con criterio de rechazo $F_0 > F_{0.05,\,4,\,25}$, por lo tanto, como en el caso de los grafos de 100 nodos de baja densidad, la hipotesis alternativa es aceptada. Lo que permite concluir nuevamente que hay un efecto significativo en el desempeño algorítmico causado por la estructura de la instancia, independientemente del algoritmo utilizado en el experimento.

Tabla 6.4: *Resultados de las pruebas ANOVA para las instancias de grafos con baja densidad.*

Algoritmo	400 F_0	900 F_0	1600 F_0	2500 F_0	3600 F_0	4900 F_0
PartialCol	1645	1412	1386	1919	2511	1517
React-PartialCol	1876	2355	1641	3872	3378	3246
TabuCol	3213	4088	4891	18524	44902	26815
React-TabuCol	2912	5671	7700	30263	76313	31356

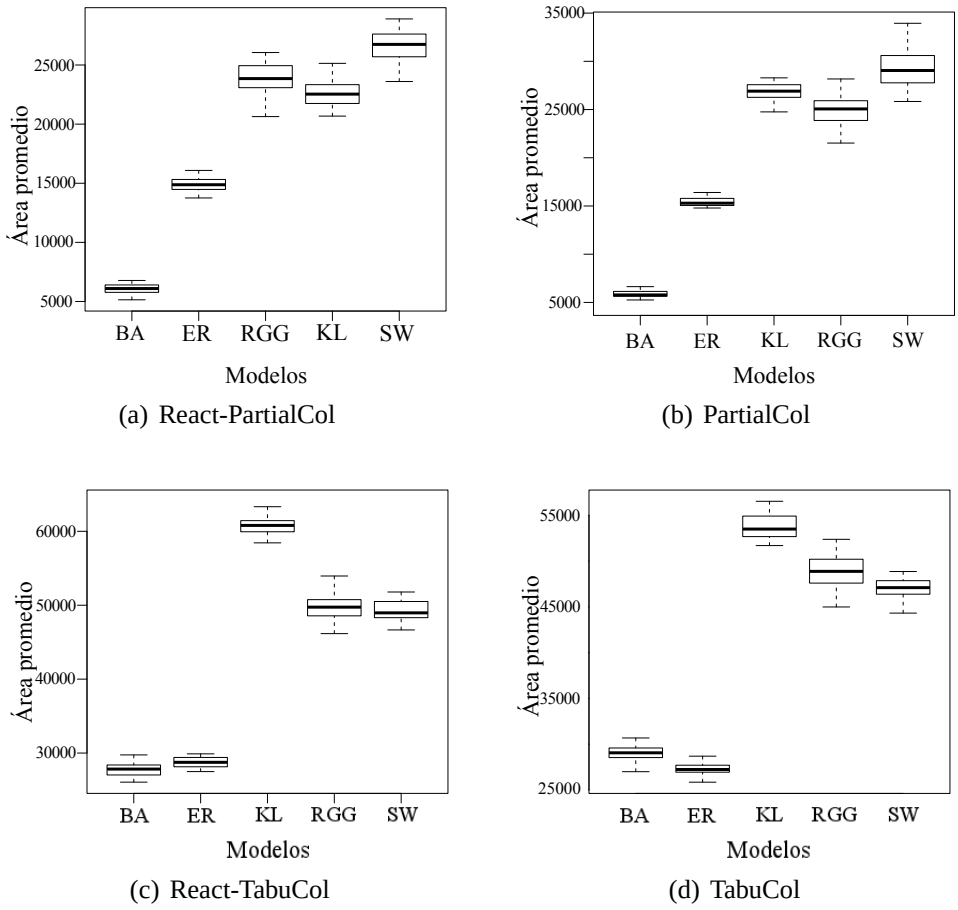
Para las instancias de grafos con 400 y 900 se tiene el mismo comportamiento que las instancias de 100, observe las Figuras 6.7 y 6.8 lo modelos KL, WS, y RGG son más difíciles de resolver que aquellos generados con los modelos BA y ER. Cuando se comienza a incrementar el orden de los grafos se observa en las instancias con 1600, 2500, 3600, 4900 nodos — Figuras 6.9, 6.10, 6.11 y 6.12 — los grafos generados con el modelo KL se incrementa se aprecia que para densidades bajas estas instancias son fáciles de resolver, conservandose el mismo comportamiento para las demás instancias, es decir, los modelos KL y WS son más difíciles de resolver que aquellos generados con los modelos BA y ER.

Tabla 6.5: *Resultados de las pruebas ANOVA para las instancias de grafos con* 64 *nodos, se señala la densidad y el numero de colores empleado (den$(\mathcal{G})$, k).*

Algoritmo	(0.2, 3) F_0	(0.4, 5) F_0	(0.6, 8) F_0	(0.8, 15) F_0
PartialCol	485.40	434.60	154.70	114.12
React-PartialCol	647.40	554.60	161.80	136.91
TabuCol	406.70	214.30	178.40	133.60
React-TabuCol	392.40	179.30	151.18	137.00

Este tipo de análisis fue llevado a cabo para todas las instancias de mediana y alta densidad, ver Tabla 6.1, usando valores de k que caen dentro del régimen difícil del problema. Cuando la densidad se incrementa, los valores del estadístico F_0 se decrementan, indicando que las diferencias en el comportamiento del algoritmo es menos evidente, como se muestra en la Tabla 6.5. Esto es debido a que los grafos con alta densidad comparten

propiedades estructurales similares por ser casi completos, esto hace que independientemente del modelo usado para generar los grafos estos sean similares y por tanto el efecto de la estructura de las instancias no pueden ser diferenciadas, tal como se discutio anteriormente (Figura 6.3).



(a) React-PartialCol (b) PartialCol

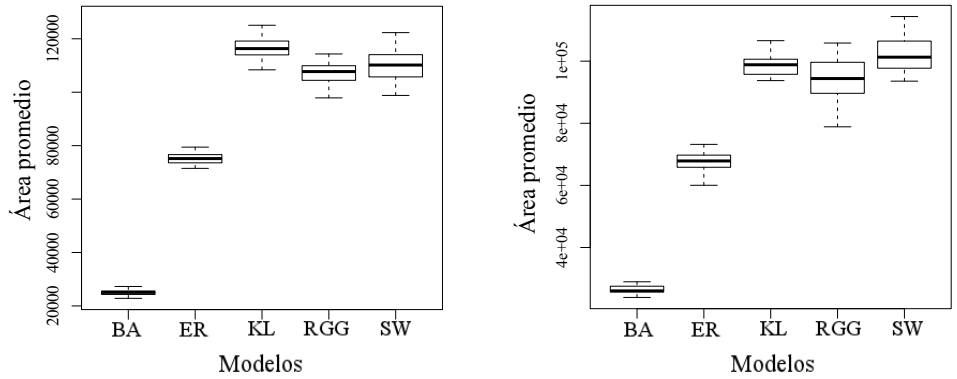
(c) React-TabuCol (d) TabuCol

Figura 6.7: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 400 *vértices con densidad* $den(\mathcal{G}) = 0.015$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

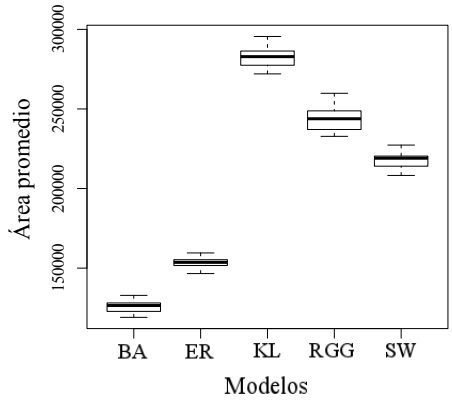
6.5.4 CLASIFICACIÓN DE INSTANCIAS

Ahora lo importante es determinar qué propiedades estructurales permiten diferenciar entre instancias fáciles y difíciles de resolver. Debido a que se controló el tamaño y orden de las instancias generadas, medidas estructurales simples como la densidad $den(\mathcal{G})$ y grado promedio $\langle k \rangle$ son necesarimente similares independientemente del modelo de generación de instancias usado.

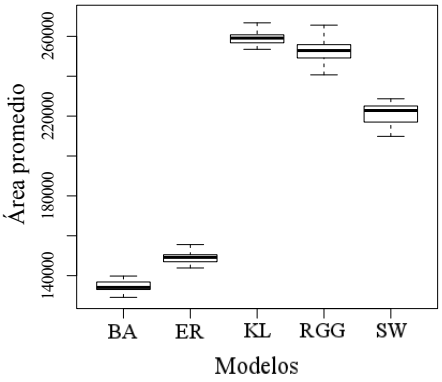


(a) React-PartialCol

(b) PartialCol



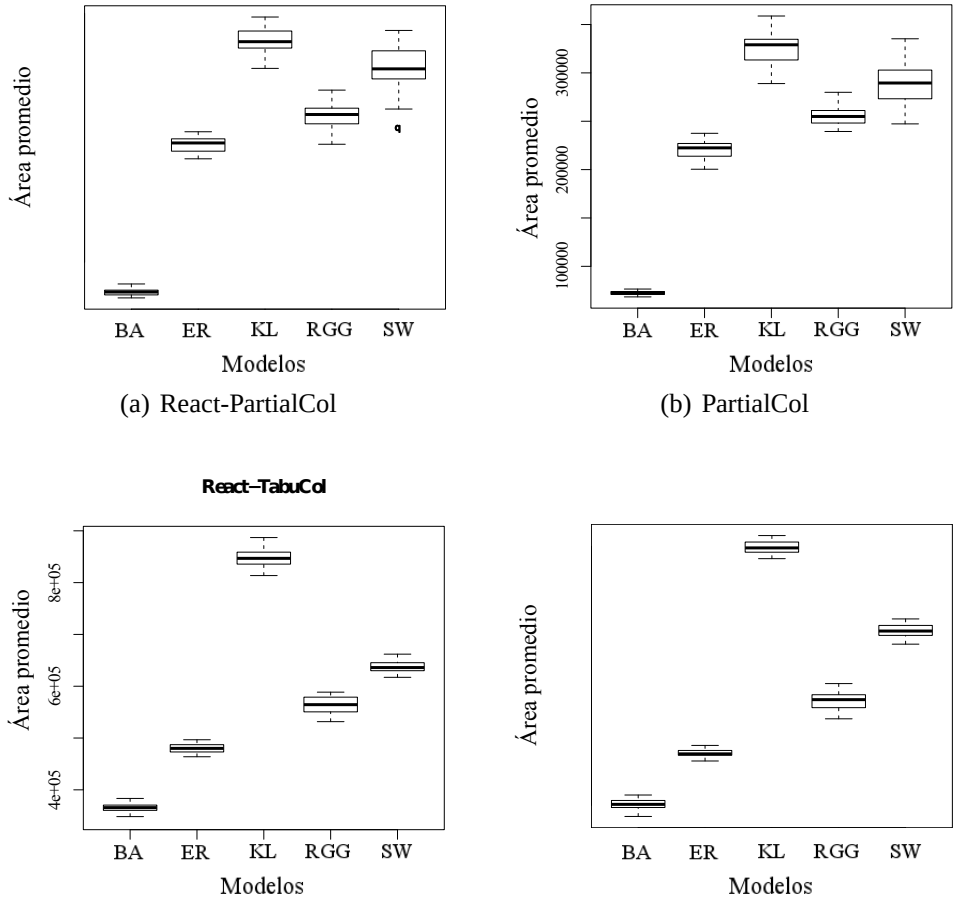
(c) React-TabuCol



(d) TabuCol

Figura 6.8: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 900 *vértices con densidad* $den(\mathcal{G}) = 0.007$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

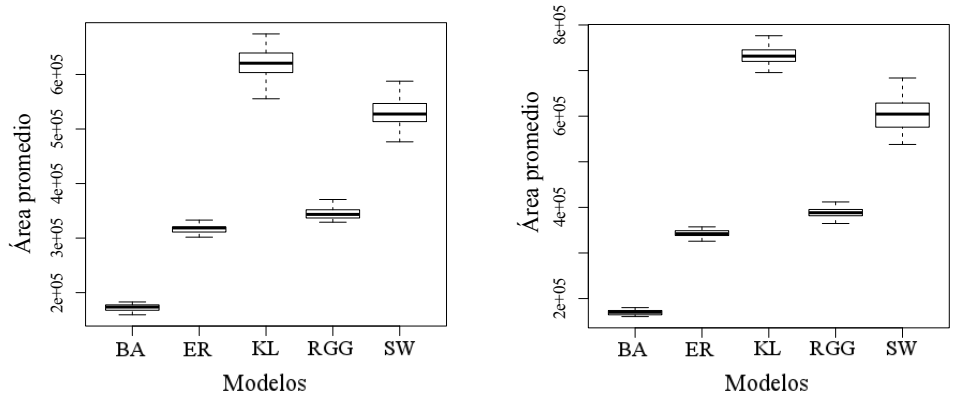


(a) React-PartialCol (b) PartialCol

(c) React-TabuCol (d) TabuCol

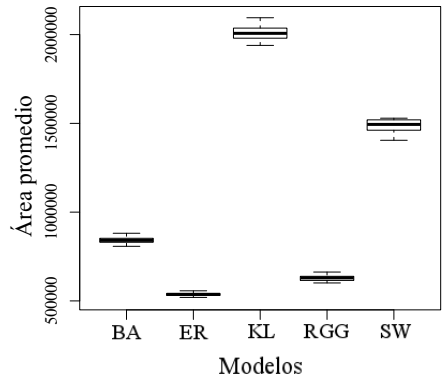
Figura 6.9: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 1600 *vértices con densidad* $\mathsf{den}(\mathcal{G}) = 0.004$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

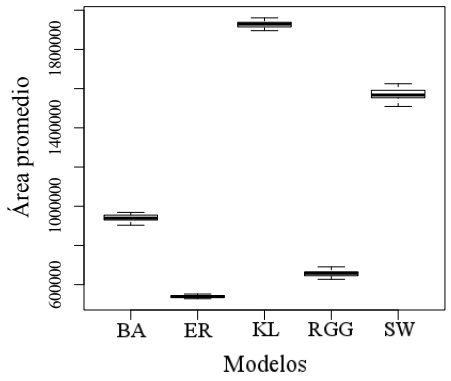


(a) React-PartialCol

(b) PartialCol



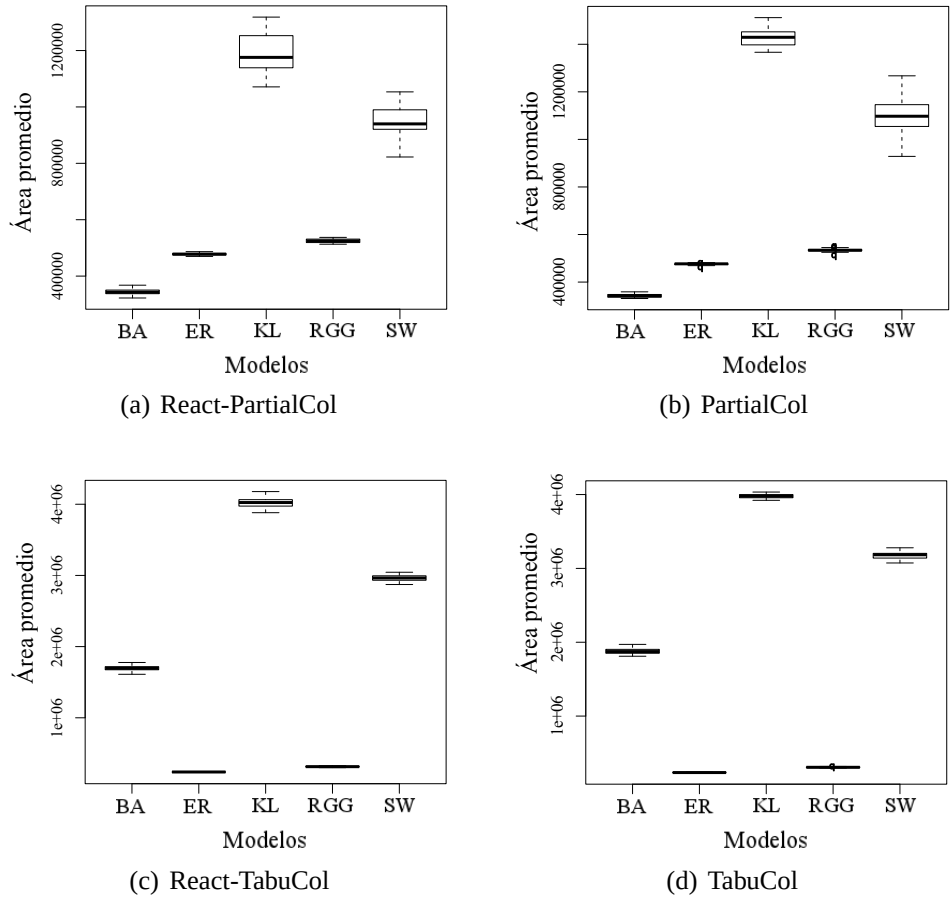
(c) React-TabuCol



(d) TabuCol

Figura 6.10: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 2500 *vértices con densidad* $den(\mathcal{G}) = 0.002$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

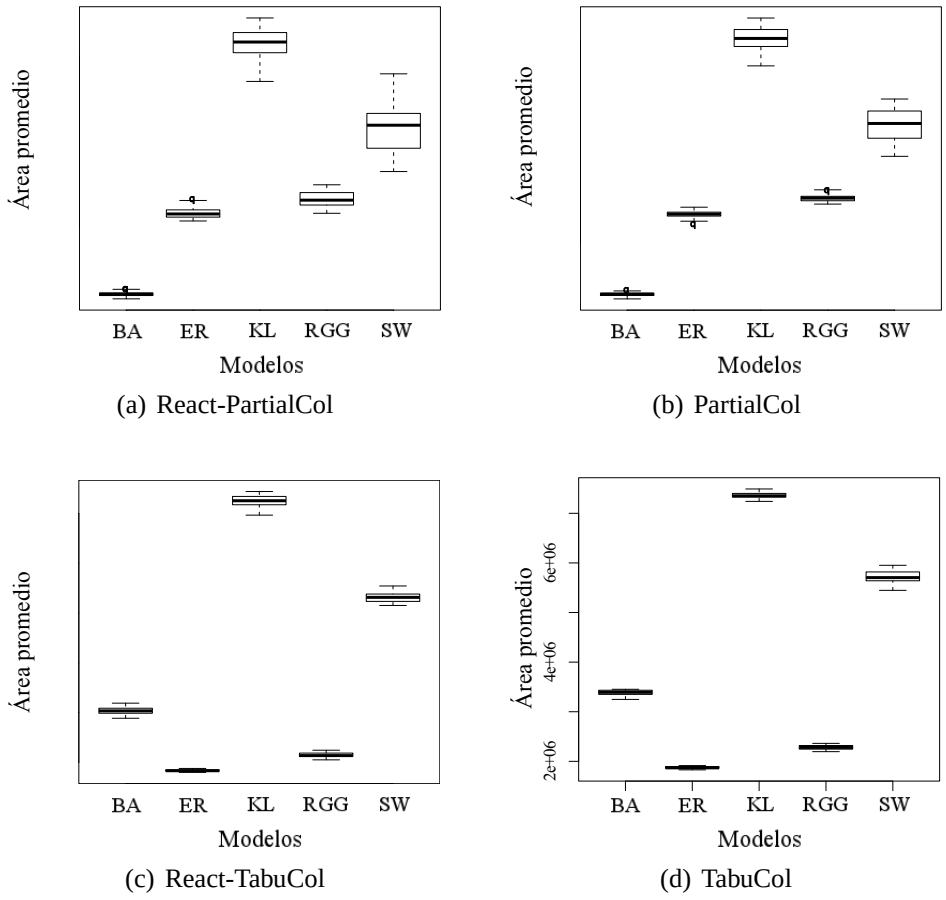


(a) React-PartialCol (b) PartialCol

(c) React-TabuCol (d) TabuCol

Figura 6.11: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 3600 *vértices con densidad* $\mathsf{den}(\mathcal{G}) = 0.001$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.



(a) React-PartialCol (b) PartialCol

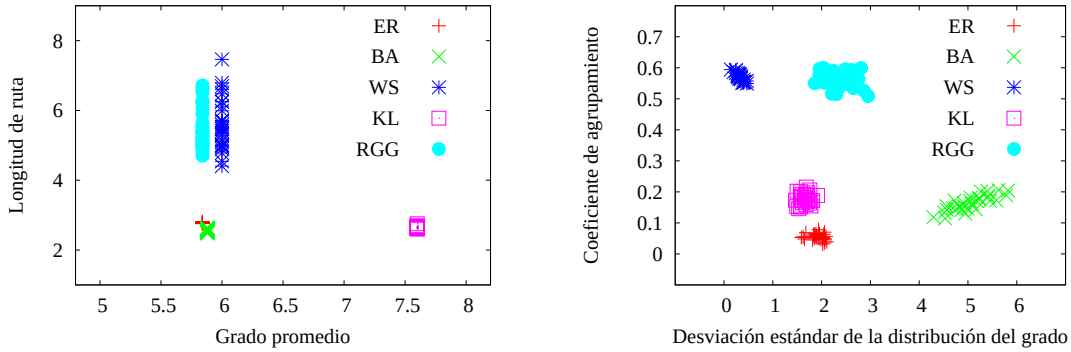
(c) React-TabuCol (d) TabuCol

Figura 6.12: *Área bajo el perfil de desempeño para los cuatro algoritmos para grafos con* 4900 *vértices con densidad* $\mathsf{den}(\mathcal{G}) < 0.001$ *sobre* 30 *replicas. En el eje* x *representa los grafos generados con diferente modelo y el eje* y *representa el área bajo el perfil de desempeño*

.

Para identificar esas propiedades estructurales que permitan diferenciar entre instancias fáciles y difíciles, se utilizó un algoritmo de agrupamiento llamado Xmeans [Pelleg y Moore, 2000], el cual está implementado en Weka [Hall *et al.*, 2009], usando los datos obtenidos en la caracterización de instancias, los cuales fueron descritos en la Sección 6.2; para cada instancia se obtiene un vector con las propiedades estructurales.

Lo primero que se corroboró fue que los grafos, agrupados de acuerdo a sus propiedades, lo hacen de acuerdo al modelo con el cual se generaron y además estas propiedades — observese con atención la Figura 6.13(b) — hacen que las instancias de grafos difíciles de resolver esten más separadas que las instancias de grafos que resultaron fáciles de resolver. Esto se cumplió para todos los grafos — los de baja densidad y los de alta densidad — cuando se utilizaron todas las propiedades estructurales, a excepción de la densidad, tamaño y orden, lo cual se explica al inicio de esta subsección. En la Figura 6.13 se muestra dos proyecciones de estos grupos como ejemplo, Weka genera una matriz de proyecciones para cada par de propiedades de las instancias e identifica mediante colores en que grupo quedo cada instancia.



(a) No se pueden identificar grupos de grafos por cada modelo de generación usado.

(b) Por cada modelo de generación usado se identifica un grupo.

Figura 6.13: *Resultados del agrupamiento realizado con Xmeans para instancias de grafos con* 100 *nodos. A la izquierda, se muestra una proyección de los grupos usando las propiedades de grado promedio y longitud de ruta, en la cual no se aprecia una separación clara de las instancias de acuerdo al modelo usado. A la derecha, se muestra una proyección de los grupos usando las propiedades de coeficiente de agrupamiento del grafo y desviación estándar de la distribución del grado, siendo estas propiedades las que permiten al algoritmo Xmeans distinguir entre instancias generadas con diferentes modelos y también permite distinguir como las instancias fáciles y díficiles se ubican en regiones diferentes de la proyección.*

Con el análisis por agrupamiento, también se pudo apreciar en la Figura 6.13(b) cuáles propiedades estructurales permiten distinguir entre instancias que resultaron fáciles y difíciles de resolver. En la Subsección 6.5.3 se concluyó que las instancias generadas con los modelos RGG y WS son díficiles de resolver; estas instancias tienen un alto valor en el coeficiente de agrupamiento. Se puede observar en la Figura 6.13(b) que se separan de las demás instancias generadas con los modelos BA, ER, y KL. También se concluyó que las instancias que fueron fáciles de resolver fueron las generadas con los modelos BA y ER, que como se puede observar nuevamente en la Figura 6.13(b): tienen valores pequeños de coeficiente de agrupamiento del grafo aunque con una diferencia importante en cuanto a la desviación estándar de la distribución del grado que en los grafos generados con el modelo BA tiende a ser mayor, esto por la presencia de pocos nodos con un alto grado y muchos nodos con un grado pequeño. Se puede distinguir como los grafos dificiles se ubican en la parte superior de la gráfica y las instancias difíciles en la parte inferior.

Para corroborar estas observaciones, se realizo el análisis por agrupamiento para los demás grafos de baja densidad, en la Figura 6.14 se puede observar que el coeficiente de agrupamiento de los grafos y la desviación estándar de la distribución del grado distinguiendo instancias fáciles e instancias dificiles, a mayor coeficiente de agrupamiento del grafo más difícil es de resolver esa instancia, el efecto del incremento del orden del grafo muestra dos fenomenos interesantes, para las instancias BA la desviación estándar aumenta, a pesar de ello el coeficiente de agrupamiento del grafo permanece bajo, lo que las mantiene dentro del régimen fácil. Por otra parte para instancias RGG el coeficiente de agrupamiento del grafo disminuye lo que hace que vaya del régimen difícil al fácil. Esto indica que el *coeficiente de agrupamiento del grafo* [2] *y la desviación estándar de la distribución de grado* son para el problema de coloreo de grafos propiedades estructurales relevantes que influyen en la dificultad de solución.

Un caso interesante es el de los grafos generados con el modelo KL, estos grafos son instancias díficiles para el problema de coloreo de grafos no obstante el coeficiente de agrupamiento no es un valor alto como en las instancias RGG y WS, sino más bien valores de coeficiente de agrupamiento similares a los grafos BA y ER; sin embargo, la desviación estándar de la distribución de grado es baja en estos grafos y prácticamente al incrementar el orden de los grafos se situan en la misma región en cada una de las proyecciones mostradas en 6.14. Un aspecto que vale la pena recordar de las instancias que permanecen como difíciles para cualquier orden del grafo, es la estructura apartir de

[2] La definición de esta propiedad y la ecuación para calcularla se encuentra en la Sección 2.2.1

los cuales se generan las instancias KL y WS, una malla y un anillo respectivamente, los cuales pueden verse como grafos regulares.

6.5.5 REGRESIÓN LINEAL MULTIPLE

Para confirmar las observaciones realizadas en la subsección anterior acerca de las propiedades estructurales que influyen en la dificultad de solución y tener una idea cuantitativa de como estas propiedades influyen, se realizó una regresión lineal múltiple con la siguiente ecuación:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 + \epsilon, \tag{6.4}$$

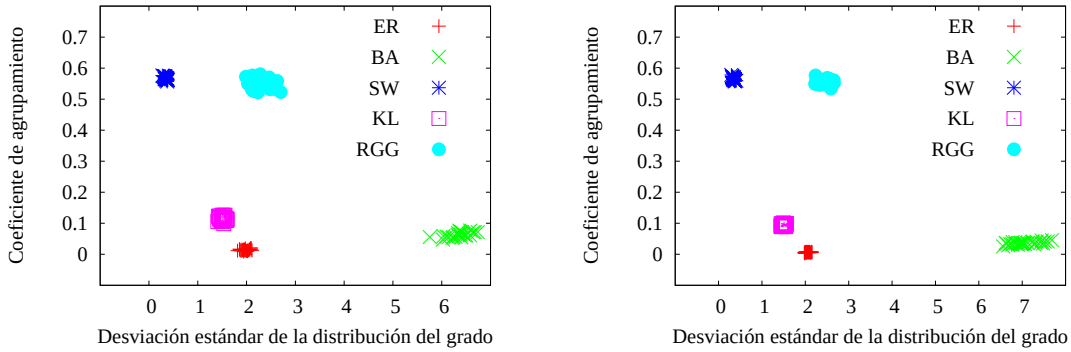
donde la variable dependiente Y es el área bajo la curva (en escala logarítimica) y las variables independientes son la desviación estándar de la distribución del grado X_1 y el coeficiente de agrupamiento X_2 de las instancias de 100 nodos generadas con los diferentes modelos. Los coeficientes de regresión β_1 y β_2 miden las intensidad del efecto de las propiedades sobre la dificultad de solución.

Tabla 6.6: *Resultados de la regresión lineal múltiple.*

Algoritmo	β_1	β_2	β_3	R^2
PartialCol	-0.2619	0.9437	7.7359	0.880
React-PartialCol	-0.2589	1.1470	7.6559	0.897
TabuCol	-0.0644	0.7959	7.9549	0.538
React-TabuCol	-0.0720	0.8012	8.0158	0.540

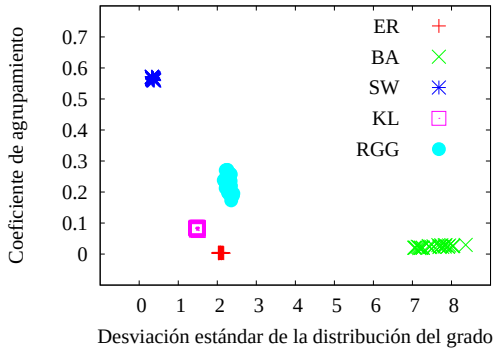
En la Tabla 6.6 se muestran los resultados obtenidos para los coeficientes β_1 y β_2 que indican el incremento en la dificultad de solución por el incremento de los valores de las propiedades estructurales, siendo el coeficiente de agrupamiento el que más influye en la dificultad de solución.

Para el caso de los algoritmos PartialCol y React-PartialCol el estadístico R^2 — mostrado en la Tabla 6.6 — indica que el modelo de regresión usado explica cerca del $88\,\%$ de la variabilidad de la dificultad de solución usando el coeficiente de agrupamiento y la desviación estándar de la distribución del grado como variables de regresion. Para los algoritmos TabuCol y React-TabuCol el modelo usado explica cerca del $54\,\%$ de la variabilidad de la dificultad de solución. En la Figura 6.15 se muestran los planos de regresión asociados a este análisis.

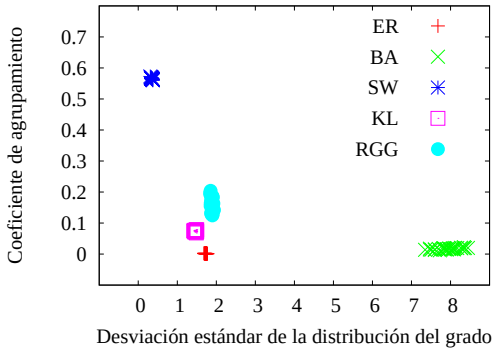


(a) Grafos con 400 nodos.

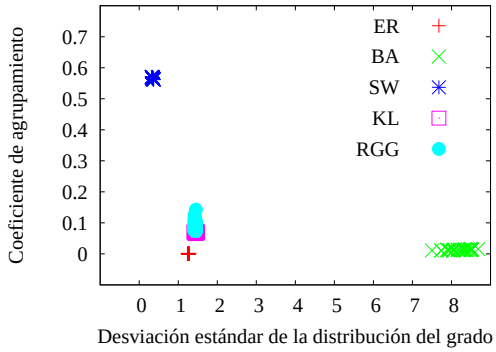
(b) Grafos con 900 nodos.



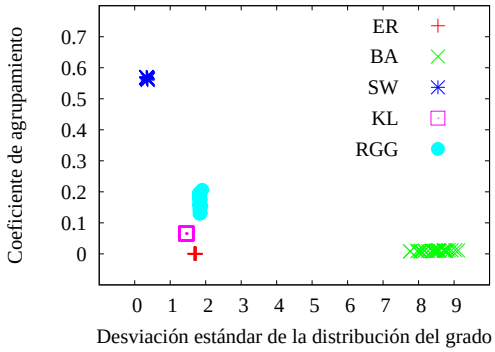
(c) Grafos con 1600 nodos.



(d) Grafos con 2500 nodos.

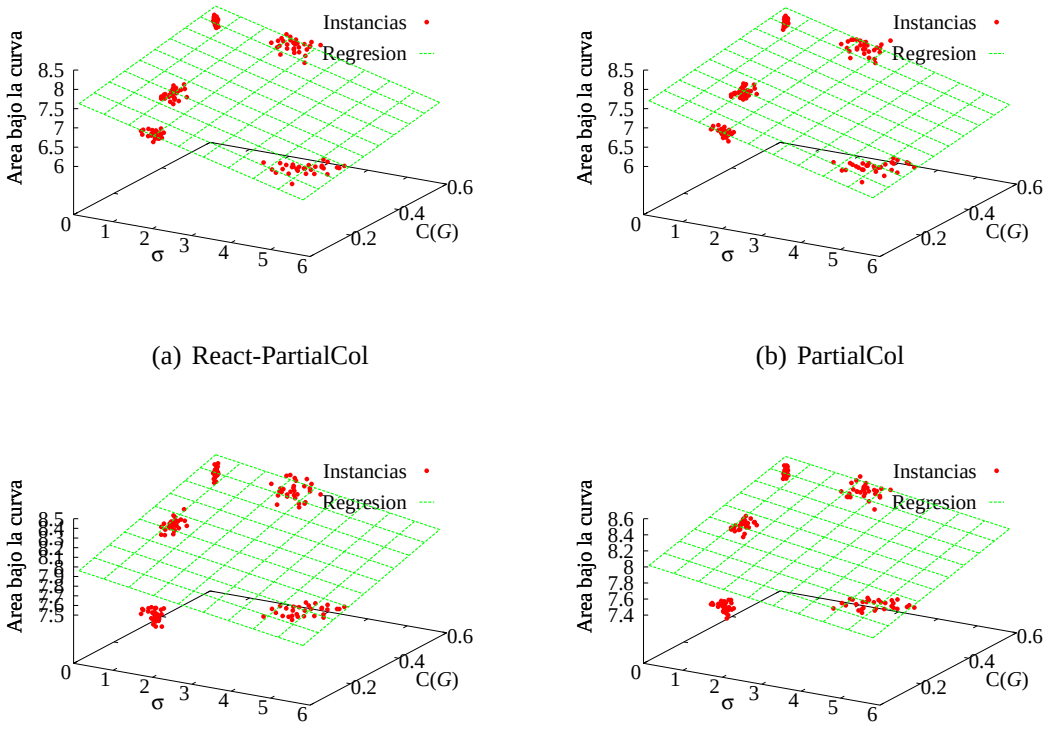


(e) Grafos con 3600 nodos.



(f) Grafos con 4900 nodos.

Figura 6.14: *Resultados del agrupamiento realizado con Xmeans para instancias de grafos con* 400*,* 900*,* 1*,* 600*,* 2500*,* 3600 *y* 4900 *nodos. Para cada conjunto de grafos de determinado orden se muestra una proyección de los grupos usando las propiedades de coeficiente de agrupamiento del grafo y desviación estándar de la distribución del grado, siendo estas propiedades las que permiten al algoritmo Xmeans distinguir entre instancias generadas con diferentes modelos y también permite distinguir como las instancias fáciles y díficiles se ubican en regiones diferentes de la proyección.*



(a) React-PartialCol

(b) PartialCol

(c) React-TabuCol

(d) TabuCol

Figura 6.15: *Planos de la regresión lineal efectuadas para explicar la dificultad de solución de las instancias con respecto a las propiedades de coeficiente de agrupamiento del grafo y desviación estándar de la distribución del grado.*

Cabe señalar que el objetivo no es establecer la causa de que las instancias sean difíciles o fáciles de resolver, ya que existen diversos factores dominantes además de las propiedades estructurales, como lo es el tamaño y orden del grafo — en general el tamaño de la instancia de un problema — y parámetros propios del problema; el objetivo es explicar de que manera ciertas propiedades estructurales afectan el desempeño de los algoritmos a pesar de ser iguales en términos del tamaño de la instancia.

CAPÍTULO 7

CONCLUSIONES Y TRABAJO FUTURO

Las verdades que revela la ciencia
superan siempre a los sueños que destruye.
- Joseph Ernest Renan.

El problema abordado en este trabajo de investigación es la *identificación de propiedades estructurales en instancias de un problema — de igual tamaño — que afectan el desempeño de los algoritmos* usados para resolver el problema y la determinación de cómo ciertos valores de propiedades estructurales hacen que las instancias sean más fáciles o más difíciles de resolver, en otras palabras, poder *caracterizar la dificultad de una instancia en términos más descriptivos que su tamaño*. La principal hipótesis es que además del tamaño de la instancia y los parámetros del problema a resolver, ***la complejidad de resolución de un problema dado una instancia también es afectada por la estructura de lo misma***.

Para resolver el problema planteado y probar la hipótesis, en este trabajo de tesis *se propusó un marco de trabajo* para estudiar el efecto de la estructura de la instancia en la complejidad de solución de un problema computacional. El marco de trabajo señala consideraciones a tomar en cuenta cuando se aborda el problema planteado, desde la selección del problema computacional, las características de la instancias, la selección de algoritmos para realizar el estudio, las medidas de desempeño algorítmico a considerar, hasta recomendaciones para establecer relaciones entre la dificultad de solución y propiedades estructurales, de manera que pueda aplicarse a diversos problemas computacionales.

Estas condiseraciones fueron tomadas en cuenta al *aplicar este marco de trabajo al problema de coloreo de grafos*, el cual es representativo de muchas clases de problemas de grafos interesantes y optimización combinatoria, diversas áreas de conocimiento tales como la teoría de grafos, teoría de redes complejas, complejidad computacional,

metaheurísticos, sirvieron para dar soporte a cada una de las actividades señaladas en el marco de trabajo.

Para efectuar el estudio del efecto de las propiedades estructurales sobre el desempeño algorítmico, se identificó la necesidad de contar con una medida de desempeño que fuera independiente de ciertos factores como las caracterísiticas del equipo en el cual se llevan a cabo experimentaciones, detalles de implementación, algoritmos usados, entre otras. Ante tal situación se *propuso una medida de desempeño algorítmico* basada en la manera en que los algoritmos convergen hacia una determinada solución.

En el caso de estudio abordado, se generaron grafos con diferente estructura e igual tamaño, siendo una parte importante en este caso la determinación de parámetros de los modelos de generación usados para logar tal objetivo, una vez generado los grafos se procedio a la caracterización mediente propiedades estructurales usadas en la teoría de redes complejas. Se eligieron algoritmos del estado del arte que resuelven el problema de coloreo de grafos, un punto importante es elegir algoritmos que formulen de manera diferente las estrategías de solución; en este paso siempre es importante tener en cuenta que *no se desea comprar entre algoritmos para determinar cual es mejor o peor que otro, lo que se desea comparar es el comportamiento del mismo algoritmo sobre un conjunto de instancias de igual tamaño y diferente estructura*.

Los resultados del desempeño algorítmico sobre diferentes estructuras de grafos obtenidos a partir de la aplicación del marco de trabajo al problema de coloreo, fueron analizados a través de gráficas y pruebas estadísticas e indicaron que la medida de desempeño algortítmico propuesta refleja adecuadamente el efecto de la estructura de la instancia en el desempeño algorítmico en el problema de coloreo de grafos. Además permitió *concluir que los grafos que resultaron más difíciles de resolver por los algoritmos usados para el estudio fueron los grafos generados con los modelos de grafos geométricos aleatorios (RGG), y Watts-Strogatz (WS) los cuales tienen valores de coeficiente alto en comparación con los demás modelos usados resultando los grafos generados con el modelo Erdős - Rényi (ER) y Barabási (BA) más fáciles de resolver*.

Un caso especial es el de los grafos generados con el modelo de Kleinberg (KL) ya que estos resultaron difíciles de resolver, pero el coeficiente de agrupamiento no es tan alto como el de los grafos generados con los modelos RGG y WS. *Se pudo concluir que el coeficiente de agrupamiento y la desviación estándar de la distribución de grafo son propiedades estructurales que afectan la dificultad de solución*. Es de principal interés en

trabajos futuros analizar más detalladamente las propiedades estructurales para los grafos KL.

En una primera aproximación se realizó una regresión lineal múltiple para establecer una relación entre el coeficiente de agrupamiento y la desviación estandar de la distribución del grado con la dificultad de solución de la instancia, siendo el coeficiente de agrupamiento la propiedad que más influye en la dificultad asociada a la instancia. *Es de interés dentro del trabajo futuro, verificar rigurosamente esta relación tomando en cuenta otro modelo diferente al lineal*.

Se puede ***concluir*** que el marco de trabajo propuesto es efectivo y práctico para caracterizar la dificultad inherente a la estructura de instancias iguales en tamaño pero distintas en términos estructurales, dadas las observaciones puntualizadas anteriormente; también se puede concluir que la medida de desempeño es fácil de calcular, fácil de incorporar en un algorítmo e intuitiva de interpretar que puede ser empleada cuando se buscan diferencias en el comportamiento algorítmico sobre un conjunto de instancias con diferente estructura e igual tamaño. Esta medida se basa en la manera en que los algoritmos convergen hacia una determinada solución.

Resultados de esta investigación fueron presentados como ponencia en ALIO - INFORMS Joint International Meeting 2010, en Buenos Aires, Argentina, bajo el título *Structural Complexity in Complex Networks*, además se envio el artículo titulado *Structural effects in algorithm performance: A framework and a case study on graph coloring* al Journal of ACM en la categoría de análisis de algoritmos y complejidad del problema.

Dentro del trabajo futuro es de interés estudiar la distribución del coeficiente de agrupamiento de los grafos en el conjunto de instancias identificadas como difíciles, ya que algunas son más dificiles que otras. El objetivo práctico es caracterizar la dificultad de una instancia directamente en términos de un rango en el cual las propiedades estructurales caen, en particular usar valores aproximados en lugar de calcular el valor exacto global para grafos de gran tamaño ya que en algunos casos el tiempo de cómputo para calcular el valor exacto de estas propiedades es elevado, siendo el cálculo aproximado de propiedades estructurales otra área de oportunidad para futuros proyectos de investigación.

Una línea de trabajo futuro prometedor es la ***optimización estructural***, donde dado un grafo, un problema computacional y la indicación de si se quiere que la instancia sea fácil o difícil de resolver, gradualmente modificar la estructura del grafo hasta que caiga dentro del régimen deseado de valores aproximados de las propiedades estructurales,

también se puede imponer un *presupuesto* que restringa la cantidad de modificaciones realizadas sobre la estructura.

Dentro del contexto de grafos, la optimización estructural como línea de investigación tiene una importancia práctica en el diseño o rediseño de redes de telecomunicaciones, vialidades, distribución de fluidos (agua, gas, drenaje, drenaje pluvial), rutas de evacuación, entre otras; de manera que los procesos que se lleven a cabo en ellas sean fáciles o difíciles de llevarse a cabo según las necesidades de su entorno y que la construcción de una nueva infraestructura de red o modificación de una infraestructura existente sea a bajo costo, estos dos aspectos: determinar la mejor estructura que satisfaga las necesidades y que sean a bajo costo, pudiesen ser objetivos que se contrapongan, lo que llevaría a trabajar con problemas multiobjetivo. Además es importante extender la aplicación del marco de trabajo a grafos dirigidos y ponderados.

Otra línea de investigación prometedora es la ***caracterización y visualización del espacio de busqueda asociado a un problema***, siguiendo la problematica planteada en este problema, es de interes poder determinar cómo la estructura de una instancia influye en la estructura del espacio de búsqueda, de manera que esta información sea usada en el diseño de estrategias de búsqueda dada una estructura de instancia en particular.

Por otra parte, es de interés incorporar la medida de desempeño propuesta en el diseño de algoritmos hiperheurísticos[1] para ayudar a decidir que estrategia está teniendo un progreso significativo sobre la instancia y si es conveniente reiniciar o no la estrategia.

[1]Un hiperheurístico es un método de búsqueda heurística que tiene como objetvio automatizar — muchas veces incorporando técnicas de inteligencia artificial — en el proceso de selección, combinación, generación y adaptación de heurísticas simples para resolver eficientemente problemas computacionales.

BIBLIOGRAFÍA

ACHLIOPTAS, D., A. NAOR y Y. PERES (2005), «Rigorous location of phase transitions in hard optimization problems», *Nature*, **435**(7043), págs. 759–764.

ALBERT, R. y A.-L. BARABÁSI (2002), «Statistical mechanics of complex networks», *Reviews of Modern Physics*, **74**(1), págs. 47–97.

ALBERT, R., H. JEONG y A.-L. BARABÁSI (2000), «Error and attack tolerance of complex networks», *Nature*, **406**(6794), págs. 378–382.

ALBERT, R. Z. (2001), *Statistical mechanics of complex networks*, Tesis Doctoral, Notre Dame, Indiana, Estados Unidos.

ALDERSON, D. L. (2008), «Catching the "network science" bug: insight and opportunity for the operations researcher», *Operations Research*, **56**(5), págs. 1047–1065.

AMARAL, L. A. N., P. GOPIKRISHNAN, M. KAUSHIK, P. VASILIKI y H. E. STANLEY (2001), «Application of statistical physics methods and concepts to the study of Science and technology systems», *Scientometrics*, **51**(1), págs. 9–36.

AMARAL, L. A. N. y J. M. OTTINO (2004), «Complex systems and networks: challenges and opportunities for chemical and biological engineers», *Chemical Engineering Science*, **59**(8-9), págs. 1653–1666.

AMARAL, L. A. N., A. SCALA, M. BARTHÉLÉMY y H. E. STANLEY (2000), «Classes of small-world networks», *Proceedings of the National Academy of Sciences*, **97**(21), págs. 11 149–11 152.

AVIN, C. (2006), *Random geometric graphs: an algorithmic perspective*, Tesis Doctoral, University of California, Los Ángeles, California, Estados Unidos.

BALUJA, S. y S. DAVIES (1997), «Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space», en *Proceedings of the 14th International Conference on Machine Learning*, Morgan Kaufmann, págs. 30–38.

BARABÁSI, A.-L. (2005), «Emergence of scaling in complex networks», en *Handbook of Graphs and Networks*, Wiley-VCH, Weinheim, Alemania, págs. 69–84.

BARABÁSI, A.-L. (2007), «The architecture of complexity», *Control Systems*, **27**(4), págs. 33–42.

BARABÁSI, A.-L. y R. ALBERT (1999), «Emergence of scaling in random networks», *Science*, **286**(5439), págs. 509–512.

BARR, R., B. GOLDEN, J. KELLY, M. RESENDE y W. STEWART (1995), «Designing and reporting on computational experiments with heuristic methods», *Journal of Heuristics*, **1**, págs. 9–32.

BAUR, M., U. BRANDES, J. LERNER y D. WAGNER (2009), *Group-level analysis and visualization of social networks*, Springer-Verlag Berlin, Heidelberg, Alemania, págs. 330–358.

BEYER, H.-G. y H.-P. SCHWEFEL (2002), «Evolution strategies: A comprehensive introduction», *Jornal Natural Computing*, **1**, págs. 3–52.

BIRATTARI, M. y M. DORIGO (2007), «How to assess and report the performance of a stochastic algorithm on a benchmark problem: *mean* or *best* result on a number of runs?», *Optimization Letters*, **1**(3), págs. 309–311.

BLÖCHLIGER, I. y N. ZUFFEREY (2008), «A graph coloring heuristic using partial solutions and a reactive tabu scheme», *Computers & Operation Research*, **35**(3), págs. 960–975.

BLUM, C. y A. ROLI (2003), «Metaheuristics in combinatorial optimization: Overview and conceptual comparison», *ACM Computing Surveys*, **35**, págs. 268–308.

BOLLOBÁS, B. (2001), *Random graphs*, Cambridge University Press, Cambridge, Reino Unido.

BOLLOBÁS, B. y O. RIORDAN (2002), «Mathematical results on scale-free random graphs», en *Handbook of graphs and networks*, Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Alemania, págs. 1–37.

BOUZIRI, H., K. MELLOULI y E.-G. TALBI (2011), «The *k*-coloring fitness landscape», *Journal of Combinatorial Optimization*, **21**, págs. 306–329.

BRÉLAZ, D. (1979), «New methods to color the vertices of a graph», *Communications of the ACM*, **22**, págs. 251–256.

CHARTRAND, G. y P. ZHANG (2008), *Chromatic graph theory*, Chapman & Hall/CRC, Boca Raton, Florida, Estados Unidos.

CHEESEMAN, P., B. KANEFSKY y W. M. TAYLOR (1991), «Where the really hard problems are», en *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, tomo 1, Morgan Kaufmann Publishers Incorporated, págs. 331–337.

COCCO, S., R. MONASSON, A. MONTANARI y G. SEMERJIAN (2006), «Analyzing search algorithms with physical methods», en *Computational complexity and statistical physics*, Oxford University Press, Nueva York, Estados Unidos.

COOK, S. A. (1971), «The complexity of theorem-proving procedures», en *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 151–158.

COUDERT, O. (1997), «Exact coloring of real-life graphs is easy», en *Proceedings of the 34th annual Design Automation Conference*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 121–126.

DA F. COSTA, L., F. RODRIGUES, G. TRAVIESO y V. VILLAS BOAS (2007), «Characterization of complex networks: A survey of measurements», *Advances in Physics*, **56**(1), págs. 167 – 242.

DEHMER, M. (2008), «Information processing in complex networks: graph entropy and information functionals», *Applied Mathematics and Computation*, **201**(1-2), págs. 82–94.

DIMACS, C. (2010), «Graph coloring instances, standard format.», Acceso: 08/05/2010, URL `http://mat.gsia.cmu.edu/COLOR/instances.html`.

ERDŐS, P. y A. RENYÍ (1960), «On the evolution of random graphs», *Publications of the Mathematical Institute of the Hungarian Academy Sciences*, **5**, págs. 17–61.

EULER, L. (1741), «Solutio problematis ad geometriam situs pertinentis», *Commentarii academiae scientiarum Petropolitanae*, **8**, págs. 128–140.

FALOUTSOS, M., P. FALOUTSOS y C. FALOUTSOS (1999), «On power-law relationships of the Internet topology», en *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communication*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 251–262.

FEO, T. A. y M. G. C. RESENDE (1995), «Greedy randomized adaptive search procedures», *Journal of Global Optimization*, **6**, págs. 109–133.

FERRANTE, A., G. PANDURANGAN y K. PARK (2008), «On the hardness of optimization in power-law graphs», *Theoretical Computer Science*, **393**, págs. 220–230.

FLEMING, P. J. y J. J. WALLACE (1986), «How not to lie with statistics: the correct way to summarize benchmark results», *Communications of the ACM*, **29**, págs. 218–221.

FLUM, J. y M. GROHE (2004), «The Parameterized Complexity of Counting Problems», *SIAM J. Comput.*, **33**(4), págs. 892–922.

FLUM, J. y M. GROHE (2006), *Parameterized complexity theory*, Springer-Verlag Berlin, Heidelberg, Alemania.

FOGEL, D. B. y L. J. FOGEL (1995), «An introduction to evolutionary programming», en *Artificial Evolution*, págs. 21–33.

FORTNOW, L. (2009), «The status of P versus NP problem», *Communications of the ACM*, **52**, págs. 78–86.

FU, Y. y P. W. ANDERSON (1986), «Application of statistical mechanics to NP-complete problems in combinatorial optimisation», *Journal of Physics A: Mathematical and General*, **19**(9), págs. 1605–1620.

GAREY, M. y D. JOHNSON (1979), *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*, W.H. Freeman and Company, San Francisco, California, Estados Unidos.

GILBERT, E. (1959), «Random graphs», *The Annals of Mathematical Statistics*, **30**, págs. 1141–1144.

GLOVER, F. (1989), «Tabu search - Part I», *INFORMS Journal on Computing*, **1**(3), págs. 190–206.

GOLDBERG, D. E. (1989), *Genetic algorithms in search, optimization and machine learning*, primera edición, Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, Estados Unidos.

GOSLING, J., B. JOY, G. STEELE y G. BRACHA (2005), *The Java language specification*, Addison-Wesley Longman, Amsterdam, Holanda.

GREENBERG, H. J. (1990), «Computational testing: why, how and how much», *INFORMS Journal on Computing*, **2**(1), págs. 94–97.

GUTFREUND, D., R. SHALTIEL y A. TA-SHMA (2007), «If NP Languages are Hard on the Worst-Case, Then it is Easy to Find Their Hard Instances», *Computational Complexity*, **16**(4), págs. 412–441.

HALL, M., E. FRANK, G. HOLMES, B. PFAHRINGER, P. REUTEMANN y I. H. WITTEN (2009), «The WEKA data mining software: an update», *Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter*, **11**, págs. 10–18.

HAMIEZ, J.-P. y J.-K. HAO (2004), «An analysis of solution properties of the graph coloring problem», en M. G. C. Resende, J. P. de Sousa y A. Viana (editores), *Metaheuristics: computer decision-making*, Kluwer Academic Publishers, Norwell, Massachusetts, Estados Unidos, págs. 325–345.

HAYES, B. (2000), «Graph theory in practice, Part II», *American Scientist*, **89**.

HEYLIGHEN, F. (1999), «The growth of structural and functional complexity during evolution», en F. Heylighen, J. Bollen y A. Riegler (editores), *"The evolution of complexity: the violet book of "Einstein meets Magritte"*, VUB University Press, Bruselas, Bélgica, págs. 17–47.

HOLLAND, J. H. (1992), *Adaptation in natural and artificial systems*, MIT Press, Cambridge, Massachusetts, Estados Unidos.

HOLME, P. (2004), *Form and function of complex networks*, Tesis Doctoral, Umea University, Faculty of Science and Technology, Umea, Suecia.

HUANG, K. (2001), *Introduction to statistical physics*, Taylor & Francis, Londres, Reino Unido.

JELASITY, M., B. TÓTH y T. VINKÓ (1999a), «Characterizations of trajectory structure of fitness landscapes based on pairwise transition probabilities of solutions», en *Proceedings of the Congress on Evolutionary Computation*, tomo 1, IEEE Press, Washington D.C., Estados Unidos, págs. 623–630.

JELASITY, M., B. TÓTH y T. VINKÓ (1999b), «Characterizations of Trajectory Structure of Fitness Landscapes Based on Pairwise Transition Probabilities of Solutions», en P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao y A. Zalzala (editores), *Proceedings of the Congress on Evolutionary Computation*, tomo 1, págs. 623–630.

JOHNSON, D. (1996), «A theoretician's guide to the experimental analysis of algorithms», URL `http://www.research.att.com/~dsj/papers/exper.ps`.

JONES, T. (1995), *Evolutionary algorithms, fitness landscapes and search*, Tesis Doctoral, University of New Mexico, New Mexico, Estados Unidas.

JUHOS, I. y J. VAN HEMERT (2008), «Graph colouring heuristics guided by higher order graph properties», en J. van Hemert y C. Cotta (editores), *Evolutionary Computation in Combinatorial Optimization*, tomo 4972, Springer-Verlag Berlin, Heidelberg, Alemania, págs. 97–108.

KAUFMANN, M. y K. ZWEIG (2009), *Modeling and designing real-world networks*, Springer-Verlag Berlin, Heidelberg, Alemania.

KHAJEHZADEH, M., M. R. TAHA, A. EL-SHAFIE y M. ESLAMI (2011), «A survey on metaheuristic global optimization algorithms», *Research Journal of Applied Sciencies, Engineering and Technology*, **3**(6), págs. 569–578.

KICINGER, R., T. ARCISZEWSKI y K. D. JONG (2005), «Evolutionary computation and structural design: A survey of the state-of-the-art», *Computers and Structures*, **83**, págs. 1943–1978.

KIRKPATRICK, S., C. D. GELATT, JR. y M. P. VECCHI (1983), «Optimization by simulated annealing», *Science*, **220**, págs. 671–680.

KIROVSKI, D. y M. POTKONJAK (1998), «Efficient coloring of a large spectrum of graphs», en *Proceedings of the 35th Annual Design Automation Conference*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 427–432.

KLEIN, P. N. y N. E. YOUNG (2010), «Algorithms and theory of computation handbook», capítulo Approximation algorithms for NP-hard optimization problems, Chapman & Hall/CRC, págs. 34–34, URL `http://dl.acm.org/citation.cfm?id=1882757.1882791`.

KLEINBERG, J. M. (2000), «The small-world phenomenon: an algorithm perspective», en *Proceedings of 32nd Annual ACM Symposium on Theory of Computing*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 163–170.

LATORA, V. y M. MARCHIORI (2001), «Efficient behavior of small-world networks», *Physical Review Letters*, **87**(19), págs. 198 701+.

LESKOVEC, J., J. KLEINBERG y C. FALOUTSOS (2005), «Graphs over time: densification laws, shrinking diameters and possible explanations», en *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge discovery in data mining*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 177–187.

LÓPEZ, T. T. (2007), *Clasificación de redes complejas usando funciones de caracterización que permitan discriminar entre redes aleatorias, power-Law y exponenciales*, Tesis de Maestría, Instituto Tecnológico de Ciudad Madero, Cd. Madero, Tamaulipas, México.

MAMMEN, D. L. y T. HOGG (1997), «A new look at the easy-hard-easy pattern of combinatorial search difficulty», *Journal of Artificial Intelligence Research*, **7**, págs. 47–66.

MARTIN, O. C., R. MONASSON y R. ZECCHINA (2001), «Statistical mechanics methods and phase transitions in optimization problems», *Theoretical Computer Science*, **265**(1–2), págs. 3–67.

MARX, D. (2004), «Graph coloring problems and their applications in scheduling», *Periodica Polytechnica Ser El. Eng*, **48**, págs. 11–16.

MARX, D. y I. SCHLOTTER (2010), «Parameterized Complexity and Local Search Approaches for the Stable Marriage Problem with Ties», *Algorithmica*, **58**(1), págs. 170–187.

MARX, D. y I. SCHLOTTER (2011), «Stable assignment with couples: Parameterized complexity and local search», *Discrete Optimization*, **8**(1), págs. 25–40.

MASON, S. J. y N. E. GRAHAM (2002), «Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation», *Quarterly Journal of the Royal Meteorological Society*, **128**(584), págs. 2145–2166.

MCGEOCH, C. C. (1996), «Feature article - Toward an experimental method for algorithm simulation», *INFORMS Journal on Computing*, **8**(1), págs. 1–15.

MCGEOCH, C. C. (2001), «Experimental analysis of algorithms», *Notices of American Mathematical Society*, **48**, págs. 304–311.

METROPOLIS, N., A. W. ROSENBLUTH, M. N. ROSENBLUTH, A. H. TELLER y E. TELLER (1953), «Equation of state calculations by fast computing machines», *Journal of Chemical Physics*, **21**, págs. 1087–1092.

MILGRAM, S. (1967), «The small world problem», *Psychology Today*, **1**(1), págs. 61–67.

MILLER, R. G. (1997), *Beyond ANOVA: Basics of applied statistics*, Chapman & Hall, Boca Raton, Florida, Estados Unidos.

MONASSON, R., R. ZECCHINA, S. KIRKPATRICK, B. SELMAN y L. TROYANSKY (1999), «Determining computational complexity from characteristic 'phase transitions'», *Nature*, **400**(6740), págs. 133–137.

MONTGOMERY, D. C. (2006), *Design and analysis of experiments*, John Wiley & Sons, Nueva Jersey, Estados Unidos.

MORET, B. M. E. (2002), «Towards a discipline of experimental algorithmics», *DIMACS Monographs in Discrete Mathematics and Theoretical Computer Science*.

MULET, R., A. PAGNANI, M. WEIGT y R. ZECCHINA (2002), «Coloring random graphs», *Physical Review Letters*, **89**(26).

NEWMAN, M. E. J. (2003), «The structure and function of complex networks», *SIAM Review*, **45**, págs. 167–256.

NISKANEN, S. y P. ÖSTERGÅRD (2003), «Cliquer User's Guide, Version 1.0», *Informe Técnico T48*, Communications Laboratory, Helsinki University of Technology, Espoo, Finlandia.

NOVAES DE SANTANA, C. (2005), «Analise da pluviometria do nordeste brasileiro segundo modelagem em redes», .

ORPONEN, P., K.-I. KO, U. SCHÖNING y O. WATANABE (1994), «Instance Complexity», *Journal of ACM*, **41**(1), págs. 96–121.

ORTEGA-IZAGUIRRE, R. (2008), *Estudio de las propiedades topológicas en redes complejas con diferente distribución del grado y su aplicación en la búsqueda de recursos distribuidos*, Tesis Doctoral, CICATA - IPN Unidad Altamira, Altamira, Tamaulipas, México.

PAPADIMITRIOU, C. M. (1994), *Computational complexity*, Addison-Wesley, Reading, Massachusetts, Estados Unidos.

PELLEG, D. y A. MOORE (2000), «X-means: Extending K-means with efficient estimation of the number of clusters», en *Proceedings of the 17th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, California, Estados Unidos, págs. 727–734.

PENROSE, M. (2003), *Random geometric graphs (Oxford Studies in Probability)*, Oxford University Press, Nueva York, Estados Unidos.

PÉREZ, J., L. CRUZ, R. PAZOS, V. LANDERO, G. REYES, C. ZAVALA, H. FRAIRE y V. PÉREZ (2008), «A Causal Approach for Explaining Why a Heuristic Algorithm Outperforms Another in Solving an Instance Set of the Bin Packing Problem», en A. An, S. Matwin, Z. Ras y D. Slezak (editores), *Foundations of Intelligent Systems*, *Lecture Notes in Computer Science*, tomo 4994, Springer Berlin / Heidelberg, págs. 591–598.

POPOVICI, E. y K. DE JONG (2005), «Understanding cooperative co-evolutionary dynamics via simple fitness landscapes», en *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, ACM, Nueva York, Nueva York, Estados Unidos, págs. 507–514.

PORUMBEL, D. C., J.-K. HAO y P. KUNTZ (2010), «A search space cartography for guiding graph coloring heuristics», *Computers & Operation Research*, **37**, págs. 769–778.

RESENDE, M. G. C. y G. VEIGA (2003), «An annotated bibliography of network interior point methods», *Networks*, **42**(2), págs. 114–121.

ROSAS, V. P. (2007), *Modelo causal del desempeño de algoritmos metaherísticos en problemas de distribución de objetos*, Tesis de Maestría, Instituto Tecnológico de Ciudad Madero, Cd. Madero, Tamaulipas, México.

ROSS, S. M. (2009), *Introduction to probability and statistics for engineers and scientists*, Elsevier Academic Press, San Diego, California, Estados Unidos.

SANTILLÁN, C. G., L. C. REYES, E. M. CONDE, E. SCHAEFFER y G. C. VALDEZ (2010), «A self-adaptive ant colony systems for semantic query routing problem in P2P networks», *Computación y Sistemas*, **13**(4), págs. 433–448.

SCHAEFFER, S. E. (2006), «Algorithms for nonuniform networks», *Research Report A102*, Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finlandia.

SELMAN, B., H. LEVESQUE y D. MITCHELL (1992), «A new method for solving hard satisfiability problems», en *Proceedings of the tenth national conference on Artificial intelligence*, AAAI Press, págs. 440–446.

SEN, P., S. DASGUPTA, A. CHATTERJEE, P. A. SREERAM, G. MUKHERJEE y S. S. MANNA (2003), «Small-world properties of the Indian railway network», *Physical Review E*, **67**(3).

SKIENA, S. S. (2008), *The algorithm design manual*, The Algorithm Design Manual, Springer-Verlag Nueva York, Nueva York, Nueva York, Estados Unidos.

SMITH-MILES, K., J. VAN HEMERT y X. LIM (2010), «Understanding TSP difficulty by learning from evolved instances», en C. Blum y R. Battiti (editores), *Learning and Intelligent Optimization*, *Lecture Notes in Computer Science*, tomo 6073, Springer-Verlag Berlin, Heidelberg, Alemania, págs. 266–280.

SMITH-MILES, K. A., R. J. JAMES, J. W. GIFFIN y Y. TU (2009), «A Knowledge discovery approach to understanding relationships between rcheduling problem structure and heuristic performance», en T. Stützle (editor), *Learning and Intelligent Optimization*, Springer-Verlag Berlin, Heidelberg, Alemania, págs. 89–103.

STADLER, P. F. (2002), «Fitness landscapes», *Applied Mathematics and Computing*, **117**, págs. 187–207.

SURRY, P. D., N. J. RADCLIFFE y I. D. BOYD (1995), «A multi-objective approach to constrained optimisation of gas supply networks: the COMOGA method», en *Selected Papers from AISB Workshop on Evolutionary Computing*, Springer-Verlag, London, Reino Unido, págs. 166–180.

TRICK, M. (2010), «Network resources for coloring a graph», Acceso: 12/07/2010, URL `http://mat.gsia.cmu.edu/COLOR/color.html`.

TUKEY, J. W. (1991), «The philosophy of multiple comparisons», *Statistical Science. A Review Journal of the Institute of Mathematical Statistics*, **6**(1), págs. 100–116.

ČERNÝ, V. (1985), «Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm», *Journal of Optimization Theory and Applications*, **45**, págs. 41–51.

VAN HEMERT, J. I. (2006), «Evolving combinatorial problem instances that are difficult to solve», *Evolutionary Computation*, **14**, págs. 433–462.

VASSILEV, V. K., T. C. FOGARTY y J. F. MILLER (2000), «Information characteristics and the structure of landscapes», *Evolutionary Computation*, **8**, págs. 31–60.

VAZIRANI, V. V. (2001), *Approximation algorithms*, Springer-Verlag Berlin, Heidelberg, Alemania.

WALSH, T. (2009), «Where are the really hard manipulation problems? the phase transition in manipulating the veto rule», en *Proceedings of the 21st international jont conference on Artifical intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, págs. 324–329.

WATSON, J.-P., L. BARBULESCU, A. E. HOWE y L. D. WHITLEY (1999), «Algorithm performance and problem structure for flow-shop scheduling», en *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, Menlo Park, CA, USA, págs. 688–695.

WATTS, D. J. y S. H. STROGATZ (1998), «Collective dynamics of "small-world" networks», *Nature*, **393**, págs. 440–442.

WEINBERGER, E. (1990), «Correlated and uncorrelated fitness landscapes and how to tell the difference», *Biological Cybernetics*, **63**, págs. 325–336.

WRIGHT, S. (1932), «The roles of mutation, inbreeding, crossbreeding and selection in evolution», *Proceedings of the 6th International Congress of Genetics*, **1**, págs. 356–366.

XU, J. (2010), *Topological structure and analysis of interconnection networks*, primera edición, Springer Publishing Company, Inc., Dordrecht, Holanda.

YOSSI, B. y R. POLI (2005), «Information landscapes and the analysis of search algorithms», en *Proceedings of the 2005 Conference on Genetic and evolutionary computation*, ACM, Nueva York, Nueva York, Estados Unidos, págs. 1287–1294.

ZANAKIS, S. H., J. R. EVANS y A. A. VAZACOPOULOS (1989), «Heuristic methods and applications: A categorized survey», *European Journal of Operational Research*, **43**(1), págs. 88–110.

ZOU, K., A. O'MALLEY y L. MAURI (2007), «Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models», **115**.

FICHA AUTOBIOGRÁFICA

Tania Turrubiates López

Candidata para el grado de Doctor en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Tesis:

COMPLEJIDAD COMPUTACIONAL ESTRUCTURAL
EN REDES COMPLEJAS

Nací el 25 de noviembre de 1979 en la ciudad y puerto de Tampico, Tamaulipas; mis padres Modesto Turrubiates Atilano y Noralba López Nieto, hermanos Modesto y Nora Ariadna. En marzo del 2002 obtuve el título de Ingeniero en Sistemas Computacionales con especialidad en Ingeniería de Software por el Instituto Tecnológico de Ciudad Madero. En el año 2005 ingresé a la Maestria en Ciencias en Ciencias de la Computación en el Instituto Tecnológico de Ciudad Madero, obteniendo el grado de Maestra en Ciencias en el año 2007. En el año 2009 ingresé al Programa de Doctorado en Ingeniería con especialidad en Ingeniería de Sistemas en la Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León, trabajando bajo la supervisión de la Dra. Satu Elisa Schaeffer en el proyecto de tesis titulado “Complejidad computacional estructural en redes complejas”.

ÍNDICE ALFABÉTICO