TESIS TESIS TESIS TESIS TESIS

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

Centro de Ciencias Básicas

Metaheurística Evolutiva Híbrida para la Minimización de Cruces de las Aristas en el Dibujado de Grafos

Tesis

Para Obtener el Grado de Maestro en Ciencias Exactas, Sistemas y de la Información: Área de Inteligencia Artificial

Presenta:

Sergio Enríquez Aranda

Tutor:

Dra. Eunice Esther Ponce de León Sentí

Sinodales:

Dra. Elva Díaz Díaz

Dr. Alejandro Padilla Díaz

Aguascalientes, Ags., Junio 2009

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

DEDICATORIA

A mis padres…..

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

AGRADECIMIENTOS

A la Universidad Autónoma de Aguascalientes por su apoyo económico y de preparación, al Rector de la Universidad Autónoma de Aguascalientes Mtro. Rafael Urzúa Macías, y al CONCYTEA por su Apoyo económico.

A todos nuestros maestros Dra. Elva Díaz Díaz, Dr. Alejandro Padilla Díaz, Dr. Hermilo Sánchez Cruz, que durante este tiempo dedicaron su tiempo y esfuerzo en la preparación y superación de todos nosotros, y especialmente a mi tutor Dra. Eunice Ponce de León Sentí por toda su paciencia, consejos y apoyo invaluable para el logro de esta nueva meta.

A mis compañeros Sergio Nevárez, Leoncio Ibarra, Mario Alberto Rodríguez, Israel de la Parra por su apoyo incondicional y de equipo que siempre mostraron a lo largo de este posgrado.

A mis padres y hermanos que siempre han estado unidos y que son el ejemplo de superación y honestidad que me motiva, en especial a mi hermano Rogelio y mi Mamá.

A mi esposa e hijas por todo su amor y sobre todo su comprensión que me han brindado durante todo este tiempo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Por este conducto autorizamos al tesista:

Sergio Enríquez Aranda

La impresión de su documento final de tesis, con título "**Metaheurística Evolutiva Híbrida para la Minimización de Cruces de las Aristas en el Dibujado de Grafos",** ya que cumple con los requisitos de contenido y forma exigidos en la Universidad Autónoma de Aguascalientes.

Asesor

Dra. Eunice Esther Ponce de León Sentí

Sinodales

Dra. Elva Díaz Díaz

Dr. Alejandro Padilla Díaz

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Centro de Ciencias Básicas

C. SERGIO ENRÍQUEZ ARANDA
ALUMNO (A) DEL LA MAESTRÍA EN CIENCIAS EXACTAS, SISTEMAS Y DE LA INFORMACIÓN
ÁREA DE INTELIGENCIA ARTIFICIAL
P R E S E N T E .

Estimado (a) Alumno (a) Enríquez:

Por medio de este conducto me permito comunicar a Usted que habiendo recibido los votos aprobatorios de los revisores de su trabajo de tesis titulado: "Metaheurística Evolutiva Híbrida para la Minimización de Cruces de las Aristas en el Dibujado de Grafos", hago de su conocimiento que puede imprimir dicho documento y continuar con los trámites para la presentación de su examen de grado.

Sin otro particular me permito saludarle muy afectuosamente.

A T E N T A M E N T E
Aguascalientes, Ags., 26 de junio de 2009
"LUMEN PROFERRE"
EL DECANO

DR. FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ

UNIVERSIDAD AUTONOMA DE AGUASCALIENTES
CENTRO DE CIENCIAS BASICAS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

RESUMEN

La presente tesis muestra la implementación de diferentes metaheurísticas para el dibujado de grafos en el plano, con el propósito de lograr de manera eficiente, la minimización de los cruces de las aristas en el dibujado del grafo. Las aristas del grafo son líneas rectas. Entre las metaheurísticas utilizadas para lograr la minimización de cruces se encuentran el AGS (Algoritmo Genético Simple), el AEDMU (Algoritmo de Estimación de la Distribución Marginal Univariada) y el EC (Escalador de Colinas), así mismo, se realizaron cambios en el diseño original de las metaheurísticas buscando mejorar los resultados en la minimización. Dentro de la aportación que la presente tesis a dado a la investigación es un algoritmo híbrido llamado HUx que es una combinación entre el algoritmo AEDMU y el EC. El HUx ha demostrando ser más eficiente en la búsqueda de la solución óptima, que en nuestro caso es, aquel dibujado que tenga el número mínimo de cruces de las aristas del grafo, que el ECE, el AEDMU y los AGS. La medición de los resultados fue hecha utilizando diferentes grafos planares (con cero cruces entre sus aristas) y no planares (con al menos un cruce entre sus aristas), que fueron los sujetos de medición a los que se les aplicó las diferentes metaheurísticas. La calidad y la frecuencia de aparición de las soluciones óptimas registradas por el algoritmo HUx, fueron superiores a las registradas por los algoritmos contra los cuales se hizo esta medición y que forman parte del estado del arte, dentro del campo de dibujado de grafos. Cabe mencionar que en cada uno de los casos estudiados, la metaheurística híbrida HUx siempre encuentra el optimo para los grafos de prueba, además, la frecuencia de aparición del optimo es un 25% mayor que el segundo mejor algoritmo de todos los estudiados. Para la visualización de los grafos se desarrolló una interfase gráfica propia llamada MinX la cual muestra el resultado de cada grafo obtenido por las metaheurísticas implementadas en esta tesis.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE GENERAL

DEDICATORIA i

AGRADECIMIENTOS ii

RESUMEN iii

ÍNDICE GENERAL iv

ÍNDICE FIGURAS vii

ÍNDICE TABLAS ix

Capítulo 1. INTRODUCCIÓN

1.1. Introducción 1

1.2. Planteamiento del problema 3

1.3. Justificación 5

1.4. Objetivos 6

1.5. Metas 7

1.6. Difusión y aprobación del trabajo 8

1.7. Contenido de la tesis 11

Capítulo 2. MARCO TEÓRICO

2.1. Teoría de grafos 13

2.1.1. Grafos 13

2.1.1.1. Grafos generales no dirigidos 14

2.1.1.2. Grafos dirigidos 17

2.1.1.3. Subgrafos 17

2.1.1.4. Grafo planar 18

2.1.2. Minimización de cruces 20

2.2. Geometría analítica 22

2.2.1. El espacio Euclidiano 22

2.2.2. Puntos del espacio Euclidiano 23

2.2.3. Distancia entre dos puntos 25

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.2.4. La recta 26
2.2.4.1. Pendiente de la recta 27
2.2.4.2. Intercepto 29
2.2.4.3. Intersección entre dos rectas 29
2.3. Dibujado de automático de grafos 32
2.3.1. Historia del dibujado de grafos 32
2.3.2. Estructura del dibujado automático de grafos 33
2.3.2.1. Objetos de dibujado 33
2.3.2.2. Convenciones de dibujado 34
2.3.2.3. Reglas de dibujado 36
2.3.2.3.1. Reglas básicas, semánticas y estructurales 36
2.3.2.4. Relaciones de Prioridad 38
2.3.2.5. Algoritmos de dibujado 38
2.4. Inteligencia artificial 39
2.4.1. Algoritmo Genético Simple (AGS) 40
2.4.2. Algoritmo Escalador de Colinas (EC) 44
2.4.3. Algoritmo de Estimación de Distribuciones (EDA) 45
2.4.3.1. Algoritmo de Estimación de la Distribución Marginal Univariada (AEDMU) 46
2.5. Revisión de algoritmos para el dibujado de grafos 48

Capítulo 3. ALGORITMOS PARA EL DIBUJADO DE GRAFOS

3.1. Representación de la solución 49
3.2. Función objetivo 50
3.3. Algoritmo Genético Simple para dibujado de grafos 52
3.4. Algoritmo Escalador de Colinas para dibujado de grafos 56
3.5. Algoritmo de la Estimación Marginal Univariada para el dibujado de grafos 58
3.6. Algoritmo híbrido HUx para el dibujado de grafos 59

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 4. SISTEMA MinX

4.1. Presentación del Sistema MinX 62

4.2. Características generales del Sistema MinX 62

4.3. Diagrama general del Sistema MinX 63

4.3.1. Modelado del sistema MinX 64

4.3.2. Objetos del Sistema MinX 66

Capítulo 5. DISEÑO DE EXPERIMENTOS Y ANALISIS DE RESULTADOS

5.1. Diseño de experimentos 71

5.1.1. Diseño del experimento # 1: AGS 71

5.1.2. Diseño del experimento # 2: EC con intercambio de vértices 72

5.1.3. Diseño del experimento # 3: EC con reinicio aleatorio 72

5.1.4. Diseño del experimento # 4: AEDMU 72

5.1.5. Diseño del experimento # 5: HUx 72

5.1.6. Diseño del experimento # 6: Grafo Compuesto 72

5.2. Análisis de resultados 73

5.2.1. Comparación de resultados entre los algoritmos implementados en esta tesis 75

5.2.2. Comparación de resultados obtenidos por el algoritmo híbrido HUx y los resultados obtenidos de los algoritmos tomados de la literatura 79

5.2.3. Estabilidad del algoritmo híbrido HUx con el grafo Compuesto 81

Capítulo 6. CONCLUSIONES

6.1. Conclusiones generales de la tesis 84

6.2. Trabajo futuro 85

BIBLIOGRAFÍA 86

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE FIGURAS

Figura 1.1. Relación de objetivos de la tesis 8
Figura 2.1. Clasificación general de los grafos 14
Figura 2.2. El grafo de V = {1,…,7} con un conjunto E = {{1,2},{1,5},{2,5},{3,4},{5,7}} 15
Figura 2.3. Grafo completo K3 (triángulo) 15
Figura 2.4. Un camino $P = P_6$ en G 16
Figura 2.5. Ejemplo de un dígrafo 17
Figura 2.6. Ejemplo de un subgrafo 18
Figura 2.7. Ejemplo de un grafo planar 20
Figura 2.8. Diferentes dibujados de un grafo abstracto con diferentes números de aristas cruzadas (51,12 y cuatro respectivamente. Criterios estéticos como pocos dobleces, longitudes de arista uniforme, o área pequeña de dibujado, están a favor de los dos primeros dibujados. Sin embargo, con respecto al número de mínimo de aristas cruzadas el último es mejor 21
Figura 2.9. Ejemplo de una recta 23
Figura 2.10. Intersección entre $\ell_1' \cap \ell_2'$ que corresponde al punto p(x, y) 24
Figura 2.11. Plano Cartesiano 24
Figura 2.12. Interpretaciones equivalentes de punto en el plano, una pareja de números reales y un vector 25
Figura 2.13. Distancia entre dos puntos 25
Figura 2.14. Pendiente de una recta 27
Figura 2.15. Pendiente con m > 0 28
Figura 2.16. Pendiente con m < 0 28
Figura 2.17. Punto y pendiente de la recta ℓ 29
Figura 2.18. Grafo compuesto dividido en capas 34
Figura 2.19. Sistema de coordenadas estándar y compuesto 35
Figura 2.20. Reglas básicas y estructurales utilizadas en esta tesis 37

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Figura 2.21. Pseudocódigo del Algoritmo Genético Simple 41
Figura 2.22. Operador de cruce de un AGS 43
Figura 2.23. Operador de mutación de un AGS 43
Figura 2.24. Pseudocódigo de la aproximación EC 45
Figura 2.25. Pseudocódigo de la aproximación EDA 46
Figura 2.26 Pseudocódigo de la aproximación UMDA 47
Figura 3.1. Representación de la solución (objeto vértice) 49
Figura 3.2. Representación de la solución (objeto arista) 50
Figura 3.3. Objeto grafo utilizado por la función objetivo 51
Figura 3.4. Función objetivo (objeto grafo) 52
Figura 3.5. Estructura de datos de un individuo de la población AGS 53
Figura 3.6. Estructura de datos de la población del AGS 54
Figura 3.7. Pseudocódigo del AGS 55
Figura 3.8. Estructura de datos del individuo general utilizado por el algoritmo EC 56
Figura 3.9. Pseudocódigo del EC 57
Figura 3.10. Estructura de datos de la población utilizada por el AEDMU 58
Figura 3.11. Pseudocódigo del Algoritmo de la Estimación Marginal Univariada 59
Figura 3.12. Hibridación hecha al Algoritmo de la Estimación Marginal Univariada 60
Figura 4.1. Modelo General del Sistema MinX 65
Figura 4.2. Tabla de resultados del grafo Hobbs minimizado por la metaheurística híbrida HUx 69
Figura 4.3. Dibujado automático del grafo Hobbs minimizado por la metaheurística híbrida HUx 70
Figura 5.1. Promedio de cruces obtenidos en cada grafo 76
Figura 5.2. Promedio general de cruces obtenidos por las diferentes metaheurísticas 76
Figura 5.3. Grafo Compuesto, minimizado por el algoritmo híbrido HUx 82
Figura 5.4. Grafo Compuesto visualizado por el Sistema MinX 83

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ÍNDICE TABLAS

Tabla 2.1. Convenciones de colocación de vértices 35
Tabla 2.2. Convenciones redirección para aristas 35
Tabla 2.3. Relaciones de prioridad del dibujado automático de grafos 38
Tabla 5.1. Propiedades Generales de los grafos tomados de la literatura 74
Tabla 5.2. Número promedio de cruces obtenidos para los grafos tomados de la literatura 75
Tabla 5.3. Número de veces que el algoritmo encuentra el óptimo de un total de 20 evaluaciones 77
Tabla 5.4. Número de veces que el algoritmo encuentra el óptimo de un total de 20 evaluaciones 77
Tabla 5.5. Alcance de los objetivos logrados por las diferentes metaheurísticas 78
Tabla 5.6. Comparación de resultados del grafo Compuesto 81

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 1. INTRODUCCIÓN

Este capítulo muestra el trabajo de tesis, de forma general, empezando por el planteamiento del problema de dibujado de grafos minimizando el número de cruces de las aristas, así como la justificación, los objetivos y metas de la presente investigación. Al final del capítulo se presenta la descripción de las contribuciones de la tesis, las publicaciones realizadas en el período de estudios de la maestría y por último se presenta un epígrafe con la explicación del contenido de esta tesis.

1.1 Introducción

El estudio de grandes cantidades de información que está interrelacionada puede ser comprendida con mayor facilidad por medio de una representación visual. Para esto existe una tecnología basada en algoritmos de dibujado de grafos que ayudan al usuario en el análisis y representación de esa información. La presentación de la información en forma visual facilita al usuario en la búsqueda patrones o anomalías existentes que de otra manera serían más difíciles de descubrir por medio de los métodos tradicionales (métodos de dibujado manual o algoritmos específicos que resuelven un problema dado). En la mayoría de los casos no existe información sobre como deben ser colocados los elementos de un grafo para facilitar su visualización, aunque normalmente las relaciones existentes entre esos elementos influyen en el diseño final del grafo [Walshaw (2003)].

Los grafos están basados en modelos matemáticos que ayudan en la representación de objetos de la vida real, como son las redes de computadoras, las redes de carreteras, las redes eléctricas, las redes hidráulicas y las redes sociales, entre otras. En el campo de la física y la química así como en bioinformática [Wiese y Eicher (2006)] se cuenta con diseños de grafos que modelan interacciones entre partículas (como es el caso de las estructuras moleculares). En ingeniería de software [Alvarez (1995)] [Sommerville (1995)] los grafos son usados para representar diagramas entidad-relación (diagramas utilizados en el diseño de bases de datos jerárquicas y relacionales), diagramas de flujo de información, por mencionar algunos. En electrónica se presenta esta necesidad de representación en el diseño de circuitos electrónicos, diseño de circuitos VLSI (Very Large Scale Integration)

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

[Buchheim et. al. (2004)], diseño de circuitos lógicos y hardware reconfigurable [Serna Pérez E. (2001)]. En inteligencia artificial los grafos son utilizados para implementar redes semánticas, redes Markovianas y redes Bayesianas, en la representación del conocimiento, [Rossete (2000)] [Rossete et. al (1999)] [Tunkelang (1994)], algoritmos de aprendizaje de modelos gráficos Markovianos [Diaz (2008).] [Lauritzen (1996)].

Las investigaciones dentro del campo de dibujado de grafos se ha llevado a cabo desde la década 1930, pero comenzó ha tomar auge a partir de la década de 1980 con el avance tecnológico, el surgimiento de las estaciones de trabajo y la creación de las computadoras personales. Este avance tecnológico ha provocado que se pueda mejorar continuamente la representación visual de los programas de computadoras utilizados por los usuarios. Desde entonces se ha reconocido la importancia que tienen las interfases humanas dentro de nuestra vida diaria. Con el paso del tiempo se ha comprobado que los grafos son más fáciles de entender y de recordar cuando son representados como diagramas, sin embargo, la utilidad que un grafo nos puede proporcionar depende en gran medida de la forma en que este ha sido dibujado. Existen varios problemas para que un grafo pueda ser visto con claridad y estos problemas están relacionados con diferentes criterios estéticos a la hora de dibujar el grafo, entre los que se encuentran el empalme de vértices y aristas, el dibujado simétrico del grafo, el balance de la longitud y amplitud de las aristas, la reducción del área de trazado, así como el problema de encontrar un trazado que minimice el número de cruces en sus aristas, entre otras. En este último criterio se han desarrollado diferentes métodos para la solución de este problema ya que es considerado el problema más importante dentro del dibujado de grafos [Purchase et. al. (1996)] y es el objeto de estudio de la presente tesis. Un buen dibujado nos da la posibilidad de comunicar de manera rápida y exacta el significado de la información que estamos analizando, por otro lado, un mal dibujado nos da una alta posibilidad de confusión o malentendido. En este sentido se entenderá por legibilidad de un grafo a la capacidad de transmitir el significado de un grafo de manera clara y rápida [Martig (2000)]. La legibilidad de un grafo se puede expresar por medio de criterios que tienen que ver con las características estéticas que tiene un grafo, como la minimización de cruces entre aristas, la minimización del área utilizada para

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

dibujar el grafo, la simetría entre las aristas del grafo, la colocación uniforme de los elementos de un grafo, entre otras [Curino et. al. (2005)].

El problema de legibilidad en la visualización de grafos es un problema abierto actualmente para la investigación y se ubica dentro del campo de dibujado de grafos. El dibujado de grafos consiste en colocar todos los elementos (vértices y aristas) pertenecientes a un grafo de tal manera que el usuario pueda ver maximizada su legibilidad. Para los programas que visualizan diagramas este problema es de suma importancia ya que impide ver con claridad el diseño de un grafo. La forma tradicional de atacar este problema es creando algoritmos específicos [Buchheim et. al. (2004)] que resuelvan un problema de visualización dado, esto provoca la individualización del problema haciendo que los algoritmos sean poco flexibles y lentos en su aplicación. Por lo tanto existe la necesidad de crear algoritmos más generales y flexibles que resuelvan esta problemática por medio de metaheurísticas, Kozo Sugiyama, plantea:

> *Cuando sea posible la creación de buenos diagramas dibujados de forma automática, las molestas operaciones de dibujado de grafos podrán ser simplificadas y será posible desarrollar herramientas de dibujado de grafos de alto nivel. En lo futuro se cree que la utilidad de los métodos de dibujado de grafos se irá incrementando conforme pase el tiempo.* [Sugiyama (2002)].

1.2 Planteamiento del problema

Los grafos son modelos abstractos con los que podemos expresar cualquier relación que exista entre los elementos de cualquier clase o información. Kozo Sugiyama, define: *"Los grafos son modelos abstractos para la estructura del mundo real, y son una poderosa forma de expresión de conceptos abstractos"* [Sugiyama (2002)].

Cuando hacemos uso del método de visualización de grafos para la comunicación de conceptos, es necesario producir buenos diagramas y esto no es una tarea sencilla, aún y cuando el dibujado de un grafo es hecho a mano existen varios inconvenientes como son el tiempo que se invierte, el grafo tiene que ser pequeño, el hecho de realizar cualquier

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

cambio en el grafo implica tener que volverlo a dibujar, el acomodo de los elementos implica un gran desafío ya que se tiene que encontrar la mejor manera posible de maximizar su legibilidad y por lo general el dibujado de grafos hecho a mano no produce buenos resultados. Es por eso que se ha incrementado el interés por la utilización de métodos de dibujado automático de grafos mediante el uso de computadoras.

La utilidad que un grafo puede brindar a los usuarios depende de su legibilidad, es decir de la capacidad de transmitir el significado de ese grafo de manera clara y rápida [Martig y Señas (2000)]. Los aspectos de legibilidad se pueden expresar a través de criterios estéticos, tales como la minimización de cruces entre aristas o la minimización del área utilizada para dibujar el grafo, por lo que diversos problemas de visualización de grafos pueden ser formalizados como problemas de optimización multiobjetivo (con múltiples objetivos), como por ejemplo la obtención de una representación con un mínimo número de cruces, simetría entre las aristas, y la colocación uniforme de los elementos de un grafo, entre otras.

La representación de grafos debe hacerse de la manera más legible posible y este es precisamente el objeto de estudio del dibujado de grafos que cuenta actualmente con un marco teórico muy amplio en donde se incluyen diferentes artículos, tesis de maestría y tesis de doctorado de todo el mundo [Kant (1993)] [Rossete (2000)] [Sugiyama (2002)] [Ebner (2005)] [Newton (2007)] [Abdullah (2007)], así como varias conferencias especializadas en el tema como es el Simposio Internacional sobre Dibujado de Grafos que se lleva cada año en diferentes países http://www.graphdrawing.org .

El problema de la minimización de cruces es uno de los problemas más básicos y comunes dentro de los problemas de optimización y ocupa un lugar muy especial dentro del dibujado automático de grafos. La forma de plantear este problema es sumamente fácil, como menciona Christoph Buchheim: *"dado un grafo dibuja este con el número mínimo de cruces entre sus aristas"* [Buchheim et. al. (2004)].

La minimización de cruces tiene muchas aplicaciones prácticas sin mencionar su importancia topológica en términos teóricos. Dentro del dibujado automático de grafos, el problema de la legibilidad de un grafo depende en gran medida del número de cruces que

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

este tiene ya que estos afectan el diseño final del grafo. La presencia de cruces distrae la concentración del usuario y confunde su visibilidad al momento de determinar si dos vértices están unidos por una determinada arista o no.

El diseño de circuitos VLSI (Very Large Scale Integration) es otra de las aplicaciones que puede ser abordada por el problema de la minimización de cruces. El objetivo del diseño de VLSI es colocar transistores sobre chips en dos dimensiones. Los transistores se deben conectar por medio de cables los cuales tienen que ser colocados sobre el chip, cada cruce de dos cables dentro del chip causa un costo adicional en el proceso de fabricación, y para la industria sería preferible disminuir el número de cruces dentro del chip. La minimización de cables cruzados traería como consecuencia una reducción en el costo de fabricación del chip [Buchheim et. al. (2004)].

Después de varias décadas de investigación el problema de la minimización de cruces continúa siendo un campo fértil para la investigación, ya que no ha sido completamente explorado. Dentro de los problemas abiertos para la investigación, el problema de la minimización de cruces es uno de los problemas más desafiantes e interesantes que existen, tanto en la parte teórica como en la parte práctica. Las aproximaciones que actualmente se han ido obteniendo en el problema de la minimización de cruces por lo general son de naturaleza heurística, y este es el objetivo de esta investigación; lograr por medio de métodos heurísticos la minimización de cruces, siendo esto uno de los elementos que ayuda a maximizar la legibilidad del grafo en un sin número de aplicaciones y maximizar su legibilidad.

El planteamiento del problema de la presente tesis consiste en encontrar métodos eficientes para el dibujado de grafos en el plano minimizando el número de cruces en las aristas de un grafo.

1.3 Justificación

El problema de la minimización de cruces entre las aristas de un grafo es un problema muy antiguo, clásico y muy importante dentro del dibujado de grafos. El resultado de una minimización proporciona una mejor legibilidad del grafo y al mismo tiempo influye

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

directamente en la efectividad de su diseño. La técnica de minimización de cruces es muy útil en grafos pequeños y de suma importancia sobre todo en grafos de gran escala, ya que se podrían identificar propiedades que de otra manera permanecerían ocultas a la vista del usuario, y aún así en muchos de los casos los cruces no pueden ser evitados, por ejemplo cuando los grafos no son planares [Bender y Williamson (2005)] [Diestel (2005)]. Un número grande de cruces decrementa la legibilidad de un grafo, especialmente en grafos densos [Zheng et. al. (2005)].

Este trabajo es motivado por ese problema, se pretende minimizar el número de cruces que surgen entre las aristas de un grafo, sujeto además, a prohibir el empalme de vértices y darle un enfoque metaheurístico para lograr con un motor de búsqueda encontrar dibujados de grafos que minimicen la cantidad de cruces de las aristas. La presente tesis tiene como principal contribución dentro del campo de dibujado de grafos específicamente en el problema de la minimización de cruces de las aristas; el diseño, desarrollo e implementación de una metaheurística híbrida HUx, que minimiza de manera más eficiente los cruces en las aristas de un grafo y la cual aprovecha las propiedades mas destacadas de dos metaheurísticas a saber, la metaheurística conocida como Algoritmos de Estimación de la Distribución (AED) y la metaheurística de búsqueda local, en particular se ha trabajado con el algoritmo AEDMU del tipo AED y el algoritmo escalador de colinas estocástico (ECE) que es un algoritmo de la metaheurística de búsqueda local. Las características que se han combinado son la capacidad del AEDMU de captar propiedades globales de las soluciones mediante la estimación de un modelo probabilístico en este caso, el más simple (modelo de independencias) que evoluciona con la tendencia a captar las características que se imponen en la función objetivo y por otro lado la rapidez y eficiencia mostrada en la literatura [Rossete (2000)], para el problema que aquí nos ocupa, del algoritmo ECE.

1.4 Objetivos

Con base en la justificación descrita anteriormente, se proponen los siguientes objetivos para el presente trabajo de investigación, dichos objetivos fueron alcanzados con éxito y se describen a continuación:

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Objetivo general

- Encontrar métodos eficientes para el dibujado de grafos en el plano minimizando el número de cruces de las aristas.

Objetivos específicos

- Encontrar una modelación adecuada para el problema de dibujado de grafos en el plano mediante un algoritmo genético.
- Encontrar una modelación adecuada para el problema de dibujado de grafos en el plano mediante un algoritmo de estimación de la distribución.
- Encontrar una modelación adecuada para el problema de dibujado de grafos en el plano mediante un algoritmo escalador de colinas.
- Encontrar una modelación adecuada para el problema de dibujado de grafos en el plano mediante un algoritmo híbrido que mejore los anteriores algoritmos.
- Diseño e implementación de un sistema para la visualización de grafos con el menor número de cruces facilitando la legibilidad del mismo.
- Utilización de “benchmarks” para comprobar la efectividad de los métodos y herramientas propuestas contra los métodos ya existentes.
- Análisis estadístico del desempeño de los algoritmos propuestos.

1.5 Meta

El trabajo de tesis tiene como meta final la creación de una herramienta que facilite la visualización de grafos maximizando su legibilidad por medio de la minimización de cruces en las aristas, basado en la metaheurística hibrida HUx. Con esta herramienta se pueden visualizar de manera mas clara grafos, con el propósito de facilitar el análisis y exploración de las relaciones que existen entre los elementos del grafo, observar e interpretar las relaciones que se presentan en el grafo. El presente trabajo de investigación tiene potencial para que en un futuro se pueda utilizar comercialmente o ser útil en el área de investigación tanto básica como aplicada.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La figura 1.1 muestra la relación entre los diferentes objetivos de esta tesis.

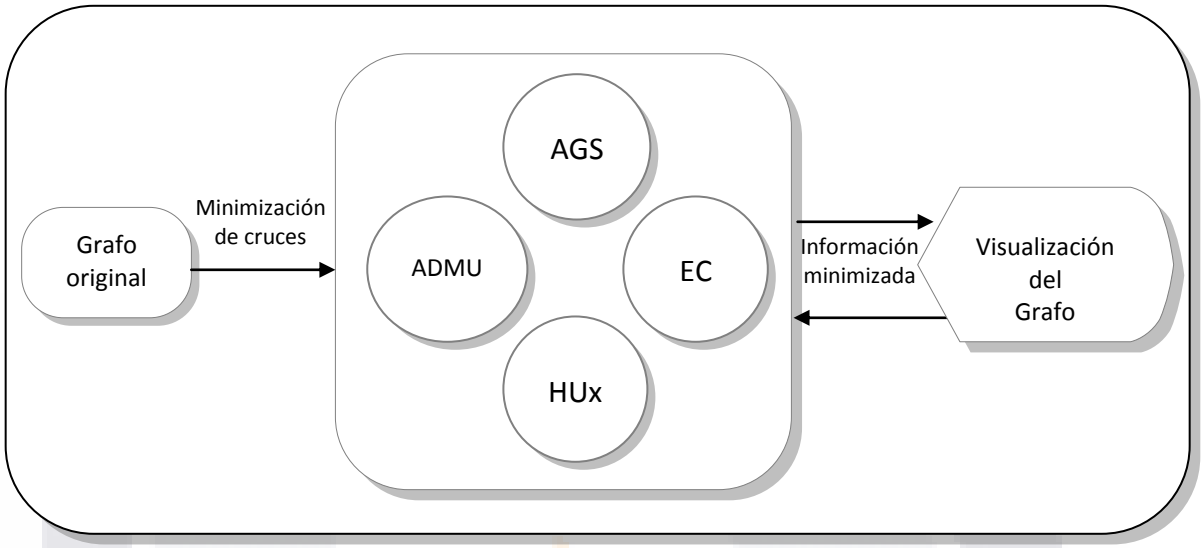


Figura 1.1. Relación de objetivos de la tesis.

1.6 Difusión y aprobación del trabajo

En el transcurso de este tesis se han publicado los siguientes trabajos:

1. Participación como "Ponente" en el Coloquio del Posgrado en Ciencias Exactas, Sistemas y de la Información de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., del 10 al 12 de Enero de 2007.

2. Participación como "Ponente" en el Coloquio del Posgrado en Ciencias Exactas, Sistemas y de la Información de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., el 21 y 22 de Junio de 2007.

3. Participación como "Ponente" en el Tercer Congreso Estatal la Investigación en el Posgrado de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., el 26 y 27 de Octubre de 2007.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

4. Publicación del articulo “Algoritmo Greedy: una Aproximación para el Problema del Conjunto Independiente Maximal”, Autores: Sergio Enríquez Aranda y Eunice Ponce de León Sentí, publicado en la memoria del Tercer Congreso Estatal la Investigación en el Posgrado de la Universidad Autónoma de Aguascalientes.

5. Participación como “Ponente” en el Cuarto Congreso Mexicano de Computación Evolutiva del Centro de Investigación en Matemáticas A. C. , llevado a cabo en Guanajuato, Gto., del 08 al 10 de Octubre de 2008.

6. Publicación del articulo “Calibración de un Algoritmo Genético para el Problema de la Minimización de Cruces en las Aristas de un Grafo”, Autores: Sergio Enríquez Aranda, Eunice Ponce de León Sentí y Elva Díaz Díaz, publicado en Avances en Computación Evolutiva, memorias del Cuarto Congreso Mexicano de Computación Evolutiva del Centro de Investigación en Matemáticas A. C.

7. Participación como “Ponente” en el Cuarto Congreso Estatal la Investigación en el Posgrado de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., el 17 y 18 de Octubre de 2008.

8. Publicación del articulo “Sistema MinX para el Dibujado Automático de Grafos con el Menor Número de Cruces en sus Aristas”, Autores: Sergio Enríquez Aranda, Eunice Ponce de León Sentí, Elva Díaz Díaz y Alejandro Padilla Díaz, publicado en la Memoria del Cuarto Congreso Estatal la Investigación en el Posgrado de la Universidad Autónoma de Aguascalientes.

9. Reconocimiento como “Mejor Ponencia” en el Cuarto Congreso Estatal la Investigación en el Posgrado de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., el 17 y 18 de Octubre de 2008.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

10. Participación como “Ponente” en la conferencia Modelado de un AG y un EDA de la Universidad Autónoma de Aguascalientes, llevado a cabo en Aguascalientes, Ags., 29 de Octubre de 2008.

11. Participación como “Ponente” en el Seminario/ Taller Data Mining de la Universidad Autónoma de Aguascalientes y La École Technologie Supéreur, llevado a cabo en Aguascalientes, Ags., del 25 al 28 de Febrero de 2009.

A continuación se presentan las aportaciones de la presente tesis:

Como aportación se presenta un algoritmo hibrido llamado HUx, el cual toma las ventajas de exploración del espacio de búsqueda del problema de optimización de los algoritmos AEDMU y el Escalador de Colinas. La forma de hibridación se realizó de la siguiente manera: se le aplica un Escalador de Colinas a la mejor solución obtenida en cada iteración del algoritmo AEDMU y la solución resultante se sustituye por la peor solución de la población obtenida en esa iteración. Los resultados de esta hibridación supero a todos los algoritmos analizados en esta tesis que han sido reportados como estado del arte en la literatura.

La otra aportación de esta tesis es un sistema programado en Java llamado MinX que sirve para dibujar grafos en forma automática con el número mínimo de cruces en sus aristas. Este es un sistema con una interfase gráfica amigable al usuario. Las características y operaciones principales del sistema se describen a continuación:

Características del sistema:

- Es un sistema amigable ya que utiliza una interfase grafica de usuario.
- Se visualiza el grafo de la solución óptima.
- El sistema utiliza dos ventanas para mostrar tanto los resultado obtenidos, como el dibujado final de los grafos
- Los resultados se visualizan en una tabla de resultados.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- En la tabla de resultados se muestran los resultados de todos los grafos junto con las coordenadas obtenidas de cada grafo (solución).
- El diseño final del dibujado de cada grafo se muestra en una ventana independiente.
- Se puede conocer visualmente la minimización de cruces obtenida de cada solución por medio del dibujado automático de los grafos.
- La minimización de cruces se puede lograr por medio de cinco algoritmos, a saber, AGS, AEDMU clásico, EC con intercambio de vértices, EC con reinicio aleatorio y el algoritmo híbrido HUx.

Operaciones del sistema:

- Para la minimización del número de cruces de las aristas del grafo, se puede seleccionar el tipo de algoritmo a utilizar, de un total de cinco posibles.
- En la ventana que contiene la tabla de resultados, se pueden ordenar los campos de información en orden descendente, con el propósito de conocer cual es la mejor solución obtenida por el sistema MinX.
- La tabla de resultados cuenta con una lista desplegable desde donde se puede seleccionar el grafo deseado y mostrarlo en una ventana independiente.
- Se puede hacer una comparación del diseño de varios grafos al mismo tiempo ya que se muestran en ventanas independientes.

1.7 Contenido de la tesis

Capítulo 1. INTRODUCCIÓN: En este capítulo se muestra, el contenido de la tesis en forma general, comenzando con una introducción al planteamiento del problema del dibujado automático de grafos en la minimización del número de cruces de las aristas de un

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

grafo, así como la justificación, los objetivos y metas de la presente investigación. Al final de este capítulo se presentan las contribuciones de la tesis y las publicaciones realizadas en el período de estudios de la maestría.

Capítulo 2. MARCO TEÓRICO: En este capítulo se revisan los fundamentos teóricos que son la base del presente trabajo de tesis. El contenido de este capítulo abarca diferentes temas relacionados con áreas del conocimiento, como son: la teoría de grafos, la geometría analítica, inteligencia artificial y el dibujado automático de grafos. Los fundamentos teóricos de estos temas son revisados en este capítulo.

Capítulo 3. ALGORITMOS PARA EL DIBUJADO DE GRAFOS: En este capítulo se presenta el pseudocódigo, y las estructuras de datos implementadas para los diferentes algoritmos utilizados en esta tesis, en el problema del dibujado de grafos, específicamente, en la minimización de cruces de las aristas de un grafo.

Capítulo 4. SISTEMA MinX: En este capítulo se hace la presentación del Sistema MinX. Aquí se describe el diagrama general del sistema, los elementos involucrados en su funcionamiento, las características principales y la interfase visual utilizada para mostrar los resultados obtenidos por las diferentes metaheurísticas utilizadas en esta tesis. Al final se comenta la importancia que ha tenido la interfase visual del Sistema MinX en este trabajo de tesis.

Capítulo 5. SISTEMA MinX: DISEÑO DE EXPERIMENTOS Y ANALISIS DE RESULTADOS: Este capítulo muestra como fueron diseñados los diferentes experimentos realizados a todo el grupo de algoritmos implementados en este trabajo de tesis. También, se hace un análisis de los resultados obtenidos con diferentes grafos utilizados, que son considerados parte del estado del arte.

Capítulo 6. CONCLUSIONES: En este capítulo se explica en forma general los resultados obtenidos en esta tesis, la importancia de los resultados y las aportaciones generadas por esta investigación, Así mismo, se describe el trabajo futuro y hacia donde puede dirigirse tomando como base este trabajo de investigación.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 2. MARCO TEÓRICO

En este capítulo se revisan los fundamentos teóricos que son la base del presente trabajo de tesis, estos abarcan varios temas de diferentes áreas del conocimiento, como son la teoría de grafos, geometría analítica, e inteligencia artificial.

2.1 Teoría de grafos

La teoría de grafos (teoría de las gráficas) se ubica dentro del campo de las matemáticas y ciencias de la computación, y se encarga de estudiar las propiedades de los grafos (gráficas).

2.1.1. Grafos

La clasificación de los grafos en [Sugiyama (2002)], están divididos en: *árboles*, *grafos dirigidos*, *grafos no dirigidos* y *grafos compuestos*. Los árboles están divididos en árboles con raíces (*rooted trees*) y árboles libres sin raíces. Todos los árboles son planares, esto quiere decir, que se pueden dibujar sin cruces en sus aristas. Los árboles con raíces representan estructuras jerárquicas como diagramas de organización, por otro lado los árboles libres no tienen especificada una raíz y por lo tanto tiene un alto grado de libertad en su colocación.

Los grafos dirigidos están divididos en *grafos dirigidos acíclicos* y *grafos generales dirigidos*. Los grafos dirigidos acíclicos representan una jerarquía, por lo que, tienen un flujo de dirección fija hacia sus ejes. En los grafos generales dirigidos, las partes cíclicas y acíclicas tiene características diferentes y pueden ser dibujados de diferentes maneras.

Los grafos no dirigidos son los objetos más comunes de dibujado y están divididos en *grafos planares no didrigidos* (o simplemete *grafos planares*) y *grafos generales no dirigidos*. Los grafos planares son la clase más importante de grafos, por lo que, han sido muy estudiados en teoría de grafos, algoritmos de grafos y dibujado de grafos.

La figura 2.1 muestra la clasificación de los grafos según [Sugiyama (2002)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

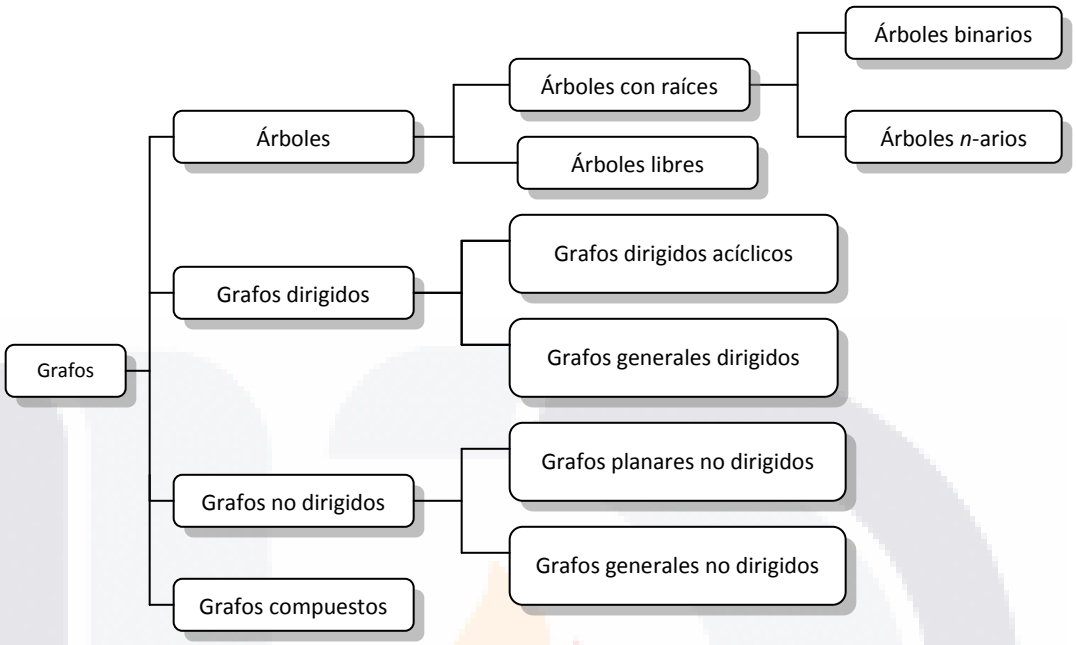


Figura 2.1. Clasificación general de los grafos.

2.1.1.1. Grafos generales no dirigidos

Un *grafo* general (*grafo simple o grafo no dirigido*) es un par $G = (V, E)$ de conjuntos tal que $E \subseteq [V]^2$; así, los elementos de E son subconjuntos de dos elementos de V, donde $V \cap E = 0$. Los elementos de V son los *vértices* (nodos, o puntos) del grafo G, los elementos de E son sus *aristas* (o líneas).

La forma usual de dibujar un grafo es dibujando un punto por cada vértice y uniendo dos de esos puntos por una línea si los dos vértices correspondientes forman una arista. La manera como esos puntos y líneas son dibujados es irrelevante; todo lo que importa es la información de cuales pares de vértices forman una arista y cuales no [Diestel (2005)]. El conjunto de vértices de un grafo G es referenciado como $V(G)$, sus aristas como $E(G)$.

La figura 2.2 muestra el ejemplo de un grafo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

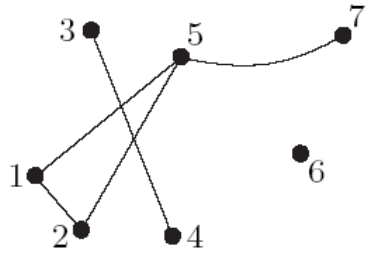


Figura 2.2. El grafo de V = {1,…,7} con un conjunto
E = {{1,2},{1,5},{2,5},{3,4},{5,7}}

Orden: el número de vértices de un grafo *G* es su *orden*, escrito como |*G*|; su número de aristas es indicado por ||*G*||. Los grafos son finitos, infinitos, contables y así sucesivamente de acuerdo a su orden. En esta tesis solo se trabaja con grafos finitos.

Incidencia: un vértice *v* es *incidente* con una arista *e* si $v \in e$; entonces *e* es una arista de *v*. Los dos vértices que inciden con una arista son sus vértices finales o *ends*, y una arista une sus *ends*. Una arista $\{x, y\}$ es normalmente escrita como *xy* (o *yx*). Si $x \in X$ y $y \in Y$, entonces *xy* es una arista *X*-*Y*. El conjunto de todas las aristas *X*-*Y* en un conjunto *E* es denotado por *E(X-Y)*. El conjunto de todas las aristas en *E* de un vértice *v* es denotado por *E*(*v*).

Adyacencia: dos vértices *x*, *y* de *G* son *adyacentes*, *o vecinos*, si *xy* están en una arista de *G*. Dos aristas $e \neq f$ son *adyacentes*, si ellas tienen un *end* en común.

Completo: si todos los vértices de *G* son pares adyacentes, entonces *G* es *completo*. Un grafo completo de *n* vértices es un K_n, un K_3 es llamado un triángulo. La figura 2.3 muestra un grafo completo K_3.

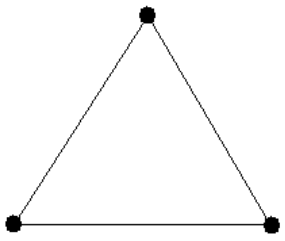


Figura 2.3. Grafo completo K_3 (triángulo).

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Grado de un vértice: si $G = (V, E)$ es un grafo (no vacío), el conjunto de vecinos de un vértice v en G es indicado por $N_G(v)$, o brevemente por $N(v)$. El *grado* (*o valencia*) $d_G(v) = d(v)$ de un vértice v es el número $|E(v)|$ de aristas en v, o este es igual al número de vecinos de v. Si todos los vértices de G tienen el mismo grado k, entonces G es *k-regular*, o simplemente *regular*. *El número de vértices de grado impar en un grafo es siempre par*.

Caminos: un *camino* es un grafo no vacío $P = (V, E)$ de la forma

$$V = \{x_0, x_1, \ldots, x_k\} \qquad E = \{x_0x_1, x_1x_2, \ldots, x_{k-1}x_k\}$$

Donde x_i son todos distintos. Los vértices x_0 y x_k están unidos por P y son llamados sus *ends*; los vértices $x_1, \ldots, x_{k-1}$ son los vértices internos de P.

Longitud: el número de aristas de un camino es su *longitud*, y la longitud del camino k es indicado por P_k. Se permite que k sea cero, por lo que $P_0 = K_1$. A menudo nos referimos a un camino por la secuencia natural de sus vértices, entonces se dice que, $P = x_0x_1 \ldots x_k$ y llamando a P un camino desde x_0 hasta x_k (así como también *entre* x_0 y x_k). La figura 2.4 muestra el ejemplo de un camino $P = P_6$.

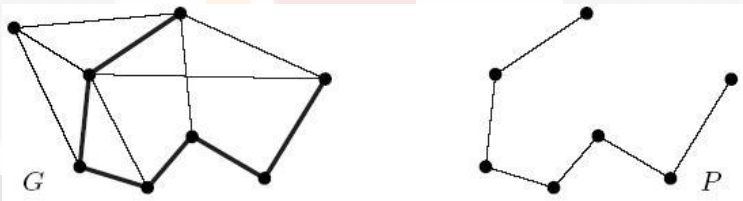


Figura 2.4. Un camino $P = P_6$ en G.

Conectividad: un grafo G no vacío se le llama conectado si cualquiera de dos de sus vértices están unidos por un camino en G. si $U \subseteq V(G)$ y $G[U]$ está conectado, se llama a U por si mismo conectado (en G). Lo contrario sería 'no conectado' o desconectado. *Los vértices de un grafo conectado G pueden ser siempre numerados, como $v_1, \ldots, v_n$ así que $G_i := G[v_1, \ldots, v_n]$ está conectado por cada i* [Diestel (2005)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.1.1.2. Grafos dirigidos

Un *grafo dirigido* (o dígrafo) *es una tripla D = (V, E, ϕ) donde* V *y* E *son conjuntos finitos* y *ϕ es una función con dominio en* E *y codominio V* x *V. E es el conjunto de aristas del dígrafo D y V el conjunto de vértices de D* [Bender y Williamson (2005)]. En la presente tesis se trabajo solo con grafos no dirigidos o simples. La figura 2.5 muestra el ejemplo de un dígrafo.

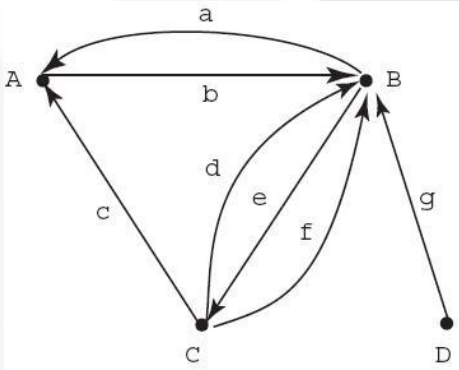


Figura 2.5. Ejemplo de un dígrafo.

La figura 2.4 se puede especificar al dar un par *D = (V, E, ϕ)* donde *ϕ* es su función. El dominio de la función *ϕ* es E = {a, b, c, d, e, f, g} y el codominio es *V* x *V*. Específicamente,

$$\phi = \left\{ (a, A), (b, B), (c, A), (d, B), (e, C), (f, B), (g, B) \right\}.$$

2.1.1.3. Subgrafos

Dado que *G = (V, E, ϕ)* es un grafo. Un grafo *G′ = (V′, E′, ϕ′)* es un subgrafo de G si *V′*⊆ V, *E′* ⊆ *E*, y *ϕ′* es la restricción de *ϕ* en *E′*. El hecho de que *G′* sea en si mismo un grafo significa que *ϕ(x)* ∈ $\mathcal{P}_2(V')$ por cada *x* que pertenece a *E′* y, el codominio de *ϕ′* debe ser $\mathcal{P}_2(V')$. La notación $\mathcal{P}_k(V)$ agrupa al conjunto de todos los subconjuntos de *k*-elementos del conjunto V [Bender y Williamson (2005)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Para el grafo $G = (V, E, \phi)$ de la figura 2.5, dado $G' = (V', E', \phi')$ se define por $V' = \{A, B, C\}$, $E' = \{a, b, c, d, e, f\}$, y ϕ' es la restricción ϕ en E', ϕ con codominio en $\mathcal{P}_2(V')$, ϕ Es determinado completamente por V', E' y ϕ. Así, para especificar un subgrafo G' la información clave son V', E'. Otro ejemplo del mismo grafo sería, dado $V = V'$ y $E' = \{a, b, c, f\}$. En este caso, el vértice D no es miembro de alguna arista del subgrafo. Dicho vértice es llamado vértice aislado de G'. Otra manera de especificar un subgrafo es dar un conjunto de aristas $E' \subseteq E$ y tomar V' que sean el conjunto de todos los vértices sobre alguna arista de E'. En otras palabras, V' es la unión del conjuntos de $\phi(x)$ sobre todos $x \in E'$. Dicho subgrafo es llamado *subgrafo inducido por la arista puesta en* E' o *el subgrafo de arista inducida de* E'. El primer subgrafo de este ejemplo es el subgrafo inducido por $E' = \{a, b, c, f\}$. Además, dado un conjunto $V = V'$, podemos tomas a E' para que sea el conjunto de todas las aristas $x \in E'$, tal que $\phi(x) \subseteq V'$. El subgrafo resultantes es llamado subgrafo inducido por V' o el subgrafo de vértice inducido de V'.

En la figura 2.6 las aristas del subgrafo inducido por $V' = \{C, B\}$, son $E' = \{d, e, f\}$.

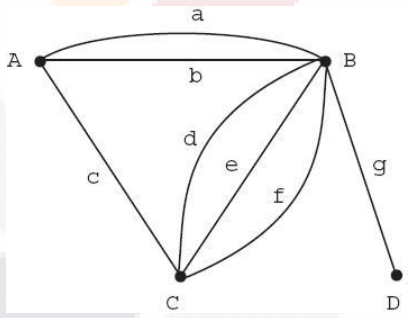


Figura 2.6. Ejemplo de un subgrafo.

2.1.1.4. Grafo planar

Si dibujamos un grafo sobre una hoja de papel, tratando de que se vea claro, dibujaremos sus líneas evitando que estas se crucen. Esto significa que tendremos que dibujar el grafo de tal manera que dos de sus aristas no se reúnan en un punto diferente a un end común [Diestel (2005)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Se dice que un grafo puede ser incrustado en el plano, o planar, si este puede ser dibujado en el plano así que sus aristas se intersectan solo en sus ends. Tal que al dibujado de un grafo planar G es llamado un incrustado planar de G. Un incrustado planar G' de G puede ser considerado como un grafo isomorfo [Diestel (2005)] de G; El conjunto de vértices de G' son el conjunto de puntos que representan los vértices de G; El conjunto de aristas de G' son el conjunto de líneas que representan las aristas de G, y un vértice de G' es incidente con todas las aristas de G' que este contiene [Bondy y Murty U. S. R. (1982)].

Un grafo plano es un par (*V*, *E*) de conjuntos finitos con las siguientes propiedades (los elementos de *V* son llamados otra vez *vértices*, los de *E* aristas):

(i) $V \subseteq \mathbb{R}^2$ (conjunto de todas las parejas ordenadas de números $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$);

(ii) cada arista es un arco entra dos vértices;

(iii) diferentes aristas tiene diferentes conjuntos de endpoints;

(iv) el interior de una arista no contiene vértices, ni tampoco puntos de ninguna otra arista.

Un grafo plano no contiene ningún grafo K5 (grafo completo de 5 vértices) ni tampoco un grafo K3,3 (grafo completo bipartito [Diestel (2005)] de 6 vértices) como una topología menor.

Un grafo es llamado *planar* si este puede ser incrustado en el plano: si este es isomorfo al grafo plano. Un grafo planar es maximal, o maximalmente planar, si este es planar pero no puede ser extendido a un grafo planar más grande al agregarle una arista (no un vértice).

(i) Cada grafo plano maximal es maximalmente planar.

(ii) Un grafo planar con $n \geq 3$ vértices es maximalmente planar si y solo si este tiene $3n - 6$ aristas.

Un grafo es planar si y solo si no contiene un subgrafo con una estructura analógica (*conocida como menor*) a K_5 o $K_{3,3}$.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La figura 2.7 muestra el ejemplo de un grafo planar.

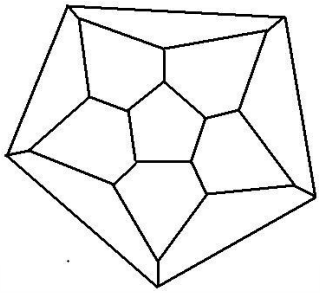


Figura 2.7. Ejemplo de un grafo planar.

2.1.2. Minimización de cruces

El *número de cruces* en un dibujado $D(G)$ es el número de cruces $\text{CR}(D(G))$ del dibujado $D(G)$. El número de cruces de un grafo abstracto G, denotado por $\text{CR}(G)$, es el número más pequeño de cruces de cualquier dibujado de G [Cerný (2008)]. Los primeros trabajos sobre el numero de cruces viene de la década de los años 50's [Kovári et. al. (1954)]. El gran avance en aplicaciones de este problema viene del teorema *Semerédi-Trotter* [Szemerédi et. al., (1982)], utilizado para la incidencia de puntos y líneas, sobre límites de distancias en el plano y en 1997 se creo método para el número de cruces [Székely (1997)].

Una propiedad extraordinaria del problema de la minimización de cruces es su dificultad. Esto fue demostrado por Garey y Johnson en "Crossing number is NP-completo" [Garey y Johnson (1983)] y demostraron que este problema es NP-duro. Diferentes problemas de optimización han surgido en el área del dibujado automático de grafos que son NP-duro y que jamás han sido resueltos en la práctica. Así mismo, la minimización de cruces es extremadamente difícil también en la práctica. Hasta ahora, aproximaciones exactas han fallado al resolver instancias grandes en un tiempo de corrida razonable. Esto es fuertemente demostrado, por el hecho de que aun los números de cruces de los grafos completos Kn son desconocidos para $n \geq 11$ [Buchheim et. al. (2004)].

El dibujado de un grafo simple no dirigido es un mapeo de f que asigna a cada vértice un punto distinto en el plano y a cada arista uv un arco conectando $f(u)$ y $f(v)$, no pasando a

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

través de la imagen de algún otro vértice. El arco asignado a *uv* es llamado una arista del dibujado y se denota por *uv*. Se asume que tres aristas no tienen un punto interior en común, y si dos aristas comparten un punto interior p, entonces ellas se cruzan en el punto p. Un punto final en común de dos aristas no cuenta como un cruce [Pach y Tóth (2000)].

La principal información que un grafo abstracto nos puede dar es si dos vértices están unidos por una arista o no. Esta información se complica con la presencia de aristas cruzadas ya que distraen la concentración de la visión humana. La figura 2.8 muestra la comparación de tres grafos con el mismo número de vértices y las mismas aristas conectadas [Buchheim et. al. (2004)].

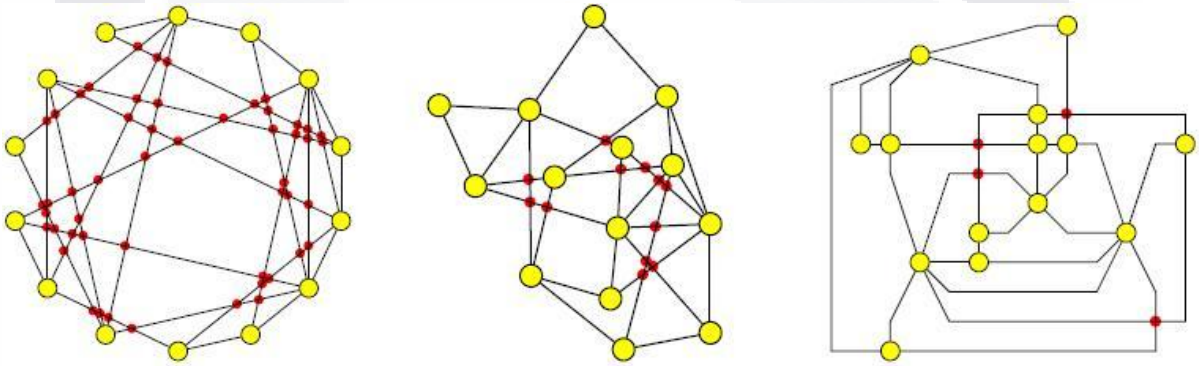


Figura 2.8. Diferentes dibujados de un grafo abstracto con diferentes números de aristas cruzadas (51,12 y cuatro respectivamente. Criterios estéticos como pocos dobleces, longitudes de arista uniforme, o área pequeña de dibujado, están a favor de los dos primeros dibujados. Sin embargo, con respecto al número de mínimo de aristas cruzadas el último es mejor.

Sea $G = (V, E)$ un grafo, V conjunto de vértices y E conjunto de aristas, sea M_G la matriz de adyacencias, P el plano cartesiano y (Cx) el número de cruces. El problema consiste en encontrar para cada $v \in V$ un par $(x, y) \in P$, tal que la cantidad de Cx entre las aristas del grafo sea mínima, con la restricción de que cualquier par de vértices $v = (x, y)$ y $v' = (x', y')$, $x \neq x'$ y $y \neq y'$ [Enríquez et. al. (2008)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.2 Geometría analítica

La geometría Analítica o llamada también Geografía Matemática, es la ciencia que combina el álgebra y la geometría, para describir figuras geométricas planas desde el punto de vista Algebraico y Geométrico. Dado una gráfica específica, se debe encontrar una ecuación y hacer que la gráfica explique dicha ecuación.

2.2.1. El espacio Euclidiano

La geometría analítica tiene sus inicios en la geometría griega a partir de Euclides de Alejandría en el año 300 AC, lo que hoy se conoce como Geometría Euclidiana, la cual se basa en cinco *postulados o axiomas* (principios básicos) que son:

I. Por cualesquiera dos puntos, se puede trazar el segmento de recta que los une.

II. Dados un punto y una distancia, se puede trazar el círculo de centro el punto y radio la distancia.

III. Un segmento de recta, se puede extender en ambas direcciones indefinidamente.

IV. Todos los ángulos rectos son iguales.

V. Dadas dos rectas y una tercera que las corta, si los ángulos internos de un lado suman menos de dos ángulos rectos, entonces las dos rectas se cortan y lo hacen de ese lado.

Estos postulados describen el comportamiento y relación entre ciertos elementos básicos de la geometría como “punto”, “trazar”, “segmento” y “distancia”, entre otros. A mediados del siglo XVII, Descartes marca la revolución en las matemáticas, introduciendo el algebra a la solución de problemas de índole geométrico. Descartes tomó el quinto postulado de Euclides y lo transformó, aportando dos nuevos postulados que son equivalentes al original.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

V.a Dada una línea recta y un punto fuere de ella, existe una única recta que pasa por el punto y que es paralele a la línea.

V.b Los ángulos interiores de un triángulo suman dos ángulos rectos.

Descartes hace un razonamiento sobre el plano Euclidiano, que son los cinco axiomas y puntos que se extienden indefinidamente sobre este plano. Se denota $\mathbb{E}^2$ a este plano, donde $\mathbb{E}$ es el espacio Euclidiano, 2 se refiere a la dimensión 2 y no al exponente 2. Los puntos de una recta $\ell \subset \mathbb{E}^2$ corresponde biunivocamente a los números reales, que se denotan por $\mathbb{R}$. Se llama origen a un punto $\mathbf{o} \subset \ell$ y que corresponde al número cero. El origen parte a la recta en dos mitades; a una de ellas se le asocian los números positivos y a la otra mitad los números negativos. Así, a cada número real *x* se le asocia un punto **p** en ℓ, y a cada punto en ℓ se le asocia un número real. A esta identificación se le da el nombre de "recta de los números reales" en el cual se basan todo el desarrollo de la geometría analítica y el cálculo [Bracho (2003)]. La figura 2.9 muestra un ejemplo de recta con origen y punto en ℓ.

Figura 2.9. Ejemplo de una recta.

2.2.2. Puntos del espacio Euclidiano

Para identificar puntos del espacio euclidiano con parejas de número se utiliza el quinto postulado. A cualquier punto $\mathbf{p} \in \mathbb{E}^2$ puede corresponder con una pareja de números de la siguiente forma: Por el Quinto, existe una única recta ℓ_1' que pasa por **p** y es paralela a ℓ_1; así mismo, existe una única recta ℓ_2' que pasa por el punto **p** y es paralela a ℓ_2. Las intersecciones $\ell_1 \cap \ell_2'$ y $\ell_2 \cap \ell_1'$ determinan los puntos $\mathbf{p}_1 \in \ell_1$ y $\mathbf{p}_2 \in \ell_2$, respectivamente, y por lo tanto, determinan dos números reales *x* y *y*; es decir una pareja ordenada (*x*, *y*). Dada la pareja de números (*x*, *y*), tomando a $\mathbf{p}_1$ como el punto que esta

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

sobre ℓ_1 a distancia x de **o**, y $\mathbf{p}_2$ como el punto que está sobre ℓ_2 a distancia y de **o**. Sea ℓ'_1 la recta que pasa por $\mathbf{p}_2$ paralela a ℓ_1 y sea ℓ'_2 la recta que pasa por ℓ_2. La intersección $\ell'_1 \cap \ell'_2$ es el punto que corresponde a la pareja (*x*, *y*) [Bracho (2003)]. La figura 2.10 muestra la intersección entre $\ell'_1 \cap \ell'_2$ que corresponden al punto (*x*, *y*).

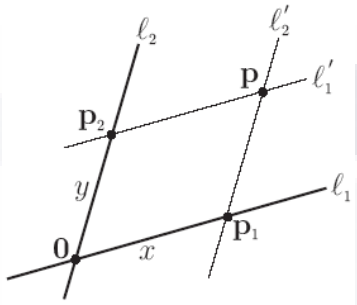


Figura 2.10. Intersección entre $\ell'_1 \cap \ell'_2$ que corresponde al punto p(x, y).

En el razonamiento de Descartes, conviene tomar las rectas ℓ_1 y ℓ_2 en forma ortogonal; a ℓ_1 horizontalmente con dirección positiva a la derecha y a ℓ_2 verticalmente con dirección positiva hacia arriba, conocidas como eje de las *x* y eje de las *y*, respectivamente. Esto simplifica el álgebra asociada a la geometría (conocido como *plano cartesiano* en honor a Descartes) [Bracho (2003)]. La figura 2.11 muestra las coordenadas del plano cartesiano.

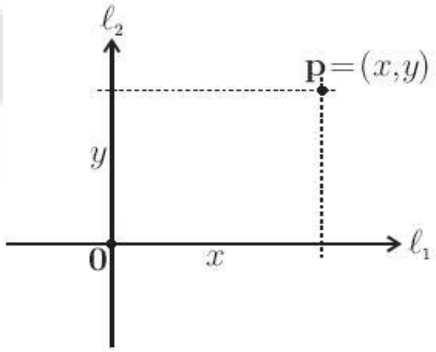


Figura 2.11. Plano Cartesiano.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Al fijar los ejes coordenados, a cada pareja de números (x, y) le corresponde un punto **p** $\in \mathbb{E}^2$, que se identifica como **p** = (x, y). Además, le puede corresponder un segmento de recta dirigido llamado *vector* (identificado por una flecha). El conjunto de todas las parejas ordenadas de números se denota por $\mathbb{R}^2 = \mathbb{R} \times \mathbb{R}$ [Bracho (2003)]. La figura 2.12 muestra un punto en el plano, una pareja de números reales y un vector que va del origen al punto.

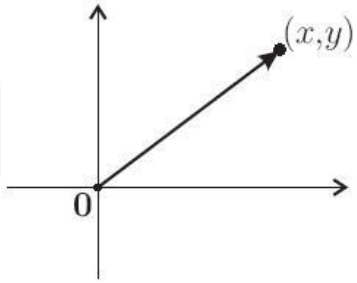


Figura 2.12. Interpretaciones equivalentes de punto en el plano, una pareja de números reales y un vector.

Si las parejas de números reales se identifican naturalmente con los puntos del plano euclidiano $\mathbb{R}^2 = \mathbb{E}^2$, se logra que converjan las áreas del álgebra y la geometría.

2.2.3. Distancia entre dos puntos

La distancia siempre ha sido un concepto de gran trascendencia por su utilidad. Euclides fue quien dio las bases para calcular la distancia entre dos puntos. La distancia entre dos puntos puede ser definida con la ayuda del teorema de Pitágoras [Rondón (2006)]. La figura 2.13 muestra la distancia entre dos puntos utilizando el teorema de Pitágoras.

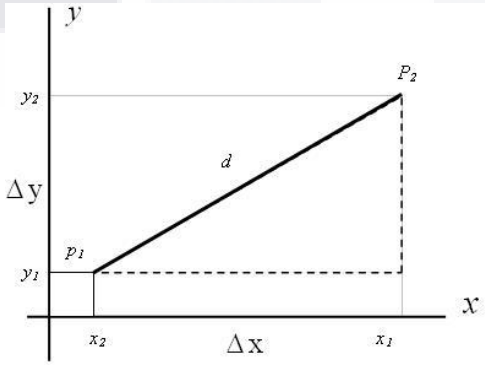


Figura 2.13. Distancia entre dos puntos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Dado:

$$d^2 = (\Delta x)^2 + (\Delta y)^2$$

donde:

$$\Delta x = x_2 - x_1$$

$$\Delta y = y_2 - y_1$$

Entonces:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

2.2.4. La recta

Una recta, es una línea de puntos colineales. Es decir, puntos ubicados uno tras otro de tal manera que uno esconde al anterior al mirar la fila de frente. La información analítica que una recta nos puede dar, se centra en que a partir de la gráfica, se obtenga su ecuación o viceversa.

Características: toda recta tiene una gráfica, una ecuación que la distingue, además, de sus parámetros conocidos como pendiente e intercepto.

Pendiente: se simboliza por la letra m, y está relacionada con la inclinación que la recta presenta con respecto al eje de x (abscisa).

Intercepto: se simboliza por la letra b, y está relacionada con el punto donde la recta corta al eje y (ordenada).

Ecuación de la recta: es la forma de expresar matemáticamente una recta. Esta tiene dos ecuaciones conocidas como canónica (o analítica) y la segunda como ecuación general de la recta.

Ecuación canónica: $y = m\ x + b$

Ecuación general: $a\ x + b\ y + c = 0$

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La primera ecuación corresponde a la función lineal donde la pendiente y el intercepto se pueden ver explícitamente [Rondón (2006)].

2.2.4.1. Pendiente de la recta

Para graficar o determinar una recta, sólo basta con dos puntos, esto permite calcular la pendiente de la recta. La pendiente se define determinado los desplazamientos que existen tanto en x como en y. La figura 2.14 muestra la pendiente de una recta, dada por los puntos $p_1(x_1, y_1)$ y $p_2(x_2, y_2)$.

$$m = \frac{\Delta x}{\Delta y}$$

Δx = desplazamiento en x

Δy = desplazamiento en y.

Pero: $\Delta x = x_2 - x_1$ y $\Delta y = y_2 - y_1$

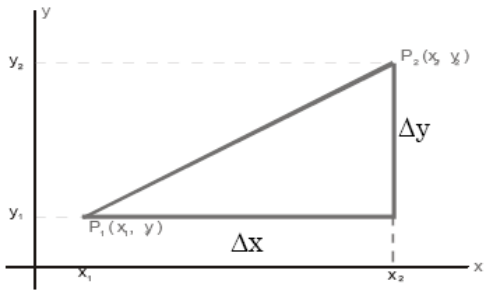


Figura 2.14. Pendiente de una recta.

La fórmula para hallar la pendiente de una recta se puede obtener conociendo dos puntos de la recta [Rondón (2006)].

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

para: $p_1(x_1, y_1)$ y $p_2(x_2, y_2)$

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La recta puede presenta cuatro comportamientos, dependiendo del valor de su pendiente:

1. **m > 0:** la recta presenta una inclinación a la derecha, es decir, el ángulo es agudo: $0^{\circ} < \sigma < 90^{\circ}$
2. **m < 0:** la recta presenta una inclinación a la izquierda, es decir, el ángulo es obtuso: $90^{\circ} < \sigma < 180^{\circ}$
3. **m = 0:** la recta es horizontal, el ángulo es $\sigma = 0$
4. **m = indeterminado (α):** la recta es vertical, el ángulo es $\sigma = 90^{\circ}$.

En la figura 2.15 se observa que la recta presenta una inclinación a la derecha, m > 0.

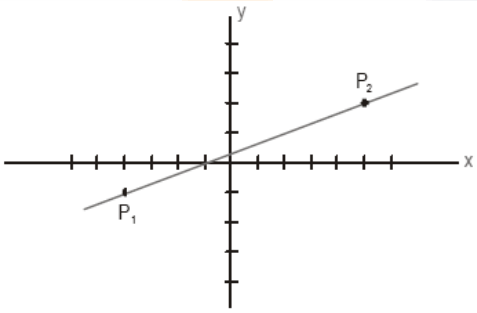


Figura 2.15. Pendiente con m > 0.

En la figura 2.16 se observa que la recta presenta una inclinación a la izquierda, m < 0.

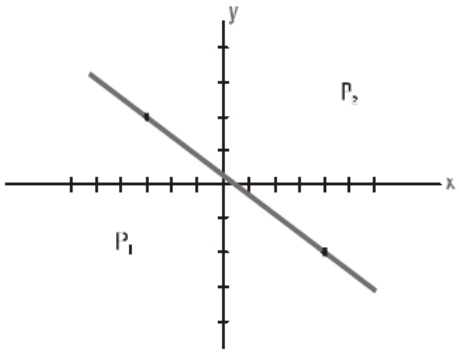


Figura 2.16. Pendiente con m < 0.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.2.4.2. Intercepto

Es el punto donde la recta corta al eje y (ordenada). Cuando m = α, la recta no tiene intercepto, pero para los otros casos si. El intercepto corresponde al valor de b, en la ecuación canónica. Para la ecuación general hay que despejar y, para conocer el intercepto. A partir de la ecuación canónica, se puede llegar a la ecuación general, igualando toda la ecuación a cero. Por otro lado, a partir de la general, se puede obtener la canónica, despejando y [Rondón (2006)].

2.2.4.3. Intersección de rectas

Ecuación punto y pendiente de la recta: Como se explico anteriormente la ecuación general de la recta se puede escribir de diferentes maneras que son equivalentes. Una de ellas se puede obtener a partir de algunas propiedades geométricas de la recta, que se analizan a continuación [Wooton et. al. (1972)].

La figura 2.17 muestra a la recta ℓ que pasa por un punto dado S($x1$, $y1$). Si U(x, y) es cualquier otro punto de ℓ, entonces un vector de dirección de ℓ es

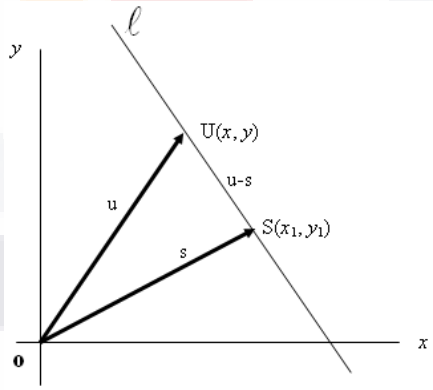


Figura 2.17. Punto y pendiente de la recta ℓ.

$$u - s = (x, y) - (x1, y1)$$

$$u - s = (x - x1, y - y1)$$

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Como se vio anteriormente si $x \neq x_1$, entonces la pendiente m de ℓ esta dada por

$$\frac{y - y_1}{x - x_1} = m$$

donde se tiene:

$$y - y_1 = m(x - x_1)$$

Esta ecuación de ℓ es la *ecuación punto y pendiente de la recta*, la cual sirve para obtener la ecuación de la recta que pasa por dos puntos dados.

Ecuación que pasa por dos puntos dados: Si la recta contiene dos puntos S(x_1, y_1) y T(x_2, y_2), con $x_2 \neq x_1$, entonces la pendiente m de ℓ esta dada por

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

Si la ecuación de m se sustituye en la ecuación punto y pendiente de la recta, se obtiene una ecuación equivalente que pasa por dos puntos dados.

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

> *En conjuntos la intersección entre A y B se denota por A ∩ B, se define como el conjunto de todos los elementos comunes a A y B. Una propiedad de los conjuntos de puntos llamados rectas es que, en el plano, la intersección de dos rectas distintas es o bien el conjunto vacío (si las rectas son paralelas) o bien tienen un elemento (un solo punto si las rectas no son paralelas). Si ℓ_1 y ℓ_2 son la misma recta, entonces ℓ_1 y ℓ_2 son paralelas y su intersección es toda la recta* [Wooton et. al. (1972)].

Sistema de ecuaciones lineales con dos incógnitas: Existe una gran cantidad de problemas que pueden ser resueltos por medio del *sistema de dos ecuaciones lineales con dos incógnitas*, como en los negocios, la economía y hasta en nuestra vida diaria [Prieto

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

(2003)]. En el presente trabajo de tesis se hace uso de este sistema, para probar si dos rectas tienen intersección o no, lo cual es considerado como un cruce de aristas en un grafo.

Si se definen dos rectas en el plano mediante un sistema de dos ecuaciones lineales con dos incógnitas, el punto de intersección entre ellas se puede calcular resolviendo el sistema de ecuaciones simultáneamente. El sistema de ecuaciones lineales con dos incógnitas puede ser resuelto por *sustitución*, *eliminación de una incógnita*, o por el *método vectorial*. En la presente tesis se hace uso del método de sustitución [Wooton et. al. (1972)].

El sistema de ecuaciones con dos incógnitas y dos variables *x*, e *y* cuenta con dos ecuaciones del tipo [Prieto (2003)]:

$$a_1 x + b_1 y = c_1$$

$$a2\, x + b2\, y = c2$$

Se tiene el siguiente sistema:

$$3x - 5y = 1 \qquad (1)$$

$$4x + 3y = 11 \qquad (2)$$

resolver por sustitución.

se despeja *y* de ecuación (1) en términos de *x*

$$y = \frac{3}{5}x - \frac{1}{5} \qquad (3)$$

se sustituye el valor de *y* en la ecuación (2) y se despeja x

$$4x + 3\left(\frac{3}{5}x - \frac{1}{5}\right) = 11$$

$$x = 2$$

se sustituye el valor de *x* en ecuación (3) y se despeja *y*

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

$$y = \frac{3}{5}(2) - \frac{1}{5}$$

$$y = 1$$

la solución única del sistema de ecuaciones es el punto (2, 1). Ahora, se sustituyen los valores obtenidos de *x* y *y* en las ecuaciones originales y se comprueba que sean iguales, si es así, se considera que el sistema de ecuaciones está resuelto y existe intersección entre las dos rectas.

comprobación:

$3(2) - 5(1) = 1$ (1)	$4(2) + 3(1) = 11$ (2)
$1 = 1$	$11 = 11$

2.3 Dibujado automático de grafos

El dibujado de grafos es el arte de dibujar un grafo de tal manera que las relaciones entre sus objetos sean fácilmente entendidas con sólo mirar el dibujo. El objetivo en el *dibujado automático de grafos* es: dibujar de forma automática (con el uso de computadoras) un grafo, con el uso de métodos heurísticos o técnicas innovadoras, para transmitir el significado de los objetos modelados, de la manera más legible (tanto como sea posible).

2.3.1. Historia del dibujado de grafos

El origen histórico del dibujado de grafos es sumamente difícil de precisar, Euler hizo algunos trabajos que involucraban grafos, así como, diferentes personas en los siglos XVII y XVIII. El dibujado de grafos se vio plasmado también, en las raíces de la moderna teoría de grafos de ese tiempo.

Se sabe que el uso del dibujado de grafos, inició alrededor del 1800, al surgir la necesidad de visualizar algunos objetos de la vida real. Uno de los primeros usos del dibujado de grafos fue, en el área de la química. En ese tiempo, se dibujaban estructuras

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

cristalinas y moleculares, muy similares a las usadas hoy en día [Kurja et. al. (2001)]. Ya en el siglo XX, la vanguardia en las investigaciones sobre los métodos de dibujado de grafos, data desde la década de los años 30s, sin embargo, no fue sino hasta la década de 1980, con la aparición de los avances tecnológicos en materia de computación, que el dibujado de grafos tomó auge desde entonces y hasta nuestros días.

Muchos de los grandes grafos son imposibles de dibujar manualmente, lo que limita la investigación de esta clase de grafos con cierto tipo de estructuras. Es por eso, que surge la necesidad de dibujar los grafos, por medio del uso de computadoras. Muchos de los problemas de optimización son NP-duro o NP-completo. Se sabe que el dibujado de grafos tiene estas características, cuando se aplican ciertos atributos estéticos al dibujado, como por ejemplo, el problema de la minimización de cruces de las aristas de un grafo incrustado en el plano, se considera que es NP-completo [Garey y Johnson (1983)], la minimización del área de dibujado, la minimización de la longitud máxima que deben tener las aristas, etc. [Sim (1996)].

2.3.2. Estructura del dibujado automático de grafos

Se consideran cinco aspectos en la estructura del dibujado automático de grafos: *objetos de dibujado*, *convenciones de dibujado*, *reglas de dibujado*, *relaciones de prioridad* y *las características de los algoritmos de dibujado* [Sugiyama (2002)].

2.3.2.1. Objetos de dibujado

La mayoría de las investigaciones han tratado principalmente con grafos simples (grafos con múltiples aristas y sin ciclos) *como objetos de dibujado*. Este tipo de grafos por lo general están clasificados como *árboles*, *grafos dirigidos*, *grafos planares* y *grafos no dirigidos* (en 2.1.1 de esta tesis). Investigaciones recientes incluyen al los *grafos compuestos*, que son grafos que manejan dos conjuntos de aristas: adyacencia de aristas y jerarquía de aristas que es interpretada como una relación de inclusión. La manera de dibujar un grafo compuesto implica que los nodos que contienen a otros nodos son dibujados como rectángulos, y los nodos contenidos por otros nodos son dibujados dentro del rectángulo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En la figura 2.18 se puede apreciar el ejemplo de un grafo dirigido compuesto, donde los nodos son particionados en capas. Todos los nodos de una capa son dibujados sobre una línea horizontal y las aristas son dirigidas de arriba hacia abajo. Sin embargo, en esta tesis sólo se trabajó con la clase de grafos planares no dirigidos [Forster (2002)].

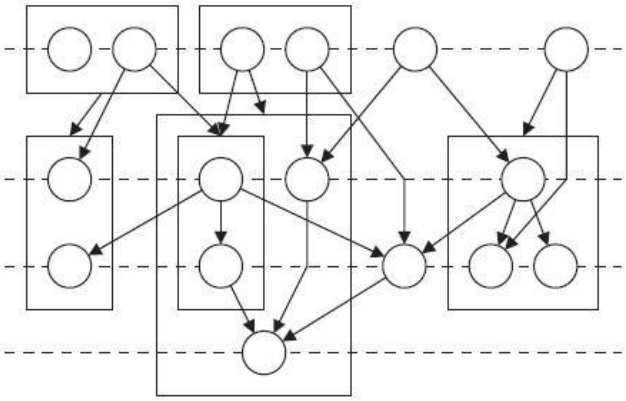


Figura 2.18. Grafo compuesto dividido en capas.

2.3.2.2. Convenciones de dibujado

La *convención de dibujado* es el protocolo para la colocación de los vértices y la dirección que deben tomar las aristas. Este protocolo consiste en las restricciones que se deben cumplir por un dibujado. Estas convenciones se dividen en *convenciones de colocación* para los vértices y las *convenciones de dirección* para las aristas [Sugiyama (2002)].

Los vértices son colocados acorde a un sistema estándar de coordenadas, la figura 2.19 muestra el sistema de coordenadas estándar y compuesta. El sistema de coordenadas estándar corresponde a las letras *a* a la *f*. También, se puede apreciar el sistema de coordenadas compuesto *m* que combina *b* y *e*.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

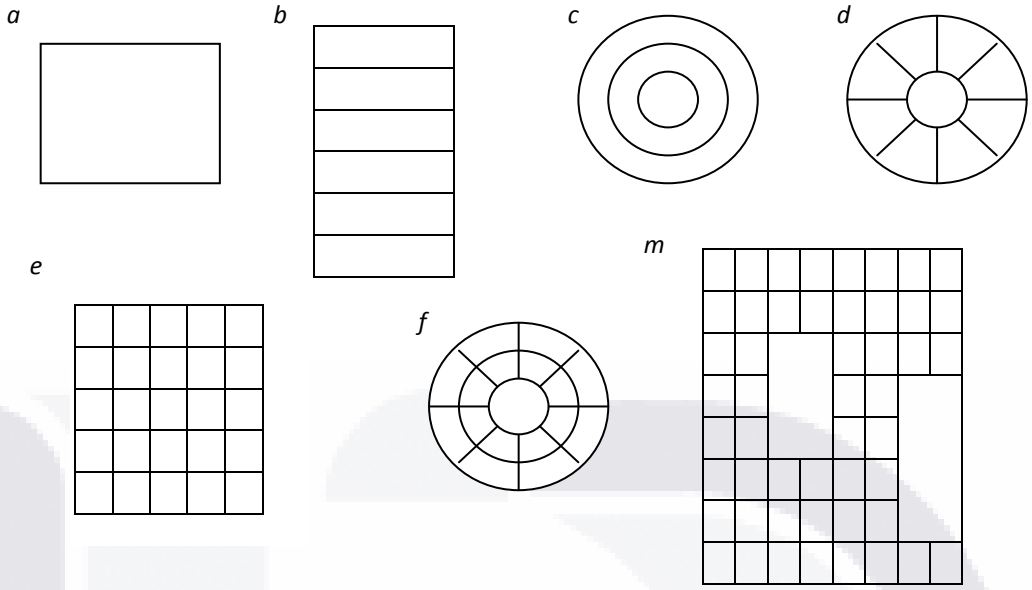


Figura 2.19. Sistema de coordenadas estándar y compuesto.

En la tabla 2.1 se muestran las convenciones de colocación de vértices.

SISTEMA DE COORDENADAS	CONVENCIONES DE COLOCACION
a	Colocación libre (no hay restricciones de colocación)
b	Colocación en línea paralela (sobre o entre líneas paralelas)
c	Colocación en círculos concéntricos (sobre o entre círculos concéntricos)
d	Colocación en línea radial (sobre o entre líneas radiales)
e	Colocación de malla ortogonal (sobre la malla entera)
f	Colocación sobre la intersección de una malla polar

Tabla 2.1. Convenciones de colocación de vértices.

La tabla 2.2 muestra las convenciones de dirección de aristas, que consiste en los tipos de líneas, y la relación con el sistema de coordenadas

SISTEMA DE COORDENADAS	CONVENCIONES DE COLOCACION
Tipo de línea	1. Direccionamiento de línea recta
	2. Direccionamiento poly-línea
	3. Direccionamiento curvo (direccionamiento con curvas)
Relación con el sistema de coordenadas	4. Independiente del sistema de coordenadas (el direccionamiento de líneas es independiente del sistema de coordenadas)
	5. Dependiente del sistema de coordenadas (el direccionamiento de líneas es paralelo a o perpendicular al sistema de coordenadas)

Tabla 2.2. Convenciones redirección para aristas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Cuando las convenciones de colocación de vértices y de dirección de aristas se combinan, se tiene la siguiente relación [Sugiyama (2002)]:

- 6 convenciones de colocación
- 3 tipos de línea
- 2 relaciones con el sistema de coordenadas

Esto genera 36 posibles convenciones de dibujado formadas por tripletas, por ejemplo, en la presente tesis se utilizó:

Colocación libre, direccionamiento de líneas rectas, e independiente del sistema de coordenadas.

2.3.2.3. Reglas de dibujado

Es muy difícil definir que es un "buen" dibujado de un grafo, ya que esto implica muchos factores como son el área de aplicación o el gusto del usuario. Existen reglas que se aplican al dibujado de grafos simples llamadas *reglas estáticas* y *reglas dinámicas*, estas últimas se aplican a cuando el grafo necesita ser cambiado y redibujado. En esta tesis se hace uso sólo de las reglas estáticas y se describen a continuación [Sugiyama (2002)].

2.3.2.3.1. Reglas básicas, semánticas y estructurales

Las reglas estáticas se dividen en *reglas básicas*, *reglas semánticas* y *reglas estructurales*. Las reglas básicas tienen que ver con el empalme de los vértices y las aristas. Como por ejemplo, el evitar el empalme de los vértices, evitar el empalme de las aristas, así como, el empalme de los vértices y las aristas. Las reglas semánticas están relacionadas a la importancia que un vértice o una arista pueda tener en el dibujado con respecto a la colocación de los vértices o la dirección de las aristas. A menudo estas reglas son dadas por el usuario o automáticamente derivadas de la información de los vértices y las aristas. Como por ejemplo, la colocación específica de los vértices sobre una línea recta, o dibujar los vértices con un tamaño específico. Las reglas estructurales son reglas sobre la colocación y dirección que tiene que ver sólo con la información estructural (con las características de la teoría de grafos) de un grafo dado, como por ejemplo, la minimización

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

de cruces de aristas, minimización del área de dibujado, colocación uniforme de los elementos del grafo, etc. En la presente tesis se hace uso de todas las reglas básicas, así como, la regla estructural de minimización de cruces en las aristas. Un ejemplo de las reglas utilizadas en esta tesis, se muestran en la figura 2.20.

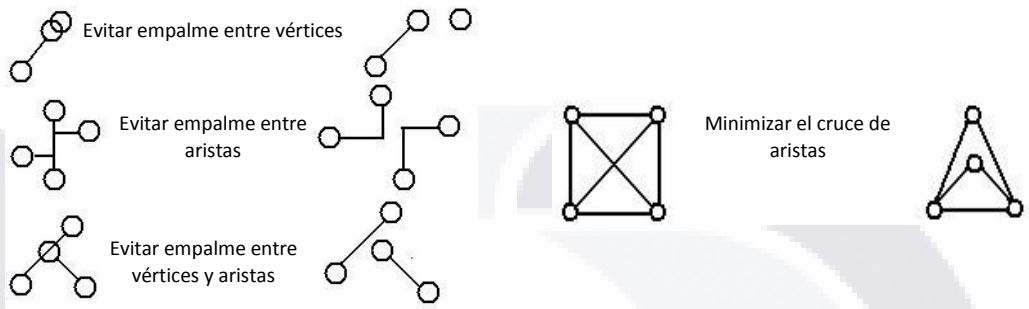


Figura 2.20. Reglas básicas y estructurales utilizadas en esta tesis.

Las reglas de dibujado, también pueden se clasificadas de acuerdo a cuatro ejes, ejemplo:

- Si la solución a una regla puede ser obtenida de forma única (U), o no (N).
- Si la regla es topológica (T) (específica solo las relaciones de colocación entre los elementos), orientado a la forma (S, shape) (especifica la dirección), o métrica (M) (especifica las distancias entre los elementos).
- Si la regla se aplica globalmente (G), a todo el dibujado, o localmente (L), sólo a una parte del dibujado.
- Si la regla es jerárquica (H, hierarchical), plana (F, flat), o ambas (B, both).

El conjunto completo de todas las reglas semánticas, estructurales, así como la clasificación de ejes, puede ser consultado en [Sugiyama (2002)].

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.3.2.4. Relaciones de prioridad

Existe una prioridad entra la convenciones de dibujado y las reglas de dibujado, ya que existe una relación de competencia y una relación de dependencia entre ellas, es decir, no existe independencia entre ellas.

Las relaciones de prioridad son puestas por cada algoritmo o aplicación, pero es necesario recordar siempre, las *relaciones generales de prioridad* que se deben aplicar a las convenciones de dibujado y a las reglas de dibujado. Por ejemplo, las convenciones de dibujado deben ser siempre satisfechas, mientras que las reglas de dibujado, deben ser satisfechas tanto como sea posibles.

De manera general las "convenciones de dibujado son > a las reglas de dibujado". Esto puede ser visto como reglas generales a las que se les deben dar preferencia sobre las reglas locales. La tabla 2.3 ejemplifica las relaciones de prioridad del dibujado automático de grafos [Sugiyama (2002)].

RELACIONES DE PRIORIDAD					
1.	Convenciones de colocación	>	Convenciones de dirección		
2.	Convenciones de dibujado	>	Reglas de dibujado		
3.	Reglas semánticas	>	Reglas estructurales		
4.	Únicas (U)	>	No únicas (N)		
5.	Reglas topológicas (T)	>	Reglas de forma (S)	>	Reglas métricas (M)
6.	Reglas globales (G)	>	Reglas locales (L)		
7.	Reglas jerárquicas (H)	>	Reglas comunes (B)	>	Reglas planas (F)

Tabla 2.3. Relaciones de prioridad del dibujado automático de grafos.

2.3.2.5. Algoritmos de dibujado

Las *convenciones de dibujado* son la restricciones que deben cumplirse, las *reglas de dibujado* son las metas de los problemas a optimizar. Un algoritmo de dibujado de grafos resuelve un numero de restricciones en problemas de optimización, teniendo siempre en cuenta el orden de prioridad. Las reglas de prioridad son de suma importancia para diseñar y conducir las estrategias para satisfacer las reglas de dibujado. Las reglas semánticas son consideradas como restricciones y las reglas estructurales son el método lograr un buen

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

dibujado por el algoritmo de dibujado de grafos. De la tabla 2.3 la relación de prioridad número 5 es de las más importantes, ya que se considera que se deben satisfacer primero las *reglas topológicas*, para definir la colocación que debe existir entre los elementos que se van a dibujar, enseguida las *reglas de forma* se deben satisfacer, para determinar las relaciones de longitud, amplitud y dirección que deben tener los elementos del dibujado, y por último, las *reglas métricas* se deben satisfacer, para indicar las distancias entre sus elementos.

Existen pocos métodos eficientes que satisfacen las reglas y que al final, difícilmente obtienen una solución óptima. Existen varios métodos heurísticos que han sido investigados, así como, algunos que tienen garantías teóricas. Es por eso, que existe la necesidad de crear nuevos métodos, que involucren estrategias heurísticas innovadoras en el campo del dibujado de grafos.

Dentro de los algoritmos que han sido utilizados en el dibujado de grafos tenemos:

- Algoritmos que usan la teoría de grafos y los algoritmos de grafos.
- Algoritmos heurísticos.
- Algoritmos que utilizan fuerza directa como son el modelo de los resortes, escalador de colinas estocástico, etc., los que utilizan modelos termodinámicos como recosido simulado, modelos bioinformáticas como los algoritmos genéticos, además de otras simulaciones.

En la presente tesis se hace uso de los siguientes métodos para el dibujado automático de grafos: A*lgoritmos Genéticos* (AGs), A*lgoritmos de Estimación de Distribuciones* (*Algoritmos de Estimación de la Distribución Marginal Univariada*, *AEDMU*) y el E*scalador de Colinas* (EC).

2.4 Inteligencia Artificial

La Inteligencia Artificial (IA) tiene sus inicios a mediados del siglo XX, su motivación inicial era, construir máquinas que pudieran simular el pensamiento humano, o cuando menos, emular alguna capacidad que mostrara cierta inteligencia.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La inteligencia artificial es capaz de proporcionar métodos que permitan realizar de forma automática algunas de las tareas que los seres humanos realizamos de manera cotidiana. Los sistemas inteligentes que existen y están en desarrollo, deben apoyarse en la investigación multidisciplinaría, ya que para integrar conceptos y métodos, necesitan de diferentes áreas como ciencias de la computación, inteligencia artificial, teoría de la comunicación, etc. [Pajares y Santos (2006)].

El concepto de "Inteligencia Artificial" es relativamente nuevo, Marvin Minsky, plantea: *La inteligencia Artificial es una ciencia orientada al diseño y construcción de máquinas que implementen tareas propias de humanos dotados de inteligencia*.

Una de las áreas donde la IA ha sido de suma importancia, es en la solución de problemas relacionados con la optimización combinatoria. Este tipo de problemas se cree son difíciles en general y son conocidos como problemas NP-duro o NP-completos [Garey y Johnson D. S. (1983)]. Para la solución de estos problemas se han utilizado en un principio, procedimientos de búsqueda directa, y después técnicas heurísticas más sofisticadas, que han sido uno de los principales campos de esta ciencia y sobre la cual se basa la presente tesis. A los algoritmos que resuelven este tipo de problemas se les conoce como *algoritmos de optimización combinatoria* o *métodos heurísticos* (algoritmos *metaheurísticos*), que son utilizados para resolver instancias de problemas que se cree son difíciles en general, explorando el espacio de soluciones (usualmente grande) para estas instancias. Algunas de las metaheurísticas utilizadas para resolver este tipo de problemas son: la búsqueda local, el recocido simulado, la búsqueda tabú, los algoritmos genéticos, optimización basada en colonia de hormigas, entre otras. La presente tesis hace uso de las metaheurísticas de algoritmos genéticos, algoritmos de la estimación marginal univariada y el algoritmo escalador de colinas, para resolver el problema de la minimización de cruces en las aristas de un grafo. Estas metaheurísticas se describen a continuación.

2.4.1. Algoritmo Genéticos Simple (AGS)

Los *Algoritmos Genéticos* son algoritmos de búsqueda basados en los mecanismos de la selección natural y la genética natural. Este tipo de algoritmos hacen evolucionar una población de individuos sometiéndolos a acciones aleatorias (parecidas a las que actúan en

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

la evolución biológica), así como también a la selección de acuerdo con algún criterio, por lo cual se determina cuales son los individuos mejor adaptados al problema, que deben sobrevivir, y cuales son los menos adaptados, que deben ser descartados.

Los algoritmos genéticos fueron desarrollados por John Holland en 1970 [Holland, J. (1975)]. El objetivo de su investigación se baso en la explicación abstracta y rigurosa del proceso adaptativo que existe en los sistemas naturales, así como también, en el diseño de programas de sistemas artificiales, que pudieran contener los mecanismos de los sistemas naturales [Goldberg (1989)].

Pseudocódigo: Los algoritmos genéticos mecánicamente son bastante simples, ya que no involucran nada más que la generación de números aleatorios, copia de cadenas de caracteres (strings), e intercambio de parcial de Springs. Los algoritmos genéticos procesan poblaciones de strings, por lo que, su estructura principal es una población de strings. A continuación se detallan las estructuras de datos que utiliza un algoritmo genético simple. La figura 2.21 muestra el pseudocódigo del AGS.

```
/* Algoritmo Genético Simple */

BEGIN
   Generar población inicial.
   Calcula la función de objetivo para cada individuo.
   WHILE NOT Fin DO
   BEGIN
      FOR Tamaño población/2 DO
      BEGIN /*Ciclo Reproductivo */
            -Selecciona dos individuos de la generación anterior,
              para el cruce (probabilidad de selección proporcional
              a la función de evaluación del individuo).
            -Cruza con cierta probabilidad los dos
              individuos obteniendo dos descendientes.
            -Muta los dos descendientes con cierta probabilidad.
            -Calcula la función de evaluación de los dos
              descendientes mutados.
            -Inserta los dos descendientes mutados en la nueva generación.
      END
      IF  población converge THEN
         Fin := TRUE
```


TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
  END
END
```

Figura 2.21. Pseudocódigo del Algoritmo Genético Simple.

Codificación: Los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros denominados *genes*, y que agrupados forma un conjunto de valores llamados *cromosomas*, representados por un alfabeto de ceros y unos {0,1}.

A éste conjunto de parámetros que representan el cromosoma se le denomina *fenotipo*. El fenotipo contiene la información requerida para construir un organismo, el cual se refiere como *genotipo*. La adaptación al problema de un individuo depende de la evaluación del genotipo. Esta última puede inferirse a partir del fenotipo, es decir puede ser calculada a partir del cromosoma, usando la función de evaluación.

La *función de adaptación* se diseña para cada problema. Dado un cromosoma, la función de adaptación le asigna un número real, que refleja el nivel de adaptación al problema del individuo que es representado por el cromosoma.

En la *fase reproductiva* se seleccionan los individuos de la población a cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos. La *selección* padres se realiza al azar utilizando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Este procedimiento está basado en una ruleta sesgada. Los individuos mejor adaptados se escogerán probablemente varias veces en cada generación, mientras que los menos adaptados, sólo se escogerán de vez en cuando. Ya seleccionados los dos padres, sus cromosomas se combinan, utilizando los operadores de cruce y mutación.

El operador de *cruce*, toma los padres seleccionados y corta los genes (ristas) de sus cromosomas en una posición escogida al azar, lo que produce dos subristras iniciales y dos subristras finales. Enseguida se intercambian las subristras finales, y se producen dos nuevos cromosomas. Ambos descendientes heredan genes de cada uno de los padres. Este

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

operador se conoce como *operador de cruce basado en un punto*. Un ejemplo de la codificación del operador de cruce se muestra en la figura 2.22

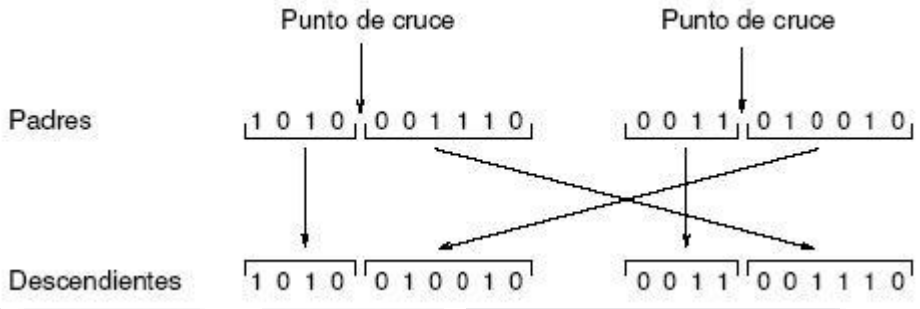


Figura 2.22. Operador de cruce de un AGS.

No siempre el operador de cruce se aplica a todos los pares de individuos que han sido seleccionados para emparejarse, sino que se aplica de manera aleatoria, normalmente con una probabilidad comprendida entre 0.5 y 1.0. Cuando el operador de cruce no se aplica, la descendencia se obtiene duplicando los padres.

El operador de *mutación* se aplica a cada hijo de manera individual, y consiste en la alteración aleatoria (normalmente con probabilidad pequeña) de cada gen componente del cromosoma. La Figura 2.23 muestra la mutación del quinto gen del cromosoma. Los dos operadores son muy importantes, por un lado, el operador de cruce realiza una exploración rápida del espacio de bsúqueda, y por el otro lado, el operador de mutación asegura que ningún punto del espacio de búsqueda tenga la probabilidad de no ser examinado, y es de suma importancia para asegurar la convergencia de los Algoritmos Genéticos.

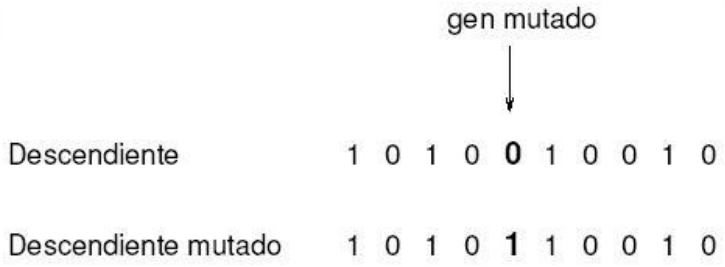


Figura 2.23. Operador de mutación de un AGS.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.4.2. Algoritmo Escalador de Colinas

El *Algoritmo Escalador de Colinas* es una técnica de optimización que pertenece a la familia de la *búsqueda local*. Este algoritmo utiliza una serie de iteraciones donde se va desplazando constantemente hacia la dirección de un mejor valor . Cuando el algoritmo llega a un punto donde no se consigue mejorar su resultado, este necesita empezar de nuevo en otro punto hacia donde diriguir su búsqueda. Esto se consigue con el *reinicio aleatorio* del algoritmo. Esta técnica realiza una serie de búsquedas escalando la cima desde estados inciciales generados aleatoriamente. El mejor resultado que hasta el momento se haya obtenido se va guardando según avanzan la iteraciones del algoritmo y se detiene cuando no se logra ningún avance significativo. La condición de paro del algoritmo se puede establecer con un número fijo de iteraciones o cuando el mejor de los resultado no haya sido mejorado durante una cierta cantidad de itraciones.

En general el Escalador de Colinas proporciona una solución cercana al óptimo, pero esto no garantiza que lo encuentre. Este algoritmo tiene la ventaja de encontrar rapidamente una solución, pero tiene algunas desventajas si son atrapados en estados que no son el objetivo, por ejemplo:

Máximo local: estado mejor que sus vecinos, pero no es mejor que otros que están más alejados.

Meseta: Es un espacio de búsqueda en el que todo un conjunto de estados vecinos tienen igual valor.

Riscos: Es un tipo de máximo local, imposible de atravesar con movimientos simples.

Para evitar máximos locales, se puede regresar a un estado anterior y explorar en una dirección diferente. Para el caso de mesetas, se puede dar un salto grande en alguna otra dirección y tratar de encontrar una nueva sección del espacio de estados. Para los riscos, se pueden aplicar dos o más reglas, antes de realizar una prueba del nuevo estado, esto equivale a moverse en varias direcciones a la vez [Rusell y Norvig (1996)]. La figura 2.24 muestra el pseudocódigo general del Escalador de Colinas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
// PSEUDOCODIGO ESCALADOR DE COLINAS

function HILL_CLIMBING(problem) returns a state is a local maximum
inputs: problem, a problem
local variables: current, a node
neighbor, a node
current ← MAKE-NODE(INITIAL-STATE[problem])
loop do
neighbor ← a higthest-valued successor of current
if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
current ← neighbor
```

Figura 2.24. Pseudocódigo de la aproximación EC.

2.4.3. Algoritmo de Estimación de Distribuciones

Los *Algoritmos de Estimación de Distribuciones* (*Estimation of Distribution Algorithms* (*EDAs*)), son un conjunto de métodos clasificados en el paradigma de la computación evolutiva. Estos métodos se basan principalmente en sustituir el cruce y la mutación por la estimación y posterior muestreo de una distribución de probabilidad aprendida a partir de los individuos seleccionados. Este tipo de algoritmos han tomado mucho auge dentro del campo de la investigación ya que son utilizados para resolver problemas de optimización combinatoria. Los EDAs fueron investigados por Mühlenbein y Paaß, y son utilizados dentro del campo de la computación evolutiva.

A diferencia de los algoritmos genéticos, los EDAs no necesitan operadores de cruce y de mutación. Los individuos de la nueva población se obtienen por una distribución de probabilidad, la cual se estima a partir de una conjunto de datos que tienen los individuos seleccionados de la generación anterior. En los EDAs las interrelaciones se expresan de manera explícita por medio de la distribución de probabilidad conjunta, asociada a los individuos seleccionados en cada generación [Larrañaga et. al. (2003)].

Pseudocódigo: Se inicia con la generanción de *M* individuos al azar, por ejemplo a partir de una distribución uniforme para cada variable. Los *M* individuos constituyen la

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

población inicial D_0, y enseguida se evalua cada individuo. En un primer paso, un número N ($N \leq M$) de individuos se seleccionan (los de mejor valor en la función objetivo). A continuación, se efectúa la inducción del modelo probabilístico *n*–dimensional que mejor refleja las interdependencias entre las variables. En un tercer paso, se obtienen M nuevos individuos (la nueva población) a partir de la simulación de la distribución de probabilidad que se ha aprendido en el paso previo. Los tres pasos anteriores se repiten hasta que se verifique una condición de parada, por ejemplo, la terminación del número de generaciones no mejora en relación con el mejor individuo obtenido en la generación previa, etc. El pseudocódigo de la aproximación EDA se puede apreciar en la figura 2.25.

```
// Aproximación EDA

   D_0  Genera M individuos al azar (población inicial)

   Repite For l = 1, 2, . . . hasta que se cumpla un criterio de
   parada
   D_{l-1}^{Se} ←  Seleccionar N≤ M individuos de D_{l-1} acorde con
   el método de selección
   p_l(x) = p(xD_{l-1}^{Se}) ←  Estimar la distribución de probabilidad
       de los individuos seleccionados

   D_l  Muestrear M individuos (la nueva población) a partir
   de p_l(x)
```

Figura 2.25. Pseudocódigo de la aproximación EDA.

2.4.3.1. Algoritmo de Estimación de la Distribución Marginal Univariada (AEDMU)

El *Algoritmo de Estimación de la Distribución Marginal Univariada* (*Univariate Marginal Distribution Algorithm (UMDA)*), fue introducido por Mühlenbein en 1998. Este algoritmo es una categoría perteneciente a los EDAs el cual se considera que no tiene dependencias. La distribución de probabilidad conjunta *n*-dimensional se factoriza como un

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

producto de *n* distribuciones de probabilidad univariantes e independientes, ejemplo: $p_l(x) = \prod_{i=1}^{n} p_l(x_i)$

Pseudocódigo: Es muy simple estimar la distribución de probabilidad conjunta de cada generación a partir de los individuos seleccionados $p_l(x)$. La distribución de probabilidad conjunta se factoriza como el producto de distribuciones univariantes e independientes, ejemplo: $p_l(x) = p(xD_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i)$

La figura 2.25 muestra el pseudocódigo de la aproximación AEDMU. Cada distribución de probabilidad univariante se estima a partir de las frecuencias marginales:

$$p_l(x) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i \left| D_{l-1}^{Se}\right.)}{N}$$

donde:

$$\delta_j(X_i = x_i \left| D_{l-1}^{Se}\right.) = \begin{cases} 1 \text{ si en } j\text{-ésimo caso de } D_{l-1}^{Se}, X_i = x_i \\ 0 \text{ en otro caso} \end{cases}$$

// PSEUDOCODIGO UMDA

D_0 Generar *M* individuos al azar (población inicial)

Repite para l = 1, 2, . . . hasta que se verifique el criterio de parada

$D_{l-1}^{Se} \leftarrow$ Selecciona $N \leq M$ individuos de D_{l-1} acorde a un método de selección

$p_l(x) = p(xD_{l-1}^{Se}) = \prod_{i=1}^{n} p_l(x_i) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i \left| D_{l-1}^{Se}\right.)}{N} \leftarrow$ estima la distribución de probabilidad conjunta

D_l Muestrea *M* individuos (la nueva población) a partir de $p_l(x)$

Figura 2.26. Pseudocódigo de la aproximación UMDA.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

2.5 Revisión de algoritmos para el dibujado de grafos

El problema del dibujado de grafos utiliza un enfoque algoritmico y un enfoque descriptivo [Lin T. Edades P. (1994)]. El algortimico crea algoritmos que son dificiles de adaptar o de extenderle nuevos requerimientos, lo quec resta flexibilidad y generalidad. El algoritmo más común es el Algoritmo de los Resortes presentado por [Edades (1984)] y adaptado por [Sugiyama y Misue (1995)], este simula un sistema mecánico en el que los vértices son los aros y las aristas los resortes, esta simulación muestra muy bien las simetrias, aunque no se garantiza la minimización de cruces. El enfoque declarativo tiene sus bases en la inteligencia artificial, los Algoritmos Genéticos [Branke et. al. (1997)], Eleoranta y Mäkinen (2001)]. Estos métodos se basan en la simulación de la reproducción sexual, creados por Holland [Holland, J. (1975)] y popularizados por Goldberg [Goldberg E. D. (1989)]. Búsqueda Tabú [Glover y Laguna (1997)], este método mantiene una lista de las soluciones ya visitadas, utilizada para evitar una nueva exploración de los puntos que son óptimos locales. Recocido Simulado [Davidson y Harel (1996)], isnpirado en el fenomeno físico del recocido de los metales. Cuando un metal se calienta hasta una temperatura alta y luego desciende la temperatura lentamente se grantiza que el sistema alcance el equilibrio termico en cada estado. Escalador de Colinas Estocástico[Rossete (2000)], inspirado en la situación imaginaria de un alpinista que está buscando la cima de una montaña, pero debido a la niebla sólo conoce los puntos inmediatos a el, utilizando esta informacion para ascender. Algoritmos de Estimación de Distribuciones, los principales trabajos son los algoritmos PBIL y COMIT en [Baluja y Davies (1998)], MIMIC de De-Bonet [De Bonet et. al. (1996)], y los algoritmos UMDA y FDA en [Mühlenbein et. al. (1998)]. La idea de estos métodos se basa en la Estadística. Estos algoritmos escojen la μ mejores soluciones de la población actual y crea un modelo estadístico de las mismas. Luego la siguiente población se genera a partir de un muestreo probabilístico del modelo creado, generandose tantas soluciones como en la población anterior y así se crea la nueva población. El algoritmo más simple es el UMDA [Rossete et. al. (1999)]. El algoritmo calcula las probabilidades marginales que tiene cada uno de los valores de cada variable, y se genera la próxima población de soluciones tomando para cada variable los valores de manera probabilística usando para ello las probabilidades marginales csalculadas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 3. ALGORITMOS PARA EL DIBUJADO DE GRAFOS

En este capítulo se presenta el pseudocódigo, y la estructura de datos de los diferentes algoritmos utilizados en esta tesis, para el problema del dibujado de grafos en la minimización de cruces de las aristas de un grafo.

3.1 Representación de la solución

A diferencia de la representación común de unos y ceros que existe en la literatura, para la codificación de una solución, en esta tesis, la *representación de la solución* en el problema de dibujado de grafos, esta dada por el conjunto de todos los *vértices* y *aristas* que forman parte de la solución final. Esta representación es utilizada en todos los algoritmos que aquí se describen. La estructura de datos utilizada para vértices y aristas es la siguiente:

Vértices: los vértices del grafo son representados por un par de coordenadas ((x_1,y_1) (x_2, y_2) … (x_i, y_i)) en un objeto de tipo vértice. La estructura de datos se muestra en la siguiente clase de la figura 3.1.

```
class Vertice                // objeto vértice
{
    private float x;         //Coordenada x (absisa)
    private float y;         //Coordenada y (ordenada)

// constructor con coordenadas  x, y

    public Vertice(float x, float y){
            this.x = x;
            this.y = y;
    }
}
```

Figura 3.1. Representación de la solución (objeto vértice).

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Aristas: las aristas del grafo son representadas por un par de vértices p_1, p_2. La distancia que separa a los vértices p_1 y p_2, es unida por una línea recta que representa una arista del grafo. Su estructura es un objeto del tipo arista, la clase de la figura 3.2 define el objeto arista.

```
class Arista
{   private int v1;                  // vértice 1
    private int v2;                  // vértice 2
    private double m;                // pendiente de la recta
    private double Y;                        // ecuación de la recta

    public Arista(int v1, int v2)    // constructor con dos vértices
    {
            this.v1 = v1;
            this.v2 = v2;
            this.m = 0.0;
            this.Y = 0.0;
    }
}
```

Figura 3.2. Representación de la solución (objeto arista).

3.2 Función objetivo

En esta tesis, la *función objetivo* (o *función de evaluación*) es utilizada por todos los algoritmos que aquí se describen. Esta función de evaluación tiene el propósito de determinar la adaptabilidad (*fitness*) de cada solución, es decir, tiene la capacidad de definir que aristas están cruzadas y cuales no, y de esta manera proporcionar un valor de adaptación a cada solución para que sea evaluada por el algoritmo.

La función objetivo para poder proporcionar este valor de adaptación, necesita apoyarse en algunos conceptos básicos relacionados con la geometría analítica, como son: ecuación general de la recta, pendiente de la recta, ecuación de la recta que pasa por dos puntos, intersección entre dos rectas y en el sistema de ecuaciones lineales con dos incógnitas por sustitución.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Los conceptos anteriores, están definidos dentro de funciones (*métodos*), que forman parte de un objeto de tipo grafo, el cual contiene la estructura datos adecuada para llevar acabo todas las operaciones relacionadas con la geometría analítica. En la figura 3.3 se muestra la estructura de datos que define el objeto tipo grafo que es utilizado por todos los algoritmos presentados en esta tesis.

```
class Grafo {
        String nombre;              // Nombre del grafo
        Vertice V[];                // conjunto de vértices del grafo
        Arista E[];                 // conjunto de aristas del grafo
        int tv;                     // total de vértices en el grafo
        int te;                     // total de aristas en el grafo
        int tc;                     //Total de cruces
        double fitness;             // valor de adaptabilidad del grafo

// constructor con el nombre del grafo
        public Grafo(String nombre, int v, int e)
        {                                   //total de vértices y total de aristas
                this.nombre = nombre;
                this.V = new Vertice[v];
                this.E = new Arista[e];
                this.tv = v;
                this.te = e;
                this.tc = 0;
                this.fitness = 0.00;
        }
}
```

Figura 3.3. Objeto grafo utilizado por la función objetivo.

En la figura 3.4 se describe el pseudocódigo de la función objetivo y las dos funciones más importantes para definir la intersección entre rectas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
private double funcionObjetivo(Grafo G)  {
    double accum = 0.0;
    G.obtenPendiente();              // obtiene pendiente y ecuación de la recta
    accum = G.obtenInterseccion();   // obtiene intersección de la recta
    return accum;                    // regresa total de cruces
}
public void obtenPendiente()  {
    for (int i = 0; i < total_aristas; i++ )
    {
        obten_vertices;              // obtiene vértices y coordenadas
        obten_coordenadas;           // para encontrar la pendiente de la
        obtenPendienteRecta;         // recta y la ecuación de la recta
        obtenEcuacionRecta;          // de cada arista del grafo
    }
}
public double obtenInterseccion()    // determina el total de cruces entre las
{                                    // aristas del grafo
     for (int i = 0; i < total_aristas - 1; i++ )    // ciclo externo  (recta i)
     {
            obtenPendienteRecta[i];
            obtenEcuacionRecta[i];
            for (int j = i + 1; j < total_aristas; j++ )// ciclo interno (recta i + 1)
            {
                obtenPendienteRecta[i+1];
                obtenEcuacionRecta[i+1];
                compruebaInterseccion; // comprueba intersección entre la
                if interseccion then // recta i  y  la recta  i+1
                cruce++;             // total de cruces
            }
     }  return cruce;                // regresa total de cruces
}
```

Figura 3.4. Función objetivo (objeto grafo).

3.3 Algoritmo Genético Simple para el dibujado de grafos

En este apartado se describe la estructura de datos y el pseudocódigo utilizado por el Algoritmo Genético Simple implementado en esta tesis, para el problema del dibujado de grafos en la minimización de cruces de las aristas de un grafo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

En este trabajo se utiliza un AGS (*Algoritmo Genético Simple*) con elitismo, con el objetivo de preservar siempre el mejor individuo de cada generación. Como parámetros el algoritmo utiliza: número de generaciones, tamaño de la población, probabilidad de cruzamiento y probabilidad de mutación. El algoritmo tiene dos estructuras de datos esenciales para su funcionamiento, las cuales están contenidas en dos objetos. Un objeto tipo individuo y un objeto tipo población. La figura 3.5 muestra la estructura del objeto tipo individuo. En esta figura se puede apreciar que en el cromosoma utilizado por el AGS, cada gen es un par de coordenadas de tipo vértice, a diferencia de la codificación habitual de unos y ceros utilizada en la mayoría de los algoritmos genéticos.

```
public class Individuo  {
    Vertice chrom[];                // Fenotipo (cromosoma)
    double x;                       // Genotipo
    double fo;                      // Valor de la función objetivo (función
de adaptación)
    double fitness;                 // Probabilidad de selección
    double acumfitness;             // Probabilidad de selección acumulada
    int parent1, parent2, xsite;    // Padres y punto de cruce
    char cruceNS;                   // Padre cruzado Si o No

    public Individuo(int lchrom)  {  // constructor de objeto individuo
        this.chrom = new Vertice[lchrom];
        this.x = 0.0;
        this.fo = 0.0;
        this.fitness= 0.0;
        this.acumfitness= 0.0;
        this.parent1 = 0;
        this.parent2 = 0;
        this.xsite = 0;
        this.cruceNS = 'N';
    }
}
```

Figura 3.5. Estructura de datos de un individuo de la población AGS.

En la figura 3.6 muestra la estructura de un objeto tipo población, la cual contiene un arreglo de individuos que forman parte de la población del Algoritmo Genético Simple,

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

además, contiene otros elementos para llevar el registro estadístico del comportamiento general del algoritmo.

```
public class Poblacion   {
        public int nmutation, ncross;            // total de cruces y mutaciones
        public double tavgcross;                 // porcentaje de total cruces
        public double max, min, sumfitness; // valor máximo, mínimo y suma
                                                    adaptabilidad
        Individuo poblacion[];                   // arreglo de individuos se la
                                                    población
        public int best, worst;                  // valor de la  mejor y peor
                                                    solución

        public Poblacion(int popsize)
        {
                this.nmutation = 0;
                this.ncross = 0;
                this.tavgcross = 0.0;
                this.max = 0.0;
                this.min = 0.0;
                this.sumfitness = 0.0;
                this.poblacion = new Individuo[popsize];
                this.best = 0;
                this.worst = 0;
        }
}
```

Figura 3.6. Estructura de datos de la población del AGS.

En la figura 3.7 se muestra el pseudocódigo del AGS con *elitismo*. Este algoritmo guarda el mejor individuo de la población de cada generación.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
// PSEUDOCODIGO DEL ALGORITMO GENETICO SIMPLE
   public class AGS
   {
         Poblacion Oldpop, Newpop;
         Individuo mejor, peor, unico;
         For (i = 0; i < tamaño_poblacion; i++)  {
            Oldpop[i] = Individuo;                          //genera población inicial
            Newpop = Oldpop;                                //genera nueva población inicial
            For (j = 0; j < total_vertices; j++)
                  Oldpop[i].vertice[j] = generaVertices;          // generación aleatoria de
                                                                  // vértices
            evaluaFobjetivo;                                // evalua la calidad de la
                                                            // población
            unico = guardaMejor;                            // guarda el mejor individuo en
                                                            // único
         }

         For (i = 0; i < maximo_generaciones; i++) {
            For (j = 0; j < tamaño_poblacion; j++) {
               Selección;                   // seleccióna padres de Oldpop
               Cruce;                       // cruce padres en Newpop si son elegibles
               Mutacion;                    // mutación en Newpop si son elegibles
               evaluaFobjetivo;             // evalua la calidad de la población Newpop
               mejor = guardaMejor;         // guarda el mejor individuo
               peor = guardaPeor;           // guarda el  peor individuo
            }
            If mejor < unico then           // si menos cruces
            unico = mejor;                  // guarda el mejor de la generación
            else
               peor = unico;              // guarda mejor anterior en peor de la poblac.
            oldpop = newpop;                // copia neva población en vieja población
         }
   }
```

Figura 3.7. Pseudocódigo del AGS.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Si el individuo actual es mejor (*menos cruces*) que el anterior individuo, entonces se guarda el mejor actual en el anterior; pero si no es así, entonces se guarda el mejor anterior en el peor individuo de la población actual. Con esta acción se asegura la convergencia del algoritmo al no perder nunca la mejor solución encontrada.

3.4 Algoritmo Escalador de Colinas para el dibujado de grafos

En este apartado se describe el pseudocódigo del algoritmo y la estructura de datos utilizada en el cromosoma del Algoritmo Escalador de Colinas.

El algoritmo trabaja por medio de iteraciones (*ciclos*) y siempre se va guardando el mejor resultado de la búsqueda que hasta el momento se haya obtenido, si el resultado de la búsqueda actual no es mejor que la anterior, el algoritmo regresa al estado anterior de la mejor solución encontrada. Se hace la aclaración que, el algoritmo se diseño con dos variantes en la selección de búsqueda, con el propósito de mejorar los resultados de la investigación. Una de esas variantes, utiliza el intercambio de dos vértices tomados aleatoriamente del cromosoma del individuo. La otra variante, utiliza el reinicio aleatorio del algoritmo, este toma aleatoriamente un sólo vértice del cromosoma y lo cambia por un vértice nuevo generado aleatoriamente. La estructura de datos utilizada por el algoritmo EC es un objeto de tipo IndividuoG, ver figura 3.8.

```
public class IndividuoG
{
   Vertice chrom[];              // Fenotipo (cromosoma)
   double x;                     // Genotipo
  tipo                           // variante de algoritmo Escalador de
Colinas

   public IndividuoG(int lchrom, int tipo)  // constructor de objeto individuo
                                            // para EC y UMDA
   {
      this.chrom = new Vertice[lchrom];
      this.x = 0.0;
   }
}
```

Figura 3.8. Estructura de datos del individuo general utilizado por el algoritmo EC.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El pseudocódigo del algoritmo Escalador de Colinas se muestra en la figura 3.9. Al principio el algoritmo sólo crea un individuo generando las coordenadas de sus vértices de manera aleatoria. El algoritmo entra en un ciclo donde el individuo intenta mejorar en cada iteración, si en alguna iteración su resultado no mejora, el individuo regresa al estado anterior de su mejor resultado, y termina hasta que el ciclo se cumple. También, se puede apreciar las dos variantes utilizadas para resolver el problema de minimización de cruces.

```
// PSEUDOCODIGO DEL ALGORITMO ESCALADOR DE COLINAS
   public class Hill  {
        public Hill(int tipo) {
            this.tipo = tipo;
        }
        IndividuoG  mejor, actual  ;
        FOR (j = 0; j < total_vertices; j++)
             actual.vertice[j]= generaVertices;          // generación aleatoria de vértices
        actual = evaluaFobjetivo;                        // evalúa la calidad del individuo
        mejor.x = 99999999;

        WHILE actual.x > 0 AND j < total_iteraciones {          // condición de paro
            IF tipo = 1
               Intercambio;                              // variante 1 intercambio de vértices
            ELSE
               Reinicio_aleatorio;                       //variante 2 de reinicio aleatorio
            actual = evaluaFobjetivo;                    // evalúa la calidad del individuo
            If actual < mejor then                       // si menos cruces
               mejor = actual;                           // actualiza mejor
            else
               actual  = mejor;                          // regresa al estado anterior mejor.
        }
   }
   Public Intercambio {
        Int v1, v2 = 0;
        WHILE v1 = v2  {
            v1 = selecciona_vertice(actual.vertice);    // selección aleatoria (vértice en actual)
             v2 = selecciona_vertice(actual.vertice);   // selección aleatoria (vértice en actual)
        }
        actual.vertice[v1] = v2;                         // intercambia posición de vértices
        actual.vertice[v2] = v1;
   }
```


TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
Public Reinicio_aleatorio {
    Int v;
    v = selecciona_vertice(actual.vertice);          // selección aleatoria de un vértice en actual)
    actual.vertice[v]= generaVertices;      // cambia un solo vértice aleatoriamente
}
```

Figura 3.9. Pseudocódigo del EC.

3.5 Algoritmo de la Estimación Marginal Univariada para el dibujado de grafos

Tratando de mejorar los resultados obtenidos hasta el momento, con los dos algoritmos descritos anteriormente, se decidió hacer la implementación de un tercer algoritmo con características diferentes.

El Algoritmo de la Estimación Marginal Univariada trabaja con una distribución de probabilidad univariada e independiente, los parámetros que utiliza para su funcionamiento son: número de generaciones, tamaño de la población y porcentaje de truncamiento de la población. La estructura de datos del algoritmo utiliza un objeto tipo IndividuoG, descrito anteriormente en la figura 3.8, así como también, un objeto tipo PoblacionG, que se muestra en la figura 3.10. En esta estructura se muestra un arreglo de individuos que pertenecen a la población utilizada por el AEDMU.

```
public class PoblacionG  {
      IndividuoG poblacion[];              // arreglo de individuos se la población
      public int best;                     // valor de la  mejor y peor solución

      public Poblacion(int popsize)  {
            this.poblacion = new IndividuoG[popsize];
            this.best = 0;
            this.worst = 0;
      }
}
```

Figura 3.10. Estructura de datos de la población utilizada por el AEDMU.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La figura 3.11 muestra el pseudocódigo del Algoritmo de la Estimación Marginal Univariada, en el se describen los pasos utilizados por el algoritmo para minimizar el número de cruces del grafo. El primer paso consiste en crear la población de individuos, generando aleatoriamente los vértices del grafo. Después se entra en un ciclo que termina cuando se alcanza el total de generaciones del algoritmo. Este ciclo comienza evaluando todos los individuos de la población, para saber la adaptabilidad que tienen con respecto al problema de la minimización de cruces. En seguida, se ordena la población en forma descendente tomando en cuenta la cantidad de cruces generados en el paso anterior, es decir, del menor número de cruces al número mayor de ellos. Por último, se calcula el vector de probabilidad tomando sólo un porcentaje de la población, con el fin de obtener la distribución de probabilidad y en base a esto generar la nueva población.

```
// PSEUDOCODIGO DEL ALGORITMO AEDMU
   public class UMDA  {
        PoblacionG oldpop, newpop;
        IndividoG mejor;
        For (i = 0; i < tamaño_poblacion; i++)          //genera población inicial
           oldpop[i] = Individuo;
        For (j = 0; j < total_vertices; j++)    // generación aleatoria de vértices
           oldpop[i].vertice[j] = generaVertices;
        oldpop = newpop;                        // copia vieja población en nueva población
        For (i = 0; i < maximo_generaciones; i++) {
           evaluaFobjetivo;                     // evalua la calidad de la población
           ordenaPoblacion;                     // ordena descendente x total de cruces
           mejor = guardaMejor;                 // guarda el mejor individuo
           calculaVectoProb;                    // calcula vector de probabilidad
           oldpop = newpop;                     // copia nueva población en vieja población
         }
   }
```

Figura 3.11. Pseudocódigo del Algoritmo de la Estimación Marginal Univariada.

3.6 Algoritmo híbrido HUx para el dibujado de grafos

Los resultados obtenidos por el Algoritmo de la Distribución Marginal Univariada muestran una similitud con los resultados obtenidos por el Escalador de Colinas, en el problema de la minimización de cruces en el dibujado de grafos. Es por eso que, surge la

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

idea de unir los dos métodos de optimización, con el propósito de mejorar aun más los resultados obtenidos hasta ese momento. El diseño de los experimentos y el análisis de los resultados de todos los algoritmos se comentan en el capítulo 4 de la tesis.

El algoritmo híbrido HUx utiliza la misma estructura de datos que es utilizada por el AEDMU, es decir, los objetos IndividuoG descrito anteriormente en la figura 3.8 y el objeto PoblacionG mostrado en la figura 3.10, también son utilizados por el algoritmo híbrido HUx, por lo que sólo se comenta el pseudocódigo de este algoritmo.

HUx esta basado principalmente en el algoritmo AEDMU, por lo que fue necesario modificar este algoritmo. La hibridación del algoritmo fue realizada utilizando 10 ciclos del algoritmo EC con reinicio aleatorio, y se modifico con elitismo, ya que el algoritmo se estancaba en un óptimo local. La figura 3.12 muestra la hibridación hecha al algoritmo AEDMU.

```
// PSEUDOCODIGO DEL ALGORITMO HIBRIDO HUx
   public class  UMDA  {
        boolean hux;                          // para hibridación y elitismo
        public  UMDA(boolean hux) {       // constructor para hibridación
             this.hux = hux;
        }
        PoblacionG oldpop, newpop;
        IndividoG mejor;
        For (i = 0; i < tamaño_poblacion; i++)          //genera población inicial
           oldpop[i] = Individuo;
        For (j = 0; j < total_vertices; j++)    // generación aleatoria de vértices
           oldpop[i].vertice[j] = generaVertices;
        newpop = oldpop;
```


TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

```
For (i = 0; i < maximo_generaciones; i++) {    // ciclo del algoritmo para nueva población
        evaluaFobjetivo;                       // evalua la calidad de la población
        ordenaPoblacion;                       // ordena descendente x total de cruces
        if this.hux then {
            elitismo;                          // aplica elitismo a la población
            aplicaEC;                          // 10 ciclos del EC al mejor de la  población
        }
        mejor = guardaMejor;                   // guarda el mejor individuo
        calculaVectoProb;                      // calcula vector de probabilidad
        oldpop = newpop;                       // copia nueva población en vieja población
    }
}
```

Hibridación

Figura 3.12. Hibridación hecha al Algoritmo de la Estimación Marginal Univariada.

La implementación de la hibridación hecha en el algoritmo AEDMU, se realizó de la siguiente manera:

- Se aplica elitismo, sólo si el mejor individuo de la población tiene una mejor adaptabilidad al problema (menos cruces) que el peor individuo de la población. en este caso, se guarda el mejor individuo de la población en el peor individuo y se vuelve a ordenar toda la población.

- Se realiza la hibridación, aplicando 10 ciclos del algoritmo Escalador de Colinas con reinicio aleatorio, al mejor individuo de la población. El resultado de esta hibridación puede mejorar la adaptabilidad del individuo, que es lo que sucede en la mayoría de los casos, o su adaptabilidad puede empeorar; si este fuera el caso, la convergencia del algoritmo se logra de todas formas, ya que, al aplicar elitismo en la población, la mejor solución encontrada hasta ese momento nunca se pierde.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 4. SISTEMA MinX

En este capítulo se hace la presentación del Sistema MinX, que es la interfase visual utilizada para mostrar los resultados obtenidos por las diferentes metaheurísticas usadas en esta tesis. Así mismo, se explican todos los elementos involucrados en el funcionamiento del sistema que ayudan en la visualización del dibujado automático de los grafos. Al final se comenta la importancia que ha tenido la interfase visual del Sistema MinX en este trabajo de tesis.

4.1 Presentación del Sistema MinX

Una interfase grafica de usuario (*Graphical User Interfce* (*GUI*)) es un programa que permite a las personas poder interactuar con los dispositivos electrónicos (como computadoras, dispositivos de juegos, etc.), por medio de imágenes y objetos gráficos para representar la información, a diferencia de los comandos de texto que se utilizaban en los equipos de computadoras de años anteriores [Jansen (1998)].

El sistema MinX es una interfase gráfica de usuario, con el propósito de servir como una herramienta visual en el dibujado automático de grafos. El Sistema MinX es una herramienta amigable y por lo tanto, fácil de usar. Por medio de este sistema, el usuario puede obtener la salida visual del dibujado automático de uno o varios grafos dependiendo de los parámetros seleccionados al inicio del programa. El sistema MinX contiene un grupo de algoritmos de optimización, utilizados para resolver el problema de la minimización de los cruces que se puede dar entre las aristas de un grafo.

4.2 Característica generales del Sistema MinX

Aquí se muestra brevemente las características generales del Sistema MinX, la tecnología utilizada para la implementación del sistema, las bases de datos que soportan las entradas y salidas de información, las salida visual de los grafos, así como los métodos utilizados para resolver el problema del dibujado de grafos en la minimización de los cruces en las aristas de un grafo. A continuación se describen las características generales de este sistema.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- *Implementado en lenguaje Java*: el sistema MinX fue desarrollado en lenguaje Java [Sun Microsystems Java (2008)] [Horstman y Cornell (2000)], por lo tanto sigue una metodología orientada a objetos, la decisión de utilizar este lenguaje de programación y no otro, se debió a las ventajas que tiene para reutilizar los objetos ya creados, también, a la portabilidad que tiene para ejecutarlo en diferentes plataformas, así como, la robustez de su interfase gráfica, entre otras.

- *Base de datos:* es sistema cuenta con dos tipos de bases de datos. Una parte del sistema trabaja con archivos de texto, que son los archivos de entrada y salida, y la otra parte utiliza la base de datos relacional MySQL [Sun Microsystems MySQL (2008)] [Maslakowoski (2001)], que se usa como soporte de datos para dibujar cualquier grafo en la interfase grafica.

- *Ventanas:* utiliza dos tipos de ventanas que muestran los resultados obtenidos por las diferentes metaheurísticas del sistema. Una ventana contiene una tabla de datos con la información del número mínimo de cruces encontrados y las coordenadas de cada grafo, otra ventana muestra el dibujado automático del grafo.

- *Metaheurísticas utilizadas*: el sistema MinX contiene tres metaheurísticas implementadas dentro de su sistema que son: el Algoritmo genético Simple, el Escalador de Colinas y el Algoritmo de la Estimación Marginal Univariada. Todos los algoritmos trabajan internamente en el sistema, de manera independiente, para resolver el problema de la minimización de los cruces en las aristas de un grafo.

4.3 Diagrama general del Sistema MinX

Este apartado muestra el diseño lógico del sistema MinX basado en la metodología del Desarrollo Estructurado de Jackson (JSD, Jackson Structured Development) [Jackson (1992)]. Con la ayuda de esta herramienta se pueden mostrar las diferentes relaciones que

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

existen entre los objetos del sistema y las estructuras de datos utilizadas por el sistema MinX.

El JSD son metodologías centro-fuera (middle-out), que hace menos énfasis en la parte funcional y comienza el análisis del sistema como un proceso de modelamiento. Modelan el mundo real con un enfoque parcialmente orientado por objetos, donde se utilizan métodos basados en las estructuras de datos [Sommerville (1995)]. La metodología JSD es útil para explicar las relaciones lógicas que existen entre los diferentes objetos desarrollados dentro del sistema MinX,

El Sistema MinX utiliza una metodología de programación orientada a objetos (OOP, Object Oriented Programming), y el JSD muestra la relación entre los diferentes objetos utilizados por el sistema. Para la implementación del sistema se utiliza el lenguaje de programación Java (jdk1.9.0_07) [Sun Microsystems Java (2008)]. Para la salida y explotación de información se utiliza la base de datos MySQL Server 5.0 [Sun Microsystems MySQL (2008)] [Maslakowoski (2001)], así como un conector de base de datos J/JDBC (mysql-connector-java-5.1.6) [Sun Microsystems JDBC (2008)] [Melton y Eisenberg (2002)] para la comunicación entre el programa y la base de datos.

El modelado del Sistema MinX se explica en dos etapas; en la primera etapa se muestra un modelo general que explica la relación entre el programa principal y los diferentes objetos que intervienen en el ciclo de vida del sistema, en la segunda etapa se muestra la relación individual que tiene cada objeto con respecto al sistema, así como la función que desempeña cada uno de ellos dentro del sistema.

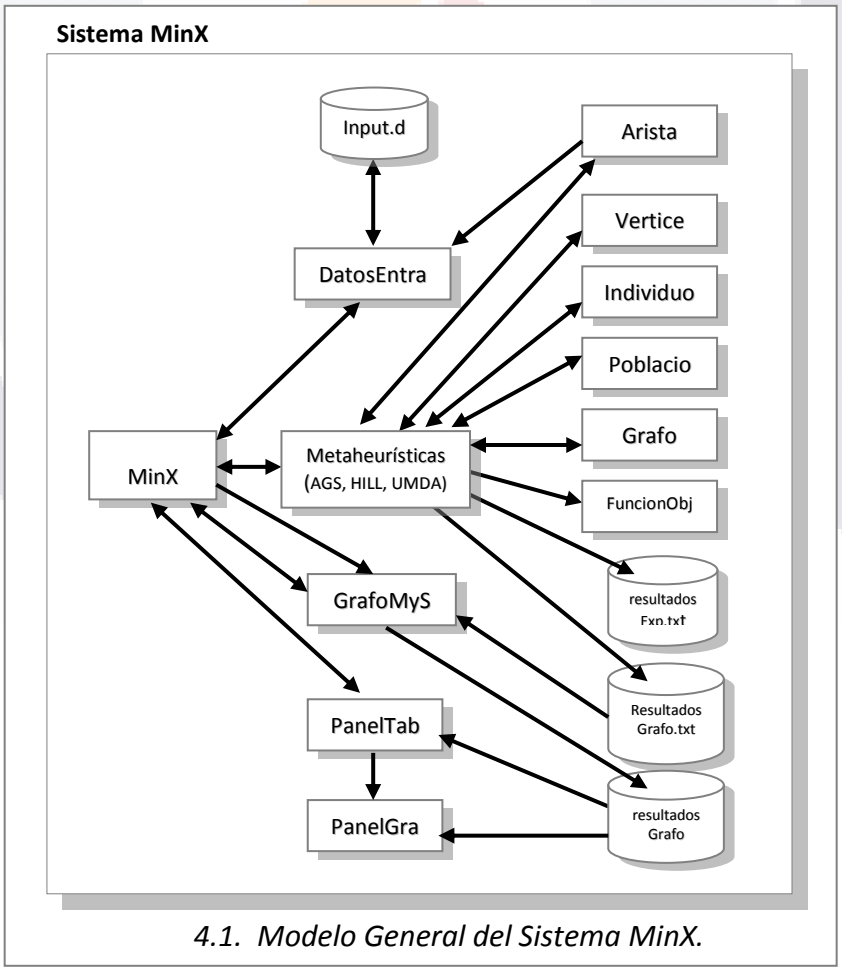
4.3.1. Modelado del sistema MinX

El modelo global del sistema MinX esta basado en un objeto o programa principal llamado MinX y doce objetos que realizan una función específica dentro del sistema. El objetivo de este diagrama es mostrar en forma general el funcionamiento lógico del sistema y las relaciones que existen entre los diferentes elementos del sistema, es decir, las relaciones que existen entre los propios objetos y las relaciones que existen entre los objetos y las bases de datos del sistema.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El diagrama general del Sistema MinX se muestra en la figura 4.1, en el se pude ver la relación que existe entre los diferentes componentes del sistema. En esta figura se observa que el objeto el objeto MinX tiene una relación directa con cuatro de los doce objetos del sistema (*DatosEntrada, Metaheurísticas, GrafoMySQL y PanelTabla*). A excepción del objeto *PanelGraph* los seis objetos restantes (*Arista, Vertice, Individuo, Poblacion, Grafo y FuncionObj*) tienen una relación directa con las metaheurísticas del sistema. El objeto *Metaheurísticas* graba directamente los resultados obtenidos en dos archivos de texto (*resultadosExp.txt* y *resultadosGrafo.txt*) del sistema. El objeto *GrafoMySQL* toma la información del archivo de texto *resultadosGrafo.txt* y la convierte a un archivo de base de datos de MySQL (*resultadosGrafo*). Por último se puede ver que los resultados de la minimización de cruces es mostrada en una interfase grafica de usuario por medios de los objetos *PanelTabla* y *PanelGraph* que lee directamente la información de la base de datos *resultadosGrafo* de MySQL.



4.1. Modelo General del Sistema MinX.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

4.3.2. Objetos del Sistema MinX

En esta parte de la tesis se describe el funcionamiento individual de cada uno de los objetos que comprenden el Sistema MinX y su relación con los demás objetos y las bases de datos del sistema.

- *MinX*: es el programa principal del sistema, este contiene la información inicial con la que se ejecuta el programa, como puede ser, el tipo de metaheurística que se va a utilizar para resolver el problema de la minimización de cruces, y sus parámetros de ejecución.

- *DatosEntrada*: este objeto es utilizado por el programa principal MinX para leer el archivo de entrada *input.dat*. Este archivo contiene la información del problema que se pretende resolver. De este archivo que se obtienen la información del total de vértices y el total de aristas, que posteriormente serán utilizados por una de las metaheurística del sistema, además, este archivo contiene la información de la relación que existe entre los vértices que forman cada una de las aristas del grafo.

- *AGS*: este objeto se utiliza para resolver el problema de la minimización de cruces de las aristas de un grafo mediante la metodología de los algoritmos genéticos. El algoritmo comienza con la generación de la población inicial, generando aleatoriamente las *coordenadas* de los vértices de cada individuo de la población (*coordenadas x, y de pantalla*). Estas coordenadas forman parte del cromosoma de cada individuo, que es el fenotipo que se utiliza para realizar el cruce y la mutación de cada uno de los individuos de la población. Este objeto genera poblaciones de tipo *Grafo* los cuales son evaluados por medio de una función objetivo (*FuncionObj*) para determinar el número de cruces que existen entre sus aristas durante el ciclo de vida del algoritmo genético. El objeto *AGS* siempre se queda con la mejor solución encontrada de cada ejecución del programa y la guarda en los archivos *resultadosExp* y *resultadosGrafo*, que serán utilizados posteriormente.

- *HILL*: este objeto se usa también, para resolver el problema de la minimización de cruces de las aristas de un grafo, mediante el la metodología del Escalador de Colinas de primer ascenso. El algoritmo genera aleatoriamente las *coordenadas* de los vértices de un sólo

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

individuo (*coordenadas x, y*). Estas coordenadas forman el cromosoma del individuo que es el fenotipo utilizado al cual se la aplica una de las dos variantes (*intercambio de vértices o reinició aleatorio del algoritmo*) que tiene el algoritmo durante el ciclo de evaluaciones del algoritmo. Este objeto genera un solo individuo de tipo *Grafo* el cual es evaluado por medio de la función objetivo (*FuncionObj*) que determina el número de cruces que existen entre las aristas del grafo. El objeto *HILL* siempre guarda la mejor solución de cada ejecución del programa y la guarda en los archivos *resultadosExp* y *resultadosGrafo*, que serán utilizados posteriormente.

- *UMDA*: al igual que las demás metaheurísticas, este objeto es utilizado para resolver el problema de la minimización de cruces de las aristas de un grafo, mediante el método de la estimación de la distribución marginal univariada. El algoritmo comienza su ciclo con la generación de la población inicial, generando aleatoriamente las *coordenadas* de los vértices de cada individuo de la población (*coordenadas x, y*). Estas coordenadas forman parte del cromosoma de cada individuo, que es el fenotipo que se utiliza para crear la distribución de probabilidad de cada variable $x_1, x_2,\ldots, x_n$ y $y_1, y_2,\ldots, y_n$ de los individuos de la población. Este objeto al igual que objeto *AGS*, genera poblaciones de tipo *Grafo* los cuales son evaluadas por medio de la función objetivo (*FuncionObj*) para determinar el número de cruces que existen entre sus aristas durante su el ciclo de vida. El objeto *AEDMU* siempre guarda la mejor solución encontrada en cada ejecución del programa y la escribe en los archivos *resultadosExp* y *resultadosGrafo*, que se utilizaran posteriormente.

- *Poblacion*: este objeto contiene información referente a cada individuo de la población, además, guarda la información general de cada generación que fue generada por el objeto AGS o por el objeto AEDMU, como es, el total de cruces realizados entre los individuos de la población, el total de mutaciones, la suma total de fitness, un índice al mejor y peor individuo de la población, así como una array de objetos tipo Individuo que forman parte de la población.

- *Individuo*: este objeto contiene individuos que pertenecen a una población generada por los objetos AGS o AEDMU. Entre los atributos que cada individuo tiene están, el valor

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

del genotipo, el valor de la función objetivo, un arreglo de tipo Vertice que es utilizado como valor del fenotipo de cada individuo, entre otros atributos..

- *Grafo*: el objeto *Grafo* contiene los atributos de cada instancia generada que es utilizada para la visualización de los grafos generados por las diferentes metaheurísticas. Entre sus atributos están, un objeto tipo vértice que contiene los vértices generados aleatoriamente por cada metaheurística, un objeto de tipo arista que contiene las aristas conectadas en el grafo, el nombre del grafo, etc., este objeto contiene las características reales de un grafo, esto significa, que puede ser tratado como un grafo en su conjunto.

- *FuncionObj*: este objeto es utilizado por las diferentes metaheurísticas del sistema para saber la adaptabilidad que cada individuo tiene con respecto a la minimización de cruces en las aristas de un grafo. Entre las operaciones que realiza el objeto *FuncionObj* para conocer el número de cruces que hay entre sus aristas están, la pendiente de la recta para saber si dos rectas están empalmadas o nunca se cruzan, la ecuación general de la recta que se utiliza para conocer la ecuación de la recta que cada arista tiene, y el sistema de ecuaciones con dos incógnitas para determinar en que punto dos rectas se cruzan.

- *Arista*: este objeto tiene los atributos de cada arista del grafo, como son, vértice uno y vértice dos que forman una arista, así como el valor de la pendiente de la arista y su ecuación de la recta.

- *Vertice*: este objeto contiene las coordenadas (*x, y del plano cartesiano*) de cada vértice generado aleatoriamente por las diferentes metaheurísticas del sistema.

- *GrafoMySQL*: este objeto sirve para generar la base de datos en MySQL de los grafos generados en la salida del archivo de texto (resultadoGrafo.txt), el cual contiene el número de grafo, el total de cruces y las coordenadas de cada vértice del grafo. Este objeto convierte el archivo de texto en tablas de MySQL utilizando un conector JDBC, estas tablas son utilizadas por el objeto *PanelTabla* y el objeto *PanelGraph* para la visualización de la interfase gráfica.

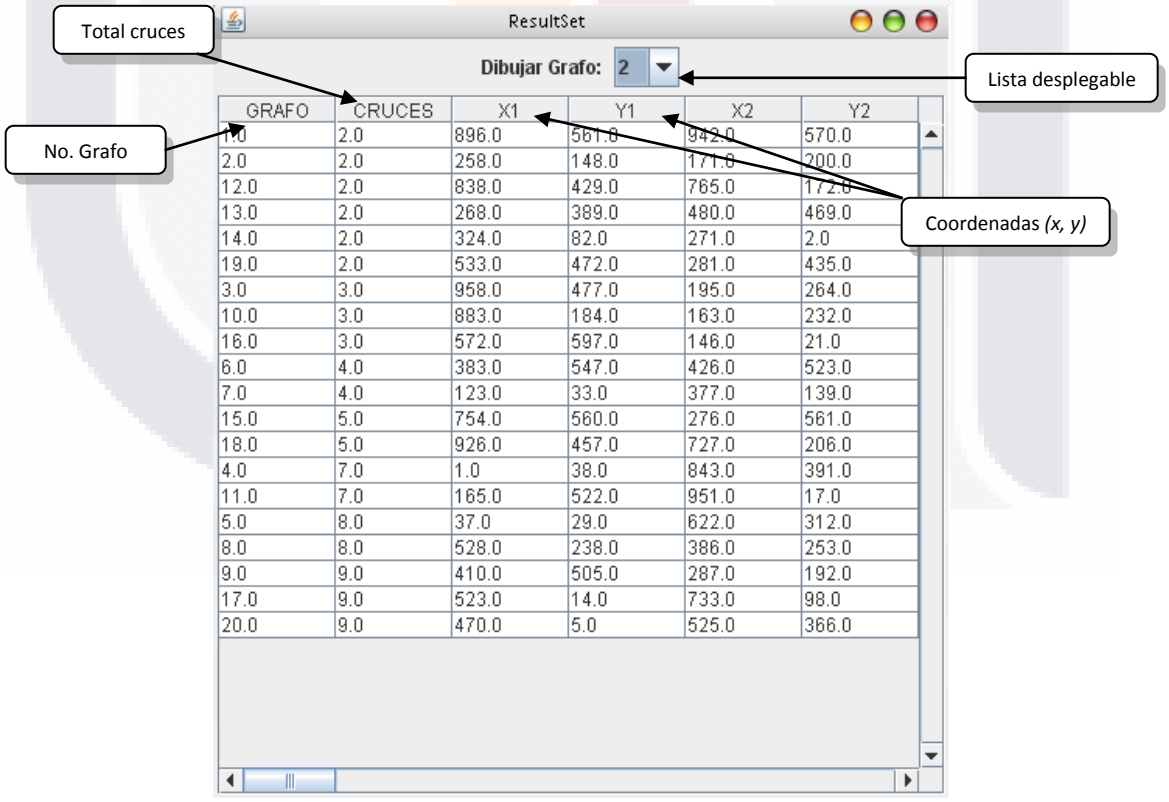
- *PanelTabla*: este objeto es utilizado para mostrar en una ventana los resultados de los grafos generados por las diferentes metaheurísticas del sistema. Esta ventana puede

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

ordenar descendentemente los diferentes campos de la tabla haciendo doble clic en el encabezado de cada campo, esta característica es útil para ordenar descendentemente el campo de *número de cruces*, con el propósito de ver el total de cruces encontrados de cada grafo, del menor número de cruces al mayor número de cruces, de forma se visualiza el valor óptimo de cada grafo del menor al mayor. Además, se puede seleccionar cualquier grafo contenido en esta ventana haciendo clic en la lista desplegable de esta ventana.

- La figura 4.2 muestra la salida visual de la tabla de resultados que el usuario ve. En la figura se puede ver que la tabla contiene los resultados de la minimización de cruces después de 20 ejecuciones para el grafo Hobbs usando el algoritmo híbrido HUx. Aquí se muestra como los resultados se ordenaron descendentemente por su valor óptimo (*campo de cruces*), y además, la selección de la salida gráfica de cualquier grafo se hace por medio de la lista desplegable.



GRAFO	CRUCES	X1	Y1	X2	Y2
1.0	2.0	896.0	561.0	942.0	570.0
2.0	2.0	258.0	148.0	171.0	200.0
12.0	2.0	838.0	429.0	765.0	172.0
13.0	2.0	268.0	389.0	480.0	469.0
14.0	2.0	324.0	82.0	271.0	2.0
19.0	2.0	533.0	472.0	281.0	435.0
3.0	3.0	958.0	477.0	195.0	264.0
10.0	3.0	883.0	184.0	163.0	232.0
16.0	3.0	572.0	597.0	146.0	21.0
6.0	4.0	383.0	547.0	426.0	523.0
7.0	4.0	123.0	33.0	377.0	139.0
15.0	5.0	754.0	560.0	276.0	561.0
18.0	5.0	926.0	457.0	727.0	206.0
4.0	7.0	1.0	38.0	843.0	391.0
11.0	7.0	165.0	522.0	951.0	17.0
5.0	8.0	37.0	29.0	622.0	312.0
8.0	8.0	528.0	238.0	386.0	253.0
9.0	9.0	410.0	505.0	287.0	192.0
17.0	9.0	523.0	14.0	733.0	98.0
20.0	9.0	470.0	5.0	525.0	366.0

Figura 4.2. Tabla de resultados del grafo Hobbs minimizado por la metaheurística híbrida HUx.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

- *PanelGraph*: este objeto es utilizado por el objeto *PanelTabla* para mostrar en una ventana el diseño final de un grafo en particular. En esta ventana, el usuario puede ver como se resolvió el problema del dibujado automático del grafo, ya que esta ventana muestra la topología final del grafo. Cada grafo dibujado automáticamente contiene la información visual del número mínimo de cruces obtenidos por la minimización de una de las metaheurísticas del sistema. La figura 4.3 muestra el dibujado automático del grafo Hobbs utilizando la metaheurística híbrida HUx y dibujado por el sistema MinX. En esta figura se puede ver el resultado de la minimización de cruces en las aristas del grafo número dos, las flechas en color azul indican los cruces que no pudieron ser resueltos en este grafo.

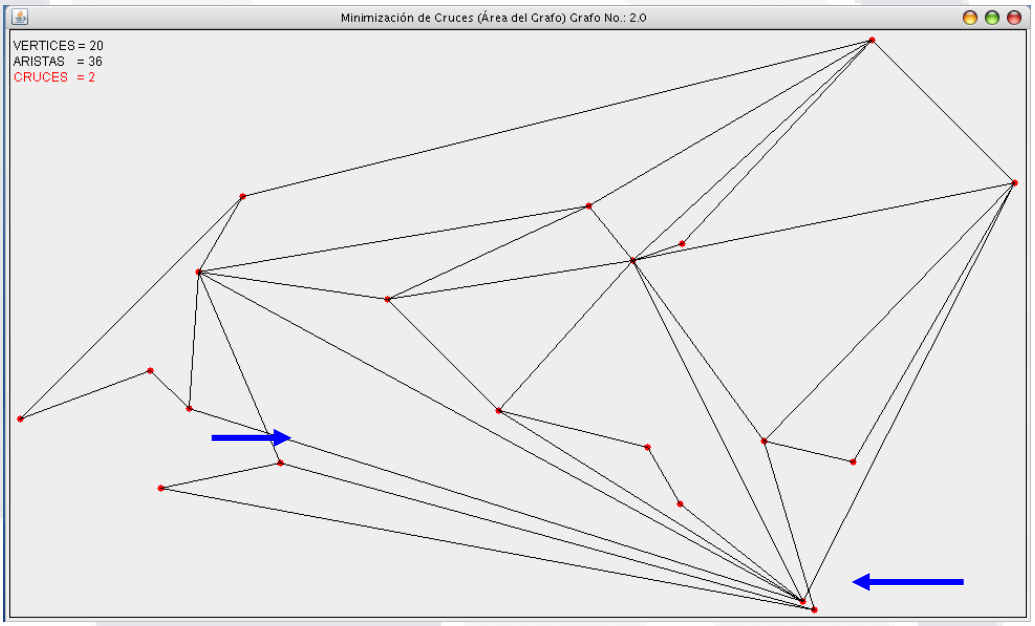


Figura 4.3. Dibujado automático del grafo Hobbs minimizado por la metaheurística híbrida HUx.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 5. DISEÑO DE EXPERIMENTOS Y ANALISIS DE RESULTADOS

Todos los algoritmos vistos en este trabajo de tesis son útiles para resolver en forma general y flexible el problema del dibujado de grafos en la minimización de cruces de las aristas de un grafo, por lo tanto, todos tienen la flexibilidad para encarar este problema. Ahora es necesario comparar los algoritmos experimentalmente para conocer la calidad de los resultados obtenidos de cada uno de ellos, y así poder definir, cual es el mejor algoritmo que resuelve este problema.

En este capítulo se presenta el diseño de los experimentos realizados a todos los algoritmos creados en este trabajo de tesis. También, se hace un análisis de los resultados obtenidos con diferentes grafos planares y no planares, que son considerados como parte del estado del arte. Estos grafos fueron los sujetos de medición con los que se trabajó, y a los que se les aplicaron las diferentes metaheurísticas implementadas en esta tesis.

5.1 Diseño de experimentos

Para hacer una medición de la calidad con la que cuenta cada una de las metaheurísticas implementadas en esta tesis y reportar cual de ellas tiene los mejores resultados, fue necesario diseñar seis experimentos con un grupo específico de parámetros. Los seis experimentos fueron aplicados a cada uno de los grafos estudiados, es decir, se realizaron 42 experimentos (*6 experimentos x 7 grafos*) en total. A continuación se describe el diseño de cada experimento y los resultados obtenidos por cada uno de ellos.

5.1.1. Diseño del experimento # 1: AGS

Algoritmo Genético Simple: se utilizó una selección con elitismo como se describe en el Capítulo 3, con un ciclo de reproducción de 500 generaciones, una población de 40 individuos, una probabilidad de cruzamiento de 90% y una probabilidad de mutación de 10%. Estos parámetros fueron los que permitieron obtener los mejores resultados para este algoritmo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

5.1.2. Diseño del experimento # 2: EC con intercambio de vértices

Algoritmo Escalador de Colinas con intercambio de vértices: este algoritmo utiliza iteraciones (*ciclos*) en la búsqueda de la mejor solución, por lo que se trabajó con 20,000 iteraciones. Se tomó como base el escalador de colinas de primer acenso y se utilizó la variante de intercambio de de vértices descrita en el Capítulo 3 de esta tesis.

5.1.3. Diseño del experimento # 3: EC con reinicio aleatorio

Algoritmo Escalador de Colinas con reinicio aleatorio: este algoritmo utiliza también al igual que en el experimento dos, iteraciones para la búsqueda de la solución, por lo que se trabajó con 20,000 iteraciones. Se tomó como base el escalador de colinas de primer acenso y se utilizó la variante de reinicio aleatorio del algoritmo descrita en el Capítulo 3 de esta tesis.

5.1.4. Diseño del experimento # 4: AEDMU

Algoritmo de la Estimación Marginal Univariada: para este algoritmo se utilizó un total de 200 generaciones, el tamaño de la población fue de 100 individuos y el porcentaje de truncamiento fue del 50% de la población. Estos parámetros fueron los que obtuvieron los mejores resultados para este algoritmo. Los detalles del funcionamiento del algoritmo, fueron descritos en el Capítulo 3 de esta tesis.

5.1.5. Diseño del experimento # 5: HUx

Algoritmo Híbrido HUx: este algoritmo utiliza el método de elitismo para no perder la mejor solución encontrada, además, utiliza 10 ciclos del Escalador de Colinas con reinicio aleatorio. Los parámetros del algoritmo son, un total de 200 generaciones, tamaño de la población de 100 individuos y un porcentaje de 50% de truncamiento de la población. Esta es la mejor combinación de parámetros que obtienen los mejores resultados del algoritmo. Los detalles funcionales del algoritmo, fueron descritos en el Capítulo 3 de esta tesis.

5.1.6. Diseño del experimento # 6: Grafo Compuesto

Grafo compuesto: en el caso de este grafo se utilizaron 800 generaciones, un tamaño de población de 100 individuos, con una probabilidad de cruzamiento del 90% y el 10% de mutación, para los algoritmos basados en poblaciones (*AGS, AEDMU y HUx*). En el caso

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

del algoritmo EC se realizaron 80,000 evaluaciones (800 generaciones x 100 individuos) en la búsqueda de la mejor solución. Estos parámetros fueron los que obtuvieron los mejores resultados para cada algoritmo. El grafo compuesto se comenta más adelante en el apartado 5.2.3 de esta tesis.

Se trabajo de manera general con un total de 3,600, 20,000 y 80,000 evaluaciones (ciclos), así como, un total de 20 ejecuciones independientes de cada algoritmo. El propósito de esta medida es, comparar los resultados obtenidos por nuestros algoritmos, contra los resultados obtenidos de un grupo de trabajos tomados de la literatura en los que se dibujan grafos utilizando métodos de propósito general como los algoritmos genéticos, el escalador de colinas y el algoritmo de la distribución marginal univariada. Por ejemplo, el grafo llamado Simple [Rossete y Ochoa (1998)] utiliza 3,600 evaluaciones por cada ejecución. Para poder compararlo con los algoritmos de esta tesis, se necesitaron 60 generaciones x 60 individuos, para los algoritmos basados en poblaciones (*AGS, AEDMU y HUx*).

5.2 Análisis de resultados

En este apartado se comparan los resultados obtenidos entre los diferentes algoritmos implementados en esta tesis, además, se hace en análisis de los resultados obtenidos por el algoritmo híbrido HUx y algunos de los trabajos reportados en la literatura que son considerados como parte del estado del arte. Para pode llevar acabo los dos tipos de comparaciones antes descritas, se seleccionaron algunos de los trabajos que permitían hacer una medición de resultados en el problema del dibujado de grafos, específicamente en la minimización de cruces de las aristas de un grafo [Branke et. al. (1997)], [Eleoranta y Mäkinen (2001)], [Hobbs y Rodgers (1998)], [Rossete y Ochoa (1998)], [Rossete (2000)]. Estos trabajos son considerados benchmarks en esta tesis, y se tomaron como base para poder llevar acabo dicha comparación.

Los aspectos generales de todos los grafos investigados en esta tesis se muestran en la tabla 5.1. En esta tabla se puede ver el total de vértices y de aristas de cada grafo, así como, una columna de densidad, que es la razón entre el número de aristas existentes en el grafo

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

con respecto al máximo posible, que sería el del grafo completo Kp, con el mismo número de vértices p [Rossete (2000)].

Grafo	Vértices (p)	Aristas (q)	Densidad
Simple	8	12	0.429
Herschel	11	18	0.327
Hobbs	20	36	0.189
Árbol	34	33	0.059
Estrella	17	40	0.294
Rejilla 4 x 4	16	24	0.200
Compuesto	40	69	0.088

Tabla 5.1. Propiedades Generales de los grafos tomados de la literatura.

El grafo llamado *Simple* se tomó de [Rossete (1998)] y es el grafo más simple de todos, este grafo tiene 8 vértices y 12 aristas. El grafo llamado *Herschel* se tomó del trabajo de [Branke et. al. (1997)], este grafo tiene 11 vértices y 18 aristas. De este mismo trabajo también se tomaron los grafos *Árbol* con 34 vértices y 33 aristas y *Estrella* con 17 vértices y 40 aristas. El grafo llamado *Hobbs* se tomó del trabajo de [Hobbs y Rodgers (1998)], este grafo tiene 20 vértices y 36 aristas, la característica de este grafo es que es un grafo no planar. El grafo llamado *Rejilla* se tomó de [Eleoranta y Mäkinen (2001)], este grafo tiene 16 vértices y 24 aristas, este grafo corresponde a una rejilla de 4 x 4. El grafo *Compuesto* es el grafo más grande de todos, este tiene un total de 40 vértices y 69 aristas. Este es un grafo propio, creado para probar la estabilidad del algoritmo híbrido HUx ante grafos con un mayor número de vértices y aristas que los descritos anteriormente. El grafo *Compuesto* se construyó con los vértices y las aristas de los grafos Hobbs [Hobbs y Rodgers (1998)] y Ebner [Ebner (2005)]. El grafo llamado Ebner tiene un total de 20 vértices y 33 aristas, este grafo se tomó de la literatura y sólo se utilizó para construir el grafo Compuesto.

Como una característica general de todos los grafos descritos en este trabajo de tesis, se puede comentar que las dimensiones de todos estos grafos son frecuentes en programas de visualización de grafos. Otra característica común es que la mayoría de los grafos tomados de la literatura utilizaron a los Algoritmos Genéticos como método de optimización de propósito general, con excepción del grafo *Simple* que utilizó el *Escalador de Colinas Estocástico* [Rossete (2000)] como método de optimización.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

5.2.1. Comparación de resultados entre los algoritmos implementados en esta tesis

Todos los algoritmos implementados en esta tesis se ejecutaron con el mismo número de iteraciones para todos los grafos tomados de la literatura, 3,600 evaluaciones en el caso del grafo Simple y 20,000 evaluaciones en los demás casos. Además, se realizaron 20 ejecuciones independientes para cada algoritmo.

El promedio del número de cruces encontrados por las metaheurísticas implementas en este trabajo de tesis, muestran que el algoritmo híbrido HUx fue superior a todos los demás algoritmos. La tabla 5.2 muestra el promedio general de cruces encontrados por cada algoritmo, así como también, el promedio general de cruces encontrados para cada grafo tomado de la literatura.

	Promedio de cruces encontrados					
	AGS	AEDMU	HUX	EC (intercambio)	EC (reinicio)	
SIMPLE	0.00	0.00	0.00	1.65	0.00	0.33
HERSCHEL	1.70	2.05	0.40	4.65	0.75	1.91
ARBOL	7.70	1.05	0.25	2.35	1.20	2.51
HOOBS	11.35	9.90	4.80	12.20	6.15	8.88
ESTRELLA	12.60	8.20	0.55	10.70	0.70	6.55
REGILLA 4X4	1.65	1.10	0.15	3.65	0.80	1.47
	5.83	3.72	1.03	5.87	1.60	

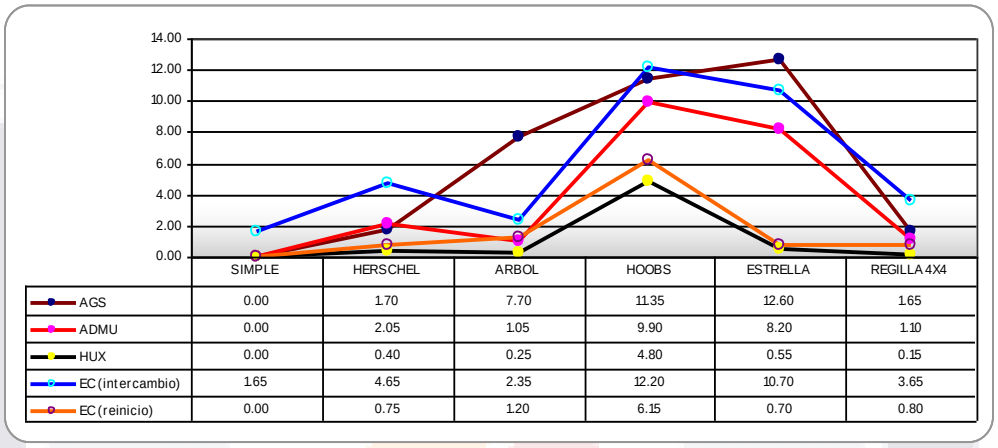
Tabla 5.2. Número promedio de cruces obtenidos para los grafos tomados de la literatura.

En esta tabla se ve como el algoritmo híbrido HUx en su mayoría promedia poco menos de 1 cruce por grafo para las 20 ejecuciones realizadas a cada a cada uno de ellos, seguido por el EC con reinicio aleatorio. Así mismo, se puede apreciar que el algoritmo híbrido HUx es el mejor de todos los algoritmos con un promedio general de 1.03 cruces por grafo, mientras que el peor de ellos es el algoritmo EC con intercambio de cruces, con un promedio general de 5.87 cruces por grafo. Haciendo otro análisis de los resultados obtenidos, se puede ver que el grafo llamado Hobbs es el grafo más difícil de todos, con un promedio general de 8.88 cruces encontrados por todas la metaheurísticas implementadas, mientras que el grafo llamado Simple, es el grafo más fácil de todos, con un promedio

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

general de 0.33 cruces encontrados por cada una de las metaheurísticas utilizadas en esta tesis. La figura 5.1 muestra en forma gráfica el promedio de cruces obtenidos en cada uno de los grafos por las diferentes metaheurísticas implementadas en este trabajo de tesis. En esta gráfica se ve claramente que el algoritmo HUx (negro) reduce mejor el número de cruces que sus contrapartes.



	SIMPLE	HERSCHEL	ARBOL	HOOBS	ESTRELLA	REGILLA 4X4
AGS	0.00	1.70	7.70	11.35	12.60	1.65
ADMU	0.00	2.05	1.05	9.90	8.20	1.10
HUX	0.00	0.40	0.25	4.80	0.55	0.15
EC (intercambio)	1.65	4.65	2.35	12.20	10.70	3.65
EC (reinicio)	0.00	0.75	1.20	6.15	0.70	0.80

Figura 5.1. Promedio de cruces obtenidos en cada grafo.

La figura 5.2 muestra una gráfica de columnas donde se puede ver el porcentaje general obtenido en la minimización de cruces de las aristas a los grafos tomados de la literatura. Aquí se ve como el algoritmo hibrido HUx supera a los demás métodos utilizados para reducir el número de cruces.

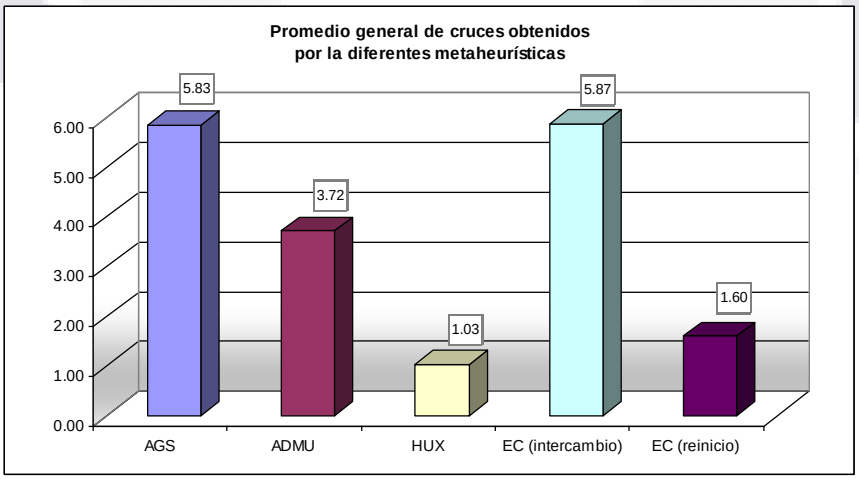


Figura 5.2. Promedio general de cruces obtenidos por las diferentes metaheurísticas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Continuando con el análisis de los resultados de los algoritmos implementados en esta tesis, en seguida, se muestra la frecuencia de aparición del óptimo reportado por cada uno de los grafos. La frecuencia de aparición es el número de veces que el algoritmo alcanzó el óptimo. En la tabla 5.3 se muestra la frecuencia de aparición del óptimo alcanzado en las 20 evaluaciones independientes que se corrieron de cada algoritmo.

	Frecuencia de aparición del óptimo				
	AGS	**AEDMU**	**HUX**	**EC (intercambio)**	**EC (reinicio)**
SIMPLE	20	20	20	1	20
HERSCHEL	8	2	16	0	14
ARBOL	0	7	15	5	8
HOOBS	0	0	6	0	3
ESTRELLA	0	2	17	0	16
REGILLA 4X4	6	9	17	5	12
	28%	**33%**	**76%**	**9%**	**61%**

Tabla 5.3. Número de veces que el algoritmo encuentra el óptimo de un total de 20 evaluaciones.

En la tabla 5.4 se puede ver que la frecuencia de aparición del óptimo del algoritmo HUx supera a todos los demás algoritmos; en algunos casos en más del 50%. La siguiente tabla muestra el porcentaje de aparición del óptimo y en ella se puede ver que el algoritmo HUx tiene una efectividad del 76% en los resultados obtenidos, contra un 61% del EC con reinició aleatorio, esto es una diferencia de un 15% más de efectividad que el segundo mejor de todos los algoritmos.

	Promedio en la Frecuencia de aparición del óptimo				
	AGS	**AEDMU**	**HUX**	**EC (intercambio)**	**EC (reinicio)**
SIMPLE	100%	100%	100%	5%	100%
HERSCHEL	40%	10%	80%	0%	70%
ARBOL	0%	35%	75%	25%	40%
HOOBS	0%	0%	30%	0%	15%
ESTRELLA	0%	10%	85%	0%	80%
REGILLA 4X4	30%	45%	85%	25%	60%
	28%	**33%**	**76%**	**9%**	**61%**

Tabla 5.4. Número de veces que el algoritmo encuentra el óptimo de un total de 20 evaluaciones.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

La tabla 5.4 muestra que, el porcentaje de aparición del óptimo del algoritmo HUx, se encuentra entre el 100% y el 75% de efectividad, en los grafos Simple, Herschel, Árbol, Estrella y Rejilla, sólo en el grafo Hobbs, que es el grafo con mayor dificultad de todos, se obtiene un 30% de efectividad, pero aún así, se supera en un 50% la efectividad del algoritmo EC con reinicio aleatorio, que es el segundo mejor de todos.

Por ultimo se debe mencionar que el alcance de los objetivos, que en nuestro caso es, aquel dibujado que tenga el número mínimo de cruces de las aristas del grafo, se alcanzó en su totalidad por el algoritmo híbrido HUx, así como también, por el algoritmo EC con reinicio aleatorio. La búsqueda de la solución óptima para todos los grafos que fueron tomados de la literatura, es de cero cruces en las aristas de sus grafos; sólo en el grafo llamado Hobbs que es no planar, la solución óptima es de 2 cruces en las aristas de este grafo. La tabla 5.5 muestra el alcance de los objetivos realizada por las diferentes metaheurísticas a cada uno de los grafos de la literatura.

	Alcance de los Objetivos				
	AGS	AEDMU	HUX	EC (intercambio)	EC (reinicio)
SIMPLE	0	0	0	0	0
HERSCHEL	0	0	0	2	0
ARBOL	2	0	0	0	0
HOOBS	5	3	2	5	2
ESTRELLA	2	0	0	2	0
REGILLA 4X4	0	0	0	0	0

Tabla 5.5. Alcance de los objetivos logrados por las diferentes metaheurísticas.

Al momento de analizar los resultados obtenidos entre los algoritmos AEDMU y el EC con reinicio aleatorio, se hace evidente que el alcance de sus objetivos son muy parecidos, tomando en cuenta esta característica se toma la decisión de implementar un nuevo algoritmo que tuviera las bondades de estos dos algoritmos, con la finalidad de superar los resultados que hasta ese momento se tenían por estos dos algoritmos, es de esta manera que se implementa el algoritmo híbrido HUx.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

5.2.2. Comparación de resultados obtenidos por algoritmo híbrido HUx y los resultados obtenidos de los algoritmos tomados de la literatura

A continuación se comparan los resultados obtenidos por el algoritmo híbrido HUx y algunos de los trabajos reportados en la literatura que forman parte del estado del arte del dibujado de grafos para la minimización de cruces en sus aristas. La manera de presentar los resultados en este apartado de la tesis, diferirá de la forma en que fueron presentados los resultados en índice anterior, esto es por que, la mayoría de los trabajos que aquí se citan no permiten una comparación directa, ya que en muy pocos casos se presentan resúmenes estadísticos. Los autores sólo hacen una descripción y muestran algunos de los resultados que se obtuvieron. Por esta razón, la manera de comparar los resultados para cada grafo se hará por separado.

Para el grafo Simple presentado en el trabajo de [Rossete y Ochoa (1998)], obtiene un 93% de dibujados sin cruces después de 3,600 evaluaciones (60 generaciones x 60 individuos). Para el mismo grafo presentado por [Rossete (2000)], el Escalador de Colinas Estocástico (ECE), obtiene un 100% de dibujados sin cruces después de 550 evaluaciones, mientras que el algoritmo híbrido HUx, obtiene el 100% de dibujados sin cruces después de 252 evaluaciones en promedio, de un total de 20 ejecuciones independientes. Este resultado es mejor ya que utiliza sólo la mitad de las evaluaciones con respecto al ECE para alcanzar el objetivo.

El trabajo de [Hobbs y Rodgers (1998)], se obtiene una solución con cuatro cruces después de 20,000 evaluaciones (1000 generaciones x 20 individuos). El ECE de [Rossete (2000)], obtiene dos soluciones con tres cruces, cuatro soluciones con cuatro cruces y tres soluciones con cinco cruces en las 20 ejecuciones, no se dan detalles del número de evaluaciones necesitadas para alcanzar la solución. El algoritmo híbrido HUx presenta seis soluciones con dos cruces, tres soluciones con tres cruces, dos soluciones con cuatro cruces y dos soluciones con cinco cruces después de 9,034 evaluaciones en promedio en las 20 ejecuciones del algoritmo. Estos resultados son muy superiores a los resultados obtenidos

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

por Hobbs y Rossete, ya que se obtuvieron seis soluciones con dos cruces lo que el ECE no pudo obtener.

Los resultados presentados por [Branke et. al. (1997)] para el grafo llamado Árbol, obtienen sólo una solución sin cruces. El ECE de [Rossete (2000)], obtuvo diez soluciones sin cruces, seis soluciones con un cruce después de 3,000 evaluaciones. El algoritmo híbrido HUx, obtuvo quince soluciones con cero cruces y cinco soluciones con un cruce, en tan sólo en 4,841 evaluaciones en promedio en las veinte ejecuciones. Estos resultado también son favorables al algoritmo hibrido HUx ya que obtuvo cinco soluciones más con cero cruces que el ECE.

Para el grafo llamado Estrella de [Branke et. al. (1997)], los autores obtuvieron una solución sin cruces. El ECE de [Rossete (2000)], obtuvo cinco soluciones sin cruces y dos soluciones con un cruce en las 20 ejecuciones, no se dan detalles del número de evaluaciones necesarias para alcanzar estos resultados. El algoritmo híbrido HUx, obtuvo diecisiete soluciones con cero cruces, una solución con un cruce, una solución con tres cruces y una solución con siete cruces después de 4909 evaluaciones en promedio en las 20 ejecuciones del algoritmo, los resultados obtenidos para este grafo también son muy superiores, ya que el HUx obtuvo doce resultados más con cero cruces que el ECE.

Para el grafo llamado Herschel el ECE de [Rossete (2000)], obtuvo diecisiete soluciones sin cruces después de 3,000 evaluaciones en las 20 ejecuciones. Los autores del trabajo [Branke et. al. (1997)], obtuvieron sólo una solución sin cruces cuando se usa la mutación basada en el algoritmo de los Resortes. Sin embargo, cuando no es usado no puede obtener una sola solución sin cruces. El algoritmo híbrido HUx, obtuvo 16 soluciones sin cruces después de 4,194 evaluaciones en promedio y cuatro soluciones con dos cruces después de 20 ejecuciones del algoritmo. Para este grafo los resultados obtenidos por el ECE y el algoritmo híbrido HUx son muy parecidos, a pesar de que el ECE obtuvo solamente 1 cruce más que el algoritmo híbrido HUx.

Para el grafo llamado Rejilla de [Eleoranta y Mäkinen (2001)], se obtiene una solución sin cruces después de 50,000 evaluaciones (5000 generaciones x 10 individuos). En el trabajo de [Rossete (2000)], el ECE obtiene 12 soluciones sin cruces después de 3,000

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

evaluaciones en las 20 ejecuciones del algoritmo, mientras que en el algoritmo híbrido HUx, se obtuvieron diecisiete soluciones con cero cruces y tres soluciones con un cruces después de 4,018 evaluaciones en promedio en las 20 ejecuciones del algoritmo. Los resultados son otra vez favorables al algoritmo híbrido HUx ya que obtuvo cinco cruces más que el ECE.

5.2.3. Estabilidad del algoritmo híbrido HUx con el grafo Compuesto

Para poner a prueba la estabilidad del algoritmo híbrido HUx fue necesario diseñar un algoritmo más grande que contara con más vértice y arista que los grafos tomados de la literatura. Para esto de diseño un algoritmo con los vértices y las aristas de dos grafos, el grafo Hobbs con 20 vértices y 36 aristas, y el grafo Ebner con 20 vértices y 33 aristas. El grafo que resultó de la combinación de estos dos grafos se le dio el nombre de grafo *Compuesto*. Con esta unión, el *Compuesto* se construyo con 40 vértices y 69 aristas, que es el grafo más grande de todos. Se utilizó el experimento número seis para comparar los resultados obtenidos con el grafo Compuesto de todas la metaheurísticas implementadas en esta tesis. Los resultados se muestran a continuación en la tabla 5.6.

	Grafo *Compuesto*			
	Promedio de Cruces	**Alcance del Objetivo mínimo 3 cruces**	**Frecuencia de Aparición**	**Promedio frecuencia de aparición**
AGS	63	35	0	0%
AEDMU	34	19	0	0%
HUX	11	3	1	5%
EC (intercambio)	23	10	0	0%
EC (reinicio)	10	5	0	0%
	28.22	**14.40**	**0.20**	**1%**

Tabla 5.6. Comparación de resultados del grafo Compuesto.

En la primer columna de la grafica se puede ver que el promedio de cruces es de 11 en algoritmo HUx, mientras que el algoritmo EC con reinicio obtuvo un total de 10 cruces, uno menos que el algoritmo híbrido HUx, lo que lo hace el mejor de todos, sin embargo, este resultado es muy parecido al del algoritmo híbrido HUx.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Por otro lado, se sabe que el número mínimo de cruces del grafo *Compuesto* es de 3 cruces, debido a que, el número mínimo de cruces del grafo *Ebner* es de 1, y el número mínimo de cruces del grafo *Hobbs* es de 2 cruces, como se mostró en la tabla 5.6.

Si se analiza la segunda columna de la tabla 5.6, la única metaheurística que alcanza el objetivo es el grafo híbrido HUx, como lo demuestran las columnas tres y cuatro de la tabla 5.6, aquí se puede ver que la única metaheurística que contribuye a la estadística es el algoritmo hibrido HUx. La figura 5.3 muestra que el número mínimo de cruces sólo fue alcanzado por el algoritmo híbrido HUx.

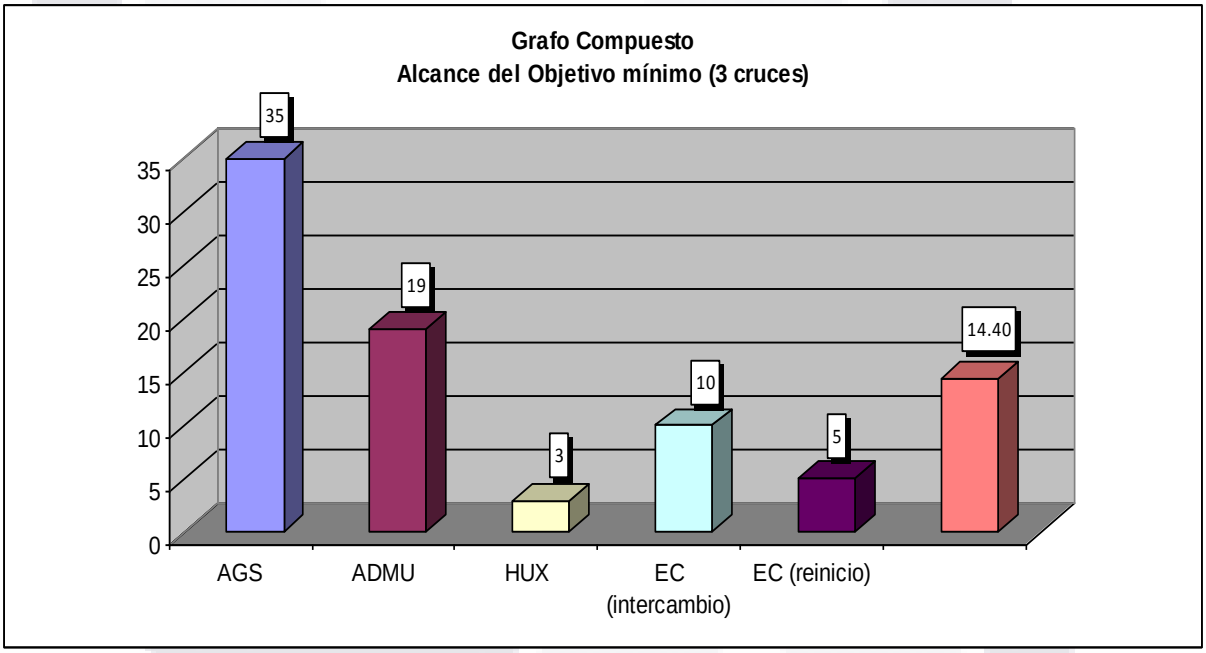


Figura 5.3. Grafo Compuesto, minimizado por el algoritmo híbrido HUx.

La siguiente figura 5.4 muestra el dibujado del grafo Compuesto, minimizado por el algoritmo híbrido HUx. Este fue el único grafo dibujado automáticamente que alcanzo el objetivo de tres cruces. La figura 5.4 muestra un acercamiento del primer cruce del grafo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

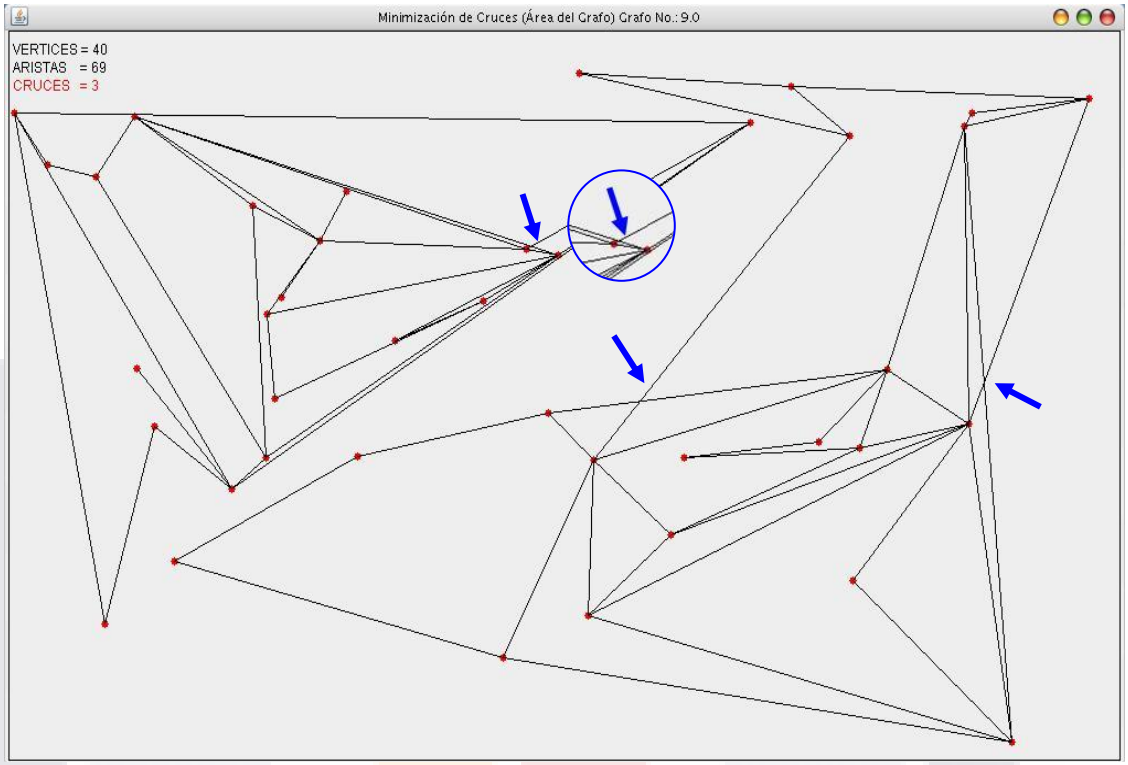


Figura 5.4. Grafo Compuesto visualizado por el Sistema MinX.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Capítulo 6. CONCLUSIONES

En este capítulo se explica en forma general los resultados obtenidos en esta tesis, la importancia de los resultados y las aportaciones generadas por esta investigación, Así mismo, se describe el trabajo futuro y hacia donde puede dirigirse tomando como base este trabajo de investigación.

6.1 Conclusiones generales de la tesis

Dentro de las principales aportaciones que ha generado este trabajo de tesis, está la creación del algoritmo híbrido HUx, que se utiliza en el problema de la minimización del número de cruces de las aristas de un grafo, y la programación del sistema MinX, utilizado para el problema del dibujado automático de grafos.

Se demostró en el capítulo cinco, que estas dos herramientas se han aplicado con éxito en los experimentos realizados, por lo tanto, son eficientes para resolver ambos problemas. Además, en el análisis de los resultados del capítulo cinco, se demostró que estas dos herramientas superaron la calidad de los resultados obtenidos por las diferentes metaheurísticas implementadas en esta tesis, así como, los resultados obtenidos por las demás metaheurísticas que fueron tomadas de la literatura y que forman parte del estado del arte.

El diseño, desarrollo e implementación de una metaheurística híbrida HUx, a sido de suma importancia ya que aprovecha las propiedades mas destacadas de dos metaheurísticas AEDMU que es una metaheurística de búsqueda global, y el algoritmo escalador de colinas estocástico (ECE) que es un algoritmo de la metaheurística de búsqueda local. Las características del AEDMU de captar propiedades globales de las soluciones mediante la estimación de un modelo probabilístico (modelo de independencias) y por otro lado la rapidez y eficiencia mostrada por el algoritmo ECE hacen del algoritmo híbrido HUx un herramienta sumamente eficaz en la búsqueda de la minimización de cruces en las aristas de un grafo.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

El sistema MinX proporciona al usuario una herramienta de visualización de grafos amigable y fácil de usar. El dibujado automático de los grafos ya minimizados, proporciona al usuario la facilidad para comparar los resultados en ventanas independientes, por lo que el usuario puede decidir utilizar el diseño del grafo que le se más útil a sus necesidades.

6.2 Trabajo futuro

Existe un gran potencial de desarrollo en este trabajo de tesis que se deberá continuar el futuro inmediato. Entre las potencialidades de este trabajo futuro están:

- Continuar con la hibridación de nuevos algoritmos para la minimización de cruces en las aristas de un grafo, en este caso utilizando las ventajas del Escalador de Colinas y los Algoritmos de la Estimación de Distribuciones, utilizando alguna de sus variantes de los algoritmos que manejan la interdependencia entre sus variables.
- Pasar al campo de los grafos densos (con cientos de vértices y aristas).
- La búsqueda de nuevos “benchmarks” mas grandes con los que podamos comparar nuestros resultados.
- Cambiar al problema de la legibilidad de grafos multiobjetivo, es decir no sólo la minimización de los cruces entre las aristas de un grafo, sino también otras características estéticas como son la reducción del área de dibujado, la simetría entre sus aristas, la colocación equitativa de los vértices, entre otras.
- Continuar en el desarrollo de la interfase gráfica del sistema MinX, adicionando nuevas herramientas para la exploración, la explotación y la edición de los grafos.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

BIBLIOGRAFÍA

Abdullah A [2007]. Data Mining Using the Crossing Minimization Paradigm, Doctoral Thesis, Department of Computing Science and Mathematics, University of Stirling, Scotland, 142 páginas.

Alvarez, S. (1995). "Metodología de Análisis y Diseño Orientada a Objetos de Sistemas Informáticos", Tesis de Doctorado, ISPJAE Cuba.

Baluja S, Davies S. (1998). Fast Probabilistic Modeling for combinatorial Optimization, Proceedings of 15th National Conference on Artificial Intelligence, AAAI-98, American Association for Artificial Intelligence, Madison, Wisconsin.

Bender E., Williamson G. (2005). Mathematics for Algorithm and Systems Analysis, Basic concepts in graph Theory, ISBN: 0-486-44250-0, Dover Publications, 54 páginas.

Bondy J. A., Murty U. S. R. (1982). Graph Theory With Applications, Department of Combinatorics and Optimization, University of Waterloo, Ontario Canada, ISBN: 0-4444-19451-7, Elsevier Science Publishing Co., Inc., 135-170.

Bracho J. (2003). Introducción Analítica a la Geometría, Capitulo 1, Facultad de Ciencias de la UNAM, 13-81.

Branke, J., Bucher, F., Schmeck, H. (1997). A genetic Algorithm for drawing undirected graphs, Proceedings of 3rd Nordic Workshop on Genetics Algorithms and their Applications, Alander, J. T. (Ed.), 193-206.

Buchheim C., Ebner D., Gutwenger C., Juenger M., Mutzel P. (2004). Crossing and Planarization, Universities of Cologne, Vienna and Dortmund, 33 páginas.

Cerný J. (2008). Combinatorial and Computational Geometric, Doctoral Thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Applied Mathematics, Malostranské nám. 25, Prague, Czech Republic, 29 páginas.

Curino R., Martig S., Castro S. (2005). Visualización de Grafos, VII Workshop de Investigadores en Ciencias de la Computación (WICC 2005), Universidad Nacional de Río

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Cuarto, Facultad de Ciencias Exactas, Físico-Químicas y Naturales, Departamento de computación. 185-187.

Davidson R., Harel D. (1996). Drawing Graphs Nicely Using Simulated Annealing, ACM Transactions on Graphics, Vol. 15, No. 4, 301-331.

De Bonet J., Isbell C., Viola P. (1996). MIMIC: Finding Optima by Estimating Probability Densities, Advances in Neural Processing Systems 97, MIT Press, Cambrige, MA.

Diaz E. (2008). Metaheurísticas Híbridas para el Aprendizaje de Modelos Gráficos Markovianos y Aplicaciones. Tesis para optar por el Grado de Doctor en Ciencias de la Computación. Universidad Autónoma de Aguascalientes, Ags., México (2008).

Diestel R. (2005). Graph Theory, Springer-Verlag Heilderberg, New York, 1-32, 86-110.

Ebner, D. (2005). Optimal Crossing Minimization Using Integer Linear Programming, University Technology of Vienna, 87 páginas.

Edades P. (1984). A Heuristic for Graph Drawing, Congressus Numerantium, Vol. 42, 149-160.

Eleoranta T., Mäkinen E. (2001). A genetic Algorithm for Drawing Undirected Graphs, Departament of Computer and Information Science, University of Tampere Finland, Work supported by the Academy of Finland (Project 35025), 13 páginas.

Enríquez S., Ponce E., Díaz E. (2008). Calibración de un Algoritmo Genético para el Problema de la Minimización de Cruces en las Aristas de un Grafo, Avances en Computación Evolutiva, Memorias del IV Congreso Mexicano de Computación Evolutiva, Centro de Investigación en Matemáticas (CIMAT), ISBN: 968-5733-10-4, 61-66.

Forster M. (2002). Appling Crossing Reduction Strategies to Layered Compound Graphs, M.T. Goodrich and S.G. Kobourov (Eds.): GD 2002, LNCS 2528, © Springer-Verlag Berlin Heidelberg, 276–284.

Garey M. R., Johnson D. S. (1983). Crossing number is NP-completo. SIAM Journal on Algebraic and Discrete Methods, 4, 312–316.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Goldberg E. D. (1989). Genetic Algorithm in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., ISBN: 0-201-15767-5, 411 páginas.

Glover F., Laguna M. (1997). Tabu Search, Kluwer Academic Publisher, Boston, Hardbound.

Hobbs, M. H. W., Rodgers P. J. (1998). Representing Space: A Hybrid Genetic Algorithm for Aesthetic Graph Layout, Frontiers in Evolutionary Algorithms, FEA'98, Proceedings of 4th Joint Conference on Information Sciences, JCIS'98, Vol. 2, 415-418.

Holland, J. (1975). Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 183 páginas.

Horstman, C., Cornell (2000), Core Java Volume II – Advanced Features, Sun Microsystems Press a Prentice Hall Title, 203-249, 319-602.

Jackson M. (1992). The Jackson Development Methods, Wiley Encyclopaedia of Software Engineering, ed J Marciniak, 15 páginas.

Jansen B. J. (1998). The Graphical User Interface: An introduction, SIGCHI Bulletin, 30(2), 22-26.

Kant G. (1993). Algorithms for Drawing Planar Graphs, University Utrecht, Nederlands, ISBN 90-393-0416-5, 227 páginas.

Kovári, Sós, Turán (1954). On a problem of K. Zarankiewicz. Colloquium Math, 3:50-57.

Kurja E., Marks E., Blair A., Waters R. (2001). A short note on the history of graph drawing, proceedings of Graph Drwing 2001, vol. 2265, 272-286.

Larrañaga P., Lozano J. A., Mühelenbeing H. (2003). Algoritmos de Estimación de Distribuciones en Problemas de Optimización Combinatoria, Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial. No. 19, 149-168, ISSN: 1137-3601. © AEPIA (http://www.aepia.org/revista).

Lauritzen, S. L. (1996). Graphical Models. Oxford University Press, USA, 298 páginas.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Lin T. Edades P. (1994). Integration of Declarative and Algorithmic Approaches for Layout Creation, Proceedings of Graph Drawing 94, Lecture Notes in Computer Science 894, Springer.Verlag, 376-387.

Martig, S, Señas, P. (2000). Grafo Integrador de un Mapa Conceptual Hipermedial, Departamento de Ciencias de la Computación, Universidad Nacional de Sur, Bahía Blanca Argentina, 8 páginas.

Maslakowoski M. (2001), Aprendiendo MySQL en 21 Días, Prentice Hall, 10-65, 78-88, 107-154.

Melton, J, Eisenberg, A. (2002), SQL y Java, Guía para SQLJ, JDBC y tecnologías relacionadas, Alfaomega RA-MA, 1-51, 97-325.

Mühlenbein H., Paaß G. (1996). From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV, 178–187.

Mühlenbein H., Mahnig T., Ochoa A. (1998). Schemata Distributions and Graphical Models in Evolutionary Optimization, Journal of Heuristic, Vol. 5, No. 2.

Newton M. (2007). Sequential and Parallel Algorithms for Low-Crossing Graph Drawing, A Doctoral Thesis, University Loughborough, 159 páginas.

Pach J., Tóth G. (2000). Which crossing number is it, anyway? J. Comb. Theory Ser. B, 80(2):225–246.

Pajares G., Santos M. (2006). Inteligencia Artificial e Ingeniería del Conocimiento, Alfa Omega Grupo Editor, RA-MA Editorial, Madrid, España, ISBN 84-7897-676-0, 1-65.

Prieto F. (2003). Sistemas de Ecuaciones con Dos Incógnitas, Facultad de Ingeniería, Universidad de la Palma, Argentina, 24 páginas.

Purchase, H., Cohen, R. F., James, M. (1996). “Validating Graph Drawing Aesthetics”, Proceeding of Graph Drawing’95, Passau, Germany, Branderburg, F. (Ed.) Lecture Notes in Computer Science 1027, Springer-Verlag, 435-446.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Rondón Durán J. E. (2006). Algebra, Trigonometría y Geometría Analítica, Ediciones Hispanoamericanas, Universidad Nacional y Abierta, Ed. UNAD, Bogotá D. C., 347-360.

Rossete A., Ochoa A. (1998). Genetic Graph Drawing, Proceedings of 13th International Conference of Applications of artificial Intelligence in Engineering, AIENG'98, Adey, R. A., Rzevski, G., Nolan, P. (Ed.) Galway, Computational mechanics Publications, 37-40.

Rossete Suárez, A. (2000). Un Enfoque General y Flexible para el Trazado de Grafos, Tesis presentada en opción al grado científico de doctor en Ciencias Técnica, Facultad de Ingeniería Industrial , CEIS, La Habana, Cuba.

Rossete Suárez, A., Sebag M., Ochoa Rodriguez A. (1999). A study of Evolutionary Graph Drawing, Ecole Politechnique, France, Université Paris-Sud, Francia, CEIS, ISPJAE, Habana, Cuba, Centre of Mathemetics, Institute of Cybernetics, Mathematics and Physics, Havana, Cuba, 26 páginas.

Rusell S., Norvig P. (1996). Inteligencia Artificial, Un enfoque Moderno, Prentice Hall Hispanoamerica, México, ISBN: 968-880-682-X, 1-78.

Serna Pérez E. (2001). Diseño de Circuitos Lógicos Combinatorios utilizando Programación Genética, Tesis, Maestría en Inteligencia Artificial, Universidad Veracruzana LANIA, 1-33.

Sim S. (1996). Automatic Graph Drawing Algorithms, http:www.ics.uci.edu/~ses/ papers/grafdraw.pdf, 16 páginas.

Sommerville I. (2007). Software Engineering, Fifth Edition, Pearson education Limited, ISBN 10: 0-321-31379-8, 824 páginas.

Sugiyama, K., Misue, K. (1995). Graph Drawing by The Magnetic Spring Model, Journal of Visual Languages and Computing, No. 6, 217-231.

Sugiyama, K. (2002). Graph Drawing and Applications for Software And Knowledge Eginners, Japan Advanced Institute of Science and Technology, 1-48.

TESIS TESIS TESIS TESIS TESIS

TESIS TESIS TESIS TESIS TESIS

Sun Microsystems Java (2008). JavaTM 2 Platform Standard Edition 5.0API Specification, http://java.sun.com/j2se/1.5.0/docs/api/

Sun Microsystems JDBC (2008), MySQL Connector/J Manual, http://dev.mysql.com/ downloads/connector/j/5.1.html

Sun Microsystems MySQL (2008), MySQL 6.0 Reference Manual, http://dev.mysql. com/doc/refman/6.0/en/news-6-0-x.html

Székely, L. A. (1997). Crossing numbers and hard Erdos problems in discrete geometry. Combinatorics, Probability & Computing, 6(3), 353–358.

Szemerédi E., Ajtai M., Chvátal V., Newborn M., (1982). Crossing-free subgraphs. Annals of Discrete Mathematics, 12, 9–12.

Tunkelang, D. (1994). A Practical Approach to Drawing Undirected Graphs, School of Computer Science, Carnegie Mellon University, Pittsburg, USA, 29 páginas.

Walshaw, C. (2003). A Multinivel Algorithm for Forced-Directed Graph Drawing, Journal of Graph Algorithms and Applications, vol. 7, no. 3, 253-285.

Wiese C.K., Eicher C. (2006). Graph Drawing Tools for Bioinformatics Research: An Overview. Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems (CBMS'06) 0-7695-2517-1/06, IEEE, 6 páginas.

Wooton W., Beckenbach E. F., Fleming F. J. (1972). Modern Analytic Geometry, Houghton Mifflin Company, ISBN: 968-439-040-8, 1-47.

Zheng L., Song L., Eades P. (2005). Crossing Minimization Problems of Drawing Bipartite Graphs in Two Clusters, Australian Computer Society, Inc. Asia Pacific Symposium on Information Visualisation (APVIS 2005), Sydney, Australia. Conferences in Research and Practice in Information Technology, Vol. 45, 1-2.

TESIS TESIS TESIS TESIS TESIS