UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Modelo basado en grafos reducidos para la representación y análisis de redes en Sistemas de Información Geoespacial

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Rafael Rodríguez Puente

La Habana, 26 de septiembre de 2012

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Modelo basado en grafos reducidos para la representación y análisis de redes en Sistemas de Información Geoespacial

Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas

Autor: MSc. Rafael Rodríguez Puente

Tutor: Dr. Manuel Sabino Lazo Cortés

La Habana, 26 de septiembre de 2012

AGRADECIMIENTOS

A la Revolución Cubana por brindarme la posibilidad de estudiar y superarme.

A la Universidad de las Ciencias Informáticas, por formarme como profesional.

A mi familia, por su apoyo constante, en especial a mi mamá.

A Baby, por todo.

Al Dr. Manuel Lazo Cortés por su guía y exigencia en el rigor científico.

A la Dra. Tatiana Delgado por introducirme en el mundo de la Geomática.

Al Dr. Edel García, por introducirme en el mundo de las gramáticas de grafo.

Al Dr. Sergio Marrero por su ayuda en la aplicación de la investigación en la Ingeniería Mecánica.

Al Dr. Yusnier Valle por su ayuda con LaTeX y la revisión realizada a este documento.

A Maikel Arcia por la revisión realizada a este documento.

A los compañeros del PEFCI por el apoyo brindado.

A mis compañeros de trabajo por su apoyo y colaboración.

A todos los que de una forma u otra han sido partícipes de este trabajo.

DEDICATORIA

A mi hija Betsy.

SÍNTESIS

Los Sistemas de Información Geoespacial constituyen una importante herramienta para el apoyo a la toma de decisiones. Entre sus diversas aplicaciones se puede mencionar la búsqueda de caminos óptimos. Uno de los modelos más utilizados para realizar búsqueda de caminos óptimos es el modelo de grafo.

Se han definido varios algoritmos de reducción de grafos con el objetivo de disminuir el tiempo de respuesta a los usuarios, pero en todos los casos se pueden utilizar en ámbitos muy específicos debido a que dichos algoritmos no garantizan que no haya pérdida de información.

La presente tesis tiene como objetivo principal desarrollar un modelo para la representación y análisis de redes basado en grafos, que permita reducir el número de nodos sin perder información y que garantice escalabilidad y eficiencia en la búsqueda de caminos óptimos cuando las redes son grandes. Los principales aportes de esta investigación son: un modelo que contribuye a la realización eficiente de búsquedas de caminos óptimos en grafos grandes, el diseño de un algoritmo que reduce un grafo sin pérdida de información y que permite obtener el grafo original a partir del reducido, una modificación del algoritmo de Dijkstra para la búsqueda de caminos óptimos en grafos reducidos con el algoritmo propuesto en este trabajo, una biblioteca de clases que implementa el modelo propuesto, un plugin para el sistema Quantum GIS, que realiza búsqueda de caminos óptimos haciendo uso de la biblioteca de clases implementada y la aplicación del algoritmo de reducción a un entorno diferente a los Sistemas de Información Geoespacial, específicamente al Método de los Grafos Dicromáticos. Este método ha sido utilizado en el Diseño Racional en Ingeniería Mecánica.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE ALGORITMOS

INTRODUCCIÓN

El uso de los Sistemas de Información Geoespacial (SIG) ha aumentado considerablemente desde la década de los ochenta. Como consecuencia, estos sistemas han pasado del total desconocimiento a la práctica cotidiana en el mundo de los negocios [1, 2, 3], en las universidades [4, 5, 6] y en los organismos gubernamentales [7, 8, 9, 10, 11], usándose para resolver problemas diversos. Además, en los últimos años se ha producido un incremento de las conferencias organizadas por las principales asociaciones internacionales relacionadas con la Informática (Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y Association for Computing Machinery (ACM)) sobre la temática de los SIG [12]. En Cuba se desarrollan varios eventos relacionados con dicha temática, entre los principales se encuentra el Congreso Internacional de Geomática. Además, en el año 2009 se celebró el primer GIS Day [13], este evento se ha convertido en un foro mundial sobre la temática SIG.

Desde el punto de vista práctico un SIG es un sistema informático capaz de gestionar datos georreferenciados. Por georreferenciados se entiende que estos datos tienen asociadas coordenadas geográficas (longitud, latitud). Los SIG también deben facilitar la relación de datos de diversa procedencia (densidad de población, información financiera, etc.) con datos geográficos.

La mayor utilidad de un SIG está estrechamente relacionada con la capacidad que posee de construir modelos o representaciones del mundo real. Este tipo de sistema tiene gran relevancia ya que permite formar elementos de juicio para el proceso de toma de decisiones, contribuye al ahorro de energía y tiene una elevado impacto social.

Un SIG está formado por cuatro componentes: hardware, software, datos y recursos humanos [14]. Como parte de los datos tienen vital importancia los mapas. De forma intuitiva se puede decir que un mapa es un modelo que representa el mundo real y se almacena utilizando varios

formatos, por ejemplo: Shape [15], TAB [16], entre otros.

Por otra parte, existen proyectos que utilizan el modelo de datos relacional extendido con soporte para tipos de datos espaciales (punto, línea, polígono, etc.) para almacenar mapas. Estos proyectos representan los valores geométricos haciendo uso del estándar Well-known binary (WKB) [17], tal es el caso del proyecto OpenStreetMap [18].

Varios SIG cuentan con funcionalidades de análisis de redes, tales como: ¿cuál es el camino más corto entre los lugares x e y?, ¿cuál es el camino óptimo entre los lugares x e y según un determinado criterio?, ¿cómo llegar del lugar x al y pasando por los lugares $x_1, x_2, \ldots, x_n$?, entre otras.

Para responder este tipo de preguntas existen varios modelos de representación de redes en un SIG. Entre ellos los más relevantes son: el modelo relacional extendido y sistemas de archivos propios de determinadas herramientas informáticas. Por otra parte, cuando se habla de análisis de redes en SIG, no se pueden dejar de mencionar los grandes proveedores de servicios como Google, Microsoft, etc.; a pesar de que los mismos no publican los detalles de los mecanismos de almacenamiento y análisis que utilizan para brindar este tipo de funcionalidad.

El sistema más utilizado entre los que se distribuyen bajo licencias de software libre para realizar búsqueda de caminos óptimos en SIG es pgRouting [19]. Este software, realiza análisis de rutas en un mapa almacenado en una base de datos relacional extendida. Además, es una extensión del Sistema Gestor de Bases de Datos (SGBD) objeto-relacional PostgreSQL [20], lo que permite utilizar las potencialidades que este brinda, entre las cuales destaca el sistema de almacenamiento y el de consultas. Sin embargo, presenta las siguientes deficiencias:

- Utiliza un modelo de grafo demasiado simple y poco escalable [21].
- Es recomendado su uso cuando las redes modeladas en una cartografía son de tamaño pequeño o mediano [21].
- Presenta problemas en la búsqueda de caminos cuando el origen o el destino no coinciden con una intersección de calles [22].

Estas deficiencias se deben a que la representación interna de las distintas redes presentes en un mapa, haciendo uso del modelo relacional extendido, es ineficiente [23].

Entre los sistemas que no utilizan un SGBD relacional extendido para realizar análisis de redes, está el líder ArcGIS [24]. Esta herramienta brinda la posibilidad de definir una jerarquía de calles, de forma tal que la búsqueda de caminos óptimos se realiza priorizando las calles de mayor nivel en la jerarquía cuando la red de viales es grande. La jerarquía se utiliza para disminuir el tiempo de respuesta al usuario, lo cual introduce un error en el camino obtenido según la descripción en la ayuda del propio sistema; aunque brinda la posibilidad de realizar el análisis obviando la jerarquía mencionada, lo que implica un menor rendimiento.

La interpretación del concepto "grande" que se hace en este trabajo es relativa y no se basa en un número particular de elementos presentes en una red; este concepto puede depender de factores como características del hardware disponible, capacidad para la interpretación de características presentes en una red, etc.

En el ámbito internacional se han realizado varios trabajos para disminuir el tiempo de respuesta en la búsqueda de caminos óptimos en redes grandes. La tendencia que se aprecia es el uso de algoritmos heurísticos. Algunos de los algoritmos diseñados en este sentido son: Reach-Based Pruning [25], Landmark-A* [26], Edge flags [27, 28], Geometric containers [29], Precomputed Cluster Distances (PCD) [30]. Otros hacen uso de la jerarquía presente en las redes de viales, definiendo prioridades según varios tipos de calles [31, 32, 33, 34, 35, 36].

Los algoritmos heurísticos son importantes en la solución de problemas de alto costo computacional. Sin embargo, introducen un error y por tanto no garantizan la obtención del camino óptimo en todos los casos.

En el ámbito nacional se han desarrollado varios SIG. Estos desarrollos han estado centrados en dos áreas de aplicación principales: aplicaciones en la salud, entre las que se pueden mencionar aplicaciones en la epidemiología [37, 38], en salud y medio ambiente [39], en la gestión de estadísticas en la salud [40], entre otras; y aplicaciones en la protección del medio ambiente, por ejemplo: conservación de áreas protegidas [41], estudios de impacto ambiental [42], etcétera. También se desarrolló la plataforma LiberGIS [43] actualmente conocida como GeneSIG [44], considerada como un SIG de propósito general. Con esta plataforma se han implementado varios sistemas, un ejemplo de ello es el SIG para la gestión de objetivos petroleros [45]. Sin embargo, esta plataforma aún no cuenta con funcionalidades para el análisis de redes.

Por otra parte, el grupo empresarial Geocuba desarrolló un sistema para realizar búsquedas de caminos óptimos [46]. Dicho sistema utiliza algoritmos para el análisis que están basados en el algoritmo de Dijkstra, el cual no es conveniente utilizar en redes grandes.

Otros resultados en los últimos años están vinculados al desarrollo de la Infraestructura de Datos Espaciales de la República de Cuba (IDERC) [47], la cual brinda varios servicios pero no cuenta con servicios de análisis de redes.

A pesar de los diversos sistemas existentes, no se evidencia en las investigaciones de los autores cubanos avances significativos en cuanto al análisis de redes grandes en SIG. Esta situación es un reto para los profesionales de la rama debido al impacto que tiene este tipo de análisis en la toma de decisiones, en particular, en el ahorro y control de recursos como el combustible, así como en los servicios informativos relacionados con el transporte para la sociedad de forma general.

A partir de la situación descrita se define el siguiente **problema científico**:

Los modelos de representación y análisis de redes en SIG, existentes en la actualidad, no garantizan escalabilidad y eficiencia en la búsqueda de caminos óptimos cuando las redes son grandes.

El problema definido se enmarca en el siguiente **objeto de investigación**: Modelos para la representación y análisis de redes en SIG.

El **objetivo general** de la investigación es desarrollar un modelo basado en grafos reducidos para la representación y análisis de redes en SIG, que permita reducir el número de vértices sin perder información y garantice escalabilidad y eficiencia en la búsqueda de caminos óptimos cuando las redes son grandes.

El **campo de acción** es: Búsqueda de caminos óptimos en SIG.

A partir de un análisis preliminar se enuncia la siguiente **hipótesis**: Si se desarrolla un modelo basado en grafos reducidos para la representación y análisis de redes, que permita reducir el número de vértices sin perder información; entonces se garantizará escalabilidad y eficiencia en la búsqueda de caminos óptimos en SIG cuando las redes son grandes.

El objetivo general se desglosa en los siguientes **objetivos específicos**:

- Diseñar un algoritmo de reducción de grafos sin pérdida de información.

- Diseñar un algoritmo para realizar búsquedas de caminos óptimos en grafos reducidos.
- Validar el modelo teóricamente realizando la demostración de corrección de los algoritmos diseñados.
- Desarrollar una herramienta computacional basada en el modelo propuesto.
- Explorar otros posibles marcos de aplicación del modelo propuesto.

Para darle cumplimiento a los objetivos trazados se utilizaron los siguientes **métodos de investigación**:

- Métodos teóricos:
    - Analítico-sintético: para descomponer el problema de investigación en elementos por separado: modelos de representación y análisis de redes, búsqueda de caminos óptimos en redes grandes, eficiencia y escalabilidad, de esta forma es más sencillo profundizar en el estudio de cada elemento, para luego sintetizarlos en la solución de la propuesta.
    - Hipotético-deductivo: para elaborar la hipótesis de investigación y proponer líneas de trabajo a partir de resultados parciales.
    - Histórico-lógico: para llevar a cabo el estudio crítico de los modelos de representación de redes y realizar análisis de las mismas en SIG, así como de los algoritmos de búsqueda de caminos óptimos y de reducción de grafos más relevantes existentes en la literatura.
    - Modelado: para el diseño de la herramienta computacional y de los algoritmos propuestos.
    - Inducción-deducción: para realizar las demostraciones de corrección de los algoritmos propuestos.
- Métodos empíricos:
    - Experimental: para comprobar los resultados derivados de las demostraciones de corrección de los algoritmos propuestos.

La **novedad científica** de la presente investigación radica en un nuevo modelo que garantiza la realización eficiente y escalable de búsquedas de caminos óptimos en redes grandes a través

de un algoritmo de reducción de grafos sin pérdida de información. Las demostraciones de corrección de los algoritmos diseñados así como la realización computacional de los mismos, muestran que se garantiza la obtención de caminos de igual costo al obtenido por el algoritmo de Dijkstra en grafos sin reducir. También se muestra la obtención de tiempos de ejecución menores que los obtenidos por el algoritmo A*, el cual es utilizado como referente por la comunidad científica en la disminución de los tiempos de ejecución de búsquedas de caminos óptimos.

Aportes teóricos:

- Desarrollo de la base conceptual y de un algoritmo de reducción de grafos sin pérdida de información; lo que puede ser aplicado en diversas ramas de la ciencia.
- Un nuevo algoritmo, basado en el algoritmo de Dijkstra, para realizar búsquedas de caminos óptimos en grafos reducidos con el algoritmo de reducción propuesto. Este algoritmo permite obtener un camino óptimo en todos los casos en un tiempo similar al obtenido por algoritmos heurísticos, logrando una mejora significativa en la eficiencia sin perder exactitud en el resultado.

Aportes prácticos:

- Una herramienta computacional que brinda soporte a la reducción de grafos, así como a la búsqueda de caminos óptimos en grafos reducidos con el algoritmo de reducción propuesto.
- Un plugin que brinda soporte al sistema Quantum GIS (QGIS) para la búsqueda de caminos óptimos haciendo uso del modelo propuesto.
- Disminución del tiempo de respuesta al usuario ante una petición de búsqueda de camino óptimo en redes grandes.
- Indicaciones metodológicas para la aplicación del modelo que se propone.

La **estructura de la tesis** se expone haciendo uso de una perspectiva de anillo como se puede apreciar en la Figura 1. La misma está concebida a partir del objeto de investigación: Modelos para la representación y análisis de redes en SIG, garantizando de esta forma la consistencia estructural de la tesis que se presenta.

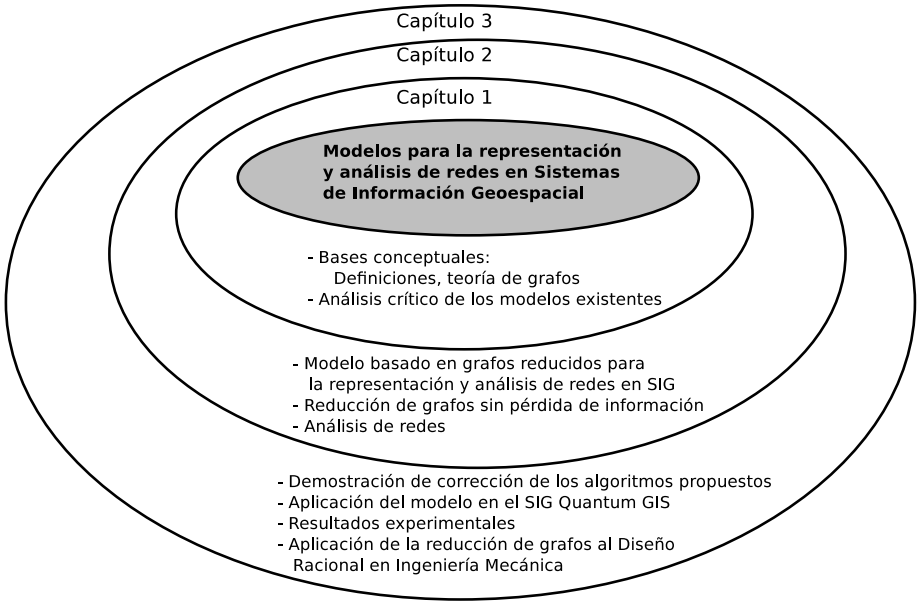


Figura 1: *Estructura de la tesis.*

Capítulo 1

Representación y análisis de redes en Sistemas de Información Geoespacial: Bases conceptuales

1. Representación y análisis de redes en Sistemas de Información Geoespacial: Bases conceptuales

En este capítulo se expone el marco conceptual y se realiza un análisis del estado del arte de los modelos que se utilizan en la actualidad para la representación y análisis de redes en SIG. Además, se analizan los mecanismos que permiten realizar búsquedas de caminos óptimos en este tipo de sistema, así como los algoritmos de reducción de grafos.

1.1. Marco teórico

En la presente sección se muestran varias definiciones relacionadas con los SIG y con la teoría de grafos.

1.1.1. Sistemas de Información Geoespacial

El uso de los SIG ha aumentado desde las décadas de los ochenta y los noventa y como consecuencia, estos sistemas han pasado del total desconocimiento a la práctica cotidiana usándose para resolver problemas diversos. Varios autores han propuesto definiciones de SIG [14, 48, 49, 50], en este trabajo se asume la siguiente:

- Un SIG es un sistema computacional para la entrada, manejo (almacenamiento y recuperación de información), manipulación, análisis y representación de datos geográficos [14].

Los conceptos clásicos de SIG han evolucionado para destacar el papel de la diseminación de los datos como una función ineludible de los SIG en ambientes distribuidos y globales de acceso de datos y en el entorno de la World Wide Web (WWW) [51].

De forma general existen dos formatos de datos que son los más utilizados en los SIG: el formato vectorial y el raster. En el primer caso se utilizan puntos, líneas (definidas por una serie de puntos) y polígonos (delimitados por líneas) para representar los objetos geográficos. En el segundo caso los datos consisten en filas de celdas, a cada celda se pueden asociar datos de diversos tipos (medida, nombre, etc.) [52].

Para realizar análisis de redes se utilizan los SIG vectoriales, o sea, los que utilizan el formato vectorial para almacenar los mapas; por lo que en el marco de este trabajo solo se hará referencia a este tipo de sistema.

La mayoría de los SIG en la actualidad dan respuesta a peticiones relacionadas con el análisis de rutas, debido a la demanda que tienen por los usuarios de este tipo de sistema, tales como:

- ¿Cuál es el camino más corto entre los lugares x e y?
- ¿Cuál es el camino más corto entre los lugares x al y pasando por los lugares $x_1, x_2, .., x_n$?
- ¿Cuál es el camino óptimo entre los lugares x e y de acuerdo a un determinado criterio de optimalidad?

Para satisfacer este tipo de petición, reviste vital importancia el modelo de datos que se utilice en la representación de la red de viales de un mapa. Antes de hacer referencia a dichos modelos, se abordarán algunos conceptos de la teoría de conjuntos y de grafos que contribuirán a una mejor fundamentación teórica de los modelos descritos y de la propuesta realizada en esta investigación.

1.1.2. Relaciones

En esta sección se presentan definiciones y notaciones que se utilizarán en el modelo propuesto, las mismas son tomadas de [53].

DEFINICIÓN 1.1.2.1 *Se denomina partición del conjunto A, a una colección P de subconjuntos de A que satisface las siguientes condiciones:*

- $\phi \notin P$.
- *Para todo par de conjuntos A_i y A_j de P, con $i \neq j$, se cumple que $A_i \cap A_j = \phi$.*

- *La unión de todos los conjuntos de P es igual a A, es decir, si $A_1, A_2, ..., A_n$ son los subconjuntos de P entonces $A = A_1 \cup A_2, \cup ... \cup A_n$.*

DEFINICIÓN 1.1.2.2 *Sean los conjuntos A y B no necesariamente distintos, se dice que R es una relación binaria de A en B si R es un subconjunto de pares ordenados de $A \times B$, es decir, si $R \subseteq A \times B$.*

Si $A = B$, entonces se dice que R es una relación binaria en A.

DEFINICIÓN 1.1.2.3 *Una relación R en A es reflexiva si y solo si $\forall x, x \in A, (x, x) \in R$.*

DEFINICIÓN 1.1.2.4 *Una relación R en A es simétrica si y solo si $\forall x, x \in A, \forall y, y \in A$ si $(x, y) \in R$ entonces $(y, x) \in R$.*

DEFINICIÓN 1.1.2.5 *Una relación R en A es transitiva si y solo si $\forall x, x \in A, \forall y, y \in A, \forall z, z \in A$ si $(x, y) \in R$ y $(y, z) \in R$, entonces $(x, z) \in R$.*

DEFINICIÓN 1.1.2.6 *Una relación binaria R en un conjunto A es una relación de equivalencia si y solo si R es reflexiva, simétrica y transitiva en A.*

DEFINICIÓN 1.1.2.7 *Sea R una relación de equivalencia entre los elementos de un conjunto A y sea $a \in A$, entonces se denomina clase de equivalencia por R de a y se denota mediante $[a]$ al conjunto de las $x, x \in A$, que mantienen con a la relación R, es decir, $[a] = \{x|(a, x) \in R\}$.*

DEFINICIÓN 1.1.2.8 *La familia de las clases de equivalencia por R de A se denomina el cociente de A por R y se denota $A/R = \{[a]|a \in A\}$.*

PROPOSICIÓN 1.1.2.1 *Si R es una relación de equivalencia en un conjunto A, entonces a partir de R puede definirse una partición de A, precisamente A/R, y viceversa, si se tiene una partición B de A entonces puede definirse una relación de equivalencia R en A tal que $B = A/R$.*

1.1.3. Grafos

En esta sección se presentan definiciones y notaciones utilizadas en los algoritmos que se proponen.

DEFINICIÓN 1.1.3.1 *Un grafo, o grafo no dirigido, $G = (V, E)$ se define como un conjunto V finito y no vacío de vértices y un multiconjunto E de aristas, donde cada arista $(v_i, v_j), v_i, v_j \in V$ es un par no ordenado de vértices. Opcionalmente una arista puede tener un valor que la identifique y una lista de atributos. Cuando los elementos de E tienen multiplicidad uno, el grafo se denomina grafo simple.*

La definición de grafo dirigido es similar a la anterior, con la única diferencia que las aristas son pares ordenados.
En los anexos A y B se muestra el Tipo de Dato Abstracto (TDA) MultiDigrafo y GrafoReducido, utilizados en la presente investigación.
Para referirse al conjunto de vértices y aristas del grafo $G = (V, E)$ se utilizará la notación $G.V$ y $G.E$ respectivamente, o simplemente V y E cuando no exista ambigüedad respecto al grafo.
Las definiciones que siguen son tomadas de [54].

DEFINICIÓN 1.1.3.2 *Sean $G = (V, E)$ y $G' = (V', E')$ dos grafos, si $V' \subseteq V, E' \subseteq E$ se dice que G' es un subgrafo de G y G es un supergrafo de G'.*

DEFINICIÓN 1.1.3.3 *La matriz de adyacencia $A = (a_{ij})_{n \times n}$ del grafo G, donde $n = |V|$, se define de la siguiente forma:*

$$a_{ij} = \begin{cases} 1 & si (v_i, v_j) \in E \\ 0 & \textit{en otro caso} \end{cases}$$

DEFINICIÓN 1.1.3.4 *Un vértice v_j es adyacente (vecino) a otro vértice v_i si $\exists (v_i, v_j) \in E$. Además, se dice que la arista (v_i, v_j) incide en el vértice v_j.*

DEFINICIÓN 1.1.3.5 *Se denomina grado de un vértice v a la cantidad de aristas que inciden en v y se representa como $g(v)$.*

DEFINICIÓN 1.1.3.6 *En un grafo dirigido, se denomina grado de entrada de un vértice v a la cantidad de aristas que tienen como vértice final a v y se representa como $g^-(v)$. De forma similar, se denomina grado de salida a la cantidad de aristas que tienen como vértice inicial a v y se representa como $g^+(v)$.*

DEFINICIÓN 1.1.3.7 *Se denomina grado de entrada máximo de un grafo dirigido $G = (V, E)$ al mayor grado de entrada asociado a un vértice $v \in V$ y se representa como $\Delta^-(G) = max[g^-(v)|v \in V]$. De forma similar, se denomina grado de salida máximo de un grafo $G = (V, E)$ al mayor grado de salida asociado a un vértice $v \in V$ y se representa como $\Delta^+(G) = max[g^+(v)|v \in V]$.*

DEFINICIÓN 1.1.3.8 *Se denomina grado de un grafo $G = (V, E)$ al mayor grado asociado a un vértice $v \in V$ y se representa como $\Delta(G) = max[g(v)|v \in V]$.*

DEFINICIÓN 1.1.3.9 *Se define un grafo ponderado como una estructura $G = (V, E, f_c)$, donde:*

- *V representa el conjunto de vértices del grafo.*
- *E representa un multiconjunto de aristas del grafo.*
- *La función $f_c : E \to \mathbb{R}^+$ le hace corresponder a cada arista un valor real positivo denominado costo, cuya interpretación es el costo de ir desde el vértice v_i al vértice v_j.*

DEFINICIÓN 1.1.3.10 *La matriz de costo $C = (c_{ij})_{n \times n}$ de un grafo ponderado $G = (V, E, f_c)$, donde $n = |V|$, se define como:*

$$c_{ij} = \begin{cases} k & \text{si } \exists (v_i, v_j) \in E \text{ y } k \text{ es el costo de ir desde } v_i \text{ hasta } v_j, k > 0, k = f_c(v_i, v_j) \\ 0 & \text{en otro caso} \end{cases}$$

DEFINICIÓN 1.1.3.11 *Se denomina camino desde el vértice v_i al vértice v_j en un grafo $G = (V, E)$ a la secuencia de vértices $CA = v_{k_1}, v_{k_2}, \ldots, v_{k_t}$ si $\exists (v_{k_1}, v_{k_2}), (v_{k_2}, v_{k_3}) \ldots, (v_{k_{t-1}}, v_{k_t}) \in E, v_i = v_{k_1}, v_j = v_{k_t}$. Además, una arista solo puede aparecer una vez.*

DEFINICIÓN 1.1.3.12 *Se denomina longitud de un camino* $CA = v_{k_1}, v_{k_2}, \ldots, v_{k_t}, t > 1$*, a la suma de los costos de todas las aristas presentes en el mismo y se representa como* $|CA| = \sum_{h=1}^{t-1} c_{k_h k_{h+1}}$*. En el caso de grafos no ponderados, la longitud del camino se puede calcular como la cantidad de aristas presentes en el mismo, o sea* $|CA| = t - 1$*.*

DEFINICIÓN 1.1.3.13 *Un grafo* G *es conexo si entre cada par de vértices existe un camino que los une.*

DEFINICIÓN 1.1.3.14 *El problema de la búsqueda del camino óptimo entre dos vértices* v_i *y* v_j *de un grafo consiste en encontrar un camino* $CA = v_{k_1}, v_{k_2}, \ldots, v_{k_t}, t > 1, v_i = v_{k_1}, v_j = v_{k_t}$ *tal que la suma* $\sum_{h=1}^{t-1} c_{k_h k_{h+1}}$ *sea óptima.*

En varios problemas que son resueltos a través del uso de la teoría de grafos es común encontrar la necesidad de transformar un grafo. Por ejemplo: modificando los nodos o las aristas, eliminando un subgrafo o adicionando un grafo a otro. También se puede utilizar un formalismo matemático denominado gramática de grafo, el cual se define a continuación.

DEFINICIÓN 1.1.3.15 *Una gramática de grafo NCE es un sistema* $G = (\Sigma, \delta, P, S)$ *donde:*

- Σ *es un conjunto finito y no vacío denominado alfabeto.*
- δ *es un subconjunto de* Σ *denominado alfabeto de los símbolos terminales.*
- P *es un conjunto finito de producciones o reglas de reescritura de la forma* (α, β, ψ)*, donde* α *es un grafo conexo,* β *es un grafo y* $\psi : \alpha.V \times \beta.V \times \Sigma \rightarrow \{0, 1\}$*,* ψ *se denomina función de empotrado y está determinada por el mecanismo de empotrado que se utilice.*
- S *es el símbolo distinguido o axioma.*

Las gramáticas de grafo se pueden clasificar según varios criterios. Atendiendo a la forma en que se transforman los grafos y siguiendo el enfoque clásico de Janssens y Rozenberg se pueden clasificar en gramáticas NCE [55], NLC [56, 57], HR [58], NR [58], entre otras. La especificación de cómo se debe transformar un grafo, se denomina mecanismo de empotrado. Debido al mecanismo de empotrado que utilizan los tipos de gramáticas antes mencionadas, se seleccionan para utilizar en este trabajo las gramáticas NCE. Para profundizar sobre las diferencias entre los distintos mecanismos de empotrado, el lector puede remitirse a [59].

Las gramáticas de grafo pueden ser aplicadas en la solución de varios problemas, en particular se pueden aplicar en los SIG [60]. Uno de los componentes fundamentales de las mismas es el conjunto de producciones. La aplicación de una producción o regla de reescritura de grafo consiste en eliminar el subgrafo α del grafo al que se le va a aplicar la regla, adicionar el grafo β a dicho grafo y conectarlo según se especifique en ψ.

La formalización de gramática NCE utiliza como base la definición de grafos no dirigidos y etiquetados. Sin embargo, cuando el grafo es dirigido se deben tener en cuenta consideraciones adicionales. Para dar soporte a grafos dirigidos se puede seguir el mismo enfoque que en [56], sustituyendo la función ψ por dos funciones: ψ_{in} y ψ_{out} para las aristas que entran o salen de un vértice respectivamente.

1.1.4. Modelos de representación y análisis de redes en Sistemas de Información Geoespacial

En la literatura se puede encontrar abundante información sobre la forma en que se representan los mapas para que puedan ser utilizados por los SIG. Se han creado varios estándares por el Open Geospatial Consortium (OGC) [61], una institución no lucrativa cuya misión es definir los estándares relacionados con el mundo geoespacial. Entre los estándares más utilizados para la representación de mapas se pueden mencionar:

- Well-known text (WKT) , WKB [62].
- Keyhole Markup Language (KML) [63].
- Geographic Markup Language (GML) [64].
- Mapinfo Data Interchange Format (TAB) [65].
- Simple Feature Specification for SQL (SFS) [66].

Estos estándares pueden ser utilizados por SIG de tres formas principales: en SGBD relacionales extendidos, en SGBD Orientados a Objetos y en implementaciones de sistemas de ficheros propias de los SIG propietarios (ArcGIS, MapInfo, etc.).

1.2. Análisis crítico de las soluciones existentes

En este epígrafe se realiza un análisis crítico de las principales soluciones existentes en la actualidad que facilitan la búsqueda de caminos óptimos en SIG. Para ello, el análisis se divide en los modelos utilizados para la representación y análisis de redes, los algoritmos de búsqueda de caminos óptimos y por último los algoritmos de reducción de grafos como herramienta para la reducción del tiempo de respuesta de los algoritmos ejecutados sobre un grafo.

1.2.1. Modelos utilizados para la representación y análisis de redes en Sistemas de Información Geoespacial

Existen varios modelos que pueden ser utilizados para representar redes en un SIG, entre ellos, los más relevantes son: el modelo relacional extendido, el modelo orientado a objetos y sistemas de archivos propios de determinadas herramientas informáticas. Por otra parte, cuando se habla de cálculo de rutas en SIG, no se puede dejar de mencionar los grandes proveedores de servicios como Google, Microsoft, etc., a pesar de que los mismos no publican los detalles de los mecanismos de almacenamiento que utilizan para brindar este tipo de funcionalidad.

1.2.1.1. Modelo relacional extendido

El modelo relacional fue propuesto por E.F. Codd en el año 1970 [67]. Este modelo está basado en la lógica de predicado de primer orden y en la teoría de conjuntos. En la actualidad las bases de datos relacionales son el tipo de bases de datos más difundido debido a la formalización del modelo relacional.

Existen varias extensiones del modelo relacional [68, 69, 70, 71, 72]. A pesar de las diferencias sintácticas de cada una de las extensiones realizadas a este modelo, todas giran alrededor de un enriquecimiento semántico del mismo [73]. Otra extensión del modelo relacional que ha surgido para su aplicación en los SIG es el modelo de datos espacial [74, 75, 76]. El mismo adiciona los tipos de datos primitivos línea, punto y polígono y funciones para el manejo de los mismos, lo cual facilita el almacenamiento de cartografías así como el análisis de las mismas.

El sistema pgRouting [19] se distribuye bajo la Licencia Pública General de GNU versión 2

(GPLv2) y es uno de los sistemas más utilizados, entre los que se distribuyen bajo licencia de software libre, que hacen la búsqueda de caminos óptimos a partir de un mapa almacenado en una base de datos haciendo uso del modelo relacional extendido. Este software es una extensión del SGBD objeto relacional PostgreSQL, lo que permite utilizar todas las potencialidades que brinda dicho SGBD; entre las cuales se destaca el sistema de almacenamiento y el de consultas. A pesar de lo anterior, pgRouting presenta las siguientes deficiencias:

- Utiliza un modelo de grafo demasiado simple y poco escalable; además no puede usarse en entornos multi-hilo [21].
- Presenta problemas en la búsqueda de caminos cuando el origen o el destino no coinciden con una intersección de calles [22]. Lo anterior se debe a que cuando un punto no coincide con una intersección de calles, el sistema realiza la búsqueda desde (o hasta) la intersección más cercana a dicho punto, lo cual puede introducir un error en el resultado.
- Cuando se realiza una búsqueda de camino óptimo carga todos los datos en memoria interna [21], lo que trae como consecuencia que el análisis dependa en gran medida de la capacidad de memoria con que se cuente, aspecto de vital importancia en los SIG debido al elevado volumen de información que manejan.
- Representa y analiza las redes de forma plana, lo que trae como consecuencia que cuando dos calles están a diferentes niveles, el sistema no distingue si se intersectan o no, lo que puede traer problemas al realizar análisis de redes. El proyecto OpenStreetMap [77] utiliza el sistema pgRouting para el cálculo de caminos óptimos y para eliminar esta problemática introduce un campo (nivel) para especificar el nivel al que se encuentra una calle.

Teniendo en cuenta las características de pgRounting, el modelo que utiliza y su rendimiento, se recomienda su uso cuando las redes modeladas en la cartografía son de tamaño pequeño o mediano [21].

Las problemáticas planteadas existen debido a que el modelo utilizado no permite modelar de forma natural una red (viales, distribución de agua, electricidad, etc.) presente en una cartografía; además las operaciones que se realizan sobre las redes como son: grado de la red (o de un vértice), camino entre dos vértices de la red, etc. no están concebidas dentro del modelo

relacional, aunque se pueden obtener utilizando las operaciones sobre las que está definido el modelo.

Cuando el modelado de un dominio específico de la realidad (modelo lógico) responde a una estructura de grafo, utilizar una base de datos de grafo [78] es la alternativa lógica para almacenar los datos. En [79] se muestran los resultados experimentales de la realización de recorridos en datos que, por su naturaleza, son representados haciendo uso de grafos dirigidos acíclicos. Los recorridos fueron realizados sobre el SGBD MySQL [80] y el motor de persistencia de grafos Neo4j [81]. Los resultados muestran de forma general que los recorridos realizados en Neo4j son más eficientes que en la base de datos relacional.

1.2.1.2. Modelo orientado a objetos

En [82] se hace una revisión de los modelos de datos orientados a objetos aplicados a SIG. El autor hace referencia a la superioridad del modelo de objetos sobre el relacional, debido a la posibilidad de poder manipular objetos complejos y su comportamiento, meta conocimiento de larga duración y transacciones. Se puede profundizar en el modelo antes mencionado en varios textos específicos de la temática de bases de datos [83, 84, 85, 86].

Haciendo uso del modelo de objetos, una red de transporte se puede representar con varios tipos de objetos como calles, autopistas, intersecciones, etc., donde cada uno especifica las distintas operaciones que se pueden realizar sobre todos los objetos de dicho tipo. En la bibliografía consultada sobre este modelo, se puede apreciar que se define una red de transporte haciendo uso de una jerarquía entre puntos, multilíneas, polígonos y listas de multilíneas. En el contexto del modelo orientado a objetos, se define qué es relación de asociación, de agregación, de herencia, así como polimorfismo y encapsulamiento. Haciendo uso de estos conceptos se puede modelar una red de transporte, pero los mismos no se pueden utilizar de forma natural para realizar análisis de redes sobre el modelo obtenido, lo que trae consigo un aumento en la complejidad de los algoritmos para dar respuesta a las peticiones de los usuarios.

1.2.1.3. Sistemas que utilizan otros modelos

Existen varios sistemas que tienen una implementación propia de almacenamiento de la cartografía, destacándose entre ellos el líder ArcGIS con el formato Shape. A continuación se

muestran las principales ventajas y desventajas de estos sistemas.

ArcGIS

ArcGIS es un SIG para visualización, manejo, creación y análisis de datos geográficos. Este sistema es desarrollado y mantenido por el Enviromental Systems Research Institute (ESRI), empresa líder en el ámbito de los SIG [24]. El mismo cuenta con la extensión ArcGIS Network Analyst para realizar análisis de redes. Este sistema, define una jerarquía de calles como estrategia para disminuir el tiempo de respuesta cuando la red de transporte es grande; esto lo hace estableciendo de tres a cinco niveles en la jerarquía. Por ejemplo, una jerarquía de tres niveles puede quedar de la siguiente forma [87, 88]:

- Vías primarias (autopistas).
- Vías secundarias (calles principales).
- Vías locales (calles locales).

Esta jerarquía permite disminuir el tiempo de respuesta al usuario pero, generalmente, el camino obtenido tiene un mayor costo que si se realizara el cálculo de la ruta sin considerar los niveles, ya que bajo este esquema se prioriza viajar por las vías de nivel superior en la jerarquía [87]. Es importante aclarar que hay conductores que prefieren viajar por las vías primarias sin importar el costo adicional del viaje, pero no es así en todos los casos.

Solución gvSIG

El proyecto gvSIG [89] tiene como objetivo el desarrollo de un SIG haciendo uso de software libre. La herramienta más utilizada de este proyecto es gvSIG Desktop [90], comúnmente conocida como gvSIG por ser la primera en desarrollarse. La misma cuenta con un módulo de análisis de redes [91] que provee diversas funcionalidades, facilitando la corrección de la topología, la creación de un grafo que representa una red de viales, así como la búsqueda de caminos óptimos.

Este módulo de análisis de redes, hace uso de los algoritmos clásicos de búsqueda de caminos óptimos, tales como el algoritmo de Dijkstra y el A*, por lo que se descarta su uso en la presente investigación.

Directed Graph Library

El sistema Geographic Resources Analysis Support System (GRASS) [92], es un SIG que se utiliza para manipulación de datos, procesamiento de imágenes, producción de gráficos y visualización de muchos tipos de datos. Es un software de código abierto liberado bajo la Licencia Pública General de GNU (GPL) [93] y es un proyecto oficial de la Open Source Geospatial Foundation [94].

GRASS utiliza la biblioteca Directed Graph Library [95] liberada bajo la licencia GPL para realizar análisis de redes. La idea original del proyecto se basa en el desarrollo de una biblioteca que soporte el análisis sobre grafos de mediano tamaño en Memoria de Acceso Aleatorio (RAM) haciendo uso de una estructura estática que no necesite ser modificada dinámicamente [96], por lo que no es conveniente utilizar la misma cuando los grafos son grandes.

IDELabRoute

IDELabRoute es una biblioteca genérica para realizar análisis de redes con gestión dinámica de memoria [21]. Surge porque en la práctica, varios problemas necesitan tratar con redes de grandes dimensiones. IDELabRoute es desarrollado por el Laboratorio de Infraestructuras de Datos Espaciales (IDELab) [97]. Esta biblioteca permite realizar análisis de redes utilizando diversas fuentes de datos:

- AllInMemoryManager: Se carga el grafo completo en memoria, lo que no es conveniente para grafos grandes.
- AllInMemoryExternalSourceMemoryManager: Utiliza memoria externa, pero al igual que el anterior también carga el grafo completo en memoria.
- BasicExternalSourceMemoryManager: Carga en memoria solo los elementos solicitados, pero esto lo hace a expensas de un aumento de las peticiones con la consecuente demora en el tiempo de respuesta.

De forma general, los autores realizan una propuesta para hacer un uso racional de la memoria RAM (mayor escalabilidad) a cambio de un decremento de la eficiencia en la búsqueda de caminos óptimos. Debido a esta razón, no se considera conveniente usar este resultado en el marco de esta investigación.

1.2.1.4. Servicios en línea

En la web se pueden encontrar varias implementaciones de SIG con la funcionalidad de búsqueda de camino óptimo:

- Google maps [98], desarrollado por la compañía Google de EEUU.
- Bing maps de Microsoft (Multimap) [99], desarrollado por la compañía Microsoft de EEUU.
- Yahoo maps [100], desarrollado por la compañía Yahoo de EEUU.
- Map24 [101], desarrollado por la compañía America Online, Inc. de EEUU.
- Maporama [102], desarrollado por la compañía Maporama Solutions de Francia.

Estos SIG, a menudo referenciados como los grandes proveedores de mapas, brindan sus funcionalidades como servicios; dichas funcionalidades pueden ser utilizadas a través de una Interfaz de Programación de Aplicaciones (API) que publica cada proveedor, la cual es utilizada en el desarrollo de otros SIG, tales como:

- Callejero PáginasAmarillas.es [103].
- El Callejero Lanetro [104].
- Callejero ELMUNDO.es [105].
- Callejero Terra [106].
- Callejero ElPAÍS.com [107].

Los SIG mencionados anteriormente brindan una amplia cantidad de información sobre las rutas, algunos de ellos en un país determinado y otros en varios países del mundo. Todos se caracterizan por: utilizar los servicios que brindan los grandes proveedores de mapas (Google, Yahoo, Microsoft, etc.), hacer búsquedas del camino más corto (también el más rápido), especificar si se está viajando en carro o a pie, entre otras posibilidades.

Estos servicios tienen características que inducirían a los usuarios finales a utilizarlos, tales como: son servicios gratis, son accesibles a través de la web y siempre están disponibles. Según [108], la mayoría de las necesidades de los usuarios finales son cubiertas por los grandes proveedores de servicios mencionados anteriormente. El rol fundamental de los desarrolladores de SIG radica en hacer uso de esos servicios para crear productos de valor agregado. También

se hace referencia que para tener buenos resultados en el uso de estos servicios, es necesario contar con un determinado ancho de banda, con el que algunos países desarrollados, por ejemplo España, no cuenta aún.

Otro aspecto importante a tener en cuenta al hacer uso de estos servicios, es que no se conoce la forma en que están implementados internamente. La mayoría, pertenecen a grandes compañías sujetas a leyes que van contra la soberanía de Cuba.

También se presenta la siguiente problemática: no se ha publicado, en la bibliografía consultada, la forma en que se maneja tal cantidad de datos, ni los algoritmos utilizados para el análisis de las rutas. Además, estos SIG están desarrollados sobre tecnología propietaria.

La teoría de grafos, provee una representación adecuada de una red, así como conceptos y algoritmos que permiten estudiar las propiedades de las mismas [109]. Por otra parte, Sedgewick afirma que una red puede ser representada con la misma estructura de datos que es utilizada para la representación de grafos [110], por lo que son aplicables los algoritmos definidos para estos últimos. También se define una red como un conjunto de nodos y un conjunto de relaciones entre ellos [111], definición que está acorde con el concepto de grafo.

Luego del estudio de los modelos más utilizados para realizar análisis de redes en SIG y teniendo en cuenta las definiciones de red y de grafo, se puede afirmar que es conveniente estudiar una red haciendo uso del concepto grafo.

1.2.2. Algoritmos de búsqueda de caminos óptimos

La búsqueda de caminos óptimos ha sido ampliamente estudiada, se pueden encontrar aplicaciones en varias ramas de la ciencia, en particular en los SIG, debido a que este tipo de sistema se considera como una de las mejores herramientas para almacenar y utilizar modelos de redes [112]. A continuación se realiza el análisis de varios algoritmos de búsqueda de caminos óptimos.

Uno de los algoritmos clásicos para el cálculo del camino óptimo desde un origen a un destino es el algoritmo Dijkstra (ver Anexo C). Enunciado por primera vez por Edsger W. Dijkstra en el año 1959 [113], es uno de los algoritmos más utilizados y discutidos en la literatura de grafos, la complejidad temporal es $O(n^2)$ siendo n la cantidad de vértices del grafo. Sin embargo, este

algoritmo no es eficiente para realizar búsquedas de camino óptimo en grafos grandes [114].

En [114] se propone un nuevo algoritmo, basado en el algoritmo de Dijkstra que realiza una optimización, basada en la disminución de los valores 0 y ∞ que aparecen en la matriz de adyacencia del grafo sobre el que se realiza el análisis, para ello construye una nueva matriz con n filas y m columnas, donde n es la cantidad de vértices del grafo y m es la mayor cantidad de vértices adyacentes. Según se muestra en los experimentos realizados, esta nueva variante ejecutada en un grafo de 12000 nodos, logra disminuir tres veces el tiempo de ejecución con respecto al algoritmo de Dijkstra si la memoria es suficiente.

Otro enfoque del uso del algoritmo de Dijkstra se expone en [115], en el cual se parte de que la complejidad temporal de dicho algoritmo aumenta drásticamente con el aumento de la cantidad de vértices del grafo. La nueva propuesta hace uso de una implementación de cola con prioridad utilizando la estructura de datos heap [116]. En este caso se reduce la complejidad temporal del algoritmo de $O(n^2)$ hasta $O(n \lg n)$.

En las dos variantes anteriores se muestra un resultado importante, pero aún presentan problemas relacionados con la escalabilidad. La complejidad de estos algoritmos, al igual que en el algoritmo de Dijkstra, siempre dependerá del tamaño del grafo, que en caso de ser muy grande implicará un tiempo de respuesta elevado.

Otro algoritmo clásico que se puede mencionar es el algoritmo de Floyd [117] (también conocido como algoritmo Floyd-Warsall). El mismo ha sido ampliamente estudiado y utilizado para realizar el cálculo de caminos óptimos. El algoritmo de Floyd no se tendrá en cuenta en la presente investigación ya que la complejidad temporal del mismo es $O(n^3)$ (superior a la del algoritmo de Dijkstra) debido a que el problema que resuelve es el de la búsqueda del camino óptimo entre cada par de vértices del grafo.

Los SIG convencionales presentan una limitación de carácter expresivo, ya que solo pueden considerar un atributo asociado a cada segmento de calle en una red de viales mientras se realiza la planeación de las rutas [118]. En [119] se da una solución a esta limitación pero no se aborda la problemática del tamaño de la red.

En [120] se realiza un análisis de diferentes algoritmos para realizar cálculo de caminos óptimos. El mismo presenta un resultado que define cómo personalizar la búsqueda de dichos caminos

utilizando múltiples criterios, estos criterios se pueden definir atendiendo a la distancia, puntos de interés, etcétera. Este algoritmo utiliza como base el algoritmo A*, por lo que no garantiza la obtención del óptimo en todos los casos.

1.2.2.1. Transporte multimodal

En un entorno urbano, sobre todo en países desarrollados, se cuenta con varios tipos de transporte público. Por ejemplo: autobús, metro, etc.; por lo que en un entorno de este tipo, es importante tener en cuenta todas las redes de transporte existentes cuando se realiza análisis de rutas. En este sentido, en los últimos años se han desarrollado varios algoritmos para su implementación en SIG.

Shunying [121] propone un algoritmo para el cálculo de caminos óptimos en el transporte público. En el mismo se asume, a partir del resultado de investigaciones [122, 123], determinadas preferencias de los pasajeros. En [124] se muestra un sistema que integra información heterogénea de diversos sistemas de información de tráfico en una plataforma de información de tráfico urbano. En el modelo utilizado se representa una única red de transporte relacionando los puntos comunes (con aristas) de los distintos tipos de redes (tren, automóviles, etc.). En el mismo se describe una variante del algoritmo MSVPP [125] que incluye restricciones respecto al tiempo.

Los trabajos mencionados anteriormente, hacen uso de información adicional para disminuir el tiempo de búsqueda de caminos óptimos, por lo que para su aplicación es necesario, en primer lugar, contar con dicha información. Existen países, como Cuba, que por su nivel de desarrollo no cuentan con la disponibilidad de dicha información.

Por otra parte, en una red de transporte multimodal, la búsqueda de un camino depende de varias restricciones, tales como: la secuencia de los tipos de transporte, la cantidad de transferencias, los horarios, entre otros.

Debido a esto, se descartan los algoritmos para este tipo de red en la solución del problema planteado en esta investigación.

1.2.2.2. Sistemas de navegación en automóviles

En [126] se propone una variante del algoritmo Dijkstra con el objetivo de disminuir el tiempo de respuesta de este último. Esto se hace restringiendo el espacio de búsqueda del camino óptimo a los vértices que se encuentran dentro de un rectángulo o dentro de un hexágono (estático o dinámico). Para definir la figura geométrica que se utilizará se hace uso del siguiente criterio: es muy probable que el camino óptimo entre dos puntos (origen,destino) esté situado en el espacio alrededor de la línea recta que los une. La aplicación de este criterio en determinadas ciudades es un resultado importante ya que disminuye considerablemente el tiempo de respuesta, pero el mismo introduce un error porque no se puede demostrar que se alcanza el óptimo en todos los casos posibles.

En [127] se muestra un algoritmo heurístico para solucionar el problema de búsqueda de caminos óptimos con múltiples destinos en equipos embebidos ARM9 teniendo en cuenta las características de las calles (calles de un solo sentido, restricciones de tránsito, etc.) y provee las siguientes funcionalidades:

- Cálculo de un camino óptimo o satisfactorio desde un origen hasta múltiples destinos.
- Cálculo del camino más corto, el de menor costo o el más rápido de acuerdo con las preferencias del chofer.
- Cálculo de caminos en tiempo real teniendo en cuenta las condiciones de las calles así como las señales de tránsito.
- Muestra el resultado en forma de mapa en la terminal de a bordo.

Debido a las limitaciones de memoria que tiene el equipamiento que se utiliza en los vehículos para el cálculo de las rutas, es necesario modificar los algoritmos clásicos para lograr un mayor rendimiento del sistema. Para cumplir con este objetivo se definen heurísticas, las cuales introducen un margen de error, por lo que disminuye la precisión del algoritmo [127]. Este error en muchos casos puede ser aceptable en dependencia del problema que se resuelve, pero no siempre es beneficioso tenerlo.

En Cuba hay carencia de datos de tráfico urbano y de tecnologías (Sistema de Posicionamiento Global (GPS), conectividad inalámbrica, etc.), por lo que no es conveniente desarrollar (en este

momento) un sistema de navegación para automóviles o un sistema de información de transporte multimodal debido al bajo nivel de aplicación que pueden tener.

1.2.3. Algoritmos de reducción de grafos

Una de las soluciones a la reducción del tiempo de respuesta en la búsqueda de caminos óptimos en un grafo ha sido reducir el grafo sobre el que se realiza el análisis, debido a que dicho tiempo depende en gran medida de la cantidad de vértices del grafo en cuestión. La reducción de grafos consiste en obtener un grafo de menor tamaño (menos aristas o vértices) que mantenga las características principales o relevantes del grafo original, de forma tal que se puedan realizar análisis sobre el grafo reducido y llegar a conclusiones sobre el grafo original. Ejemplos de reducciones de grafo estudiadas en la bibliografía incluyen: el diámetro del grafo, corte óptimo (minimal cut), número y clasificación de cliques, entre otros [128].
La reducción de grafos tiene aplicación en varias temáticas relacionadas con la computación, tales como:

- Redes de workflow.
- Redes de computadoras y cómputo distribuido.
- Procesamiento paralelo y distribuido.

A continuación, se realiza un análisis crítico de los algoritmos de reducción de grafos estudiados, agrupados por áreas de aplicación.

1.2.3.1. Redes de workflow

Un modelo de proceso es una plantilla a partir de la cual se crea una instancia de cada proceso que se debe llevar a cabo en una organización. Hay procesos que no son ejecutados completamente en la computadora. Un caso típico es un proceso en el que un documento debe ser revisado por una persona. De forma general, un proceso está constituido por actividades, parte de las cuales se realizan con la ayuda de sistemas informáticos. Las partes que son ejecutadas sobre una computadora constituyen el modelo de workflow [129].
La corrección de un modelo de workflow es la premisa básica para alcanzar los objetivos de un negocio, por lo que se necesita un mecanismo que facilite la verificación de la corrección

de dicho modelo. Para realizar esta tarea generalmente se tienen en cuenta dos aspectos: la corrección estructural del modelo, es decir, que el proceso termine normalmente si no ocurren circunstancias inusuales y la otra es la corrección semántica del modelo, lo que implica que el proceso de workflow debe poder alcanzar los objetivos deseados al final de la ejecución.

En la literatura se utiliza el modelo Red de Petri para la representación de redes de workflow [130, 131], el mismo tiene facilidades para la verificación de la corrección estructural del modelo. Este modelo puede introducir los conflictos estructurales Deadlock y Lack of Synchronization. Un método utilizado para eliminar estos conflictos está basado en la reducción de grafos, el mismo consiste en identificar ciertas estructuras (por ejemplo, ciclos) y simplificarlas como se muestra en [132, 133, 134]. En [135] se expone un conjunto de reglas de reescritura que permiten reducir el modelo así como las pruebas de corrección y completitud del algoritmo presentado para reducir la red. Luego de este proceso, se obtiene una red de Petri reducida y se procede a comprobar la corrección estructural del modelo de workflow que representa la red de Petri.

Este tipo de reducción tiene pérdida de información que es vital para la solución de algunos problemas. Por ejemplo, en una red de viales la existencia de ciclos (rotondas, calles alrededor de una cuadra o manzana) es algo a tener en cuenta y que puede influir en el resultado de una búsqueda de caminos óptimos; por lo que si se pierde esta información se pudieran dar resultados erróneos.

1.2.3.2. Redes de computadoras, procesamiento paralelo y distribuido

Los sistemas de cómputo distribuido han recibido una alta atención en los últimos años. Uno de los problemas más estudiados es el problema de consenso (consensus problem) descrito en [136]. En [137] se muestra una revisión de los algoritmos utilizados para resolver este problema. En [138] se enuncia un problema que se puede deducir del problema del consenso: ¿es posible reconstruir el estado de la red completa conociendo el estado de un número limitado de nodos?; el cual se define como un problema de observabilidad y se propone un marco de trabajo matemático, haciendo uso de la teoría de grafos, para la solución del problema de la observabilidad en un sistema que ejecuta un algoritmo de consenso. Dicho marco de trabajo se basa en la reducción de la complejidad del problema estudiando subgrafos escogidos

convenientemente.

Los algoritmos de búsqueda de caminos óptimos, se rigen por el principio de Bellman [139], que especifica que cualquier subsecuencia de una secuencia óptima es a su vez óptima, por lo que la búsqueda de camino óptimo no se puede reducir a la búsqueda del camino óptimo en dos subgrafos para luego unir los caminos encontrados y dar el camino completo.

También se utilizan las técnicas de reducción de grafos en problemas de reconfiguración de antenas. Recientemente los diseñadores de antenas han utilizado conmutadores accionados (actuated switches) para lograr la reconfiguración [140, 141]. Para instalar los mismos en las antenas es necesario adquirir hardware, generalmente costoso, para su activación y desactivación. En [142] se propone una solución a este problema, la misma consiste en optimizar la cantidad de conmutadores utilizados para reconfigurar una antena eliminando redundancias de la estructura (el grafo que representa la red) con el objetivo de reducir costos y pérdidas.

En [143] se muestra un procedimiento denominado Network Graph Reduction, en el mismo se modifica el grafo que describe una red de computadoras antes de realizar un cómputo sobre la misma, esto se hace excluyendo las aristas (tramos de red) que están congestionadas.

En [144, 145, 146, 147, 148, 149, 150, 151, 152] se enuncian varios modelos para reducir un grafo, aplicado al enrutamiento en redes de computadoras. En [153] se muestra que estos modelos aún carecen del rendimiento necesario para dar una respuesta rápida y en el mismo se define un algoritmo de reducción de grafos que resuelve esta situación, este se basa en eliminar aristas o vértices que “no son necesarios” para dar solución a determinados problemas.

En los algoritmos expuestos en este epígrafe, se elimina información de la red que es redundante para el problema que se enfrenta en cada situación. En el ámbito de análisis de redes de transporte, no se puede eliminar información (posiblemente redundante para otros problemas) ya que se pueden introducir errores en los cálculos que se realicen.

Con el nombre Model checking se conoce a un método automático de verificación de un sistema formal. El sistema se describe mediante un modelo que debe responder a una especificación formal descrita mediante una fórmula y que suele estar expresado como un grafo dirigido donde los vértices representan los posibles estados de un sistema y las aristas las posibles evoluciones (ejecuciones) del mismo, mientras que las proposiciones representan las propiedades básicas

que se satisfacen en cada punto de la ejecución.

Existen herramientas automáticas para realizar Model checking, basadas en técnicas combinatorias, explorando el espacio de estados posibles; lo que conduce al problema de explosión de estados [154]. Para evitarlo se han desarrollado varias técnicas, entre ellas reducción de orden parcial, la cual es una técnica que considera solo algunas ejecuciones representativas para realizar la verificación de una propiedad determinada, eliminando estados "redundantes". Esta técnica ha sido descrita también como model checking with representatives en [155].

En [156] se muestra un algoritmo de reducción paralela de orden parcial (Parallel Partial Order Reduction) para el chequeo de modelos LTL con el objetivo de reducir el número de estados del grafo espacio. La idea se basa en la observación siguiente: para propósitos de verificación, varias ejecuciones de sistemas son equivalentes respecto a las propiedades verificadas. Como resultado un algoritmo que implemente reducción de orden parcial puede evitar la generación de algunas ejecuciones de un sistema informático, a partir de que el mismo explora al menos una ejecución representativa de cada clase de equivalencia.

En [157] se propone un algoritmo llamado Distributed Asynchronous Constraint Optimization (DACO) basado en reducciones dinámicas de grafos. El algoritmo DACO permite a los nodos de procesamiento enviar sus soluciones parciales a otros nodos de procesamiento y desaparecer del procesamiento paralelo. En dicho algoritmo, los agentes deciden el destino de los mensajes a partir del grafo, por lo que no necesitan generar el árbol DFS. En la inicialización del algoritmo, cada agente notifica el grado (cantidad de vecinos) a todos los agentes adyacentes. Cuando un agente determina que tiene el mayor grado en su área local (entre los agentes vecinos y el propio agente), envía su solución parcial y su conocimiento a uno de los agentes adyacentes y desaparece. Este agente no volverá a participar en algún proceso de solución. El hecho de que el agente desaparezca causa la reducción dinámica del grafo.

Este tipo de algoritmo utiliza conceptos que se van más allá de la teoría clásica de búsqueda de caminos óptimos y su aplicación específica es para la verificación de modelos y la implementación de sistemas multiagente respectivamente. En este ámbito es posible realizar determinados análisis eliminando vértices "redundantes" o que no son decisivos para la solución

del problema que se resuelve, pero en el caso de la búsqueda de caminos óptimos no existen vértices que puedan ser obviados por los algoritmos y que permitan obtener en todos los casos un resultado óptimo, por lo que se descarta el uso de estos algoritmos en el ámbito de los SIG.

1.2.3.3. Otros algoritmos

En [158] se define un método de reducción basado en el grafo potencia. Este método utiliza una estrategia de poda del grafo mencionado, la cual reduce en gran medida el almacenamiento y el cómputo y mejora el rendimiento del algoritmo a expensas de la pérdida de información.
El análisis simbólico de circuitos se refiere a la búsqueda de una función de red (función de transferencia) de una señal a otra expresada en términos de símbolos de los elementos de circuitos. Varios métodos han sido estudiados a partir de 1950. Como se señala en [159], la mayoría de los métodos clásicos no son capaces de hacer frente a los circuitos integrados a gran escala debido a la complejidad de implementación. Los métodos clásicos y contemporáneos se describen en [160, 161].
En [162] se enuncia una propuesta de aplicación de la reducción de grafos en el análisis de circuitos; se formula un mecanismo Binary Decision Diagram (BDD), junto con un proceso de reducción de grafos específico. Para reducir se define un orden entre los símbolos que serían los vértices del grafo.
Otros trabajos se refieren a reducir grafos que cumplen determinadas propiedades, para lo cual realizan la reducción encontrando determinadas subestructuras geométricas [163].
Estos algoritmos obtienen un grafo reducido con información suficiente para dar solución a un problema específico, pero limitada para la búsqueda de caminos óptimos de forma general.
el mismo presenta

1.3. Conclusiones del capítulo

Luego de realizar un estudio de los modelos para la representación y análisis de redes en SIG, los algoritmos para la búsqueda de caminos óptimos y los algoritmos de reducción de grafos existentes, se concluye lo siguiente:

- Los modelos que se utilizan en los SIG en la actualidad no permiten modelar de forma

natural una red de transporte, con el consecuente aumento en la complejidad de los algoritmos utilizados para el análisis de las mismas y las limitaciones de escalabilidad.

- Los algoritmos de búsqueda de caminos óptimos clásicos no presentan un rendimiento adecuado cuando el grafo sobre el que se ejecuta es grande.
- Se han desarrollado varios algoritmos para la búsqueda de caminos óptimos en grafos grandes pero no se hace referencia a la forma de almacenamiento de los mismos para garantizar escalabilidad y eficiencia.
- En los algoritmos de búsqueda de caminos óptimos estudiados se utilizan determinadas heurísticas específicas de un problema o del lugar donde se aplica la implementación del algoritmo, lo que trae consigo que no siempre se puede obtener el camino óptimo.
- Existe una tendencia en las investigaciones relacionadas con la disminución del tiempo de ejecución de la búsqueda de caminos óptimos. La misma está relacionada con la disminución del espacio de búsqueda de solución con la consecuente introducción de un error en el resultado, sin embargo no se realiza transformación en el grafo que representa la red, lo cual motiva el uso de algún mecanismo de reducción de grafos que permita realizar búsquedas de caminos óptimos garantizando escalabilidad y eficiencia.
- Los algoritmos de reducción de grafos estudiados eliminan información del grafo original, lo que trae consigo la introducción de un error en la búsqueda de caminos óptimos de forma general. A partir de esto se puede decir que no se cuenta con un mecanismo de reducción general que sea aplicable en varias ramas de la ciencia independientemente de la información que se represente.
- Existe la necesidad de desarrollar un modelo para la representación de redes basado en grafos reducidos, que garantice escalabilidad y eficiencia en la búsqueda de caminos óptimos cuando las redes son grandes.

Capítulo 2

Modelo basado en grafos reducidos para la representación y análisis de redes en Sistemas de Información Geoespacial

2. MODELO BASADO EN GRAFOS REDUCIDOS PARA LA REPRESENTACIÓN Y ANÁLISIS DE REDES EN SISTEMAS DE INFORMACIÓN GEOESPACIAL

En este capítulo se presenta un modelo para la representación y análisis de redes en SIG. Haciendo uso del mismo se realiza la búsqueda de caminos óptimos de forma eficiente y escalable cuando las redes son grandes.

Existe una amplia variedad de definiciones de modelo, a pesar de sus diferencias, todas tienen en común que un modelo es una representación teórica o gráfica de un fenómeno, en el mismo se trata de especificar los elementos fundamentales que caracterizan el fenómeno así como las relaciones entre estos.

En el marco de este trabajo se utilizará la definición de modelo que propone François E. Cellier, la cual plantea que un modelo (M) para un sistema (S) y un experimento (E) es cualquier cosa a la cual se le puede aplicar E para responder preguntas sobre S [164].

En la Figura 2.1 se muestra un esquema que representa los componentes del modelo que se propone. A continuación se expone en detalle cada uno de los componentes de este modelo así como las relaciones existentes entre los mismos.

2.1. Representación de una red mediante un grafo

Para representar una red mediante un grafo se parte de un mapa que contenga una capa vectorial que represente dicha red. En primer lugar se propone el uso de un algoritmo que obtenga las intersecciones entre cada par de segmentos de la red, para ello se propone el uso del algoritmo FINDINTERSECTIONS presentado en [165]; el mismo tiene complejidad temporal $O(n \lg n + I \lg n)$, donde n es la cantidad de segmentos e I la cantidad de intersecciones entre todos los

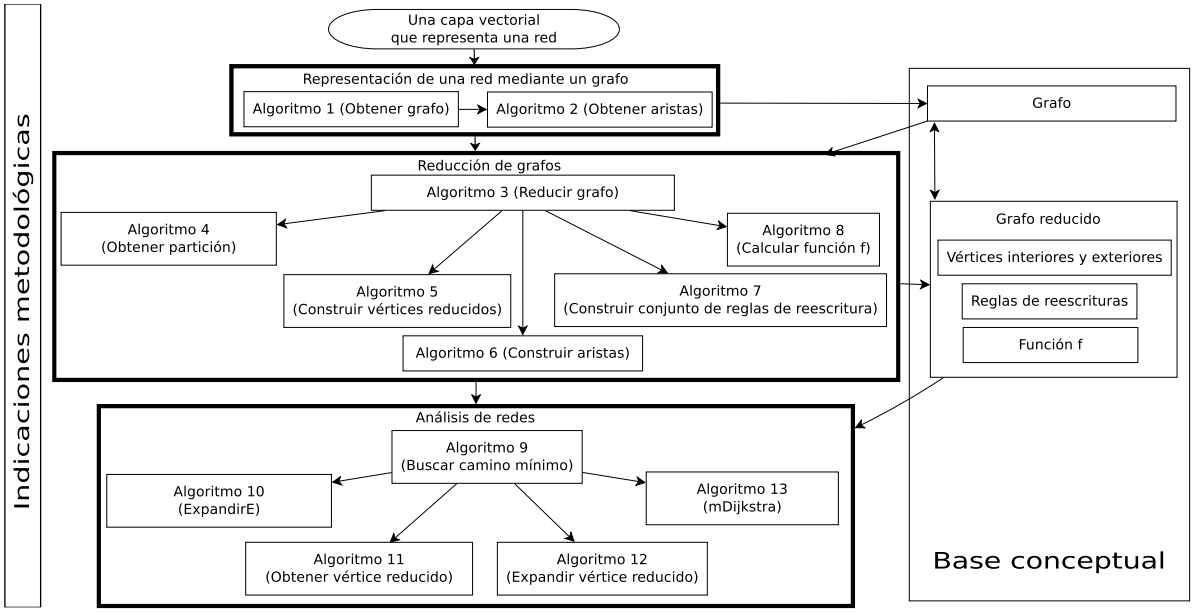


Figura 2.1: *Componentes del modelo propuesto.*

segmentos.

Este algoritmo tiene como salida el conjunto de intersecciones entre todos los segmentos que recibe como parámetro, así como los segmentos relacionados con cada intersección. A partir de estos datos, se puede crear un grafo haciendo uso del Algoritmo 1.

Algoritmo 1 $ObtenerGrafo$

Entrada: Lista de líneas (l_lineas) y una lista de puntos por cada segmento ($l\_intersecciones$) que representa las intersecciones que contiene cada segmento con el resto de los segmentos

Salida: Un grafo

1: $G = (\{\}, \{\})$
2: **Para todo** $linea \in l\_lineas$ **hacer**
3: $\quad aristas = DeterminarAristas(linea, l\_intersecciones[c])$
4: $\quad adicionarAristas(G, aristas)$
5: **Fin Para**
6: **Retornar** G

Para determinar las aristas (paso 3 del Algoritmo 1) se propone el Algoritmo 2. Se debe tener en cuenta que una arista está conformada por cuatro valores: vértice de origen, vértice de destino, costo de la arista (puede ser un vector de valores que represente distintos tipos de costo) y su geometría. Como esta arista modela una porción de una red existente en el mundo real, es importante tener acceso a su geometría. Esto puede ser útil si en un futuro se desea obtener

imágenes relacionadas con análisis realizados sobre el grafo. Por ejemplo, para dibujar en un mapa el camino óptimo buscado sobre una determinada red. En el diseño de este algoritmo se asume, sin perder generalidad, que existe una función denominada $Costo$ que devuelve el costo de una arista. Esto es necesario debido a que el costo de transitar una arista puede calcularse de diversas formas en dependencia de la información que exista sobre la misma. Por ejemplo, el costo puede ser la longitud de la arista, puede tenerse en cuenta variables como el tiempo, la calidad de la arista (la calle en caso de una red de viales), entre otros factores.

Algoritmo 2 $DeterminarAristas$

Entrada: Una línea ($linea$) que representa una porción de la red y una lista de puntos (l_puntos)

Salida: Un conjunto de aristas que representan la línea de entrada

1: $aristas = \{\}$
2: **Si** $l\_puntos[0] \neq linea[0]$ **entonces** $Insertar(l\_puntos, 0, linea[0])$ {Insertar en la primera posición de la lista de puntos el primer punto de la línea que se analiza}
3: **Si** $l\_puntos[Longitud(l\_puntos)] \neq linea[Cantidad\_de\_Puntos(linea)]$ **entonces**
4: $Adicionar(l\_puntos, linea[Cantidad\_de\_Puntos(linea)])$ {Adicionar al final de la lista de puntos el último punto de la línea que se analiza}
5: **Fin Si**
6: Calcular la distancia desde el primer punto de la línea hasta cada punto de la lista de puntos haciendo uso de la geometría
7: Ordenar la lista de puntos según la distancia calculada
8: **Para** $i = 0$ hasta $i < Longitud(l\_puntos) - 1$ **hacer**
9: $a = ExtraerGeometria(linea, l\_puntos[i], l\_puntos[i+1])$
10: $Adicionar(aristas, l\_puntos[i], l\_puntos[i+1], Costo(a), a)$
11: **Fin Para**
12: **Retornar** $aristas$

2.1.1. Análisis de complejidad

En este epígrafe se calcula la complejidad temporal, según el enfoque teórico, de los algoritmos 1 y 2 propuestos en el epígrafe anterior. Para el cálculo de la misma se determina la complejidad de cada paso del algoritmo y se utilizan las reglas de la suma y la multiplicación de la notación asintótica "o grande" (O) para obtener la complejidad de dichos algoritmos.

Análisis del Algoritmo 2: Determinar las aristas del grafo que representa la red

Haciendo uso de una implementación de lista enlazada con apuntador al inicio y al fin de la lista, se pueden realizar las operaciones de inserción y adición en un tiempo $O(1)$. A partir de este criterio, se muestra la complejidad de cada paso del algoritmo.

- Pasos 1-5 $O(1)$.
- Paso 6 $O(m)$, m es la cantidad de segmentos que contenga la línea recibida por parámetro.
- Paso 7 $O(n \lg(n))$, n es la cantidad de puntos recibidos por parámetro o la cantidad de intersecciones sobre la calle. Este tiempo se logra haciendo uso de un algoritmo de ordenamiento como el algoritmo heapsort.
- Pasos 8-11 $O(n * m)$.
    - Paso 9 $O(m)$.
    - Paso 10 $O(1)$.

Aplicando la regla de la suma se puede concluir que la complejidad del Algoritmo 2 es $O(n*m)$.

Análisis del Algoritmo 1: Obtener un grafo que representa una red

Sea n la cantidad de líneas existentes en la capa del mapa que constituye la entrada del modelo, m el mayor valor entre las cantidades de segmentos de dichas líneas e I la cantidad de intersecciones entre todos los segmentos.
A continuación se muestra la complejidad de cada paso del algoritmo.

- Paso 2-5 $O(n * I * m)$.
    - Paso 3 $O(I * m)$.
    - Paso 4 $O(I)$.

Se puede concluir que la complejidad es $O(n * I * m)$.
La capa del mapa que representa la red puede ser construida de forma tal que un elemento de la red (una calle en el caso de una red de viales) esté divido en varios segmentos según se interseque con otros elementos; sistemas como OpenStreetMap almacenan los mapas siguiendo este principio. Si el mapa de entrada cumple con la condición anterior, la complejidad estaría dada por $O(n * I)$.

2.2. Reducción de grafos

En este epígrafe se presentan definiciones y algoritmos relacionados con el proceso de reducción de grafos propuesto como parte de la presente investigación. Se muestra además la complejidad temporal de los algoritmos enunciados.

En las gramáticas de grafo (ver Definición 1.1.3.15) reviste particular importancia la definición de regla de reescritura. Este formalismo es utilizado para transformar un grafo en otro siguiendo determinados principios.

En el modelo propuesto, las reglas de reescritura de grafo se utilizan con el objetivo de garantizar que no exista pérdida de información. En el caso de la presente investigación, siempre estarán referidas a un solo vértice. Por otra parte, la función ψ se debe sustituir por las funciones ψ_{in} y ψ_{out}; las cuales especificarán la información de empotrado relacionada con las aristas que entran y salen respectivamente del vértice relacionado con la regla de reescritura.

Además, la imagen de las funciones ψ_{in} y ψ_{out} es el conjunto $\{0, 1\}$ por lo que, desde un punto de vista práctico, se puede asumir que se especifican explícitamente solo los valores para los cuales la imagen de las funciones toma el valor uno y para los que toma el valor cero no se representan en la regla de reescritura. Teniendo esto en cuenta, las funciones antes mencionadas se pueden definir como cuádruplos de la forma $(V_j, \mathbb{R}^+, \mathbb{R}^+, (V - \{v_i\}))$.

A continuación se introducen las definiciones de regla de reescritura de grafos, grafo reducido y grafo reducido a partir de un grafo, las cuales pertenecen al autor del presente trabajo [166].

DEFINICIÓN 2.2.1 *Una regla de reescritura de grafos sobre un grafo $G = (V, E, f_c)$ es un cuádruplo de la forma $(G_i, G_j, \psi_{in}, \psi_{out})$, donde:*

- *$G_i = (\{v_i\}, \{\})$ es un grafo, donde $v_i \in V$.*
- *$G_j = (V_j, E_j)$ es un grafo.*
- *ψ_{in}, ψ_{out} son dos conjuntos formados por cuádruplos de la forma (v_m, c_1, c_2, v_n), donde: $c_1, c_2 \in \mathbb{R}^+, v_m \in V_j, v_n \in (V - \{v_i\})$.*

Para que un cuádruplo (v_m, c_1, c_2, v_n) pertenezca a ψ_{in}, se debe cumplir que exista la arista (v_n, v_i) en el grafo G; además, el costo de la arista debe ser c_1. Luego de aplicar la regla de reescritura, se obtiene el grafo $G_1 = (V_1, E_1, f_{c1})$ y se cumple que la arista (v_n, v_m) pertenece al nuevo grafo y tiene costo c_2. Análogamente se define ψ_{out}, con la única diferencia de la orientación de las aristas.

Las aristas que unen el vértice v_i y los vértices del grafo $G - G_i$ se denominan aristas preempotradas. Después de aplicar una regla de reescritura, las aristas que unen los vértices del

grafo G_j con los vértices del grafo $G - G_i$ se denominan aristas postempotradas. La información de empotrado ψ_{in} permite transformar el conjunto de aristas preempotradas que inciden en el vértice v_i en aristas postempotradas que inciden en uno o más vértices $v_j \in V_j$; de forma similar ψ_{out} permite transformar las aristas preempotradas que salen de v_i en aristas postempotradas que salen de uno o más vértices $v_j \in V_j$.

Teniendo en cuenta la definición de regla de reescritura enunciada anteriormente, en la Figura 2.2 se muestra una regla de reescritura utilizando el mecanismo NCE. Se puede comprobar, que luego de aplicar dicha regla al grafo de la Figura 2.3(a) según el mecanismo seleccionado, se obtiene el grafo de la Figura 2.3(b).

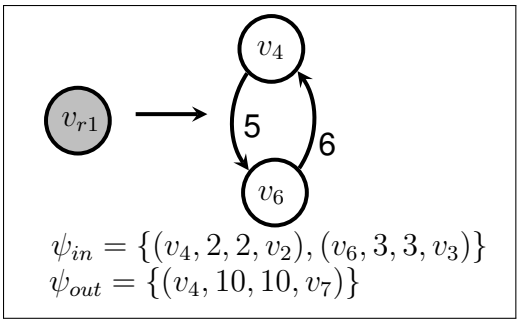
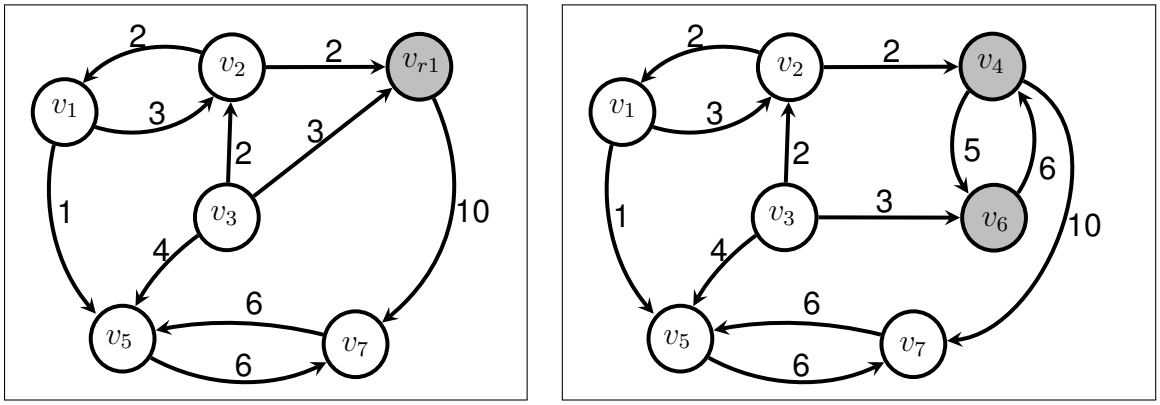


Figura 2.2: *Ejemplo de regla de reescritura sobre el grafo de la Figura 2.3(a). En la parte izquierda se muestra el grafo $G_i = (\{v_{r1}\}, \{\})$, en la parte derecha se muestra el grafo $G_j = (\{v_4, v_6\}, \{(v_4, v_6), (v_6, v_4)\})$ y en la parte inferior se muestra la información de empotrado ψ_{in} y ψ_{out}. Los vértices v_2 y v_3 que aparecen ψ_{in} pertenecen al grafo sobre el que definió la regla de reescritura.*



(a) Grafo reducido

(b) Ejemplo de grafo

Figura 2.3: *Ejemplos de grafos.*

Para referirse a un grafo de la regla de reescritura asociada a un vértice reducido v_r se utilizará la notación $v_r.G_i$ y $v_r.G_j$. Análogamente, se utilizará la notación $v_r.\psi_{in}$ y $v_r.\psi_{out}$ para referirse a los conjuntos ψ_{in} y ψ_{out} de la regla de reescritura asociada al vértice v_r.

DEFINICIÓN 2.2.2 *Un grafo reducido es un cuádruplo $G_r = (V_r, E_r, f, R)$, donde:*

- *V_r es un conjunto de vértices.*
- *E_r es un multiconjunto de aristas, donde cada arista es un par ordenado de vértices. Opcionalmente puede tener un valor que la identifique y una lista de atributos.*
- *$f : V_r \times V_r \times V_r \rightarrow (\mathbb{R}^+ \bigcup \{\infty\})$, es una función que para cada trío de vértices (v_i, v_j, v_k) retorna el costo de ir desde v_i hasta v_k a través de v_j, siendo v_k adyacente a v_j y v_j adyacente a v_i.*
- *R es un conjunto de reglas de reescritura sobre (V_r, E_r).*

Si el conjunto R de un grafo reducido es vacío, la función f puede ser calculada a partir de la función de costo del grafo ponderado de la siguiente forma: $f(v_i, v_j, v_k) = f_c(v_i, v_j) + f_c(v_j, v_k)$.

DEFINICIÓN 2.2.3 *Sean G y G_r dos grafos y $R = \{r_1, r_2, ..., r_n\}$ un conjunto de reglas de reescritura de grafos. Se puede afirmar que G_r es un grafo reducido a partir de G, si al aplicar las reglas de reescritura especificadas en R al grafo G_r se obtiene el grafo G.*

2.2.1. Algoritmo de reducción de grafos

Cuando se menciona el grafo original, se hace referencia al grafo que es entrada de la iteración del algoritmo que se esté ejecutando. Este grafo puede ser distinto al existente antes de ejecutar el algoritmo de reducción por primera vez.

El algoritmo de reducción diseñado se considera el principal aporte de la investigación ya que reduce un grafo sin que exista pérdida de información en el proceso, lo que contribuye a realizar análisis sobre el grafo reducido y obtener los mismos resultados que se obtienen en el grafo sin reducir. Además, el hecho de que no exista pérdida de información hace posible su uso en la reducción de varios tipos de grafos.

La propuesta tiene como entrada un grafo y una partición sobre los vértices del grafo de entrada. Sin embargo, puede ser necesario refinar esta partición; ya que para obtener un camino en

el grafo reducido, de igual costo que el que se obtiene con el algoritmo de Dijkstra en el grafo original, es necesario que en dicho grafo reducido no existan pares de vértices que sean adyacentes y a su vez reducidos. Para contribuir a garantizar tal propósito, se introduce la siguiente definición:

DEFINICIÓN 2.2.1.1 *Sea un grafo $G = (V, E)$ y una partición P sobre V, un vértice $v_i \in V$ es interior si $\forall v_j \in V$, tal que v_i y v_j son adyacentes, se cumple que v_i y v_j pertenecen a la misma clase de P. Un vértice v_i es exterior si $\exists v \in V$, tal que $((v, v_i) \in E$ o $(v_i, v) \in E)$ y v_i y v_j no pertenecen a la misma clase de P.*

El primer paso del algoritmo de reducción consiste en refinar la partición que es entrada de dicho algoritmo. Para ello se sigue la siguiente estrategia:

- Dos vértices están en la misma clase de la partición si y solo si:
  - Están en la misma clase de P.
  - Son vértices interiores.
- Si un vértice es exterior, se crea una nueva clase que contiene solo a dicho vértice.

En la Figura 2.4 se ilustra con un ejemplo el uso de la definición de vértice interior y exterior para refinar la partición que es entrada del algoritmo.

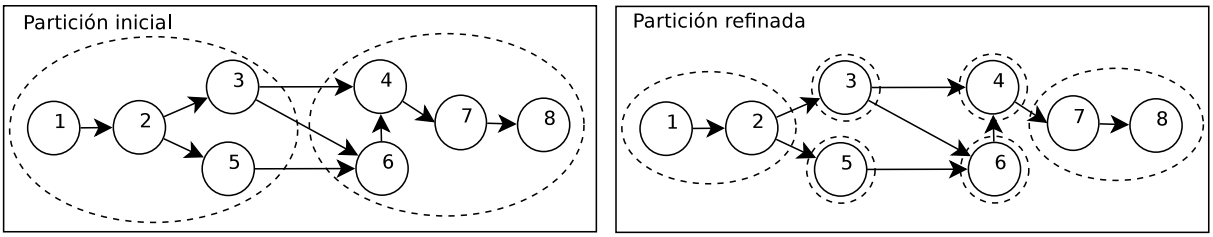


Figura 2.4: *Ejemplo de refinamiento de la partición para crear los vértices del grafo reducido. En el grafo de la izquierda se puede apreciar que los vértices 3, 4, 5 y 6 son exteriores; por cada uno de dichos vértices se creará una clase en la nueva partición.*

Teniendo en cuenta lo anterior, se propone el Algoritmo 3 para refinar una partición.

Algoritmo 3 $Refinar Particion$

Entrada: Un grafo $G = (V, E)$ y una partición P sobre el conjunto de vértices V

Salida: Una partición Pa

```
1:  Pa = {}
2:  Para todo v_i ∈ V hacer
3:      Int_v = {}
4:      Para todo v_j ∈ V, v_j ≠ v_i hacer
5:          Si v_i y v_j pertenecen a la misma clase de P and EsInterior(G, P, v_j) entonces
6:              Adicionar(Int_v, v_j)
7:          Sino Si v_i y v_j pertenecen a la misma clase de P entonces
8:              Adicionar(Pa, {v_j})
9:          Fin Si
10:     Fin Para
11:     Si Int_v ≠ φ entonces Adicionar(Pa, Int_v)
12: Fin Para
13: Retornar Pa
```

Por cada clase de la partición refinada se crea un vértice en el grafo reducido. Si la cardinalidad de la clase es mayor que uno, el vértice que se crea se considera reducido; en otro caso se considera no reducido. Además, se propone el uso de una función (en este caso $ObtenerNombre$) que a partir de una clase de la partición, devuelve un identificador que será asociado a dicho vértice. En caso de que la clase tenga cardinalidad mayor que uno, la función retornará el valor del atributo utilizado para crear la clase; si la cardinalidad es uno, se retorna el identificador del vértice correspondiente a la clase en el grafo original.

Para construir el conjunto de vértices V_r del grafo reducido a partir de la partición P se utiliza el Algoritmo 4.

Las aristas del grafo reducido se crean a partir de las clases de la partición refinada y del grafo original. Para ello se analiza cada par de clases de la partición, y si existe una arista entre dos vértices que pertenecen a clases distintas, se adiciona al grafo reducido.

A medida que se van adicionando las aristas al grafo reducido se debe ir actualizando la función de costo f_r del grafo reducido G_r que se devolverá como salida del algoritmo de reducción. El pseudo-código para crear las aristas se muestra en el Algoritmo 5.

Para construir las reglas de reescritura del grafo reducido se parte de la Definición 2.2.1. Se crea una regla por cada vértice reducido (o por cada clase de la partición que tenga cardinalidad mayor que uno). Este es un paso esencial en el algoritmo de reducción, es el que permite que no

Algoritmo 4 $ConstruirVerticesReducidos$

Entrada: Una partición $P = \{A_1, A_2, \ldots, A_s\}$

Salida: El conjunto de vértices V_r formado por s vértices

1: V_r={}
2: **Para todo** $A_i \in P, i = 1..s$ **hacer**
3: **Si** $|A_i|$>1 **entonces**
4: $Adicionar(V_r, ObtenerNombre(A_i))$ {La función $ObtenerNombre$ devuelve el valor del atributo utilizado para crear la clase A_i.}
5: **Sino**
6: $Adicionar(V_r, ObtenerNombre(A_i))$ {En caso que $|A_i| = 1$, la función $ObtenerNombre$ devuelve el identificador, en el grafo original, del único vértice que pertenece a la clase.}
7: **Fin Si**
8: **Fin Para**
9: **Retornar** V_r

Algoritmo 5 $ConstruirAristas$

Entrada: Una partición $P = \{A_1, A_2, \ldots, A_s\}$ y un grafo reducido $G = (V, E, f, R)$

Salida: El conjunto de aristas E_r

1: **Para todo** $e_m \in A_i, e_n \in A_j, i \neq j$, tal que $i, j = 1..s$ **hacer**
2: $v_i = ObtenerNombre(A_i), v_j = ObtenerNombre(A_j)$
3: **Si** $(e_m, e_n) \in E$, **and** $f(e_m, e_m, e_n) < f(v_i, v_i, v_j)$ **entonces**
4: $Adicionar(E_r, v_i, v_j)$
5: $f(v_i, v_i, v_j) = f(e_m, e_m, e_n)$
6: **Fin Si**
7: **Fin Para**
8: **Retornar** E_r

haya pérdida de información en la reducción del grafo y además garantiza que se pueda obtener el grafo original a partir del grafo reducido.

Según la Definición 2.2.1, una regla de reescritura sobre un grafo $G = (V, E, f_c)$ es un cuádruplo de la forma $(G_i, G_j, \psi_{in}, \psi_{out})$. En este paso se crea una regla de reescritura por cada clase $A_i, |A_i| > 1$ como se muestra en el Algoritmo 6, donde:

- $G_i = (\{v_i\}, \{\})$, $A_i = [v_i]$.
- $G_j = (A_i, E_i, f_{cj})$ es un subgrafo de $G = (V, E, f_c, R)$, donde $\exists (u, v) \in E_i$ si y solo si $(u, v) \in E$ y $u, v \in A_i$, además $f_{cj}(u, v) = f_c(u, v)$.
- El conjunto ψ_{in} está formado por cuádruplos de la forma (v_m, c_1, c_2, v_n), que deben cumplir lo siguiente: $c_1 = c_2 = f_c(v_n, v_i), v_m \in A_i, v_n \in (V - A_i), \exists (v_n, v_i) \in E$.
- Análogamente, el conjunto ψ_{out} está formado por cuádruplos de la forma (v_m, c_1, c_2, v_n)

que deben cumplir lo siguiente: $c_1 = c_2 = f_c(v_i, v_n), v_m \in A_i, v_n \in (V - A_i), \exists (v_i, v_n) \in E$.

Algoritmo 6 $ConstruirReglasReescritura$

Entrada: Una partición $P = \{A_1, A_2, \ldots, A_s\}$ y un grafo reducido $G = (V, E, f, R)$

Salida: El conjunto de reglas de reescritura R

1: $R = \{\}$
2: **Para todo** $A_i \in P, i = 1..s$, tal que $|A_i| > 1$ **hacer**
3: $G_i = (\{ObtenerNombre(A_i)\}, \{\})$,$V_j = A_i$,$E_j = \{\}$
4: **Para todo** $v_m, v_n \in V_j, m \neq n$ **hacer**
5: **Si** $(v_m, v_n) \in E$ **entonces** $Adicionar(E_j, (v_i, v_j))$, $f_j(v_i, v_i, v_j) = f(v_i, v_i, v_j)$
6: **Si** v_j es reducido en G **entonces**
7: **Para todo** $v_k \in Adyacentes(v_j, G)$ **hacer**
8: $f_j(v_i, v_j, v_k) = f(v_i, v_j, v_k)$
9: **Fin Para**
10: **Fin Si**
11: **Fin Para**
12: $R_j = ObtenerR(V_j, R)$ {Obtiene las reglas de reescritura asociadas a V_j. Note que si no es la primera vez que se reduce el grafo original, G_j puede ser un grafo reducido y se deben tener en cuenta las reglas de reescritura asociadas a los vértices del mismo.}
13: $G_j = (V_j, E_j, f_j, R_j)$
14: **Para todo** $v_k \in V_j$ **hacer**
15: $ady = AristasQueEntran(v_k, G, [v_k] - V_j)$ {Devuelve todos los vértices $v \in ([v_k] - V_j)$, tal que $\exists (v, v_k) \in E$}
16: **Para todo** $v \in ady$ **hacer**
17: $Adicionar(\psi_{in}, (v, f(v, v, v_k), f_c(v, v, v_k), v_k))$
18: **Fin Para**
19: $ady = AristasQueSalen(v_k, G, [v_k] - V_j)$ {Devuelve todos los vértices $v \in ([v_k] - V_j)$, tal que $\exists (v_k, v) \in E$}
20: **Para todo** $v \in ady$ **hacer**
21: $Adicionar(\psi_{out}, (v, f(v_k, v_k, v), f(v_k, v_k, v), v_k))$
22: **Fin Para**
23: **Fin Para**
24: $Adicionar(R, (G_i, G_j, \psi_{in}, \psi_{out}))$
25: **Fin Para**
26: **Retornar** R

En el caso de la reducción de un grafo, teniendo en cuenta el costo de las aristas para la búsqueda de caminos óptimos, los valores de c_1 y c_2 son iguales. Sin embargo, de forma general estos valores pueden ser distintos.

En primer lugar, se crea el grafo G_i; el mismo está formado por un vértice que representa la clase correspondiente de la partición refinada. Luego se crea el grafo G_j, formado por todos los vértices que pertenecen a la clase en cuestión y las aristas existentes entre dichos vértices en el

grafo original. Para crear la información de empotrado, por cada arista del grafo original que incide en vértices que pertenecen a la clase de la partición, se adiciona un cuádruplo al conjunto ψ_{in} y por cada arista del grafo original que sale de un vértice de la clase hacia un vértice que no pertenece a dicha clase se adiciona un cuádruplo en ψ_{out} que representa la arista en cuestión. De esta forma se garantiza que cuando se apliquen las reglas de reescritura al grafo reducido, se obtenga un nuevo grafo con los mismos vértices y aristas del grafo original; por lo que se puede afirmar que el proceso de reducción es reversible y no presenta pérdida de información.

El último paso consiste en calcular la función f del grafo reducido. Esta función almacena, para cada trío de vértices (v_i, v_j, v_k), con v_k adyacente a v_j y v_j adyacente a v_i, el costo de ir desde v_i hasta v_k a través v_j. Además, dicha función puede ser vista como el mecanismo para almacenar el costo de pasar por un vértice reducido.

Para calcular la función f, se crea un grafo auxiliar por cada vértice reducido a partir del grafo G_j de la regla de reescritura correspondiente. El objetivo de este grafo auxiliar es tener un grafo sobre el que se pueda realizar modificaciones para utilizarlo en el cálculo de la función antes mencionada.

Luego de adicionar los vértices y aristas del grafo G_j al grafo auxiliar, se adicionan los vértices adyacentes a cada vértice que pertenezca al grafo G_j. Estos vértices adyacentes se almacenan en la variable $verticesAdyacentes$ para ser utilizados en pasos posteriores.

Sobre el grafo auxiliar creado se aplica el algoritmo de búsqueda de caminos en grafos reducidos propuesto en el epígrafe 2.3.1 (ver Algoritmo 12), tomando como vértice de origen (v_o) cada uno de los vértices almacenados en la variable $verticesAdyacentes$; cada vez que se invoca este algoritmo se debe ir actualizando la función f como se muestra en el paso 15.

La función calculada almacena, además del costo de ir de un vértice a otro pasando por uno reducido, el propio camino; o sea, la secuencia de vértices a seguir de forma tal que con este valor precalculado se reduce el tiempo necesario para mostrar el camino óptimo.

Para calcular el camino antes mencionado se utiliza la función $Camino$. La implementación de esta función parte del hecho que Pr es un vector que representa los predecesores de cada vértice en el camino óptimo desde el vértice de origen (v_o) hasta el resto de los vértices del grafo. Para obtener el camino desde v_o hasta v_d se recorre el vector Pr calculando cuál es el predecesor

v_k del vértice v_d ($v_k = Pr[v_d]$), después se calcula el predecesor de v_k, este proceso se repite hasta que el predecesor de algún vértice del camino coincida con el vértice de origen v_o; luego la concatenación de cada vértice, en orden inverso al encontrado, sería el camino óptimo desde v_o hasta v_d.

De la explicación anterior se deduce que la complejidad del paso 15 es $O(n)$, donde n es la cantidad de vértices del grafo G_{aux}.

El pseudo-código para calcular la función f se muestra en el Algoritmo 7.

Algoritmo 7 $Calcular f$

Entrada: Un grafo reducido $G = (V, E, f_r, R)$, los subgrafos $G_i = (\{v_i\}, \{\})$ y $G_j = (A_i, E_i, f_j, R_j)$ de la regla de reescritura asociada a una clase de equivalencia A_i de la partición P y la función f (este último parámetro es pasado por dirección)

Salida: La función f para el vértice reducido correspondiente a A_i

1: Crear un grafo auxiliar $G_{aux} = (A_{aux}, E_{aux}, f_{aux}, R_{aux}) = G_j = (A_i, E_i, f_j, R_j)$
2: $verticesAdyacentes = \{\}$ {Inicializar la variable $verticesAdyacentes$}
3: **Para todo** $v_i, v_j, v_i \in A_{aux}, v_j \in Adyacentes(v_i, G)$ **hacer**
4: **Si** $v_j \notin A_i$ **entonces**
5: $AdicionarVertice(G_{aux}, v_j)$
6: $AdicionarArista(G_{aux}, v_i, v_j)$
7: $f_{aux}(b, b, v) = f_r(v_j, v_j, v_i)$
8: $f_{aux}(v, v, b) = f_r(v_i, v_i, v_j)$ {Note que el vértice v_j adyacente a v_i no es un vértice reducido}
9: $Adicionar(verticesAdyacentes, v_j)$
10: **Fin Si**
11: **Fin Para**
12: **Para todo** $v_o \in verticesAdyacentes$ **hacer**
13: $MDijkstra(v_o, G_{aux})$ {Ver Algoritmo 12}
14: **Para todo** $v_d \in verticesAdyacentes, v_o \neq v_d$ **hacer**
15: $f(v_o, v_i, v_d) = (D[v_d], Camino(v_o, v_d, Pr))$ { D y Pr forman parte del resultado del Algoritmo 12. Para descripción de la función $Camino$ ver epígrafe 2.2.1}
16: **Fin Para**
17: **Fin Para**

Finalmente, se crea un grafo reducido utilizando los conjuntos de vértices, aristas, reglas de reescritura y la función f. El pseudo-código para reducir un grafo se muestra en el Algoritmo 8. El algoritmo de reducción de grafos enunciado en este trabajo tiene particular importancia en el trabajo con mapas, debido a que los mismos siempre se muestran al usuario a una determinada escala. Contar con un grafo reducido con el algoritmo propuesto permite realizar análisis de redes en función de una determinada escala. Por ejemplo, si el grafo que representa la red de viales de Cuba se reduce agrupando todos los vértices que están en una misma provincia, el

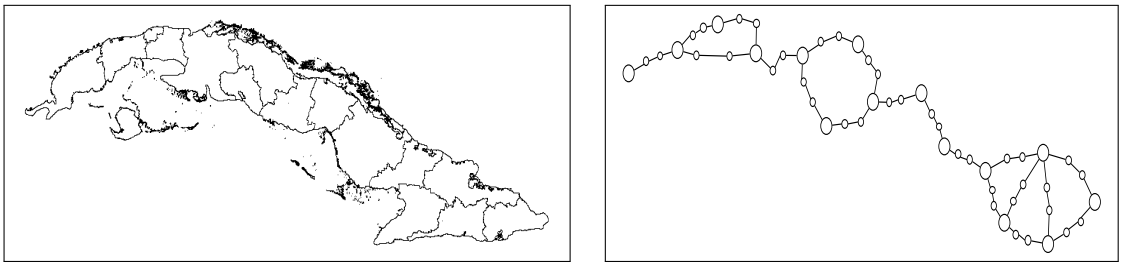
Algoritmo 8 $ReducirGrafo$

Entrada: Un grafo $G = (V, E, f, R)$, donde R es un conjunto de reglas de reescritura, posiblemente vacío y una partición P sobre el conjunto de vértices V

Salida: Un grafo ponderado reducido que representa el mapa

1: $P = RefinarParticion(G, P)$
2: $V_r = ConstruirVerticesReducidos(P)$
3: $E_r = ConstruirAristas(P, G)$
4: $R_r = ConstruirReglasReescritura(P, G)$
5: $f_r = \phi$
6: **Para todo** $A_i \in P, |A_i| > 1$ **hacer**
7: $Calcular f(G, R_r[A_i].G_i, R_r[A_i].G_j, f_r)$ {Calcular el costo de pasar por el vértice reducido correspondiente a la clase A_i de P (ver Algoritmo 7). Note que f_r es un parámetro pasado por dirección.}
8: **Fin Para**
9: Crear el grafo reducido $G_r = (V_r, E_r, f_r, R_r)$
10: **Retornar** G_r

grafo reducido resultante representa el mapa a escala de provincia como se muestra en la Figura 2.5; esto trae como consecuencia simplicidad en el análisis que se realice, ya que a una escala determinada pudieran perder importancia algunos objetos geográficos, debido a que a esa escala los mismos no son visibles o no son de interés para el análisis que se realiza.



(a) Mapa de Cuba

(b) Grafo reducido de las redes de viales de Cuba

Figura 2.5: *Representación de redes de viales en un grafo reducido.*

2.2.2. Análisis de complejidad

En este epígrafe se calcula la complejidad temporal, según el enfoque teórico, de los algoritmos propuestos en la sección anterior.

En un grafo que modela una red de un mapa, un vértice representa la intersección entre dos o más líneas y una arista representa un fragmento de línea que une dos intersecciones. Debido a esto,

generalmente no existirán aristas que se crucen, es por ello que se asume en esta investigación que los grafos que representan redes modeladas a través de un mapa son grafos planares.
Además, en un grafo con estas características el grado de un vértice es, generalmente igual a 4, excepto en algunos casos; por lo que se asume que el grado de un grafo que representa una red de este tipo será menor o igual que 10 ($\Delta^+(G) \leq 10$).

Análisis del algoritmo 3: $ObtenerParticion$

La complejidad temporal, en este caso, está determinada por la cantidad de pares de vértices que se pueden formar a partir del conjunto V y por el grado del grafo; por lo que sería $O(n^2 * \Delta^+(G))$, donde n es la cantidad de elementos del conjunto V. Teniendo en cuenta que $\Delta^+(G) \leq 10$, la complejidad temporal de este algoritmo es $O(n^2)$.

Análisis del algoritmo 4: $ConstruirVerticesReducidos$

Este algoritmo realiza $s = |P|$ iteraciones y en cada iteración se realizan operaciones de complejidad $O(1)$, por lo que la complejidad en este caso es $O(s)$.

Análisis del algoritmo 5: $ConstruirAristas$

Este algoritmo itera sobre cada posible par que se pueda formar entre las clases de P, por lo que el ciclo presente en el mismo itera s^2 veces, donde s es la cantidad de clases de P. Además, por cada iteración se debe comprobar si existe una arista en el grafo original que cumpla con las condiciones expuestas en el algoritmo, acción que requiere $|E|$ iteraciones.

- Pasos 1-7 $O(s^2 * |E|)$.
    - Pasos 3-6 $O(|E|)$. Note que hay que comprobar, en el peor de los casos, todas las aristas.

Teniendo en cuenta la planaridad del grafo y por tanto la relación entre la cantidad de vértices y aristas, la complejidad temporal de este algoritmo es $O(s^2 * n)$, donde n es la cantidad de vértices del grafo original.
En la presente propuesta se recomienda que la cantidad de vértices del grafo reducido sea menor o igual a la raíz cuadrada de la cantidad de vértices del grafo original, para ello se debe crear una partición que tenga no más de $\sqrt{n}$ clases. Sustituyendo s por $\sqrt{n}$, la complejidad sería $O(n^2)$.

Análisis del algoritmo 6: $ConstruirReglasReescritura$

Para el análisis de la complejidad en este caso, se toma en cuenta que $a_i = max[|A_i|, A_i \in P]$ y c_{v_r} es la cantidad de clases de P que tienen más de un elemento (la cantidad de vértices reducidos).

- Pasos 2-25 $O(c_{v_r} * a_i^2 * \Delta^+(G))$.
    - Pasos 4-11 $O(a_i^2 * \Delta^+(G))$.
    - Pasos 14-23 $O(a_i * \Delta^+(G)))$.
        - Pasos 16-18 $O(\Delta^-(G))$.
        - Pasos 20-22 $O(\Delta^+(G))$.

Teniendo en cuenta que $\Delta^+(G) \leq 10$, la complejidad temporal es $O(c_{v_r} * a_i^2)$. Además, suponiendo que se hace una reducción donde la cantidad de vértices reducidos es $\sqrt{n}$ (como se recomienda en este trabajo) y n es la cantidad de vértices del grafo original, el valor máximo que puede tener c_{v_r} es $\frac{\sqrt{n}}{2}$ y el valor máximo de a_i sería $\frac{2(n-\sqrt{n})}{\sqrt{n}}$, ya que los vértices del grafo original deben ser distribuidos entre los vértices reducidos. Siendo la complejidad, en este caso, $O(n^{\frac{3}{2}})$.

Análisis del algoritmo 7: $Calcular f$

Sea $a_i = max[|A_i|, A_i \in P]$ el tamaño de la instancia para este algoritmo:

- Paso 1 $O(a_i)$.
- Pasos 3-11 $O(a_i * \Delta^+(G))$. Note que se itera por cada par de vértices $v_i \in A_{aux}$ y $v_j \in Adyacentes(v, G)$, donde $|A_{aux}| = a_i$ y $|Adyacentes(v, G)| \leq \Delta^+(G)$. Las operaciones que están dentro de la estructura condicional tienen complejidad $O(1)$.
- Pasos 12-17 $O((a_i * \Delta^+(G)) * max[(a_i * \Delta^+(G)) \lg(a_i * \Delta^+(G)), (a_i * \Delta^+(G))])$.
    - Paso 13 $O((a_i * \Delta^+(G)) \lg(a_i * \Delta^+(G)))$.
    - Pasos 14-16 $O((a_i * \Delta^+(G)))$.

Teniendo en cuenta que el grafo es planar y $\Delta^+(G) \leq 10$, la complejidad es, en el peor de los casos, $O(a_i^2 \lg a_i)$.

Análisis del Algoritmo 8: $ReducirGrafo$

- Paso 1 $O(n^2)$.
- Paso 2 $O(s)$.
- Paso 3 $O(s^2 * n)$.
- Paso 4 $O(c_{v_r} * a_i^2)$.
- Pasos 6 - 8 $O(c_{v_r} * a_i^2 \lg a_i)$.
    - Paso 7 $O(a_i^2 \lg a_i)$.

Suponiendo que el grafo reducido tiene $\sqrt{n}$ vértices ($s = \sqrt{n} = |P|, n > 0$) y que $a_i = max[|A_i|, A_i \in P]$, el máximo valor que puede tomar a_i es $n - \sqrt{n}$, por tanto, se cumple lo siguiente:

$a_i^2 \lg a_i \leq (n - \sqrt{n})^2 \lg (n - \sqrt{n})$, desarrollando el miembro derecho de la desigualdad:

$a_i^2 \lg a_i \leq (n^2 - 2n\sqrt{n} + n) \lg (n - \sqrt{n})$, de donde,

$O(a_i^2 \lg a_i) \leq O((n^2 - 2n\sqrt{n} + n) \lg (n - \sqrt{n})) = O(n^2 \lg (n - \sqrt{n}))$

La cantidad de vértices reducidos c_{v_r} (c_{v_r} es la cantidad de clases A_i de P, tal que $|A_i| > 1$), a lo sumo $c_{v_r} = \frac{\sqrt{n}}{2}$ (ver Corolario 3.1.3.1). Luego, la complejidad, en el peor de los casos, es $O(max(n^2, c_{v_r} * a_i^2 \lg a_i)) = O(n^2 \sqrt{n} \lg (n - \sqrt{n}))$.

2.3. Análisis de redes. Búsqueda de caminos óptimos

En este epígrafe se describe cómo hacer la búsqueda de caminos óptimos sobre un grafo reducido según el Algoritmo 8. Para ello se debe tener en cuenta que se puede realizar el análisis según dos enfoques:

Enfoque 1: A escala, es decir, donde no se tengan en cuenta todos los objetos que existen en la capa vectorial utilizada como entrada del algoritmo de reducción. Por ejemplo, ¿cuál es el camino óptimo entre el municipio x y el y? Este tipo de pregunta debe estar acorde con la reducción realizada. Para ello se debe haber reducido el grafo según una partición que agrupe en la misma clase todos los vértices que pertenecen a un mismo municipio.

Enfoque 2: Teniendo en cuenta todos los objetos presentes en la capa del mapa utilizada como entrada del algoritmo de reducción.

La diferencia principal entre estos dos enfoques radica en la forma de obtener el grafo sobre el cual se va a realizar el análisis.

En el primer enfoque se debe utilizar un grafo en el que todos sus vértices estén a un mismo nivel de reducción, es decir, que todos hayan sido reducidos utilizando la misma partición y por tanto están a la misma "escala".

En el segundo enfoque, en el grafo deben existir vértices con distintos niveles de reducción. Esto se debe a que cuando se hace una búsqueda de camino óptimo en grafos grandes, el mecanismo que provee el modelo para lograr eficiencia es tener varios vértices reducidos. Los vértices de origen y de destino no pueden estar reducidos en aras de obtener resultados exactos.

2.3.1. Algoritmo para la búsqueda de caminos óptimos

Para hacer búsqueda de caminos óptimos siguiendo los dos enfoques planteados anteriormente, se utiliza la estrategia que se explica a continuación.

En primer lugar, se modifica el grafo reducido para realizar la búsqueda de caminos óptimos en dependencia del enfoque que se desee seguir. Si el enfoque es el primero, es necesario tener todos los vértices del grafo a un mismo nivel de reducción, según la escala a la que se desee realizar el análisis. Para obtener este grafo se puede recorrer la lista de vértices del grafo y expandir cada vértice reducido utilizando el Algoritmo 9. De esta forma, se obtendría el nivel de reducción anterior del grafo. Si es más de un nivel, bastaría con volver a aplicar lo descrito anteriormente.

Si el enfoque es el segundo, habría que expandir vértices reducidos aplicando el Algoritmo 10, hasta que los vértices de origen (v_o) y destino (v_d) estén en el nuevo grafo.

Se deben realizar dos tipos de expansiones, en uno de ellos se expande un vértice reducido (aplicando la regla de reescritura correspondiente) y en el otro, se deben realizar expansiones de vértices reducidos hasta que aparezca en el grafo un vértice dado.

En el primer caso (Algoritmo 9) se parte de un vértice del grafo reducido y se aplica la regla de reescritura asociada al mismo. Para ello se debe adicionar el grafo G_j al grafo reducido, luego se conectan los vértices de G_j con los vértices del nuevo grafo, según ψ_{in} y ψ_{out} y por último, se elimina el vértice reducido que se está expandiendo.

Algoritmo 9 $ExpandirVerticeReducido$

Entrada: Vértice v_r a expandir y el grafo reducido (G_r) al que pertenece

Salida: Un grafo, posiblemente reducido, resultante de aplicar la regla de reescritura asociada al vértice v_r

1: **Para todo** $e \in v_r.G_j.E_j$ **hacer**
2: $AdicionarArista(G_r, e)$
3: **Fin Para**
4: **Para todo** $(u_1, c_1, c_2, u_2) \in v_r.\psi_{in}$ **hacer**
5: $AdicionarArista(G_r, (u_2, u_1, costo = c_2))$
6: **Fin Para**
7: **Para todo** $(u_1, c_1, c_2, u_2) \in v_r.\psi_{out}$ **hacer**
8: $AdicionarArista(G_r, (u_1, u_2, costo = c_2))$
9: **Fin Para**
10: $EliminarVertice(v_r)$
11: **Retornar** G_r

Algoritmo 10 $ExpandirE$

Entrada: Vértice a expandir (v_e) y grafo reducido G_r

Salida: Grafo reducido $G_r = (V_r, E_r, f, R_r)$, tal que $v_e \in V_r$

1: Inicializar la variable $nivel$ en la cantidad de niveles de reducción del grafo
2: **Repetir**
3: $v = ObtenerVerticeReducido(G_r, v_e)$ {Obtener el vértice reducido que contiene al vértice ve}
4: **Si** $v = null$ **or** v no es reducido **entonces**
5: Terminar
6: **Fin Si**
7: $Expandir(v)$
8: $nivel = nivel - 1$
9: **Hasta** $nivel = 0$ {Si el grafo se redujo n veces, hay que aplicar n expansiones para llegar a un vértice original.}

En el segundo caso (Algoritmo 10) se tiene en cuenta el carácter jerárquico de las particiones. Por ejemplo, si el grafo se redujo por municipio y luego por provincia, para obtener a través de expansiones de vértices reducidos un vértice que se encuentra situado geográficamente en un determinado municipio, se debe expandir en primer lugar el vértice que se corresponde con la provincia y luego el que se corresponde con el municipio.

Para obtener el vértice reducido que contiene a un vértice dado (paso 3 del Algoritmo 10), se realiza un recorrido sobre el conjunto de vértices del grafo reducido y se comprueba con cuál vértice reducido está relacionado el vértice pasado como parámetro. Para esto se hace uso de las particiones utilizadas como entrada del Algoritmo 8 para reducir el grafo. Cuando se encuentra el vértice reducido, se expande aplicando la regla de reescritura asociada a dicho vértice. El pseudo-código correspondiente se muestra en el Algoritmo 11.

Algoritmo 11 $ObtenerVerticeReducido$

Entrada: Vértice a buscar (v) y grafo reducido G_r

Salida: Vértice reducido (v_r) que contiene al vértice v

1: **Para todo** $v_i \in V_r$ **hacer**
2: **Si** (v_i no es reducido **and** $v_i == v$) **or** (v_i es reducido **and** v_i y v_j pertenecen a la misma clase) **entonces**
3: **Retornar** v_i
4: **Fin Si**
5: **Fin Para**
6: **Retornar** $null$

Una vez que se tiene un grafo reducido en el cual los vértices de origen y destino son vértices no reducidos, se aplica el Algoritmo 12 ($MDijkstra$). El mismo está diseñado a partir de una modificación al algoritmo de Dijkstra.

Algoritmo 12 $MDijkstra$

Entrada: Un grafo ponderado reducido $G = (V, E, f, R)$ y un vértice de origen v_{origen}

Salida: Vector D_n de distancias mínimas, vector Pr_n de predecesores

1: $C_n = \{\}, cola = \{\}$ {La variable cola representa una cola con prioridad}
2: **Para todo** $v \in V$ **hacer**
3: $D_n[v] = f(v_{origen}, v_{origen}, v)$, $Pr_n[v] = v_{origen}$
4: $Adicionar(cola, v, D_n[v])$ {Se adiciona el vértice v tomando como prioridad la distancia hasta el mismo.}
5: **Fin Para**
6: **Mientras** $!Vacia(cola)$ **hacer**
7: $w_n = Extraer(cola)$, $C_n = C_n \cup \{w_n\}$
8: **Para todo** $v \in Adyacentes(w_n)$ **hacer**
9: **Si** $D_n[v] > D_n[Pr_n[w_n]] + f(Pr_n[w_n], w_n, v)$ **entonces**
10: $D_n[v] = D_n[Pr_n[w_n]] + f(Pr_n[w_n], w_n, v)$, $Pr_n[v] = w_n$
11: $DecrementarLlave(cola, v, D_n[v])$ {Si $D_n[v]$ es menor que la prioridad de v en la cola se modifica la prioridad.}
12: **Fin Si**
13: **Si** v es reducido **entonces**
14: Actualizar distancia y predecesor de cada vértice adyacente a v utilizando la función f
15: **Fin Si**
16: **Fin Para**
17: **Fin Mientras**

La modificación realizada sigue el siguiente principio: cada vez que se actualiza la distancia desde el pivote (ver paso 7, el vértice w_n se denomina pivote) hasta un vértice adyacente que sea reducido (paso 13), se actualizan las distancias desde el pivote hasta todos los vértices adyacentes a dicho vértice reducido, haciendo uso de la función f. Este algoritmo retorna un vector D con las distancias mínimas desde el vértice de origen al resto de los vértices, así como el vector Pr que contiene los predecesores de cada vértice en el camino óptimo desde el vértice

de origen.

La función f, calculada durante el proceso de reducción, es necesaria ya que para obtener un camino óptimo se requiere conocer el costo de pasar por un vértice reducido. Este valor no queda asociado directamente a las aristas porque, en el caso de los grafos reducidos, el costo de una arista depende del camino en el cual se encuentren los vértices que conforman la misma.

Como se puede apreciar en la Figura 2.6, el efecto que tiene el costo de ir desde el vértice 2 al 13 en el cálculo de un determinado camino, depende si se llega al vértice 2 desde el vértice 7 o desde el 6. Cada vértice reducido representa un subgrafo del grafo original por lo que el costo de recorrer dicho subgrafo dependerá del vértice desde el cual se llega al mismo.

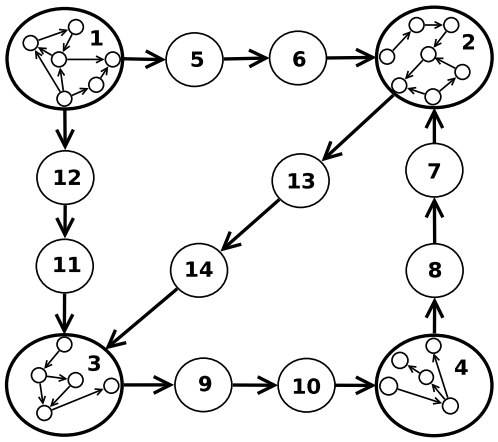


Figura 2.6: *Ejemplo de grafo reducido.*

El algoritmo $MDijkstra$ es uno de los principales aportes del presente trabajo ya que garantiza la obtención de un camino óptimo en el grafo reducido, de igual costo que el camino óptimo encontrado por el algoritmo de Dijkstra en el grafo original (sin reducir).

Para obtener el camino óptimo a partir del vector Pr de predecesores se calcula el antecesor del vértice v_d ($v_{aux} = Pr[v_d]$) en el camino desde v_o. Luego, el antecesor del vértice encontrado (v_{aux}) se calcula de la misma forma: $v_{aux} = Pr[v_{aux}]$. Este proceso se repite hasta que v_{aux} sea igual a v_o. El camino que se retorna es la concatenación de todos los vértices encontrados, en orden inverso.

El pseudo-código para la búsqueda de caminos óptimos en grafos reducidos se muestra en el Algoritmo 13.

A partir del Algoritmo 13 se puede afirmar que se brinda la posibilidad de realizar una búsqueda

Algoritmo 13 $BuscarCaminoMinimo$

Entrada: Grafo reducido G_r, vértice de origen v_o, vértice de destino v_d, $enfoque$ y $nivel = 0$

Salida: Camino óptimo desde v_o hasta v_d

1: **Si** $enfoque = 1$ **entonces**
2: $ObtenerGrafoNivel(G_r, nivel)$
3: **Sino**
4: $ExpandirE(G_r, v_o)$
5: $ExpandirE(G_r, v_d)$
6: **Fin Si**
7: $MDijkstra(G_r, v_o, v_d)$ {Ver Algoritmo 12}
8: **Retornar** $Camino(v_o, v_d, Pr)$

de camino óptimo entre cada par de vértices del grafo original aplicando las reglas de reescrituras apropiadas.

En un SIG, los vértices de origen y destino para una petición de búsqueda de camino óptimo se seleccionan generalmente haciendo uso del mapa, es decir, un usuario selecciona estos puntos haciendo clic en el mapa que muestra el sistema.

En este sentido, se asume que cuando un usuario selecciona un vértice de origen o destino, el SIG realiza una expansión de un vértice reducido teniendo en cuenta la porción del mapa que se está mostrando y el punto seleccionado.

Si un sistema para realizar búsquedas de caminos óptimos es implementado de esta forma, el tiempo que se requiere para expandir un vértice reducido sería irrelevante para la búsqueda de caminos óptimos, considerando que la expansión de un vértice reducido tiene complejidad lineal $O(a_i)$, donde $a_i = max\{|Ai|, A_i \in P\}$.

2.3.2. Análisis de complejidad

A continuación se muestra la complejidad de los algoritmos propuestos en la sección anterior. Para ello se asume que $G_r = (V_r, E_r, f_r, R_r)$ es el grafo reducido sobre el que se ejecutan los algoritmos, G_j es el subgrafo de la parte izquierda de una regla de reescritura asociada a un vértice reducido y $a_i = max[|A_i|, A_i \in P] = max[|v.G_j|, v \in V_r]$.

Análisis del algoritmo 9: $ExpandirVerticeReducido$

- Pasos 1-3 $O(|v_r.G_j.E_j|)$.
- Pasos 4-9 $O(a_i * \Delta^+(G_r))$.

Teniendo en cuenta el criterio de planaridad del grafo G_j y el grado máximo del mismo, la complejidad de este algoritmo es $O(a_i)$.

Análisis del algoritmo 10: $ExpandirE$

- Pasos 2-9 $O(n_e) = O(n_{cr}) * max(O(n_{vr}), O(a_i))$, donde n_{cr} es la cantidad de reducciones realizadas para obtener el vértice reducido v_r que contiene el vértice v_e que recibe el algoritmo como parámetro

    - Paso 3 $O(n_{vr})$, es la complejidad asociada a invocar el Algoritmo 11: $ObtenerVerticeReducido$
    - Pasos 4-6 $O(1)$
    - Paso 7, $O(a_i)$ es la complejidad asociada a invocar el Algoritmo 9: $ExpandirVerticeReducido$

Análisis del algoritmo 11: $ObtenerVerticeReducido$

- Pasos 1-6 $O(n_{vr})$, siendo $n_{vr} = |V_r|$ la cantidad de vértices del grafo reducido.

    - Pasos 2-4 Estructura condicional donde todas las operaciones tienen complejidad lineal $O(1)$.

La complejidad temporal de este algoritmo es $O(n_{vr})$.
Como se puede apreciar, la complejidad de los algoritmos 11 y 9 es lineal cualquiera sea el tamaño de la instancia. Por otra parte, en la presente investigación se recomienda realizar hasta 10 reducciones, de esta forma, la cantidad de iteraciones del ciclo presente en el algoritmo sería $n_{cr} = 10$, por lo que el tiempo de ejecución del Algoritmo 10 sería lineal y se puede acotar con $O(n_e) = max(O(n_{vr}), O(a_i))$.
En la medida que se incrementa la cantidad de vértices del grafo original, la cantidad de vértices del grafo G_j asociado a un vértice reducido se incrementa para poder lograr una reducción de $\sqrt{n}$ vértices (n es la cantidad de vértices del grafo original); como se recomienda en la presente investigación. Si el grafo es suficientemente grande, la cantidad de vértices de G_j será mayor que la cantidad de vértices reducidos, pudiéndose definir la complejidad temporal, en este caso, como $O(a_i)$.

Análisis del algoritmo 12: $MDijkstra$

La modificación introducida consta de una estructura condicional que contiene la iteración sobre una lista de vértices adyacentes y una llamada a la función f. Haciendo uso del criterio de planaridad del grafo así como del grado del mismo, se puede afirmar que la complejidad de recorrer la lista de vértices adyacentes es $O(1)$. Por otra parte, la función f se calcula durante el proceso de reducción, debido a esto, una llamada a la misma tiene complejidad lineal $O(1)$.
De esta forma, la complejidad temporal del algoritmo modificado es del mismo orden que el algoritmo original $O(n \lg(n))$. Sin embargo, note que el tamaño de la instancia, en este caso la cantidad de vértices del grafo reducido, puede ser sustancialmente menor que el tamaño de la instancia en el caso del algoritmo de Dijkstra que sería la cantidad de vértices del grafo original. Siendo más eficiente la búsqueda de caminos óptimos en grafos reducidos.

Análisis del algoritmo 13: $BuscarCaminoMinimo$

- Pasos 1-6 $O(n_{vr} * a_i)$, n_{vr} es la cantidad de vértices del grafo reducido.
  - Paso 2 $O(n_{vr} * a_i)$
  - Paso 4 $O(a_i)$, ver análisis de complejidad del Algoritmo 10: ExpandirE
- Paso 7 $O(n_{vr} * \lg n_{vr})$
- Paso 8 $O(n_{vr})$

Para realizar el paso 2 bastaría con recorrer todos los vértices del grafo reducido y expandirlos (utilizando el Algoritmo 9), esto se realizaría la cantidad de veces especificadas en el parámetro nivel que como se mencionó anteriormente se recomienda que su valor máximo sea 10, por lo que la complejidad de este paso es $O(n_{vr} * a_i)$.
La complejidad de este algoritmo estaría dada por $O(max(n_{vr} * a_i, n_{vr} * \lg n_{vr}))$. En un grafo suficientemente grande, $a_i > n_{vr}$, de donde $a_i > \lg n_{vr}$; luego, la complejidad sería $O(n_{vr} * a_i)$.
La complejidad temporal del algoritmo A*, utilizando una heurística óptima, es $O(n)$, donde n es el número de vértices del grafo sobre el que se realiza el análisis. Por otra parte, la complejidad temporal del Algoritmo 12 ($MDijkstra$) es $O(n_1 \lg n_1)$, menor que $O(n_1^2)$, donde n_1 es la cantidad de vértices del grafo reducido. Por tanto, si en el proceso de reducción se obtiene un grafo $G_r = (V_r, E_r)$ a partir de un grafo $G = (V, E)$, tal que $n_1 = |V_r|, n = |V|, n_1 \leq$

$\sqrt{n}$, la complejidad temporal del Algoritmo 12 ($MDijkstra$) sería, en teoría, menor que la del Algoritmo A*.

Generalmente existe una relación inversa entre eficiencia y exactitud en los algoritmos que tienen como entrada un volumen elevado de datos. El principal resultado que se muestra es una mejora de eficiencia en la búsqueda de caminos óptimos sin afectar la exactitud de la respuesta.

La complejidad espacial de un algoritmo está estrechamente relacionada con las estructuras de datos que se utilicen en el mismo. Debido al nivel de abstracción del modelo propuesto, no se especifican como parte del mismo detalles de implementación; por lo que no se realiza el cálculo de la complejidad espacial de los algoritmos propuestos.

2.4. Indicaciones metodológicas para la aplicación del modelo en Sistemas de Información Geoespacial

Para la aplicación del modelo propuesto en los SIG se enuncian las siguientes indicaciones metodológicas:

Niveles de reducción: Los niveles de reducción no deben ser más de 10, debido a que la cantidad de reducciones influye en el tiempo de respuesta de la búsqueda de camino óptimo. Además, se considera que con 10 reducciones se puede garantizar eficiencia en el análisis que se desee realizar por grande que sea el grafo. Por otra parte, realizando una sola reducción de forma tal que $n_1 \leq \sqrt{n}$, donde n_1 es la cantidad de vértices del grafo reducido y n la cantidad de vértices del grafo original, el Algoritmo 12 (MDijkstra) da una respuesta eficiente y tiene el mismo orden que el algoritmo A*, ya que el algoritmo A* puede, en el mejor de los casos, ejecutarse en tiempo lineal $O(n)$.

Particiones: Las particiones que se utilicen deben ser escogidas apropiadamente. En las mismas, la diferencia entre la cantidad de elementos de cada clase no debe ser demasiado grande, esto facilitará los análisis posteriores. Además, cuando se definan varias particiones para realizar más de una reducción, se debe tener en cuenta el carácter jerárquico de las mismas, debido a que en un mapa todo depende de las coordenadas geográficas (latitud, longitud), es por ello que las particiones deben agrupar de un área

menor a una mayor.

Escalas del mapa: Una vez definidas las escalas a las que el mapa se presentará al usuario, se puede asociar a cada escala una partición. De esta forma el grafo reducido representará una escala del mapa por cada nivel de reducción, esto facilita el análisis a escala sobre el mapa.

2.5. Conclusiones del capítulo

En el presente capítulo se ha descrito un modelo para la representación de redes en SIG, como parte del mismo se han enunciado varias definiciones y se ha realizado el diseño y análisis de varios algoritmos. Se propone además el uso de grafos reducidos para realizar análisis de redes y se parte de las redes representadas en un mapa. En el modelo tiene particular importancia el algoritmo de reducción planteado.
Finalmente se concluye lo siguiente:

- El uso de las gramáticas de grafo y la reescritura de grafos permite reducir un grafo sin que exista pérdida de información.
- El uso de grafos reducidos para realizar búsqueda de caminos óptimos permite dar respuestas de forma eficiente cuando las redes son grandes. Además facilita la realización de análisis de redes a diferentes escalas.
- El modelo presentado hace uso de un algoritmo de reducción de grafos sin pérdida de información para garantizar la reducción del tiempo de ejecución de la búsqueda de caminos óptimos. Este enfoque mantiene la exactitud debido a que el algoritmo de reducción garantiza que no exista pérdida de información. Además, se verificó que la complejidad temporal, utilizando el enfoque teórico, es la misma que los algoritmos estudiados.
- El modelo propuesto tiene un elevado nivel de generalidad, ello se evidencia en su posible uso en distintos tipos de redes presentes en un mapa.

Capítulo 3

Validación de la propuesta

3. VALIDACIÓN DE LA PROPUESTA

EN 1969, C.A.R. Hoare expresó "Programación es una ciencia exacta en la que todas las propiedades de un programa y todas las consecuencias de ejecutarlo en un ambiente dado, pueden ser determinadas, en principio, a partir del texto mismo del programa a través de razonamiento inductivo puro"[167]. En este artículo Hoare expuso lo que posteriormente se conoció como tripleta de Hoare: $P\{Q\}R$, lo cual se interpreta de la siguiente forma: si la precondición P se cumple luego de la inicialización del programa Q entonces la postcondición R se cumplirá cuando termine la ejecución de Q. Si no existen precondiciones, la tripleta se escribe como $true\{Q\}R$.

La tripleta de Hoare está estrechamente relacionada con la demostración de corrección de algoritmos. La demostración se basa en definir para un algoritmo (Q) las precondiciones (P) y postcondiciones (R) y demostrar que la tripleta de Hoare es válida.

La mayoría de los algoritmos involucran algún tipo de ciclo, cuando se realiza el análisis de este tipo de algoritmo, es imprescindible determinar cuáles son las invariantes de ciclo.

Basado en lo anterior, la demostración de corrección de un algoritmo con ciclos se realiza demostrando que a partir de las precondiciones se cumplen las invariantes de ciclo. Por último, se verifica que también se cumplen las postcondiciones haciendo uso del resultado anterior.

A continuación se realiza la demostración de corrección de los algoritmos de reducción de grafos y de búsqueda de caminos óptimos en grafos reducidos propuestos en el presente trabajo.

3.1. Demostración de corrección del Algoritmo 8: $ReducirGrafo$

El algoritmo de reducción enunciado utiliza varios algoritmos para garantizar la reducción de un grafo, por lo que para demostrar la corrección del mismo es necesario demostrar, en primer lugar, la corrección de los algoritmos que son invocados.

3.1.1. Demostración de corrección del Algoritmo 3. $ObtenerParticion$

Precondiciones:

- Sea RE una relación de equivalencia sobre un conjunto V, sin perder generalidad se asume, por simplicidad para la demostración, que $V = \{0, 1, 2, ..., N-1\}$.
- $\forall n < N$, se define:
  - Un conjunto Int_{v_n} que representa los vértices interiores de la clase de equivalencia $[v]$, tanto como se conozca en el paso n.
  - Un conjunto Pa_n que representa una partición sobre V tanto como se conozca en el paso n.

Invariantes de ciclo:

- $\forall n < N$ (invariantes del ciclo interior del algoritmo):
  1. $\forall n < N$, $\exists v \in V$, tal que $\forall v_i \in V, [(\text{Si } (v, v_i) \in RE \wedge v_i \text{ es un vértice interior}) \rightarrow v_i \in Int_{v_n}]$. Si el vértice v_i pertenece a la clase de equivalencia del vértice v y además es interior, entonces v_i pertenece al conjunto de los vértices interiores de v.
  2. $\forall n < N$, $\exists v \in V$, tal que $\forall v_i \in V, [(\text{Si } (v, v_i) \in RE \wedge v_i \text{ es un vértice exterior}) \rightarrow \{v_i\} \in Pa_n]$. Si el vértice v_i pertenece a la clase de equivalencia del vértice v y es exterior, entonces $\{v_i\}$ es una clase de la partición Pa.
- $\forall n < N$ (invariante del ciclo exterior del algoritmo):

3. $\forall v \in V, Int_{v_n} \neq \phi \rightarrow Int_{v_n} \in Pa_n$. El conjunto de los vértices interiores asociados a un vértice v forma una clase de equivalencia en Pa. Note que en el paso n del ciclo exterior ya se calculó completamente el conjunto Int_v.
4. $\forall n < N, \cap_{i=1}^{|Pa_n|} A_i = \phi, A_i \in Pa_n$. La intersección de los elementos de Pa es igual a ϕ.

Además de las invariantes mencionadas anteriormente, se debe cumplir lo siguiente al finalizar la ejecución del algoritmo:

- $V = \cup_{i=1}^{|Pa|} A_i, A_i \in Pa$. La unión de los elementos de Pa es igual al conjunto V.

Dada la sencillez de un algoritmo para determinar cuándo un vértice es interior o exterior (ver Definición 2.2.1.1) se asume que el mismo existe, es correcto y que se cumple el siguiente teorema:
Sean:

- $G = (V, E)$ un grafo.
- RE una relación de equivalencia que define una partición $P = V/RE = (A_1, A_2, \ldots, A_s)$ sobre el conjunto V.
- $\exists v_i \in A_i, A_i \in P$.
- $\exists v_j \in A_j, A_j \in P$.

TEOREMA 3.1.1 $((v_i, v_j) \in E) \rightarrow [((v_i, v_j$ *son vértices interiores)*$\wedge (v_i, v_j) \in RE) \vee ((v_i$ *es interior*$, v_j$ *es exterior*$) \wedge (v_i, v_j) \in RE) \vee ((v_i, v_j$ *son vértices exteriores*$) \wedge (v_i, v_j) \notin RE)]$

LEMA 3.1.1 $\forall n < N$, $\exists v \in V$, *tal que* $\forall v_i \in V, [($*Si* $(v, v_i) \in RE \wedge v_i$ *es un vértice interior*$) \rightarrow v_i \in Int_{v_n}]$.

DEMOSTRACIÓN: (Por inducción sobre n)
Caso base $n = 0$, $Int_{v_0} = \{\}$
Para $n = k + 1$:
Caso 1: v_i es interior $\wedge (v, v_i) \in RE$
$Int_{v_{k+1}} = Int_{v_k} \cup \{v_i\} \therefore v_i \in Int_{v_{k+1}}$

Caso 2: v_i es exterior $\wedge(v, v_i) \in RE$

$Pa_{k+1} = Pa_k \cup \{v_i\}$,

$Int_{v_{k+1}} = Int_{v_k}$ por lo que $v_i \notin Int_{v_{k+1}}$

Caso 3: $\neg(v, v_i) \in RE$:

No se hace nada, por lo que $v_i \notin Int_{v_{k+1}}$ □

LEMA 3.1.2 $\forall n < N$, $\exists v \in V$, *tal que* $\forall v_i \in V, [($*Si* $(v, v_i) \in RE \wedge v_i$ *es un vértice exterior*$) \rightarrow \{v_i\} \in Pa_n]$.

La demostración de este lema es similar a la del lema anterior.

LEMA 3.1.3 $\forall n < N, \forall v \in V, (Int_v \neq \phi \rightarrow Int_{v_n} \in Pa_n)$.

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$, $Pa_0 = \{\}$

Para $n = k + 1$

Caso 1: Si $Int_v \neq \phi$

$Pa_{k+1} = Pa_k \cup Int_v, \therefore Int_v \in Pa_{k+1}$

Caso 2: Si $Int_v = \phi$

$Pa_{k+1} = Pa_k$ □

LEMA 3.1.4 $\forall n < N, \cap_{i=1}^{|Pa_n|} A_i = \phi, A_i \in Pa_n$

DEMOSTRACIÓN:

Para demostrar este lema bastaría con demostrar que si $v \in A_i$ y $v \in A_j$ tal que $A_i, A_j \in Pa$, entonces $A_i = A_j$. Note que Pa se define en el algoritmo como un conjunto, por lo que no puede tener elementos repetidos.

La cardinalidad de las clases de Pa puede ser uno o mayor que uno. En el primer caso la demostración es trivial.

Segundo caso:

Si $v \in A_i$ se cumple que todos los vértices interiores relacionados con v por la relación de equivalencia RE también pertenecen a A_i

Si $v \in A_j$ se cumple que todos los vértices interiores relacionados con v por la relación de equivalencia RE también pertenecen a A_j

$\therefore A_i = A_j$ □

LEMA 3.1.5 *Después de la ejecución del Algoritmo 3 se cumple que* $V = \cup_{i=1}^{|Pa|} A_i, A_i \in Pa$.

DEMOSTRACIÓN:

Note que a partir del Algoritmo 3 se deduce lo siguiente: $\exists v \notin V \rightarrow v \notin A_i, i = 1..s$.

Sea $v_i \in V$, se cumple que $(v_i, v_i) \in RE$, por ser RE una relación de equivalencia. A partir de esto se pueden tener dos casos:

Caso 1: $v_i \in Int_{v_i}$

$Int_{v_i} \in Pa$, por el Lema 3.1.3

Caso 2: $v_i \notin Int_{v_i}$

$v_i \in Pa$, por el Lema 3.1.2 □

TEOREMA 3.1.2 *Después de la ejecución del Algoritmo 3 se cumple que* Pa *es una partición sobre* V.

La demostración de este teorema es directa a partir de los lemas 3.1.4 y 3.1.5.

3.1.2. Demostración de corrección del Algoritmo 4: $ConstruirVerticesReducidos$

Precondiciones:

- Sea $Pa = \{A_1, A_2, \ldots, A_{N-1}\}$ una partición.
- Sea V_{r_n} el conjunto de vértices reducidos, tanto como se conozca en el paso n.

Invariante de ciclo:

1. $\forall n < N$, Sea v_{r_n} el vértice representativo de la clase de equivalencia A_n, $A_n \in Pa$, $v_{r_n} \in V_{r_n}$.

Debido a la simplicidad del algoritmo se asume que el mismo es correcto y por tanto que se cumplen los siguientes lemas:

LEMA 3.1.6 $\forall n < N$, *Sea* v_{r_n} *el vértice representativo de la clase de equivalencia* A_n, $A_n \in Pa$, $v_{r_n} \in V_{r_n}$.

LEMA 3.1.7 $|V_r| = |Pa| = N - 1$, *o sea, el conjunto de vértices reducidos* V_r *tiene la misma cantidad de elementos que la partición* Pa.

3.1.3. Demostración de corrección del Algoritmo 5: $ConstruirAristas$

Precondiciones:

- Sea $G = (V, E, f, R)$ un grafo.
- Sea $G_r = (V_r, E_r, f_r, R_r)$ un grafo, inicialmente vacío.
- Sea $Pa = \{A_1, A_2, \ldots, A_{N-1}\}$ una partición sobre V.

Invariante de ciclo:

1. $\forall v_i \in A_i, A_i \in Pa, \forall n < N$, si $\exists v_j \in A_n, i \neq n$, tal que $(v_i, v_j) \in E$, entonces $(v_k, v_t) \in E_{r_n}$, donde v_k y v_t son los vértices representativos de las clases de equivalencia A_i y A_n respectivamente. Entre cada par de vértices v_k, v_t del grafo reducido representativos de las clases A_i y A_j respectivamente, siendo $A_i \neq A_j$; se adicionan todas las aristas existentes entre vértices que pertenezcan a A_i y A_j.

LEMA 3.1.8 $\forall v_i \in A_i, A_i \in Pa, \forall n < N$, *si* $\exists v_j \in A_n, i \neq n$, *tal que* $(v_i, v_j) \in E$, *entonces* $(v_k, v_t) \in E_{r_n}$, *donde* v_k *y* v_t *son los vértices representativos de las clases de equivalencia* A_i *y* A_n *respectivamente.*

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$, $E_{r_0} = \{\}$

Para $n = k + 1$:

Caso 1: $\exists (v_i, v_j) \in E, v_i \in A_i, v_j \in A_n, i \neq n$:

$E_{r_{k+1}} = E_{r_k} \cup \{(v_k, v_t)\}$, siendo v_k y v_t los vértices representativos de las clases A_i y A_n respectivamente; $\therefore (v_i, v_j) \in E_r$

Caso 2: $\nexists (v_i, v_j) \in E, v_i \in A_i, v_j \in A_n, i \neq n$:

$E_{r_{k+1}} = E_{r_k}$ □

TEOREMA 3.1.3 *Un vértice reducido (que representa a un conjunto de vértices interiores) solo es adyacente a vértices no reducidos (exteriores) que pertenecen a su misma clase de equivalencia.*

DEMOSTRACIÓN:

$\forall v_i \in A_i, A_i \in Pa$, $\forall n < N$, si $\exists v_j \in A_n, i \neq n$, tal que $(v_i, v_j) \in E$, entonces $(v_k, v_t) \in E_{r_n}$, donde v_k y v_t son los vértices representativos de las clases de equivalencia A_i y A_n respectivamente, por el Lema 3.1.8

$(\exists (v_i, v_j) \in E) \rightarrow [((v_i, v_j$ son vértices interiores$) \wedge (v_i, v_j) \in RE) \vee ((v_i$ es interior$, v_j$ es exterior$) \wedge (v_i, v_j) \in RE)]$, por el Teorema 3.1.1

Teniendo en cuenta que $A_i \neq A_j$ y $v_i \neq v_j$ se puede afirmar que v_i y v_j no son vértices interiores relacionados por RE, por lo que se cumple que uno es interior y el otro exterior; además se cumple que $(v_i, v_j) \in RE$. □

COROLARIO 3.1.3.1 *En cada camino $Ca = (v_1, v_2, \ldots, v_n)$ existente en un grafo reducido $G = (V, E, f, R)$, entre dos vértices reducidos existen al menos dos que no están reducidos.*

3.1.4. Demostración de corrección del Algoritmo 6: $ConstruirReglasReescritura$

Precondiciones:

- $G = (V, E, f, \{\})$, un grafo ponderado reducido.
- $G_r = (V_r, E_r, f_r, R)$, un grafo reducido a partir de G.
- $G_i = (\{v_i\}, \{\})$, grafo de la parte izquierda de la regla de reescritura R_i asociada a la clase de equivalencia $A_i, |A_i| > 1$.
- $G_j = (V_j, E_j, f_j, R_j)$, grafo de la parte derecha de la regla de reescritura R_i asociada a la clase de equivalencia $A_i, |A_i| > 1$.

Invariantes de ciclo:

1. Sea $v_k \in V_j, \forall n > N, \forall v_n \in V_j, k \neq n, [\exists(v_k, v_n) \in E \rightarrow [(v_k, v_n) \in E_{j_n} \wedge f_{j_n}(v_k, v_k, v_n) = f(v_k, v_k, v_n)]]$. Esta invariante garantiza la creación de las aristas del grafo G_j.
2. Sea $v_m, v_n \in V_j$, si v_n es un vértice reducido, $\forall v_k \in V_j, k \neq m, k \neq n, n \neq m, f_{j_k}(v_m, v_n, v_k) = f(v_m, v_n, v_k)$. El grafo G_j es un grafo reducido, por lo que para cada vértice reducido v_n se actualiza el valor de la función $f(v_m, v_n, v_k), \forall v_m, v_k \in V_j$.
3. Sea $v_j \in V_j$, $\forall v_k \in ([v_j] - V_j)$ se cumple que $(\exists(v_k, v_j) \in E) \rightarrow ((v_k, f_c(v_k, v_j), f_c(v_k, v_j), v_j) \in \psi_{in})$. Para todos los vértices interiores (V_j), se debe crear información de empotrado que contenga las aristas que inciden en los mismos desde los vértices exteriores que pertenecen a la misma clase de equivalencia ($[v_j] - V_j$).
4. Sea $v_j \in V_j$, $\forall v_k \in ([v_j] - V_j)$ se cumple que $(\exists(v_j, v_k) \in E) \rightarrow ((v_k, f_c(v_k, v_j), f_c(v_k, v_j), v_j) \in \psi_{out})$. Para todos los vértices interiores (V_j), se debe crear información de empotrado que contenga las aristas que inciden en los vértices exteriores de la misma clase de equivalencia ($[v_j] - V_j$) y salen de los vértices del grafo G_j.

LEMA 3.1.9 *Sea* $v_m \in V_j, \forall n > N, \forall v_n \in V_j, m \neq n, [\exists(v_m, v_n) \in E \rightarrow [(v_m, v_n) \in E_{j_n} \wedge f_{j_n}(v_m, v_m, v_n) = f(v_m, v_m, v_n)]]$.

DEMOSTRACIÓN: (Por inducción sobre n)

Para $n = 0, E_0 = \{\}$

Para $n = k + 1$:

Caso 1: $\exists(v_m, v_{k+1}) \in E$

$E_{k+1} = E_k \cup \{(v_m, v_{k+1})\}$, $f_{j_n}(v_m, v_m, v_n) = f(v_m, v_m, v_n)$.

Caso 2:$\nexists(v_m, v_{k+1}) \in E$

$E_{k+1} = E_k$ □

LEMA 3.1.10 *Sea* $v_m, v_n \in V_j$*, si* v_n *es un vértice reducido,* $\forall v_k \in V_j, k \neq m, k \neq n, n \neq m, f_{j_k}(v_m, v_n, v_k) = f(v_m, v_n, v_k)$.

La demostración de este lema es trivial, está referida a las líneas 7-9 del Algoritmo 6.

LEMA 3.1.11 *Sea* $v_j \in V_j$*,* $\forall v_k \in \{[v_j] - V_j\}$*, tal que* $(\exists(v_k, v_j) \in E)$*, entonces* $((v_k, f(v_k, v_k, v_j), f(v_k, v_k, v_j), v_j) \in \psi_{in_k})$.

DEMOSTRACIÓN: (Por inducción sobre k)

Caso base $k = 0$, $\psi_{in_0} = \{\}$

Para $k + 1$:

$\psi_{in_{k+1}} = \psi_{in_k} \cup \{(v_{k+1}, f(v_{k+1}, v_j, v_j), f(v_{k+1}, v_j, v_j), v_j)\}$ □

LEMA 3.1.12 *Sea* $v_j \in V_j$, $\forall v_k \in \{[v_j] - V_j)\}$, *tal que* $(\exists(v_j, v_k) \in E)$, *entonces* $((v_k, f_c(v_k, v_j), f_c(v_k, v_j), v_j) \in \psi_{out})$.

La demostración de este lema es similar a la del Lema 3.1.11.

En este paso, se procede a verificar que se cumplen las postcondiciones del Algoritmo 8.

TEOREMA 3.1.4 *Sea* $G_r = (V_r, E_r, f_r, R_r)$ *un grafo reducido a partir del grafo* $G = (V, E, f, R)$ *y la relación de equivalencia* RE, *si se aplica la regla de reescritura* $R_{r_i} = ((\{v_i\}, \phi), (V_j, E_j, f_j, R_j), \psi_{in}, \psi_{out})$, *asociada al vértice* $v \in V_r$ *sobre el grafo* G_r, *se obtiene un grafo* $G'_r = (V'_r, E'_r, f'_r, R'_r)$ *y se cumple lo siguiente:*
Sea $v_i \in V_r$, $\forall v_k \in V_j$, *tal que* $(v_i, v_k) \in RE$, $[(v_i, v_k) \in E \rightarrow (v_i, v_k) \in E'_r]$.

DEMOSTRACIÓN: Sea $(v_i, v_k) \in E$, tal que $v_i, v_k \in [v]$, se tratará de demostrar que $(v_i, v_k) \notin E'_r$ luego de aplicar la regla R_{r_i} asociada al vértice reducido v. Note que si existe una arista entre un vértice de un grafo G_j y uno del grafo G_r, dichos vértices pertenecen a la misma clase de equivalencia.

Cuando se aplica una regla R_{r_i} se sustituye el grafo $G_i = (\{v_i\}, \{\})$ por el grafo G_j y se conecta G_j con $(G - G_i)$ según se especifica en ψ_{in} y ψ_{out}.

Sea $v_j \in V_j$, $\forall v_k \in ([v_j] - V_j)$ (note que $[v_k \in ([v_j] - V_j)] \rightarrow v_k \in (V_r - \{v_i\})$):

- si $(\exists(v_k, v_j) \in E)$, entonces $(v_j, f(v_k, v_k, v_j), f(v_k, v_k, v_j), v_k) \in \psi_{in}$, por el Lema 3.1.11.
- si $(\exists(v_j, v_k) \in E)$, entonces $(v_j, f(v_j, v_j, v_k), f(v_j, v_j, v_k), v_k) \in \psi_{out}$, por el Lema 3.1.12.

lo cual es una contradicción. □

COROLARIO 3.1.4.1 *Sea* $G_r = (V_r, E_r, f_r, R_r)$ *un grafo reducido a partir del grafo* $G = (V, E, f, R)$ *y la relación de equivalencia* RE, *si se aplica el conjunto de reglas de reescritura* R_r, *asociadas a los vértices reducidos del grafo* G_r, *se obtiene el grafo* $G = (V, E, f, R)$ *a partir del cual se redujo* G_r.

3.2. Demostración de corrección del Algoritmo 12: $MDijkstra$

Las precondiciones que se deben cumplir para demostrar la corrección del algoritmo se expresan a través de las siguientes definiciones y notaciones:

- $G = (V, E, f_c)$ un grafo ponderado. Sin perder generalidad se asume, por simplicidad para la demostración, que $V = \{0, 1, ..., M-1\}$.
- $G_r = (V_r, E_r, f, R)$ un grafo reducido a partir de G y la relación de equivalencia RE. Sin perder generalidad se asume, por simplicidad para la demostración, que $V_r = \{0, 1, ..., N-1\}$.
- En cada camino existente en un grafo reducido, entre dos vértices reducidos existen al menos dos que no están reducidos (por el Teorema 3.1.3.1).
- $\forall n < N$, en la ejecución del Algoritmo 12 ($MDijkstra$) se define:
  - w_n es el vértice seleccionado en la iteración n.
  - Un conjunto $C_n \subseteq V_r$ tal que a C_n pertenecen los nodos visitados hasta la iteración n. $C_0 = \{v_o\}$, $C_{n+1} = C_n \bigcup \{w_n\}$.
  - Una función $D_n : V_r \to \mathbb{R}^+ \bigcup \{0, \infty\}$. $D_0(v_o) = 0$, $D_{n+1}(v) = Min(D_n(w_n) + f_c(w_n, v), D_n(v)) = Min(D_n(P_n(w_n)) + f(P_n(w_n), w_n, v), D_n(v))$ que representa la distancia mínima desde v_o hasta cada vértice $v \in V_r$ tanto como se conozca en la iteración n.
  - Una función $P_n : V_r \to V_r$, la cual devuelve para cada vértice, el predecesor en el camino óptimo desde v_o hasta v_d, tanto como se conozca en la iteración n. $P_0(v_o) = v_o$, $P_{n+1}(v) = \begin{cases} w_n & \text{si } D_n(v) > D_n(w_n) + f_c(w_n, v) \\ P_n(v) & \text{en otro caso} \end{cases}$
- $\forall m < M$, en la ejecución del algoritmo de Dijkstra (ver Anexo C) se define:
  - wd_m es el vértice seleccionado en la iteración m.
  - Un conjunto $Cd_m \subseteq V$ tal que a Cd_m pertenecen los nodos visitados hasta la iteración m. $Cd_0 = \{v_o\}$, $Cd_{m+1} = Cd_m \bigcup \{wd_m\}$.

- Una función $d_m : V \rightarrow \mathbb{R}^+ \bigcup \{0, \infty\}$. $d_0(v_o) = 0$, $d_{m+1}(v) = Min(d_m(wd_m) + f_c(wd_m, v), d_m(v))$ que representa la distancia mínima desde v_o hasta cualquier vértice $v \in V$ tanto como se conozca en la iteración m.
- Una función $Pd_m : V \rightarrow V$, la cual devuelve para cada vértice, el predecesor en el camino óptimo desde v_o hasta v_d, tanto como se conozca en la iteración m. $Pd_0(v_o) = v_o$, $Pd_{m+1}(v) = \begin{cases} wc_m & \text{si } Dc_m(v) > Dc_m(w_m) + f_c(w_m, v) \\ Pd_m(v) & \text{en otro caso} \end{cases}$

Para realizar la demostración de corrección es necesario demostrar que se cumplen las siguientes invariantes de ciclo presentes en el algoritmo:

$\forall n < N$ se cumple:

1. $w_n \notin C_n \wedge \forall v \in V(D_n(v) < D_n(w_n) \rightarrow v \in C_n)$. El vértice w seleccionado en la iteración n no ha sido visitado, además si la distancia hasta un vértice $v \in V$ es menor que la distancia hasta w, entonces el vértice v ya se visitó.
2. $|C_n| = n + 1$. En la iteración n se han visitado $n + 1$ vértices.
3. $D_n(v_o) = 0 \wedge P_n(v_o) = v_o$. La distancia desde el vértice de origen hasta él mismo es 0 en cualquier iteración y el predecesor de dicho vértice es el propio vértice de origen.
4. $v \neq v_o \rightarrow P_n(v) \neq v$. Para todos los vértices distintos del vértice de origen, el predecesor es distinto del propio vértice.
5. $P_n(v) \in C_n$. El predecesor de un vértice $v \in V$ está visitado.
6. $D_n(v) = D_n(P_n(P_n(v))) + f(P_n(P_n(v)), P_n(v), v)$. La distancia hasta un vértice $v \in V$ depende de la distancia hasta sus predecesores en el camino óptimo.
7. Si $P_n(v)$ no es un vértice reducido y $P_n(v) \in C_n$, entonces $D_n(v) = D_n(P_n(v)) + f(P_n(v), P_n(v), v)$. Si el predecesor de un vértice v en la iteración n no está reducido y está visitado, la distancia hasta v, en este paso, es igual a la distancia hasta su predecesor más la distancia desde su predecesor hasta el propio vértice v.
8. $\forall v \in C_n$, $D_{n+1}(v) = D_n(v)$. Para los vértices visitados en la iteración n, la distancia en ese paso será igual a la distancia hasta dichos vértices en la iteración $n + 1$.
9. $\forall v_i, v_j \in V_j[v_j \in C_{n+1} \rightarrow D_{n+1}(v_i) \leq D_{n+1}(P_n(v_j)) + f(P_n(v_j), v_j, v_i)]$. La distancia hasta cualquier vértice $v_i \in V$ es menor o igual que la distancia hasta un vértice visitado $v \in C_n$ más el costo de ir desde el vértice visitado hasta el vértice v_i.

LEMA 3.2.1 *En la iteración n se cumple lo siguiente:* $w_n \notin C_n \wedge \forall v \in V(D_n(v) < D_n(w_n) \rightarrow v \in C_n)$

DEMOSTRACIÓN: $w_n \in V \backslash C_n$, además se cumple que $D_n[w_n]$ es óptimo, o sea, $\nexists v \in V \backslash C_n$ tal que $D_n[v] < D_n[w_n]$ por definición. Luego, si $\exists v \in V$ tal que $D_n[v] < D_n[w_n]$, el vértice v tuvo que ser escogido como el de menor distancia en una iteración $k < n$, de donde $v = w_k \in C_n$ por definición de C_n. □

LEMA 3.2.2 $\forall n < N$ *se cumple que* $|C_n| = n + 1$

DEMOSTRACIÓN: (Por inducción sobre n)
A partir de la definición del algoritmo, en cada paso se visita un vértice, antes de entrar en la primera iteración se visita el vértice de origen v_o, por lo que en el caso base se tiene $C_0 = \{v_o\}$, de donde $|C_0| = 1$, suponiendo que para $n = k$ se cumple, en la iteración $k + 1$ se tendría que $C_{k+1} = C_k \bigcup \{v\}$, siendo v el vértice visitado en el paso $k+1$, por tanto $|C_{k+1}| = |C_k| + |\{v\}| = k + 2$. □

LEMA 3.2.3 $\forall n < N$, *se cumple que* $D_n(v_o) = 0 \wedge P_n(v_o) = v_o$

DEMOSTRACIÓN: Inicialmente se visita el vértice v_o y se actualiza $D_n(v_o) = 0$, lo que quiere decir que la distancia mínima desde el vértice de origen v_o hasta él mismo es 0, la función D_n tiene su dominio en $\mathbb{R}^+ \bigcup \{0, \infty\}$, por lo que el menor valor posible que puede alcanzar es 0;
Sea $costo = D_n(P_n(w_n)) + f(P_n(w_n), w_n, v), \forall w_n, v \in V$ se cumple que $0 \leq 0 + costo$, debido a que la imagen de la función f es $\mathbb{R}^+ \cup \{0, \infty\}$ y el vector D es inicializado a partir de f.
Como nunca se cumple que $D_n(v_o) > D_n(w_n) + f(P_n(w_n), w_n, v_o)$, nunca se actualiza $D_n[v_o]$ ni $P_n[v_o]$. □

LEMA 3.2.4 $\forall n < N$, *se cumple que* $v \neq v_o \rightarrow P_n(v) \neq v$

DEMOSTRACIÓN: Inicialmente $\forall v \in V, P_0(v) = \{v_o\}$, por otra parte $\forall v \in Adyacentes(w_m), D_n(v) > D_n(P_n(w_n)) + f(P_n(w_n), w_n, v) \rightarrow (P_n(v) = w_n \wedge C_n = C_{n-1} \bigcup \{w_n\})$, suponiendo que $v = w_n$, $f(v, v, v) = 0$ por definición, luego $D_n(v) > D_n(P_n(w_n)) + f(P_n(w_n), w_n, v)$ nunca se cumple, por tanto nunca se le asigna a $P_n(v)$ el valor v. □

LEMA 3.2.5 $\forall n < N$, *se cumple que* $\forall v \in V, P_n(v) \in C_n$

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$, $P_0(v) = v_o, \forall v \in V$ y $C_0 = \{v_o\}$

para $n = k + 1$:

$P_{k+1}(v) \in \{P_k(v), w_k\}$, por definición de P

$P_k(v) \in C_k \subseteq C_{k+1}$, por hipótesis de inducción

$w_k \in C_{k+1} = C_k \cup \{w_k\}$ □

LEMA 3.2.6 $\forall n < N, D_n(v) = D_n(P_n(P_n(v))) + f(P_n(P_n(v)), P_n(v), v)$

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$, $\forall v \in V_r, D_0 = f_c(v_o, v)$, por precondiciones

$f(v_o, v_o, v) = f_c(v_o, v_o) + f_c(v_o, v) = f_c(v_o, v)$, por definición de f y f_c, sustituyendo f por f_c:

$D_0(v) = 0 + f(v_o, v_o, v)$

$D_0(v) = D_0(v_o) + f(v_o, v_o, v)$, por Lema 3.2.3

$D_0(v) = D_0(P_0(v_o)) + f(P_0(v_o), v_o, v)$, por Lema 3.2.3

Suponiendo que para $n = k$ la proposición se cumple, para $n = k + 1$ se tendría lo siguiente:

Seleccionar $w_{k+1} \in V \setminus C_k$ tal que $D_{k+1}(w_{k+1})$ sea óptimo, $C_{k+1} = C_k \bigcup \{w_{k+1}\}$.

Caso 1: Si $D_{k+1}(v) > D_{k+1}(P_{k+1}(w_{k+1})) + f(P_{k+1}(w_{k+1}), w_{k+1}, v)$ se hace $D_{k+1}(v) = D_{k+1}(P_{k+1}(w_{k+1})) + f(P_{k+1}(w_{k+1}), w_{k+1}, v) \wedge P_{k+1}(v) = w_{k+1}$

Caso 2: Si no se cumple el caso 1, $D_{k+1}(v) = D_k(v), P_{k+1}(v) = P_k(v)$, sustituyendo $D_k(v)$ según la hipótesis se tiene que:

$D_{k+1}(v) = D_k(P_k(P_k(v))) + f(P_k(P_k(v)), P_k(v), v)$, sustituyendo $P_k(v)$ por $P_{k+1}(v)$

$D_{k+1}(v) = D_k(P_{k+1}(P_{k+1}(v))) + f(P_{k+1}(P_{k+1}(v)), P_{k+1}(v), v)$. □

LEMA 3.2.7 $\forall n < N$, *se cumple que, si* $P_n(v)$ *no es reducido y* $P_n(v) \in C_n$, *entonces* $D_n(v) = D_n(P_n(v)) + f(P_n(v), P_n(v), v)$

La demostración de este lema es similar a la del Lema 3.2.6.

LEMA 3.2.8 $\forall v \in C_n \; D_{n+1}(v) = D_n(v)$

DEMOSTRACIÓN: Sea $v \in C_n, w_n \in V \setminus C_n$

$w_n \in C_{n+1}$ por definición.

$D_n(v) \leq D_n(w_n)$, de otra forma el vértice w_n hubiera sido visitado antes que el vértice v

$D_n(v) \leq D_n(w_n) + f_c(w_n, v) = D_n(P_n(w_n)) + f(P_n(w_n), w_n, v)$, aplicando la definición de $D_{n+1}(v)$, se obtiene que:

$D_{n+1}(v) = D_n(v)$. □

LEMA 3.2.9 $\forall n < N, \forall v_i, v_j [v_j \in C_{n+1} \rightarrow D_{n+1}(v_i) \leq D_{n+1}(P_n(v_j)) + f(P_n(v_j), v_j, v_i)]$

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$,

$C_0 = \{v_o\}, D_0(v_o) = 0$, por definición, note que el único vértice que pertenece a C_0 es v_o.

$P_n(v_o) = v_o$ por Lema 3.2.3.

$\forall v \in V, f(v_o, v_o, v) = f(P_n(v_o), v_o, v) \geq 0$, por definición de f, por tanto

$D_0(v) \leq 0 + f(P_n(v_o), v_o, v)$

$D_0(v) \leq D_0(P_n(v_o)) + f(P_n(v_o), v_o, v)$

Para $n = k + 1$:

Caso 1: $v_j \in C_k$

$D_{k+1}(v_i) \leq D_k(v_i)$ por definición

$D_{k+1}(v_i) \leq D_k(P_k(v_j)) + f(P_k(v_j), v_j, v_i)$ por hipótesis de inducción

$D_{k+1}(v_i) \leq D_{k+1}(P_k(v_j)) + f(P_k(v_j), v_j, v_i)$ por el Lema 3.2.8

Caso 2: $v_j = w_k$.

$D_{k+1}(v_i) \leq D_k(P_k(w_k)) + f(P_k(w_k), w_k, v_i)$, por definición

$D_{k+1}(v_i) \leq D_{k+1}(P_k(w_k)) + f(P_k(w_k), w_k, v_i)$, por el Lema 3.2.8, sustituyendo w_n por v_j:

$D_{k+1}(v_i) \leq D_{k+1}(P_k(v_j)) + f(P_k(v_j), v_j, v_i)$ □

Según el Lema 3.2.6, $D_{N-1}(v)$ tiene la distancia mínima hasta el vértice v partiendo del vértice v_o, para demostrar la corrección del Algoritmo 12 ($MDijkstra$) bastaría con demostrar que para cualquier camino $Ca = (v_o, v_1, v_2, ..., v_d)$ con su vector de distancia Dc y su vector de predecesores Pc, se cumple que $\forall v \in V, D_{N-1}(v) \leq Dc_{N-1}(v)$, siendo v un vértice sin reducir.

TEOREMA 3.2.1 $\forall n \in \{1, 2, .., N-1\}[Ca(0) = 0 \rightarrow \forall m < n+1(Ca(m) < N) \rightarrow \forall m < n+1[Dc(Ca(m)) + f_c(Ca(m), Ca(m+1)) = Dc(Ca(m+1))] \rightarrow D_{N-1}(Ca(n)) \leq Dc(Ca(n))]$

DEMOSTRACIÓN: (Por inducción sobre n)

Caso base $n = 0$, directo por el Lema 3.2.3

Para $n = k + 1$:

Por el Lema 3.2.2 en el paso $N - 1$ todos los vértices se han visitado.

$D_{N-1}(Ca(k+1)) \leq D_{N-1}(Ca(k-1)) + f(Ca(k-1), Ca(k), Ca(k+1))$ por Lema 3.2.9

$D_{N-1}(C(k+1)) \leq Dc(Ca(k-1)) + f(Ca(k-1), Ca(k), Ca(k+1))$ por hipótesis de inducción

véase que $Ca(k) = Pc_{N+1}(Ca(k+1))$ y $Ca(k-1) = Pc_{N-1}(Pc_{N-1}(Ca(k+1)))$, sustituyendo:

$D_{N-1}(Ca(k+1)) \leq Dc(Pc_{N-1}(Pc_{N-1}(Ca(k+1)))) + f(Pc_{N-1}(Pc_{N-1}(Ca(k+1))), Pc_{N-1}(Ca(k+1)), Ca(k+1))$, luego

$D_{N-1}(Ca(k+1)) \leq Dc(Ca(k+1))$, por el Lema 3.2.6. □

Realizando un razonamiento similar se puede demostrar la corrección del algoritmo de Dijkstra debido a que el mismo cumple con las mismas invariantes sustituyendo la función f por la función f_c (la función de costo de la Definición 1.1.3.9 de grafo ponderado). Por lo que se asume para la siguiente demostración que el algoritmo Dijkstra es correcto y cumple con las invariantes definidas para el Algoritmo 12 ($MDijkstra$).

Con lo anterior queda demostrado que el Algoritmo 12 retorna el camino óptimo en el grafo reducido, ahora resta demostrar que el costo del camino óptimo obtenido por este algoritmo es el mismo que el que se obtiene en el grafo original (sin reducir) con el algoritmo de Dijkstra.

Sean:

- $G = (V, E, f_c)$ un grafo.
- $G_r = (V_r, E_r, f)$ un grafo reducido obtenido a partir del grafo G.

TEOREMA 3.2.2 *Sea $Ca = (v_1, ..., v_n)$ un camino de costo c, obtenido al aplicar el algoritmo Dijkstra sobre el grafo G, donde v_1 y v_n son vértices no reducidos en el grafo G_r, entonces $\exists Ca' = (u_1, u_2, \ldots, u_t)$ de costo c, $u_1 = v_1, u_t = v_n$, tal que Ca' es un camino óptimo en G_r.*

DEMOSTRACIÓN: A partir de Ca se puede construir un camino Ca' de costo c sobre el grafo G_r como sigue:

- Sustituir cada sub-camino $v_i, v_{i+1}, \ldots, v_{i+m}$ por un camino v_i, v_k, v_{i+m} donde:

  - $v_{i+j} \in [v_i], j = 1..m$.
  - v_i, v_{i+m} son vértices exteriores. Los otros vértices son interiores.
  - v_k es un vértice reducido en el grafo G_r que representa la clase A_i, tal que $v_{i+1}, \ldots, v_{i+m-1} \in A_i$.

El costo del camino v_i, v_k, v_{i+m} es igual al costo del camino $v_i, v_{i+1}, \ldots, v_{i+m}$, por la definición de la función f. Luego, los caminos Ca y Ca' tienen el mismo costo.
Suponga que $\exists Cb' = (u_1, u_2, \ldots, u_p)$ de costo $c_1 < c$ en el grafo G_r, donde $u_i \in V_r, i = 1..p$. Entonces se puede obtener un camino Cb de costo c_1 sobre el grafo G como sigue:

- Sustituir cada sub-camino u_{i-1}, u_i, u_{i+1} por un camino $u_{i-1}, u_j, u_{j+1}, u_{j+m}, \ldots, u_{i+1}$ de costo c_3 donde:

  - u_{i-1}, u_{i+1} son vértices no reducidos.
  - $u_{j+t} \in A_i, j = 0..m$, siendo u_i el vértice representativo de la clase A_i.
  - $c_3 = f(u_{i-1}, u_i, u_{i+1})$.

Finalmente, los caminos Cb y Cb' tienen el mismo costo (c_1), esto es una contradicción porque se partió de que el camino Ca de costo c es un camino óptimo. Luego, no existe un camino que tenga un menor costo que Ca. □

COROLARIO 3.2.2.1 *Sea $Ca = (v_1, ..., v_n)$ un camino obtenido aplicando el algoritmo de Dijkstra en el grafo G, $\forall i \in \{1, 2, ..., n\}$ tal que $Ca[i]$ es un vértice no reducido en G_r, se cumple que $D_{N-1}(Ca[i]) = d_{M-1}(Ca[i])$*

En el teorema 3.2.2 se demuestra que el costo del camino óptimo entre un vértice v_i y un vértice v_j (v_i y v_j son vértices no reducidos en G_r) obtenido al aplicar el Algoritmo 12 ($MDijkstra$) sobre G_r es igual al costo del camino óptimo obtenido luego de ejecutar el algoritmo de Dijkstra en el grafo original (sin reducir) para el mismo par de vértices. El hecho de que el vértice de origen y destino tienen que ser vértices no reducidos podría suponer una limitante (en cuanto a la cantidad de vértices hasta los que se puede calcular el camino óptimo) si no se contara con un mecanismo que permita, para cualquier vértice v, obtener un grafo reducido G'_r a partir de G_r

que contenga al vértice v como vértice no reducido. Esto se puede lograr aplicando una o varias expansiones al vértice reducido que contenga al vértice v.

3.3. Aplicación del modelo propuesto en el SIG Quantum GIS

La realización computacional del modelo propuesto se dividió en dos partes. En primer lugar se desarrolló una biblioteca de clases que implementa los algoritmos propuestos. Luego, se desarrolló un plugin para el SIG QGIS que brinda la funcionalidad de búsqueda de caminos óptimos haciendo uso del modelo presentado.

3.3.1. Biblioteca de clases

La biblioteca de clases fue implementada utilizando el lenguaje de programación Python (versión 2.7.3), la biblioteca NetworkX [168] (versión 1.6-2), entre otras bibliotecas auxiliares. Se implementaron tres clases principales:

- AlgoritmoReducción
- GrafoReducido
- Útiles

La clase AlgoritmoReducción es la que implementa la lógica relacionada con la reducción de grafos, la misma hace uso de otras clases implementadas que representan una partición, la función f, las reglas de reescritura, entre otras.
La clase GrafoReducido representa la definición de grafo reducido que se utiliza en el presente trabajo (ver Definición 2.2.2). En la misma se implementan los algoritmos relacionados con la búsqueda de caminos óptimos propuestos en esta investigación, los cuales son propios de grafos reducidos. Se utilizó como estructura de datos para la implementación listas de adyacencia.
La clase GrafoReducido hereda de la clase MultiDiGrafo definida en la biblioteca NetworkX, por lo que en la primera están disponibles todas las operaciones definidas en la clase MultiDiGrafo de la biblioteca antes mencionada.
Por último, la clase Útiles implementa varios métodos que son utilizados por las dos clases mencionadas anteriormente. Por ejemplo, las relaciones de equivalencia que son entrada del algoritmo de reducción.

3.3.2. BCM: plugin para el análisis de rutas en QGIS

La implementación computacional del modelo propuesto fue incluida como parte de las funcionalidades que brinda el SIG QGIS. Este sistema es uno de los más utilizados en el ámbito de software libre para el manejo y consulta de datos geográficos.

Para el desarrollo de este plugin se partió del diagrama de componentes de QGIS especificado en la Figura 3.1. En la Figura 3.2 se muestra la forma en que se integra el plugin desarrollado con el sistema QGIS. Como se puede apreciar, se hace uso de funcionalidades que se encuentran en el componente QGIS Canvas, QGIS GUI y QGIS Core; además se utiliza la biblioteca para el trabajo con grafos reducidos (GrafoReducido), desarrollada en el marco de esta investigación, así como una interfaz de usuario (BCM.UI) que permite seleccionar el origen y el destino para luego realizar la búsqueda de camino óptimo. Finalmente, el componente BCM es el encargado de comunicar la biblioteca para el trabajo con grafos reducidos y las funcionalidades implementadas en QGIS con la interfaz de usuario diseñada.

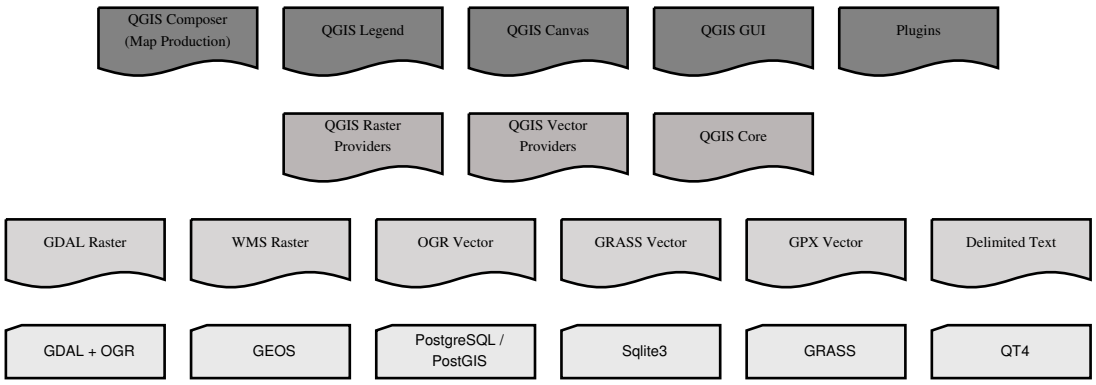


Figura 3.1: *Diagrama de componentes de QGIS. Tomado de [169].*

El plugin desarrollado se aplicó utilizando un mapa de La Habana, descargado del proyecto OpenStreetMap. Para la búsqueda de caminos óptimos sobre este mapa, haciendo uso del modelo propuesto, se definió una partición en la cual dos puntos están relacionados si ambos están dentro del mismo polígono. Los polígonos utilizados fueron creados dividiendo la región de La Habana en polígonos que tienen un área similar.

Una vez creado el grafo a partir de la capa de la red de viales del mapa descargado, se pudo constatar que dicha capa presentaba problemas de conexión entre las calles (note que

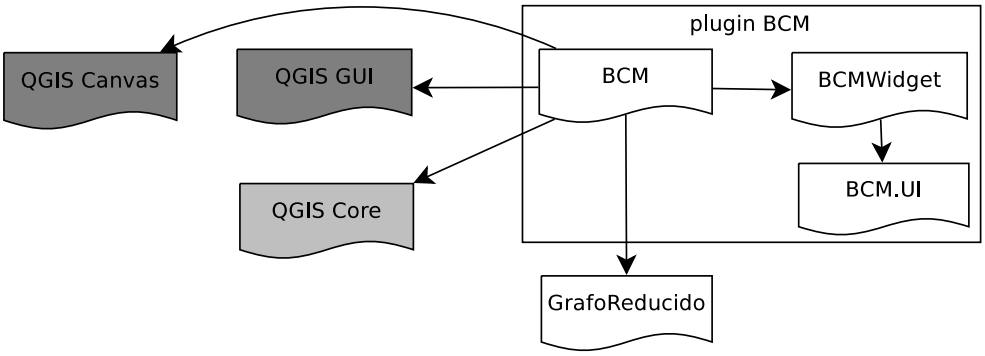


Figura 3.2: *Diagrama de integración del plugin desarrollado con QGIS.*

OpenStreetMap es un proyecto libre, donde las personas adicionan datos sin responsabilizarse por la calidad de los mismos) ya que el grafo generado no era conexo. Debido a esto, se realizaron las pruebas de búsqueda de camino óptimo con la componente conexa del grafo obtenido de mayor tamaño, o sea, de una mayor cantidad de vértices. Esta componente se corresponde con un grafo de 1413 vértices. Luego de la reducción, se obtuvo un grafo de 200 vértices.

En la Figura 3.3 se muestran cuatro ejemplos de búsqueda de camino óptimo con el plugin desarrollado, haciendo uso del grafo reducido obtenido. Se comprobó que el costo del camino que se obtuvo coincide con el costo del camino obtenido haciendo uso del algoritmo de Dijkstra. De forma general, resulta difícil realizar una comparación del tiempo de respuesta de varios algoritmos en un sistema que cuenta con interfaz gráfica de usuario como es el caso de QGIS. Es por ello que en este epígrafe se muestran ejemplos de búsqueda de camino óptimo obtenidos con el plugin desarrollado. Sin embargo, se hace necesario realizar una comparación del tiempo de ejecución del algoritmo de búsqueda de camino óptimo propuesto con otros algoritmos, en aras de corroborar los resultados demostrados teóricamente. Para ello, se realizaron experimentos, definidos en la próxima sección.

3.4. Resultados experimentales

Como parte de la validación del presente trabajo se propone utilizar el método experimental. A continuación se describen los materiales y métodos que se utilizaron y se muestran los datos resultantes de la realización de los experimentos definidos. Por último se realiza una

Figura 3.3: *Ejemplo de búsqueda de camino óptimo con el plugin desarrollado para el SIG QGIS.*

comparación entre los algoritmos de Dijkstra y A* y el algoritmo de búsqueda de camino óptimo propuesto, en cuanto a la eficiencia y escalabilidad de los mismos.

3.4.1. Materiales y métodos

Para las pruebas experimentales se utilizaron tres grafos. A continuación se describe la forma en la que se obtuvieron los mismos.

El primer grafo se obtuvo a partir del mapa del estado Carolina del Norte[1] y tiene 41810 vértices (G_1). Este grafo fue reducido dos veces (utilizando relaciones de equivalencia basadas en el código postal) obteniéndose un grafo de 250 vértices ($G_{r1.1}$) y el otro de 1826 vértices ($G_{r1.2}$).

El segundo grafo se obtuvo a partir de la red de viales de Nova Scotia[2] (Canadá) y tiene 63509 vértices (G_2). Este grafo fue reducido dos veces (utilizando dos particiones de vértices de forma

[1] Disponible en `http://grass.osgeo.org/sampledata/north_carolina/`

[2] Disponible en `http://geodepot.statcan.ca/diss/2006dissemination/data/frr_rnf_e.cfm`

tal que todos los vértices de una clase de la partición estuvieran conectados a través de un camino) obteniéndose un grafo de 517 vértices ($G_{r2.1}$) y otro de 936 vértices ($G_{r2.2}$).
El tercer grafo representa la red de viales de la ciudad de San Francisco[3], cuenta con 174956 vértices (G_3) y fue reducido dos veces (utilizando dos particiones de vértices de forma tal que todos los vértices de una clase de la partición estuvieran conectados a través de un camino), obteniéndose un grafo de 769 vértices ($G_{r3.1}$) y el otro de 2617 vértices ($G_{r3.2}$).
A partir de los datos anteriores, se definieron los siguientes elementos presentes en el diseño de los experimentos:

- Unidad básica de análisis: grafos.
- Población: grafos obtenidos a partir de los mapas mencionados anteriormente. La característica distintiva que tienen todos los miembros de la población es que deben ser grafos conexos.
- Muestra:
  - G_1, grafo con 41810 vértices.
  - $G_{r1.1}$, grafo reducido a partir del grafo G_1 con 250 vértices.
  - $G_{r1.2}$, grafo reducido a partir del grafo G_1 con 1826 vértices.
  - G_2, grafo con 63509 vértices.
  - $G_{r2.1}$, grafo reducido a partir del grafo G_2 con 517 vértices.
  - $G_{r2.2}$, grafo reducido a partir del grafo G_2 con 936 vértices.
  - G_3, grafo con 174956 vértices.
  - $G_{r3.1}$, grafo reducido a partir del grafo G_3 con 769 vértices.
  - $G_{r3.2}$, grafo reducido a partir del grafo G_3 con 2617 vértices.

A partir de la muestra seleccionada, se hará referencia a los grafos G_1, G_2 y G_3 como los grafos originales y a los restantes como los grafos reducidos.
Se considera que no es necesario utilizar una población y muestra más amplia ya que se demostró en los epígrafes anteriores, teóricamente, que los resultados son generalizables a todos los grafos.
El modelo propuesto fue implementado utilizando el Lenguaje de Programación Python, la biblioteca de clases NetworkX [168], entre otras bibliotecas auxiliares. La biblioteca de clases

[3]Disponible en `http://www.cs.fsu.edu/~lifeifei/SpatialDataset.htm`

NetworkX, brinda una implementación de los algoritmos de Dijkstra, A*, entre otros; esto permitió comparar el tiempo de ejecución del Algoritmo $MDijkstra$ con el tiempo de ejecución de los dos algoritmos anteriormente mencionados, haciendo uso de la misma tecnología.
Los experimentos fueron ejecutados en una Computadora Personal (PC) con un procesador Intel(R) Pentium(R) 4 de 3.20GHz [512 Kb de cache] con 1.5Gb RAM, sobre el sistema operativo Kubuntu 11.10.
A continuación se enumeran los experimentos realizados:

Experimento 1: Aplicar los algoritmos de Dijkstra y A* a los grafos originales.

Experimento 2: Aplicar el Algoritmo 12 ($MDijkstra$) a los grafos reducidos.

Cada algoritmo se ejecutó 10 veces, descartándose el mayor y menor valor en cada caso. Finalmente, se calculó el promedio de los restantes 8 valores.

3.4.2. Resultados y discusión

Los resultados de los experimentos 1 y 2 se muestran en la Tabla 3.1, en la cual la última columna indica si el algoritmo en cuestión obtuvo el valor óptimo. Adicionalmente, la Tabla 3.2 muestra la razón entre el tiempo que el algoritmo seleccionado (algoritmo de Dijkstra o el A*) demora ante una petición de búsqueda de camino óptimo en los grafos originales y el tiempo que el Algoritmo 12 ($MDijkstra$) demora en responder a la misma petición en los grafos reducidos. Este valor significa la cantidad de veces que los algoritmos de Dijkstra y A* son más lentos que la propuesta realizada en la presente investigación.
Adicionalmente, en la Figura 3.4 se aprecia que los tiempos de respuesta del Algoritmo 12 ($MDijkstra$) son menores, en los experimentos realizados, que los tiempos obtenidos por los algoritmos de Dijkstra y A*.
Varios algoritmos heurísticos se han diseñado para reducir el tiempo de respuesta de la búsqueda de caminos óptimos en SIG reduciendo el espacio de búsqueda de la solución, para ello se asume que un bajo porcentaje de error es admisible en esta área.
Como se puede apreciar en la Tabla 3.1, el algoritmo A* retorna una respuesta en tiempos menores que el algoritmo de Dijkstra, pero en algunos casos no alcanza el óptimo. Sin embargo,

Tabla 3.1: *Tiempo de ejecución del algoritmo de Dijkstra y A* en los dos grafos originales (G_1, G_2) y del Algoritmo 12 ($MDijkstra$) en los grafos reducidos.*

Grafo	Algoritmo	CV[a]	Tiempo (segundos)	Óptimo
G_1	Dijkstra		0.6160	sí
	A* (h=0)	41810	0.4938	sí
	A* (h=distancia euclidiana)		0.0200	no
$G_{r1,1}$	Algoritmo 12 ($MDijkstra$)	250	0.0036	sí
$G_{r1,2}$	Algoritmo 12 ($MDijkstra$)	1826	0.0265	sí
G_2	Dijkstra		1.0112	sí
	A* (h=0)	63509	0.1454	sí
	A* (h=distancia euclidiana)		0.2846	sí
$G_{r2.1}$	Algoritmo 12 ($MDijkstra$)	517	0.0158	sí
$G_{r2.2}$	Algoritmo 12 ($MDijkstra$)	936	0.0338	sí
G_3	Dijkstra		3.0249	sí
	A* (h=0)	174956	2.2108	sí
	A* (h=Euclidean distance)		0.1011	no
$G_{r3,1}$	Algoritmo 12 ($MDijkstra$)	765	0.0193	sí
$G_{r3,2}$	Algoritmo 12 ($MDijkstra$)	2617	0.0722	sí

[a] Cantidad de vértices del grafo.

con el modelo propuesto es posible obtener un camino óptimo en un tiempo similar al que presentan algunos algoritmos heurísticos, e incluso menor, como se muestra en la tabla antes mencionada. Además, el Algoritmo 12 ($MDijkstra$) es escalable en la muestra seleccionada, el tiempo de respuesta varía entre 0.0036 y 0.0722 segundos (una diferencia de 6,86 milisegundos) para un incremento de 107955 vértices del grafo original.

Se puede afirmar que el modelo propuesto garantiza escalabilidad en la búsqueda de caminos óptimos de manera general, porque dado un grafo conexo se puede encontrar una partición que genere un grafo reducido que tenga una cantidad de vértices óptima (ver epígrafe 2.4) para la

Tabla 3.2: *Razón entre los tiempos de ejecución de los algoritmos de Dijkstra y A* y el Algoritmo 12 ($MDijkstra$).*

Grafo	Algoritmo	Razón	
		$G_{r1.1}$	$G_{r1.2}$
G_1	Dijkstra	171.1111	23.2453
	A* (h=0)	137.1667	18.6340
	A* (h=distancia euclidiana)	5.5556	0.7547
		$G_{r2.1}$	$G_{r2.2}$
G_2	Dijkstra	63.9873	29.9171
	A* (h=0)	9.2025	4.3017
	A* (h=distancia euclidiana)	18.0126	8.4201
		$G_{r3.1}$	$G_{r3.2}$
G_3	Dijkstra	156.7306	41.8961
	A* (h=0)	114.5492	30.6205
	A* (h=distancia euclidiana)	5.2383	1.4003

búsqueda de caminos óptimos.

Por otra parte, el algoritmo de búsqueda de caminos óptimos propuesto es más eficiente que los algoritmos de Dijkstra y A*, como se puede apreciar en la Tabla 3.2.

Adicionalmente, en [170] se presenta una implementación del modelo propuesto en esta investigación, mostrando la viabilidad de implementación como parte de un motor de persistencia de grafos que realiza el análisis en memoria externa. Lo cual sería conveniente si se cuenta con una PC que no cuenta con recursos de hardware suficientes para realizar el análisis en RAM.

En este caso, la implementación relacionada con análisis de redes se realizó como parte del propio motor de persistencia, lo cual garantiza eficiencia en cuanto al acceso a los datos. Por otra parte, la implementación del modelo realizada es dependiente de Neo4j, pero hace uso de las facilidades que brinda de acceso a los datos reduciendo el tiempo de carga de los mismos. Estamos en presencia de un balance entre generalización y eficiencia.

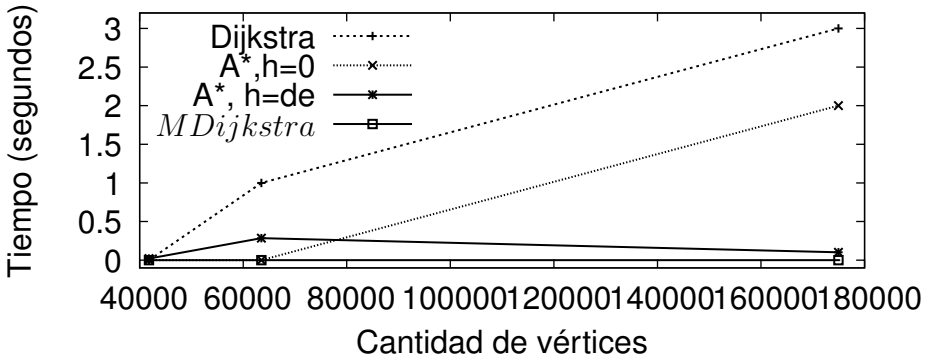


Figura 3.4: *Comparación de tiempos de respuesta de los algoritmos Dijkstra, A* y* $MDijkstra$.

3.5. Aplicación de la reducción de grafos al Método de los Grafos Dicromáticos

El algoritmo de reducción de grafos que se presenta en esta investigación, fue diseñado para la representación y análisis de redes en SIG. Sin embargo, en este epígrafe se muestra que su valor práctico no se limita a ese entorno, sino que puede ser aplicado para resolver otros tipos de problemas en los que aparece un grafo con determinada complejidad y resulta útil obtener un grafo reducido. Tal es el caso del Diseño Racional en Ingeniería Mecánica.

Con el fin de obtener algoritmos para resolver problemas cuya solución es un conjunto de magnitudes, es decir, problemas de cómputo, Martínez Escanaverino comenzó a publicar desde 1997 lo que denominó el Método de los Grafos Dicromáticos (MGD) [171, 172, 173, 174].

Por otra parte, los modelos matemáticos de ingeniería suelen ser construidos por partes, o sea, al ir vinculando los diferentes sub-modelos entre sí se va componiendo un modelo que describe de algún modo la ingeniería de un objeto determinado. La representación de dicho modelo puede realizarse de acuerdo al MGD y en ese caso, es posible depurar el grafo obtenido de diferentes maneras. Una de ellas puede basarse en la decisión de cuál parte del modelo es necesario considerar y cuál no. Por ejemplo, en un caso podría interesar exclusivamente el modelado riguroso de la geometría de uno solo de los mecanismos de una máquina, sin considerar la cinemática, la dinámica, las tensiones o los costos. El modelo es entonces restringido a una parte del modelo matemático general, eligiéndose mediante ese criterio cuáles ecuaciones y variables

incluir en este subconjunto.

Otro ejemplo se puede encontrar en el nivel de detalle con que se representa el sistema objeto de estudio; en este caso, al realizar cálculos simplificados en algunos momentos del proceso de diseño de máquinas, solo se abarcan los criterios elementales de cálculo. Además, se suelen asignar valores característicos provisionales a determinadas variables, las cuales quedan convertidas así en datos, mientras que otras variables son declaradas a ese nivel como salidas y de esta forma el modelo resulta simplificado. A esta forma de simplificación se le nombrará reducción por partes.

La representación de los modelos matemáticos de ingeniería puede generar grafos de gran complejidad, lo que lleva a la necesidad de acudir a herramientas de reducción de grafos para su análisis y manipulación.

3.5.1. Breve descripción del Método de los Grafos Dicromáticos

El MGD representa la estructura de modelos matemáticos, situaciones, problemas y algoritmos, mediante grafos dicromáticos que se van transformando sucesivamente. Ha sido aplicado por varios autores según se puede ver en tesis doctorales [175, 176, 177, 178] y publicaciones [179, 180], en las cuales se demuestra que este significa un paso de avance desde el punto de vista práctico en la obtención directa de algoritmos basados en modelos matemáticos.

En la Figura 3.5 se puede apreciar el esquema de ejecución del MGD.

Inicialmente se construye el grafo del modelo, cuyos vértices representan las variables con un color y las ecuaciones (relaciones) con otro; las aristas simbolizan cuáles variables se encuentran en cada ecuación. Una vez planteado un problema de cómputo determinado, se descartan de dicho grafo las variables de entrada (datos), quedando transformado así en el grafo de la situación. A partir de este, suprimiendo las componentes conexas que no contengan variables de salida, se obtiene el grafo del problema. Este último grafo es sometido a un pareo que orienta, hacia su correspondiente variable, a una sola de las aristas conectadas con cada ecuación, de modo que el mismo sea un pareo máximo. Así pasa a obtenerse una nueva representación llamada grafo del problema pareado. Posteriormente se obtiene el grafo del resolvente, asignándole orientación de variable a ecuación a las aristas que aún no la tienen.

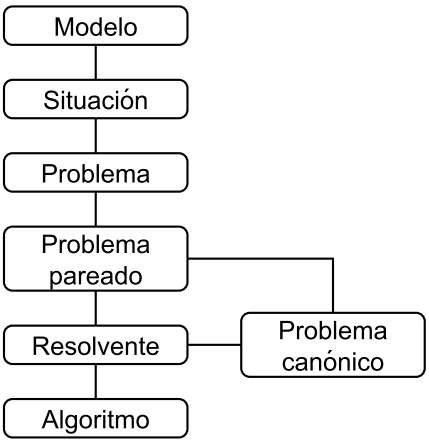


Figura 3.5: *Esquema de ejecución del MGD.*

Para los problemas que no pueden ser caracterizados plenamente, a partir de este grafo del resolvente es necesario obtener un nuevo grafo llamado problema canónico (si el problema tiene un pareo perfecto ya se encuentra en su forma canónica) y de este se llega al resolvente definitivo. A partir de dicho resolvente (que ya se encuentra en su forma canónica) se alcanza el grafo del algoritmo descartando los caminos que no conducen a ninguna de las variables de salida definidas al plantear el problema. A través de este método, siguiendo los pasos indicados en el grafo del algoritmo, se puede dar solución al problema inicialmente planteado.

Existen herramientas de edición de grafos que facilitan en cierto grado la aplicación del MGD; sin embargo, no se cuenta con un sistema informático para aplicar el mismo, lo que limita en cierta medida su valor práctico.

3.5.2. Aplicación del Algoritmo 8 al Método de los Grafos Dicromáticos

A partir de la descripción del MGD se puede apreciar que, de forma general, el método consiste en la ejecución de varias transformaciones de grafos. Por otra parte, atendiendo a la cantidad de vértices y aristas del grafo obtenido en cada paso, el MGD se puede ver como una serie de reducciones de grafos donde cada reducción descarta información que no es necesaria para el paso siguiente.

En la práctica los ingenieros e investigadores, durante el proceso de diseño, dividen los modelos matemáticos en diferentes sub-modelos. Esto se hace debido a que, en no pocas ocasiones, el modelo obtenido alcanza dimensiones inmanejables desde el punto de vista práctico. Como ejemplo de lo anterior se pueden citar los trabajos realizados por Llamos [175], en el cual se divide el modelo en seis sub-modelos; y por Alemán, quien descompone un modelo matemático matricial en 15 sub-modelos [176]. Es válido mencionar que estas divisiones son realizadas

manualmente, por lo que su aplicación no está generalizada debido a la dificultad que representa en la práctica.

Es posible aplicar el Algoritmo 8 en función de facilitar el trabajo de los ingenieros, ya sea en la aplicación del MGD o en la división de un modelo matemático en varios sub-modelos como se muestra en [181]. Esto traería consigo los siguientes beneficios:

- A partir de un grafo obtenido mediante un proceso de reducción con el Algoritmo 8, se pueden obtener los grafos generados en cualquiera de las etapas anteriores. Esto se debe a que se garantiza que no haya pérdida de información.
- Se puede contribuir a la automatización del MGD, considerando que este es un proceso de reducción de grafos.
- Se puede facilitar la reducción por partes o sub-modelos del grafo de un modelo matemático determinado, proceso comúnmente necesario en el diseño mecánico.
- Es posible contribuir a la modularidad del algoritmo de resolución de un determinado problema obtenido a partir del MGD y a su posterior mantenimiento. Esto se lograría, identificando partes que estén presentes en el grafo del algoritmo, para generar componentes de software de acuerdo a las mismas.
- Se pueden efectuar reducciones automatizadas del grafo del modelo o del grafo del resolvente, atendiendo a criterios semánticos y estructurales del fenómeno que se modela; en lugar de reducir (o agrupar) solamente según la estructura del grafo obtenido en algunos de los pasos del MGD.

Los referidos beneficios serán ejemplificados a continuación en dos casos:

1. Descripción de algunas de las transformaciones que se realizan a partir del grafo del modelo:
    - Descartar vértices que representen variables de entrada (datos).
    - Identificar componentes conexas y descartar las que no tengan variables de salida.
    - Descartar caminos que no contengan variables de interés para la solución.
2. Descripción de la reducción según las partes del fenómeno que se modela.

El algoritmo de reducción de grafos que se propone utilizar, recibe como entrada una relación de equivalencia para obtener una partición sobre el conjunto de vértices del grafo que se desea reducir, aunque también pudiera tener como entrada una partición sobre dicho conjunto. Para definir las transformaciones se asume que un vértice del grafo del problema tiene la siguiente información:

- $Tipo\_vértice$, puede tomar los valores variable y ecuación.
- $Tipo\_variable$, puede tomar los valores entrada, incógnita y salida.
- $Es\_de\_interés$, puede tomar los valores sí y no.
- Sub_modelo, el valor de esta variable es el nombre del sub-modelo al que pertenece el vértice.

3.5.2.1. Transformaciones a realizar a partir del grafo del modelo

Sea $G = (V, E)$, el grafo de un modelo definido según el MGD, se definen tres relaciones de equivalencia para realizar las transformaciones propuestas:

1. Descartar los vértices que representen variables de entrada: Sea $v \in V$ una variable de entrada y la clase de equivalencia $[v] = \{u | u \text{ es una variable de entrada}\}$, se define la partición $P = (\{v_i\}_{v_i \notin [v]}, [v])$.
2. Identificar las componentes conexas y descartar las que no tengan variables de salida: Dos vértices $u, v \in V$ están conectados si y solo si existe un camino que los une. La relación de conexión R_c definida en V por $R_c(u, v)$ si y solo si u y v están conectados, es una relación de equivalencia. El sub-grafo de G determinado por una clase de equivalencia de R_c es una componente conexa de G.
3. Descartar los problemas elementales que no sean necesarios para obtener las variables de salida: Sea la clase de equivalencia $[v] = \{u | (u, v) \in R_3\}$, $u, v \in R_3$ si y solo si $\exists (u, v) \in E$, u representa una ecuación y v representa una variable en u que no tiene vértices adyacentes, se define una partición $P = (\{v_i\}_{v_i \notin [v]}, [v])$.

3.5.2.2. Reducción por partes

Para realizar este tipo de reducción se define una relación de equivalencia $R_4 = \{(u, v) | u, v \in V \wedge u.Sub\_modelo = v.Sub\_modelo\}$, a partir de la igualdad en el atributo Sub_modelo de

los vértices del grafo. Para realizar este tipo de reducción, bastaría con invocar el algoritmo de reducción de grafos dando como entrada el grafo del modelo y la partición que define la relación de equivalencia R_4. Advierta que cuando se crea el grafo reducido, las variables que son comunes a más de un sub-modelo matemático serán representadas como vértices exteriores (ver DEFINICIÓN 2.2.1.1), mostrando cómo se relacionan los sub-modelos de las diferentes partes del modelo original.

3.5.3. Estudio de caso

A continuación se muestra una ejecución del algoritmo de reducción sobre el grafo del modelo de unas pinzas de fricción cuyo proceso de diseño se describe en [180]. En la Figura 3.6 se muestra el grafo del modelo del ejemplo seleccionado. A partir del mismo se realizará una reducción por partes, teniendo en cuenta los sub-modelos que componen el modelo completo.

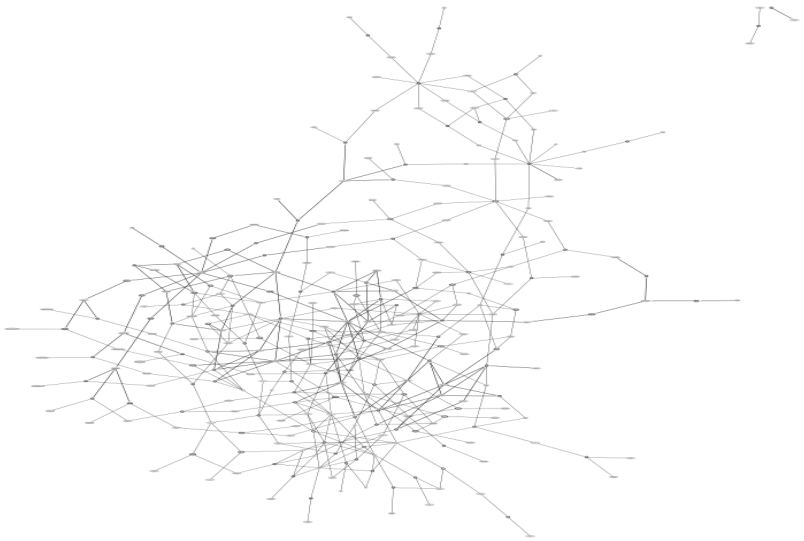


Figura 3.6: *Grafo del modelo. Tomado de [180].*

En la Figura 3.7 se muestra el primer paso de la reducción. En el mismo se agruparon todas las ecuaciones correspondientes al sub-modelo "Parámetros del modelo geométrico" y las variables de este sub-modelo que no tienen influencia en otros. Las variables que son comunes a varios sub-modelos se dejan sin agrupar, para que gráficamente se pueda observar cómo se relacionan los diferentes sub-modelos.

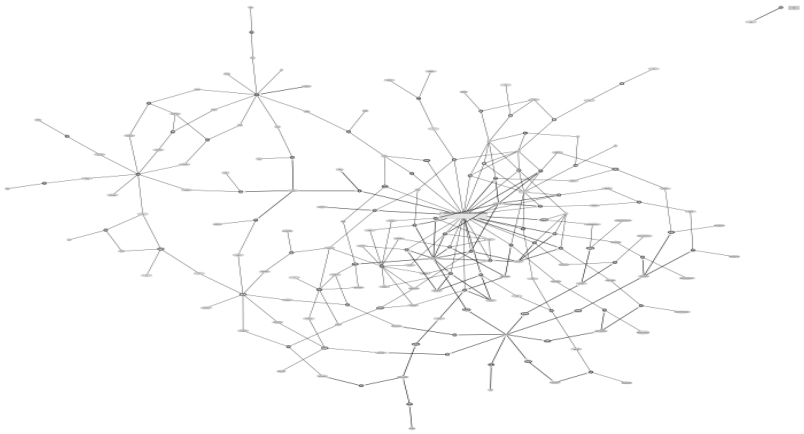


Figura 3.7: *Grafo reducido por Modelo Geométrico.*

Luego de este paso, se procede a agrupar las ecuaciones y variables que pertenecen solo a los sub-modelos “Cargas y reacciones”, “Masas”, “Resistencia Mecánica”, “Operaciones tecnológicas”, “Costo aproximado de producción” y “Tiempo de fabricación”. Posteriormente, se agrupan las variables que son comunes a dos sub-modelos. Las variables comunes a más de dos sub-modelos no se agrupan, ya que en la mayoría de los casos existe una sola variable común a cada trío posible de sub-modelos. Sin embargo, en caso de existir más de dos sub-modelos que tengan varias variables en común, se puede seguir un procedimiento similar al caso de las variables comunes a dos sub-modelos.

Al finalizar este proceso se obtiene el grafo de la Figura 3.8. En el mismo se han representado con rectángulos los vértices reducidos que corresponden a un sub-modelo, con elipses de fondo gris los vértices reducidos que contienen grupos de variables que relacionan exactamente dos sub-modelos (especificados en la etiqueta del vértice) y con elipses de fondo blanco los vértices no reducidos que representan el resto de las variables del modelo. Como se puede apreciar, se mantiene la condición de que el modelo matemático está representado con un grafo dicromático, lo que le permite al ingeniero una mejor interpretación de dicho modelo.

Un aspecto importante a recalcar es que el proceso aquí presentado es reversible. Esto se sostiene a partir de que el algoritmo de reducción de grafos no tiene pérdida de información y además propone un mecanismo que permite que, dado un grafo reducido (con el algoritmo en cuestión), se puede obtener cualquiera de los grafos generados durante el proceso de obtención del mismo,

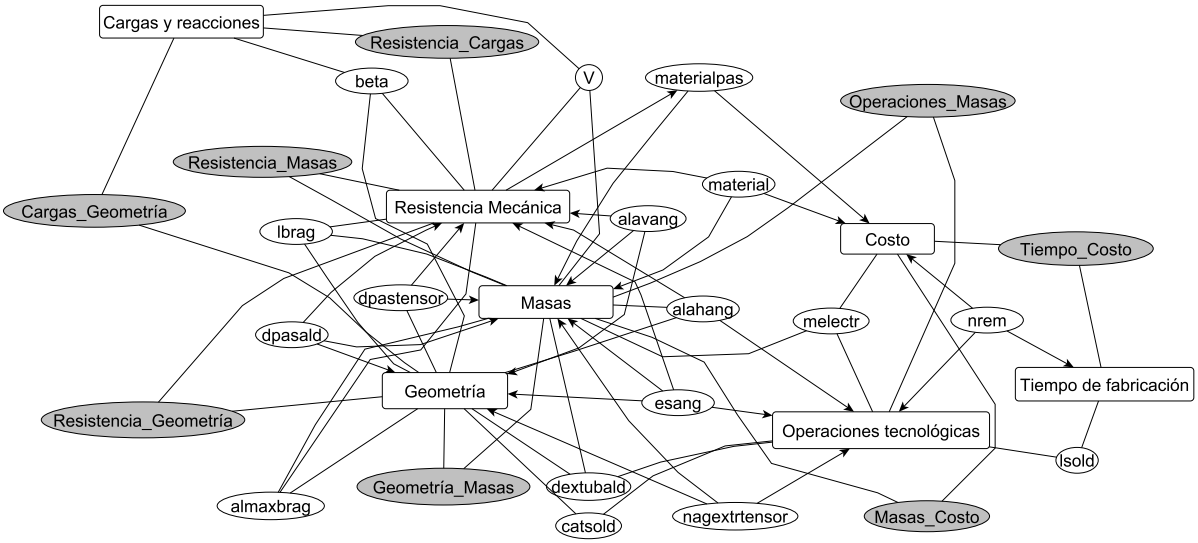


Figura 3.8: *Grafo reducido teniendo en cuenta todos los sub-modelos.*

incluyendo el grafo original (en este caso el grafo del modelo).

3.6. Conclusiones del capítulo

Luego del trabajo realizado se arribó a las siguientes conclusiones:

- Haciendo uso de grafos reducidos se puede reducir considerablemente el tiempo de respuesta en la búsqueda de caminos óptimos.
- Realizar la búsqueda de caminos óptimos sobre un grafo reducido garantiza escalabilidad respecto al tamaño del grafo sobre el que se realiza el análisis.
- La búsqueda de caminos óptimos en un grafo reducido con el algoritmo propuesto permite obtener un camino óptimo de igual costo al obtenido en una búsqueda realizada en el grafo sin reducir.
- A través de un algoritmo de reducción de grafos es posible formalizar el proceso de reducción de un grafo que representa un modelo matemático, específicamente el obtenido a partir de la aplicación del MGD.
- El análisis de un modelo matemático, resultante del diseño mecánico, se ha realizado en la práctica dividiendo el mismo en sub-modelos. La propuesta realizada en la presente

investigación contribuye a la automatización del proceso que realizan los ingenieros de manera natural y empírica, facilitando el análisis del modelo completo.

- A partir de los resultados obtenidos se justifica el estudio de la aplicación de la reducción de grafos como mecanismo de reducción de modelos matemáticos.

CONCLUSIONES

Como resultado de la presente investigación se obtuvo un modelo para la representación y análisis de redes en SIG basado en grafos reducidos, el mismo garantiza eficiencia y escalabilidad en la búsqueda de caminos óptimos cuando las redes son grandes. En base a los resultados obtenidos se arribó a las siguientes conclusiones:

1. El algoritmo de reducción de grafos diseñado como parte de esta investigación garantiza que no se pierde información en el proceso de reducción. Además, se demostró que un grafo reducido se puede utilizar para resolver el problema de la búsqueda de caminos óptimos.
2. Con las demostraciones de corrección realizadas se garantiza que el Algoritmo 12 ($MDijkstra$), en grafos reducidos, obtiene caminos de igual costo a los obtenidos por el algoritmo de Dijkstra en los grafos sin reducir.
3. Haciendo uso de grafos reducidos se puede disminuir considerablemente el tiempo de respuesta en la búsqueda de caminos óptimos, garantizando así la eficiencia; lo cual queda evidenciado en los resultados experimentales obtenidos. Además, con la reducción, se garantiza escalabilidad respecto al tamaño del grafo sobre el que se realiza el análisis.
4. El uso de grafos reducidos facilita la realización de análisis de redes a diferentes escalas. Este tipo de análisis es utilizado en los SIG y de forma general en el proceso de toma de decisiones.
5. Es factible la aplicación del modelo propuesto en los SIG.
6. La generalidad del modelo propuesto radica en la posibilidad de su aplicación en distintos tipos de redes presentes en un mapa, así como en el Diseño Racional en Ingeniería Mecánica.

RECOMENDACIONES

Existen diversas formas de representar las redes existentes en un SIG, así como varios algoritmos para realizar análisis sobre las mismas; es por ello que el presente trabajo no agota este campo de investigación, por lo que se recomienda lo siguiente:

1. Utilizar heurísticas con información de la red (topología, tráfico, GPS, histórico de peticiones de cálculo de caminos óptimos, etc.) para la construcción de la partición que constituye la entrada del algoritmo de reducción.
2. Definir un mecanismo de almacenamiento persistente de grafos (y de grafos reducidos), específico para el análisis de redes.
3. Modificar otros algoritmos de análisis de redes para ser aplicados a grafos reducidos.
4. A partir del modelo propuesto, diseñar estructuras de datos que reduzcan la complejidad espacial de los algoritmos propuestos.

REFERENCIAS BIBLIOGRÁFICAS

[1] Bozkaya, B. GeoLink: An LBS Platform for Mobile Geo-Marketing. In *Esri Business GIS*, San Diego, CA, julio 2010.

[2] Moore, R. GIS Applications for Commercial Real Estate. In *Esri Business GIS*, San Diego, CA, julio 2010.

[3] GIS for Bussiness [en línea]. Disponible en: `http://www.esri.com/industries/business/index.html` [citado 16 de marzo de 2011].

[4] Lamela Fung, L. & Rodríguez Puente, R. Experiencia del desarrollo de un sistema de información geográfica en la Universidad de las Ciencias Informáticas. *Mapping*, (128):76–80, 2008.

[5] UCGIS: University Consortium for Geographic Information Science [en línea]. Disponible en: `http://www.ucgis.org/Default.asp` [citado 16 de marzo de 2011].

[6] GIS at University of Chicago [en línea]. Disponible en: `http://gis.uchicago.edu/` [citado 16 de marzo de 2011].

[7] Arai, C.; Matsuda, N. & Shjkada, M. Management of mapping in local government using remote sensing and the REAL TIME GIS. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS '02)*, volume 6, pages 3145 – 3147, 2002. `doi:10.1109/IGARSS.2002.1027113`.

[8] Weihua, D.; Jiping, L. & Qingsheng, G. Construction of E-Government GIS Based on Net Platform and Web Service. In *IEEE International Conference on Geoscience and Remote Sensing Symposium (IGARSS 2006)*, pages 921 –923, August 2006. doi:10.1109/IGARSS.2006.237.

[9] Lu, X. A Unified E-Government Information Management Platform Based on Web GIS Technology. In *International Conference on Computational Intelligence and Software Engineering*, pages 1 –4, December 2009. doi:10.1109/CISE.2009.5362937.

[10] Lu, X. A GIS-Based Based Integrated Platform for E-Government Application. In *First International Conference on Information Science and Engineering*, pages 1939 –1942, 2009. doi:10.1109/ICISE.2009.41.

[11] Wang, L.; Wang, Y.; Li, Y.; Qiu, A. & Tao, K. Research on Government GIS Construction and Application Technology Based on CNGI. In *International Conference on Web Information Systems and Mining*, volume 2, pages 138 –142, October 2010. doi:10.1109/WISM.2010.130.

[12] Oliva Santos, R.; Maciá Perez, F. & Garea Llano, E. Esbozo de un modelo de integración de datos, metadatos y conocimiento geográfico. In *VII Congreso Internacional Geomática 2011*, La Habana, Febrero 2011.

[13] Gómez Rodríguez, Y. Celebración del GIS Day por primera vez en Cuba. *Geografía y Sistemas de Información Geográfica (GeoSIG)*, (1):1–4, 2009.

[14] Aronoff, S. *Geographical Information Systems: A management perspective*. WDL Publications, Otawa Canadá, 1989.

[15] ESRI Shapefile Technical Description [en línea]. julio 1998. Disponible en: http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf [citado 16 de marzo de 2011].

[16] Daniel, L.; Loree, P. & Whitener, A. *Inside MapInfo Professional*. OnWord Press, Santa Fe, 3 rd edition, 2001.

[17] Well-Known Binary (WKB) Format [en línea]. Disponible en: `http://dev.mysql.com/doc/refman/5.0/en/gis-wkb-format.html` [citado 16 de marzo de 2011].

[18] OpenStreetMap project [en línea]. Disponible en: `http://wiki.openstreetmap.org/wiki/Main_Page` [citado 16 de marzo de 2011].

[19] pgRouting Project [en línea]. Disponible en: `http://www.pgrouting.org/` [citado 16 de marzo de 2011].

[20] PostgreSQL: The world's most advanced open source database [en línea]. Disponible en: `http://www.postgresql.org/` [citado 16 de marzo de 2011].

[21] Campos Gutiérrez, F.; Castro Fernández, J. P. & García Martín, R. IDELabRoute: Librería para la gestión de grafos escalable. In *IV Jornadas SIG Libre*, Girona, Marzo 2010. Disponible en: `http://www.sigte.udg.edu/jornadassiglibre2010/uploads/Articles/a34.pdf`.

[22] pgRouting. Routing from point to point [en línea]. Disponible en: `http://pgrouting.postlbs.org/discussion/topic/353` [citado 16 de marzo de 2011].

[23] Rodríguez Puente, R. & Silverio Castro, R. Enfoque de hipergrafos en Sistemas de Información Geográfica para la modelación de redes. In *VII Congreso Internacional Geomática 2011*, La Habana, Febrero 2011.

[24] ArcGIS for Desktop - Advanced Desktop GIS Mapping | GIS Editing Software [en línea]. Disponible en: `http://www.esri.com/software/arcgis/arcgis-for-desktop` [citado 17 de marzo de 2011].

[25] Gutman, R. J. Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In *6th Workshop on Algorithm Engineering and Experiments*, pages 100–111, New Orleans, Louisiana, enero 2004.

[26] Goldberg, A. V. & Harrelson, C. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165, Vancouver, British Columbia, Canada, enero 2005. Society for Industrial and Applied Mathematics. Disponible en: `http://dl.acm.org/citation.cfm?id=1070432.1070455`.

[27] Koehler, E.; Moehring, R. H. & Schilling, H. Acceleration of Shortest Path and Constrained Shortest Path Computation. *Experimental and Efficient Algorithms*, 3503(1126):126–138, 2005.

[28] Möhring, R. H.; Schilling, H.; Schütz, B.; Wagner, D. & Willhalm, T. Partitioning graphs to speedup Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics*, 11, 2006.

[29] Wagner, D. & Willhalm, T. Speed-Up Techniques for Shortest-Path Computations. In Thomas, W. & Weil, P., editors, *24th International Symposium on Theoretical Aspects of Computer Science*, volume 4393 of *Lecture Notes in Computer Science*, pages 23–36. Springer Berlin / Heidelberg, 2007. `doi:10.1007/978-3-540-70918-3_3`.

[30] Maue, J.; Sanders, P. & Matijevic, D. Goal-directed shortest-path queries using precomputed cluster distances. *ACM Journal of Experimental Algorithmics*, 14(2):3.2–2:3.27, January 2010. `doi:10.1145/1498698.1564502`.

[31] Sanders, P. & Schultes, D. Highway hierarchies hasten exact shortest path queries. In Brodal, G. S. & Leonardi, S., editors, *Proceedings of the 13th annual European conference on Algorithms*, pages 568–579, Palma de Mallorca, España, octubre 2005. Springer-Verlag. `doi:10.1007/11561071_51`.

[32] Gonzalez, H.; Han, J.; Li, X.; Myslinska, M. & Sondag, J. P. Adaptive fastest path computation on a road network: a traffic mining approach. In *Proceedings of the 33rd international conference on Very large data bases*, pages 794–805, Vienna, Austria, septiembre 2007. VLDB Endowment. Disponible en: `http://dl.acm.org/citation.cfm?id=1325851.1325942`.

[33] Geisberger, R.; Sanders, P.; Schultes, D. & Delling, D. Contraction hierarchies: faster and simpler hierarchical routing in road networks. In McGeoch, C. C., editor, *Proceedings of the 7th international conference on Experimental algorithms*, WEA'08, pages 319–333, Provincetown, MA, USA, mayo 2008. Springer-Verlag. Disponible en: `http://dl.acm.org/citation.cfm?id=1788888.1788912`.

[34] Jagadeesh, G. & Srikanthan, T. Route computation in large road networks: a hierarchical approach. *Intelligent Transport Systems, IET*, 2(3):219 –227, septiembre 2008. `doi:10.1049/iet-its:20080012`.

[35] Pfoser, D.; Efentakis, A.; Voisard, A. & Wenk, C. Exploiting road network properties in efficient shortest path computation. Technical report, International Computer Science Institute, July 2009. Disponible en: `http://www.icsi.berkeley.edu/pubs/techreports/TR-09-007.pdf`.

[36] Song, Q. & Wang, X. Efficient Routing on Large Road Networks Using Hierarchical Communities. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):132 –140, marzo 2011. `doi:10.1109/TITS.2010.2072503`.

[37] Serpa, I. M. Los Sistemas de Información Geográfica en Epidemiología. *Salud Pública y Nutrición*, 2(2), 2001.

[38] Martínez-Piedra, R.; Nájera-Aguilar, P.; Vidaurre M, L. E. & Castillo-Salgado, C. SIGEpi: Sistema de Información Geográfica en Epidemiología y Salud Pública. *Boletín Epidemiológico de la Organización Panaméricana de la Salud*, 22(3), 2001.

[39] Pérez Jiménez, D.; Más Bermejo, P.; Prieto Díaz, V. & Rodríguez González, M. Geosalud: relaciones geográficas entre salud y ambiente. *Revista Cubana de Higiene y Epidemiología*, 42(2), 2004. Disponible en: `http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1561-30032004000200003&nrm=iso`.

[40] Fernández Núñez, H. M. SIG-ESAC: Sistema de Información Geográfica para la gestión de la estadística de salud de Cuba. *Revista Cubana de Higiene y Epidemiología*, 44(3),

2006. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1561-30032006000300003.

[41] Primelles Fariñas, J.; Junco Garzón, N. & Reyes Artiles, G. SIG para asistir la estrategia de conservación de las Áreas Protegidas de la provincia de Camaguey, Cuba. *Cuba: Medio Ambiente y Desarrollo*, 2(3), 2002.

[42] Rodríguez Gámez, M.; Mas Ruiz, A.; Destrade Giraudy, A. & Rabelo Parra, L. V. Utilización de un Sistema de Información Geográfica para realizar los estudios de impacto ambiental en la Empresa Eléctrica Provincial de Santiago de Cuba. *Ecosolar*, (24), 2008. Disponible en: http://www.cubasolar.cu/biblioteca/Ecosolar/Ecosolar24/HTML/articulo03.htm.

[43] Pantoja, Y. Plataforma Soberana LiberGIS. *Mapping*, (135):32–34, 2009. Disponible en: http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1604.

[44] Colectivo de autores. Plataforma soberana GeneSIG. Software Reg. 2871-2010, Centro Nacional de Derecho de Autor, Cuba., 2010.

[45] López Costa, A.; León Companioni, A. & Bravo León, L. Sistema de información geográfica para la representación de objetivos petroleros. In *I Taller de Geoinformática*, 2012. Disponible en: http://uciencia.uci.cu/es/node/963.

[46] Velazco Villares, D.; Muñoz Castro, A.; Velazco Villares, P. & Sánchez Mendoza, F. Sistema de información geográfica Rutas SIG, para el cálculo automatizado de tablas de distancias y trazado de rutas. In *VI Congreso Internacional Geomática 2009*, 2009.

[47] Delgado Fernández, T. *Infraestructuras de Datos Espaciales en paises de bajo desarrollo tecnológico. Implementación en Cuba*. Tesis en opción al grado científico de doctor en ciencias técnicas, Instituto Técnico Militar "José Martí", 2005.

[48] Great Britain Committee of Enquiry into the Handling of Geographic Information and Great Britain Dept of the Environment. *Handling geographic information: report to the*

Secretary of State for the Environment of the Committee of Enquiry into the Handling of Geographic Information. Number v. 1. H.M.S.O., 1987. Disponible en: `http://books.google.com/books?id=Q2Eh8Rrfb7UC`.

[49] Clarke, M. Geographical information systems and model based analysis: towards effective decision support systems. In *Proceedings of the GIS Summer Institute Kluwer*, Amsterdam, 1989.

[50] Cowen, D. Lectura en el Centro Nacional de Análisis e Información Geográfica. Universidad de California, 1989.

[51] Delgado Fernández, T. Infraestructura Cubana de Datos Geoespaciales: Una necesidad nacional para la integración y diseminación de datos geoespaciales. In *II Congreso Internacional Geomática 2000*, La Habana, Febrero 2000.

[52] Korte, G. *The Gis Book*. OnWord Press, NY, USA, fifth edition, 2001. Disponible en: `http://books.google.com.cu/books?id=_C6oPvJ5S_EC`.

[53] Makinson, D. *Sets, Logic and Maths for Computing*. Undergraduate Topics in Computer Science. Springer, first edition, 2008.

[54] Tucker, A. B. *Computer Science Handbook, Second Edition*. Chapman & Hall/CRC, second edition, 2004.

[55] Janssens, D. & Rozenberg, G. Graph grammars with neighbourhood-controlled embedding. *Theoretical Computer Science*, 21(1):55 – 74, 1982. `doi:10.1016/0304-3975(82)90088-3`.

[56] Janssens, D. & Rozenberg, G. On the structure of node-label-controlled graph languages. *Information Sciences*, 20(3):191–216, 1980.

[57] Janssens, D. & Rozenberg, G. Graph grammars with node-label controlled rewriting and embedding. In *Proceedings of the 2nd International Workshop on Graph-Grammars and Their Application to Computer Science*, pages 186–205, London, UK, 1983.

Springer-Verlag. Disponible en: http://portal.acm.org/citation.cfm?id=647559.730195.

[58] Rozenberg, G. & Salomaa, A. *Handbook of Formal Languages*, volume 3. Springer, 1997.

[59] Blostein, D.; Fahmy, H. & Grbavec, A. Issues in the Practical Use of Graph Rewriting. In Cuny, J. E.; Ehrig, H.; Engels, G. & Rozenberg, G., editors, *5th International Workshop on Graph Gramars and Their Application to Computer Science*, pages 38–55, London, UK, 1996. Springer-Verlag.

[60] Rodríguez Puente, R. Aplicación de las gramáticas de grafo en Sistemas de Información Geográfica. *Revista Cubana de Ciencias Informáticas*, 4(1/2):5–10, 2010.

[61] Open Geospatial Consortium [en línea]. Disponible en: http://www.opengis.org [citado 7 de marzo de 2011].

[62] Simple Feature Access [en línea]. Disponible en: http://www.opengeospatial.org/standards/sfa [citado 15 de marzo de 2011].

[63] OGC KML [en línea]. abril 2008. Disponible en: http://www.opengeospatial.org/standards/kml/ [citado 3 de marzo de 2011].

[64] OpenGIS Geography Markup Language (GML) Encoding Standard [en línea]. agosto 2007. Disponible en: http://www.opengeospatial.org/standards/gml [citado 3 de marzo de 2011].

[65] Mapinfo Data Interchange Format [en línea]. Disponible en: http://www.gissky.com/Download/Download/DataFormat/Mapinfo_Mif.pdf [citado 15 de marzo de 2011].

[66] Simple Feature Specification [en línea]. Disponible en: http://www.opengeospatial.org/standards/sfs [citado 3 de marzo de 2011].

[67] Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970. doi:10.1145/362384.362685.

[68] Teorey, T. J.; Yang, D. & Fry, J. P. A logical design methodology for relational databases using the extended entity-relationship model. *ACM Computing Surveys*, 18:197–222, June 1986. doi:10.1145/7474.7475.

[69] Navathe, S.; Elmasri, R. & Larson, J. Integrating User Views in Database Design. *Computer*, 19:50–62, January 1986. doi:10.1109/MC.1986.1663033.

[70] Hohenstein, U. & Gogolla, M. A Calculus for an Extended Entity-Relationship Model Incorporating Arbitrary Data Operations and Aggregate Functions. In *Proceedings of the Seventh International Conference on Enity-Relationship Approach: A Bridge to the User*, pages 129–148, Amsterdam, The Netherlands, 1989. North-Holland Publishing Co. Disponible en: http://portal.acm.org/citation.cfm?id=647512.726712.

[71] Czejdo, B.; Elmasri, R.; Rusinkiewicz, M. & Embley, D. W. A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. *Computer*, 23:26–36, March 1990. doi:10.1109/2.50270.

[72] Markowitz, V. M. & Shoshani, A. Representing extended entity-relationship structures in relational databases: a modular approach. *ACM Transactions on Database Systems*, 17:423–464, September 1992. doi:10.1145/132271.132273.

[73] Saiedian, H. Una evaluación del modelo entidad relación extendido. *Information and Software Technology*, 39:449–462, 1997.

[74] Erwig, M. & Güting, R. H. Explicit Graphs in a Functional Model for Spatial Databases. *IEEE Transactions on Knowledge and Data Engineering*, 6:787–804, October 1994. doi:10.1109/69.317707.

[75] Samet, H. & Aref, W. G. Modern database systems. chapter Spatial data models and query processing, pages 338–360. ACM Press/Addison-Wesley Publishing Co., New

York, NY, USA, 1995. Disponible en: http://portal.acm.org/citation.cfm?id=187362.187437.

[76] Paredaens, J. & Kuijpers, B. Data models and query languages for spatial databases. *Data & Knowledge Engineering*, 25(1-2):29–53, 1998. doi:10.1016/S0169-023X(98)00052-4.

[77] Haklay, M. & Weber, P. OpenStreetMap: User-Generated Street Maps. *IEEE Pervasive Computing*, 7:12–18, 2008. doi:10.1109/MPRV.2008.80.

[78] Angles, R. & Gutierrez, C. Survey of graph database models. *ACM Computing Surveys*, 40(1):1:1–1:39, febrero 2008. Disponible en: http://doi.acm.org/10.1145/1322432.1322433.

[79] Vicknair, C.; Macias, M.; Zhao, Z.; Nan, X.; Chen, Y. & Wilkins, D. A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 42:1–42:6, New York, USA, 2010. ACM. doi:10.1145/1900008.1900067.

[80] MySQL :: The world's most popular open source database [en línea]. Disponible en: http://mysql.com/ [citado 7 de marzo de 2011].

[81] Eifrem, E. Neo4j - The Benefits of Graph Databases. In *OSCON 2009, The O'Reilly Open Source Convention*, 2009.

[82] Mei-Po, K.; Golledge, R. G. & Speigle, J. M. A Review of Object-Oriented Approaches in Geographical Information Systems for Transportation Modeling. Technical Report 412, Earlier Faculty Research, University of California Transportation Center, UC Berkeley, 1996.

[83] Khoshafian, S. & Abnous, R. *Object-orientation: Concepts, Languages, Databases, User Interfaces*. New York: Wiley, 1990.

[84] Kim, W. Introduction to Object-oriented Databases. *Cambridge, MA: MIT Press*, 1990.

[85] Gupta, R. & Horowitz, E. *Object-oriented databases with applications to CASE, networks, and VLSI CAD*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1991.

[86] Loomis, M. E. S. Object programming and database management: differences in perspective between the two. *Journal of Object-Oriented Programming*, 6(2), mayo 1993.

[87] Sobre el análisis de redes con jerarquías [en línea]. Disponible en: `http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#/na/004700000057000000/` [citado 17 de marzo de 2011].

[88] Chandrasekhar, T. & Sandhu, J. *ArcGIS 9: ArcGIS Network Analyst Tutorial*. Redlands, CA: ESRI, 2006.

[89] Acevedo, V. gvSIG Portal [en línea]. Disponible en: `http://www.gvsig.org/` [citado 9 de julio de 2011].

[90] Anguix, A. & Díaz, L. gvSIG: A GIS desktop solution for an open SDI. *Journal of Geography and Regional Planing*, 1(3):8, 2008.

[91] Penarrubia, F. J. Plugin de redes – gvSIG [en línea]. Disponible en: `http://www.gvsig.org/web/docdev/docs/desarrollo/plugins/redes/components/core/descripcion/` [citado 7 de junio de 2012].

[92] GRASS GIS - The World Leading Free Software GIS [en línea]. Disponible en: `http://grass.osgeo.org/` [citado 16 de marzo de 2011].

[93] The GNU General Public License [en línea]. Disponible en: `http://www.gnu.org/licenses/gpl.html` [citado 16 de marzo de 2011].

[94] The Open Source Geospatial Foundation [en línea]. Disponible en: `http://www.osgeo.org/` [citado 16 de marzo de 2011].

[95] GRASS Programmer's Manual: Directed Graph Library [en línea]. 2008. Disponible en: `http://grass.osgeo.org/programming6/dglib.html` [citado 16 de marzo de 2011].

[96] Blazek, R.; Neteler, M. & Micarelli, R. The new GRASS 5.1 vector architecture. In *Open source GIS - GRASS users conference*. University of Trento, Italy, 2002. Disponible en: `http://www.ing.unitn.it/~grass/conferences/GRASS2002/proceedings/proceedings/pdfs/Blazek_Radim.pdf`.

[97] El laboratorio IDE Valladolid | IDELab [en línea]. Disponible en: `http://idelab.uva.es/` [citado 16 de marzo de 2011].

[98] Google Maps [en línea]. Disponible en: `http://maps.google.com` [citado 7 de marzo de 2011].

[99] Bing Maps - driving directions, routes, and traffic [en línea]. Disponible en: `http://www.bing.com/maps` [citado 7 de marzo de 2011].

[100] Yahoo Maps [en línea]. Disponible en: `http://maps.yahoo.com` [citado 7 de marzo de 2011].

[101] MapQuest Maps - Driving Directions - Map [en línea]. Disponible en: `http://www.mapquest.com/` [citado 7 de marzo de 2011].

[102] Maps & Mapping softwares: Location based services - Maporama Solutions [en línea]. Disponible en: `http://www.maporama.com/` [citado 7 de marzo de 2011].

[103] Callejero Páginas Amarillas [en línea]. Disponible en: `http://callejero.paginasamarillas.es/vemaps/mapa.asp` [citado 6 de marzo de 2011].

[104] Callejero Lanetro [en línea]. Disponible en: `http://callejero.lanetro.com/` [citado 16 de marzo de 2011].

[105] Callejero, mapas y foto de satélite -elmundo.es- [en línea]. Disponible en: `http://www.elmundo.es/callejero/` [citado 6 de marzo de 2011].

[106] Terra -Callejero- [en línea]. Disponible en: `http://callejero.terra.es/` [citado 16 de marzo de 2011].

[107] Callejeros con el país [en línea]. Disponible en: `http://www.elpais.com/callejero/` [citado 6 de marzo de 2011].

[108] Miranda, J. Cartografía y Bases de Datos Espaciales. In *VII Congreso Internacional de Geomática*, La Habana, febrero 2011.

[109] Wasserman, S. & Faust, K. *Social Network Analysis. Methods and Applications*. Structural Analysis in the Social Sciences(No. 8). Cambridge University Press, 1995. Disponible en: `http://www.cambridge.org/gb/knowledge/isbn/item1138907/?site_locale=en_GB`.

[110] Sedgewick, R. & Wayne, K. *Algorithms*. Addison-Wesley Professional, fourth edition, march 19, 2011.

[111] Borgatti, S. P. & Halgin, D. S. On Network Theory. *Organization Science*, 22(5):1168–1181, 2011. Disponible en: `http://orgsci.journal.informs.org/content/22/5/1168.abstract`, `doi:10.1287/orsc.1100.0641`.

[112] Rodrigue, J.-P.; Comtois, C. & Slack, B. *The Geography of Transport Systems*. Routledge, second edition, 2009. Disponible en: `http://people.hofstra.edu/geotrans/eng/content.html`.

[113] Dijkstra, E. W. A Note on Two Problems in Connection with Graphs. *Numerische Mathematik*, 1:269–271, 1959.

[114] Fuhao, Z. & Jiping, L. An Algorithm of Shortest Path Based on Dijkstra for Huge Data. *Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 4:244–247, 2009. `doi:10.1109/FSKD.2009.848`.

[115] Hu, Y.; Chang, Z.; Sun, L. & Wang, Y. Analysis of the Shortest Repaired Path of Distribution Network Based on Dijkstra Algorithm. *International Conference on Energy and Environment Technology*, 2:73–76, 2009. `doi:10.1109/ICEET.2009.254`.

[116] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L. & Stein, C. *Introduction to algorithms*. MIT electrical engineering and computer science series. The MIT Press, third edition, 2009.

Disponible en: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11866.

[117] Floyd, R. W. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345–, June 1962. doi:10.1145/367766.368168.

[118] Viana, J. & Gentil, F. Um Sistema de Informaçao Geográfico inteligente para escolha de trajetos: uso do modelo de rede e da lógica fuzzy. Master's thesis, CEFET-MG - Belo Horizonte, Brazil, 2004.

[119] da Silva, G. On The Shortest Path Problem: a New Approach with Fuzzy Inference Systems and Conventional Geographic Information Systems. In *International Conference on Intelligent Systems Design and Applications*, pages 427–432, Rio de Janeiro, Brasil, octubre 2007. IEEE Computer Society. doi:10.1109/ISDA.2007.148.

[120] Rodrigues da Silva, E.; de Souza Baptista, C.; Cavalcante de Menezes, L. & Cardoso de Paiva, A. Personalized Path Finding in Road Networks. In *Fourth International Conference on Networked Computing and Advanced Information Management*, volume 2, pages 586–591, Gyeongju, Korea, septiembre 2008. IEEE Computer Society. doi:10.1109/NCM.2008.211.

[121] Shunying, Z.; Yongfei, Y.; Hong, W. & Shangbin, L. An Optimal Transit Path Algorithm Based on the Terminal Walking Time Judgment and Multi-mode Transit Schedules. In *International Conference on Intelligent Computation Technology and Automation*, volume 1, pages 623–627, Changsha, Hunan, China, julio 2010. IEEE Computer Society. doi:10.1109/ICICTA.2010.485.

[122] Xinmiao, Y.; Wei, W. & Wenteng, M. GIS-Based Public Transit Passenger Route Choice Model. *Journal of Southeast University (Natural Science Edition)*, 30:87–91, 2000.

[123] Han-ying, G. & Hong-guo, S. Exploring Adaptability of City Public Traffic in View of Traveler Psychology. *Chinese Journal of Ergonomics*, 12:11–13, 2006.

[124] Lin, C.; Yang, Z. & Gong, B. Multimodal Traffic Information Service System with K-Multimodal Shortest Path Algorithm. In *International Conference on Intelligent Computation Technology and Automation*, volume 3, pages 652–655, Changsha, Hunan, China, octubre 2009. IEEE Computer Society. doi:10.1109/ICICTA.2009.623.

[125] Lozano, A. & Storchi, G. Shortest viable path algorithm in multimodal networks. *Transportation Research Part A: Policy and Practice*, 35(3):225–241, 2001. doi:10.1016/S0965-8564(99)00056-7.

[126] Nazari, S.; Meybodi, M. R.; Salehigh, M. A. & Taghipour, S. An Advanced Algorithm for Finding Shortest Path in Car Navigation System. In *International Workshop on Intelligent Networks and Intelligent Systems*, pages 671–674, Wuhan, Hubei, China, noviembre 2008. IEEE Computer Society. doi:10.1109/ICINIS.2008.147.

[127] Sun, L.; Hu, X.; Li, Y.; Lu, J. & Yang, D. A Heuristic Algorithm and a System for Vehicle Routing with Multiple Destinations in Embedded Equipment. In *International Conference on Mobile Business*, volume 0, pages 1–8, Barcelona, España, julio 2008. IEEE Computer Society. doi:10.1109/ICMB.2008.47.

[128] Diestel, R. *Graph Theory*. Springer-Verlag, 2 edition, 2000. Disponible en: http://www.math.uni-hamburg.de/home/diestel/books/graph.theory/.

[129] Solana González, P.; Alonso Martínez, M. & Pérez González, D. Análisis y modelado con redes de workflow del proceso de tratamiento de experiencias operativas. In *XX Congreso Anual de la Academia Europea de Dirección y Economía de la Empresa*, volume 1, pages 1109–1122, Palma de Mallorca, junio 2007.

[130] Ellis, C. A. & Nutt, G. J. Modelling and Enactment of Workflow Systems. In Marsan, A. M., editor, *Application and Theory of Petri Nets*, volume 691 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, Berlin, 1993.

[131] Van der Aalst, W. M. P. Verification of Workflow Nets. In *Proceedings of the 18th International Conference on Application and Theory of Petri Nets*, pages 407–426,

Toulouse, junio 1997. Springer-Verlag. Disponible en: http://portal.acm.org/citation.cfm?id=647744.733919.

[132] Sadiq, W. & Orlowska, M. E. Analyzing process models using graph reduction techniques. *Information Systems*, 25(2):117–134, April 2000. doi:10.1016/S0306-4379(00)00012-0.

[133] Lu, K. & Liu, Q. An Algorithm Combining Graph-Reduction and Graph-Search for Workflow Graphs Verification. In *11th International Conference on Computer Supported Cooperative Work in Design*, pages 772–776, Melbourne, Australia, abril 2007. doi:10.1109/CSCWD.2007.4281534.

[134] Qing-xiu, L.; Bao-xiang, C. & Yi-wei, Z. An improved verification method for workflow model based on Petri net reduction. In *The 2nd IEEE International Conference on Information Management and Engineering*, pages 252–256, Chengdu, Sichuan, China, abril 2010. doi:10.1109/ICIME.2010.5477436.

[135] Lin, H.; Zhao, Z.; Li, H. & Chen, Z. A novel graph reduction algorithm to identify structural conflicts. In Sprague, R. H., editor, *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, volume 9, pages 289–299, Big Island, Hawaii, enero 2002. doi:10.1109/HICSS.2002.994506.

[136] Pease, M.; Shostak, R. & Lamport, L. Reaching Agreement in the Presence of Faults. *Journal of the ACM*, 27(2):228–234, abril 1980. doi:10.1145/322186.322188.

[137] Olfati-Saber, R.; Fax, J. A. & Murray, R. M. Consensus and Cooperation in Networked Multi-Agent Systems. *Proceedings of the IEEE*, 95(1):215 –233, enero 2007. doi:10.1109/JPROC.2006.887293.

[138] Parlangeli, G. & Notarstefano, G. Graph reduction based observability conditions for network systems running average consensus algorithms. In *18th Mediterranean Conference on Control Automation*, pages 689 –694, Marrakech, Morocco, junio 2010. doi:10.1109/MED.2010.5547789.

[139] Bellman, R. On the theory of dynamic programming. In *Proceedings of the National Academy of Sciences*, number 38, pages 716–719, 1952.

[140] Zachou, V.; Christodoulou, C.; Chryssomallis, M.; Anagnostou, D. & Barbin, S. Planar Monopole Antenna With Attached Sleeves. *IEEE Antennas and Wireless Propagation Letters*, 5(1):286 –289, 2006. doi:10.1109/LAWP.2006.876970.

[141] Costantine, J.; Christodoulou, C. G. & Barbin, S. E. A new reconfigurable multi band patch antenna. In *Microwave and Optoelectronics Conference*, number 1, pages 75–78, Salvador, Brasil, octubre 2007. Institute of Electrical and Electronics Engineers. doi:10.1109/IMOC.2007.4404216.

[142] Costantine, J.; Christodoulou, C.; Abdallah, C. & Barbin, S. Optimization and Complexity Reduction of Switch-Reconfigured Antennas Using Graph Models. *IEEE Antennas and Wireless Propagation Letters*, 8:1072–1075, 2009. doi:10.1109/LAWP.2009.2032674.

[143] Casetti, C.; Lo Cigno, R.; Mellia, M.; Munafo, M. & Zsóka, Z. A new class of QoS routing strategies based on network graph reduction. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 41:475–487, marzo 2003. doi:10.1016/S1389-1286(02)00414-0.

[144] Margarino, A.; Romano, A.; De Gloria, A.; Curatelli, F. & Antognetti, P. A Tile-Expansion Router. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 6, pages 507 – 517, 1987.

[145] Sato, M.; Sakanaka, J. & Ohtsuki, T. A fast line-search method based on a tile plane. In *IEEE International Symposium on Circuits and Systems*, page 588–591, Philadelphia, PA, mayo 1987.

[146] Wu, Y.-F.; Widmayer, P.; Schlag, M. D. F. & Wong, C. K. Rectilinear Shortest Paths and Minimum Spanning Trees in the Presence of Rectilinear Obstacles. *IEEE Transactions on Computers*, 36(3):321–331, marzo 1987. doi:10.1109/TC.1987.1676904.

[147] Clarkson, K.; Kapoor, S. & Vaidya, P. Rectilinear shortest paths through polygonal obstacles in O(n(logn)2) time. In Soule, D., editor, *Proceedings of the third annual symposium on Computational geometry*, pages 251–257, Waterloo, Ontario, Canada, junio 1987. ACM. doi:10.1145/41958.41985.

[148] Zheng, S.-Q.; Lim, J. S. & Iyengar, S. S. Finding obstacle-avoiding shortest paths using implicit connection graphs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):103 – 110, enero 1996. doi:10.1109/43.486276.

[149] Lunow, R. E. A channelless, multilayer router. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*, pages 667–671, Anaheim, CA, USA, junio 1988. IEEE Computer Society Press. Disponible en: http://portal.acm.org/citation.cfm?id=285730.285844.

[150] Cong, J.; Fang, J. & Khoo, K.-Y. An implicit connection graph maze routing algorithm for ECO routing. In *Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 163–167, San Jose, CA, noviembre 1999. IEEE Press. Disponible en: http://portal.acm.org/citation.cfm?id=339492.339614.

[151] Cong, J.; Fang, J. & Khoo, K.-Y. DUNE: a multi-layer gridless routing system with wire planning. In *Proceedings of the 2000 international conference on Computer-aided design*, pages 12–18, San Diego, CA, USA, abril 2000. IEEE Computer Society Press. Disponible en: http://portal.acm.org/citation.cfm?id=371038.371057.

[152] Xing, Z. & Kao, R. Shortest path search using tiles and piecewise linear cost propagation. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, volume 21, pages 145 – 158, 2002. doi:10.1109/43.980255.

[153] Li, Y.-L.; Li, J.-Y. & Chen, W.-B. An Efficient Tile-Based ECO Router Using Routing Graph Reduction and Enhanced Global Routing Flow. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(2):345–358, 2007. doi:10.1109/TCAD.2006.883923.

[154] Valmari, A. The State Explosion Problem. In Reisig, W. & Rozenberg, G., editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*, volume 1491, pages 429–528. Springer-Verlag, 1998. Disponible en: http://portal.acm.org/citation.cfm?id=647444.727054.

[155] Peled, D. All from One, One for All: on Model Checking Using Representatives. In *Proceedings of the 5th International Conference on Computer Aided Verification*, pages 409–423, Elounda, Greece, junio 1993. Springer-Verlag. Disponible en: http://portal.acm.org/citation.cfm?id=647762.735490.

[156] Barnat, J.; Brim, L. & Rockai, P. Parallel Partial Order Reduction with Topological Sort Proviso. In *8th IEEE International Conference on Software Engineering and Formal Methods*, pages 222–231, Pisa, Italia, Septiembre 2010. doi:10.1109/SEFM.2010.35.

[157] Cao, J.; Cai, S. & Zhao, Y. A Distributed Asynchronous Constraint Optimization Algorithm Based on Dynamic Reduction of Constraint Graph. In Huang, X.-J., editor, *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 365 –369, Toronto, AB, Canada, agosto 2010. Institute of Electrical and Electronics Engineers. doi:10.1109/WI-IAT.2010.24.

[158] Zhang, L.; He, G. & Lei, D. An Outlying Reduction Algorithm Based on Power Graph. In Qiu, R., editor, *WASE International Conference on Information Engineering*, volume 2, pages 192–195, BeiDai, China, agosto 2010. Institute of Electrical and Electronics Engineers. doi:10.1109/ICIE.2010.141.

[159] Lin, P.-M. A survey of applications of symbolic network functions. *IEEE Transactions on Circuit Theory*, 20(6):732 – 737, November 1973. doi:10.1109/TCT.1973.1083770.

[160] Fernández, F.; Rodríguez-Vázquez, A.; Huertas, J. L. & Gielen, G. G. E. *Symbolic Analysis Techniques – Applications to Analog Design Automation*. Wiley-IEEE Press, 1 edition, December 1997.

[161] Guerra, O.; Roca, E.; Fernández, F. V. & Rodríguez-Vázquez, A. Approximate Symbolic Analysis of Hierarchically Decomposed Analog Circuits. *Analog Integrated Circuits and Signal Processing*, 31:131–145, April 2002. doi:10.1023/A:1015094011107.

[162] Shi, G.; Chen, W. & Richard Shi, C.-J. A Graph Reduction Approach to Symbolic Circuit Analysis. In *Asia and South Pacific Design Automation Conference*, pages 197–202, Yokohama, Japan, enero 2007. IEEE Computer Society. doi:10.1109/ASPDAC.2007.357985.

[163] Rinehart, M. & Dahleh, M. A. A graph reduction for bounding the value of side information in shortest path optimization. In *American Control Conference*, pages 4078–4083, Baltimore, MD, USA, junio 2010.

[164] Cellier, F. E. *Continuous System Modeling*. Springer-Verlag, Heidelberg, Berlin, 1991.

[165] de Berg, M.; Cheong, O.; van Krefeld, M. & Overmars, M. *Computational Geometry: Algorithms and Applications, Third Edition*. Springer, 3rd edition, 2008. Disponible en: http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-77973-5.

[166] Rodríguez Puente, R. & Lazo Cortés, M. S. (por aparecer). Graph-reduction algorithm for finding shortest path in Geographic Information Systems. *IEEE Latin American Transactions*, 2012.

[167] Hoare, C. A. R. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–580, October 1969. doi:http://doi.acm.org/10.1145/363235.363259.

[168] Hagberg, A. A.; Schult, D. A. & Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In Varoquaux, G.; Vaught, T. & Millman, J., editors,

Proceedings of the 7th Python in Science Conference (SciPy2008), pages 11–15, Pasadena, CA, USA, agosto 2008.

[169] Valle Martínez, Y. *Modelo de representación de superficies de terreno para su visualización en tres dimensiones*. Tesis en opción al grado científico de doctor en ciencias técnicas, Universidad de las Ciencias Informáticas, 2012.

[170] Inda Herrera, J. A.; Rodríguez Puente, R. & Lazo Cortés, M. Sistema para la búsqueda de caminos mínimos en grafos grandes. In *I Taller de Geoninformática, UCIENCIA*, 2012. Disponible en: `http://uciencia.uci.cu/es/node/316`.

[171] Martinez Escanaverino, J.; García Toll, A. & Ortiz Cárdenas, T. Algorítmica del diseño mecánico. *Ingeniería Mecánica*, 0:31–37, 1997.

[172] Martinez Escanaverino, J. Dichromatic Graphs: A Tool for the Algorithmic Education of Mechanical Engineers. In *ASME 2000 Design Engineering Technical Conferences & Computers and Information in Engineering, DETC 2000*, Baltimore, Maryland, septiembre 2000. American Society of Mechanical Engineers.

[173] Martinez Escanaverino, J.; Llamos Soriz, J.; Garcia Toll, A. & Ortiz Cardenas, T. Rational design automation by dichromatic graphs. In *ASME 2001 Design Engineering Technical Conferences and Computers and Information in Engineering, DETC'01*, Pitsburgh, Pennsylvania, septiembre 2001. American Society of Mechanical Engineers.

[174] Martinez Escanaverino, J. & Martinez Fonte, L. A problem solving rationale for conceptual design in engineering. In Pan, Y.; Vergeest, J.; Lin, Z.; Wang, C.; Sun, S.; Hu, Z. & Tang, Y., editors, *Proceedings of the 6th International Conference on Computer-Aided Industrial Design & Conceptual Design - CAID&CD 2005*, pages 130–134, Delft, The Netherlands, mayo 2005.

[175] Llamos Soríz, J. *Diseño óptimo de cajas reductoras para molinos de caña de azúcar*. Tesis en opción al grado científico de doctor en ciencias técnicas, Departamento de Mecánica Aplicada, Facultad de Ingeniería Mecánica, Instituto Superior Politécnico "José

Antonio Echeverría", 2000. Disponible en: http://biblioteca.cujae.edu.cu/Tesis/Doctorado%5CTesis8.pdf.

[176] Alemán Romero, I. *Modelo Matemático y Algoritmos para la Aplicación 3D del Método de los Elementos de Contorno*. Tesis en opción al grado científico de doctor en ciencias técnicas, Departamento de Matemática, Facultad de Ingeniería Mecánica, Instituto Superior Politécnico "José Antonio Echeverría", 2003. Disponible en: http://biblioteca.cujae.edu.cu/Tesis/Doctorado%5CTesis49.pdf.

[177] Morejón Vizcaino, G. *Primera etepa del proceso de desarrollo de un hidromotor de alto par: conceptualización y prototipo analítico*. Tesis en opción al grado científico de doctor en ciencias técnicas, Departamento de Mecánica Aplicada, Facultad de Ingeniería Mecánica, Instituto Superior Politécnico "José Antonio Echeverría", 2004. Disponible en: http://biblioteca.cujae.edu.cu/Tesis/Doctorado%5CTesis53.pdf.

[178] Marrero Osorio, S. *Diseño Paramétrico Basado en Modelos Matemáticos. Caso de estudio: Máquinas para la Construcción Sostenible de Viviendas*. Tesis en opción al grado científico de doctor en ciencias técnicas, Departamento de Mecánica Aplicada, Facultad de Ingeniería Mecánica, Instituto Superior Politécnico "José Antonio Echeverría", 2009. Disponible en: http://biblioteca.cujae.edu.cu/Tesis/Doctorado/Tesis185.pdf.

[179] González Rey, G. Establecimiento del cálculo del diámetro de cresta exterior de un engranaje cónico con técnicas de grafos. *Ingeniería Mecánica*, 10(3):33–39, 2007.

[180] Marrero Osorio, S. & Martinez Escanaverino, J. Diseño paramétrico de pinzas de fricción. *Ingeniería Mecánica*, 12(1):37–48, 2009.

[181] Rodríguez-Puente, R.; Marrero-Osorio, S. & Lazo-Cortés, M. Aplicación de un algoritmo de reducción de grafos al Método de los Grafos Dicromáticos. *Ingeniería Mecánica*, 15(2):158–169, 2012. Disponible en: http://www.ingenieriamecanica.cujae.edu.cu/index.php/revistaim/article/view/426.

PRODUCCIÓN CIENTÍFICA DEL AUTOR

Publicaciones en revistas

1. Lamela Fung, L.R; Rodríguez Puente R.(2008). Experiencia del desarrollo de un sistema de información geográfica en la Universidad de las Ciencias Informáticas. Mapping [4]. No. 128. España. pp. 76-80. Referenciada por: Latindex, ICYT, Dialnet.
2. Rodríguez Torres, A.; Rodríguez Puente R. (2009). Bases para la creación de un servicio web de mapas temáticos dinámicos. Mapping. No. 137. España. pp. 96-97. Referenciada por: Latindex, ICYT, Dialnet.
3. Rodríguez Torres, A.; Rodríguez Puente R. (2010). Servicio de mapas temáticos. Mapping. No. 139. España. pp. 36-39. Referenciada por: Latindex, ICYT, Dialnet.
4. Rodríguez Puente R. (2010). Aplicación de las gramáticas de grafo en Sistemas de Información Geográfica. Revista Cubana de Ciencias Informáticas (RCCI). 4(1/2). Cuba. pp. 5-10. Referenciada por: EBSCO Academic Premier.
5. Inda Herrera, J.; González Pérez, Y.; Rodríguez Puente, R. (2011). Sistema de reducción de grafos para la búsqueda de camino mínimo. Serie Científica, Vol. 4. No. 6. Cuba.
6. Rodríguez Puente R.; Inda Herrera, J.A.; González Pérez, Y. (2011). Representación de una red de una cartografía mediante un grafo. Mapping. No. 150. España. pp. 48-53. Referenciada por: Latindex, ICYT, Dialnet.
7. Rodríguez Puente, R.; Lazo Cortés, M.S. (**Aceptado**). Graph-reduction algorithm for finding shortest path in Geographic Information Systems. IEEE Latin American Transactions. Referenciada por: ISI, SCOPUS.

[4]Revista aceptada como requisito de publicaciones para las tesis presentadas en opción al grado científico de Doctor en Ciencias Geográficas.

8. Rodríguez Puente, R.; Lazo Cortés, M.S. (**Enviado**). Modelo para la representación de redes en Sistemas de Información Geográfica mediante el uso de transformaciones de grafos. Ingeniare. Revista chilena de ingeniería. Referenciada por: SCIELO, Directory of Open Access Journals.
9. Rodríguez Puente, R.; Marrero Osorio, S.A., Lazo Cortés, M.S. (2012). Aplicación de la reducción de grafos al Diseño Racional en Ingeniería Mecánica. Ingeniería Mecánica. Vol. 15. No 2. Cuba. pp. 158-169. Referenciada por: SCIELO.
10. Santana Puentes, L., Rodríguez Gregorio, A. R., Rodríguez Puente, R. (2012). Propuesta de software libre para la implementación de cálculo de rutas en Sistemas de Información Geográfica. Serie Científica. Vol. 5. No. 7. Cuba.

Publicaciones en memorias de eventos

1. Rodríguez Puente, R. (2006). Algoritmo de búsqueda de caminos mínimos sobre mapas extensos. I Taller de Programación, UCIENCIA 2006. ISBN: 959-16-0463-7.
2. Rodríguez Puente, R. (2006). Inferencia de gramáticas de grafo. I Taller de Programación, UCIENCIA 2006. ISBN: 959-16-0463-7.
3. Rodríguez Puente, R.; Silverio Castro, R. (2011). Enfoque de hipergrafo en Sistemas de Información Geográfica para la modelación de redes. VII Congreso Internacional Geomática 2011, INFORMÁTICA'2011. ISBN: 978-959-7213-01-7.
4. Cobo Rodríguez, J.A.; Rodríguez Puente, R. (2011). Web Semántica: Aproximación a la nueva era del comercio electrónico. IV Taller Internacional de Comercio Electrónico, INFORMÁTICA'2011. ISBN: 978-959-7213-01-7.
5. Hidalgo Delgado, Y.; Rodríguez Puente, R. (2012): La web semántica: una breve revisión. II Taller de Estudios y Tecnologías de Internet, UCIENCIA 2012. ISBN: 978-959-286-019-3.
6. Inda Herrera, J.A.; Rodríguez Puente, R.; Lazo Cortés, M. (2012): Sistema para la búsqueda de caminos mínimos en grafos grandes. I Taller de Geoinformática, UCIENCIA 2012. ISBN: 978-959-286-019-3.
7. Rodríguez Torres, A.; Rodríguez Puente, R. (2012): Especificación de implementación de un servicio de mapas temáticos. I Taller de Geoinformática, UCIENCIA 2012. ISBN:

978-959-286-019-3.

8. Rodríguez Larrazabal, Y.; Rodríguez Puente, R.; Jiménez Moya, G.E. (2012); Sistema de información al viajero: Aplicaciones actuales. I Taller de Geoinformática, UCIENCIA 2012. ISBN: 978-959-286-019-3.

Otros:

1. Mención en el Fórum de Ciencia y Técnica a nivel de universidad. Los Sistemas de Información Geográfica. Alternativas de desarrollo y puesta en marcha para la Universidad de las Ciencias Informáticas. Universidad de las Ciencias Informáticas. Julio de 2005.
2. Relevante en el Fórum de Ciencia y Técnica a nivel de facultad. Plataforma para el desarrollo de aplicaciones GIS. Universidad de las Ciencias Informáticas. Junio de 2008.
3. Destacado en el Fórum de Ciencia y Técnica a nivel de facultad. Módulo para representar automáticamente un mapa en una estructura de datos. Universidad de las Ciencias Informáticas. Junio de 2007.
4. Destacado en el Fórum de Ciencia y Técnica a nivel municipal. Búsqueda de caminos mínimos en Sistemas de Información Geográfica. Universidad de las Ciencias Informáticas. Julio de 2011.

GLOSARIO DE TÉRMINOS

ACM Association for Computing Machinery

IDERC Infraestructura de Datos Espaciales de la República de Cuba

API Interfaz de Programación de Aplicaciones (del inglés Application Programming Interface)

ESRI Enviromental Systems Research Institute

GML Geographic Markup Language

GPL Licencia Pública General de GNU (del inglés GNU General Public License, referenciada comúnmente como GNU GPL)

GPLv2 Licencia Pública General de GNU versión 2 (del inglés GNU General Public License version 2, referenciada comúnmente como GNU GPLv2)

GPS Sistema de Posicionamiento Global (del inglés Global Positioning System)

GRASS Geographic Resources Analysis Support System

IEEE Instituto de Ingenieros Eléctricos y Electrónicos (del inglés Institute of Electrical and Electronics Engineers)

KML Keyhole Markup Language

OGC Open Geospatial Consortium

QGIS Quantum GIS

RAM Memoria de Acceso Aleatorio (del inglés Random-access memory)

MGD Método de los Grafos Dicromáticos

PC Computadora Personal (del inglés Personal Computer)

SFS Simple Feature Specification for SQL

SGBD Sistema Gestor de Bases de Datos

SIG Sistemas de Información Geoespacial (también conocido por sus siglas en inglés GIS de Geographic Information Systems)

TAB Mapinfo Data Interchange Format

TDA Tipo de Dato Abstracto (también conocido por sus siglas en inglés ADT de Abstract data type)

WKB Well-known binary

WKT Well-known text

WWW World Wide Web

A. TIPO DE DATO ABSTRACTO MULTIDIGRAFO

El TDA MultiDiGrafo, implementado en la biblioteca de clases NetworkX, representa un grafo dirigido acorde a la definición utilizada en este trabajo.

```
TDA MultiDiGrafo
  Operaciones:
    AdicionarVertice(v[, diccionario_de_atributos])
    AdicionarArista(u, v[, llave, diccionario_de_atributos])
    AdicionarArista(arista) {arista es una tupla de la forma
                        (u, v[, llave, diccionario_de_atributos])}
    EliminarVertice(v)
    EliminarArista(u, v[, llave])
    ExisteVertice(v)
    ExisteArista(u, v[, llave])
    Adyacentes(v) {Retorna la lista de adyacentes al vértice v}
    Vertices() {Retorna la lista de vértices del grafo}
    Aristas() {Retorna la lista de aristas del grafo}
```

B. TIPO DE DATO ABSTRACTO GRAFOREDUCIDO

El TDA GrafoReducido representa un grafo acorde a la Definición 2.2.2 enunciada en este trabajo.

```
TDA GrafoReducido: MultiDiGrafo
  Operaciones:
    ObtenerGrafoNivel(nivel) {Retorna un grafo con el nivel de
                              reducción especificado}
    ExpandirE(v) {Expande vértices reducidos hasta que v
                  pertenezca al grafo}
    Expandir(v) {Expande el vértice reducido v}
    ObtenerVerticeReducido(v) {Retorna el vértice reducido
                               relacionado con v}
    MDijkstra(u,v)
    Camino(u,v,Pr) {Retorna el camino desde u hasta v
                    utilizando el vector de predecesores Pr}
    f(u,v,w) {Retorna el costo de ir de u a w pasando por v}
```

C. ALGORITMO DE DIJKSTRA

Algoritmo 14 $AlgoritmoDijkstra$

Entrada: Un grafo ponderado $G = (V, E, f)$ y un vértice de origen v_{origen}

Salida: Vector D_n de distancias mínimas, vector P de predecesores

```
1:  C_n = {}
2:  cola = ColaPrioridad() {Almacena tuplas de la forma (Llave,Valor)}
3:  Para todo v ∈ V hacer
4:      D_n[v] = f_c(v_origen, v), P_n[v] = v_origen
5:      Adicionar(cola, D_n[v], v)
6:  Fin Para
7:  Mientras !Vacia(cola) hacer
8:      w_n = Extraer(cola)
9:      C_n = C_n ∪ {w_n} {Actualizar vértices visitados}
10:     Para todo v ∈ Adyacentes(w_n) hacer
11:         Si D_n[v] > D_n[w_n] + f_c(w_n, v) entonces
12:             D_n[v] = D_n[w_n] + f_c(w_n, v), P_n[v] = w_n
13:             DecrementarLlave(cola, D_n[v], v) {Si D_n[v] es menor, modificar valor}
14:         Fin Si
15:     Fin Para
16: Fin Mientras
```

Complejidad temporal

Sea $G = (V, E)$ un grafo y:

- $m = |E|$ la cantidad de aristas
- $n = |V|$ la cantidad de vértices

La complejidad temporal de este algoritmo está muy asociada a la implementación de cola con prioridad que se utilice. En la tabla C.1 se muestra una comparación de la complejidad temporal

de las implementaciones de un heap binario y un heap de Fibonacci.

Tabla C.1: *Complejidad temporal de dos implementaciones de un Heap. Tomado de [116].*

Operaciones	Heap binario	Heap de Fibonacci
Make-Heap	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(\lg(n))$	$\Theta(1)$
Minimum	$\Theta(1)$	$\Theta(1)$
Extract-Min	$\Theta(\lg(n))$	$\Theta(\lg(n))$
Union	$\Theta(n)$	$\Theta(1)$
Decrease-Key	$\Theta(\lg(n))$	$\Theta(1)$
Delete	$\Theta(\lg(n))$	$\Theta(\lg(n))$

Suponiendo que se utiliza implementación con un heap binario, la instrucción $w_n = cola.extraer()$ (paso 8) tiene complejidad $O(\lg(n))$; la misma se realiza, a lo sumo, una vez por cada vértice del grafo (n veces). La instrucción $DecrementarLlave(cola, D_n[v], v)$ (paso 13) tiene complejidad $O(\lg(n))$; la misma se realiza, a lo sumo, una vez por cada arista del grafo (m veces) dado que un vértice se visita solo una vez durante la ejecución del algoritmo.

De esta forma, la complejidad temporal del algoritmo está dada por $O(m\lg(n)+n\lg(n))$, por lo que la complejidad final es $O(m\lg(n))$, siempre que la cantidad de vértices del grafo sea menor que la cantidad de aristas.

Si se utiliza la implementación de cola con prioridad con un heap de Fibonacci, la complejidad del algoritmo es $O(n\lg(n))$. Esta implementación es más eficiente, a pesar de tener el mismo orden que en el caso anterior, teniendo en cuenta la cantidad de operaciones que realiza. Por ejemplo, la instrucción $DecrementarLlave(cola, D_n[v], v)$, con un heap de Fibonacci, tiene complejidad $O(1)$.

De acuerdo a lo anterior, se propone que en la implementación del algoritmo $MDijkstra$ se utilice el TDA cola con prioridad, haciendo uso del heap de Fibonacci.