SOFTWARE PARA LA BÚSQUEDA DE ESPACIOS DISCRETOS QUE SE APROXIMEN AL ESPACIO CONTINUO EUCLIDIANO

Trabajo de Investigación

PABLO ANDRÉS VÉLEZ ARIAS
201010202
pablo.velez@correounivalle.edu.co

Ángel García Baños
Ph. D.
angel.garcia@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Cali, Mayo 24 de 2013

A mi madre y a mi padre por sus consejos, valores, motivación constante y apoyo incondicional, que me han permitido ser una persona de bien, pero más que nada por su infinito amor.

A mi hermano de quien aprendí aciertos en momentos difíciles; a mis familiares por permitirme ser ejemplo de superación brindándome su colaboración.

A mis amigos quienes nos apoyamos mutuamente en nuestra formación profesional y compartir los buenos y malos momentos

Pablo Andrés Vélez

AGRADECIMIENTOS

Son muchas las personas que han formado parte de mi vida profesional y a las que me encantaría en este momento agradecerles por sus consejos, apoyo, ánimo y compañía. Algunos están aquí y otros en lo más profundo de mi corazón, sin importar donde estén quiero darles las gracias por formar parte de mi vida.

Agradezco a mi Director de tesis Doctor Ángel García Baños y demás profesores quienes con su experiencia, paciencia, motivación y dedicación, han logrado desarrollar en mí, un profesional con una visión íntegra con capacidad para enfrentar situaciones propias del ejercicio profesional y una amplia fundamentación y visión para aportar al desarrollo de las organizaciones.

Tabla de contenido

Índice de Tablas

Índice de Figuras

Índice de Gráficas

RESUMEN

Creemos tener cierta intuición de lo que son el espacio y el tiempo, pero realmente nos resulta muy difícil dar una definición precisa de estos. Actualmente se proponen diversas teorías del espacio-tiempo, algunas son muy prometedoras y otras muy contradictorias. Por un lado, tenemos la teoría de la relatividad general que representa el espacio-tiempo de forma continua, mientras que las teorías de campo cuántico necesitan que el espacio-tiempo esté modelado de forma discreta. Unificar estas dos teorías en una teoría de la gravedad cuántica es actualmente uno de los mayores problemas sin resolver de la física.

Se han usado autómatas celulares, que son modelos de espacio discreto, para modelar el espacio en el que vivimos, pero a pesar de los resultados interesantes obtenidos, adolecen de un problema fundamental: en ellos no se cumple el teorema de Pitágoras. En este trabajo se generarán distintos tipos de modelos de espacio discreto haciendo uso de grafos, generados de forma determinística y probabilística, y al final se realizará una comparación entre ellos, buscando cuál de los dos cumple mejor el teorema de Pitágoras.

Palabras Clave: Espacio Discreto, Pitágoras, Relatividad General, Gravedad Cuántica, Algoritmo Genético, Grafos.

ABSTRACT

We have some intuition of what are space and time, but it is very difficult to give a precise definition of these. Currently, physics proposed space-time theories, some are very promising and some very contradictory. On one hand, the theory of general relativity representing space-time continuously, while quantum field theories require that space-time is modeled discretely. Unifying these two theories in a theory of quantum gravity is currently one of the major unsolved problems in physics.

Cellular automata were used, which are discrete space models to model the space in which we live, but despite the interesting results obtained, they have a fundamental problem, in them is not satisfied the Pythagorean theorem. In this paper we generate different types of discrete space models using graphs generated by deterministic and probabilistic manner, the end will be a comparison between them, looking for which one best meets the Pythagorean Theorem.

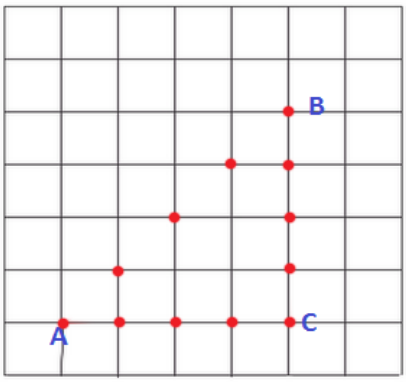
Key Words: Pythagorean theorem, genetic algorithm, graph, space-time.

1. INTRODUCCIÓN

La naturaleza discreta del espacio tiempo ha sido propuesta en diferentes áreas de la física, sin embargo, los modelos propuestos se siguen construyendo en el marco de las matemáticas continuas. Teorías como la mecánica cuántica y la teoría de cuerdas se aplican sobre el concepto de espacio discreto. Esta premisa ha sido de gran motivación para formular teorías que intentan comprobar si nuestro universo es una simulación; la teoría de la Relatividad General propuesta por Albert Einstein, describe el espacio-tiempo como un tejido continuo, explicando por qué la masa de un cuerpo produce una curvatura del espacio.

Considere el siguiente espacio y el triángulo rectángulo formado por los puntos entre los vértices A, B y C, Figura 1. En este modelo la unidad mínima indivisible de espacio es la distancia entre dos puntos, una unidad. Basándonos en este criterio para medir las distancias entre los vértices del triángulo, obtenemos lo siguiente: la distancia entre el cateto AC es cuatro, la distancia entre el cateto CB es cuatro y, por último, la hipotenusa AB es también cuatro. Lo anterior demuestra que el teorema de Pitágoras no se cumple en este espacio, un espacio discreto.

Figura 1. Modelo de espacio discreto



Por otro lado, existen modelos de espacio-tiempo inspirados en los autómatas celulares (CALM, Cell Automata Like Model). Los CALM poseen algunas características interesantes que podrían ser útiles para empezar la construcción de un nuevo paradigma digital: el espacio es discreto y formado por unidades mínimas indivisibles; una unidad de espacio (celda) puede tener uno o varios estados diferentes; cada estado es definido por un conjunto de variables; el tiempo transcurre en una unidad mínima e indivisible de tiempo; el estado cada celda se actualiza de acuerdo a las reglas de transición establecidas y de manera sincronizada.

Sin embargo no se sabe con certeza si los CALM son adecuados para representar un modelo del espacio discreto, ya que adolecen de un problema fundamental, en ellos tampoco se cumple el teorema de Pitágoras.

2. OBJETIVO GENERAL

Determinar por medio de grafos (generados por algoritmos genéticos y algoritmos probabilísticos) una aproximación del espacio discreto al espacio continuo euclidiano usando como función objetivo el teorema de Pitágoras.

2.1 Objetivos Específicos

- Desarrollar una estructura de datos general que sirva para modelar grafos.
- Desarrollar un API que opere sobre esta estructura de datos, para calcular distancias, subespacios, puntos medios y perpendiculares.
- Seleccionar o desarrollar un algoritmo genético que codifique de diversas maneras esta estructura de datos (familias).
- Determinar y comparar la familia de grafos que se aproxime mejor al modelo de espacio discreto minimizando el error del teorema de Pitágoras.

2.2 Resultados Esperados

Los resultados de este trabajo de grado se especifican en la siguiente tabla:

Tabla 1. Objetivos específicos y resultados esperados

Objetivo Específico	Producto(s) Esperado(s)	Sección
Desarrollar una estructura de datos general que sirva para modelar grafos.	Estructura de datos que se pueda usar para el modelamiento de grafos usando algoritmos genéticos.	Sección 5.1
Desarrollar un API que opere sobre esta estructura de datos, para calcular distancias, subespacios, puntos medios y perpendiculares.	Funciones que permitan operar sobre los grafos.	Sección 5.1
Seleccionar o desarrollar un algoritmo genético que codifique de diversas maneras esta estructura de datos (familias).	Algoritmo genético que creará diferentes topologías de grafos.	Sección 5.3
Determinar y comparar la familia de grafos que mejor se aproxime al modelo de espacio discreto minimizando el error del teorema de Pitágoras.	Parámetros de la familia que hacen que se cumpla mejor el teorema de Pitágoras.	Sección 5.4 Sección 5.5

3. ESPACIO-TIEMPO Y ALGORITMOS GENÉTICOS

Percibimos el espacio y el tiempo como si fueran continuos, pero si la teoría de la gravedad cuántica de bucles fuera correcta, estarían formados por elementos discretos.[1]

La teoría general de la relatividad general y la mecánica cuántica se formularon en los primeros años del siglo XX. Estas dos teorías, han permitido llevar a cabo diversos experimentos en un intento de explicar nuestro mundo físico, sin embargo, cuando se aplica una teoría en el campo de la otra existe un problema conceptual que impide su combinación, la teoría general de la relatividad es una teoría clásica, diferente por completo a la mecánica cuántica.

3.1 Gravedad Cuántica

La gravedad cuántica de bucles, formulada por Abhay Ashtekar en 1986 [2], sugiere que el espacio-tiempo sea discreto y no continuo. Esta nueva teoría sería capaz de combinar la teoría general de la relatividad y la mecánica cuántica. En ésta teoría, el espacio se modela como una fina red tejida con un número finito de lazos o bucles. Las distancias entre los lazos están cuantizadas y su unidad mínima indivisible es la distancia de Planck, 10^{-33} centímetros. El tiempo también avanza de forma discreta y su valor es el tiempo de Planck, el tiempo que tarda un fotón viajando a la velocidad de la luz en recorrer la longitud de Planck.

Según la física Fotini Markopoulo Kalamara del Perimeter Institute for Theoretical Physics [4], la teoría de la relatividad de Einstein no puede ser completamente exacta porque carece de una teoría cuántica subyacente, es decir, cuando revertimos el proceso de convertirla en una teoría cuántica volvemos a obtener la

teoría de relatividad general. Este problema se debe a que la teoría de la relatividad general utiliza variables continuas, mientras que la mecánica cuántica emplea variables discretas, y es por esta razón que ambas teorías no sean totalmente compatibles.

Con estos nuevos enfoques a la gravedad cuántica, las grandes preguntas sobre el origen del universo y la posibilidad de que el tiempo y el espacio sean considerados como partículas están siendo seriamente consideradas por los físicos modernos.

3.2 Teoría de Cuerdas

La teoría de cuerdas ha surgido como un candidato prometedor para la teoría microscópica de la gravedad, y pretende ser un modelo que conduzca a una teoría del todo que unifique las cuatro fuerzas fundamentales y toda la materia conocida del universo.

Esta teoría fundamenta las partículas del modelo estándar en un objeto básico, una cuerda. Nuestro concepto básico de física, nos dice que los objetos se encuentran constituidos por átomos, y estos a su vez contienen otras partículas subatómicas. Imaginemos, por ejemplo, un electrón como un punto en el espacio, es una partícula que sólo puede moverse, pero si observamos en una escala muy pequeña, nos daremos cuenta que el electrón no es un punto sino un lazo (cuerda) que oscila de diferentes formas. Dependiendo de la frecuencia de vibración de la cuerda podemos percibir un electrón, un fotón o cualquier otra partícula fundamental del modelo estándar.

Sin embargo, esta teoría requiere de la existencia de hasta 12 dimensiones espaciales y aún no se ha encontrado evidencia experimental que demuestre que así se describe el mundo que nos rodea, dado que este modelo se encuentra en su etapa de desarrollo.

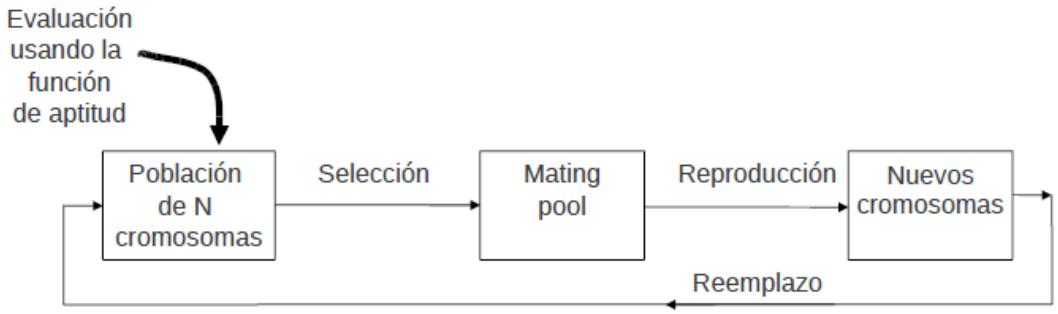
3.4. Algoritmos Genéticos

Los algoritmos genéticos (AGs) son algoritmos de búsqueda heurística que imitan el proceso de selección natural. La esencia de los AGs es simular los procesos necesarios de un sistema natural para su evolución, específicamente aquellos que siguen los principios básicos establecidos por Charles Darwin de la supervivencia del individuo más apto.

Inicialmente, se cuenta con un conjunto de soluciones (representado por cromosomas) llamado población. Las soluciones de la población inicial se seleccionan en función de su aptitud, los más aptos son los que más posibilidades tienen de reproducirse, para formar una nueva población por medio de operadores de reproducción. La idea es que la nueva población (descendencia) será mejor que la anterior.

Este proceso se repite hasta que se cumpla alguna condición, por ejemplo, alcanzar un número determinado en la población o encontrar la mejor de las mejores soluciones.

Figura 2. Esquema de un algoritmo genético



Un algoritmo genético requiere, para su correcto funcionamiento, los siguientes elementos:

- *Una representación genética del dominio de soluciones (cromosomas*). Una representación estándar de una solución es un conjunto de parámetros de tamaño fijo, a este conjunto se le llama cromosoma y los parámetros son denominados genes. La característica principal para que esta representación sea conveniente, es que sus partes se alinean fácilmente debido a su tamaño fijo, facilitando las operaciones de cruce.
- *Una función de aptitud para evaluar el dominio de la solución.* La función de aptitud se define sobre la representación del cromosoma y mide la calidad de la solución representada. La función de aptitud siempre depende del problema que queramos abordar.

Algunas de las características de los algoritmos genéticos son:

- Son algoritmos estocásticos. Dos ejecuciones distintas pueden dar dos soluciones distintas.
- Dado que tienen elementos aleatorios, las soluciones encontradas no son exactas, son aproximadas.
- Su búsqueda es paramétricamente robusta. Esto quiere decir que no es necesario escoger bien los parámetros iníciales del algoritmo para que este converja a una solución óptima.

3.4.1 Población

Tamaño de la Población

El tamaño de la población depende de la naturaleza del problema, ya que si generamos una población con un tamaño pequeño corremos el riesgo de no cubrir adecuadamente el espacio de búsqueda de soluciones, por otro lado, si generamos una población demasiado grande tendremos problemas relacionados con el costo computacional.

Inicialización

Habitualmente las soluciones individuales son generadas aleatoriamente para formar una población inicial. Podríamos pensar en generar soluciones obtenidas como resultado de algún proceso de optimización o alguna técnica heurística, pero esto conlleva a una convergencia prematura si no tenemos en cuenta que dentro de nuestra población inicial debemos garantizar la mayor diversidad estructural de las soluciones.

3.4.2 Función de Aptitud

La función de aptitud o fitness es de suma importancia para un AG, ya que esta indica cuan bueno es un individuo de la población con respecto a la solución del problema.

3.4.3 Selección

Durante cada generación sucesiva, una proporción de la población existente será seleccionada, estos individuos privilegiados son puestos en el Matting pool, para crear una nueva generación. Las soluciones individuales se seleccionan a través de un proceso basado en la aptitud, donde las soluciones más adecuadas (determinadas al aplicar una función de aptitud) son típicamente las que más probabilidad tienen de ser seleccionadas. Sin embargo debemos incluir un factor

de aleatoriedad para permitir la reproducción de los individuos con una aptitud baja, dado que estos pueden contener información valiosa para posteriores generaciones y así mantener cierta diversidad en cada población.

Algunas técnicas de selección que podemos emplear son las siguientes:

- *Selección directa*. Las soluciones se seleccionan de acuerdo a un criterio objetivo, por ejemplo "las k soluciones con mayor función de aptitud", "las k soluciones con menor función de aptitud".
- *Selección por ruleta*. Sea N el tamaño de la población y la aptitud del i-ésimo individuo. La probabilidad que tiene de ser seleccionado está dada por:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \quad (1)$$

  Este método de selección permite que los individuos con mayor aptitud tengan más probabilidad de ser escogidos, pero al mismo tiempo permite que los individuos con menor aptitud sean elegidos, manteniendo así la diversidad de individuos en la población.
- *Selección al azar*. Se seleccionan k individuos de manera aleatoria.
- *Selección por truncamiento*. La población se ordena según su función de aptitud, los k menos aptos se eliminan de la población y se sustituyen por la descendencia de alguno de los p mejores.

3.4.4 Operadores de reproducción

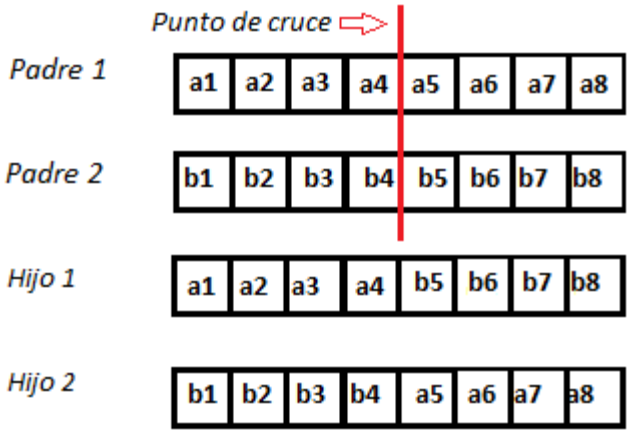
El objetivo principal de los operadores de reproducción es, teniendo dos cromosomas padres seleccionados del Matting pool, generar uno o más hijos que hereden las características de ambos padres. Se espera que con esta recombinación de características, salgan buenas soluciones, es decir, que con cierta frecuencia los hijos sean mejores soluciones que sus padres.

Para llevar a cabo este proceso, describiremos brevemente los operadores de reproducción más utilizados en los AGs.

Operador de Cruce.
Para aplicar este operador se escogen dos soluciones del Matting pool y de alguna forma combinar los cromosomas de ambos para generar dos descendientes. La forma de combinar los cromosomas varía de acuerdo al número de puntos de división a emplear y la forma de codificación del cromosoma.

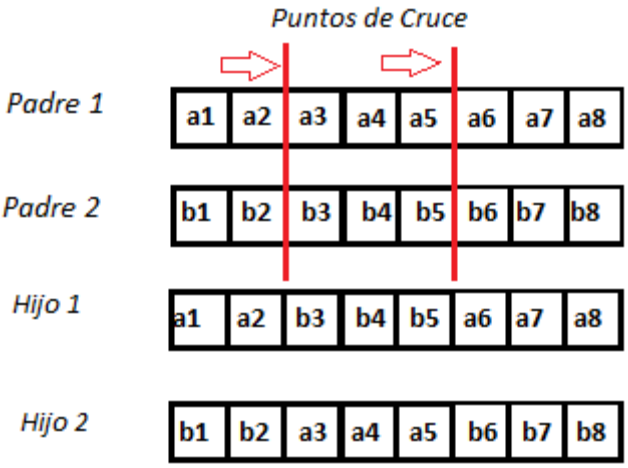
- Cruce de un punto. A cada par de soluciones escogidas anteriormente se le aplica un intercambio en su contenido desde una posición aleatoria K hasta el final como se muestra en la Figura 3., K es el denominado punto de cruce y determina la subdivisión de cada padre en dos partes que se intercambian para formar dos nuevos hijos.

Figura 3. Cruce en un punto.



- Cruce en varios puntos o multipunto. A cada par de soluciones escogidas anteriormente se le aplica un intercambio en su contenido desde dos o más posiciones aleatorias como se muestra en la Figura 4. Habitualmente se escogen dos puntos para el cruce. De Jong [15] sugiere que entre más puntos de cruce se definan, el espacio de búsqueda puede ser explorado con mayor rapidez. Adicionalmente propone que el número de puntos de cruce está determinado por L/2, siendo L la longitud del cromosoma, sin embargo, esto aumenta la probabilidad de ruptura de buenas soluciones.

Figura 4. Cruce en dos puntos.



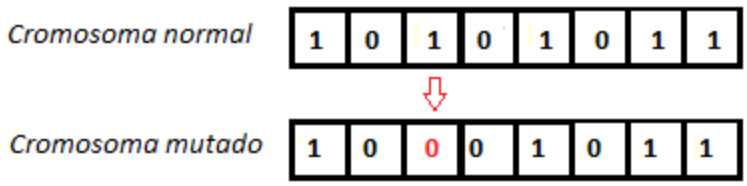
Operador de Mutación

Una vez es realizado el cruce entre los individuos de la población, hacemos uso del operador de mutación. La mutación es considerada un operador básico, que proporciona un pequeño grado de aleatoriedad a los individuos de la población. El operador de mutación consiste en la alteración de uno o más genes del cromosoma con una probabilidad de mutación de PM. Su objetivo principal es generar diversidad en la población y prevenir que las soluciones converjan en un óptimo local.

Las dos técnicas más habituales de mutación son:

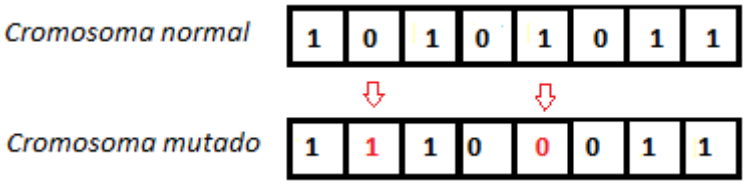
- Mutación de un bit. Consiste en cambiar aleatoriamente un bit de la cadena del cromosoma para generar un nuevo individuo. Para codificaciones binarias podemos cambiar aleatoriamente un bit por los valores 0 ó 1 como se muestra en la Figura 5.

Figura 5. Mutación de un bit para un cromosoma.



- Mutación multibit. Consiste en escoger dos o más posiciones del cromosoma y cambiar aleatoriamente el valor de los bits. En la Figura 6 podemos observar cómo se han cambiado los valores de los bits en las posiciones 2 y 5 del cromosoma.

Figura 6. Mutación multibit para un cromosoma.



3.4.5 Reemplazo

El reemplazo consiste en incorporar los nuevos cromosomas en la población. Existen muchas formas de realizar este proceso, pero las más usadas son:

- Reemplazo inmediato. Los nuevos cromosomas sustituyen a los padres.
- Reemplazo por inserción. Se eliminan los peores cromosomas de la población vieja, y se sustituyen por los nuevos.

4. ESTADO DEL ARTE

El físico matemático Achim Kempf [5], de la Universidad de Waterloo, propuso una nueva estructura del espacio-tiempo en la escala de Planck. Sugiere que el espacio-tiempo podría ser discreto y continuo al mismo tiempo, posiblemente satisfaciendo las teorías de la relatividad general y teorías de campo cuántico simultáneamente. La propuesta de Kempf está inspirada por la teoría de la información, ya que la información puede ser tanto continua como discreta al mismo tiempo. [5]

“Hay escuelas de pensamiento que intentan resolver la pregunta, ¿Es el espacio-tiempo fundamentalmente continuo o discreto? Aquí, consideraremos la posibilidad de que el espacio-tiempo puede ser simultáneamente continuo y discreto, de la misma forma en que matemáticamente la información puede ser simultáneamente continua y discreta". [5]

Kempf añadió que, por lo menos, el nuevo enfoque proporciona algunas herramientas técnicas prácticas para los estudios de gravedad cuántica, tales como la resolución de problemas discretos utilizando métodos continuos. En el futuro, Kempf planea aplicar los nuevos métodos a una variedad de problemas.[5]

Por otra parte Roger Penrose [13], físico matemático de la Universidad de Cambridge, ha dirigido sus trabajos para encontrar una teoría que pueda unificar la relatividad general y la mecánica cuántica.

En 1971 Penrose define un modelo llamado Spin Network [13], un spin network es un grafo que posee aristas etiquetadas con un número entero y algunos vértices etiquetados

por los operadores de entrelazamiento. Este modelo es discreto y tiene una aproximación geométrica al espacio euclidiano de dimensión 3, por lo tanto es un modelo aplicado al concepto de espacio. Gracias a la introducción de esta teoría, ahora tenemos una imagen matemáticamente rigurosa y convincente de los aspectos cinéticos de la gravedad cuántica de bucles. Físicos como Lee Smolin [1] y Markopoulo [4] han aplicado esta teoría para reformular la gravedad cuántica de bucles.

Dado que el modelo de spin network posee tres dimensiones, habría que añadir una cuarta dimensión temporal para que este sea un modelo relativista. En lugar de tratar de explicar cómo las partículas se mueven e interactúan en el espacio y el tiempo, Penrose propuso que el espacio y el tiempo son construcciones secundarias que emergen de un nivel más profundo de la realidad. En 1967 Penrose plantea un modelo relativista llamado twistors [13].

Pero su teoría de twistors no tuvo éxito, y los problemas conceptuales que tenía no fueron de agrado para sus pocos seguidores. Al igual que otros físicos que han fallado en su intento de unificar las dos grandes teorías, los twistors se dieron por olvidados [16]. Este trabajo de grado pretende aportar a la teoría de espacio-tiempo discreto y la teoría de la gravedad cuántica, un modelo computacional aproximado del espacio discreto, comprobando que el Teorema de Pitágoras, en dicho espacio, se cumpla gracias a la configuración que resultará de la ejecución de algoritmos evolutivos sobre las familias de grafos. Logrando de esta manera, para la comunidad científica, nuevas perspectivas sobre la unificación de la teoría de la Relatividad General y la mecánica cuántica.

5. METODOLOGÍA

5.1 Familia de Grafos

Los grafos son estructuras de datos que agrupan un conjunto de nodos o vértices, y aristas. Estos nos permiten representar o modelar diversos tipos de problemas. La estructura de grafos, usada en este trabajo, se diseñará en el lenguaje de programación *Ruby.* Como cualquier otra estructura de grafos conocida, se espera que esta estructura sea utilizada para futuros trabajos de grado.

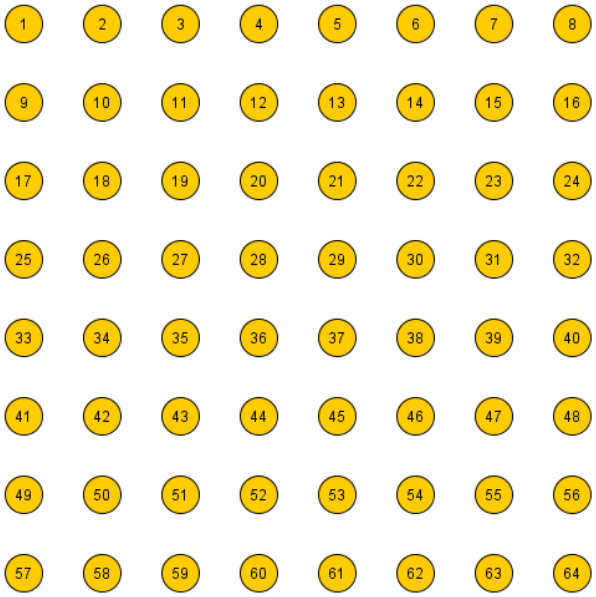
Las características de los grafos empleados en este trabajo son las siguientes:

1. Los grafos son bidireccionales
2. Un nodo puede estar conectado con uno o más nodos, pero un nodo no puede estar conectado consigo mismo (multigrafo).
3. En caso de detectarse modelos complejos estos se deben descartar para que el problema no crezca en complejidad espacial.
4. La disposición de una familia de grafos será una cuadricula de NxN, donde N es la cantidad de nodos que posee una familia.
5. La forma en cómo se conectaran los nodos se definirá de acuerdo al modelo para la búsqueda de soluciones.

El API que operará sobre esta estructura de datos debe permitir funciones que permitan calcular distancias entre nodos, subespacios, puntos medios, perpendiculares, entre otras. Para este trabajo se utilizará la librería *Igraph*, la cual soporta las funciones necesarias para la creación y manipulación de grafos.

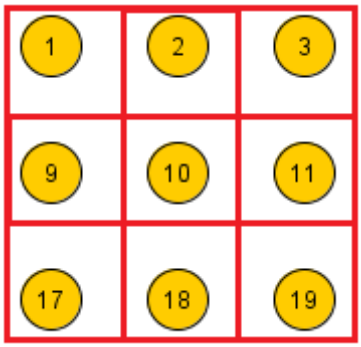
A partir de la disposición de nodos ilustrada en la Figura 7, se definirán dos modelos para la búsqueda de soluciones. Se ha optado por usar esta disposición de nodos porque representa un modelo de espacio similar al usado en los autómatas celulares.

Figura 7. Disposición de Grafos



Para codificar los cromosomas de los modelos de soluciones, se hará uso del principio de vecindad de Moore, el cual indica que para un nodo central *i* existen ocho vecinos a su alrededor. El número de vecinos puede variar de acuerdo al rango de profundidad o vecindad *r* que deseemos emplear y está dado por la formula $(2r+1)^2-1$; un nodo central con rango 1 posee 8 vecinos y con rango 2, 24 vecinos. En la Figura 8, el nodo central es 10 y sus ocho vecinos circundantes son los nodos 1, 2, 3, 9, 11, 17, 18 y19.

Figura 8. Modelo de vecindad de Moore



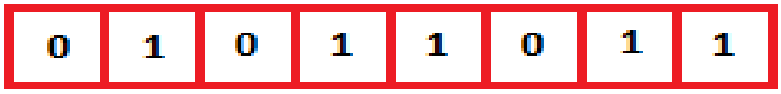
De acuerdo a este principio, podemos identificar plenamente los vecinos de un nodo cualquiera y por ende representarlos mediante un arreglo de tamaño $(2r+1)^2-1$, puesto que para definir el cromosoma de soluciones, un arreglo facilita las operaciones de reproducción en el algoritmo genético.

Figura 9. Vecinos del nodo central 10

Este arreglo de ocho elementos, Figura 9, dado por el rango de vecindad 1, define la posición que ocupará cada vecino del nodo central 10. Para cada posición del arreglo se define un elemento que establecerá la conexión con el nodo central y el nodo situado en dicha posición. Este elemento, denominado gen, será un número cuyos valores dependerán del modelo de solución propuesto.

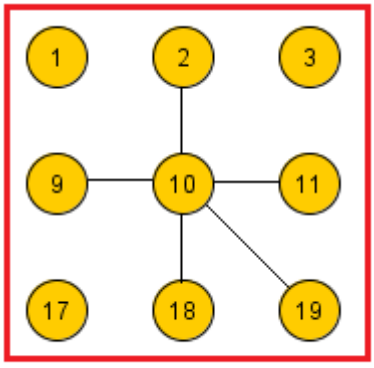
En el primer modelo de soluciones, el gen del cromosoma tomará los valores 0 ó 1, Figura 10. Estos valores actúan como condición para establecer o no, una arista entre el nodo central y un nodo vecino.

Figura 10. Cromosoma del primer modelo de soluciones



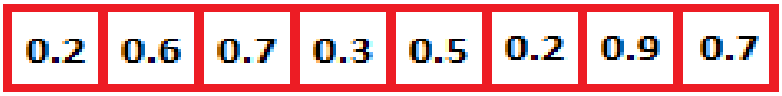
El valor 0 significa que no existe arista entre el nodo central y el nodo vecino ubicado en dicha posición del cromosoma; el valor 1 significa que existe una arista que conecta estos dos nodos. De acuerdo con el cromosoma ilustrado en la Figura 10, los nodos quedarían conectados de la siguiente forma:

Figura 11. Nodos conectados de acuerdo al cromosoma.



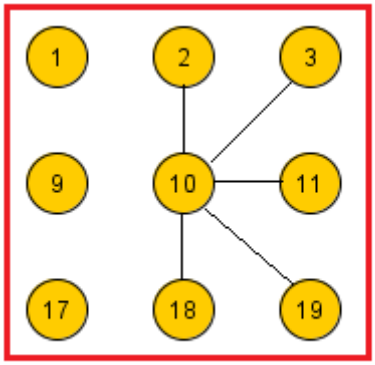
Los genes del cromosoma del segundo modelo de solución, tomarán valores aleatorios decimales entre 0 y 1. Adicionalmente, se debe definir un umbral para decidir si existe o no, una arista entre los nodos. Este umbral es un número fijo para cada familia de grafos y su valor es 0.5.

Figura 12. Cromosoma del segundo modelo de soluciones.



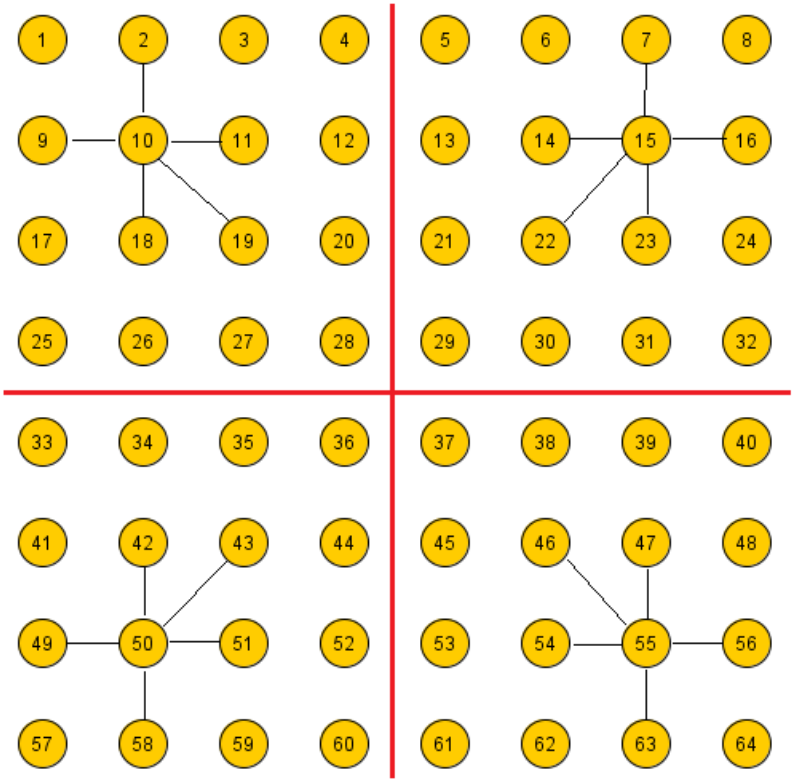
La conexión entre el nodo central y un nodo vecino depende de las dos siguientes condiciones, si el número en la posición del vecino i es menor que 0.5, no existe arista entre estos dos nodos; si el número en la posición del vecino i es mayor o igual que 0.5, existe una arista entre los dos nodos. Conforme al cromosoma presentado en la Figura 12 y a las dos condiciones anteriores, los nodos quedarían conectados de la siguiente forma:

Figura 13. Nodos conectados de acuerdo al cromosoma.



A estas dos formas de conectar los nodos, que se en repite para cada nodo en el grafo, se le añade una característica de isotropía, la cual consiste en girar 90 grados la dirección de conexión en cada cuadrante de la grilla de nodos, Figura 14. Además de generar más diversidad con esta característica, se garantiza que una solución encontrada hacia el sur, ocurra también hacia el norte.

Figura 14. Isotropía sobre los nodos

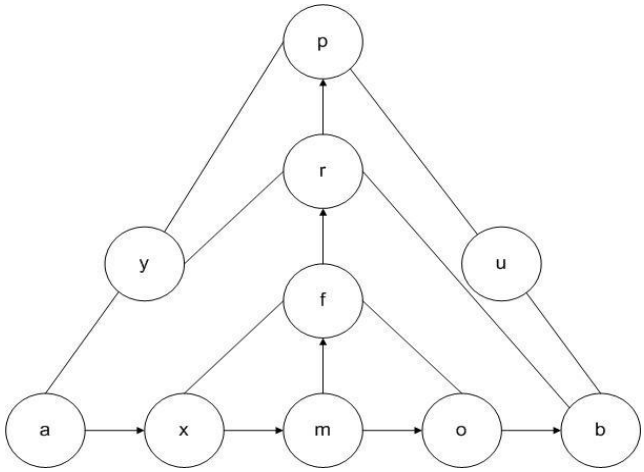


Al final de este trabajo, ambos modelos se compararán y se especificará cuál fue el modelo que minimizó el error en el teorema de Pitágoras.

5.2 Triángulo Rectángulo y Pitágoras sobre grafos

En esta sección se detallará la metodología empleada para representar un triángulo rectángulo mediante grafos, dado un grafo, como el de la siguiente figura:

Figura 15. Ilustración de un triángulo rectángulo sobre un grafo



Para hallar una aproximación de un triángulo rectángulo, dentro de un modelo de espacio discreto representado por medio de grafos, aplicamos los pasos enumerados a continuación:

1. Calcular la distancia entre dos nodos (se tomará como referencia los nodos a y b). Esta distancia siempre es un número entero y se calcula contando el número de saltos entre nodos.
2. Calcular el punto medio entre estos dos nodos. En el dibujo es el nodo m.
3. Buscar un nodo p tal que el recorrido entre los nodos p y m sea perpendicular al recorrido entre el nodo a y nodo b. Es perpendicular si la distancia entre el nodo p y el nodo a es igual a la distancia entre los nodos p y b. El resultado puede ser una lista de nodos, ya que pueden existir varios nodos que cumplan con este criterio.

4. Si el resultado anterior es una lista de nodos, por cada nodo p se debe verificar si se cumple el teorema de Pitágoras:

$$asb(\ d(a,m)^2 + \ d(p,m)^2 - d(p,a)^2\ ) = Error \quad (2)$$

5. El teorema de Pitágoras se cumple si el error se aproxima a cero.

Estos pasos se deben repetir en proporción al tamaño de la población del grafo para permitir una búsqueda de los posibles triángulos rectángulos dentro de la estructura del grafo.

5.3 Algoritmo Genético

El enfoque de algoritmos genéticos aplicado a este trabajo de grado, permite encontrar una solución óptima o sub-óptima, que minimiza el error en el Teorema de Pitágoras desarrollando los siguientes elementos del AG.

- El primer elemento es la correcta codificación del cromosoma, el cual representa la solución del problema. Su codificación se especifica en la sección 5.1.
- El segundo elemento es el tamaño de la población inicial. Este elemento se establece de acuerdo a la magnitud del problema. En una población con pocos individuos, posiblemente no se encuentre una solución factible; por el contrario, en una población muy grande, el algoritmo puede necesitar mucho tiempo para encontrar o no una solución.
- El tercer elemento son los operadores de reproducción. El cruce y la mutación son los principales operadores de este algoritmo. Proporcionan diversidad en la población con nuevos individuos que poseen mejores características con respecto a sus antecesores
- El último y más importante elemento es la evaluación de la función objetivo a través de la función fitness. Para cada generación de la población los cromosomas se deben evaluar para determinar cuál se aproxima mejor a la

solución del problema. El evaluación de la función objetivo se realiza de la siguiente manera:

A partir del número de repeticiones de la búsqueda de un triángulo rectángulo se obtiene un listado de errores del teorema de Pitágoras y se clasifican en el siguiente histograma:

Tabla 2. Histograma de frecuencias de los errores del teorema de Pitágoras

Índice	Rango	Cant. Errores Pitágoras
0	0..1	2
1	2..3	0
2	4..7	1
3	8..15	3
4	16..31	2
5	32..1024	1

Posteriormente se realiza la siguiente operación para obtener un número, el cual será la aptitud del cromosoma.

Tabla 3. Aplicando el factor a las frecuencias

Índice	Factor	Resultado
1	2 * 2^0	2
2	0 * 2^1	0
3	1 * 2^2	4
4	3 * 2^3	24
5	2 * 2^4	32
6	1 * 2^5	32

De acuerdo con la tabla anterior y la siguiente sumatoria obtenemos la aptitud del cromosoma:

$$\sum_{i=1}^{5} Cant.Error * 2^i = 94 \quad (3)$$

Este número se deberá minimizar en cada generación. Como se puede observar en la columna Resultado de la tabla 2, la aptitud del cromosoma puede ser un número menor, si se maximiza la cantidad de errores de Pitágoras en los rangos 0..1 , 2..3 y 4..7. Por el contrario, la aptitud será un número mayor si la cantidad de errores en los rangos 8..15, 16..31 y 32..1024, es alta. Esto último, significa que la solución no es buena, por lo tanto se debe descartar para las siguientes generaciones.

Luego de definir los componentes esenciales de un AG, se lleva a cabo la ejecución del algoritmo como sigue:

1. Inicializar la población de grafos.
2. Evaluar la aptitud de cada cromosoma.
3. Seleccionar por torneo k individuos de la población.
4. A partir de los k individuos seleccionados, crear nuevos cromosomas aplicando los operadores genéticos.
5. Reemplazar los nuevos individuos en la población por torneo.
6. Si el algoritmo converge a una solución óptima (minimizar el error en el Teorema de Pitágoras), se detiene su ejecución y se entrega la solución del problema; de lo contrario ir al paso 2 para crear una nueva generación.

5.4 Parámetros y Resultados

La presentación de los resultados constará de dos secciones, una por cada modelo de grafo y tomando como referencia la lista de errores del teorema de Pitágoras, a su vez, se subdividen por el rango de vecindad entre nodos. Los errores de cada ejecución se promedian por cada generación y por los rangos de interés 0..1 y 2..3.

También se presentarán los cromosomas más aptos para cada modelo en la siguiente estructura:

[cromosoma]
[[error de Pitágoras 1, nodo A, nodo B, nodo P, nodo M], [error de Pitágoras 2, nodo A, nodo B, nodo P, nodo M]]. *Esta lista puede tener una cantidad significativa de elementos, por tanto, solo se mostrarán los resultados más relevantes.*
[histograma]

5.4.1 Modelo Determinista

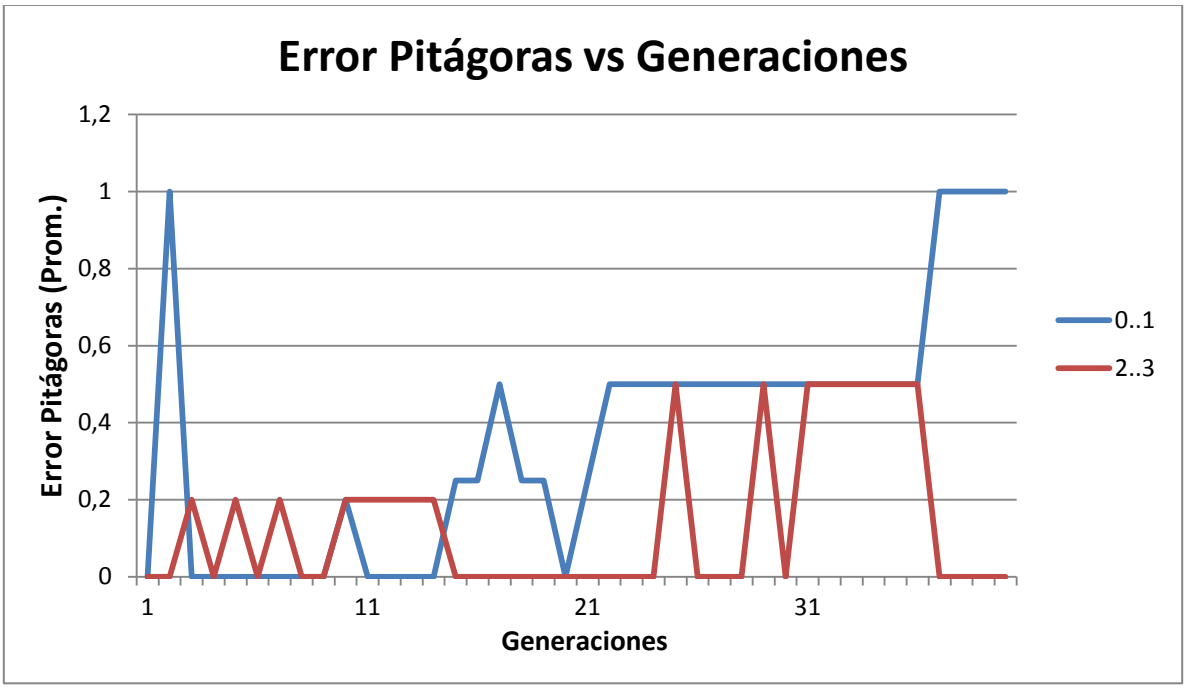
Los parámetros usados para el primer modelo con un rango de vecindad igual a uno, es decir generar una familia de grafos a partir de un cromosoma con 8 elementos, son los siguientes:

Cantidad de nodos: 30
Número de Generaciones: 40
Vecindad: 1
Cantidad de Ejecuciones: 5
Tiempo promedio de ejecución: 35 min.

Tabla 4. Resultados promedios de las frecuencias de errores del teorema de Pitágoras

Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)	Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)
1	0	0	21	0,25	0
2	1	0	22	0,5	0
3	0	0,2	23	0,5	0
4	0	0	24	0,5	0
5	0	0,2	25	0,5	0,5
6	0	0	26	0,5	0
7	0	0,2	27	0,5	0
8	0	0	28	0,5	0
9	0	0	29	0,5	0,5
10	0,2	0,2	30	0,5	0
11	0	0,2	31	0,5	0,5
12	0	0,2	32	0,5	0,5
13	0	0,2	33	0,5	0,5
14	0	0,2	34	0,5	0,5
15	0,25	0	35	0,5	0,5
16	0,25	0	36	0,5	0,5
17	0,5	0	37	1	0
18	0,25	0	38	1	0
19	0,25	0	39	1	0
20	0	0	40	1	0

Gráfica 1. Frecuencia promedio de errores de Pitágoras contra las generaciones



Los cromosomas más destacados en este modelo son:

[0, 0, 0, 1, 0, 1, 1, 0]
[1, 688, 628, 706, 658]
[[0..1, 1], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

[0, 1, 0, 0, 0, 0, 0, 1]
[[1, 75, 77, 124, 76]]
[[0..1, 1], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

[0, 1, 0, 0, 0, 0, 0, 1]
[[2, 732, 856, 736, 765]]
[[0..1, 0], [2..3, 1], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

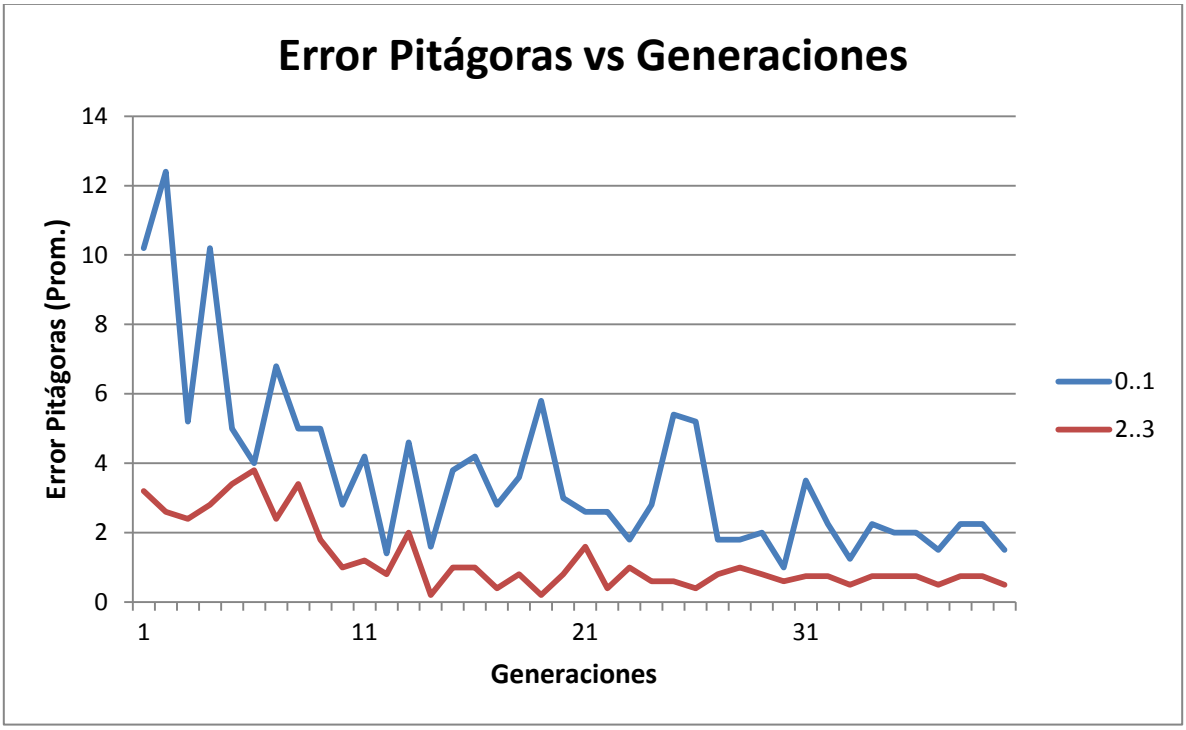
Los parámetros usados para el primer modelo con un rango de vecindad igual a dos, es decir generar una familia de grafos a partir de un cromosoma con 24 elementos, son los siguientes:

Cantidad de nodos: 30
Número de Generaciones: 40
Vecindad: 2
Cantidad de Ejecuciones: 5
Tiempo promedio de ejecución: 120 min.

Tabla 5. Resultados promedios de las frecuencias de errores del teorema de Pitágoras

Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)	Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)
1	10,2	3,2	21	2,6	1,6
2	12,4	2,6	22	2,6	0,4
3	5,2	2,4	23	1,8	1
4	10,2	2,8	24	2,8	0,6
5	5	3,4	25	5,4	0,6
6	4	3,8	26	5,2	0,4
7	6,8	2,4	27	1,8	0,8
8	5	3,4	28	1,8	1
9	5	1,8	29	2	0,8
10	2,8	1	30	1	0,6
11	4,2	1,2	31	3,5	0,75
12	1,4	0,8	32	2,25	0,75
13	4,6	2	33	1,25	0,5
14	1,6	0,2	34	2,25	0,75
15	3,8	1	35	2	0,75
16	4,2	1	36	2	0,75
17	2,8	0,4	37	1,5	0,5
18	3,6	0,8	38	2,25	0,75
19	5,8	0,2	39	2,25	0,75
20	3	0,8	40	1,5	0,5

Gráfica 2. Frecuencia promedio de errores de Pitágoras contra las generaciones



Los cromosomas más destacados en este modelo son:

[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]
[[0, 345, 550, 528, 433]]
[[0..1, 1], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1]
[[4, 852, 400, 606, 552], **[1, 1093, 1195, 702, 1094], [1, 1093, 1195, 2175, 1094], [1, 1093, 1195, 2256, 1094], … [1, 1093, 1195, 1701, 1094], [1, 1093, 1195, 2182, 1094], [1, 619, 2364, 1509, 1457]**,...]
[[0..1, 15], [2..3, 0], [4..7, 1], [8..15, 1], [16..31, 1], [32..1024, 8]]

[0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0]

[[4, 275, 312, 605, 249], … [4, 275, 312, 666, 249], **[0, 564, 752, 483, 629]**, [9, 564, 752, 774, 629], … **[0, 564, 752, 484, 629]**, [-6, 564, 752, 400, 629], [0, 564, 752, 484, 629], … **[1, 501, 528, 310, 559],** [12, 501, 528, 227, 559], [18, 501, 528, 99, 559], **[1, 501, 528, 376, 559],** [20, 501, 528, 38, 559], … [3, 283, 741, 635, 497], [-4, 283, 741, 609, 497], **[0, 283, 741, 473, 497], [1, 891, 770, 573, 830]**]

[[0..1, 8], [2..3, 2], [4..7, 21], [8..15, 23], [16..31, 30], [32..1024, 8]]

5.4.2 Modelo Aleatorio

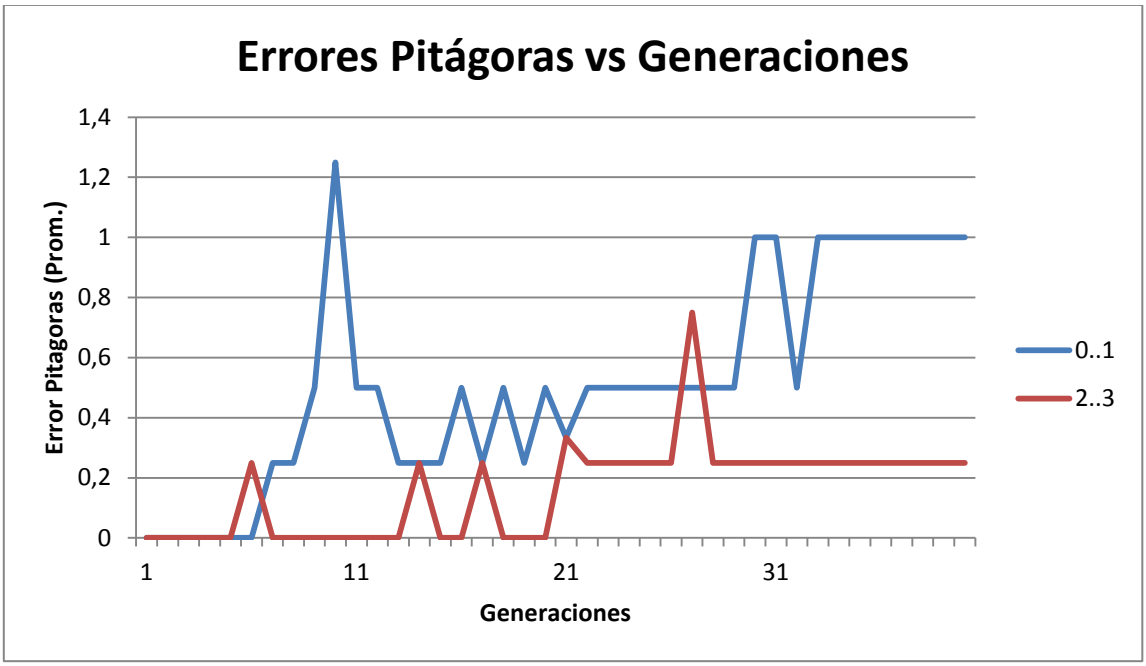
Los parámetros usados para el segundo modelo asignando un rango de vecindad igual a uno, es decir generar una familia de grafos a partir de un cromosoma con 8 elementos, son los siguientes:

Cantidad de nodos: 30
Número de Generaciones: 40
Vecindad: 1
Cantidad de Ejecuciones: 5
Tiempo promedio de ejecución: 45 min.

Tabla 6. Resultados promedios de las frecuencias de errores del teorema de Pitágoras

Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)	Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)
1	0	0	21	0,333333333	0,333333333
2	0	0	22	0,5	0,25
3	0	0	23	0,5	0,25
4	0	0	24	0,5	0,25
5	0	0	25	0,5	0,25
6	0	0,25	26	0,5	0,25
7	0,25	0	27	0,5	0,75
8	0,25	0	28	0,5	0,25
9	0,5	0	29	0,5	0,25
10	1,25	0	30	1	0,25
11	0,5	0	31	1	0,25
12	0,5	0	32	0,5	0,25
13	0,25	0	33	1	0,25
14	0,25	0,25	34	1	0,25
15	0,25	0	35	1	0,25
16	0,5	0	36	1	0,25
17	0,25	0,25	37	1	0,25
18	0,5	0	38	1	0,25
19	0,25	0	39	1	0,25
20	0,5	0	40	1	0,25

Gráfica 3.Frecuencia promedio de errores de Pitágoras contra las generaciones



Los cromosomas más destacados en este modelo son:

[0.8, 0.1, 0.7, 1.0, 0.1, 0.6, 0.6, 0.2]
[[1, 457, 516, 417, 486], [1, 457, 516, 568, 486]]
[[0..1, 2], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

[0.7, 0.9, 0.1, 1.0, 0.5, 0.9, 0.3, 0.3]
[[1, 460, 144, 472, 227], [25, 460, 144, 348, 227]**]**
[[0..1, 1], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 1], [32..1024, 0]]

[0.7, 0.5, 0.9, 0.8, 0.7, 0.7, 0.3, 0.1]
[[0, 438, 416, 471, 412]]
[[0..1, 1], [2..3, 0], [4..7, 0], [8..15, 0], [16..31, 0], [32..1024, 0]]

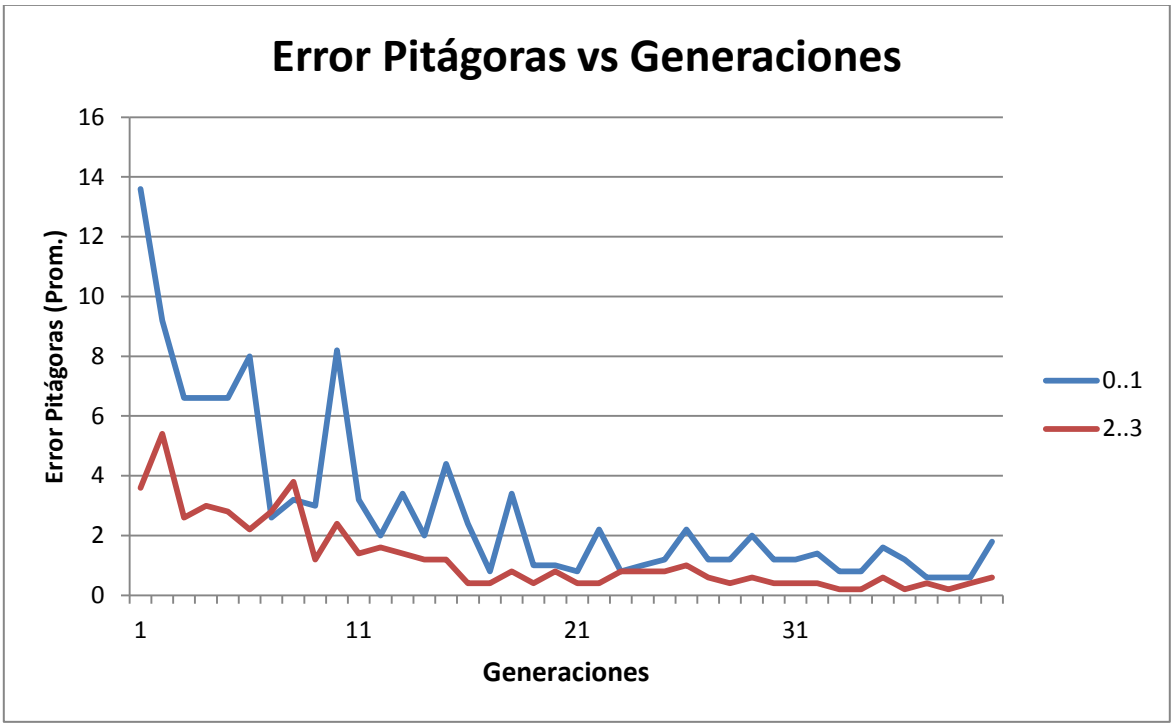
Los parámetros usados para el segundo modelo asignando un rango de vecindad igual a uno, generar una familia de grafos a partir de un cromosoma con 24 elementos, son los siguientes:

Cantidad de nodos: 30
Número de Generaciones: 40
Vecindad: 2
Cantidad de Ejecuciones: 5
Tiempo promedio de ejecución: 120 min.

Tabla 7. Resultados promedios de las frecuencias de errores del teorema de Pitágoras

Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)	Generación	Error Pitágoras (Prom. Rango 0..1)	Error Pitágoras (Prom. Rango 2..3)
1	13,6	3,6	21	0,8	0,4
2	9,2	5,4	22	2,2	0,4
3	6,6	2,6	23	0,8	0,8
4	6,6	3	24	1	0,8
5	6,6	2,8	25	1,2	0,8
6	8	2,2	26	2,2	1
7	2,6	2,8	27	1,2	0,6
8	3,2	3,8	28	1,2	0,4
9	3	1,2	29	2	0,6
10	8,2	2,4	30	1,2	0,4
11	3,2	1,4	31	1,2	0,4
12	2	1,6	32	1,4	0,4
13	3,4	1,4	33	0,8	0,2
14	2	1,2	34	0,8	0,2
15	4,4	1,2	35	1,6	0,6
16	2,4	0,4	36	1,2	0,2
17	0,8	0,4	37	0,6	0,4
18	3,4	0,8	38	0,6	0,2
19	1	0,4	39	0,6	0,4
20	1	0,8	40	1,8	0,6

Gráfica 4. Frecuencia promedio de errores de Pitágoras contra las generaciones



Los cromosomas más destacados en este modelo son:

[0.7, 0.7, 0.3, 0.6, 0.8, 0.1, 0.2, 0.8, 0.6, 0.3, 0.8, 0.1, 0.8, 0.3, 0.0, 0.0, 0.7, 0.0, 0.2, 0.9, 0.1, 0.9, 0.7, 0.8]

[[-2, 246, 751, 478, 511], [10, 246, 751, 614, 511], [-8, 246, 751, 384, 511], [1, 246, 751, 481, 511], [9, 246, 751, 547, 511], [-4, 115, 259, 8, 140], …, **[0, 661, 185, 427, 421], [1, 661, 185, 524, 421]**, … , **[0, 468, 638, 339, 522], [-2, 468, 638, 308, 522], [-2, 468, 638, 278, 522],** [22, 468, 638, 388, 522], [5, 170, 4, 330, 56], [16, 170, 4, 451, 56], [16, 170, 4, 453, 56], [16, 170, 4, 392, 56], [36, 108, 765, 476, 462], [1, 108, 765, 434, 462]]

[[0..1, 22], [2..3, 5], [4..7, 24], [8..15, 12], [16..31, 21], [32..1024, 11]]

[0.6, 0.4, 0.2, 0.1, 0.7, 0.7, 0.4, 0.8, 0.2, 0.4, 1.0, 0.9, 0.9, 0.7, 0.3, 0.6, 0.2, 0.8, 0.6, 0.2, 0.9, 0.4, 0.8, 0.9]

[[4, 392, 72, 217, 216], [-24, 392, 72, 2, 216], **[2, 392, 72, 280, 216],** [9, 392, 72, 494, 216], [4, 392, 72, 218, 216], **[0, 392, 72, 370, 216],** [36, 873, 433, 826, 603], …, [4, 747, 535, 671, 625], [19, 747, 535, 611, 625], [4, 747, 535, 456, 625], [9, 368, 180, 531, 185], [9, 368, 180, 501, 185], [37, 368, 180, 349, 185], **[0, 355, 666, 657, 481],** …, [-7, 748, 182, 434, 447]]

[[0..1, 2], [2..3, 2], [4..7, 11], [8..15, 10], [16..31, 6], [32..1024, 7]]

[0.3, 0.4, 0.5, 0.6, 0.2, 0.6, 0.7, 0.3, 0.5, 0.5, 0.7, 0.7, 0.3, 0.9, 0.0, 0.6, 0.4, 0.8, 0.0, 0.9, 0.8, 0.6, 0.7, 0.9]

[[-11, 678, 442, 187, 559], …, [9, 454, 722, 378, 601], **[2, 454, 722, 507, 601], [1, 454, 722, 629, 601]**, …, [16, 694, 757, 530, 695], [17, 635, 569, 864, 511], [19, 635, 569, 286, 511], **[4, 635, 569, 513, 511], …, [5, 302, 530, 194, 384]**]

[[0..1, 8], [2..3, 2], [4..7, 12], [8..15, 12], [16..31, 20], [32..1024, 9]]

5.5 Análisis de los resultados

El siguiente análisis pretende comparar, presentado sus ventajas y desventajas, los dos modelos descritos en este trabajo de grado.

De acuerdo a los resultados obtenidos y por medio de las gráficas de la sección anterior, observamos que en las primeras generaciones de ambos modelos de grafos, determinista y aleatorio, surgen soluciones con frecuencias altas en los rangos de errores 0 a 1, y posteriormente divergen disminuyendo las frecuencias en los rangos no deseados sin desviarse del objetivo del problema.

Sin embargo, en el modelo aleatorio el número de errores de Pitágoras igual a cero decae drásticamente cuando pasa de una generación a otra, esto se debe a que las conexiones de las aristas dependen de una condición en función de un umbral y los genes del cromosoma son números aleatorios entre 0 y 1, a diferencia del modelo determinista que este asigna 1 o 0 a los genes de sus cromosomas para realizar la conexión entre los nodos, haciendo este último menos sensible a las variaciones entre una generación y la siguiente.

Adicionalmente, podemos percibir que las familias de grafos generadas a partir de un cromosoma con un rango de vecindad dos, proporcionan considerablemente un mayor número de resultados comparados con los grafos formados por cromosomas de vecindad uno. Una vez que se aumenta el rango de vecindad logramos alcanzar un mayor nivel de profundidad de conexión entre los nodos, de esta forma, obtenemos diagonales entre dos nodos muy distantes, por tanto, la probabilidad de encontrar una solución óptima es mayor.

Otra forma de incrementar el número de errores de Pitágoras igual a cero es aumentando la cantidad de nodos en una familia de grafos, pues con base a este parámetro podemos ampliar el espacio de búsqueda de soluciones, además, al establecer una mayor cantidad de nodos iniciales podemos establecer, a su vez,

un rango de vecindad mayor. Lo anterior, aumentaría aún más la cantidad de mejores candidatos que minimicen el error en el teorema de Pitágoras.

No obstante, la complejidad espacial y temporal depende de estas dos variables, puesto que a mayor cantidad de nodos y mayor rango de conexión, el problema requerirá más recursos de cómputo para realizar los cálculos. Las tablas 8 y 9 ilustran el tiempo de ejecución de los dos modelos de grafos con 30 nodos y 50 generaciones para el algoritmo genético.

Tabla 8. Tabla de tiempos de ejecución del modelo determinístico.

Numero de Ejecución	Tiempo de ejecución con Vecindad 1 (min)	Tiempo de ejecución con Vecindad 2 (min)
1	30	94
2	31	124
3	40	80
4	35	106
5	32	123

Tabla 9. Tabla de tiempos de ejecución del modelo aleatorio.

Numero de Ejecución	Tiempo de ejecución con Vecindad 1 (min)	Tiempo de ejecución con Vecindad 2 (min)
1	40	150
2	36	142
3	26	91
4	28	105
5	32	126

Vemos que el tiempo de ejecución se llega a triplicar, o incluso más, cuando se aumenta el rango de vecindad en ambos modelos.

Ahora veamos cuál es la complejidad computacional del problema expuesto en este trabajo de grado con el siguiente pseudo-algoritmo:

```
Generaciones: Cantidad de generaciones en el algoritmo genético
Población: Cantidad de individuos del algoritmo genético
CantidadNodos: Número de nodos en el grafo
ListaVecinos: Listado de vecinos de un nodo en el grafo, su tamaño depende del rango de vecindad.
Cromosoma: Lista que contiene los valores que decidirán en la creación de una arista. Su tamaño es el mismo de la variable anterior.

para i desde 0 hasta Generaciones
       para j desde 0 hasta Poblacion/4
              para k desde 0 hasta CantidadNodos
                     para l desde 0 hasta Cromosoma
                            si Cromosoma[l] es igual a 1
                            arista entre ListaVecinos[l] y CantidadNodos[k]
                     Fin para
              Fin para
       Fin para
Fin para
```

Como la variable *Generaciones* no se puede estimar porque no conocemos con certeza en que generación debe parar el algoritmo, esta variable será m. Por consiguiente, este algoritmo es de complejidad $O(n^3)*O(m)$, dado que las instrucciones al interior del bucle for interno son de orden O(1), por lo que el bucle tendrá una complejidad O(n) frente a una complejidad de los dos bucles anidados externos de $O(n^2)$ y el bucle "*Generaciones*" O(m), el algoritmo tendría una complejidad de $O(n^3)*O(m)=O(n^2)${bucles externos}*O(n){bucle interno}*=O(m){bucle *Generaciones*}.

6. CONCLUSIONES

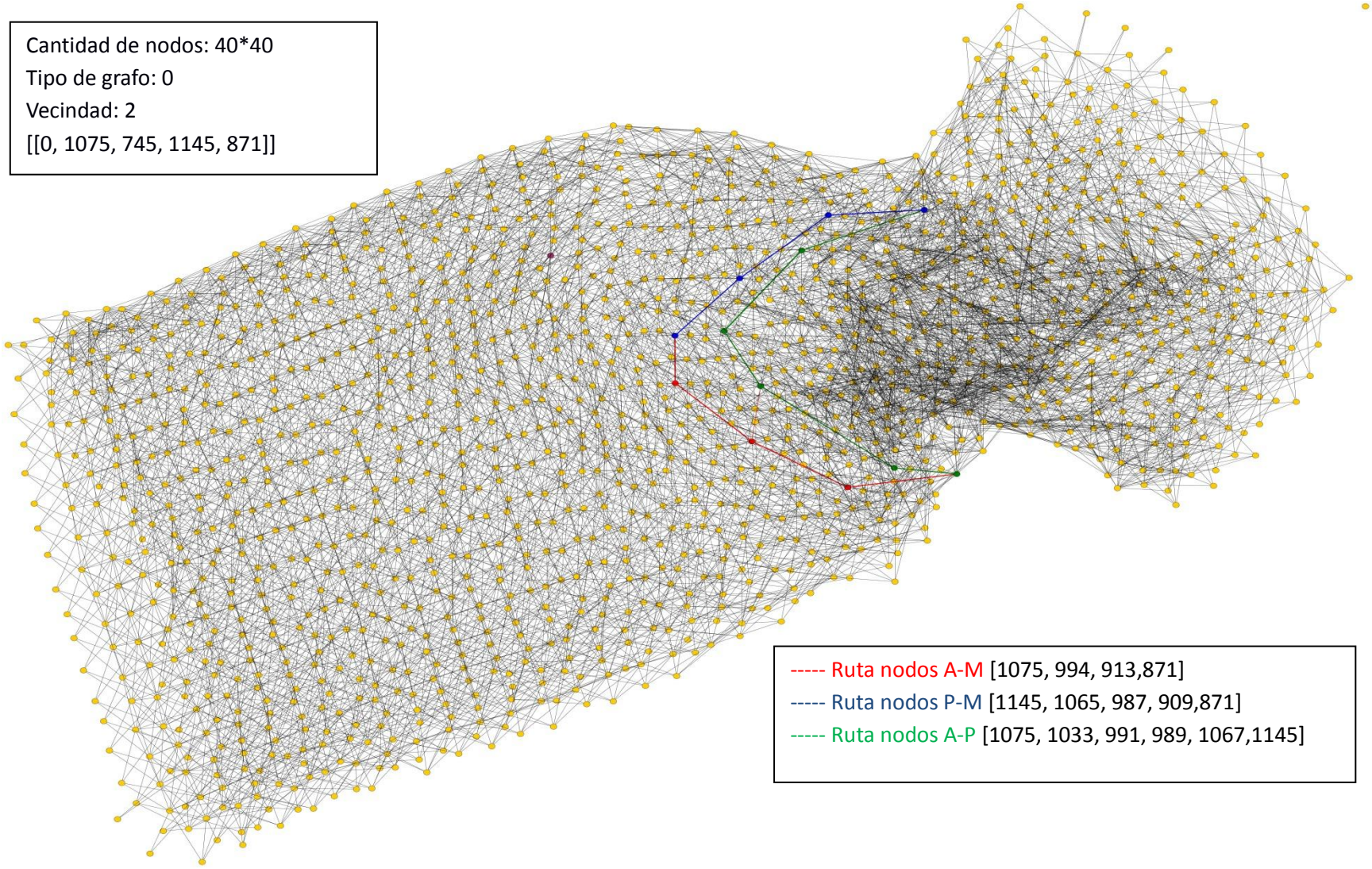
El modelo de software propuesto cumple satisfactoriamente los objetivos específicos planteados en este trabajo de grado. A través de los modelos desarrollados se logró encontrar por lo menos una combinación de parámetros que minimiza el error en el teorema de Pitágoras a cero.

Uno de los mayores limitantes en este trabajo es la complejidad del algoritmo, pues como vemos en el capítulo anterior si aumentamos el tamaño de una variable, en este caso el rango de vecindad, el tiempo de ejecución llega a triplicarse. Es por esta razón que se usaron parámetros en los cuales el tiempo de ejecución se ajusta a los recursos de cómputo usados y a un tiempo de estudio razonable del problema.

Al observar la gráfica 4, nos damos cuenta que en el modelo aleatorio los errores de Pitágoras tienden a decaer súbitamente entre una generación y la siguiente, además de deberse a que depende de un umbral y de números aleatorios, se infiere también, que su tiempo de ejecución sea menor, debido a que la cantidad de malas soluciones es significativa y el algoritmo terminará prematuramente. Contrario al modelo determinista donde su tendencia es mantenerse con al menos unas pocas soluciones optimas durante varias generaciones.

Gráfica 5. Ilustración del grafo a partir de los resultados obtenidos

Cantidad de nodos: 40*40
Tipo de grafo: 0
Vecindad: 2
[[0, 1075, 745, 1145, 871]]



----- Ruta nodos A-M [1075, 994, 913,871]
----- Ruta nodos P-M [1145, 1065, 987, 909,871]
----- Ruta nodos A-P [1075, 1033, 991, 989, 1067,1145]

La Gráfica 5, muestra un grafo obtenido a partir de los resultados conseguidos por el algoritmo implementado. Podemos observar la representación de un triángulo rectángulo y cómo se cumple el teorema de Pitágoras con una aproximación de error igual a cero. Cabe mencionar que gracias a la característica de isotropía, este triángulo se puede encontrar en otras tres direcciones diferentes. Debido a la limitación en el software de graficación los nodos no se encuentran distribuidos en una cuadricula como se mencionó en el capítulo anterior.

De lo anterior, podemos concluir que aunque en ambos modelos se cumple con el objetivo del problema y que usando cromosomas con una vecindad igual o mayor que dos se optimizan los resultados, el modelo determinista posee mejores características que el modelo aleatorio, pues el primero presenta una mayor estabilidad durante las generaciones del algoritmo genético lo cual garantiza que exista por lo menos una solución al problema en una ejecución.

Con relación a la metodología implementada en este trabajo, se utilizó el lenguaje de programación Ruby y la biblioteca de manipulación de grafos Igraph, los cuales me permitieron obtener un rápido desarrollo de los algoritmos para encontrar la aproximación de un triángulo rectángulo en un espacio discreto y la implementación del algoritmo genético para optimizar la función objetivo, minimizar el error en el teorema de Pitágoras.

Por último y como conclusión principal de este trabajo de grado, podemos decir que los modelos físicos que pretenden representar un espacio discreto, se pueden mejorar.

7. TRABAJO FUTURO

Debido a la complejidad del problema, $O(n^3)*O(m)$ y la limitación de recursos, los parámetros iniciales como cantidad de nodos, tipo de grafo, vecindad y numero de generaciones, asignados al algoritmo, deben ser seleccionados cuidadosamente, ya que una mala elección puede causar un consumo elevado de recursos.

Para dar continuidad a este trabajo, se recomienda implementar estos algoritmos en un lenguaje de programación mucho más eficiente, como C o C++, y usar técnicas de computación distribuida, de esta manera se logrará aumentar de forma considerable los parámetros iniciales, en especial la cantidad de nodos y el rango de vecindad, y tener mayores probabilidades de encontrar otros tipos de soluciones.

Finalmente, se pueden usar otras técnicas para la conexión de los nodos, por ejemplo, que la conexión entre un nodo A y un nodo B dependan de una probabilidad en función del rango de vecindad.

8. REFERENCIAS

[1] L. Smolin. Atoms of space and time. Sci. Am. 290N1:66–75, 2004

[2] Ashtekar and P. Singh, Loop quantum cosmology: A status report; Classical and Quantum Gravity 28, 213001, 1-123 (2011), arXiv: 1108.0893.

[3] J. R. Ortiz "El concepto de infinito", Electronic Library of Mathematics, Boletín Vol. I, Nº2, Año 1994.

[4] T. Konopka, F. Markopoulou, S. Severini, "Quantum Graphity: a model of emergent locality", Phys.Rev.D77:104029,2008, 25 paginas, 6 junio 2008.

[5] Kempf, "Spacetime could be simultaneously continuous and discrete, in the same way that information can be ", New Journal of Physics, 12, 115001 (2010)

[6] J. L. Mañes, “Geometría del Espacio-Tiempo.” Universidad del País Vasco-Euskal Herriko Unibertsitatea.

[7] Z. Michalewicz, "Genetic algorithms + Data structures = Evolution programs.” Springer – Verlag Berlin Heidelberg, New York, 1992.

[8] K. McDaniel, “Distance and Discrete Space” Department of Philosophy, Syracuse University, 541 Hall of Languages, Syracuse, NY 13244-1170, USA.

[9] M. Ponce, "Implementación Numérica del esquema de Discretizaciones Consistentes en el modelo cosmológico Gowdy", Trabajo de grado de Maestría en Física, Montevideo, Uruguay, Julio 2008.

[10] M. Florencia, "Propagacion de Fotones en Gravedad Cuantica", Trabajo de Grado de Doctorado en Fisica, Universidad Nacional de Córdoba, Marzo 2007.

[11] L. Frommberger, "Task Space Tile Coding: In-Task and Cross-Task Generalization in Reinforcement Learning", University of Bremen, AG Cognitive Systems, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany.

[12] M. Gestal Pose, "Introduccion a los algoritmos geneticos", Depto. Tecnologias de la Informacion y las Comunicaciones, Universidade da Coruña, http://sabia.tic.udc.es/mgestal/cv/AAGGtutorial/TutorialAlgoritmosGeneticos.pdf, Julio 8 2012

[13] R. Penrose, and W. Rindler, Spinors and Space-Time: Volume 2, Spinor and Twistor Methods in Space-Time Geometry, Cambridge University Press, 1988, ISBN 0-521-34786-6.

[14] R. Penrose (1971). Angular momentum; an approach to combinatorial space time, in Quantum Theory and Beyond, ed. T. Bastin, Cambridge University Press, Cambridge.

[15] K. A. De Jong and W.M. Spears (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of Maths. and AI, 5, 1–26

[16] G Musser, "Twistor, Theory Reignites the Latest Superstring Revolution", June 7 2010, http://www.scientificamerican.com/article.cfm?id=simple-twist-of-fate, Octubre 18 2012

9. ANEXOS

Se anexa CD con los siguientes archivos

Anexo 1. Informe final del trabajo de grado en formato PDF

Anexo 2. Carpeta con nombre “Software” contiene los siguientes archivos

- AG.rb : Clase que contiene el algoritmo genético
- GraphEvo.rb: Clase que genera los grafos determinísticos
- GraphRand.rb: Clase que genera los grafos aleatorios
- Main.rb: Clase principal que ejecuta todo el aplicativo
- Readme.txt: Instrucciones para la instalación y la ejecución del software

Anexo 3. Dentro de la carpeta Software se encuentra una carpeta adicional llamada “exportar grafica” que contiene los archivos necesarios para graficar un grafo

- Salida.rb: Clase que replica los datos de los grafos
- Main2.rb: Clase principal que recibe los parámetros para ilustrar un grafo.