UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 5

Título: Toma de decisiones en la evacuación de pasajeros en trenes aplicando el Algoritmo Colonia de Hormigas

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Lienni Daisson Columbié

Tutor(es): MsC. Roberto Millet Luaces

Co-tutor: MsC. Yuniesky Coca Bergolla

Asesor: Ing. Omar Correa Madrigal

La Habana

Junio - 2012

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y autorizo a la facultad 5 de la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

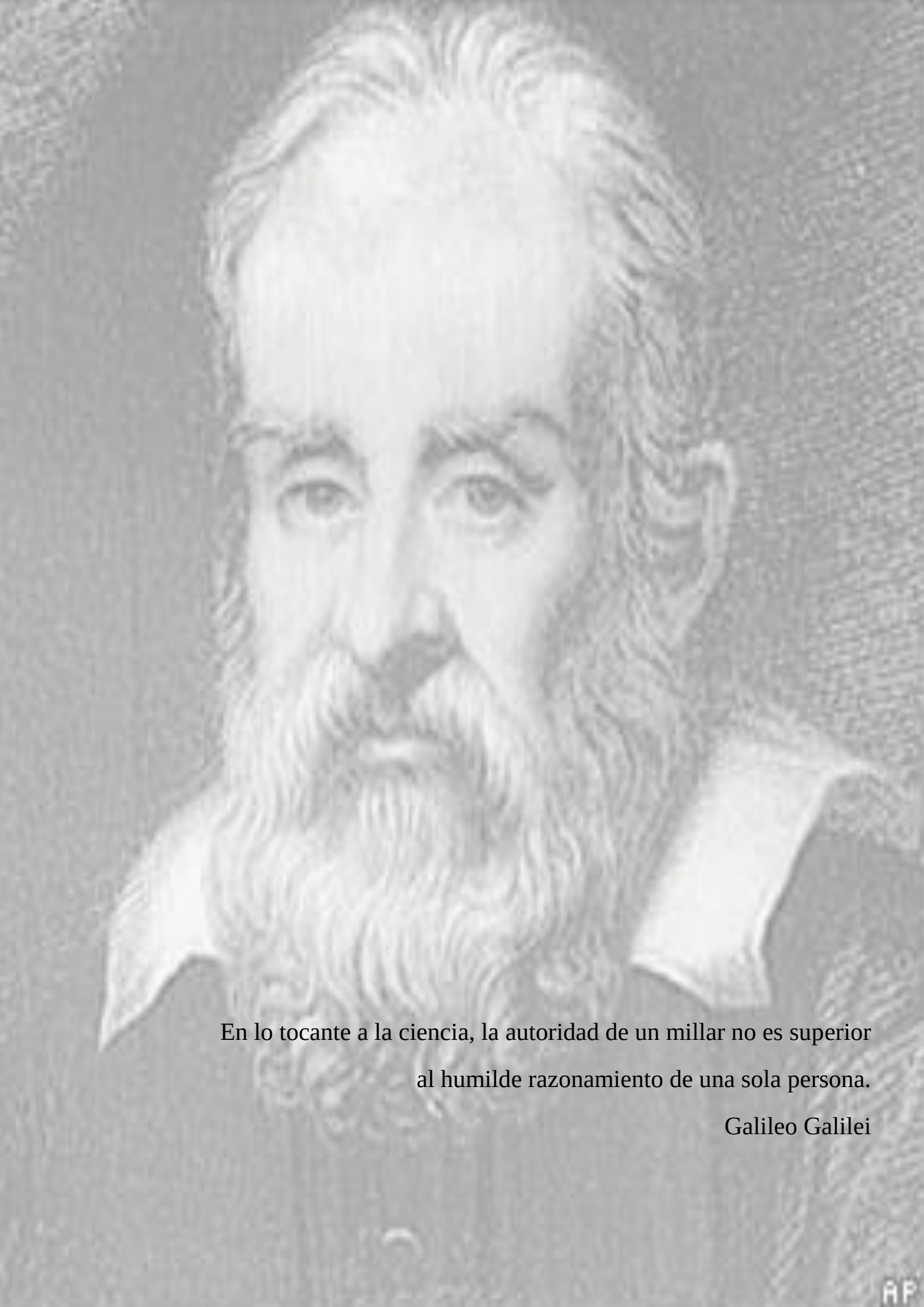
Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Lienni Daisson Columbié

_____
Firma del Autor

Roberto Millet Luaces

_____
Firma del Tutor



En lo tocante a la ciencia, la autoridad de un millar no es superior
al humilde razonamiento de una sola persona.

Galileo Galilei

Agradecimientos

Antes que todo agradecer a dios por permitir que llegara hasta este momento.

Primeramente agradecer a una persona que aún vive en mi corazón con todo el amor del mundo y cada día más me da fuerzas para seguir luchando, mi abuela Nora Silvia, le había prometido que terminaría la universidad con éxito, es una lástima que ella no esté presente pero yo sé que me está escuchando donde quiera que esté.

Agradecerles a mis padres y a mi hermano ya que son mis tesoros y mi razón de ser, por creer en mí, brindarme su apoyo en todo momento y su amor incondicional.

A mi tía Niurdis por creer en mí, quererme como una hija y ayudarme en todo momento.

A mi novio Alain Alberto por su dedicación sin límites, por apoyarme en mis momentos más difíciles.

A toda mi familia en particular a Paquito que ya es parte de mi familia.

A mis tutores Millet y Coca y mi asesor Omar que sin ellos no hubiera sido posible la realización de este trabajo.

A los profes y amigos que me ayudaron, gracias por sus valiosas sugerencias y acertados aportes durante el desarrollo del trabajo.

A todos aquellos que han venido a mi lado estos cinco años, no quisiera mencionarlos por temor a que se me quede alguno.

A la UCI, por convertirse a lo largo de estos cinco años en nuestra casa, nuestro barrio, nuestro pueblo; a sus profes, por darnos la oportunidad de crecernos como persona.

En fin a todo aquel que de una forma u otra ha contribuido con la realización del presente trabajo y todos los que han aportado su granito de arena para formar la persona que soy hoy.

A todos gracias

Lienni

Dedicatoria

Este trabajo está dedicado a:

A mi abuela Nora, mis padres, mi hermano y mi tía Niurdis, quienes soñaron noche por noche este momento, aquí se los dedico convertidos en realidad.

Lienni

RESUMEN

El desarrollo de la computación ha permitido la aparición de nuevos software y herramientas para la toma de decisiones, en el Centro de Informática Industrial (CEDIN) de la Universidad de las Ciencias Informáticas se han realizado simulaciones para el apoyo a la toma de decisiones, uno de estos productos fue la simulación de evacuación de pasajeros que viajan en trenes, la cual estuvo limitada por valores estáticos a las variables que se le asignaron, lo cual motivó en la presente investigación, aplicar una técnica basado en el comportamiento natural de las hormigas para sustentar de forma más realista la evacuación de estos pasajeros.

Se realizó un estudio relacionado con el funcionamiento del Algoritmo Colonia de Hormigas (ACH) para su utilización. Se propuso realizar una simulación que permita la toma de decisiones en evacuación de pasajeros en trenes utilizando dicho algoritmo, a partir de las características del modelo de datos de la simulación realizada por el CEDIN, se construyó un modelo matemático con los valores reales del vagón del tren. La representación del algoritmo está dada mediante un tipo de dato (grafo), donde se tiene en cuenta los pasos, estructura y procedimientos para su aplicación, realizándose una simulación de la evacuación de pasajeros en un tren para el apoyo a la toma de decisiones, la cual fue implementada en el lenguaje C++ en el software Qt-Creator.

PALABRAS CLAVE

Evacuación, Algoritmo Colonia de Hormigas, toma de decisiones.

TABLA DE CONTENIDOS

RESUMEN I

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1 Toma de decisiones 5
1.2 Evacuación de personas en tren 8
1.3. Inteligencia Artificial 9
1.3.1 Técnica de Inteligencia Artificial 10
1.3.2 Técnicas de IA para tomar decisiones 11
1.4 Optimización 14
1.5 Algoritmos Bioinspirados 16
1.6 Algoritmo Colonia de Hormigas (ACH) 19
1.6.1 Algoritmo de Optimización Colonia de Hormigas 19
1.6.2 Resultados del Algoritmo Colonia de Hormigas 22
1.7 Tipo de dato Abstracto (Grafo) 23
1.8 Software y lenguaje de programación 25

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN 26

2.1 Componentes de un sistema de evacuación para la simulación con colonia de hormigas 26
2.2 Diferenciación de modelos 28
2.3 Diferenciación de Casos 30
2.4 Distancias en el vagón del tren 33
2.5 Optimización con Colonia de Hormigas (OCH) 35
2.6 Diagrama del funcionamiento del algoritmo 43
2.7 Resultados y Discusión 44

CAPÍTULO 3: VALIDACIÓN DE LA PROPUESTA 45

3.1 Selección de la metodología 45
3.2 Exploración 45
3.2.1 Historia de Usuario 46
3.3 Planificación de la Entrega 49
3.3.1 Estimación de esfuerzo por historias de usuario 50
3.3.2 Planificación de iteraciones 50
3.3.3 Plan de duración de iteraciones 51
3.3.4 Plan de entregas 51
3.4 Diseño de la Aplicación 51
3.4.1 Tarjetas C.R.C 52
3.4.2 Interfaces de Usuario 55

3.5 Implementación y Prueba 56
3.5.1 Primera Iteración 56
3.5.2 Segunda Iteración 58
3.5.3 Pruebas de Aceptación 60

RECOMENDACIONES 65

REFERENCIAS BIBLIOGRÁFICAS 66

ANEXOS 69

ÍNDICE DE FIGURAS

Figura 1 Comportamiento de las hormigas 18
Figura 2 Modelo de datos 32
Figura 3 Grafo 1 28
Figura 4 Grafo 2 29
Figura 5 Grafo 3 29
Figura 6 Grafo Final 30
Figura 7 Seudocódigo de la metaheurísticaACO 36
Figura 8 Diagrama del funcionamiento del algoritmo 43
Figura 9 Interfaz inicial de la aplicación 55
Figura 10 Interfaz Leer Puntos 55
Figura 11 Interfaz ejecutar 56
Figura 12 Interfaz Opciones 56

ÍNDICE DE TABLAS

Tabla 1 Estructura de Historia de Usuarios 46
Tabla 2 Historia de Usuario: Mostrar el Modelo de Datos en la escena 47
Tabla 3 Historia de Usuario. Pintar los caminos en el Modelo de Datos 48
Tabla 4 Historia de Usuario. Ejecutar el algoritmo 48
Tabla 5 Historia de Usuario. Mostrar formulario de opciones 48
Tabla 6 Historia de Usuario. Modificar parámetros de las opciones 49
Tabla 7 Historia de Usuario. Reiniciar escena 49
Tabla 8 Estimación de esfuerzo por historias de usuario 50
Tabla 9 Plan de duración de iteraciones 51
Tabla 10 Plan de entregas 51

Tabla 11 Plantilla de Tarjeta C.R.C 52
Tabla 12 Tarjeta CRC: CHormiga 52
Tabla 13 Tarjeta CRC: CColoniaHormigas 53
Tabla 14 Tarjeta CRC: Point 53
Tabla 15 Tarjeta CRC: Scene 53
Tabla 16 Tarjeta CRC: View 54
Tabla 17 Tarjeta CRC: VArtTrainCognitiveModel 54
Tabla 18 Tarjeta CRC: OgreVector3 54
Tabla 19 Historias de Usuarios desarrolladas en la primera iteración 57
Tabla 20 Tarea de Ingeniería: Crear interfaz visual de la aplicación 57
Tabla 21 Tarea de Ingeniería: Mostrar el Modelo de Datos en la escena 57
Tabla 22 Tarea de Ingeniería: Pintar los caminos en el Modelo de Datos. 58
Tabla 23 Tarea de Ingeniería: Crear interfaz visual para el formulario de opciones 58
Tabla 24 Tarea de Ingeniería: Mostrar formulario de opciones 58
Tabla 25 Historias de Usuarios desarrolladas en la segunda iteración 59
Tabla 26 Tarea de Ingeniería: Ejecutar el Algoritmo 59
Tabla 27 Tarea de Ingeniería: Modificar parámetros de las opciones 59
Tabla 28 Tarea de Ingeniería: Reiniciar escena 60
Tabla 29 Estructura de la Prueba de Aceptación 60
Tabla 30 Prueba de Aceptación: Comprobar modelo de datos. 61
Tabla 31 Prueba de Aceptación: Comprobar si se pintan los caminos en el Modelo de Datos 61
Tabla 32 Prueba de Aceptación: Comprobar el formulario de opciones 62
Tabla 33 Prueba de Aceptación: Comprobar si se ejecuta el algoritmo 62
Tabla 34 Prueba de Aceptación: Comprobar si se modifican los parámetros de las opciones 63
Tabla 35 Prueba de Aceptación: Comprobar si se reinicia la escena. 63

INTRODUCCIÓN

Un sistema de transporte abarca distintas escalas espaciales de prestación de servicios (urbana, interurbana, interregional e internacional) y atiende dos tipos de demanda (traslado de pasajeros o de carga). El traslado de las personas (pasajeros) se realiza por múltiples motivos: para trabajar, intercambiar información, obtener los bienes y servicios necesarios para la supervivencia, entre otros. Unos de los problemas que frecuentan en el mundo es la evacuación de pasajeros que viajan en determinados medios de transporte como: terrestre (ferroviario y automotor), aéreo, acuático (fluvial y marítimo), al producirse algún incidente tales como: Incendios, accidentes, asaltos y otros.

La evacuación de pasajeros es uno de los temas más importantes para la seguridad, dada las vidas humanas que pueden verse involucradas ante una situación de peligro. Es un proceso muy complejo no solo por su simulación, además por las influencias psicológicas que este formaliza, por ejemplo en la toma de decisiones de los pasajeros y dado a la naturaleza humana que este encierra.

El tipo de transporte que se abordará en el presente trabajo es el terrestre (ferroviario). El tren ha pasado por muchas facetas de avance en la historia mundial, e incluso ha tenido una gran influencia en el desarrollo de muchas sociedades; su uso e importancia varía según la época en que se sitúa el análisis; ha formado parte esencial de muchas naciones y presentado una gran ventaja en la industrialización.

Existen dos formas principales de mejorar el conocimiento del proceso de evacuación: Mediante ensayos de prestaciones de evacuación y mediante el modelado y simulación por ordenador. Esta última es la que utilizaremos en el presente trabajo, particularmente en evacuación de pasajeros en trenes, ya que constituye un importante elemento sin el cual no se alcanzaría los resultados que hoy exhiben. La evacuación de pasajeros en trenes tiene un grado de importancia alto para la sociedad, debido a la gran cantidad de personas que viajan constantemente en transporte ferroviarios, por sus grandes números de vagones y la capacidad que este admite.

En la Universidad de las Ciencias Informáticas (UCI) existen distintas líneas temáticas de investigación, una de ellas es la de Visualización y Realidad Virtual del Centro de Informática Industrial, dentro de esta se han desarrollado simuladores y se han aplicado técnicas para la visualización de fenómenos físicos y su incorporación a productos nacionales e internacionales, en el marco de proyectos de colaboración, donde se han obtenido resultados satisfactorios. Uno de estos productos es

la simulación de situaciones excepcionales de la evacuación de pasajeros en trenes, este tema es muy tratado en la actualidad, lo que ha inspirado la continuidad de investigación en nuestro centro, sin embargo dicha investigación estuvo limitada a una representación visual asignando valores estáticos a las variables que intervinieron, o sea no se utilizó un algoritmo que sustentara la simulación de forma realista para el apoyo a la toma de decisiones.

El ACH viene evolucionando desde 1992, está calificado como metaheurística. Es una técnica probabilística utilizada para solucionar problemas de cómputo; este algoritmo está inspirado en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento. Los ACH son procesos iterativos. En cada iteración se "lanza" una colonia de m hormigas y cada una de las hormigas de la colonia construye una solución al problema. Las hormigas construyen las soluciones de manera probabilística, guiándose por un rastro de feromona artificial y por una información calculada de manera heurística.

A partir de las características del ACH, se prevé que con la utilización del mismo se pueda ayudar a la toma de decisiones en la evacuación de trenes, por tanto se define como problema científico:

¿Cómo utilizar el Algoritmo Colonia de Hormigas para el apoyo a la toma de decisiones en la evacuación de pasajeros en trenes?
Se define como objeto de estudio:
Proceso de toma de decisiones aplicando el Algoritmo Colonia de Hormigas.
Se define como objetivo general:
Desarrollar una simulación que ayude a la toma de decisiones en la evacuación de pasajeros en trenes utilizando el Algoritmo Colonia de Hormigas.

Dentro de los objetivos específicos está:

-Elaborar el estado del arte de las técnicas y algoritmo para ser aplicados al problema de evacuación de pasajeros en trenes.

-Simular la evacuación de pasajeros en trenes mediante el ACH.

-Proponer acciones que permitan la simulación de la evacuación de pasajeros en trenes utilizando el ACH.

Introducción

Se define como campo de acción: Proceso de implementación del Algoritmo Colonia de Hormigas para la evacuación de pasajeros en trenes.

Posibles resultados: La obtención de una aplicación del Algoritmo Colonia de Hormigas que ayude a la evacuación de pasajeros que viajan en trenes.

Tareas investigativas:

- Analizar el Estado del arte para indagar sobre las investigaciones realizadas respecto al tema.
- Elaborar el marco teórico de la investigación.
- Realizar un diagnóstico de la situación actual para identificar las principales necesidades de la investigación.
- Aplicar una heurística a partir de la metaheurística de Optimización con Colonia de Hormigas.
- Implementar el algoritmo para el modelo de datos del vagón del tren.

Métodos Teóricos:

-La Dialéctica-Materialista, método filosófico que permite el análisis del objeto de la investigación en sus múltiples manifestaciones y relaciones, así como identificar las contradicciones que lo caracterizan.

-El Histórico-Lógico permite estudiar las diferentes etapas por las que ha transitado el problema y a su vez analizar cómo se ha comportado este en el desarrollo de la temática en el mundo.

-El Análisis–Síntesis permite conocer el fenómeno a través de la descomposición de sus partes, es decir, hacer un estudio minucioso de sus elementos específicos.

Métodos empíricos:

La consulta de expertos para obtener el criterio de especialistas en el tema y validar el inicio de la investigación realizada, experimento que constituye la base de la investigación realizada para verificar la utilidad de la teoría de Colonia de Hormigas en la optimización y eficiencia a los problemas de evacuación, entrevistas, así como otras técnicas en aras de diagnosticar los problemas a partir de criterios dados por trabajadores y profesores.

Preguntas de Investigación o Ideas a Defender:

Con la utilización del Algoritmo Colonia de Hormigas se logrará simular la evacuación de pasajeros en trenes como apoyo a la toma de decisiones.

Estructura del Trabajo de Diploma.

El trabajo de diploma consta de tres capítulos:

Capítulo 1: Fundamentación Teórica.
En este capítulo se tratan conceptos y definiciones relacionados con la Toma de Decisiones, Evacuación de Pasajeros en Tren, Inteligencia Artificial, Optimización, Algoritmos Bioinspirados, Algoritmo de Optimización Colonia de Hormigas, Teoría de Grafos, Software y lenguaje de programación.

Capítulo 2: Propuesta de Solución.
Se explican las características necesarias para crear una aplicación basada en el comportamiento de la Colonia de Hormigas, se muestran los pasos y procedimientos a seguir para desarrollar la aplicación.

Capítulo 3: Validación de la Propuesta.
Se realizó una aplicación para la simulación de la evacuación de pasajeros en trenes y se muestran los resultados que se obtuvieron en la utilización del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Toma de decisiones

Existe consenso en las bibliografías consultadas a la hora de definir la toma de decisiones como un transcurso de varias opciones, mediante el cual se realiza elecciones entre ellas para resolver diferentes situaciones de la vida, consiste en elegir una de las opciones entre las disponibles para resolver un problema actual o potencial en las que se pueden presentar en diferentes contextos de la vida: Laboral, familiar, sentimental, empresarial, entre otros. Es decir en todo momento se toman decisiones.

La toma de decisiones se caracteriza porque las personas hagan uso de su razonamiento y pensamiento para tomar una decisión en un problema que se les presente en la vida. En la toma de decisiones importa la elección de un camino a seguir, por lo que en un estado anterior deben evaluarse alternativas de acción, si estas últimas no están presentes no existirá decisión.

Las decisiones se pueden clasificar teniendo en cuenta diferentes aspectos, como lo es la frecuencia con la que se presentan. Se clasifican en cuanto a las circunstancias que afrontan estas decisiones sea cual sea la situación para decidir y cómo decidir.

Existen dos tipos de decisiones Programadas y No programadas

Programadas (problemas Estructurados)

Solucionar problemas de rutina, es decir son repetitivas y se convierte en una rutina tomarlas; como el tipo de problemas que resuelve y se presentan con cierta regularidad ya que se tiene un método bien establecido de solución y por lo tanto, ya se conocen los pasos para abordar este tipo de problemas. Determinadas por reglas, procedimientos o hábitos. Aplicables tanto a cuestiones complejas como no complejas. Limitan nuestras decisiones, ya que es la organización y no el individuo el que decide qué hacer. Las decisiones programadas se toman de acuerdo con políticas, procedimientos o reglas, escritas o no escritas, que facilitan la toma de decisiones en situaciones recurrentes porque limitan o excluyen otras opciones. Si un problema es recurrente y si los elementos que lo componen se pueden definir, pronosticar y analizar, entonces puede ser candidato para una decisión programada.

Las decisiones programadas limitan nuestra libertad, porque la persona tiene menos espacio para decidir qué hacer. No obstante, el propósito real de las decisiones programadas es liberarnos. Las políticas, las reglas o los procedimientos que usamos para tomar decisiones programadas nos ahorran tiempo, permitiéndonos con ello dedicar atención a otras actividades más importantes.

NO Programadas (problemas no estructurados)

Son decisiones que se toman frente a problemas o situaciones que se presentan con poca frecuencia, o aquellas que necesitan de un modelo o proceso específico de solución. Se ocupan de problemas no habituales o excepcionales, si un problema no se ha presentado con la frecuencia suficiente como para que lo cubra una política o si resulta tan importante que merece trato especial, deberá ser manejado como una decisión no programada. Problemas como asignar los recursos de una organización, qué hacer con una línea de producción que fracasó, cómo modificar una línea de producción de autos o trenes para minimizar la ocurrencia de accidentes, cómo mejorar las relaciones con la comunidad de hecho, los problemas más importantes que enfrentará el gerente, normalmente, requerirán decisiones no programadas. Evitar lo rutinario. Ser creativos.

El proceso de toma de decisiones se divide en varias etapas:

→Identificar y analizar el problema

Consiste en encontrar el problema y reconocer que se debe tomar una solución para llegar a la solución de este. El problema puede ser actual, porque existe una brecha entre la condición presente real y la deseada, o potencial, porque se estima que dicha brecha existirá en el futuro. Es necesario tener una visión clara y objetiva, y tener bien claro el término alteridad, es decir escuchar las ideologías de los demás para así poder formular una posible solución colectiva.

→Identificar los criterios de decisión y ponderarlos

Consiste en identificar aquellos aspectos que son relevantes al momento de tomar la decisión, es decir aquellas pautas de las cuales depende la decisión que se tome. La ponderación es asignar un valor relativo a la importancia que tiene cada criterio en la decisión que se tome.

→Definir la prioridad para atender el problema

Se basa en el impacto y en la urgencia que se tiene para atender y resolver el problema. El impacto describe el potencial al cual se encuentra vulnerable y la urgencia muestra el tiempo disponible que se cuenta para evitar o al menos reducir este impacto.

→Generar las opciones de solución

Consiste en desarrollar distintas posibles soluciones al problema, si bien no resulta posible en la mayoría de los casos, conocer todos los posibles caminos que se pueden tomar para solucionar el problema, cuantas más opciones se tenga va a ser mucho más probable encontrar una que resulte satisfactoria. Para generar gran cantidad de opciones es necesaria una cuota importante de creatividad. Existen diferentes técnicas para potenciar la creatividad, tales como la lluvia de ideas, las relaciones forzadas, entre otras. En esta etapa es importante la creatividad de los tomadores de decisiones.

→Evaluar las opciones

Consiste en hacer un estudio detallado de cada una de las posibles soluciones que se generaron para el problema, es decir mirar sus ventajas y desventajas con respecto a los criterios de decisión, y una con respecto a la otra, asignándoles un valor ponderado. Existen herramientas, en particular para la administración de empresas para evaluar diferentes opciones, que se conocen como métodos cuantitativos. En esta etapa del proceso es importante el análisis crítico como cualidad del tomador de decisiones.

→Elección de la mejor opción

En este paso se escoge la opción que según la evaluación va a obtener mejores resultados para el problema. Existen técnicas (por ejemplo, análisis jerárquico de la decisión) que ayudan a valorar múltiples criterios.

→Aplicación de la decisión

Poner en marcha la decisión tomada para así poder evaluar si la decisión fue o no acertada, en su implementación probablemente se derive en la toma nuevas decisiones.

→Evaluación de los resultados

Evaluar si se solucionó o no el problema, es decir si la decisión está teniendo el resultado esperado o no. Si el resultado no es el que se esperaba se debe mirar si es porque debe darse un poco más de tiempo para obtener los resultados o si definitivamente la decisión no fue la acertada, en este caso se debe iniciar el proceso de nuevo para hallar una nueva decisión. El nuevo proceso que se inicie en caso de que la solución haya sido errónea, contará con más información y se tendrá conocimiento de

los errores cometidos en el primer intento. Además se debe tener conciencia de que estos procesos de decisión están en continuo cambio, es decir, las decisiones que se tomen continuamente van a tener que ser modificadas, por la evolución que tenga el sistema o por la aparición de nuevas variables que lo afecten.

Este proceso de toma de decisiones se pone en práctica diariamente en la vida cotidiana principalmente para resolver problemas inmediatos en cualquier esfera de la vida. Nos indica que un problema o situación es valorado y considerado profundamente para elegir el mejor camino a seguir según las diferentes alternativas y operaciones. Contribuye a mantener la armonía y coherencia del grupo, y por ende su eficiencia. En la toma de decisiones, considerar un problema y llegar a una conclusión válida, significa que se han examinado todas las alternativas y que la elección ha sido correcta. Dicho pensamiento lógico aumentará la confianza en la capacidad para juzgar y controlar situaciones. [1]

Por estas razones la toma de decisiones es utilizada para la solución de determinados problemas, como por ejemplo los problemas de evacuación de personas, en cualquier suceso, tanto en el transporte, como en cualquier otro hecho de la vida. Hoy día, las simulaciones realistas han permitido tomar decisiones para estos problemas a distintos niveles gerenciales.

1.2 Evacuación de personas en tren

Uno de los transportes más usados diariamente es el ferroviario debido a la gran cantidad de personas que viajan constantemente en este transporte y la capacidad que admite. Hay dos formas principales de mejorar el conocimiento del proceso de evacuación: Mediante ensayos de prestaciones de evacuación y mediante el modelado y simulación por ordenador. Esta última es la que se tratará en la presente investigación ya que constituye un importante elemento, además porque resuelve más rápido el problema, ahorra recursos materiales y humanos, permite a través de esta misma simulación dar solución a todos los problemas similares de evacuación y sobre todo porque disminuye la probabilidad de accidente por desorden.

¿Cómo se lleva a cabo un proceso de evacuación? Ante la posible ocurrencia de una emergencia y los múltiples incidentes presentados con gran concentración de personas, nos han enseñado que para afrontar con éxito la situación, la única fórmula válida, además de la prevención, es la planeación anticipada de los procedimientos a seguir en el momento crítico. Otro enfoque a seguir pero menos

dúctil lo constituye la ejecución en pre-procesamiento del modelo. Por lo general se realiza cuando la ejecución del mismo es muy compleja tanto por el número de agentes a analizar como por la complejidad física y psicosocial que presentan estos. [2]

1.3. Inteligencia Artificial

La Inteligencia Artificial (IA) es la rama de la ciencia de la computación que estudia la resolución de problemas no algorítmicos mediante el uso de cualquier técnica de computación disponible, sin tener en cuenta la forma de lógica subyacente a los métodos que se apliquen para lograr esa resolución. Existen varias definiciones de IA a continuación algunas ideas que facilitan la comprensión del objetivo de esta rama de las ciencias de la computación:

Es la habilidad de crear máquinas con capacidad de realizar funciones realizadas por personas que requieren de inteligencia.

Es el estudio de cómo lograr que las computadoras realicen tareas que, por el momento, los humanos hacen mejor.

Es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente.

Es el campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales.

Es la ciencia e ingeniería de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes.

En resumen está dedicada a la creación de hardware y software que reproduce el pensamiento humano, su principal objetivo es llevar a la computadora las amplias capacidades del pensamiento humano y para ello, se convierten a las computadoras en “mentes inteligentes” con la creación de software que les permite imitar algunas de las funciones del cerebro humano en aplicaciones particulares [3].

1.3.1 Técnica de Inteligencia Artificial

Una técnica de IA es un método que utiliza conocimiento representado de tal forma que:

-El conocimiento represente las generalizaciones, o sea no es necesario representar de forma separada cada situación individual. En lugar de esto, se agrupan las situaciones que comparten propiedades importantes. Si el conocimiento no posee esta propiedad, puede necesitarse demasiada memoria. Si el conocimiento no posee esta propiedad es mejor hablar de “datos” que de conocimiento.

-Debe ser comprendido por las personas que lo proporcionan. Aunque muchos programas, los datos pueden adquirirse automáticamente (por ejemplo, mediante lectura de instrumentos), en muchos dominios de la IA, la mayor parte del conocimiento que se suministra a los programas lo proporcionan personas, haciéndolo siempre en términos que ellos comprenden.

-Puede modificarse fácilmente para corregir errores y reflejar los cambios en el mundo y en nuestra visión del mundo.

-Puede usarse en gran cantidad de situaciones aun cuando no sea totalmente preciso o completo.

-Puede usarse para ayudar a superar su propio volumen, ayudando a acotar el rango de posibilidades que normalmente deben ser consideradas. [3]

Aunque las técnicas de IA deben diseñarse de acuerdo con las restricciones impuestas por los problemas de IA, existe cierto grado de independencia entre los problemas y las técnicas de resolución de problemas. Es posible resolver problemas de IA sin usar técnicas de IA. También es posible aplicar técnicas de IA para resolver problemas ajenos a la IA. Dos de las técnicas más destacadas en la IA en el campo de la informática son las redes neuronales y los algoritmos genéticos.

Una red neuronal es como un sistema que recibe un conjunto de entradas (percepciones), las procesa por medio de un conjunto de elementos ocultos que se encuentran conectados entre sí (en un sistema parecido a las conexiones de las neuronas) y que produce un resultado de salida (la acción que se va a tomar).

Los algoritmos genéticos permiten modelar el proceso de selección natural para poderlo aplicar a la resolución de varios tipos de problemas. Los algoritmos genéticos también tienen como base de su

funcionamiento un cromosoma. En este caso, un cromosoma describe una serie de atributos o propiedades de un elemento perteneciente al sistema. [4]

Existen otras técnicas importantes de IA principalmente para la toma de decisiones como se mencionan a continuación:

1.3.2 Técnicas de IA para tomar decisiones

A continuación se explicará la idea principal que se halla detrás de cada una de las técnicas:

→Sistemas de Reglas

En su forma más simple consisten en un conjunto de instrucciones "If... then..." que se utilizan para evaluar estados y tomar decisiones. Los sistemas de reglas presentan dos ventajas principales respecto al resto de sistemas de IA. En primer lugar, se trata de un sistema que intenta reproducir la manera de pensar y razonar de los humanos ante una situación a partir de la información que tienen de la misma. Y en segundo lugar, es un sistema muy fácil de implementar.

→Máquinas de estados finitos

Las máquinas de estados finitos permiten modelar el comportamiento de un sistema, especificando una serie de estados y de condiciones que deben cumplirse para realizar las transiciones entre ellos. El sistema se encuentra siempre en un estado activo y, a partir de la información que lea el sistema y los condicionales de las transiciones, se decide si se debe cambiar hacia otro estado.

→Árboles de decisión

Un árbol de decisión es otro sistema utilizado para analizar las diferentes opciones que se pueden llevar a cabo. Se trata de una estructura en forma de árbol donde se colocan todas las decisiones que podemos realizar y sus posibles consecuencias. La decisión en este tipo de árboles se lleva a cabo recorriéndolo desde la raíz hasta las hojas, para alcanzar así una decisión.

→Lógica Difusa

Una proposición en la lógica clásica sólo admite dos valores posibles: verdadero o falso. No obstante, en la vida real las observaciones no son tan claras como para poder afirmar que una proposición es

verdadera o falsa, sino que puede haber diferentes puntos de vista que sean relativos al observador. La lógica difusa es una alternativa que permite cuantificar esta incertidumbre, da la opción de asignar cierta probabilidad a cada uno de los posibles valores y, por lo tanto, añadir la relatividad del observador. Según su creador, la idea original que se halla detrás de la lógica difusa es la de imitar el funcionamiento del razonamiento humano, el cual normalmente no trabaja con valores exactos sino con valores relativos.

La lógica difusa es una extensión de la lógica tradicional (Booleana) que utiliza conceptos de pertenencia más parecidos a la manera de pensar humana. Se basa en lo relativo de lo observado como posición diferencial. Este tipo de lógica toma dos valores aleatorios, pero contextualizados y referidos entre sí. Así, por ejemplo, una persona que mida 2 metros es claramente una persona alta, si previamente se ha tomado el valor de persona baja y se ha establecido en 1 metro. Ambos valores están contextualizados a personas y referidos a una medida métrica lineal. La lógica difusa se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo "hace mucho calor", "no es muy alto", "el ritmo del corazón está un poco acelerado", etc. [5].

Aplicaciones de la Lógica Difusa

La lógica difusa se utiliza cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos, para procesos no lineales y cuando se envuelven definiciones y conocimiento no estrictamente definido (impreciso o subjetivo). En cambio, no es una buena idea usarla cuando algún modelo matemático ya soluciona eficientemente el problema, cuando los problemas son lineales o cuando no tienen solución. Esta técnica se ha empleado con mucho éxito en la industria, principalmente en Japón y cada vez se está usando en multitud de campos.

En IA, la lógica difusa se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general. Los sistemas de lógica difusa están también muy extendidos en la tecnología cotidiana, por ejemplo en cámaras digitales, sistemas de aire acondicionado, entre otros. Los sistemas basados en lógica difusa imitan la forma en que toman decisiones los humanos, con la ventaja de ser mucho más rápidos. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada. Algunos lenguajes de programación lógica que han incorporado la lógica difusa serían por ejemplo las diversas implementaciones de Fuzzy PROLOG o el lenguaje Fril.

Consiste en la aplicación de la lógica difusa con la intención de imitar el razonamiento humano en la programación de computadoras. Con la lógica convencional, las computadoras pueden manipular valores estrictamente duales, como verdadero/falso, sí/no o ligado/desligado. En la lógica difusa, se usan modelos matemáticos para representar nociones subjetivas, como caliente/tibio/frío, para valores concretos que puedan ser manipuladas por los ordenadores. En este paradigma, también tiene un especial valor la variable del tiempo, ya que los sistemas de control pueden necesitar retroalimentarse en un espacio concreto de tiempo, pueden necesitarse datos anteriores para hacer una evaluación media de la situación en un período anterior.

→Redes bayesianas

La idea es muy simple: cuando se quiere tomar una decisión se asigna una serie de probabilidades a cada una de las opciones que se tiene a partir de la observación y sobre estas probabilidades se toma la decisión. Una de las mejores técnicas para modelar la incertidumbre de las decisiones son las redes bayesianas. Las redes bayesianas son grafos dirigidos y acíclicos que representan la relación entre diferentes variables y sus relaciones de dependencia. Estas redes están basadas en la regla de Bayes.

→Mapas de influencia

Los mapas de influencia son una representación discreta del conocimiento que se tiene acerca del mundo. Son un elemento necesario para procesos de decisión estratégicos y tácticos. Los mapas de influencia se pueden utilizar para diferentes objetivos:

- Detectar aquellas regiones que interesan, aquellas otras regiones que se debe evitar y el lugar donde se encuentra la frontera entre estas regiones.

- Buscar puntos débiles, analizando la localización de sus defensas.

- Guiar el movimiento de un elemento.

- Cuantificar si un oponente tiene más fuerzas que nosotros. Si los valores negativos indican la fuerza del contrario y los positivos las nuestras, la suma de todos los valores nos indicará quién es más fuerte. [4]

La IA se está aplicando a numerosas actividades realizadas por los seres humanos y se destacan entre otras: La robótica, la visión artificial, técnicas de aprendizaje y la gestión del conocimiento. Estas aplicaciones de la IA son las que más directamente se aplican al campo de la informática así como las técnicas ya mencionadas.

En la búsqueda de nuevas técnicas de IA se han implementado estructuras auto-organizativas que simulen o imiten el funcionamiento de la vida biológica, tal es el caso de las redes neuronales artificiales, las cuales intentan representar el conocimiento desde el estrato más básico de la inteligencia (el estrato físico) que es el cerebro humano. Estas estructuras auto-organizativas se componen en sistemas robustos para el sistema de la información, cuya potencialidad aún es objeto de investigación.

Por otra parte, las redes neuronales se están ajustando con otras técnicas de reciente aparición tales como la lógica difusa, que provoca que las decisiones que se tomen tengan una vertiente más relativista y por tanto más humana y los algoritmos genéticos, los cuales facilitan un conjunto de herramientas potentes para la resolución de problemas complejos de optimización.

1.4 Optimización

La Optimización es la acción y efecto a optimizar, busca la mejor manera de realizar una actividad. Es la forma de dar respuesta a algún problema en el cual de acuerdo a sus posibilidades se desea elegir la mejor opción. Es el proceso de modificar un sistema para mejorar su eficiencia o también el uso de los recursos disponibles. En forma general la optimización puede realizarse en diversos ámbitos, siempre con el mismo objetivo: mejorar el funcionamiento de algo a través de una gestión perfeccionada de los recursos. La optimización puede realizarse en distintos niveles, aunque lo recomendable es concretarla hacia el final de un proceso. Existen distintos tipos de optimización según el nivel de generalidad que tome el problema, será la resolución que se plantee por ejemplo:

Optimización clásica: Si la restricción no existe, o es una restricción de igualdad, con menor o igual número de variables que la función objetivo entonces, el cálculo diferencial, da la respuesta, ya que solo se trata de buscar los valores extremos de una función.

Optimización con restricciones de desigualdad u Optimización no clásica: Si la restricción contiene mayor cantidad de variables que la función objetivo, o la restricción contiene restricciones de

desigualdad, existen métodos en los que en algunos casos se pueden encontrar los valores máximos o mínimos.

Optimización estocástica: Cuando las variables del problema (función objetivo y/o restricciones) son variables aleatorias, el tipo de optimización realizada es optimización estocástica.

Optimización con información no perfecta: En este caso la cantidad de variables, o incluso la función objetivo puede ser desconocida o variable. En este campo, la matemática conocida como matemática borrosa, está realizando esfuerzos, por resolver el problema. Sin embargo, como el desarrollo de esta área de la matemática es aún demasiado incipiente, son escasos los resultados obtenidos. [6]

Los problemas de optimización se componen de tres partes:

Función objetivo: Es la medida cuantitativa del funcionamiento del sistema que se desea optimizar (maximizar o minimizar).

Variables: Representan las decisiones que se pueden tomar para afectar el valor de la función objetivo. Se pueden clasificar en variables independientes o principales o de control y en variables dependientes o auxiliares o de estado, aunque matemáticamente todas son iguales.

Restricciones: Representan el conjunto de relaciones que ciertas variables están obligadas a satisfacer. [7]

Los métodos de optimización se pueden clasificar en: métodos clásicos y métodos metaheurísticos.

Métodos Clásicos

Son los que habitualmente se explican en los libros Investigación de Operaciones y se encuentran:

Programación Lineal; Programación Lineal Entera; Programación Lineal Entera Mixta; Programación cuadrática; Programación estocástica; Programación dinámica; Teoría de grafos u optimización en redes.

De forma muy general y aproximada se puede decir que los métodos clásicos buscan y garantizan un óptimo local. [8]

Métodos metaheurísticos

La metaheurística es una estrategia que guía y modifica otras heurísticas, aunque no garantiza que la solución encontrada sea la solución óptima global, la experimentación de implementaciones metaheurística muestra que son capaces de encontrar soluciones de alta calidad a problemas difíciles. Aparecieron ligados a lo que se denominó inteligencia artificial y se incluyen los algoritmos evolutivos (genéticos entre otros). [9]

Para tomar una decisión ya sea para maximizar o minimizar un criterio determinado es recomendable definir un algoritmo que permita la optimización de la solución a tomar. Para ello existen varios algoritmos y entre ellos los bioinspirados los cuales nos facilitan la solución a cualquier problema de optimización.

1.5 Algoritmos Bioinspirados

Los algoritmos bioinspirados simulan el comportamiento de sistemas naturales para el diseño de métodos heurísticos no determinísticos de búsqueda, aprendizaje, comportamiento. Son uno de los campos más prometedores de investigación en el diseño de algoritmos. Modelan (de forma aproximada) un fenómeno existente en la naturaleza. Metáfora biológica. Son no determinísticos, toman decisiones aleatorias. A menudo presentan, implícitamente, una estructura paralela (múltiples agentes). Son adaptativos (utilizan realimentación con el entorno para modificar el modelo y los parámetros).

Ejemplos:

Redes Neuronales: Basados en la simulación del comportamiento del Sistema Nervioso del cerebro humano. Paradigma de Aprendizaje Automático.

Algoritmos Evolutivos: Inspirados en los principios Darwinianos de Evolución Natural. Está compuesta por modelos de evolución basados en poblaciones cuyos elementos representan soluciones a problemas.

Inteligencia de enjambres: La inteligencia colectiva emergente de un grupo de agentes simples. Algoritmos o mecanismos distribuidos de resolución de problemas inspirados en el comportamiento colectivo de colonias de insectos sociales u otras sociedades de animales por ejemplo:

- Colmena de abejas
- Algoritmos basados en cúmulos de partículas
- Colonia de Hormigas [10]

- Colmena de Abejas

Los algoritmos basados en abejas no cuentan con el mismo nivel de agrupamiento. Por ahora el uso de estos algoritmos no es extensivo y la bibliografía es limitada. Una primera metaheurística propuesta dentro de los algoritmos de abejas corresponde a BCO. Donde las abejas artificiales realizan movimientos de construcción y selección de soluciones. En el proceso de decisión, se asume que cada abeja puede obtener información acerca de la calidad de las soluciones generadas por la población. De esta manera, las abejas intercambian información acerca de la calidad de las soluciones parciales creadas. Basándose en la calidad, cada una decide si abandonar o no su solución parcial y convertirse en un seguidor no comprometido, si continuar con la expansión de su solución parcial sin reclutar a otras compañeras, o si danza y recluta a otras compañeras antes de regresar a su solución parcial. Dependiendo de la cualidad de las soluciones parciales generadas, cada abeja posee un nivel de lealtad hacia la solución parcial descubierta. El proceso iterativo permite que las soluciones parciales se complementen hasta obtener una solución completa factible. Como ejemplo de los algoritmos de abejas se presenta el algoritmo ABHA.

Este algoritmo permite realizar búsquedas sobre espacios continuos, basados en el modelo orientado al individuo, donde se define una serie de reglas de comportamiento para cada agente que determinan las acciones a realizarse, incluyendo algunas adaptaciones no consideradas en el modelo biológico para incrementar el desempeño en la búsqueda de mejores soluciones. [11]

- Algoritmos basados en cúmulos de partículas

Un Algoritmo Basado en Cúmulos de Partículas es una técnica metaheurística basada en poblaciones e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces. La toma de decisión por parte de cada individuo o partícula se realiza teniendo en cuenta una componente social y una componente individual, mediante las que se determina el movimiento de esta partícula para alcanzar una nueva posición.

-Bandadas de Pájaros

Supongamos que una bandada de pájaros busca comida en un área y que solamente hay una pieza de comida en dicha área. Los pájaros no saben dónde está la comida pero sí conocen su distancia a la misma, por lo que la estrategia más eficaz para hallar la comida es seguir al ave que se encuentre más cerca de ella. PSO emula este escenario para resolver problemas de optimización. Cada solución (partícula) es un ave en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere. Las partículas son agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la mejor solución que han encontrado. Cada partícula tiene una posición y un vector velocidad que dirige su movimiento. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual. [12]

- Colonia de Hormigas

- Las Hormigas

Las hormigas son insectos sociales que viven en colonias y debido a su colaboración son capaces de mostrar comportamientos complejos, realizan tareas complicadas desde la óptica de una hormiga individual, una hormiga aislada se mueve a ciegas, es decir, sin ninguna señal que pueda guiarla.

Las hormigas vagan, transitan aleatoriamente en su búsqueda de alimentos y después de recorrer un largo camino depositan una sustancia química denominada feromona a lo largo del camino seguido (Transmitiéndose información) entre ellas, si las otras hormigas encuentran su rastro estas siguen este camino para depositar el alimento en la colonia. En el transcurso del tiempo esta feromona se evapora poco a poco lo cual reduce su fuerza atractiva pero las hormigas que siguen el rastro, aumentan la cantidad de la feromona con lo que este rastro se hace más fuerte y dura más tiempo, cuantas más hormigas recorran este camino más intenso será el olor de la feromona, lo que estimula a más hormigas a seguir esa trayectoria.

Feromona:

Es una sustancia química que depositan las hormigas mientras recorren el camino entre el hormiguero y la fuente de alimento, si alguna hormiga no encuentra este rastro de feromona, deambula el azar, en caso contrario sigue el rastro como una tendencia probabilística. Si el rastro se divide en dos una hormiga sigue y elige con probabilidad más alta el rastro que contiene mayor concentración de esta sustancia química.

Esta figura ilustra el comportamiento de las hormigas:

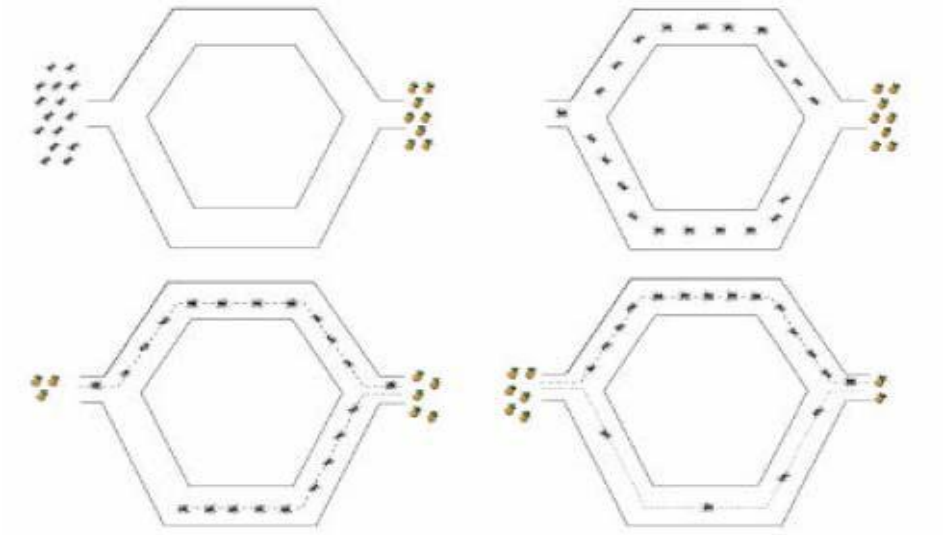


Figura 1 Comportamiento de las hormigas

Las hormigas llegan a un punto donde tienen que decidir por uno de los caminos que se le presentan, lo que les resuelve de manera aleatoria. La mitad de las hormigas se dirigirán hacia un extremo y la otra mitad hacia el otro extremo, ya que las hormigas se mueven aproximadamente a una velocidad constante, las que eligieron el camino más corto alcanzarán el otro extremo más rápido que las que tomaron el camino más largo, considerando que la evaporación de la sustancia química hace que los caminos menos transitados sean cada vez menos deseables.

Los caminos que son más corto provocan el regreso más temprano de las hormigas y así el depósito de feromona más pronto. Por tanto las hormigas siguientes verán condicionado su comportamiento por el camino más corto, igualmente el entorno favorece a la eliminación de los caminos no óptimos, ya que continúan recibiendo feromona de las hormigas con más frecuencia. [13]

1.6 Algoritmo Colonia de Hormigas (ACH)

→Hormiga Artificial

Una hormiga artificial es un agente computacionalmente simple que intenta construir solución a un problema explorando los rastros de feromonas disponibles. Sus características son:

-Busca soluciones válidas de coste mínimo para el problema a resolver.

-Tiene una memoria en la que almacena el camino recorrido para poder reconstruir soluciones válidas, evaluar las soluciones generadas, o bien reconstruir el camino completo llevado por la hormiga.

- Lleva a cabo los movimientos aplicando reglas de transición, que en función de los rastros de feromona que están disponibles, de los valores heurísticos, de la memoria privada de la hormiga y de las restricciones del problema.

- Durante su recorrido, modifica el entorno depositando una cantidad de feromona.

- Una vez que una hormiga ha construido una solución válida puede recorrer el camino de vuelta actualizando los espacios visitados. [13]

1.6.1 Algoritmo de Optimización Colonia de Hormigas

El ACH es una técnica probabilística utilizada para solucionar problemas de cómputo; este algoritmo está inspirado en el comportamiento que presentan las hormigas para encontrar las trayectorias desde la colonia hasta el alimento. Los Algoritmos de Optimización de Colonia de Hormigas se han utilizado para producir soluciones cuasi-óptimas al problema del viajante de comercio. El ACH puede funcionar continuamente y adaptarse a los cambios en tiempo real. Un ejemplo claro se puede observar en el problemas de enrutamiento de redes y sistemas urbanos del transporte. El ACH son procesos

iterativos. En cada iteración se "lanza" una colonia de m hormigas y cada una de las hormigas de la colonia construye una solución al problema. Las hormigas construyen las soluciones de manera probabilística, guiándose por un rastro de feromona artificial y por una información calculada de manera heurística. [14]

Heurística: En esencia una heurística es simplemente un conjunto de reglas que evalúan la posibilidad de que una búsqueda va en la dirección correcta. Generalmente los métodos de búsqueda heurísticas se basan en maximizar o minimizar algunos aspectos del problema. Un ejemplo sencillo de heurística es el siguiente:

NORTE: vegetación verde y movimiento de animales

SUR: vegetación amarilla

ESTE: vegetación amarilla

OESTE: vegetación verde

Evidentemente la vegetación verde es un indicio de que hay humedad, luego es muy probable que exista agua en la superficie o subterránea. El movimiento de animales puede indicar que ellos se dirigen allí a beber, lo cual sugiere que el agua está en la superficie. Esta información le dice al hombre que debe dirigirse al norte, constituye una heurística.

La Heurística no garantiza que siempre se tome la dirección de la búsqueda correcta, por eso este enfoque no es óptimo sino suficientemente bueno. Frecuentemente son mejores los métodos heurísticos que los métodos de búsquedas a ciegas. Las desventajas y limitaciones principales de la heurística son:

- La flexibilidad inherente de los métodos heurísticos pueden conducir a errores o a manipulaciones fraudulentas.
- Ciertas heurísticas se pueden contradecir al aplicarse al mismo problema, lo cual genera confusión y hacen perder credibilidad a los métodos heurísticos.
- Soluciones óptimas no son identificadas. Las mejoras locales determinadas por la heurística pueden cortar el camino a soluciones mejores por la falta de una perspectiva global. La brecha entre la solución óptima y una generada por heurística puede ser grande. [15]

Capítulo I

Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. Luego, con el propósito de obtener mejores resultados que los alcanzados por los heurísticos tradicionales surgen los denominados procedimientos metaheurísticos. [16]

Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la IA, la evolución biológica y los mecanismos estadísticos.

Como refieren Osman y Laporte [17] una metaheurística es un proceso iterativo de generación que guía a una heurística subordinada combinando de forma inteligente diferentes conceptos para la exploración del espacio de búsqueda y usando estrategias de aprendizaje para estructurar la información con objeto de encontrar eficientemente soluciones cercanas al óptimo. Dentro de los algoritmos metaheurísticos se distinguen los denominados métodos de trayectoria (búsqueda local, templado simulado, búsqueda tabú,...) y los métodos basados en poblaciones. En este último se enmarca el algoritmo utilizado en este trabajo.

Los Algoritmos de Optimización Colonia de Hormigas son esencialmente algoritmos constructivos: en cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo. Cada arista del grafo, que representa los posibles pasos que la hormiga puede dar, tiene asociada dos tipos de información que guían el movimiento de la hormiga:

→ Información heurística, que mide la preferencia heurística de moverse desde el nodo r hasta el nodo s, o sea, de recorrer la arista **A**rs, se denota por **N**rs. Las hormigas no modifican esta información durante la ejecución del algoritmo.

→Información de los rastros de feromona artificiales, que mide la “deseabilidad aprendida” del movimiento de r a s. Imita a la feromona real que depositan las hormigas naturales. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas. Se denota por **T**rs.

1.6.2 Resultados del Algoritmo Colonia de Hormigas

El ACH ha sido utilizado para máquinas de aprendizaje y para problemas con una gran cantidad de datos. Por ejemplo, se ha estudiado crear un modelo del mantenimiento del cementerio donde las hormigas arraciman los cadáveres de sus semejantes. Este algoritmo ha resuelto problemas tales como:

1. El Problema de la Mochila (PM) es un problema NP- Duro el cual tiene diversas aplicaciones prácticas como son: la asignación de procesos en sistemas distribuidos, el presupuesto de capital, entre otras. El objetivo del problema de la mochila es encontrar un subconjunto de objetos con el cual se maximice el beneficio o utilidad que proporcionan los objetos mientras que satisface la restricción de no sobrepasar la capacidad (espacio o peso) de un contenedor o depósito (en este caso la mochila) donde serán colocados los objetos. [18]

2. El Problema del Clique Máximo (PCM) con un Optimizador Local K-opt. Un clique es un conjunto C de vértices donde todo par de vértices de C está conectado con una arista en G, es decir C es un subgrafo completo. Un clique es parcial si esta forma parte de otro clique, de otra forma es máxima. La meta del PCM es el encontrar un clique máximo con mayor cardinalidad. [19]

3. Distribución en plantas orientadas a procesos. El objetivo es determinar la forma más efectiva de localizar un conjunto de secciones o talleres en una serie de áreas en la planta. En este caso la eficiencia se mide a partir del coste que supone el traslado de materiales o personas entre las diferentes secciones. El objetivo del problema es minimizar la función del coste sujeta a diferentes restricciones de carácter espacial. [20]

4. Costo de producción del proyecto SCADA. [21]

5. Un problema de transporte de carga desde varios orígenes a varios destinos. El problema consiste en elegir la alternativa de ruta que resulte óptima. [22]

6. El problema del viajante. Se dispone de un conjunto N = {1,..., n} de ciudades, que han de ser visitadas una sola vez, volviendo a la ciudad de origen, y recorriendo la menor distancia posible

7. Implementación paralela asíncrona del algoritmo ACS en una red de computadoras personales. [23]

El Algoritmo de Optimización Colonia de Hormigas implementado en los resultados anteriormente mencionados da la solución óptima, se requiere de la implementación de métodos heurísticos para obtener una solución en un tiempo polinomial. Es una metaheurística bien definida y con un buen rendimiento que se aplica para resolver cada vez un mayor número de problemas de optimización combinatoria complejos, presenta una inteligencia emergente que los hace capaces de obtener buenos resultados en problemas de alta calidad. Este algoritmo trabaja con el tipo de dato abstracto grafo ya que constituyen una herramienta básica para modelar fenómenos discretos y son fundamentales para la comprensión de las estructuras de datos y el análisis de algoritmos.

1.7 Tipo de dato Abstracto (Grafo)

La teoría de grafos es una rama de las matemáticas que examina las propiedades de los grafos. Un grafo en matemáticas e informática es una generalización del concepto simple de un conjunto de puntos, llamados vértices o nodos y una selección de pares de vértices, llamados aristas que pueden ser orientados o no. Típicamente, un grafo se representa mediante una serie de puntos (los vértices) conectados por líneas (las aristas). Los grafos constituyen una herramienta básica para modelar fenómenos discretos y son fundamentales para la comprensión de las estructuras de datos y el análisis de algoritmos. Son una abstracción útil para modelar diversas situaciones reales por ejemplo: redes de computadoras, redes telefónicas o eléctricas, circuitos eléctricos, sistema de carreteras, sistema de transporte y distribución de mercancías y sistemas organizacionales.

Un grafo simple G (V, E) consta de V, un conjunto no vacío de vértices, y de E, un conjunto de pares no ordenados de elementos distintos de V. A esos pares se les llama aristas o lados.

Un multigrafo G (V, E) consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en {{u, v} |u, v ∈ V, u = v}. Se dice que las aristas e1, e2 son aristas múltiples o paralelas si f (e1) = f (e2). Los multigrafos definidos no admiten bucles o lazos (aristas que conectan un vértice consigo mismo). Se usa en este caso, pseudografos que son más generales que los multigrafos.

Un pseudografo G (V, E) consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en {{u, v} |u, v ∈ V}. Se dice que una arista e es un bucle o lazo si f (e) = {u, u} = {u} para algún u ∈ V.

La diferencia entre grafo y digrafo es que el último tiene las aristas dirigidas y se entiende como un grafo dirigido.

Un grafo no dirigido es aquel en el que las conexiones no tienen una dirección establecida o si se prefiere, una conexión entre dos vértices está definida en los dos sentidos posibles.

Un multigrafo dirigido G (V, E) consta de un conjunto V de vértices, un conjunto E de aristas y una función f de E en {(u, v) |u, v ∈ V}. Se dice que las aristas e1, e2 son aristas múltiples o paralelas si f (e1) = f (e2).

Un grafo es ponderado si a cada arista e de G se le asigna un número no negativo w (e) denominado peso o longitud de e. El peso (o longitud de un camino en un grafo ponderado G se define como la suma de los pesos de las aristas del camino. Un importante problema en teoría de grafos es encontrar el camino más corto (liviano), esto es, el camino con el peso (longitud) mínimo entre dos vértices dados.

Una sucesión alternada de vértices y lados u1, e1, u2, e2,. . ., ek, uk+1 tal que ei = [ui, ui+1] se denomina cadena en un grafo y camino en un digrafo. Los caminos deben realizarse de acuerdo a la dirección de los lados. Si no existen lados múltiples se puede denotar sin ambigüedad la cadena como una sucesión de vértices (vértices consecutivos adyacentes). Una cadena es cerrada si el vértice inicial y final es el mismo. La cadena cerrada es un ciclo si todos los vértices (excepto los extremos) son distintos. El camino cerrado es un circuito si todos los vértices (excepto los extremos) son distintos.

Un camino en un grafo G = (V, A) un camino de longitud n (n ≥ 0) es una sucesión de vértices v0, v1,…, vn donde cada vértice vk es adyacente vk-1 para 1 ≤ k ≤ n. En este caso se dice que el camino va de v0 a vn.

Un camino simple que contiene cada vértice de G se denomina camino Hamiltoniano de G; análogamente, un ciclo Hamiltoniano de G es un ciclo que contiene todos los vértices de G. Tales caminos y ciclos son así llamados después que Hamilton (1856) describió, en una carta a su amigo Graves, un juego matemático sobre el dodecaedro en el cual una persona coloca cinco alfileres en cinco vértices consecutivos y a otra se le exige completar un camino simple hasta completar un ciclo. Un grafo es hamiltoniano si contiene un ciclo Hamiltoniano.

Los vértices adyacentes en un grafo G = (V, A), un vértice y es adyacente a otro vértice x si el par <x, y> es una arista del grafo G. [24]

1.8 Software y lenguaje de programación

El grafo que se propone en el presente trabajo es el ponderado el cual se implementará en el lenguaje de programación C++. Este es un lenguaje de programación diseñado a mediados de los años 1980 Bjarne Stroustrup. La intención de su creación fue extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Es un lenguaje de programación multiparadigma. Una particularidad del C++ es la posibilidad de redefinir los operadores y de poder crear nuevos tipos que se comporten como tipos fundamentales. [25]

El lenguaje de programación C++ se puede utilizar en varios entornos de desarrollo, como el Visual Studio, Borland C++ Builder y QT-Creator. Se utilizará este último debido a que con él se pueden programar desde los sistemas más sencillos, hasta los más complejos. Es un excelente entorno de desarrollo integrado multiplataforma para desarrollar aplicaciones en C++ de manera sencilla y rápida.

Presenta muchas ventajas como: auto-completar; sugerencias; marcado de palabras clave; marcado de comienzo y terminación de terminadores; depurador.

Qt-Creator como su nombre lo indica, está basado en la biblioteca Qt y cuenta con las siguientes características principales:

→Editor avanzado para C++.

→Diseñador de formularios (GUI) integrado.

→Herramientas para la administración y construcción de proyectos.

→Completado automático.

→Depurador visual.

Está disponible para Linux, Mac OSX y las plataformas de Windows. [26]

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

En este capítulo se realizará la propuesta de solución, primeramente se conocerán los componentes de un sistema de evacuación. Se mostrará un modelo matemático (grafo) teniendo en cuenta la estructura del diseño del vagón del tren, además los cálculos para hallar la distancia dentro del vagón. Se realizará un estudio acerca de los posibles casos que puede haber en este problema. Se exponen las características del software a realizar así como los pasos, estructura y procedimientos que se tuvieron en cuenta en el ACH para la realización del mismo. Luego de haber realizado una propuesta se presentan los resultados que se obtuvieron en la misma.

2.1 Componentes de un sistema de evacuación para la simulación con colonia de hormigas

El sistema de evacuación es un conjunto de elementos que se relacionan de manera dinámica entre sí, para salvaguardar la vida, movilizando personas de un punto de riesgo o impactado a un lugar seguro a través de rutas señalizadas. La evacuación permite salvar las vidas de quienes habiten la zona afectada o de posible afectación. Los evacuados han de trasladarse a un sitio seguro, por la naturaleza del evento, se puede realizar una evacuación por periodos cortos o tiempo indefinido. Los sistemas de evacuación están compuestos por dos elementos esenciales: el Modelo de Evacuación y el Modelo de Visualización. [27]

- Modelo de Evacuación

Este modelo instrumenta una determinada cantidad de roles, que son funciones que se asignan al personal y que deben ser llevadas a cabo durante todo el tiempo que el establecimiento permanezca en funcionamiento. Por ejemplo, uno de los roles lo describe como “Jefe Técnico”, el mismo, cuando se declare una emergencia es el encargado de cortar el suministro de gas y electricidad. En el modelo de evacuación se encuentra todo lo relacionado con el modelo matemático que rige una evacuación. Es importante señalar que los modelos buscan ser lo más objetivo posible pero hay otros modelos que tienden más al altruismo por la forma en que emprenden el problema. [28]

- Modelo de Visualización

El modelo de visualización, constituye una parte importante de todo sistema de evacuación. Es la formación de imágenes visuales, es el mapeo de datos en representaciones que pueden ser percibidas. Consiste en el uso de recursos gráficos, de animación y multimedia con importante interacción entre el usuario y la computadora.

Capítulo II

Existen tres partes fundamentales en un sistema de visualización:

1. Construcción de un modelo empírico de los datos: Este modelo puede tener consideraciones sobre teoría del muestreo, también se toma en cuenta la probabilidad de que haya errores en los datos.

2. Selección de esquemas: Significa tomar como modelo un objeto de visualización abstracta (un mapa por ejemplo).

3. La representación de la imagen en un ambiente gráfico. [29]

- Visualización de Software

La visualización de software comprende la visualización de algoritmos y de programas. La primera consiste en la visualización de abstracciones de alto nivel que describen el software, mientras que la segunda se refiere al código real de programa y a sus estructuras de datos. Ambas pueden darse en forma estática o dinámica. [30]

- Visualización de grafos

El trazado de grafos direcciona el problema de visualizar información estructural o relacional construyendo representaciones visuales geométricas de grafos o redes que son los modelos subyacentes en una gran cantidad de datos abstractos. Pretende visualizar información abstracta mediante representaciones o que describen relaciones entre entidades. La generación automática del dibujo de un grafo tiene importancia en aplicaciones clave tales como la Ingeniería de Software, la Visualización de Información, el diseño de Base de Datos e Interfaces Visuales, la Representación del conocimiento y las Telecomunicaciones entre otros dominios. [31]

El modelo de visualización se puede representar de dos formas en cuanto a las dimensiones como son: 2D y 3D.

El modelo de visualización que se utilizó fue el 2D.

Las representaciones 2D por lo general se utilizan para hacer visualizaciones del proceso de evacuación en construcciones que están compuestas por una única planta. En el caso de los trenes esta es la manera en que se ha abordado. Entre sus deficiencias puede encontrarse la escasez de recursos para visualizar el proceso en toda su magnitud en cuanto a claridad y nivel de aproximación espacial; esto último repercute en la manera en que el sistema aborda el entorno. [32]

2.2 Diferenciación de modelos

Para modelar el vagón del tren se analizaron varias topologías del grafo, para lo cual se analizaron varios aspectos dirigidos a lograr la mayor credibilidad de la simulación.

Inicialmente se analizó el grafo de las siguientes maneras:

1- Primeramente se estudió un grafo para la mitad del vagón, donde se analizó además, que los pasajeros de los asientos que estaban sentados del lado contrario y de frente al otro asiento correspondiente saldrían al pasillo por un punto además del que estaban sentado.

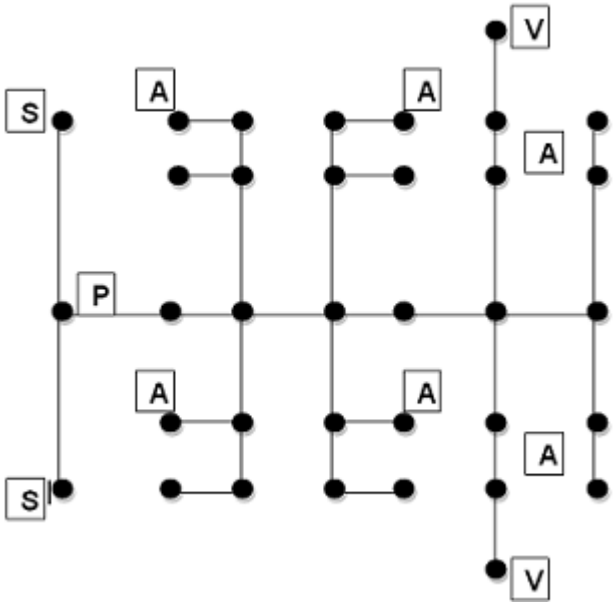


Figura 2 Grafo 1

2- Luego se estudió un segundo grafo donde se había analizado que en el grafo anterior, el resto de los asientos que salen por un punto hasta el pasillo deberían salir por otro punto igual que los primeros asientos, por tanto se generó un grafo de la siguiente manera:

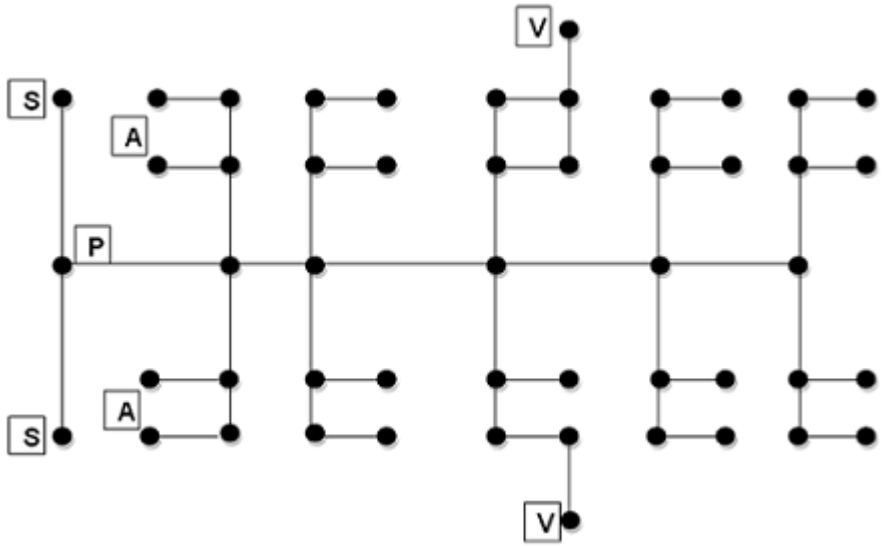


Figura 3 Grafo 2

3- Posteriormente se analizó que las distancias entre un asiento cuando un pasajero está sentado y cuando está parado es muy pequeña se desprecia, o sea las distancias muy cortas son despreciables, principalmente en el ACH que seguidamente se le aplicará a este problema, por tanto se creó un grafo que sería de la siguiente manera con solo la mitad del vagón.

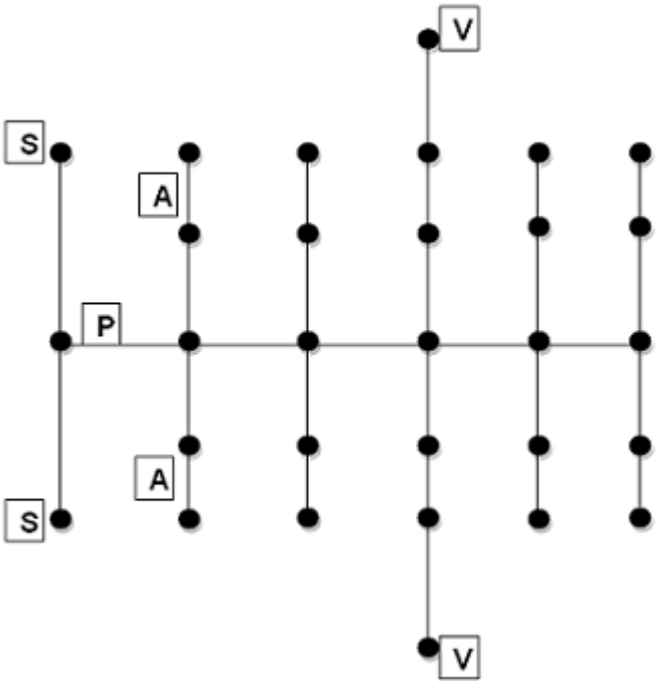


Figura 4 Grafo 3

4- Finalmente se llegó a la conclusión que se analizará el grafo para el vagón completo para un mejor funcionamiento del ACH y para que el usuario pueda tener una mejor visualización del problema. Además porque se ajusta al modelo de datos correspondiente donde el grafo queda más real para el problema.

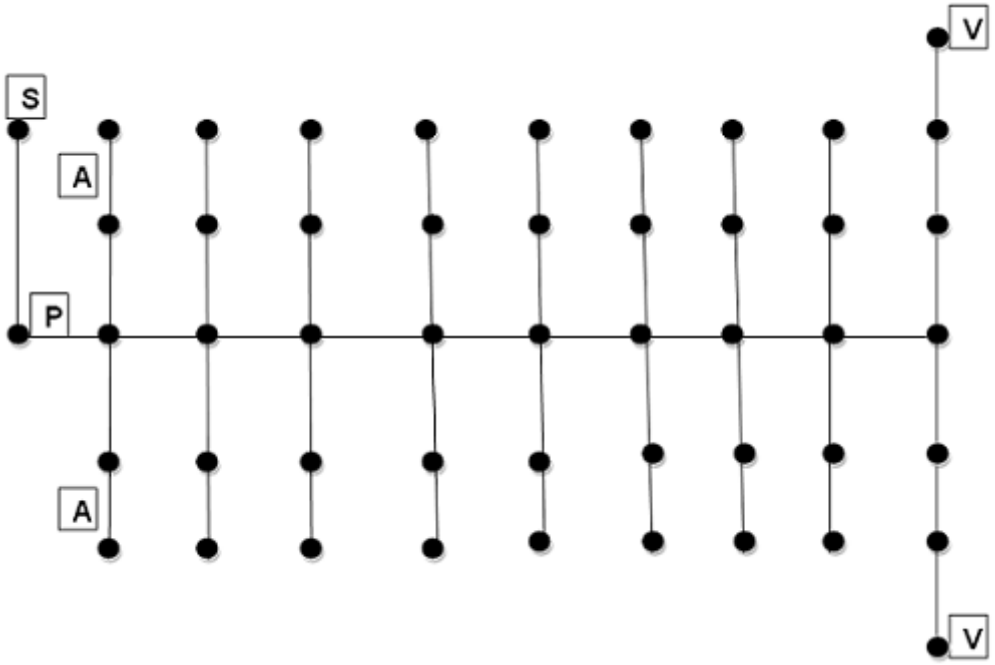


Figura 5 Grafo Final

P: Representa el pasillo del vagón.

A: Representa los asientos.

S: Representa las salidas.

V: Representa las ventanas.

2.3 Diferenciación de Casos

Inicialmente para la presente investigación se ha tomado como base un modelo de tren cuya capacidad del vagón es de 36 personas.

La capacidad del vagón del tren es de 36 personas.

La variable X define la cantidad de personas.

Se trabajará solo en un vagón.

Se asume que en la evacuación existirán diversas situaciones relacionadas con la cantidad de personas que se encuentren en el vagón en el momento del incidente, las cuales deben ser tomadas en cuenta en la simulación.

Casos

1. 0 < X <= 9 (muy pocas personas)
2. 9 < X <= 18 (pocas personas)
3. 18 < X <=27 (varias personas)
4. 27 < X < =36 (muchas personas)

Observaciones:

- Para 0 < X <= 9 No es necesario aplicar la simulación con este algoritmo si hay muy pocas personas en el vagón.
- Para 9 < X <=18 Se ejecuta el algoritmo sugiriendo que las personas estén sentadas equitativamente, o sea es más cómodo que el vagón del tren mantenga un equilibrio de personas para esta cantidad.
- Para 18 < X < =36 se ejecutaría el algoritmo.

Cuando se ejecute el algoritmo se debe tomar en cuenta un conjunto de situaciones como son:

1-Cada persona irá a la salida más cercana si está más factible de llegar que otra, es decir que no hallan obstáculos; por ejemplo: muchas personas tratando de salir por esa salida.

2- Siempre explorará esa salida si cumple con la situación anterior, si no cumple acudirá a otros nodos prometedores si los hay, sino esperará la próxima iteración.

3-Cada iteración es un movimiento de todas las hormigas en la colonia, o sea en este caso el movimiento de todas las personas en el vagón.

4-Si hay personas en el pasillo y aún queda personal en los asientos estos deberán esperar hasta que la cola se desocupe sin saltar de asientos ya que se corre el riesgo de fractura o posible accidente, además si el pasillo está ocupado es debido a que las salidas también estén ocupadas y si saltaría, igual tendrá que esperar a que se desocupe la salida por tanto para evitar un riesgo es mejor seguir el orden de las personas por el pasillo.

5-Los asientos, el pasillo y las salidas representan vértices.

6-Se define una función de acceso a cada nodo, o sea si hay 2 o 3 personas en un mismo nodo, no se podrá acceder a él, se buscarán otros nodos prometedores, de no existir se esperará la próxima iteración.

7- El objetivo no es esperar, es tratar de buscar la salida más rápida sin tener que esperar a que las personas salgan por una sola salida habiendo otras disponibles, o sea tratar que salgan la mayor cantidad de personas ocupando todas las salidas disponibles al mismo tiempo.

8- La longitud del vagón es de 25 metros y entre cada nodo hay una distancia de 0,5 metros.

9- El vagón del tren está trabajado por celdas y cada celda es un vector, la cantidad de celdas es 125, cada celda es una posición en el vagón, por tanto la cantidad de asientos es 36, las celdas de vector (1000, 1000, 1000) son las que están ocupadas por objetos.

-Modelo del vagón:

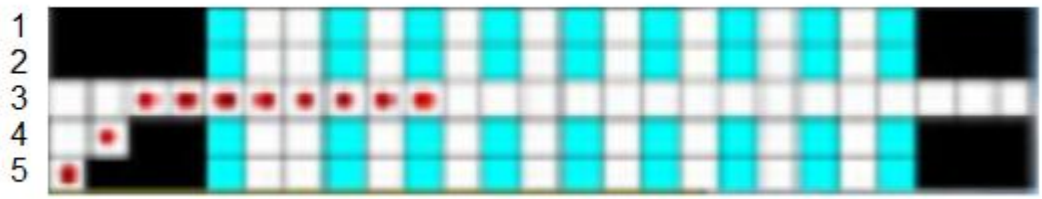


Figura 6 Modelo de datos

La información de cada una de estas celdas (vectores) se encuentran almacenadas en un fichero a continuación se muestra un ejemplo de algunas de ellas:

Empezando la enumeración comenzando por la fila 5 de izquierda a derecha hasta la fila 1 cada celda es un vector:

1.397 0.477 0.3 –1	9.161 0.484 0.3 –17
1000 1000 1000 –2	9.661 0.484 0.3 –18
1000 1000 1000 –3	10.161 0.484 0.3–19
1000 1000 1000 –4	10.661 0.484 0.3–20
3.159 0.484 0.3 –5	11.162 0.484 0.3–21
3.662 0.484 0.3 –6	11.662 0.484 0.3–22
4.162 0.484 0.3 –7	1000 1000 1000 –23
4.662 0.484 0.3 –8	1000 1000 1000 –24
5.162 0.484 0.3 –9	1000 1000 1000 –25
5.662 0.484 0.3 –10	1.134 0.984 0.3 –26
6.162 0.484 0.3 –11	1.638 0.984 0.3 –27
6.662 0.484 0.3 –12	1000 1000 1000 –28
7.162 0.484 0.3 –13	1000 1000 1000 –29
7.661 0.484 0.3 –14	3.159 0.984 0.3 –30
8.161 0.484 0.3 –15	3.662 0.984 0.3 –31
8.661 0.484 0.3 –16	4.162 0.984 0.3 –32

2.4 Distancias en el vagón del tren

Para hallar la distancia entre dos posiciones o sea entre cada nodo del grafo se hace mediante la fórmula de la distancia:

Sea: A (X_o, Y_o, Z_o) y B (X_l, Y_l, Z_l)

$AB = \sqrt{(X_o - X_I)^2 + (Y_o - Y_I)^2 + (Z_o - Z_I)^2}$

No todas las celdas son nodos, los nodos serán los que se analiza a continuación para hallar la distancia entre cada uno de ellos, a continuación se muestran algunos de estos cálculos:

→Entre la celda 5 y la celda 30 ocupan dos asientos y la distancia entre ellos es 0,5 la cual se halló mediante la fórmula expuesta quedando de la siguiente manera:

Cálculo entre los asientos del vagón del tren:

Vector 5 (3,159; 0,484; 0,3) Vector 30 (3,159; 0,984, 0,3)

$Vector5Vector30 = \sqrt{(3,159 - 3,159)^2 + (0,484 - 0,984)^2 + (0,3 - 0,3)^2}$

$Vector5Vector30 = \sqrt{0 + (-0,5)^2 + 0}$

$Vector5Vector30 = \sqrt{0,25}$

Vector5Vector30 = 0,5.

→Entre la celda 8 y 33

Vector 8 (4,662; 0,484; 0,3) Vector 33 (4,662; 0,984; 0,3)

$Vector8Vector33 = \sqrt{(4,662 - 4,662)^2 + (0,484 - 0,984)^2 + (0,3 - 0,3)^2}$

$Vector8Vector33 = \sqrt{0 + (-0,5)^2 + 0}$

Vector8Vector33 = 0,5.

Acudir al anexo 1 para conocer el resto de los cálculos

→La distancia entre el pasillo y los nodos de los asientos es de 0,5 metros.

Cálculo entre los nodos del pasillo:

→Entre la celda 55 y 58

Vector 55 (3,162; 1,484; 0,3) Vector 58 (4,662; 1,484; 0,3)

$Vector55Vector58 = \sqrt{(3,162 - 4,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

Vector55Vector58 = √ (-1,5)² + 0 + 0

Vector55Vector58 = √ 2,25 = 1,5.

Acudir al anexo 2 para conocer el resto de los cálculos

2.5 Optimización con Colonia de Hormigas (OCH)

Como se explicó en el capítulo anterior el ACH tiene grandes ventajas, una de estas, es que establece la solución con menos esfuerzo computacional y tiempo, sin deteriorar la calidad de la solución, se utiliza el tipo de dato abstracto grafo por lo que es más conveniente aplicar a estos tipos de problemas, como lo es la evacuación de pasajeros en tren, que cada persona ocupa un lugar diferente dentro del vagón, lo que significa que cada posición representará los nodos del grafo que utilizará el ACH.

Existen varios tipos de OCH algunos de estos son:

- Sistema de Hormigas (SH): Este algoritmo se aplicó al problema del viajante (TSP) que es uno de los problemas de optimización combinatoria más conocido, su única diferencia es en la actualización de la feromona.

$$\tau_{rs}(t) = (1-\rho)\cdot\tau_{rs}(t-1) + \sum_{k=1}^{m} \Delta\tau_{rs}^{k}$$

Este algoritmo fue la base para el desarrollo posterior de la OCH.

- El Sistema de Hormigas Elitista (SHE): Este algoritmo surgió a partir del anterior, la única diferencia entre ambos es en la regla de actualización que aplica un refuerzo adicional en los buenos arcos.

$$\tau_{rs}(t) = (1-\rho)\cdot\tau_{rs}(t-1) + \sum_{k=1}^{m} \Delta\tau_{rs}^{k} + e\cdot\Delta\tau_{rs}^{mejor\_global}$$

- El Sistema de Colonia de Hormigas (SCH): Este algoritmo extiende a su predecesor (SH) en tres aspectos:

  -La regla de transición establece un equilibrio entre la exploración de nuevos arcos y la explotación de la información acumulada.

-Para la actualización de la feromona sólo se considera la hormiga que generó la mejor solución hasta ahora. Sólo se evapora feromona en los arcos que componen ésta.

-Se añade una nueva feromona basada en que cada hormiga modifica automáticamente la feromona de cada arco que visita para diferenciar la búsqueda.

Por lo antes planteado de utilizó en el presente trabajo el sistema de colonia de hormigas (SCH).

Esquema general del funcionamiento de estos algoritmos:

```
Procedimiento metaheurísticaOCH
        Mientras criterio de parada no satisfecho
          ProgramarActividades
                construirSolucionesDeHormigas()
                actualizarFeromona()
                decisionesGlobales()
          Fin ProgramarActividades
        Fin Mientras
Fin metaheuríticaOCH
```

Figura 7 Seudocódigo de la metaheurísticaACO [33]

Inicialización: Primeramente se inicializan los parámetros del algoritmo.
Una vez concluida la inicialización, se repite una serie de acciones hasta que se cumpla una condición de parada predefinida en dependencia del problema que se esté resolviendo
ConstruirSolucionesDeHormigas: m hormigas construyen soluciones al problema de manera asíncrona e incremental. Aunque es dependiente del algoritmo OCH en particular, este procedimiento consiste en la adición de una nueva componente a la solución de construcción. Las hormigas se mueven aplicando una regla de decisión probabilística que hace uso de la información heurística y los rastros de feromonas disponibles.
ActualizarFeromona: Se actualiza la feromona en cada iteración de una hormiga. Los rastros de feromona pueden incrementarse al ser depositada por las hormigas o pueden disminuir mediante un proceso de evaporación. Esta evaporación constituye un mecanismo que evita una rápida

convergencia a óptimos locales, favoreciendo la exploración de nuevas áreas del espacio de búsqueda.

DecisionesGlobales: Opcionalmente se llevan a cabo otras acciones centralizadas que no pueden ser ejecutadas por las hormigas individuales, que los autores denominan “acciones del demonio”. Estas pueden manejar información global y permiten la implementación de mejoras del algoritmo, como pueden ser búsquedas locales para refinar una solución obtenida.

Este esquema permite suficiente flexibilidad para el desarrollo de algoritmos OCH aplicables a una amplia gama de problemas.

Para llevar a cabo la simulación se parte de un software elaborado anteriormente, que simula la ejecución del ACH el cual será adaptado a las necesidades del problema actual.

El software inicial tiene las siguientes características:

1. La aplicación permite agregar puntos (nodos) aleatorios en el formulario.
2. Conecta los nodos en el grafo todos con todos.
3. Busca el camino mínimo del primer punto (nodo) que pintó hasta el último.
4. Luego de haber llegado al punto (nodo) final dibuja una conexión entre ese punto (nodo) final y el nodo inicial.
5. Solo visita al punto (nodo) una sola vez.
6. Visita todos los puntos (nodos) hasta llegar al último punto (nodo) con el menor camino posible guiándose por una fórmula de manera probabilística.

A partir de esas características y las necesidades concretas del problema a resolver se realizaron varias adecuaciones prácticas en la aplicación:

1. Muestra un modelo de datos con los puntos del vagón del tren, además permite agregar puntos (nodos) aleatorios en el formulario hasta que el usuario desee y además tiene la opción de asignarle la heurística a cada nodo y si es o no un nodo final. Esta información se carga desde un fichero.

2. Permite definir la conexión en el grafo entre nodos hasta que el usuario desee establecer, guardando la información en una matriz de aristas.
3. Busca el camino mínimo desde cualquier punto hasta un punto (nodo) de salida
4. La hormiga termina su recorrido cuando encuentra una salida.
5. Puede visitar cada nodo más de una vez.
6. Visita todos los puntos (nodos) hasta llegar al último punto (nodo) con el menor camino posible guiándose por una fórmula de manera probabilística.

Para darle cumplimiento al objetivo del problema se elaboró una serie de pasos y procedimientos por el cual es importante regirse a la hora de la implementación del ACH para el problema actual.

→Pasos para aplicar el algoritmo:

Para aplicar el algoritmo en la solución de un problema, se debe guiar por una serie de pasos que serán detallados a continuación:

1. Representar el problema a través de un grafo ponderado que será recorrido por las hormigas para construir soluciones.
2. Definir de una manera apropiada el significado de los rastros de feromonas, esto es el tipo de decisión que inducen. Este es un paso decisivo en la implementación de un algoritmo de OCH.
3. Poner pesos en la información heurística en cada nodo o arco que debe tomar una hormiga mientras que construye una solución. Este también es un paso decisivo para la implementación del algoritmo.
4. Si es posible desarrollar algoritmos que permitan realizar optimizaciones locales a fin de mejorar el rendimiento, porque el resultado de muchas aplicaciones de la OCH demuestran que el mejor rendimiento se alcanza cuando se complementan con optimizaciones locales.
5. Escoger un algoritmo OCH específico y aplicarlo al problema que hay que solucionar teniendo en cuenta todos los aspectos desarrollados.

6. Refinar los parámetros del algoritmo de OCH, o sea, se deben obtener los valores de influencia de feromona (alfa), información heurística (beta) y el número de hormigas óptimos, para el problema en específico.

→Estructura y procedimiento del algoritmo

- Estructura

La aplicación del ACH permitirá encontrar el camino más corto desde un nodo hasta alguna de las salidas disponibles, para la solución de este resultado se proponen funcionalidades como:

-Agregar puntos (cada punto representa un vértice en el grafo).

-Pintar las aristas de los puntos, o sea el usuario puede establecer la conexión que desee entre los puntos.

-Bloquear la escena, no se permitirá agregar nuevos puntos.

-Ejecutar el algoritmo.

- Modificar variables del algoritmo.

- Reiniciar la escena.

- Procedimientos

1. El algoritmo utiliza una matriz de feromonas para la construcción de soluciones potenciales, la cual estará inicializada en la información de las hormigas además de su influencia de feromonas (alfa), tasa de evaporación de la feromona y la concentración inicial de feromonas en cada arista representado por una variable siendo esta mayor que cero. La matriz de feromonas representa la cantidad de feromona que se va almacenando en cada par de nodos (i, j).

2. El algoritmo utiliza una matriz de distancias para guardar las distancias a recorrer del nodo i al nodo j, en este tipo de problema las distancias son constante.

3. La probabilidad de escoger el nodo j estando en el nodo i está definida por la siguiente expresión:

$$p_k(i,j) = \begin{cases} \dfrac{[t_{i,j}]^{\alpha} * [n_{i,j}]^{\beta}}{\sum_{h \in J_k(i)} [t_{ih}]^{\alpha} * [n_{ih}]^{\beta}} & \text{si} \quad j \in J_k(i) \\ 0 & \text{en otro caso} \end{cases}$$

Donde:

$p_k(i, j)$ es la probabilidad de que la hormiga K se mueva del nodo *i* al *j*.

$t_{i,j}$ es la construcción de feromonas que hay en la arista a_{ij}.

$J_k(i)$ es el conjunto de nodos alcanzables desde *i* no visitados aún por la hormiga *k*.

$n_{i,j}$ es la información heurística de la arista a_{ij}.

α es la importancia concedida a las feromonas en las aristas.

β es la importancia concedida a la heurística de la distancia entre las aristas.

Por lo tanto la probabilidad de que un nodo sea elegido está en función de qué tan cerca del nodo se está y que cantidad de feromona existe en ese camino. Además, se puede determinar cuál de ellos tiene un peso mayor al ajustar los parámetros α y β.

La heurística está definida de la siguiente manera:

- Un valor que se encuentra en el rango de 0.1 a 0.9
- Los nodos más cerca de la salida tendrán menor heurística que los restantes nodos.
- Los nodos de la salida tendrán una heurística de 0.1 al igual que los nodos del pasillo más cercanos a la salida.

- El resto de los nodos tendrán una heurística a partir de 0.2 en adelante según la posición del nodo.

4. Actualización de feromonas: Después de generar una solución, se evapora y deposita, un cierto valor de feromonas. Para actualizar las feromonas en las aristas, se utiliza una fórmula que esta expresada de la siguiente manera:

$$t(i,j) = pt(i,j) + \sum_{k-1}^{m} \Delta t_k(i,j)$$

$$\triangle t_k(i,j) = \begin{cases} \frac{1}{L_k} & \text{si la hormiga } k \text{ ha visitado la arista } a_{ij} \\ 0 & \text{en otro caso} \end{cases}$$

Donde:

pt (i, j) es la multiplicación entre la concentración que hay en la arista a_{ij} y *p* que es el coeficiente de evaporación de feromonas, *p* será un número entre 0 y 1. A valores más pequeños de *p,* más rápido será la evaporación de feromonas.

$\sum_{k-1}^{m} \Delta t_k(i,j)$ Es la cantidad de feromonas depositada por la hormiga *k* en la arista, es definido por L_k que es la longitud del camino creado por esta hormiga. Por tanto, viajes cortos se traducirá en mayores niveles de feromona depositada en las aristas.

Funcionamiento del ACH:

1. Buscar un camino desde el nodo inicial hasta el nodo final o los nodos finales.

2. Cada hormiga construye una solución al problema:

La hormiga posee:

Nodo actual; Nodo siguiente; Inicio camino (se cuentan los nodos por las que van transitando); Longitud (la longitud del camino); Nodos visitados (es un arreglo de los nodos que han sido visitados por cada hormiga); Camino (arreglo donde se va guardando el camino que va creando cada hormiga).

3. Para calcular la distancia entre los puntos se hace con la fórmula de distancia anteriormente explicada.

4. Primeramente se inicializa la colonia de hormigas:

    - Se crea y se llena la lista de hormigas (información).

    - Se inicializa la matriz de distancia y feromonas.

    - Las hormigas partirán desde el nodo inicial.

    - Carga el nodo actual donde está la hormiga en la lista de los nodos visitados.

    - Se ejecuta el algoritmo donde:

      Se obtiene el siguiente nodo:

        1. Se recorre los nodos en la lista de puntos.

        2. Si hay una conexión entre el nodo actual donde está la hormiga que está siendo analizada entre el siguiente nodo se calcula la fórmula de la probabilidad.

        3. Si el próximo nodo que encontró es un nodo final, termina el recorrido sino está visitado ese nodo se guarda para que sea el próximo nodo a visitar.

        4. En caso de que el nodo estuviera visitado y no fuera nodo final, se analiza entonces los que están visitados.

      - Se actualiza los valores del camino del siguiente nodo

      - Se actualiza las aristas con la feromona.

5- Finalmente si se halló un camino se reinicia la matriz y los demás valores del algoritmo. Si el problema se resolvió es porque ya no hay más iteraciones.

2.6 Diagrama del funcionamiento del algoritmo

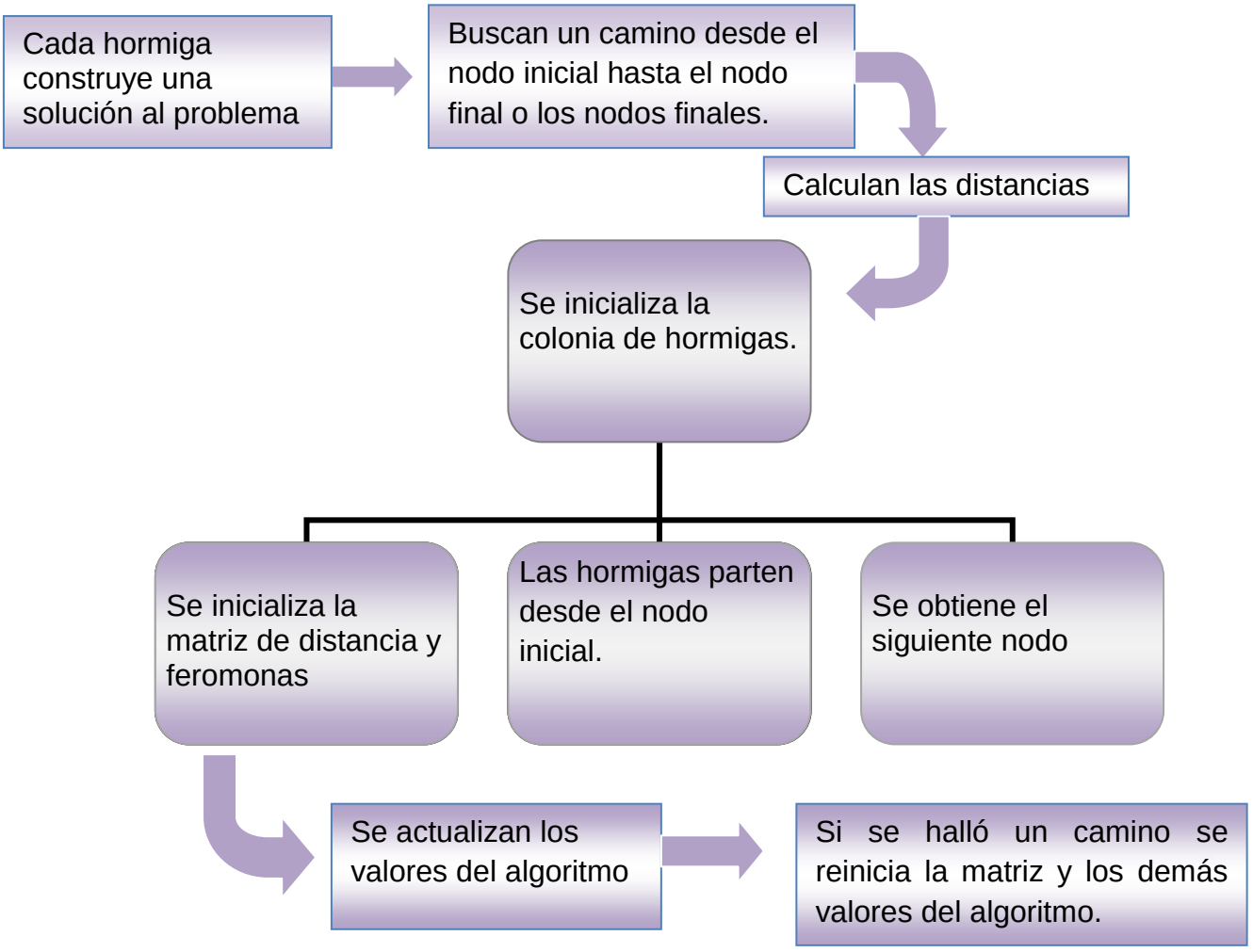


Figura 8 Diagrama del funcionamiento del algoritmo

2.7 Resultados y Discusión

Como parte del proceso de investigación llevado a cabo se determinó que algunas de las ideas iniciales propuestas para aplicar el ACH para este problema no pudieron ser aplicadas por varias razones. Cada hormiga no puede representar una persona, ya que en cada iteración del ACH solo mostrarán el camino que debe seguir una única persona. Inicialmente se había analizado distribuir las hormigas por cada uno de los nodos, pero se llegó a la conclusión que solo se necesitaría un camino desde un nodo inicial hasta cualquiera de los nodos finales llevando a la necesidad de ubicar todas las hormigas en el nodo inicial para que ellas pudieran buscar ese camino mínimo, entonces las personas seguirán ese camino marcado por las hormigas.

El grafo obtenido del modelo del vagón del tren limita algunas de las ventajas del algoritmo por su poca conectividad entre nodos ya que mientras más conexiones se tengan en el grafo las hormigas tendrán más posibilidades de buscar la mejor solución.

En el algoritmo, cada hormiga encuentra una solución de forma independiente a otras de la colonia, además ya se había determinado que las hormigas no representaban personas, por tanto el algoritmo no permite saber si en el momento de encontrar el nodo siguiente existen o no otras hormigas en dicho nodo, esto pudiera lograrse a partir de un nuevo algoritmo derivado de la metaheurística colonia de hormigas lo cual quedaría propuesto para futuras investigaciones.

CAPÍTULO 3: VALIDACIÓN DE LA PROPUESTA

En este capítulo se realizará todo el proceso de desarrollo aplicando la metodología seleccionada como respaldo al software desarrollado la cual consta de cuatro fases. Dentro de estas fases se presenta cada una de las tareas de ingeniería generadas por cada historia de usuario, la planificación, donde se realiza un plan de iteraciones y se seleccionan las historias de usuario que serán desarrolladas durante el transcurso de la iteración, el diseño y las pruebas de aceptación realizada a cada tarea de ingeniería.

3.1 Selección de la metodología

Se selecciona la metodología XP (Extreme Programming) debido a que es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software propiciando un buen clima de trabajo. XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, se define como especialmente adecuada para proyectos muy cambiantes. Se basa en la simplicidad, la comunicación y el reciclado continuo de código, o sea aplicar una pura lógica. Los objetivos de XP son muy simples se basa en la satisfacción al cliente y en potenciar al máximo el trabajo en grupo, XP es liviana y ágil, está orientada más a las personas que a los procesos. Los individuos e interacciones son más importantes que los procesos y herramientas. El individuo es el principal factor de éxito de un proyecto software. El software que funcione es más importante que documentación exhaustiva. La respuesta ante el cambio es más importante que el seguimiento de un plan. La colaboración con el cliente es más importante que la negociación de contratos. Un solo desarrollador con apoyo de un segundo. Cliente cercano para poder intercambiar y revisar el avance sistemáticamente. [33]

3.2 Exploración

La metodología de desarrollo XP comienza con su fase de exploración, los clientes detallaron las historias de usuario que el sistema debía poseer y que son de beneficio para la inicial entrega del producto. El equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. La fase de exploración consta de poco tiempo según la familiaridad que tengan los programadores con la tecnología.

3.2.1 Historia de Usuario

Las historias de usuario son la técnica utilizada para especificar los requisitos del software sean funcionales o no funcionales, representar una breve descripción del comportamiento del sistema, donde se emplea la terminología del cliente sin lenguaje técnico.

Se realiza una historia de usuario por cada funcionalidad del sistema y cada una de estas deben ser lo suficientemente clara y determinada para que un programador pueda desarrollarla, se emplean para hacer estimaciones de tiempo de desarrollo de la parte de la aplicación que se describe. Las funcionalidades reflejadas en las historias de usuarios deben ser programadas en un tiempo que lo define el propio equipo del proyecto que debe ser entre 1 y 3 semanas.

Basado en las necesidades del proyecto y un estudio previo se propone a continuación la siguiente estructura para la descripción de la historia de usuarios:

Historia de Usuario	
Número:	**Usuario:**
Nombre:	
Prioridad en el negocio:	**Riesgo de desarrollo:**
Puntos de Estimación:	**Iteración:**
Programador Responsable:	
Descripción:	
Observaciones:	

Tabla 1 Estructura de Historia de Usuarios

- Número: Número de la historia de usuario (identificador).
- Usuario: Usuario del sistema que realiza la acción (creador).
- Nombre: Nombre de la historia de usuario.
- Prioridad en el negocio: Se refiere a la importancia que tiene la historia de usuario para el cliente o el equipo de desarrollo la cual se define en la escala de media, alta o baja.
- Riesgo en desarrollo: Nivel de dificultad para desarrollar la historia de usuario por el desarrollador.

- Puntos de Estimación: Es el tiempo estimado para realizar la actividad la cual la establecen los programadores y varían en dependencia de la complejidad de la historia de usuario.
- Iteración: Iteración a la que pertenece.
- Programador Responsable: El encargado de programar la actividad.
- Descripción: Se describe en que consiste la historia de usuario teniendo en cuenta las acciones de los usuarios y la respuesta del sistema.
- Observaciones: Información de interés en caso que sea necesaria agregar para un mejor entendimiento de la historia de usuario.

Historias de usuarios a desarrollar:

1. Mostrar el Modelo de Datos en la escena.
2. Pintar los caminos en el Modelo de Datos.
3. Ejecutar el algoritmo.
4. Mostrar formulario de opciones.
5. Modificar parámetros de las opciones.
6. Reiniciar escena.

Historia de Usuario	
Número: 1	**Usuario:** Todos
Nombre: Mostrar el Modelo de Datos en la escena	
Prioridad en el negocio: Alta	**Riesgo de desarrollo:** Media
Puntos de Estimación: 1	**Iteración:** 1
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario selecciona la opción Leer Puntos. El sistema muestra el modelo de datos en la escena.	
Observaciones: El sistema lee la información desde un fichero y luego muestra esa información en la escena que es el modelo de datos del tren.	

Tabla 2 Historia de Usuario: Mostrar el Modelo de Datos en la escena

Historia de Usuario	
Número: 2	**Usuario:** Todos
Nombre: Pintar los caminos en el Modelo de Datos	

Prioridad en el negocio: Alta	**Riesgo de desarrollo:** Baja
Puntos de Estimación: 1	**Iteración:** 1
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario pinta todos los caminos que desee establecer en el modelo de datos.	
Observaciones: Si el usuario no desea establecer una conexión determinada se le da un clic a parte en la escena y luego puede seguir pintando las conexiones o no que desee pintar.	

Tabla 3 Historia de Usuario. Pintar los caminos en el Modelo de Datos

Historia de Usuario	
Número: 3	**Usuario:** Todos
Nombre: Ejecutar el algoritmo	
Prioridad en el negocio: Alta	**Riesgo de desarrollo:** Alta
Puntos de Estimación: 3	**Iteración:** 2
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario selecciona la opción ejecutar el algoritmo. El sistema mostrará el grafo resultante del proceso.	
Observaciones:	

Tabla 4 Historia de Usuario. Ejecutar el algoritmo

Historia de Usuario	
Número: 4	**Usuario:** Todos
Nombre: Mostrar formulario de opciones	
Prioridad en el negocio: Media	**Riesgo de desarrollo:** Baja
Puntos de Estimación: 1	**Iteración:** 1
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario selecciona la opción Opciones. El sistema mostrará el formulario con los parámetros correspondientes	
Observaciones: Los parámetros serán visualizados con sus valores originales.	

Tabla 5 Historia de Usuario. Mostrar formulario de opciones

Historia de Usuario	
Número: 5	**Usuario:** Todos
Nombre: Modificar parámetros de las opciones	
Prioridad en el negocio: Media	**Riesgo de desarrollo:** Media
Puntos de Estimación: 1	**Iteración:** 2
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario introduce los valores deseados para cada parámetro y selecciona la opción aceptar. El sistema ejecuta los cambios y regresa a la interfaz anterior.	
Observaciones:	

Tabla 6 Historia de Usuario. Modificar parámetros de las opciones

Historia de Usuario	
Número: 6	**Usuario:** Todos
Nombre: Reiniciar escena	
Prioridad en el negocio: Baja	**Riesgo de desarrollo:** Media
Puntos de Estimación: 1	**Iteración:** 2
Programador Responsable: Lienni Daisson Columbié	
Descripción: El usuario selecciona la opción Reiniciar escena. El sistema limpia la escena borrando el modelo de datos y las conexiones creadas.	
Observaciones:	

Tabla 7 Historia de Usuario. Reiniciar escena

3.3 Planificación de la Entrega

En esta fase el cliente define la prioridad que posee cada historia de usuario, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Las estimaciones de esfuerzo asociado a la implementación de las historias de usuario son establecidas por los programadores utilizando como medida la cantidad de puntos por lo que un punto equivale a una semana de programación y generalmente las historias de usuario valen de 1 a 3 puntos. Se toman acuerdos sobre el contenido y plazo de la primera entrega modelándolo en un cronograma, estas actividades son realizadas en conjunto con el cliente. Las planificaciones serán respetadas por todos los implicados en el desarrollo y permitirán un desarrollo ágil y eficiente de la aplicación.

3.3.1 Estimación de esfuerzo por historias de usuario

En la siguiente tabla se muestran los resultados de la estimación realizada con cada una de las historias de usuarios determinadas para la elaboración del sistema propuesto:

Historias de Usuario	Puntos de Estimación
Mostrar el Modelo de Datos en la escena	1
Pintar los caminos en el Modelo de Datos	1
Ejecutar el algoritmo	3
Mostrar formulario de opciones	1
Modificar parámetros de las opciones	1
Reiniciar escena	1

Tabla 8 Estimación de esfuerzo por historias de usuario

3.3.2 Planificación de iteraciones

Después de haber identificado las historias de usuario por los clientes y la estimación de esfuerzo dedicado a cada una de ellas por los desarrolladores se procede a realizar tantas iteraciones como se consideren necesarias definiendo cuales serán desarrolladas en cada iteración del proceso de implementación. Se decide implementar el sistema en dos iteraciones:

- Primera Iteración:

  Para esta iteración se seleccionaron las historias de usuario más básicas del sistema y las que permiten modelar la estructura de la aplicación. Al concluir dicha iteración se contará con una nueva versión del producto (1.0), las historias de usuario para esta iteración son:

  - Mostrar el Modelo de Datos en la escena
  - Pintar los caminos en el Modelo de Datos
  - Mostrar formulario de opciones

- Segunda Iteración:

  En esta última iteración se seleccionaron las historias de usuario con mayor peso en la aplicación. Al concluir esta iteración se obtendrá el producto en su versión (1.1). Al culminar esta iteración la aplicación podrá ser sometida a diversos procesos de prueba para comprobar su eficiencia y desempeño, las historias de usuario para esta iteración son:

  - Ejecutar el algoritmo

- Modificar parámetros de las opciones
- Reiniciar escena

3.3.3 Plan de duración de iteraciones

Se propone la realización de un plan de duración de iteraciones con el fin de ilustrar la planificación temporal de las diferentes iteraciones y las historias de usuario que serán implementadas en cada una de ellas.

Iteración	Historias de Usuario	Duración total de iteraciones
1	Mostrar el Modelo de Datos en la escena Pintar los caminos en el Modelo de Datos Mostrar formulario de opciones	3 semanas
2	Ejecutar el algoritmo Modificar parámetros de las opciones Reiniciar escena	5 semanas

Tabla 9 Plan de duración de iteraciones

3.3.4 Plan de entregas

A continuación se muestra el plan de entregas la cual es una planificación donde los desarrolladores y los clientes establecen los períodos de implementación a través de las historias de usuario, clasificando las mismas según sus prioridades. Se establece efectuar dos entregas estas son:

# Iteración	Duración de la iteración	Fecha de inicio	Fecha de fin
1	3 semanas	12/2/2012	4/3/2012
2	5 semanas	4/3/2012	8/4/2012

Tabla 10 Plan de entregas

3.4 Diseño de la Aplicación

La metodología XP se guía por técnicas como las tarjetas CRC (Clases, Responsabilidad y Colaboración) la cual permite al programador centrarse y apreciar el desarrollo. Una clase es cualquier persona, objeto, evento, concepto, pantalla o reporte. Las responsabilidades de una clase son los

elementos que conoce y las acciones que realiza sobre ellos, sus atributos y métodos. Los colaboradores de una clase son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades.

3.4.1 Tarjetas C.R.C

Las tarjetas C.R.C reducen al mínimo la complejidad del diseño. Esto logra que el programador se centre en el diseño de las bases fundamentales de la clase. Permiten que el equipo completo contribuya en la tarea del diseño. Se utilizan para estructurar las clases y definir las responsabilidades sobre las mismas, así como la simulación de escenarios en el sistema.
Se propone la utilización de la siguiente plantilla:

Tarjeta C.R.C	
Clase: *Nombre de la clase que se está modelando*	
Súper Clase: *Nombre de la clase padre en la herencia*	
Sub Clase(s): *Nombre de la(s) clase(s) hija en la herencia*	
Responsabilidades: *Es una descripción del propósito de la clase.*	**Colaboraciones:** *Indica con cuáles otras clases se requiere relación para cumplir la responsabilidad*.

Tabla 11 Plantilla de Tarjeta C.R.C

Tarjeta C.R.C	
Clase: CHormiga	
Súper Clase:-	
Sub Clase(s): -	
Responsabilidades: Permite la creación de las hormigas, con sus nodos visitados, el camino recorrido y la longitud total de su camino.	**Colaboraciones:-**

Tabla 12 Tarjeta CRC: CHormiga

Tarjeta C.R.C	
Clase: CColoniaHormigas	
Súper Clase:-	
Sub Clase(s): -	
Responsabilidades: Permite la creación de la colonia de hormigas. Simula las hormigas, actualiza las aristas y reinicia la colonia.	**Colaboraciones:** Point Chormiga QList

Tabla 13 Tarjeta CRC: CColoniaHormigas

Tarjeta C.R.C	
Clase: Point	
Súper Clase: QGraphicsEllipseItem	
Sub Clase(s): -	
Responsabilidades: Permite la creación de los puntos, guardando sus coordenadas y los valores para mostrarlos.	**Colaboraciones:** QObject

Tabla 14 Tarjeta CRC: Point

Tarjeta C.R.C	
Clase: Scene	
Súper Clase: QGraphicsScene	
Sub Clase(s): -	
Responsabilidades: Permite la creación de la escena y es donde se muestran los nodos y se pintan los caminos.	**Colaboraciones:** ccoloniahormigas Heuristica QList QPainter Point VArtTrainCognitiveModel QGraphicsSceneMouseEvent Qmath QDebug

Tabla 15 Tarjeta CRC: Scene

Tarjeta C.R.C	
Clase: View	
Súper Clase: QGraphicsView	
Sub Clase(s): -	
Responsabilidades: Permite la creación de un widget para mostrar el contenido de la escena.	**Colaboraciones:-**

Tabla 16 Tarjeta CRC: View

Tarjeta C.R.C	
Clase: VArtTrainCognitiveModel	
Súper Clase:-	
Sub Clase(s): -	
Responsabilidades: Permite leer y mostrar la información del modelo de datos que está guardado en un fichero.	**Colaboraciones:** OgreVector3 Vector

Tabla 17 Tarjeta CRC: VArtTrainCognitiveModel

Tarjeta C.R.C	
Clase: OgreVector3	
Súper Clase:-	
Sub Clase(s): -	
Responsabilidades: Permite que se muestre la información de los vectores que están en el fichero.	**Colaboraciones:** OgrePrerequisites OgreMath OgreQuaternion

Tabla 18 Tarjeta CRC: OgreVector3

3.4.2 Interfaces de Usuario

La aplicación desarrollada dispone de diferentes funcionalidades, a continuación se muestran las interfaces de usuario para cada una de ellas.

Al ejecutar la aplicación se mostrará la interfaz inicial y sus diferentes funcionalidades:

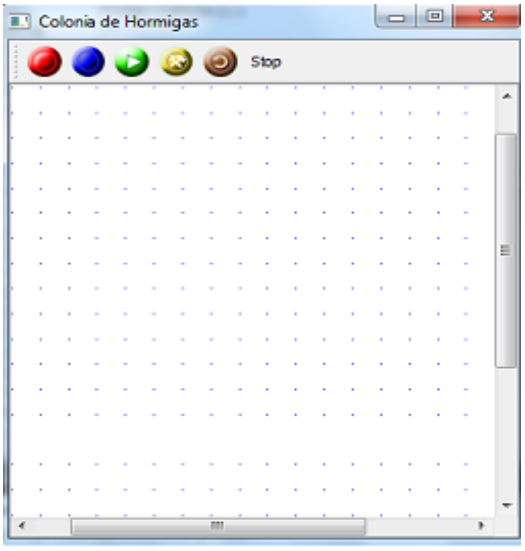


Figura 9 Interfaz inicial de la aplicación

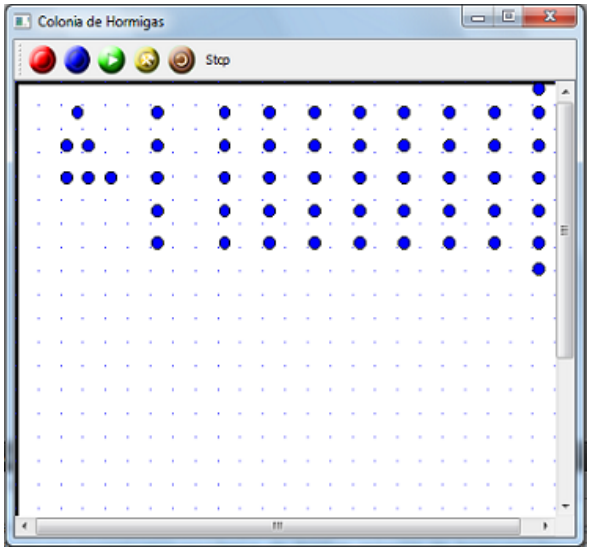


Figura 10 Interfaz Leer Puntos

→Leer Puntos: Lee los puntos del modelo de datos que están guardado en el fichero y lo muestra en la interfaz. Ver figura 3.2

→Agregar puntos: Si el usuario no desea trabajar sobre el modelo de datos la aplicación le permite agregar puntos aleatorios para diseñar un grafo diferente. Ver figura 3.3

→Ejecutar: Ejecuta la funcionalidad principal del algoritmo dando paso a la solución y mostrando el camino final. Ver figura 3.4

→Opciones: Muestra un nuevo formulario donde posibilita la modificación de los diferentes parámetros del algoritmo. Ver figura 3.5

→Reiniciar: Reinicia la escena y todos los valores para comenzar a ejecutar el algoritmo nuevamente.

→Stop: No permite agregar más puntos en el formulario, esto es solo si el usuario ejecuta la opción agregar puntos.

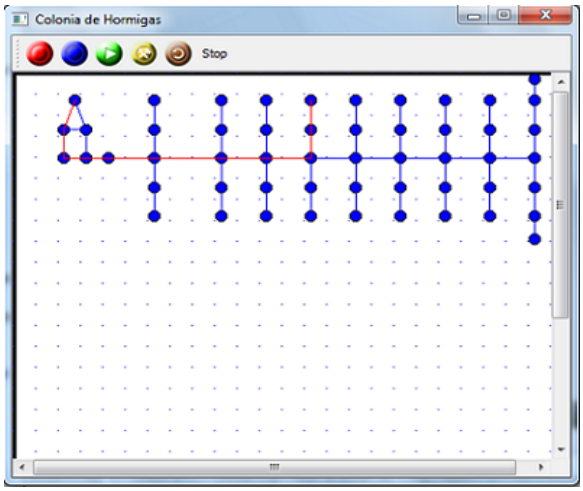


Figura 11 Interfaz ejecutar

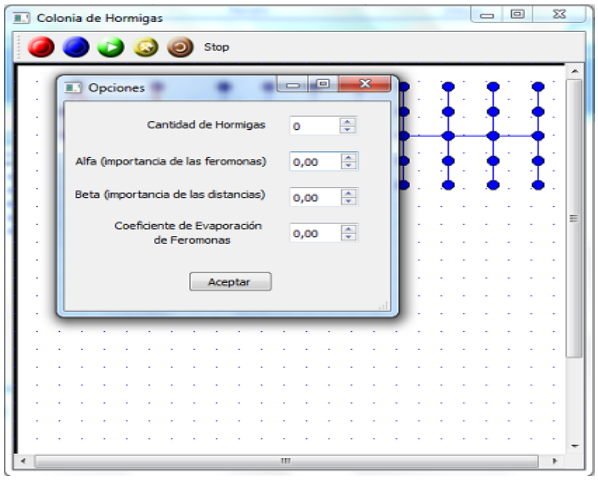


Figura 12 Interfaz Opciones

3.5 Implementación y Prueba

En la fase de Planificación se detallaron las historias de usuarios agrupadas en cada iteración. En esta fase se descomponen las historias de usuarios en tareas de ingeniería. Las tareas de ingeniería se efectúan para detallar mejor las historias de usuario, facilitando con ellos el entendimiento en el proceso de implementación. Cada historia de usuario puede poseer una o más tareas de ingeniería en caso de ser necesario, explicando paso a paso las acciones que se realizan en la misma. A partir de la planificación efectuada se llevaron a cabo dos iteraciones de desarrollo sobre la aplicación ilustrativa, lo que permite que al final de la última iteración obtener una aplicación ejecutiva cumpliendo con todas las funcionalidades definidas. A continuación se detallan cada una de las iteraciones efectuadas.

3.5.1 Primera Iteración

En esta iteración se desarrollaron las funcionalidades básicas de la aplicación que darán soporte a las funciones más complejas de la segunda iteración.

Historia de Usuarios	Estimado	Real
Mostrar el Modelo de Datos en la escena	1	1
Pintar los caminos en el Modelo de Datos	1	1
Mostrar formulario de opciones	1	1

Tabla 19 Historias de Usuarios desarrolladas en la primera iteración

Tareas de ingeniería de las historias de usuario (HU) correspondiente a esta iteración:

Tarea de Ingeniería	
No. De la tarea: 1	**No. De la HU:** 1
Nombre de la tarea: Crear interfaz visual de la aplicación	
Tipo de tarea: Diseño	**Puntos estimados:** 0.3
Fecha inicio: 12/2/2012	**Fecha fin:** 14/2/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Define el diseño de la pantalla que mostrará la escena donde el usuario mostrará el modelo de datos.	

Tabla 20 Tarea de Ingeniería: Crear interfaz visual de la aplicación

Tarea de Ingeniería	
No. De la tarea: 2	**No. De la HU:** 1
Nombre de la tarea: Mostrar el Modelo de Datos en la escena	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 15/2/2012	**Fecha fin:** 19/2/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Implementación de las clases y métodos necesarios para mostrar el modelo de datos en la escena.	

Tabla 21 Tarea de Ingeniería: Mostrar el Modelo de Datos en la escena

Tarea de Ingeniería	
No. De la tarea: 3	**No. De la HU:** 2
Nombre de la tarea: Pintar los caminos en el Modelo de Datos	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 20/2/2012	**Fecha fin:** 25/2/2012
Programador Responsable: Lienni Daisson Columbié	

Descripción: Implementación de las clases y métodos necesarios para pintar los caminos en el modelo de datos.

Tabla 22 Tarea de Ingeniería: Pintar los caminos en el Modelo de Datos.

Tarea de Ingeniería	
No. De la tarea: 4	**No. De la HU:** 4
Nombre de la tarea: Crear interfaz visual para el formulario de opciones	
Tipo de tarea: Diseño	**Puntos estimados:** 0.3
Fecha inicio: 26/2/2012	**Fecha fin:** 28/2/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Define el diseño de la pantalla que mostrará el formulario donde se podrán modificar diversos valores del algoritmo.	

Tabla 23 Tarea de Ingeniería: Crear interfaz visual para el formulario de opciones

Tarea de Ingeniería	
No. De la tarea: 5	**No. De la HU:** 4
Nombre de la tarea: Mostrar formulario de opciones	
Tipo de tarea: Desarrollo	**Puntos estimados:** 0.5
Fecha inicio: 29/2/2012	**Fecha fin:** 4/3/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Implementa la funcionalidad para mostrar el formulario opciones a partir de un botón en la interfaz principal.	

Tabla 24 Tarea de Ingeniería: Mostrar formulario de opciones

3.5.2 Segunda Iteración

En esta iteración se desarrollaron las funcionalidades de mayor peso para culminar el desarrollo de la iteración.

Historia de Usuarios	Estimado	Real
Ejecutar el algoritmo	3	3

Modificar parámetros de las opciones	1	1
Reiniciar escena	1	1

Tabla 25 Historias de Usuarios desarrolladas en la segunda iteración

Tareas de ingeniería de las historias de usuario correspondientes a esta iteración:

Tarea de Ingeniería	
No. De la tarea: 1	**No. De la HU:** 3
Nombre de la tarea: Ejecutar el Algoritmo	
Tipo de tarea: Desarrollo	**Puntos estimados:** 3
Fecha inicio: 4/3/2012	**Fecha fin:** 24/3/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Implementación de las clases y funcionalidades necesarias para ejecutar el algoritmo y mostrar el camino resultante.	

Tabla 26 Tarea de Ingeniería: Ejecutar el Algoritmo

Tarea de Ingeniería	
No. De la tarea: 2	**No. De la HU:** 5
Nombre de la tarea: Modificar parámetros de las opciones	
Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 25/3/2012	**Fecha fin:** 31/3/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Implementación de las funcionalidades necesarias para modificar las variables correspondientes.	

Tabla 27 Tarea de Ingeniería: Modificar parámetros de las opciones

Tarea de Ingeniería	
No. De la tarea: 3	**No. De la HU:** 6
Nombre de la tarea: Reiniciar escena	

Tipo de tarea: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 1/4/2012	**Fecha fin:** 8/4/2012
Programador Responsable: Lienni Daisson Columbié	
Descripción: Implementación de las funcionalidades para reiniciar la escena, así como todos los valores necesarios para ejecutar el algoritmo desde el inicio.	

Tabla 28 Tarea de Ingeniería: Reiniciar escena

3.5.3 Pruebas de Aceptación

A continuación se muestran las pruebas de aceptación realizadas a las diferentes tareas de ingeniería, estas se realizan para comprobar si cada tarea funciona correctamente y en concordancia con el propósito que se desarrolló.

Prueba de Aceptación	
Código:	**No.Historia de Usuario:**
Nombre:	
Descripción:	
Condiciones de Ejecución:	
Entrada:	
Resultado Esperado:	
Evaluación de la Prueba:	

Tabla 29 Estructura de la Prueba de Aceptación

- Código: Índice del caso de prueba. Es un número único que se le asigna a cada caso de prueba que pertenece a una historia de usuario determinada con el fin de lograr una mejor organización de estos.
- Nombre: Nombre del caso de prueba. Debe ser descriptivo, en la medida de las posibilidades, de lo que se comprobará y no muy extenso.
- No.Historia de Usuario: Número de la historia de usuario a la que corresponde. Índice de la historia de usuario a la que se le desea comprobar este aspecto.
- Descripción: Se describe qué es lo que se desea probar. La descripción debe ser corta y precisa.

- Condiciones de Ejecución: Condiciones especiales que deben tenerse en cuenta para ejecutar el caso de prueba.
- Entrada: Entrada al caso de prueba en caso de necesitarlas.
- Resultado Esperado: Resultado que se desea que tenga el caso de prueba. Descripción breve de lo que debe suceder.
- Evaluación de la Prueba: Se evalúa si el caso de prueba tuvo éxito o no. En caso de ser exitoso se asigna un resultado satisfactorio, en caso contrario no satisfactorio.

Prueba de Aceptación	
Código: 1	**No.Historia de Usuario:** 1
Nombre: Comprobar modelo de datos.	
Descripción: Prueba para verificar si se muestra correctamente el modelo de datos en la interfaz. Interfaz	
Condiciones de Ejecución: Comparar la figura 2.1 y el diseño del grafo correspondiente con el modelo de datos de la interfaz para ver si se muestran los puntos correctamente.	
Entrada: Compilar	
Resultado Esperado: Se muestra el modelo de datos correctamente	
Evaluación de la Prueba: Evaluación satisfactoria	

Tabla 30 Prueba de Aceptación: Comprobar modelo de datos.

Prueba de Aceptación	
Código: 2	**No.Historia de Usuario:** 2
Nombre: Comprobar si se pintan los caminos en el Modelo de Datos.	
Descripción: Prueba para verificar si se pintan los caminos	
Condiciones de Ejecución: Verificar en la matriz de las aristas si se adiciona el camino al dar clic en dos puntos del modelo de datos	
Entrada: Hacer clic en dos puntos del modelo de datos	
Resultado Esperado: Se pintan los caminos deseados por el usuario	
Evaluación de la Prueba: Evaluación satisfactoria	

Tabla 31 Prueba de Aceptación: Comprobar si se pintan los caminos en el Modelo de Datos

Prueba de Aceptación	
Código: 3	**No.Historia de Usuario:** 4
Nombre: Comprobar el formulario de opciones.	
Descripción: Prueba para verificar si se muestra el formulario de opciones	
Condiciones de Ejecución: Verificar si se muestra el formulario de opciones	
Entrada: Hacer clic en la interfaz opciones después de compilar.	
Resultado Esperado: Se muestra correctamente el formulario de opciones	
Evaluación de la Prueba: Evaluación satisfactoria	

Tabla 32 Prueba de Aceptación: Comprobar el formulario de opciones

Prueba de Aceptación	
Código: 4	**No.Historia de Usuario:** 3
Nombre: Comprobar si se ejecuta el algoritmo.	
Descripción: Prueba para verificar si se ejecuta correctamente el algoritmo.	
Condiciones de Ejecución: Verificar si se muestra un camino desde el nodo inicial hasta el nodo final.	
Entrada: Hacer clic en la interfaz ejecutar.	
Resultado Esperado: Se muestra el camino correctamente desde el primer nodo hasta el nodo final	
Evaluación de la Prueba: Evaluación satisfactoria	

Tabla 33 Prueba de Aceptación: Comprobar si se ejecuta el algoritmo

Prueba de Aceptación	
Código: 5	**No.Historia de Usuario:** 5
Nombre: Comprobar si se modifican los parámetros de las opciones.	
Descripción: Prueba para verificar si modifica los parámetros de las opciones correctamente.	
Condiciones de Ejecución: Verificar si se modifican los parámetros de las opciones al insertar otros valores como la cantidad de hormigas, alfa (importancia de las feromonas),	

beta (importancia de la heurística) y el coeficiente de evaporación de feromonas.
Entrada: Inserta valores en el formulario de opciones.
Resultado Esperado: Se modifican correctamente los valores
Evaluación de la Prueba: Evaluación satisfactoria

Tabla 34 Prueba de Aceptación: Comprobar si se modifican los parámetros de las opciones

Prueba de Aceptación	
Código: 6	**No.Historia de Usuario:** 6
Nombre: Comprobar si se reinicia la escena.	
Descripción: Prueba para verificar si reinicia la escena.	
Condiciones de Ejecución: Verificar si se reinicia la escena borrando el camino que se había mostrado del algoritmo	
Entrada: Hacer clic en la interfaz reiniciar.	
Resultado Esperado: Se muestra el modelo de datos que había inicialmente.	
Evaluación de la Prueba: Evaluación satisfactoria	

Tabla 35 Prueba de Aceptación: Comprobar si se reinicia la escena.

En esta fase se realizaron las tareas de ingeniería necesarias para la implementación de las historias de usuario que con la utilización del ACH permitirán encontrar el camino óptimo desde un nodo inicial hasta el nodo final en el modelo de datos correspondiente. Se confirma después de realizar pruebas de aceptación que las historias de usuario han sido implementadas correctamente al final de cada iteración.

CONCLUSIONES

En el trabajo se trataron aspectos y conceptos importantes relacionados con la Evacuación, Toma de decisiones, Inteligencia Artificial, Algoritmos Bioinspirados, Algoritmo Optimización con Colonia de Hormigas, el tipo de dato abstracto y el software a emplear. El estudio del estado del arte relacionado proporcionó los elementos teóricos necesarios para guiar el proceso de desarrollo.

Se confeccionó un modelo matemático donde se utilizó un modelo datos con valores reales según el diseño del tren en correspondencia con los parámetros exigidos que garantizaron la evacuación de pasajeros en un tren en condiciones normales.

La selección de la metodología XP utilizada para el desarrollo de la aplicación posibilitó la creación de los artefactos fundamentales, que permitió el desarrollo de la misma con calidad y cumpliendo las funcionalidades necesarias.

Se desarrolló una aplicación que permite el apoyo a la toma de decisiones, mostrando los pasos, estructura y procedimientos a seguir para desarrollar la misma y sus principales características, teniendo siempre como principal objetivo la utilización del Algoritmo Colonia de Hormigas.

RECOMENDACIONES

- Visualizar la simulación de este algoritmo en 3D para un mejor entendimiento visual de la solución.
- Aplicar el Algoritmo Colonia de Hormigas a problemas similares de evacuación con otros modelos de datos.
- Proponer un nuevo algoritmo derivado de la metaheurística colonia de hormigas que tome en cuenta la cantidad de personas que se encuentre en el vagón.

REFERENCIAS BIBLIOGRÁFICAS

[1] El agronegocio y la empresa agropecuaria frente al siglo XXI: Guillermo E Guerra,- San José. : IICA, 2002.

[1] Toma de decisiones basada en técnicas de la SoftComputing. Prof. Dr. Rafael Bello Pérez Centro de Estudios de Informática Universidad Central de Las Villas. Cuba. rbellop@uclv.edu.cu

[2] [28] EvacTrainViewer3D. Sistema para el Análisis 3D del Proceso de Evacuación en Trenes. MSc.Omar Correa Madrigal. Dr. Reinaldo Togores, Ing. Manuel Obregón, Lic. Enrique Gándara. Universidad de Cantabria, España 2010.

[3] [3] [15] Departamento Inteligencia Artificial UCI. Curso 2011-2012. eva.uci.cu

[4] Inteligencia artificial. Jordi Duch Gavaldá. Heliodoro Tejedor Navarro. www.uoc.edu

[5] Introducción a la Lógica Difusa. Tomás Arredondo Vidal. 26/6/09

[6] Investigación Operativa: Ing. Santiago Javez Valladares: Perú

[7] Simulación de comportamiento de una colonia de hormigas. Luis Enrique Corredera de Colsa. Doctorando en Ingeniería del software. Universidad Pontificia de Salamanca en Madrid.

[8]FACULTAD DE CIENCIAS MATEMÁTICAS E. A. P. DE INVESTIGACIÓN OPERATIVA.TESIS para optar el título profesional de Licenciado en Investigación Operativa. ASIGNACIÓN DE MÁQUINAS A ÓRDENES DEPRODUCCIÓN MEDIANTE PROGRAMACIÓNLINEAL ENTERA AUTOR: Raúl Eloy Araujo Cajamarca. Lima-Perú2009

[9] María Belén Melián Batista.2003.Optimización metaheurística para la planificación de redes WDM: Metaheuristic optimization for WDM network planning. http://dialnet.unirioja.es/servlet/tesis?codigo=1109

[10] ALGORITMOS BIOINSPIRADOS: LA EVOLUCIÓN BIOLÓGICA EN LA COMPUTACIÓN. César Luis Alonso y José Luis Montaña. Departamento de Matemáticas, Estadística y Computación.

[11] REVISTA INGENIERÍA E INVESTIGACIÓN VOL. 28 No. 2, AGOSTO DE 2008 (119-130). Inteligencia de enjambres: sociedades para la solución de problemas (una revisión). Mario A. Muñoz, Jesús A. López y Eduardo F. Caicedo.

[12] Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos. José Manuel García Nieto. Septiembre de 2006.

[13] Modelos Matemáticos de Optimización: Alberto Aguilera: Universidad Pontificia Comillas: 1998.

[14] SISTEMAS COMPLEJOS, ALGORITMOS EVOLUTIVOS Y BIOINSPIRADOS. OTRAS METAHEURÍSTICAS. Oscar Cordón.

[16] La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. Sergio Alonso, Oscar Cordón, Iñaki Fernández de Viana, Francisco Herrera. Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada (España).

[17][20] UN ALGORITMO HÍBRIDO BASADO EN COLONIAS DE HORMIGAS PARA LA RESOLUCIÓN DE PROBLEMAS DE DISTRIBUCIÓN EN PLANTA ORIENTADOS A PROCESOS. Ángel Cobo Ortega, Ana María Serrano Bedia. Universidad de Cantabria

[18] ACHPM: ALGORITMO DE OPTIMIZACIÓN CON COLONIA DE HORMIGAS PARA EL PROBLEMA DE LA MOCHILA. Carlos A. Ochoa Facultad de Ingeniería Eléctrica, Universidad Autónoma de Zacatecas

[19] Algoritmo de Colonia de Hormigas para el Problema del Clique Máximo con un Optimizador Local K-opt Julio C. Ponce, Eunice E. Ponce de León, Alejandro Padilla, Felipe Padilla, Alberto Ochoa O. Zezzatti. Departamento de Sistemas Electrónicos – Universidad Autónoma de Aguascalientes. México

[21] Costo de producción del proyecto SCADA mediante la aplicación del Algoritmo basado en Colonia de Hormigas. Maikelis Ananka Rosales Almaguer. Mirna Indiana Beyris Bringuez. Ciudad de La Habana .Junio-2010

[22] Algoritmo basado en la optimización mediante colonias de hormigas para la resolución del problema del transporte de carga desde varios orígenes a varios destinos. Lucía Barcos, Victoria M. Rodríguez y Mª Jesús Álvarez. Departamento de Organización Industrial, Tecnun, Universidad de Navarra, España. 2002.

[23] Colonia de Hormigas en un Ambiente Paralelo Asíncrono. Benjamín Barán. Universidad Nacional de Asunción. Centro Nacional de Computación. San Lorenzo, Paraguay

[24] Introducción a la Teoría de Grafos. Matemáticas Discretas. Prof. José Luis Chacón. Semestre A2005

[24] Dpto. Técnicas de Programación. Universidad de las Ciencias Informáticas. El Tipo de Dato Abstracto Grafo.

[25] Lenguajes de Programación. Segunda Edición. Principios y práctica. Kenneth C.Louden.

[26] Qt-Creator, desarrollando aplicaciones rápidamente. Por PaBLoX Viernes Mayo 15, 2009

[27] SISTEMA DE EVACUACIÓN SECTORIAL. Coordinadora Nacional para la Reducción de Desastres Secretaría Ejecutiva. Guatemala, C.A. www.conred.gob.gt

[29] Universidad Autónoma del Estado de Morelos. La Visualización y la Modelación en la Adquisición del Concepto Función. Orlando Planchart Márquez. Cuernavaca, Morelos.

[30] Stasko, Domingue, Brown, Price. (1998). Software Visualization: Programming as a Multimedia Experience. MIT Press.

[31] Artículo Científico desarrollado en base a la memoria con el mismo nombre, para obtener la Titulación de Máster en Sistemas Inteligentes por la Universidad de Salamanca, Federico Medrano, España 2010-2011

[32] Geosoluciones. Cl-Soluciones Integrales en Geometría. Geoinformática, Modelación Ambiental, Asesoría, Capacitación. 11 de junio de 2009.

[33] Dorigo, M. y Stützle, T., Ant Colony Optimization: MIT Press, 2004. ISBN: 0-262-04219-3.

[34] Análisis, Diseño, e Implementación de un software, para la administración de los proyectos de grado en el programa de ingeniería de sistemas, aplicando una metodología ágil. Juan Pablo Roche y Julián Mauricio Suarez. Universidad Tecnológica de Pereira. Colombia. 2009.

ANEXOS

Anexo 1

→ Entre la celda 10 y 35

Vector 10 (5,662; 0,484; 0,3) Vector 35 (5,662; 0,984; 0,3)

$Vector10Vector35 = \sqrt{(5{,}662 - 5{,}662)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector10Vector35 = 0,5.

→ Entre la celda 12 y 37

Vector 12 (6,662; 0,484; 0,3) Vector 37 (6,662; 0,984; 0,3)

$Vector12Vector37 = \sqrt{(6{,}662 - 6{,}662)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector12Vector37 = 0,5.

→Entre la celda 14 y 39

Vector 14 (7,661; 0,484; 0,3) Vector 39 (7,661; 0,984; 0,3)

$Vector14Vector39 = \sqrt{(7{,}661 - 7{,}661)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector14Vector39 = 0,5.

→Entre la celda 16 y 41

Vector 16 (8,661; 0,484; 0,3) Vector 41 (8,661; 0,984; 0,3)

$Vector16Vector41 = \sqrt{(8{,}661 - 8{,}661)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector16Vector41 = 0,5.

→Entre la celda 18 y 43

Vector 18 (9,661; 0,484; 0,3) Vector 43 (9,661; 0,984; 0,3)

$Vector18Vector43 = \sqrt{(9{,}661 - 9{,}661)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector18Vector43 = 0,5.

→Entre la celda 20 y 45

Vector 20 (10,661; 0,484; 0,3) Vector 45 (10,661; 0,984; 0,3)

$Vector20Vector45 = \sqrt{(10{,}661 - 10{,}661)^2 + (0{,}484 - 0{,}984)^2 + (0{,}3 - 0{,}3)^2}$

Vector20Vector45 = 0,5.

1 →Entre la celda 22 y 47

2 Vector 22 (11,662; 0,484; 0,3) Vector 47 (11,662; 0,984; 0,3)

3 Vector22Vector47 = √ $(11,662 – 11,662)^2 + (0,484 – 0,984)^2 + (0,3 – 0,3)^2$

4 Vector20Vector45 = 0,5.

5 →Entre la celda 80 y 105

6 Vector 80 (3,162; 1,984; 0,3) Vector 105 (3,162; 2,482; 0,3)

7 Vector80Vector105 = √ $(3,162 – 3,162)^2 + (1,984 – 2,482)^2 + (0,3 – 0,3)^2$

8 Vector80Vector105 = √$0 + (-0,498)^2 + 0$

9 Vector80Vector105 = 0,498 🏁 0,5.

10 →Entre la celda 83 y 108

11 Vector 83 (4,662; 1,984; 0,3) Vector 108 (4,662; 2,482; 0,3)

12 Vector83Vector108 = √ $(4,662 – 4,662)^2 + (1,984 – 2,482)^2 + (0,3 – 0,3)^2$

13 Vector83Vector108 = 0,498 🏁 0,5.

14 →Entre la celda 85 y 110

15 Vector 85 (5,662; 1,984; 0,3) Vector 110 (5,662; 2,482; 0,3)

16 Vector85Vector110 = √ $(5,662 – 5,662)^2 + (1,984 – 2,482)^2 + (0,3 – 0,3)^2$

17 Vector85Vector110 = 0,498 🏁 0,5.

18 →Entre la celda 87 y 112

19 Vector 87 (6,662; 1,984; 0,3) Vector 112 (6,662; 2,482; 0,3)

20 Vector87Vector112 = √ $(6,662 – 6,662)^2 + (1,984 – 2,482)^2 + (0,3 – 0,3)^2$

21 Vector87Vector112 = 0,498 🏁 0,5.

22 → Entre la celda 89 y 114

23 Vector 89 (7,662; 1,984; 0,3) Vector 114 (7,662; 2,482; 0,3)

24 Vector89Vector114 = √ $(7,662 – 7,662)^2 + (1,984 – 2,482)^2 + (0,3 – 0,3)^2$

25 Vector89Vector114 = 0,498 🏁 0,5.

→ Entre la celda 91 y 116

Vector 91 (8,661; 1,984; 0,3) Vector 116 (8,661; 2,482; 0,3)

Vector91Vector116 = $\sqrt{}$ (8,661 – 8,661)² + (1,984 – 2,482)² + (0,3 – 0,3)²

Vector91Vector116 = 0,498 🏁 0,5.

→ Entre la celda 93 y 118

Vector 93 (9,661; 1,984; 0,3) Vector 118 (9,661; 2,482; 0,3)

Vector93Vector118 = $\sqrt{}$ (9,661 – 9,661)² + (1,984 – 2,482)² + (0,3 – 0,3)²

Vector93Vector118 = 0,498 🏁 0,5.

→ Entre la celda 95 y 120

Vector 95 (10,661; 1,984; 0,3) Vector 120 (10,661; 2,482; 0,3)

Vector95Vector120 = $\sqrt{}$ (10,661 – 10,661)² + (1,984 – 2,482)² + (0,3 – 0,3)²

Vector95Vector120 = 0,498 🏁 0,5.

→ Entre la celda 97 y 122

Vector 97 (11,662; 1,984; 0,3) Vector 122 (11,662; 2,482; 0,3)

Vector97Vector122 = $\sqrt{}$ (11,662 – 11,662)² + (1,984 – 2,482)² + (0,3 – 0,3)²

Vector97Vector122 = 0,498 🏁 0,5.

Anexo 2

→Entre la celda 58 y 60

Vector 58 (4,662; 1,484; 0,3) Vector 60 (5,662; 1,484; 0,3)

Vector58Vector60 = $\sqrt{}$ (4,662 – 5,662)² + (1,484 – 1,484)² + (0,3 – 0,3)²

Vector58Vector60 = $\sqrt{}$ (-1)² + 0 + 0

Vector58Vector60 = 1.

→Entre la celda 60 y 62

Vector 60 (5,662; 1,484; 0,3) Vector 62 (6,662; 1,484; 0,3)

Vector60Vector62 = $\sqrt{}$ (5,662 – 6,662)² + (1,484 – 1,484)² + (0,3 – 0,3)²

1 Vector60Vector62 = $\sqrt{(-1)^2 + 0 + 0}$

2 Vector60Vector62 = 1.

3 →Entre la celda 62 y 64

4 Vector 62 (6,662; 1,484; 0,3) Vector 64 (7,662; 1,484; 0,3)

5 Vector62Vector64 = $\sqrt{(6,662 - 7,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

6 Vector62Vector64 = $\sqrt{(-1)^2 + 0 + 0}$

7 Vector62Vector64 = 1.

8 →Entre la celda 64 y 66

9 Vector 64 (7,662; 1,484; 0,3) Vector 66 (8,661; 1,484; 0,3)

10 Vector64Vector66 = $\sqrt{(7,662 - 8,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

11 Vector64Vector66 = $\sqrt{(-1)^2 + 0 + 0}$

12 Vector64Vector66 = 1.

13 →Entre la celda 66 y 68

14 Vector 66 (8,661; 1,484; 0,3) Vector 68 (9,661; 1,484; 0,3)

15 Vector66Vector68 = $\sqrt{(8,662 - 9,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

16 Vector66Vector68 = $\sqrt{(-1)^2 + 0 + 0}$

17 Vector66Vector68 = 1.

18 →Entre la celda 68 y 70

19 Vector 68 (9,661; 1,484; 0,3) Vector 70 (10,661; 1,484; 0,3)

20 Vector68Vector70 = $\sqrt{(9,662 - 10,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

21 Vector68Vector70 = $\sqrt{(-1)^2 + 0 + 0}$

22 Vector68Vector70 = 1.

23 →Entre la celda 70 y 72

24 Vector 70 (10,661; 1,484; 0,3) Vector 72 (11,662; 1,484; 0,3)

25 Vector70Vector72 = $\sqrt{(10,661 - 11,662)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

26 Vector70Vector72 = $\sqrt{(-1)^2 + 0 + 0}$

1 Vector70Vector72 = 1.

2 →Cálculo entre los nodos de salida

3 →Entre la celda 51 y 55

4 Vector 51(1,136; 1,484; 0,3) Vector 55 (3,162; 1,484; 0,3)

5 $Vector51Vector55 = \sqrt{(1,136 - 3,162)^2 + (1,484 - 1,484)^2 + (0,3 - 0,3)^2}$

6 $Vector51Vector55 = \sqrt{(-2,026)^2 + 0 + 0}$

7 Vector51Vector55 = 2,026 🏁 2.

8 →Entre la celda 1 y 51

9 Vector 1 (1,397; 0,477; 0,3) Vector 51 (1,136; 1,484; 0,3)

10 $Vector1Vector51 = \sqrt{(1,397 - 1,136)^2 + (0,477 - 1,484)^2 + (0,3 - 0,3)^2}$

11 $Vector1Vector51 = \sqrt{(0,261)^2 + (-1,007)^2 + 0}$

12 Vector1Vector51 = 0,261 + 1,014049 + 0

13 Vector1Vector51 = 1,275049 🏁 1,3.

14 →Entre la celda 51 y 100

15 Vector 51(1,136; 1,484; 0,3) Vector 100 (1000; 1000; 1000)

16 Como el vector 100 es un espacio ocupado utilizaremos los vectores 55 y 105.

17 Vector 55 (3,162; 1,484; 0,3) Vector 105 (3,162; 2,482; 0,3)

18 $Vector55Vector105 = \sqrt{(3,162 - 3,162)^2 + (1,484 - 2,482)^2 + (0,3 - 0,3)^2}$

19 $Vector55Vector105 = \sqrt{0 + (-0,998)^2 + 0}$

20 $Vector55Vector105 = \sqrt{0,996004} = 0,998$ 🏁 1