UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

(Universidad del Perú, DECANA DE AMÉRICA)

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

Escuela Profesional De Ingeniería De Sistemas

ENTREGABLE FINAL

Asignatura

Inteligencia Artificial

Grupo 8

Ccolcca Avalos, Mariluz – 19200177
Estrada Salazar, Jhony Kevin - 15200119
Noriega Vela, Diego- 19200190
Ramirez Teran, Brithany Antonella – 19200168
Sandoval Juarez,Ariana Ayelen – 19200153
Solis Rivera, Angel Jeanpierre – 19200194

Docente

Ing. Gisella Maquen Niño

LIMA – PERÚ

2022

Resumen

Dentro del tema de optimización, se busca conseguir la mejor solución posible a cualquier tipo de problema utilizando los recursos disponibles, ya sea maximizar o minimizar. Respecto a esto, para lograr una solución existen muchos algoritmos como el de Ascenso de Colina que es considerado una técnica de búsqueda local entre dos puntos o locales si nos referimos a este proyecto. En ese sentido, el presente proyecto tuvo como objetivo principal el estudio del algoritmo de optimización "Ascenso de Colina" o también llamado Hill-Climbing para la solución a la problemática dentro de la empresa de transporte CIVA respecto a la búsqueda del camino más corto entre dos locales de envíos de cargos y encomiendas a nivel nacional.
A partir de lo mencionado, en este informe se presenta el capítulo uno con la situación problemática, objetivos y justificación apropiada, el capítulo dos incluye un marco teórico con antecedentes locales, nacionales e internacionales, dentro de las teorías relacionadas tenemos la descripción del algoritmo y herramientas tecnológicas usadas, en el capítulo tres tenemos al desarrollo de la propuesta que contiene la codificación del software y pruebas. Finalmente, como resultado, se obtuvo la implementación lógica y gráfica del algoritmo a detalle, se determinó que la complejidad del algoritmo es de orden $O(N^5)$ y se detalló los aciertos y tiempos de ejecución mediante tablas y gráfico. Finalmente, se concluyó que correcto estudio del algoritmo de optimización "Ascenso de Colina" útil para la resolución del problema de búsqueda del camino más corto entre dos puntos de la red de envío de cargos y encomiendas de la empresa de transportes CIVA demostrado en la aplicación web implementada.

Palabras claves: optimización, búsqueda local, algoritmo de ascenso de colina, empresa de transporte CIVA, camino más corto, aplicación web.

Abstract

Within the theme of optimization, the aim is to achieve the best possible solution to any type of problem using the available resources, either maximizing or minimizing. Regarding this, to achieve a solution there are many algorithms such as Hill Ascent, which is considered a local search technique between two points or local if we refer to this project. In this sense, the present project had as its main objective the study of the optimization algorithm "Ascenso de Colina" or also called Hill-Climbing for the solution to the problem within the transport company CIVA regarding the search for the shortest path between two local cargo and parcel shipments nationwide.

Based on the above, this report presents chapter one with the problematic situation, objectives and appropriate justification, chapter two includes a theoretical framework with local, national and international background, within the related theories we have the description of the algorithm and technological tools used, in chapter three we have the development of the proposal that contains the software coding and tests. Finally, as a result, the logical and graphical implementation of the algorithm was obtained in detail, it was reduced that the complexity of the algorithm is of order O(N5) and the successes and execution times were detailed through tables and graphs. Finally, it was concluded that a correct study of the optimization algorithm "Ascenso de Colina" useful for solving the search problem of the shortest path between two points of the cargo and parcel delivery network of the CIVA transport company demonstrated in the application. website implemented.

Keywords: optimization, local search, hill climbing algorithm, CIVA transport company, shortest path, web application.

Índice

Introducción

Muchos autores sostienen una definición distinta respecto a la Inteligencia Artificial. Sin embargo, se coincide con el enfoque propuesto por Takeyas (2007), quien ha definido a la Inteligencia Artificial como una rama de las ciencias de la computación encargada del estudio de modelos informáticos capaces de realizar actividades típicas humanas en base a sus dos características principales: el razonamiento y el comportamiento. Entonces, se puede aseverar que la labor de la Inteligencia Artificial parte de la idea de crear agentes inteligentes diseñados para realizar acciones racionales, una de ellas, la de encontrar soluciones ante una problemática. Por ejemplo, existen diversas áreas de aplicación donde la Inteligencia Artificial se ha desarrollado en menor y mayor medida, algunas de ellas corresponden con el tratamiento de lenguajes naturales, creación de sistemas expertos, robótica, percepción y aprendizaje automático (Galipienso et al.,2003). Estas y otras aplicaciones se centran en el diseño de un agente inteligente que se encarga de percibir y actuar en el medio donde se encuentra. Este comportamiento es evaluado por la medida de rendimiento y pueden responder directamente a las percepciones (agentes reactivos simples), seguir la evidencia de esas percepciones (agentes reactivos basados en modelos), intentar alcanzar sus metas (agentes basados en objetivos) o intentar maximizar la optimización deseada (agentes basados en utilidad).

Siguiendo la línea de los agentes basados en metas, uno de ellos corresponde a los agentes encargados de la resolución de problemas, estas la realizan mediante herramientas de búsqueda, dentro de las cuales se encuentran las no informadas (búsqueda a ciegas) o las informadas (heurística).

Respecto a las búsquedas no informadas o a ciegas, cabe resaltar que estas generan sucesores y distinciones sobre si han llegado o no a la meta. El único parámetro que contienen es la formulación del problema en sí, mas no información adicional acerca de los estados. Entre

ellas se encuentran la búsqueda Primero en Anchura, Costo Uniforme, Primero en Profundidad, Primero en Profundidad con Profundidad Iterativa y la búsqueda Bidireccional.

Por otro lado, las estrategias basadas en búsqueda informadas o heurísticas tratan sobre si un estado se acerca más a la meta a comparación de otro cuando este no es objetivo. Entre ellas destaca la búsqueda Primero el Mejor, la Búsqueda A* , los métodos de búsqueda local como la Ascensión de Colinas, entre otros.

La evaluación de la eficiencia de cada una de estas estrategias de búsqueda recae sobre si estas garantizan encontrar una solución (completitud), además del tiempo que necesitan (complejidad en tiempo), la memoria que necesita (complejidad en espacio) y encontrar la solución óptima (optimización). Sin embargo, más allá de lo teórico, también se debe tomar en cuenta la eficiencia de las estrategias para resolver problemas de la vida real. En ese contexto, surge la idea del presente proyecto, el cual trata específicamente sobre estudiar la aplicación del algoritmo de búsqueda Ascensión de Colina para la solución de la problemática de una empresa de transportes.

Capítulo I: Situación Problemática

1.1. Planteamiento Del Problema

1.1.1. Descripción Del Problema

La empresa de transportes Civa S.A. se ha esforzado por cumplir con la necesidad de transportar bienes o mercancías a cualquier región del Perú; ya sea por razones de negocio o simplemente enviándoles a otras personas.

Figura 1

Logo de empresa Civa S.A.

Fuente: Civa.com

A partir de ello, surge su necesidad de contar con un sistema diseñado para brindar un servicio de transporte seguro y eficiente en cada una de sus agencias mediante la búsqueda de optimización de rutas nacionales para entregar los envíos de manera puntual al destino indicado. Esta necesidad está orientada a la mejora del servicio de la empresa que conlleve a una mejor eficiencia y, por tanto, a una mayor competitividad de interés económico y social.

1.1.2. Formulación Del Problema

¿De qué manera se puede contribuir al sistema de envío de cargos o encomiendas de la empresa Civa en su búsqueda de hallar el camino más corto entre dos agencias (optimización de rutas)?

1.1.3. Formulación De La Hipótesis

Se puede encontrar soluciones óptimas mediante la implementación del algoritmo de búsqueda Ascenso de Colina para una correcta representación del camino más corto entre dos agencias.

1.2. Objetivo General

Estudiar el algoritmo de optimización Ascenso de Colina para la resolución del problema de búsqueda del camino más corto entre dos puntos cualesquiera de la red de envío de cargos y encomiendas de la empresa de transportes Civa.

1.3. Objetivos Específicos

- Desarrollar un código de aplicación web que muestre la implementación del algoritmo Ascenso de colina de manera gráfica.
- Adaptar la lógica del algoritmo de búsqueda Ascenso de Colina dentro del problema objeto de estudio de la empresa.
- Establecer los nodos que serán tomados en cuenta dentro de la codificación del algoritmo tanto en la lógica como en la muestra gráfica.
- Identificar las dificultades y/o errores de la implementación del algoritmo de búsqueda Ascenso de Colina siguiendo la filosofía de esta.
- Implementar y comprobar la eficacia del algoritmo de búsqueda Ascenso de Colina en un determinado sistema.

1.4. Justificación e Importancia

1.4.1. Justificación Teórica

Para un problema dado, cuando el espacio de soluciones incluye cientos o miles de posibilidades, es poco práctico utilizar métodos de búsqueda exhaustivos (Soto et al., 2008). En ese sentido, Gosavi (2003) asevera que expertos han desarrollado arduamente estrategias que generan buenas soluciones de manera óptima, discreta y a gran escala denominadas metaheurísticas. Estas incluyen entre sus técnicas, la aplicación de diferentes algoritmos de búsqueda local. Mencionado lo anterior, el proyecto busca, mediante la aplicación de la teoría y los conceptos básicos de las estrategias de algoritmos de búsqueda más sofisticadas (como lo es el algoritmo de búsqueda local Ascenso de Colina), encontrar soluciones cuyas características garanticen o se aproximen a la solución óptima global de problemas.

1.4.2. Justificación Práctica

Un algoritmo de búsqueda local como el algoritmo Ascenso de Colina es considerado un algoritmo codicioso, ya que siempre toma la mejor decisión disponible en cada paso, con la esperanza de obtener el mejor resultado general (Soto et al., 2008) que se puede aplicar a distintas problemáticas relacionadas con la actividad económica y/o comercial. En ese sentido, de acuerdo con los objetivos del proyecto, el resultado de este permite encontrar soluciones concretas a los problemas mencionados anteriormente, lo cual mejorará sustancialmente el servicio de envío que ofrece la empresa de transportes Civa S.A. reduciendo sus costes al mínimo.

1.4.3. Justificación Económica

La implementación de una aplicación de búsqueda de Ascenso de Colina como método de optimización de rutas en la empresa Civa, podría significar una disminución de costos de distribución, ya que estas pueden proveer a la empresa una ruta corta para llegar a un destino, minimizando la cantidad de recursos que se necesitan para el transporte

(combustible, recursos humanos, etc). Además del aprovechamiento del ahorro de tiempo para realizar otras actividades relacionadas al servicio.

1.4.4. Justificación Social

Toda investigación debe involucrar cierta importancia social, obteniendo ser trascendente para la sociedad o evidenciar cierta proyección social (Arias,2012). En ese sentido, tenemos que en la actualidad uno de los problemas más sonados en la ciudad de Lima es el congestionamiento vehicular, esto se evidencia en las vías limeñas comprometiendo el envío de encomiendas y/o cargos de la empresa Civa, que también necesita seguridad en ese tipo de transportes. Al enfoque social comentado se suma el algoritmo de ascensión de colina que busca la mejor ruta entre dos agencias o destinos, de esa manera se logrará ser más eficiente respecto a otras empresas y el cliente estará satisfecho con la atención brindada que posiblemente nos recomiende a otros familiares y/o amigos.

1.4.5. Justificación Tecnológica

El actual estudio busca la implementación del algoritmo Hill-Climbing en la solución para la búsqueda de la ruta más corta entre dos puntos, Rjoub (2020) menciona sobre el algoritmo que este tiene ventajas respecto a otros como lo es sus bajos requisitos de energía, su convergencia bastante rápida y sus requisitos razonables de poca memoria, por lo que la empresa Civa puede tener un eficiente aprovechamiento de recursos computacionales. Por otro lado, la implementación de la aplicación dentro de la empresa mencionada contribuye a la automatización y digitalización de procesos de distribución de carga.

Capítulo II: Marco Teórico

2.1. Antecedentes

2.1.1. Antecedente Internacionales

2.1.1.1. Algoritmo ß-Hill Climbing para juego de Sudoku

AAzmi Al-Betar, Awadallah ,Bolaji y Alijla (2017) en el artículo científico dentro de Palestinian International Conference on Information and Communication Technology con el título "Algoritmo ß-Hill Climbing para juego de Sudoku" tuvo como objetivo central la adaptación del algoritmo en mención para el juego de rompecabezas sudoku referente a la optimización, también se presenta una breve descripcion y formulacion del juego, los cambios del algoritmo de ascensión de colina o Hill Climbing para finalmente evaluar lo propuesto. Utilizó un diseño cuasiexperimental que consta de un plan de trabajo para estudiar el impacto de los procesos de cambio en diversas situaciones. Su principal resultado es la verificación del algoritmo implementado que luego del análisis mediante parámetros (ß y N) se logra resolver en 19 iteraciones con tiempo de 2 segundos considerando el valor de 0,01 para N y 0.5 para ß. Finalmente, se concluye que el algoritmo funciona de manera eficiente cuando los valores de N y ß son bajos. Además, la capacidad de convergencia del algoritmo ß-Hill Climbing se prueba usando un ejemplo práctico de Sudoku que en un futuro se puede medir con diferentes dificultades.

2.1.1.2. Una solución metaheurística al problema de planificación de rutas de autobuses escolares con flota homogénea y selección de paradas

Pérez, A. C. P., Ansola, E. S., & Rosete, A. (2021), en el artículo de la revista Ingeniería titulado "Una solución metaheurística al problema de planificación de rutas de autobuses escolares con flota homogénea y selección de paradas" tuvo como objetivo la planificación de rutas de autobuses escolares para transportar estudiantes desde una parada

hasta la escuela, considerando que cada estudiante no debe caminar más de una distancia máxima definida. Se diseñó e implementó un modelo matemático del problema y se adaptó para su solución usando algoritmos metaheurísticos. Para evaluar los resultados, se realizó una comparación estadística con otra solución de la literatura a partir de la evaluación de 112 instancias del problema. En la solución de las 112 instancias del problema se utilizaron los siguientes algoritmos:búsqueda tabú, variantes del escalador de colinas, recocido simulado y un portafolio de algoritmos que incluye a los anteriores. Los resultados reflejaron que el mejor comportamiento lo obtuvo el portafolio con resultados comparables a los de la literatura. Sin embargo, a medida que crece el número de instancias, los resultados tienden a empeorar. Con esta investigación se obtuvo un modelo que permite representar el problema inicial, así como dos algoritmos para la búsqueda de soluciones utilizando metaheurísticas. Según los resultados obtenidos, los trabajos futuros deben encaminarse hacia nuevas formas de construcción de la solución inicial y la implementación de nuevos operadores de asignación y mutación.

2.1.1.3. Diseño e implementación de una meta-heurística multi-poblacional de optimización combinatoria enfocada a la resolución de problemas de asignación de rutas a vehículos

Osaba (2015) en su tesis doctoral dentro del programa de doctorado en Ing. Informática y Telecomunicación titulada “Diseño e implementación de una meta-heurística multi-poblacional de optimización combinatoria enfocada a la resolución de problemas de asignación de rutas a vehículos”. En la presente investigación se mencionan que existen problemas de optimización en la asignación de rutas o vehículos, es por ello que se plantea como objetivo general plantear una técnica metaheurística, la cual ayude a resolver los problemas encontrados y optimizarlo a través de la optimización por combinatoria y de esta

manera aportar un valor añadido comparado con los demás métodos que ya existen. Se hicieron uso de 6 problemas distintos de optimización combinatoria, donde 4 de ellas son pertenecientes al problema de asignación de rutas y las otras restantes al de combinatoria. Como resultado se obtuvo que la técnica metaheurística planteada fue satisfactoria, la cual también puede ser usada para distintos problemas más complejos de optimización donde se presentan tiempos de ejecución admisibles y un comportamiento de convergencia destacables.

2.1.1.4. Application of a Hill-Climbing Algorithm to Public Transportation Routes Design in Grid Networks

Zarrinmehr, A. y Moloukzade, H. (2021) en su trabajo de investigación en la Revista Internacional de Ingeniería de Transporte que tiene como título "Application of a Hill-Climbing Algorithm to Public Transportation Routes Design in Grid Networks" se enfoca en un problema de diseño de rutas de tránsito (TRD) sobre una red de transporte grid urbana. Para este problema se tiene como objetivo maximizar la cobertura del servicio a través de la red, manteniendo un límite presupuestario. Para ello, se propone y evalúa un algoritmo heurístico de búsqueda local Hill-Climbing (HC). El algoritmo HC propuesto realiza varias replicaciones en las que comienza con una combinación de rutas seleccionadas aleatoriamente y las mejora iterativamente moviéndose al "mejor vecino" hasta que alcanza una solución óptima local. Los resultados para una red de cuadrícula de 6 × 10 para tres niveles de presupuesto, es decir, niveles de presupuesto bajo, medio y alto, indican que, el algoritmo HC propuesto puede producir soluciones de alta calidad con 0,12 %, 4.16% y 2.22% de diferencia de los algoritmos óptimos globales. Se concluye que el algoritmo propuesto puede lograr resultados prometedores en términos de rendimiento y calidad de sus soluciones.

2.1.1.5. Courses timetabling based on hill climbing algorithm.

Rjoub (2020) en el artículo titulado “Courses timetabling based on hill climbing algorithm” publicado en la revista “International Journal of Electrical and Computer Engineering” tuvo como objetivo superar la programación manual de horarios y reemplazarla por una automatizada. Utilizó una técnica consta de dos etapas: en la primera, se crea una solución inicial de una manera que evitaría cualquier conflicto aprovechando las representaciones orientadas a la clase y orientadas al instructor con varias otras funciones para garantizar que no se seleccionen intervalos de tiempo de manera que terminen en conflictos. En la segunda etapa, se utiliza el algoritmo Hill Climbing, con algunas modificaciones, aprovechando la cantidad de ventajas que conlleva, incluidos sus bajos requisitos de energía, su convergencia bastante rápida y sus requisitos razonables de poca memoria. El sistema propuesto fue desarrollado usando varias tecnologías de soporte, incluyendo SQL Server 2008, Visual Studio .NET y ASP .NET. Como resultado este sistema aprovechó el algoritmo Hill Climbing que proporcionó buenos resultados para diferentes tamaños de escuelas, además se descubrió que la técnica propuesta ahorra más tiempo y esfuerzo al idear múltiples versiones alternativas del horario escolar en minutos, con el resultado final de crear horarios más prácticos a costos de tiempo reducidos.

2.1.2. Antecedentes Nacionales

2.1.2.1. Aplicación móvil de algoritmos de rutas óptimas y su efecto en el desplazamiento de los conductores de vehículos en la ciudad de Trujillo.

Mera y Salinas (2018) en su tesis para optar el título profesional de Ingeniero de Sistema Computacionales en la Universidad Privada del Norte titulada "Aplicación móvil de algoritmos de rutas óptimas y su efecto en el desplazamiento de los conductores de vehículos en la ciudad de Trujillo", tuvo como objetivo general proponer rutas viables para los conductores de vehículos, con la finalidad de evitar obstrucciones por accidente, como puede ser encontrarse con obras de construcción, tráfico de vehículos o accidentes automovilísticos en la ciudad de Trujillo. La presente investigación fue de tipo experimental, donde se tomaron como muestra a 33 conductores de esta ciudad, los cuales pasaron una serie de evaluación. Se obtuvieron como resultados que las evaluaciones fueron óptimas, hubo una mejora en el tiempo de recorrido en un 34%, la distancia se redujo al 8.6% y el número de incidentes evitados aumentó en un 60.32%, lo cual significa que la aplicación móvil que usó distintos algoritmos como el algoritmo de rutas cortas, búsqueda por profundidad, anchura, Dijkstra u otros algoritmos tuvo efectos positivos en cuanto al desplazamiento de los conductores.

2.1.2.2. Sistema web basado en algoritmo de ruta más corta para optimización de rutas en la empresa de servicios logísticos de Courier Seminario Martínez Servicios Generales S.A.C."

Milian (2019) en su tesis de pregrado para obtener el grado académico de Ingeniero de Sistemas y Computación en la Universidad Católica Santo Toribio de Mogrovejo titulada "Sistema web basado en algoritmo de ruta más corta para optimización de rutas en la empresa de servicios logísticos de courier Seminario Martínez Servicios Generales SAC.", en el presente trabajo, se detalla que en la empresa mencionada anteriormente presenta un

problema de pérdida de dinero y tiempo dado que las rutas de distribución eran designadas de manera aleatoria. Para lo cual se planteó como objetivo implementar un Sistema Web utilizando el algoritmo de ruta más corta lo cual optimizaría las rutas, tiempos, coste de gestión de gestión y distribución de paquetes de la empresa. Para ello, se utilizó la base de datos MySQL, PHP como lenguaje de programación y para la determinación del camino más corto se utilizó el algoritmo Dijkstra. Como resultado, se logró reducir el porcentaje de entregas impuntuales en más de un 50% lo cual ha impactado de manera positiva en la satisfacción del cliente y por ende han mejorado las ganancias netas de la empresa.

2.1.2.3. Implementación de un modelo matemático para la planificación de un viaje personalizado basado en optimización multiobjetivo para los turistas en la región Puno.

Anquise Jihuaña, Y. (2019) en la tesis de la Universidad Peruana Unión, titulada "Implementación de un modelo matemático para la planificación de un viaje personalizado basado en optimización multiobjetivo para los turistas en la región Puno", se tiene como objetivo general la implementación de un modelo matemático para la planificación de un viaje personalizado basado en optimización multiobjetivo para los turistas en la región Puno. Para obtener la solución al modelo se eligió un método aproximado para obtener la solución en un tiempo razonable. Para problema abordado en este proyecto se eligió la implementación desarrollada en jMetal del algoritmo NSGA-II. Para alcanzar el primer objetivo se logró recolectar datos de ubicaciones de 200 atractivos turísticos ubicados en la región de Puno a través de un proceso manual del inventario realizado por el Ministerio de Comercio Exterior y Turismo del Perú. Para alcanzar el segundo objetivo se logró aplicar el algoritmo Non-dominated Sorting Genetic Algorithm II en el lenguaje de programación de Java, usando una herramienta denominada jMetal que contiene algoritmos metaheúristicos

para resolver problemas de optimización multiobjetivo. Finalmente, para el último objetivo se logró implementar una interfaz en un entorno de trabajo interactivo que permite desarrollar código en Python denominado Jupyter Notebook para simular la planificación de ruta de un viaje turístico. Se observó que para un turista que desea planificar su propio viaje sin depender de opciones prediseñadas es complicada la tarea de evaluar todas las posibilidades para tener la ruta óptima de los atractivos que desea visitar, al recurrir por ayuda a las computadoras tampoco es posible tener soluciones exactas para una cantidad de atractivos considerable ya que el tiempo computacional crece exponencialmente al aumentar el tamaño del problema.

2.1.2.4. Sistema de búsqueda heurística para la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo utilizando el algoritmo A Star.

Quiroz & Ramírez (2019), en la tesis sustentada en la Universidad Nacional Pedro Ruíz Gallo con el título “Sistema de búsqueda heurística para la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo utilizando el algoritmo A Star” tuvo como objetivo central desarrollar un sistema de búsqueda heurística para la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo utilizando el algoritmo A Star. Utilizó un diseño no experimental transeccional de tipo explicativo, ya que se recolectan datos sobre las incidencias de los alumnos, en la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo mediante encuestas y observación; el sistema de búsqueda heurística es un software compuesto principalmente por una aplicación móvil que interactuara con una interfaz de programación de aplicaciones (API), una base de datos y aplicación Web que implementa el algoritmo A star el cuál genera las rutas. Se llegó a concluir que el desarrollo del sistema de búsqueda heurística permitió localizar las principales oficinas de la Universidad Nacional

Pedro Ruiz Gallo con mayor facilidad gracias a las rutas generadas por el algoritmo A star gracias a que se desenvuelve mejor en un grafo bidireccional sin importar la distribución de los nodos o el coste de las aristas.

2.1.2.5 Modelacion Hidrologica Precipitacion-Escorrentia, usando el modelo de Thomas y algoritmos genéticos en la subcuenca del río Qui Vire

Saucedo Oxacopa. M (2014) en la tesis sustentada en la "Universidad Nacional Jorge Basadre Grohmann-Tacna" con el título de "Modelación Hidrológica Precipitación-Escorrentia, usando el modelo de Thomas y algoritmos genéticos en la subcuenca del río Qui Vire" tuvo como objetivo principal emplear los algoritmos genéticos (Hill Climbing) para calibrar el modelo hidrológico precipitación - escorrentía de Thomas. Utilizó un diseño cuasiexperimental que consta de un plan de trabajo para estudiar el impacto de los procesos de cambio en diversas situaciones y mejoras de optimización. Su principal resultado se rige bajo la evaluación de algunos parámetros importantes como la superficie, elevación, pendiente de la cuenca que serán trabajados en MATLAB junto a los algoritmos genéticos que incluye la lógica del algoritmo de ascenso de colina y la correcta función a minimizar se logra una comparación de dos simulaciones. Finalmente, se concluye que los algoritmos genéticos tienen propiedades que los hacen muy robustos para resolver problemas de optimización, toma de decisiones, diseño y otros usos en la ingeniería. En el presente trabajo al aplicar los algoritmos genéticos y graduar la ecuación tenemos como resultados datos muy semejantes a la realidad.

2.1.3. Antecedentes Locales

2.1.3.1 Implementación de un sistema de información que apoye el proceso diario de elaboración de cronogramas del personal de medicina fisica y rehabilitacion de un hospital

Marquez Redhead, J.M (2013) en la tesis sustentada en la "Pontificia Universidad Catolica del Peru" con el titulo "Implementación de un sistema de información que apoye el proceso diario de elaboración de cronogramas del personal de medicina fisica y rehabilitacion de un hospital" tuvo como objetivo central la implementación de un sistema de información que funcione como soporte en el desarrollo de cronogramas de terapias, posibilitando la planificación automática y eliminación de citas de pacientes que abandonan su tratamiento, dentro del área de medicina fisica y rehabilitacion de un hospital, específicamente en el objetivo tres se busca implementar un algoritmo Hill-Climbing para solucionar el problema de programación dinámica de horarios. Utilizó un diseño cuasiexperimental que consta de un plan de trabajo para estudiar el impacto de los procesos de cambio en diversas situaciones. Su principal resultado es la correcta implementación de una nueva interfaz gráfica orientada a los pacientes, personal y el registro de historia clínica, medicamentos y turnos usando python (framework django 1.5.1) y MySql. Finalmente, referente al algoritmo se llegó a concluir que se consiguió la implementación del algoritmo de optimización Hill-Climbing que soluciona el problema de programación dinámica de horarios para citas de terapia física y/o rehabilitación, se basa en el cálculo de una función de valor que está conformada por los días de espera antes de aceptar la primera terapia; y también valora otras consideraciones como la prioridad de algún tipo de tratamiento, turnos de trabajo disponibles y otras acciones por parte del tecnólogo.

2.1.3.2. Algoritmo genético para la elaboración de horario de exámenes en universidades

Zelada Cabezudo, H. E. (2018) en la tesis de la Universidad Científica del Sur, titulada "Algoritmo genético para la elaboración del horario de exámenes en universidades" tuvo como objetivo la implementación de una aplicación que permita generar los horarios de exámenes del primer cuatrimestre, el segundo, y la convocatoria extraordinaria, de las dos titulaciones "Doble grado en informática - ADE" y "Grado en ingeniería informática". El tipo de investigación que se realizó corresponde a un estudio correlacional, donde se mide la variable Independiente (algoritmo genético) con la variable dependiente (elaboración del Horario de exámenes) y poder someter a prueba las hipótesis establecidas en esta tesis. El algoritmo alcanza un 91% de efectividad en asignación de aulas y docentes. El algoritmo genético muestra los primeros resultados válidos a los 10 minutos en promedio a comparación de expertos, que los primeros resultados lo tienen en 3 a 5 días. Aplicando la combinación de técnicas de algoritmos entre la búsqueda tabú y algoritmo genético recomendado por R. raghavjee and N. Pillay en su publicación del libro "An Application of Genetic Algorithms to the School Timetabling Problema SAICSIT 2008". Moldea el algoritmo a la realidad de restricciones que tiene la facultad. Concluyendo que para que los algoritmos sean eficientes tienen que ser adaptados, de lo contrario no se podrá realizar con eficiencia la optimización deseada.

2.1.3.3. Implementación de un algoritmo genético para elaborar un conjunto de rutas óptimas para el transporte de la comunidad universitaria desde y hacia el campus principal

Castillo (2018) en su tesis titulada "Implementación de un algoritmo genético para elaborar un conjunto de rutas óptimas para el transporte de la comunidad universitaria desde y hacia el campus principal" para optar por el Título de Ingeniero Informático en la Pontificia Universidad Católica del Perú, tuvo como objetivo general implementar un algoritmo genético, el cual se encargue de elaborar un conjunto de rutas óptimas para el transporte de la

comunidad universitaria, ya sea alumnos, docentes o personal administrativo, este algoritmo debe ir desde y hacia el campus principal. En la presente investigación se usaron distintas herramientas informáticas y algoritmos de búsquedas, al hacer distintas evaluaciones se llegó al algoritmo Pair Insertion Algorithm con el que se elaboró una serie de rutas óptimas para el transporte de la comunidad universitaria. Como resultados se obtuvo que el algoritmo implementado alcanzó mejoras del 20% en promedio, lo cual significa que este algoritmo es más flexible a la hora de configurar la entidad a la que se desea generar las rutas, dando prioridad a la cantidad de usuarios o la distancia en la que se encuentra cada uno de ellos.

2.1.3.4. Optimización del transporte urbano en Lima aplicando los algoritmos genéticos Tabú y Colonia de Hormigas.

Benites et. al (2021) en su trabajo de investigación en la Universidad Privada del Norte titulada "Optimización del transporte urbano en Lima aplicando los algoritmos genéticos Tabú y Colonia de Hormigas", tuvo como objetivo analizar los métodos de optimización de rutas de transporte urbano en la ciudad de Lima mediante la aplicación de los algoritmos genéticos Tabú y Colonia de Hormigas y también conocer los beneficios que implica su implementación. Esta investigación se llevó a cabo sobre la base de una revisión sistemática que recopila artículos científicos del mundo académico para así realizar una amplia gama de métodos para aplicar el "algoritmo genético" al "transporte urbano". En cuanto a los resultados, la búsqueda de artículos en las bases de datos y motores de búsqueda arrojó un total de 14 artículos que se relacionan con el título y objetivo de la investigación, el cual el 26% de los artículos pertenecen a Perú y Colombia, debido a la alta investigación que se realiza en las universidades de estos países, gracias a la congestión vehicular que ha ido creciendo en los últimos años. Se llegó a la conclusión que el método que se debe implementar para optimizar el transporte urbano en la ciudad de Lima es el algoritmo tabú

debido a que es muy fácil su implementación, tiene mayor respuesta y realiza la búsqueda de una manera inteligente.

2.1.2.5.Construcción de un componente de software para la búsqueda del camino más corto y el control de movimiento en un videojuego de estrategia en tiempo real.

Montesinos (2019) en la tesis de la Pontificia Universidad Católica del Perú con el título "CONSTRUCCIÓN DE UN COMPONENTE DE SOFTWARE PARA LA BÚSQUEDA DEL CAMINO MÁS CORTO Y EL CONTROL DE MOVIMIENTO EN UN VIDEOJUEGO DE ESTRATEGIA EN TIEMPO REAL" tuvo como objetivo central realizar el análisis, diseño e implementación de un componente de búsqueda de caminos y control de movimiento, para implementar un mecanismo de navegación autónoma de los personajes en un videojuego de estrategia en tiempo real en dos dimensiones. Para el presente trabajo se utilizó PMBOK, y para la implementación del producto la metodología ágil Extremme programming. Se describe los movimientos y mecanismos de interacción que se requieren implementar asimismo se diseñó una arquitectura de software necesaria para gestionar el movimiento e interacción de los personajes, incluyendo las clases y tipos de datos que permitan adaptar los algoritmos de búsqueda de camino A* y HPA* aplicados a un videojuego RTS. Se llegó a concluir que cada uno de los resultados esperados cumple con las principales características de un videojuego RTS, además de el correcto funcionamiento de arquitectura diseñada y que mediante dos muestras independientes se pudo comprobar que la media muestral del tiempo de ejecución del algoritmo HPA* es menor que la del A*, así como el uso de espacio de memoria por parte del algoritmo HPA* es menor que la del A*.

2.2. Teorías Relacionadas

2.2.1. Conceptos De Agentes Artificiales

Maquen (2021) menciona que un agente artificial es “cualquier cosa que pueda decirse que percibe su entorno a través de sensores y actúa sobre ese entorno a través de actuadores” lo que significa que es un ente o sujeto autónomo, el cual percibe todo su entorno y estas reaccionan o responden de manera racional. Todo esto se da gracias a los sensores los cuales se encargan de percibir su medio ambiente y los actuadores que reaccionan a ese estímulo.

Debido al gran avance en la rama de la IA se podría considerar que un agente artificial podría tener reacciones como expresiones faciales ante cualquier situación emocional que se le presente, las cuales son propias del ser humano, tal como Hortensius, Hekele y Cross (2018) señalan que “Artificial agents are becoming progressively better at reading and appropriately responding to emotions expressed by humans, with many artificial agents programmed to display emotional reactions, such as sadness or joy.” [Los agentes artificiales son cada vez mejores para leer y responder adecuadamente a las emociones expresadas por los humanos, con muchos agentes artificiales programados para mostrar reacciones emocionales, como tristeza o alegría].

2.2.1.1. Estructura Del Agente

Maquen (2021) sostiene que “este programa se ejecutará en algún tipo de dispositivo informático con sensores y actuadores físicos; a esto lo llamamos la arquitectura”, donde se señala que el programa es el objetivo de la IA donde se busca diseñar e implementar un programa para lo que es el mapeo de su entorno a través de las percepciones y presentar una acción de respuesta. Es por ello que se menciona la siguiente ecuación:

Agente = arquitectura + programa

2.2.2. Algoritmo Ascensión de Colina

"El algoritmo de ascensión es simplemente un bucle que continuamente se mueve en dirección del valor creciente, es decir cuesta arriba. Termina cuando alcanza un pico donde ningún vecino tiene un valor más alto" (Russell y Norving, 2004, p. 126). Este algoritmo se inicia a partir de una solución arbitraria, mediante el cual empezará un bucle que, en cada paso del proceso, el nodo actual se dirigirá hacia el vecino con el mejor valor, el cuál será el nodo con mayor o menor valor con respecto al nodo actual dependiendo de lo que se busque. El proceso finaliza cuando se encuentra un nodo que no posea nodos vecinos con mejor valor. La ascensión de colinas solo mira a los vecinos inmediatos del estado actual. Este algoritmo también es llamado voraz local porque toma un estado vecino sin pensar en la próxima acción.

El algoritmo puede representarse de la siguiente manera:

```
Algoritmo Hill Climbing
        Actual = Estado_ínicial
        fin = falso
        Mientras ¬fin hacer
                Hijos = generar_sucesores(Actual)
                Hijos = ordenar_y_eliminar_peores(Hijos, Actual)
                si ¬vacio?(hijos) entonces
                        Actual = Escoger_mejor(Hijos)
                sino
                        fin = cierto
        fMientras
fAlgoritmo
```

El algoritmo de ascensión de colina no garantiza que se encuentre el mejor valor global, ya que depende del valor inicial que se utilice. "En funciones con muchos picos (funciones multimodales) es probable que el algoritmo se detenga en el primer pico que encuentre, incluso si no es el más alto. Una vez que se alcanza un pico, ascensión de colinas ya no puede progresar, y eso es problemático cuando este punto es un óptimo local."

(Sivanandam y Deepa, 2007, p. 27). Este problema puede ser remediado ejecutando reinicios o esquemas más complejos basados en iteraciones.

Ascensión de colinas suele atascarse por los siguientes motivos:

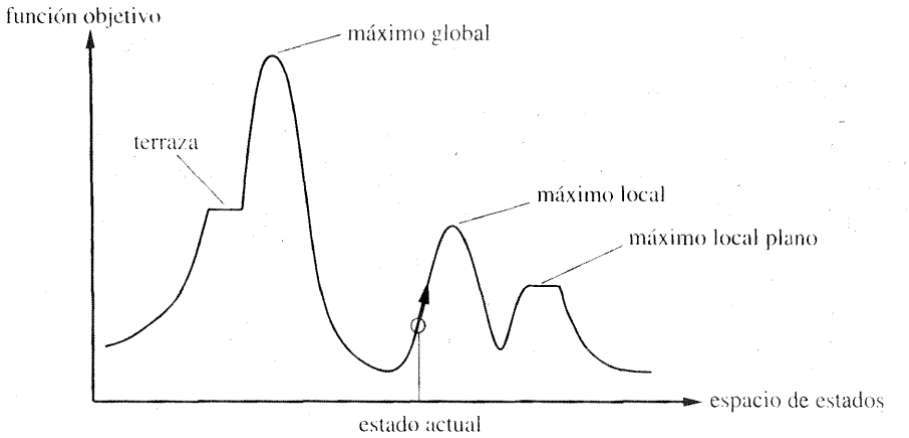
Máximo Local: Soria et al. (2017) define a un máximo local como Picos del paisaje cuya altura es menor que el máximo global. (p. 106)

Crestas: "Las crestas causan una secuencia de máximos locales que hace muy difícil la navegación para los algoritmos avaros" (Russell y Norving, 2004, p. 129). Por lo anterior dicho, al haber una alta cantidad de máximos locales es muy probable que no se llegue a el máximo global.

Meseta: Russell y Norving (2004) describen a una meseta como un área del paisaje del espacio de estados donde la función de evaluación es plana. Puede ser un máximo local plano, del que no existe ninguna salida ascendente, o una terraza, por la que se pueda avanzar. Una búsqueda de ascensión de colinas podría ser incapaz de encontrar su camino en la meseta. (p. 129)

Figura 2

Descripción de ascenso de colina.



Fuente: Russell y Norving (2004)

Algunas variantes del algoritmo ascensión de colinas son:

Ascensión de colinas estocástica: “Escoge aleatoriamente de entre los movimientos ascendentes; la probabilidad de selección puede variar con la pendiente del movimiento ascendente.” (Russell y Norving, 2004, p. 129). Esta variante suele tardar más en encontrar el valor óptimo, ya que no siempre escogerá el camino más rápido.

Ascensión de colinas de primera opción: “Se selecciona un vecino en forma aleatoria. Si el vecino es mejor que el actual, se hace actual = vecino y el proceso continúa. Si no es mejor, se selecciona otro en forma aleatoria.” (Gonzáles et. al, 2022, consulte la Sección 7.3)

Ascensión de colinas de reinicio aleatorio: “Conduce a una serie de búsquedas en ascensión de colinas desde los estados iniciales generados aleatoriamente, parándose cuando se encuentra un objetivo.” (Russell y Norving, 2004, p. 129).

2.2.3. Herramientas tecnológicas

2.2.3.1. Sistema Web

Según Luciano (2018) “Se denomina sistema web a aquellas aplicaciones de software que se pueden usar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador”. (p. 9)

Asimismo, Cabanilla y Rodríguez (2017), mencionan que es una aplicación que facilita la comunicación entre un cliente y un servidor en tiempo real. De igual modo permite realizar el mantenimiento a través de la red, consume escasos recursos del lado del cliente, pues la aplicación se instala en otro ordenador, siendo este un servidor web.

2.2.3.2. Lenguajes De Programación

2.2.3.2.1. PHP. Martín et. al (2021) señalan que PHP es un lenguaje de scripting diseñado específicamente para aplicaciones Web. El código PHP es interpretado en el servidor Web cuando un documento HTML en el que está embebido es solicitado por un navegador. El código PHP normalmente genera como salida código HTML que reemplaza al código PHP. Por tanto, un navegador web nunca llega a ver código PHP.

2.2.3.2.2. JavaScript. Pérez (2019) señala que JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Además, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (p. 5)

2.2.3.2.3. Python. Martín et. al (2021) señalan que "Python es un lenguaje de programación muy popular, especialmente para computación científica, ciencia de datos e inteligencia artificial. Además, soporta algunos de los paradigmas de programación más populares, facilita la programación concurrente y se usa para el desarrollo web". (p. 27)

2.2.3.3. Otros Lenguajes

2.2.3.3.1. HTML. Se afirma que el Hyper Text Markup Language "Se encarga de definir la estructura del documento de tal forma que establecen encabezado y contenido mediante etiquetas lo cual permite crear párrafos, encabezados, enlaces, imágenes y listas con el fin de personalizar la presentación" (Gauchat , 2017, p. 25).

2.2.3.3.2. CSS. Para definir CSS se considera que “Es un lenguaje de hojas de estilos creado para controlar el aspecto de los documentos definidos con HTML lo cual permite cambiar las propiedades de maquetación, bordes, estilo de texto, alineación del texto, color y fondo de las páginas web”. (Velázquez, 2019, p. 25)

2.2.3.4. Servidor

2.2.3.4.1. XAMPP. Galvan (2017) lo define como “Un servidor independiente de plataforma, software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web apache y los intérpretes para lenguajes de script: PHP y PERL” (p. 26).

2.2.3.5. Google Maps API

Alcón et. al (2008) señalan que se trata de una tecnología que permite la visualización de Google Maps en tus propias páginas web con JavaScript. El API proporciona unas determinadas herramientas para interaccionar con los mapas y añadir contenido a los mismos a través de una serie de servicios, permitiendo llegar a crear aplicaciones con mapas de gran complejidad y robustez.

Capítulo 3: Desarrollo de la propuesta

3.1. Codificación del software: Algoritmo Ascenso de Colina (Hill Climbing Algorithm)

El algoritmo HC utilizado en el presente trabajo para resolver el problema de diseño de rutas, está posicionado dentro de los tipos de algoritmos heurísticos de búsqueda local. Este empieza con una solución inicial aleatoria que se mueve iterativamente hacia la mejor solución "vecina" hasta el término de una solución que posteriormente ya no se puede mejorar. Bajo ese concepto, el algoritmo es denominado "ascenso o escalada de colinas", porque la progresión anteriormente mencionada se asemeja a subir una colina hasta llegar a la cima.

3.1.1. Solución inicial aleatoria

Siguiendo la lógica del negocio, la solución inicial aleatoria se obtuvo considerando la ubicación de cada una de las agencias disponibles de la empresa, dicha ubicación toma como referencia la latitud y longitud de las mismas. Gracias a esta información se podrán formar los nodos que, de manera aleatoria, serán seleccionados para la formación de la ruta candidata en cada iteración; este proceso continúa hasta llegar a una solución que satisfaga el objetivo de brindar la ruta más óptima que pase por cada uno de los nodos.

3.1.2. Vecinos

Los nodos vecinos pueden afectar al método de búsqueda ya que dentro de la ruta poseen distancias que pueden ser seleccionadas o no por la solución actual, influyendo considerablemente en el rendimiento y calidad de la solución por parte del algoritmo.

3.1.3 Cobertura

Dentro del objetivo de encontrar la ruta óptima entre los nodos establecidos, la cobertura muestra la distancia total de la ruta como solución. En ese sentido lo que se considera es la distancia establecida entre dos nodos como resultado de una fórmula que tiene como parámetros de entrada los valores de la latitud y longitud de cada uno de ellos.

3.1.4. El algoritmo Ascenso de colina aplicado a la lógica del negocio

Se consideran estos pasos para llevar a cabo la codificación del algoritmo:

- Establecer los nodos de la red como parte de la información del problema desde la entrada.
- Generar una solución inicial aleatoria como 'solución actual'
- Evaluar la cobertura para todos los vecinos de la solución actual para poder designar al vecino con la máxima cobertura como el mejor vecino.
- Si la cobertura del mejor vecino es menor que la de la solución actual, se considera mejor solución, se establece como solución actual y vuelve al paso 3. En caso contrario, se arroja la solución actual como la mejor solución del algoritmo en la salida.

Dichos pasos se pueden codificar en el lenguaje Python

```
import math
import random
from math import sin, cos, asin, acos
from geopy.distance import geodesic

def distancia(coord1,coord2):
  lat1=coord1[0]
  lon1=coord1[1]
  lat2=coord2[0]
  lon2=coord2[1]
  c1=((lat1,lon1))
  c2=((lat2,lon2))
  return geodesic(c1,c2).km

#Calcular distancia cubierta por una ruta
def evalua_ruta(ruta):
  total=0
```

```
  for i in range(0,len(ruta)-1):
    ciudad1=ruta[i]
    ciudad2=ruta[i+1]
    total=total+distancia(coord[ciudad1],coord[ciudad2])
    ciudad1=ruta[i+1]
    ciudad2=ruta[0]
    total=total+distancia(coord[ciudad1],coord[ciudad2])
  return total

def i_hill_climbing():
  #crear ruta inicial aleatoria
  ruta=[]
  for ciudad in coord:
    ruta.append(ciudad)
  mejor_ruta=ruta[:]
  max_iteraciones=8

  while max_iteraciones>0:
    mejora=True
    #Generamos una nueva ruta aleatoria
    random.shuffle(ruta)
    while mejora:
      mejora=False
      dist_actual=evalua_ruta(ruta)
      #evaluar vecinos
      for i in range(0,len(ruta)):
        if mejora:
          break
        for j in range(0,len(ruta)):
          if i != j:
            ruta_tmp=ruta[:]
            ciudad_tmp=ruta_tmp[i]
            ruta_tmp[i]=ruta_tmp[j]
            ruta_tmp[j]=ciudad_tmp
            dist=evalua_ruta(ruta_tmp)
            if dist<dist_actual:
              #Encontrado vecino que mejora el resultado
              mejora=True
              ruta=ruta_tmp[:]
              break
    max_iteraciones=max_iteraciones-1
    if evalua_ruta(ruta)<evalua_ruta(mejor_ruta):
      mejor_ruta=ruta[:]
  return mejor_ruta


coord={
    'Sede Javier Prado':[-12.089218091755795, -77.01574868971116],
    'Sede 28 de Julio':[-12.063226877463787, -77.03352476099145],
    'Sede Atocongo':[-12.156600691212441, -76.98371335902573],
    'Sede Terminal Plaza Norte':[-12.005051785087128, -
77.05494476760562],
    'Sede Arequipa Central':[-16.42303396711053, -71.5453360854306],
    'Sede Arequipa Camaná':[-16.402741432272975, -71.51235965474616],
    'Sede Piura Loreto':[-5.201091472336525, -80.63118770328701],
    'Sede Piura Talara':[-4.586560718171606, -81.27041074562112]
}

ruta=i_hill_climbing()
d=print(ruta)
e=print("Distancia total: "+str(evalua_ruta(ruta)))
```

3.1.5. Implementación de la API de Google Maps

Para mostrar el mapa con las terminales de la empresa en la página web de nuestra aplicación, se consideró los siguientes pasos:

1. Como primer paso, se instancia la variable “map” de la clase Map de google.
2. Para mostrar el mapa se hace una referencia al elemento HTML que hará de contenedor con el método document.getElementById(“map”).
3. Luego, se crea la variable “bounds” en el cual se almacena las instancias de cada punto (terminal) tomando en cuenta su longitud y latitud.
4. En el arreglo “markers” guardamos el nombre de cada sede o terminal con la que cuenta la empresa, además de su respectiva longitud y latitud.

Figura 2

Registro de cada marker o terminal.

```
var map=new google.maps.Map(document.getElementById("map"));
var bounds = new google.maps.LatLngBounds();
// Multiple markers location, latitude, and longitude
var markers = [
    ['Sede Javier Prado',-12.089218091755795, -77.01574868971116],
    ['Sede 28 de Julio',-12.063226877463787, -77.03352476099145],
    ['Sede Atocongo',-12.156600691212441, -76.98371335902573],
    ['Sede Terminal Plaza Norte',-12.005051785087128, -77.05494476760562],
    ['Sede Arequipa Central',-16.42303396711053, -71.5453360854306],
    ['Sede Arequipa Camaná',-16.585170973998583, -72.70553736332874],
    ['Sede Piura Loreto',-5.201091472336525, -80.63118770328701],
    ['Sede Piura Talara',-4.586560718171606, -81.27041074562112],
    ['Sede Abancay', -13.639179280953257, -72.88528518972556],
    ['Sede Terminal Bagua Grande', -5.610295414784535, -78.43606746099624],
    ['Sede Terminal Cajamarca', -7.17104578365114, -78.50138495686839],
    ['Sede Terminal Bolognesi Chiclayo', -6.775742750626063, -79.83828664670624],
    ['Sede Terminal Cusco', -13.533136022933917, -71.97171081885351],
    ['Sede Terminal Huaraz', -9.525636997019953, -77.52745021214375],
    ['Sede Terminal Ica', -14.063957987457615, -75.73318988968508]
];
```

Fuente: Elaboración propia.

5. Para colocar cada marcador en el mapa se recorre el arreglo marker a través de un bucle for. Dentro del bucle se almacenan las coordenadas de cada marker en la

variable “position” para luego mostrar cada punto en el mapa con el método fitBounds().

Figura 3

Presentación de cada marker o terminal en el mapa.

```
// Place each marker on the map
for( i = 0; i < markers.length; i++ ) {
    var position = new google.maps.LatLng(markers[i][1], markers[i][2]);
    bounds.extend(position);
    marker = new google.maps.Marker({
        position: position,
        map: map,
        title: markers[i][0]
    });

    map.fitBounds(bounds);
}
```

Fuente: Elaboración propia.

Para trazar las rutas en el mapa desde un punto de inicio a un punto final con la condición de pasar por ciertas terminales, se codificó de la siguiente manera:

1. Primero se crean las constantes “directionsService” y “directionsRender” que se utilizan para trazar las rutas y representarlas en el mapa respectivamente.

Figura 4

Llamado a la función calculateAndDisplayRoute().

```
const directionsService = new google.maps.DirectionsService();
const directionsRenderer = new google.maps.DirectionsRenderer();
directionsRenderer.setMap(map);

document.getElementById("submit").addEventListener("click", () => {
  calculateAndDisplayRoute(directionsService, directionsRenderer);
});
```

Fuente: Elaboración propia.

2. Luego de señalar el punto de inicio, los puntos intermedios y el destino final en la interfaz se hace llamado a la funcion “calculateAndDisplayRoute()” el cual se envían con los parámetros “directionsService” y “directionsRender”.
3. Ya en la función, se crea el arreglo “waypts” el cual almacena los puntos intermedios que fueron seleccionados por el usuario.

Figura 5

Desarrollo de la función calculateAndDisplayRoute().

```
function calculateAndDisplayRoute(directionsService, directionsRenderer) {
  const waypts = [];
  const checkboxArray = document.getElementById("waypoints");

    for (let i = 0; i < checkboxArray.length; i++) {
      if (checkboxArray.options[i].selected) {
        waypts.push({
          location: checkboxArray[i].value,
          stopover: true,
        });
      }
    }
```

Fuente: Elaboración propia.

4. Una vez almacenada toda la información, se procede a trazar la ruta a través del método “route()” el cual tiene el punto de origen, destino, las terminales intermedias y el tipo de modo de viaje, en este caso “Driving”.

Figura 6

Trazar la ruta según los puntos a recorrer.

```
  directionsService
    .route({
      origin: document.getElementById("start").value,
      destination: document.getElementById("end").value,
      waypoints: waypts,
      optimizeWaypoints: true,
      travelMode: google.maps.TravelMode.DRIVING,
    })
    .then((response) => {
      directionsRenderer.setDirections(response);
```

Fuente: Elaboración propia.

5. Por último , se muestra en la interfaz la ruta de punto a punto con su respectiva distancia.

Figura 7

Mostrar la ruta obtenida.

```
const route = response.routes[0];
const summaryPanel = document.getElementById("directions-panel");

summaryPanel.innerHTML = "";

// For each route, display summary information.
for (let i = 0; i < route.legs.length; i++) {
  const routeSegment = i + 1;

  summaryPanel.innerHTML +=
  `<div class="ruta">
    <h3 class="numeroRuta">Segmento de ruta: ${routeSegment}</h3>
    <div class="ruta-descripcion"
      <span>${route.legs[i].start_address}<b> hacia</b></span>
      <span>${route.legs[i].end_address}</span>
      <p class="distancia">Distancia: ${route.legs[i].distance.text}</p>
    </div>
  </div>`
}
```

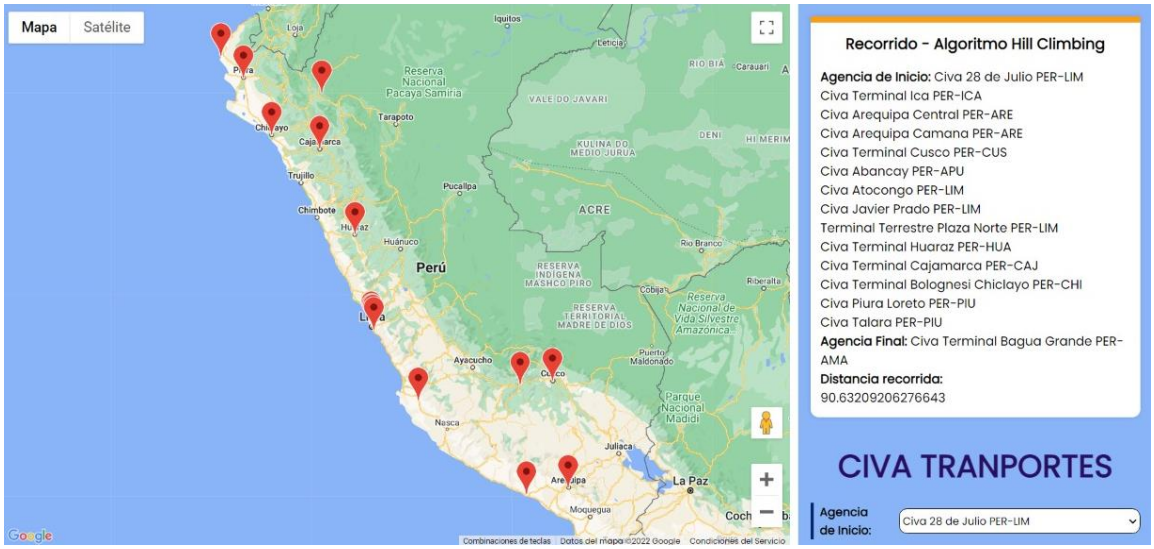
Fuente: Elaboración propia.

3.2. Resultados

Una vez realizada la codificación y la adaptación lógica del algoritmo al negocio, la página web muestra el accionar de la API de Google Maps bajo la implementación del Algoritmo Hill Climbing.

Figura 6

Vista general de la página

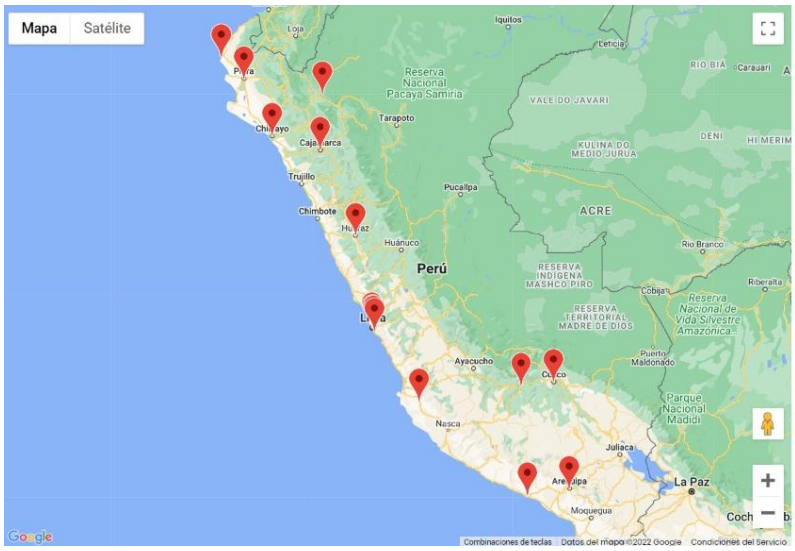


Fuente: Elaboración propia.

La interfaz contiene la muestra gráfica de los nodos establecidos para el cálculo de la ruta, en el contexto de negocio, dichos nodos refieren a las agencias de la empresa Civa S.A. ubicadas a nivel nacional.

Figura 7

Establecimiento de nodos



Fuente: Elaboración propia.

Posteriormente, gracias a la implementación del algoritmo en Python, se genera la ruta óptima tomando en cuenta cada uno de los nodos, sus vecinos, y la distancias entre ellos.

Figura 8

Implementación del algoritmo de Hill Climbing

Fuente: Elaboración propia.

Dicho resultado de ruta se encuentra lista para ser graficada en el mapa virtual. Haciendo clic en el botón Enviar.

Figura 9

Interfaz de generación de ruta

Fuente: Elaboración propia.

Una vez presionado el botón, se puede observar el gráfico de la ruta brindada por el algoritmo de Ascenso de Colina.

Figura 10

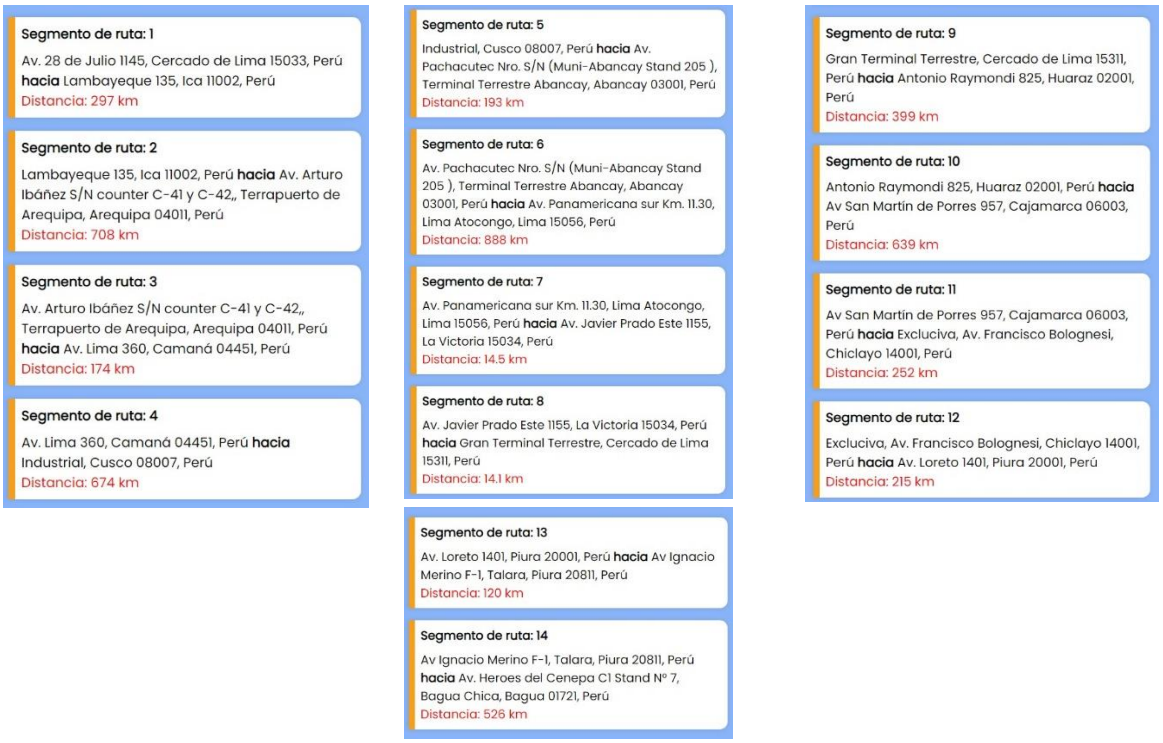
Gráfico de ruta

Fuente: Elaboración propia.

También se puede observar en la parte derecha inferior, el detalle del recorrido a través de cada nodo

Figura 11

Detalle de recorrido



Fuente: Elaboración propia.

Al hacer la evaluación del algoritmo i_hill_climbing() se observa que la complejidad temporal de este es de orden $O(N^5)$, ya que incluye 4 bucles iterativos y una función dentro de la última iteración la cual contiene el método evalua_ruta, cuyo orden es lineal.

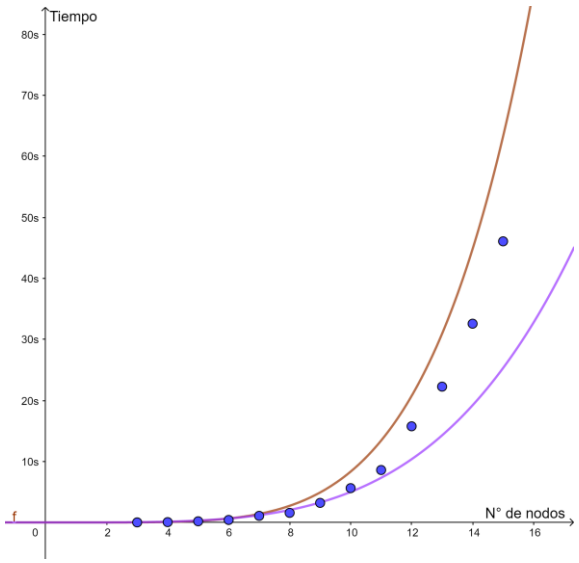
Se realizaron diferentes pruebas del algoritmo en una misma computadora Lenovo de 8 GB de RAM en el sistema operativo Windows 10, modificando el número de nodos de 3 hasta 15, iterando 99 veces con cada uno para poder hallar un promedio de la variación del tiempo.

N° de nodos	Tiempo transcurrido en segundos	Promedio de tiempo por iteración en segundos
3	0.0139122009	0.00014053
4	0.0566909313	0.00057264
5	0.1941697598	0.00196131
6	0.4148068428	0.00418997
7	1.0842208862	0.01095173
8	1.5767636299	0.01592691
9	3.2106478214	0.03243079
10	5.6247177124	0.05681533
11	8.5993726254	0.08686235
12	15.7618877888	0.15921099
13	22.2597634792	0.2248461
14	32.5745806694	0.32903617
15	46.0749831200	0.46540387

En la tabla anterior se muestran los diferentes resultados obtenidos al efectuar las pruebas donde podemos notar que a mayor número de nodos hay un mayor tiempo de ejecución. A continuación se muestra una gráfica del tiempo de duración de 99 iteraciones en función de los nodos utilizados, donde se expresan los datos de la tabla anterior. Se colocaron 2 curvas, una de orden 4 (en la parte inferior) y otra de orden 5 (en la parte superior).

Figura 11

Gráfica de tiempo



Fuente: Elaboración propia.

Para medir la precisión del algoritmo se calculó el número de veces que el algoritmo logra encontrar la menor distancia total en 1000 iteraciones. Los resultados se encuentran en la siguiente tabla.

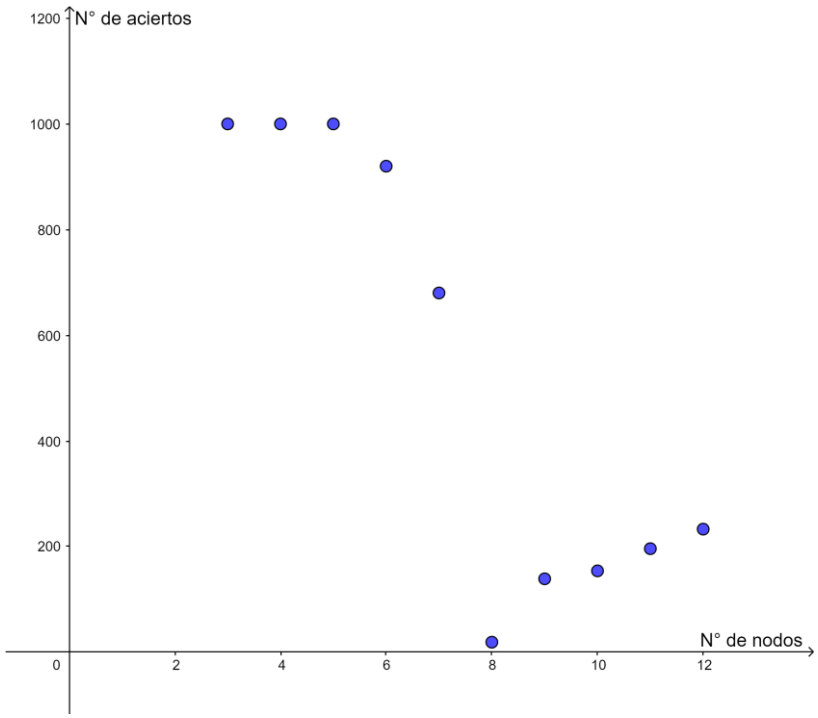
N° de nodos	Número de aciertos	Porcentaje de aciertos
3	1000	100%
4	1000	100%
5	1000	100%
6	920	92%
7	680	68%
8	19	1.9%
9	139	13.9%
10	154	15.4%
11	196	19.6%
12	233	23.3%

Con los datos de la tabla anterior se aprecia que al incrementar la cantidad de nodos de la prueba la cantidad de aciertos en los que el algoritmo logra la menor distancia disminuye. Con la cantidad de 8 nodos el número de aciertos disminuye drásticamente y a partir de ahí comienza a aumentar en menor medida debido a que el número máximo de iteraciones depende del número de nodos que se utilice.

En el siguiente gráfico se muestra el número de aciertos obtenido en función del número de nodos utilizados para la prueba en 1000 iteraciones.

Figura 11

Gráfica aciertos en función a nodos.



Fuente: Elaboración propia.

Conclusiones

Lo explicado a lo largo de este proyecto permite presentar las siguientes conclusiones:

- Según el apartado de antecedentes donde se investiga trabajos con similitud a lo propuesto y conceptos en el marco teórico, además de la implementación del código, se logró el correcto estudio del algoritmo de optimización “Ascenso de Colina” útil para la resolución del problema de búsqueda del camino más corto entre dos puntos de la red de envío de cargos y encomiendas de la empresa de transportes CIVA.

- Con el uso correcto de las herramientas tecnológicas propuestas para el caso se logró desarrollar un código de aplicación web de manera local que muestre la implementación del algoritmo “Ascenso de Colina” de manera gráfica dando posibilidad a saber el camino más corto entre dos sedes de la empresa de transportes CIVA, ya que dentro de sus funciones se encuentran el envío de cargos y encomiendas.

- Gracias a la comprensión del algoritmo de “Ascenso de Colina” y su implementación en el lenguaje Python se logró adaptar la lógica de este algoritmo dentro del caso propuesto para habilitar la ruta más corta entre dos sedes de la empresa CIVA.

- Dentro del plano de la codificación tenemos que se logra establecer los nodos considerados las direcciones de las sedes de la empresa de transportes CIVA con sus respectivas coordenadas que serán tomados en cuenta tanto en la parte lógica como en la muestra gráfica en la aplicación web.

- Dentro del plano de las pruebas e implementación se logró identificar las dificultades y/o errores de la adaptación del algoritmo “Ascenso de Colina” para el caso propuesto, específicamente en la correcta digitación de las coordenadas y nombre de las sedes para la búsqueda de ruta óptima.

- Respecto al apartado de resultados, tenemos que se logró implementar y comprobar la eficacia del algoritmo de búsqueda “Ascenso de Colina” en un determinado sistema como en este caso.

Recomendaciones

- Antes de iniciar este tipo de proyecto se recomienda analizar las herramientas tecnológicas a usar y seleccionar las más adecuadas para el proceso, además de verificar la correcta instalación de estas.

- Realizar un bosquejo o prototipo de la aplicación web a implementar, durante el proceso ir modificando ubicación, botones, contraste y diseño para una mejor visualización del público.

- Dentro de la codificación es importante tomar en cuenta la participación de los nodos, ya que estos son considerados como las sedes de la empresa de transporte CIVA, en ese sentido, se recomienda ser exactos con sus coordenadas y el nombre de la sede dentro de Google Maps.

- Respecto a la obtención de la API de Google Maps para la aplicación web, se recomienda utilizar una tarjeta debito solicitada en cualquier entidad bancaria o de algún familiar cercano. Además, investigar sobre la posibilidad de mostrar de manera gráfica con el uso de otra API que sea gratuita y no requiera registro de tarjetas.

- Para un mayor análisis se recomienda la implementación de otros algoritmos de búsqueda para diferenciar y saber cual es el más eficiente en este tipo de casos y no solo aplicarlo para empresas de transportes sino también para una escala social como rutas de colegios, avenidas principales, etc.

Bibliografía

Alcón, J., Arauz, F. J., & Carmona, I. (2008). Aplicación web para la geolocalización y monitorización en tiempo real de los recursos integrantes de una red grid [Trabajo de curso, Universidad Complutense de Madrid]. Repositorio Institucional de la UCM. https://eprints.ucm.es/id/eprint/9085/1/Memoria.pdf

Anquise Jihuaña, Y. (2019). Implementación de un modelo matemático para la planificación de un viaje personalizado basado en optimización multiobjetivo para los turistas en la región Puno.

Arias, F. (2012). El proyecto de investigación. Introducción a la metodología científica (6ta ed.). Venezuela: Editorial Episteme

Benites, A., Campos, N., Sotomayor, J., Rabanal, E. & Perez, C. (2021). Optimización del transporte urbano en Lima aplicando los algoritmos genéticos Tabú y Colonia de Hormigas. 19th LACCEI International Multi-Conference for Engineering, Education, and Technology: "Prospective and trends in technology and skills for sustainable social development" "Leveraging emerging technologies to construct the future". http://dx.doi.org/10.18687/LACCEI2021.1.1.321

Cabanilla, E. & Rodríguez, R. (2017). Implementación de aplicación web para la gestión de rutas de operación vehicular empresa de transporte pesado trans LE&MA SA [Tesis de grado, Universidad Politécnica Salesiana]. Repositorio Institucional de la UPS. https://dspace.ups.edu.ec/handle/123456789/14544

Carbajal Montesinos, H. (2017). CONSTRUCCIÓN DE UN COMPONENTE DE SOFTWARE PARA LA BÚSQUEDA DEL CAMINO MÁS CORTO Y EL CONTROL DE MOVIMIENTO EN UN VIDEOJUEGO DE ESTRATEGIA EN TIEMPO REAL (Bachiller). Pontificia Universidad Católica del Perú.

Castillo, J. (2018). Implementación de un algoritmo genético para elaborar un conjunto de rutas óptimas para el transporte de la comunidad universitaria desde y hacia el campus principal.

Galipienso, A., Isabel, M., Cazorla Quevedo, M. A., Colomina Pardo, O., Escolano Ruiz, F., & Lozano Ortega, M. A. (2003). Inteligencia artificial: modelos, técnicas y áreas de aplicación. Editorial Paraninfo.

Galvan, R. (2017). Aplicación web para la gestión de rutas, distancias y geolocalizaciones del cableado de fibra óptica y sucursales [Tesis de grado, Universidad Tecnológica del Centro de Veracruz]. Repositorio Institucional de la UTCV. http://reini.utcv.edu.mx/handle/123456789/668

Gauchat , J. D. (2017). HTML5 para Mentes Maestras: Cómo aprovechar HTML5 para crear sitios web adaptables y aplicaciones revolucionarias (2da ed.). JD Gauchat. https://books.google.es/books?id=oH2xDgAAQBAJ&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

González Guerra, L. H., Pérez Murueta, P. Ó., & Cueva Hernández, V. M. Algoritmos: análisis, diseño e implementación.

Gosavi, A. (2003). “Simulation Based Optimization. Parametric Optimization Techniques and Reinforcement Learning”, Kluwer Academic Publishers.

Luciano, M. (2018). Aplicación Web de monitoreo para control de extensiones telefónicas [Tesis de grado, Universidad Tecnológica del Centro de Veracruz]. Repositorio Institucional de la UTCV. http://reini.utcv.edu.mx/handle/123456789/661

M. A. Al-Betar, M. A. Awadallah, A. L. Bolaji and B. O. Alijla, "β-Hill Climbing Algorithm for Sudoku Game," 2017 Palestinian International Conference on Information and Communication Technology (PICICT), 2017, pp. 84-88, doi: 10.1109/PICICT.2017.11.

Maquen, G. (2021). Metodología Integrativa en búsqueda de caminos con agente artificial para localización geográfica en instituciones educativas contextualizada en plano tridimensional. [Tesis de doctorado, Universidad Señor de Sipán]. Repositorio institucional USS. https://hdl.handle.net/20.500.12802/9033

Márquez Redhead, J. M. (2013). Implementación de un sistema de información que apoye el proceso diario de elaboración de cronogramas del personal de medicina física y rehabilitación de un hospital.

Martin, C., Urquía, A. & Rubio, M. (2021). Lenguajes de programación. UNED. https://books.google.es/books?id=qms4EAAAQBAJ&dq=lenguajes+de+programaci %C3%B3n&lr=&hl=es&source=gbs_navlinks_s

Mera Dávila, R. D., & Salinas Acosta, W. E. (2018). Aplicación móvil de algoritmos de rutas óptimas y su efecto en el desplazamiento de los conductores de vehículos en la ciudad de Trujillo.

Milian, J. (2019). Sistema web basado en algoritmo de ruta más corta para optimización de rutas en la empresa de servicios logísticos de courier Seminario Martínez Servicios Generales S.A.C. [Tesis de pregrado, Universidad Católica Santo Toribio de Mogrovejo]. Repositorio Institucional de la USAT. http://hdl.handle.net/20.500.12423/2237.

Osaba, E. (2020). Diseño e implementación de una meta-heurística multi-poblacional de optimización combinatoria enfocada a la resolución de problemas de asignación de rutas a vehículos.

Pérez, A. C. P., Ansola, E. S., & Rosete, A. (2021). A metaheuristic solution for the school bus routing problem with homogeneous fleet and bus stop selection. Ingeniería, 26(2), 233-253.

Pérez, J. E. (2019). Introducción a JavaScript. http://www.librosweb.es/javascript.

Quiroz Dávila, J. L., & Ramírez López, K. D. (2019). Sistema de búsqueda heurística para la localización de las principales oficinas de la Universidad Nacional Pedro Ruiz Gallo utilizando el algoritmo A Star (Título Profesional). Universidad Nacional Pedro Ruíz Gallo.

Rjoub, A. (2020). Courses timetabling based on hill climbing algorithm. 10(6), 6558–6573. https://doi.org/10.11591/ijece.v10i6.pp6558-6573

Russell, S. J., & Norvig, P. (2004). Inteligencia Artificial: un enfoque moderno (No. 04; Q335, R8y 2004.).

Soria, C., Da Silva, H., & Martin, A. E. (2017). Una propuesta para diversificar el campo de aplicación de un algoritmo Hill Climbing. Informe Científico Técnico UNPA, 9(1), 102-114. https://dialnet.unirioja.es/servlet/articulo?codigo=5919086

Soto, M. D. T., Soto, A. T., de la Torre Sifuentes, J. R., de León Sentí, E. E. P., & Rosas, F. J. L. Estudio Comparativo de Algoritmos de Búsqueda Local. CISCI 2008, 1-6.

Takeyas, B. L. (2007). Introducción a la inteligencia artificial. Instituto Tecnológico de Nuevo Laredo. Web del autor: http://www. itnuevolaredo. edu. mx/takeyas.

Velázquez Castro, J. (2019). Creación de Sitios Web. Users. https://books.google.es/books?hl=es&lr=&id=QyylDwAAQBAJ&oi=fnd&pg= PA1&dq=Crear+sitio+web:+con+css&ots=FeJiOTRCu&sig=j6ZsiO6brirVGJCXr31 ZvnD_cik#v=onepage&q=Crear%20sitio% 20web%3A%20con%20css&f=false

Welling, L., & Thomson, L. (2017). Desarrollo web con PHP y MYSQL (5ta ed.). ANAYA MULTIMEDIA. https://www.casadellibro.com/libro-desarrollo-web-con-php-y-mysql-5-ed/9788441536913/4911719

Zarrinmehr, A., Moloukzade, H. (2021). Aplicación de un Algoritmo de Subida de Colinas al Diseño de Rutas de Transporte Público en Redes Grid. Revista internacional de ingeniería de transporte, 9(2), 597-612. https://doi.org/10.22119/ijte.2021.285454.1569

Zelada Cabezudo, H. E. (2018). Algoritmo genético para la elaboración del horario de exámenes en universidades.