Evaluación de Representaciones Basadas en *Physarum polycephalum* para la Solución del Problema de Ruteo de Vehículos con Restricciones de Capacidad (CVRP)

Programa Matemáticas Aplicadas

Presentado por:

Laura Sofía Blanco Ríos

Director:

Juan Camilo Paz Roa

Pontificia Universidad Javeriana Cali

Facultad de Ingenieria y Ciencias

2025

Agradecimientos

Quisiera simplemente decir gracias a todas las personas que hicieron este proyecto posible, a Juan Camilo por ofrecer su mirada crítica y aconsejar en cada momento, a Andrés por el apoyo y la disposición para acompañarme en este proceso. A mis amigos que siempre estuvieron para escucharme, a mis padres por darme la oportunidad de existir en este plano y apoyarme en cada paso, a mis hermanos por ser luz en mi vida. A Edu por su apoyo incondicional.

Índice general

Índice de cuadros

Índice de figuras

Índice de algoritmos

Introducción

La necesidad de innovación a la hora de resolver problemas ha estado presente desde el inicio de nuestras civilizaciones, pues no solo se trata de encontrar una solución, sino de encontrar la mejor solución. Siendo lo anterior la razón de la existencia de la optimización como campo de estudio, las aproximaciones más comunes en este campo suelen ser desde las matemáticas y mediante algoritmos que eventualmente convergen a un óptimo matemático, sin embargo, estos métodos no pueden utilizarse para todos los casos y aunque se pueda, el costo computacional y el tiempo que se requiere para encontrar la solución aumenta según la complejidad del problema.

Los métodos exactos no lo son todo en la optimización, pues dentro de las características de la calidad de una solución no solo está la exactitud, también se encuentra el tiempo y los recursos que se requieren para solucionar el problema, es así como surgen los métodos heurísticos y metaheurísticos que se explicarán de forma más detallada en las siguientes páginas. Los profesionales del área requieren de alta creatividad para la creación de estos métodos y como en todo proceso creativo, necesitan de inspiración, la cual puede llegar de diversas fuentes, entre estas: la naturaleza. En la actualidad ya existen variados métodos inspirados en comportamientos naturales para la solución de diferentes tipos de problemas, sin embargo, lo importante de estos métodos no es su categoría, si no el alcance que tiene sus soluciones.

Este trabajo propone una evaluación de un método metaheurítico para la solución de un problema de ruteo de vehículos con restricciones de capacidad (CVRP), específicamente un método inspirado en *Physarum polycephalum*, un ser clasificado como *slim mould* (o moho mucilaginoso, en español) destacado por su capacidad de generar rutas óptimas para obtener alimento. Como era de esperarse, los descubrimientos acerca del comportamiento de este moho tuvieron gran impacto en la comunidad científica, en particular, en el área de la optimización, debido a esto, surgieron diversos algoritmos para la solución de problemas de optimización de redes inspirados en *Physarum polycephalum*.

Planteamiento del problema

2.1. Contextualización

El día a día de todos los seres humanos se ve atravesado por una cantidad enorme de problemas de optimización, problemas que la mayoría de las veces se resuelven desde la intuición y la experiencia. No obstante, cuando estos problemas crecen en dimensión se vuelven aún más complejos de tratar, a veces imposibles de resolver desde la intuición, aquí es dónde los métodos computacionales entran a hacerse cargo. Algunos de estos problemas suelen ser considerados problemas NP-hard, es decir, que no existen algoritmos exactos que solucionen estos problemas en tiempo polinomial. Sin embargo, el hecho de que no puedan encontrarse soluciones exactas, no quiere decir que no puedan solucionarse, por eso para este tipo de problemas es común encontrarse con métodos heurísticos y metaheurísticos para su solución.

A pesar de que estos métodos son muy importantes y útiles, al no ser exactos es muy importante evaluar su eficiencia, esto, tomando en cuenta tiempos, costo computacional, y, por supuesto, la cercanía a las soluciones óptimas (o las mejores encontradas), pues lo importante no es solo entregar una solución, si no una buena solución. El panorama académico está en la búsqueda constante de mejores soluciones, más rápidas y menos costosas computacionalmente. Derivado de la necesidad de innovar, es bastante común encontrarse con métodos que dicen ser innovadores, tomando inspiración de todo tipo de interacción en la naturaleza, lo cuál no es mala idea en sí misma, pues la creación de algoritmos heurísticos y metaheurísticos se alimenta de todo tipo de inspiración que ofrezca una buena representación (entiendase representación como la capacidad de ver similitudes y relacionar dos situaciones aparentemente paralelas) de la situación a modelar, las complicaciones llegan cuando, al analizar la esencia de los métodos, se detecta un método idéntico con métodos ya existentes, publicando trabajos académicos que prometen ser innovadores y no lo son.

Por esta razón, este trabajo plantea un estudio y evaluación metaheuríticas basadas en *Physarum polycephalum*, pues los métodos no solo deben proponerse, también deben evaluarse. Y con el fin de realizar una evaluación de este método en un contexto particular, se toma un problema clásico de optimización de redes, problema de ruteo de vehículos con restricciones de capacidad (CVRP).

2.2. Pregunta de investigación

¿Qué tan significativo es el aporte de la implementación de representaciones basadas en Physarum polycephalum para la solución a problemas de optimización CVRP?

2.3. Objetivos

2.3.1. Objetivo General

- Evaluar el aporte en eficiencias de un algoritmo de solución para CVRP con representaciones basadas en *Physarum polycephalum* con respecto a otro algoritmo sin estas.

2.3.2. Objetivos Específicos

1. Encontrar un algoritmo metaheurístico para la solución de CVRP que incorpore representación basada en *Physarum polycephalum* a un algoritmo ya conocido.
2. Comprender tanto la representación como el diseño de los algoritmos encontrados.
3. Implementar computacionalmente los algoritmos elegidos en el lenguaje Python.
4. Comparar la eficiencia de los dos algoritmos, a través de métricas habituales para la evaluación de eficiencia.

2.4. Alcance

Este trabajo tiene como objetivo principal documentar un proyecto de grado para la aspiración al título: Profesional en Matemáticas Aplicadas de la Pontificia Universidad Javeriana Cali. Este proyecto no tiene la ambición de generar conocimientos nuevos en el campo o el diseño de una nueva herramienta, por otro lado, tiene el objetivo de consolidar los conocimientos ya existentes y de aplicar estos conocimientos desde una perspectiva crítica, pues el conocimiento no solo debe entenderse y adoptarse, también se debe discernir y evaluar la veracidad del mismo.

Por esta razón, este proyecto no solo se centra en la aplicación de algoritmos de solución a problemas de optimización, si no también ofrece una mirada crítica a los métodos que prometen ser innovadores. Debido a lo cual, este proyecto demuestra el

entendimiento y la capacidad de aplicación de los conocimientos obtenidos a lo largo del pregrado, en paralelo a la capacidad de análisis y discenimiento del conocimiento.

2.5. Justificación

En un mundo intersecado con lo digital como en el que vivimos hoy en día, estamos cada vez más inmersos en el mar de conocimiento que como humanidad hemos construido durante milenios, aún contando con hechos tan lamentables como el incendio de la biblioteca de Alejandría, la humanidad ha estado en la capacidad, no solo de llegar al conocimiento, si no de conservarlo y transmitirlo, lo que a simple vista es un panorama maravilloso, sin embargo, el dimensionar que llevamos más de 6000 años guardando grandes cantidades de información de diferentes culturas y épocas, teniendo en cuenta también las épocas de oro en la generación de conocimiento, y que encima está acompañado con la era digital, con la que vino la capacidad de guardar más información en menos espacio, nos ubica en una época donde la democratización del conocimiento está relativamente viendo una luz, es decir, cada vez más personas "del común" tienen acceso a una cantidad exorbitante de información, información que al venir de diferentes fuentes, no siempre está verificada y es confiable, este efecto de infoxicación global produce desinformación generalizada, hay tanta información que ya no se sabe cuál es la verdad (Casas et al., 2022), (Pinto-Santos et al., 2018).

Sumado a esa infoxicación, en el mundo académico, particularizando al area de la optimización, se ha venido experimentando un fenómeno bastante curioso, que se trata con más profundidad en (Sörensen, 2015), el autor llama este problema como el problema de la metáfora, que nos habla de cómo las representaciones en los algoritmos metaheurísticos se volvieron tan abstractas que llegan a carecer de veracidad, debido a lo que se presentan trabajos prometiendo una solución completamente innovadora, cuando la realidad es que es una solución ya implementada con un nombre y una representación distinta.

El cruzarse con información en redes sociales acerca del potencial de *Physarum polycephalum* de solucionar problemas de optimización, despertó una curiosidad por ver cómo lo hacía, al encontrarse con tanta información y algoritmos que prometían estar basados en el comportamiento de *Physarum polycephalum* y resultaron ser un ya conocido algoritmo genético, generó la necesidad de poner a prueba los beneficios que puede traer la implementación de representaciones basadas en *Physarum polycephalum* para problemas de distribución de redes, en este caso, se acotó a problemas ruteo de vehículos con capacidad (CVRP). En este proyecto se pone a prueba el verdadero aporte de *Physarum polycephalum* en la solución de problemas CVRP, si vale la pena implementarlos o si con los algoritmos ya existentes puede llegarse a soluciones similares, a su vez, se comprueba la veracidad o validez de algún algoritmo ya existente.

2.6. Metodología

El esquema que se utilizó para llevar a cabo este proyecto, fue con el cumplimiento de objetivos, pues se diseñó un plan de acción para cumplir con cada uno de los objetivos específicos y, objetivo por objetivo, ir completando el objetivo general del proyecto.

2.6.1. Objetivo específico 1

Encontrar un algoritmo metaheurístico para la solución de CVRP que incorpore representación basada en Physarum polycephalum a un algoritmo ya conocido.

Para cumplir con este primer objetivo, se realizó una amplía revisión bibliográfica con respecto a diferentes algoritmos con representaciones basadas en *Physarum polycephalum*, otros algoritmos clásicos para la solución de CVRP. Esta revisión bibliográfica, se realizó tanto por recomendación a artículos específicos como de manera digital por las diferentes bases bibliográficas, como Scopus, ScienceDirect, Web of Science, IEEE, Dialnet, etc. Por últimos, se eligió un artículo en particular, que iba a permitirnos implementar y evaluar el algoritmo propuesto en el mismo.

2.6.2. Objetivo específico 2

Comprender tanto la representación como el diseño de los algoritmos encontrados.

Una vez elegido el trabajo en el que se iban a sentar las bases de este estudio, se leyó detenidamente, haciendo anotaciones e invertigando acerca de los temas que tal vez no eran tan conocidos antes. Se realiza un desglose tanto de las ecuaciones utilizadas, como de los algoritmos que se presentan, de esta manera, se estudio la descripción de la representación que se utiliza en cada algoritmo.

2.6.3. Objetivo específico 3

Implementar computacionalmente los algoritmos elegidos en el lenguaje Python.

Tomando como apoyo el desglose de los algortimos y las ecuaciones que se describe anteriormente, se comienza la implementación de los algoritmos en el leguaje de programación Python, lenguaje de libre acceso que se utilizó durante gran parte del pregrado. Esta implementación se dividió en dos partes, la parte de interpretación conceptual y la parte de la implementación algorítmica.

2.6.4. Objetivo específico 4

Comparar la eficiencia de los dos algoritmos, a través de métricas habituales para la evaluación de eficiencia.

Por último, se ejecutan los dos algoritmos y se mide su eficiencia con métricas como: tiempo de ejecución y cercanía a las soluciones óptimas.

Depués de realizar estas comparaciones, viene el análisis y las conclusiones que nos permiten darle respuesta a nuestra pregunta de invertigación.

Marco teórico

3.1. Estado del Arte

Como primera instancia del proyecto, siguiendo la metodología presentada anteriormente, se procede a realizar la revisión bibiográfica necesaria para enmarcar teóricamente nuestro proyecto y tomarlo como base para las posteriores interpretaciones y evaluaciones de la bibliografía.

3.1.1. *Physarum polycephalum*

Physarum polycephalum está clasificado como un moho mucilaginoso unicelular perteneciente al reino Protista, aunque por su aspecto y otras características en común sea confundido con un ser perteneciente al reino Fungi. Este moho tiene una estructura celular diferente durante su desarrollo como individuo, el primer estado por el que pasa, se conoce como estado de ameba, en este estado, se clasifica como un organismo unicelular uninucleíco, es decir, es un individuo constituido por una célula con un núcleo, esta escructura celular puede cambiar siempre y cuando las condiciones ambientales sean favorables, pasando al estado plasmoidal, que es el estado que realmente nos interesa en este trabajo, en este caso, *Physarum polycephalum* (*Physarum* por simplicidad) sigue siendo un organismo unicelular, sin embargo, ya no es uninucleíco, pues este organismo crece a partir de la mitosis de su núcleo, en otras palabras, *Physarum* sigue teniendo solo una célula, sin embargo, es capaz de copiar su material génetico y crecer físicamente de tan manera que tenga control de la totalidad de su cuerpo, como lo hacen algunos organismos pertenecientes al reino Fungi (Awad et al., 2023; Le Verge-Serandour & Alim, 2024).

Otra de sus similitudes con el reino Fungi está en su aspecto físico, pues su forma de ocupar el espacio es bastante parecida a la de las hifas fúngicas, que conforman el micelio, estas estructuras son una especie de tubos por los que se transportan señales físico-químicas, nutrientes y en caso de *Physarum* hasta la masa de su propio cuerpo, estos tubos están recubiertos por plasma, lo que le da el nombre del estado plasmodial, es un material viscoso, húmedo y amarillo, que le da a *Physarum* ese aspecto tan reconocible. Si tuviéramos una vista de lo que pasa dentro, nos encontraríamos con un fluído newtoniano incompresible, es decir, un fluído que no puede ser comprimido, que por lo tanto, tiene una densidad constante, este fluído se llama citosol, que es lo que se encuentra dentro del citoplasma de toda célula, sin embargo, no nos encontraríamos el fluído estático, pues recordemos que estos

tubos funcionan como una especie de autopista para el moho, por lo que se vería como este líquido es desplazado por movimientos hacia atrás y hacia adelante, un flujo pulsátil, casi como el flujo sanguíneo en el reino animal (Awad et al., 2023; Le Verge-Serandour & Alim, 2024). En adición a todo lo anterior, estos tubos están en la capacidad de hacerse más anchos, dependiendo de la cantidad de nutrientes que transporten, favoreciendo la conductividad, por otro lado, si hay algún tubo que no esté transportando nutrientes, se adelgazan hasta desaparecen, eventualmente, lo que nos muestra que este moho busca simplificar la obtención de alimento (Zhang et al., 2016).

Hasta ahora, vemos que *Physarum* es un organismo con comportamientos bastantes particulares, lo que eventualmente llevó a experimentar y estudiar su interacción tanto con factores internos, como con factores externos, es por eso que a comienzos de milenio se publicó un experimento bastante curioso, dónde se colocaba a Physarum en medio de un laberinto y este debía encontrar la única fuente cercana de alimento (una hojuela de avena) (Nakagaki et al., 2000), los resultados de este experimento demostraron que *Physarum* estaba en la capacidad de resolver un laberinto y de encontrar el camino más corto hacía el alimento, este experimento llevó a otros más, como a uno de los experimentos más famosos con *Physarum*, en el que se simula Tokyo y algunas de sus ciudades cercanas, en este caso, *Physarum* era la representación de Tokyo y las hojuelas de avena eran las ciudades cercanas, el objetivo de este experimento era demostrar la capacidad de *Physarum* de diseñar redes de transporte (Tero et al., 2010), lo sorprendente de los resultados de este experimento no fue solo que "Tokyo" fue capaz de conectar con "las demás ciudades", si no, que la red que diseñó *Physarum* era una red extremadamente similar a las existentes de vías férreas que fueron perfeccionadas por años, demostrando que *Physarum* no solo diseñaba buenas redes de transporte, pues diseña óptimas o cercanas a ellas, los esperimentos del tipo continuaron, como con la simulación de carreteras de Estados Unidos (Adamatzky, 2014), se puede encontrar un seguimiento más profundo de estos experimentos en Awad et al. (2023). En este mismo artículo se nos habla de varias otras aplicaciones de la inteligencia de *Physarum* como en computación biológica, en algoritmos de competencia, algoritmos de solución de optimización de grafos y optimización de algoritmos evolutivos híbridos, guardemos un nuestra memoria estas dos últimas aplicaciones. Al notar este gran potencial de este moho en diferentes campos del conocimiento, se comienza a tener reconocimiento en diferentes campos fuera del biológico así como lo ilustra **Fig. 3.1**.

Figura 3.1: Esquema gráfico aplicaciones *Physarum* Fuente: Le Verge-Serandour, Mathieu and Alim, Karen, 2024, http://dx.doi.org/10.1146/annurev-conmatphys-040821-115312

3.1.2. Problema de ruteo de vehículos (VRP)

Uno de los problemas de optimización en logística más encontrados en la industria es el problema de ruteo de vehículos (VRP, por sus siglas en inglés), en este problema hay uno o varios nodos origen/almacenes que buscan satisfacer la demanda de diferentes nodos/clientes a través de repartir sus productos con cierta cantidad de vehículos, como solución a esto, se espera encontrar la mejor distribución en la que los vehículos repartan los productos de tal manera que todos los clientes queden satisfechos, y se realice en el menor tiempo o la menor distancia posible (Sanchez L. A., 2022). Este problema se ha venido trabajando desde la década de los años 60, aproximadamente (Braekers et al., 2016), convirtiendose en uno de los problemas más estudiados en el campo, es exactamente por esta razón que el estudio de este no solo se quedó en el problema clásico, se han estudiando una amplía cantidad de variantes del problema, no obstante, debemos recordar que en el campo de la optimización lo importante no son los problemas, si no, las soluciones, en la medida en la que ha complejizado el problema también ha nacido la necesidad de encontrar mejores soluciones, soluciones más rápidas, más exactas, o en algunos casos, el simple hecho de encontrar una solución es una ganancia enorme, en cualquier caso, esto nos demuestra una amplia variedad de soluciones para los problemas VRP lo que nos permite hacer comparaciones un poco más confiables y discernir si una solución está verdaderamente aportando o no en el campo.

El VRP a pequeña escala es solucionable con métodos exactos, sin embargo,

entre mayor sea la dimensión del problema, mayor es la complejidad de estos métodos, en particular, este problema, a las escalas en las que se encuentra en la vida real, se considera un problema NP-hard, es decir, es necesaria la búsqueda de otros métodos de solución, por eso también se encontrarán métodos multicriterio, que toman varios objetivos al tiempo, dividiendo así el problema, por otro lado, también están los método heurísticos, que son métodos que priorizan el hallar una solución, al menos una solución preliminar, son computacionales y usualmente siguen lógicas intuitivas, aunque estos métodos no aseguran una solución óptima, ofrecen una opción en tiempos razonables, por último, tenemos los algoritmos metaheurísticos de solución, estos métodos son heurísticos, con lo diferencia de que tienen un transfondo conceptual usualmente más fuerte, pues son algoritmos que se inspiran en el funcionamiento o comportamiento de algún fenómeno, usualmente un fenómeno natural, estos son el tipo de soluciones que estamos evaluando en este estudio.

Como lo explicábamos anteriormente, el VRP tiene una gran diversidad en variaciones del problema junto con sus soluciones, como el algoritmo genético que se expone en (Ombuki-Berman & Hanshar, 2009), es una opción metaheúristica que se volvió un algoritmo bastante conocido en este campo y para la solución de este problema, de hecho, no solo se utiliza para la solución del VRP clásico, pues también se ha utilizado para problemas de VRP multi-depósito (MDVRP) como en (Surekha P, 2011), o también agregándole ventanas de tiempo (MDVRPTW) que se puede ver en (Osorio-Mora et al., 2021; Roa et al., 2020; Vidal et al., 2012), otra variación de VRP es agregrándole una restricción de capacidad a los vehículos (CVRP) que se ofrecen soluciones en (Cardozo, 2013; Ruiz, 2005; Vidal, 2022), hasta se llega a proponer problemas con restricción de capacidad y ventanas de tiempo (CVRPTW) como se expone en (Ruiz, 2005). Así como aquí tenemos un vista general de lo amplio que puede ser los problemas VRP, invito al lector a visitar (Braekers et al., 2016) si quiere una recopilación más amplia acerca de esto.

3.1.3. Representación basada en *Physarum polycephalum*

Después de apliado el panorama, es necesario comenzar a acotar el rango de búsqueda por cuestiones de alcance del proyecto, por lo que se comienza con una búsqueda más específica: Aplicaciones de *Physarum* en la optimización de grafos, particularmente en problemas VRP o hasta en el problema del agente viajero (TSP), que es un versión del VRP con un solo vehículo y una sola ruta (Sanchez L. A., 2022), caso que se encuentra en (Qian et al., 2013).

En esta búsqueda nos encontramos con gran una amplia gamma de aplicaciones, como en problemas de tráfico (Watanabe et al., 2011), por otro lado están los grandes complilados de información como se puede ver en (Adamatzky, 2019; Awad et al., 2023; Zhang et al., 2016), en las que nos dan una amplia visión de la cantidad de aplicaciones y algunos trabajos dónde se han aplicado, o en el caso de (Zhang et al., 2016) mostrando varios modelos del comportamiento de Physarum con su respectiva aplicación. Poco a poco y a partir de (Qian et al., 2013) se llega a (Chen et al., 2015) trabajo en el que comparten un autor, en este artículo derivado de una conferencia se nos muestra cómo, al igual que hicieron en el caso del TSP, se pretende optimizar

un algoritmo metaheurístico de colonia de hormigas (ACO) creando un híbrido con los conceptos de *Physarum*. Por lo que se decide tomar los algoritmos presentados en este artículo e implementarlos.

3.1.4. Optimización de Colonia de Hormigas (ACO)

Al decidir implementar algoritmos que involucran metaheurísticas con conceptos de colonia de hormigas, es pertinente realizar una búsqueda de antecedentes de estos algoritmos.

Los algoritmos metaheurísticos inspirados en la naturaleza suelen ser de dos tipos: evolutivos o de inteligencia de enjambre, los algoritmos genéticos caen en la primera categoría, estos toman inspiración de conceptos de la evolución de las especies, por otro lado, están los segundos, que toman como inspiración el comportamiendo de una colonia o un grupo de individuos que trabajan en conjunto (Roa, 2024), ese es el caso de ACO y, si lo pensamos, también es el caso de los basados en Physarum, pues aunque es un individuo unicelular, se divide de tal manera que trabaja de manera descentralizada, pues todas sus partes trabajan de manera independiente para un fin colectivo.

La investigación biológica sobre las hormigas descubrió como las colonias de hormigas encuentran los caminos óptimos desde el hormiguero, hasta la fuente de alimento y el camino de regreso. Para el comportamiento de las colonias de hormigas y el concepto más importante para la construcción de estos algoritmos son las feronomas que son una especie de marca química que dejan las hormigas a pasar por un lugar, esta feromonas son detectadas por las otras hormigas, por lo que terminan siendo un método de comunicación entre los individuos de la colonia, es de esta manera, como las hormigas comunican a sus compañeras los caminos que tomaron, lo interesante es que los caminos por los que se demoran menos, son caminos por los que pasan más veces, por lo que dejan más feromonas, como se ilustra en **Fig. 3.2**, lo que genera que poco a poco, las hormigas vayan seleccionando los caminos con más feromonas, terminando por utilizar los caminos más cortos (Algarín, 2010; Dorigo & Gambardella, 1997).

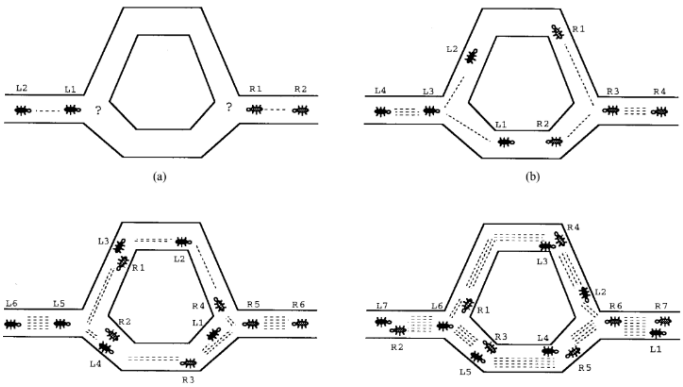


Figura 3.2: Ilustración aumento feronomas ACO Fuente: Dorigo, M. and Gambardella, L. M., 1997, http://dx.doi.org/10.1109/4235.585892

3.2. Artículo propuesto

El segundo objetivo en nuestro proyecto se trata de comprender el artículo elegido a implementar (Chen et al., 2015), para cumplir con este objetivo se decide dividir el artículo en cuatro partes, la primera tratándose de comprender los objetivos e intenciones que se tiene al realizar este trabajo; la segunda, el comprender el algoritmo ACO original al que se le implementarán cambios más adelante; posterior a eso, tenemos el comprender los conceptos clave del modelo de redes basado en Physarum, de dónde se obtendrán los nuevos cambios; por último, está la comprensión del algoritmo híbrido propuesto en el trabajo.

3.2.1. Intenciones del artículo

En la introducción del documento se nos explica como este trabajo es la continuación de un trabajo anterior de uno de los autores: (Qian et al., 2013), este artículo publicado dos años antes al artículo base de este proyecto, es la propuesta del ACO presentado en (Dorigo & Gambardella, 1997), que es un algoritmo construído para la solución de un TSP, híbridado con conceptos de Physarum. Al querer expandir el trabajo anterior, se tomó el CVRP, teniendo en cuenta la cercanía conceptual entre el TSP y el VRP, y vuelve a adaptarse el ya conocido algoritmo ACO al CVRP, agregándole algunos conceptos de Physarum, una de las razones que se expone en esa misma introducción es la tendencia del ACO de converger a óptimos locales, por lo que se espera que el hecho de agregarle una vista más general de toda la red con los conceptos del comportamiento de Physarum, permita la mayor cercanía a la solución óptima.

3.2.2. Algoritmo ACO

Como se explicó anteriormente, los autores utilizaron el ACO de (Dorigo & Gambardella, 1997), algoritmo del que se explican las bases teóricas en la sección anterior (estado del arte) y cuya representación en el problema CVRP será más detallada en la sección de la implementación, sin embargo, es importante entender, las reglas de transferencia, las reglas de actualización y las reglas de selección.

Regla de transferencia

La regla de transferencia nos habla del momento en el que la hormiga se encuentra en el punto anterior a elegir un camino, esta regla nos plantea una pizca de azar en aquella elección, pues se nos introduce un parámetro $q_0 \in [0, 1]$ que representa el umbal de elección, umbral que si $q_{rand} \in [0, 1]$, un valor que se calcula en cada iteración, sobrepasa a q_0, la decisión se toma por la probabilidad de cada

camino, de otra manera, se toma por la proporción entre la distancia y la cantidad de feromonas del mismo.

$$s = \begin{cases} argmax(\tau_{iu}\eta_{iu}^{\beta})_{u \in alk} & q_{rand} \leq q_0 \\ p & \text{de otra manera} \end{cases} \tag{3.1}$$

Es importante tener en cuenta que τ es la cantidad de feromonas de cada camino, en este caso, representa la cantidad de feronomas que hay del nodo i al nodo u, que recorre el conjunto alk donde se encuentran todos los nodos que pueden visitarse, por otro lado η es una variable que no habla del inverso de la distancia entre nodo y nodo.

Regla de actualización

Esta regla nos habla de la forma en la que se actualizan las feromonas al final de los recorridos, tengamos en cuenta que este es el cálculo de las feromonas de la siguiente iteración.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \rho\left(\frac{F}{Sbest(t)}\right) \tag{3.2}$$

Aquí podemos observar dos nuevos parámetros y una variable, por un lado tenemos F que representa la cantidad de feronomas que deja una hormiga al pasar por un camino y ρ representa el porcentaje de esas feronomas que desaparecen del tiempo t al tiempo $t+1$, y también tenemos $Sbest(t)$ que es la distancia mínima encontrada hasta ahora (tiempo t).

Regla de selección

En el caso en el que la regla de transferencia diga que la hormiga debe decidir según la probabilidad de elección de cada camino, el cálculo de la probabilidad de cada camino a ser elegido está definido por la siguiente equación:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}g_{ij}^{\gamma}(t)}{\sum_{u \in allowedk}\tau_{iu}^{\alpha}\eta_{iu}^{\beta}g_{ij}^{\gamma}(t)} & j \in alk \\ o & \text{de otra manera} \end{cases} \tag{3.3}$$

Aquí también aparece una variable nueva, una variable bastante importante, g, esta es una variable que nos habla de la distancia relativa entre cada nodo, es decir, qué tan cerca está un nodo del otro en comparación a lo cerca de está del nodo de origen, está definida por $g_{ij} = d_{i0} + d_{1j} - d_{ij}$, además podemos observar la aparición de α, β y γ, que son tres parámetros que nos indican el peso de cada variable en el cálculo de la probabilidad.

3.2.3. Algoritmo SMPN

La información más importante que se obtiene de esta parte son las ecuaciones que se utilizan para modelar el flujo y la conductividad de cada tubo en cada iteración. El cálculo del flujo de cada tiempo depende de la conductividad del tiempo anterior:

$$Q_{ij} = \frac{D_{ij}}{d_{ij}}(pr_i - pr_j) \tag{3.4}$$

Donde Q_{ij} representa el fujo del $nodo_i$ al $nodo_j$, D_{ij} es la conducticidad del tiempo anterior entre estos dos nodos y pr_i representa la presión de entrada que es la representación de la capacidad del vehículo y pr_j, el fluo de salida, es la demanda del $nodo_j$. Por otro lado, tenemos a la ecuación de la conductividad:

$$\frac{dD_{ij}}{dt} = \frac{|Q_{ij}|}{1 + |Q_{ij}|} - D_{ij} \tag{3.5}$$

3.2.4. Algoritmo SMPNACO

Para este punto el trabajo comienza a definir qué, de lo anterior se va a combinar y cómo. Aquí es dónde se define que la representación de la presión sería la demanda, por ejemplo. El artículo propone que la hibridación de estos algoritmos se de en la actualización de las feronomas, en decir, τ ya no solamente nos hablará de las feromonas actuales del tubo/camino, será un indicador de la viabilidad del mismo. Después de algunas simplificaciones al mezclar las ecuaciones nos queda:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \frac{F}{Sbest(t)} + \epsilon\left(\frac{Q_{ij}(t+1)}{\sum_{o=1}^{n} O_o}\right) \tag{3.6}$$

Donde O_o es la demanda del cliente, además, se le agrega la variable ϵ que es un indicador del peso que tiene el flujo y la conductividad en el valor de las feromonas y está definido por:

$$\epsilon = 1 - \frac{1}{1 + \lambda^{\frac{itmax}{2} - (t+1)}} \tag{3.7}$$

Aquí se agrega un último parámetro λ que nos ofrece información acerca de la cercanía a terminar la cantidad de iteraciones estipuladas.

3.3. Conceptos clave

En esta sección consolidaremos los conceptos importantes de este proyecto, así ofrecerle mayor claridad al lector, definiremos los conceptos: Optimización, *Physarum polycephalum*, Grafo, Nodo, Arista, Problema de ruteo de vehículos (VRP), Problema de ruteo de vehículos con restricción de capacidad (CVRP), Representación, Flujo, Conductividad, Feromonas, Optimización de colonia de hormigas (ACO), Modelo de optimización de red basado en *Physarum polycephalum* (SMPN), Algoritmo híbrido de colonia de hormigas y *Physarum* (SMPNACO).

Physarum polycephalum Moho mucilaginoso con comportamientos particulares utilizados como inspiración para diferentes aplicaciones en diversos campos. 15

Algoritmo híbrido de colonia de hormigas y *Physarum* (SMPNACO) Propuesta de solución híbrida al problema CVRP con representación basada en el comportamiento de *Physarum* y una colonia de hormigas. Este algoritmo es propuesto en (Chen et al., 2015). 15

Arista Las aristas son las conexiones entre los nodos en los grafos. Estas suelen representar los caminos, conexiones interpersonales, etc. 15

Conductividad La conductividad se refiere al grosor que toma cada tubo de Physarum al aumentar o disminuir el flujo. 15

Feromonas Las feromonas se refieren al rastro químico que deja cada hormiga al pasar por un camino, en este documento también puede ser tratado como un fluído por cuestiones de representación. 15

Flujo En este caso, cuando se hable de flujo, se refiere a la cantidad de líquido, o cualquier otra interpretación de fluído, que cada camino o tubo de Physarum puede transportar. 15

Grafo Estructura matemática constituida por nodos y aristas, en dónde las aristas conectan los nodos unos con otros. Esta estructura es bastante utilizada para modelar situaciones donde se necesita evidenciar relaciones o conexiones. 15

Modelo de optimización de red basado en *Physarum polycephalum* (SMPN) Este es un modelo del diseño/optimización de un grafo basado en el comportamiento de *Physarum*, es un modelo expuesto en (Chen et al., 2015; Zhang et al., 2016). 15

Nodo Puntos u objetos de un grafo. Esta figura puede representar entidades, personas, puntos en el espacio. 15

Optimización Disciplina que estudia la solución más eficiente a problemas cotidianos de gran escala. 15

Optimización de colonia de hormigas (ACO) Optimización metaheurística basada en el comportamiento de una colonia de hormigas. 15

Problema de ruteo de vehículos (VRP) Problema clásico de optimización de grafos, donde se tiene un centro de operaciones o nodo origen, y otros nodos que pueden representar clientes, en este caso, las aristas del grafo representan los caminos que se pueden tomar. 15

Problema de ruteo de vehículos con restricción de capacidad (CVRP) Esta es una variación al problema VRP, la diferencia de este con el anterior es que en este caso se le agregan restriciones de capacidad a los vehículos. 15

Representación Cuando nos referimos a representación a lo largo de este documento, estamos hablando del relacionamiento abstracto que se realiza al reflejar elementos de una situación con los elementos de otra, según los roles que cumplen estos elementos. 15

Implementación

En esta sección profundizaremos en la etapa de implementación de los dos algoritmos expuestos en (Chen et al., 2015), tanto el algoritmo ACO (colonia de hormigas) y el algoritmo híbrido SMPNACO (colonia de hormigas y *Physarum polycephalum*). Explicaremos las variables y la representación de estas, mostraremos los algoritmos y los pseudo-códigos que se construyeron como guía para su implementación en Python, hablaremos del ajuste de parámetros para ambos algoritmos, por último, se habla de la salida computacional de los algoritmos.

A lo largo de la lectura de esta sección es importante tener en cuenta que, aunque son algoritmos diferentes, son algoritmos que se diferencian por un cálculo adicional en el que se profundizó en la sección anterior, por lo tanto, las carácterísticas como variables o parámetros se mantendrán en ambos algoritmos, a menos de que se aclare lo contrario a través de un asterisco (*****) que indica que la característica solo aplica a SMPNACO.

4.1. Variables

4.1.1. Variables de entrada

Para las variables de entrada de ambos algoritmos tenemos:

- $k :=$ Número de hormigas/vehículos del modelo. $k \in \mathbb{Z}$.
- $cap :=$ Capacidad de carga de cada hormiga/vehículo. $cap \in \mathbb{Z}$
- $nd :=$ Arreglo coordenadas de los nodos/clientes.
- $dem :=$ Arreglo demanda de cada nodo/cliente.
- $itmax :=$ Cantidad máxima de iteraciones del algoritmo. $cap \in \mathbb{Z}$

4.1.2. Variables internas

Las variables del funcionamiento interno de ambos algoritmos son las siguientes (Algunas variables pertenecen solo al SMPNACO, será aclarado en ese caso):

- $p :=$ Diccionario de todos los posibles caminos/rutas.

- $p('Distance')$:= Distancia entre nodo i y nodo j. (Información propia de cada camino/ruta). $p('Distance') \in \mathbb{R}^+$.
- $p('Pheromones')$:= Cantidad de feromonas actuales del camino/ruta. (Información propia de cada camino/ruta). $p('Pheromones') \in \mathbb{R}$.
- $p('Conductivity')$:= Conductividad actual del camino/ruta. (Información propia de cada camino/ruta). $p('Conductivity') \in \mathbb{R}$. *****
- O := Suma de la demanda total. $O \in \mathbb{R}$
- $Sbest$:= Menor distancia recorrida. $Sbest \in \mathbb{R}$

Ahora las variables que se utilizan en cada iteración del algoritmo:

- esn := Si el nodo ya fue visitado y suplido. Arreglo binario: $\{f(x)\}$

$$f(x) = \left\{ \begin{array}{lll} o & no & visita \\ \\ 1 & si & visita \end{array} \right\}$$

- $capk$:= Capacidad de hormiga/vehículo en la itecaión.
- ks := Diccionario de caminos/rutas utilizados por cada hormiga.

En el paso de obtener una red de utilizan las siguientes variables:

- wa := Arreglo de rutas tomadas hormiga/vehículo.
- u := Nodos/clientes que la hormiga/vehículo puede visitar.
- $capkd$:= Capacidad del vehículo mientras recorre. $capkd \in \mathbb{R}$
- nod := Nodos visitados por vehículo k. $nod \in \mathbb{Z}$

Para elegir el siguiente nodo en la caminata tenemos:

- q := Probabilidad a utilizar. $q \in \mathbb{R}$
- pro := Arreglo con la probabilidad de cada nodo/cliente a ser elegido.

Finalmente, para el paso de validación y actualización se usa:

- sol := Distancia total de caminata. $sol \in \mathbb{R}$
- mf := Promedio del flujo de la red. $mf \in \mathbb{R}$.*****

4.1.3. Variables de salida

Las variables de salida para ambos algoritmos son:

- ksf := Diccionario con caminatas/rutas de la solución del algortimo.
- so := Arreglo con todas las soluciones del experimento.
- $solu$:= Distancia total solución. $solu \in \mathbb{R}$.

4.2. Funciones auxiliares

Los algoritmos, al ser una representación del mundo real, pueden ser complejos y engorrosos de llevar al mundo computacional a través de un código, sin embargo, existen herramientas que nos permiten organizar el código de una manera más ligera a la vista, una de estas herramientas es la creación de funciones auxiliares, estas no solo aportan una mejora estética del código, pues trabajar de esta manera también permite detectar posibles errores en el código de una manera más rápida y localizada, por otro lado, también brinda replicabilidad al código que al estar tan explícitamente codificado facilita su comprensión y utiliación en caso de que quiera replicarse. Para simplificación del código estos algoritmos se diseñaron algunas funciones auxiliares que se describen detalladamente a continuación:

- **Funciones que representan las ecuaciones:** Este es un grupo de funciones auxiliares que representan computacionalmente las ecuaciones vistas en el capítulo del marco teórico, estas funciones calculan las variables siguiendo cada ecuación.

- **paths:** Una función utilizada para inicializar la variable p, esta función genera un diccionario con todos los caminos posibles de la red.

- **allowed-cities:** Esta función nos devuelve la variable u, esta función busca los nodos que el vehículo k puede visitar a continuación, ya sea por capacidad, cercanía o tomando en cuenta si el nodo necesita la visita o no.

- **next-city:** La función next-city cumple con la tarea de elegir la siguiente ciudad que k visitará, teniendo en cuenta u y las ecuaciones de probabilidad que vimos en el capítulo del marco teórico a través de las ecuaciones 3.1 y 3.3.

- **walk:** Esta es la función que se encarga de generar los caminos de cada uno de los vehículos, esta función utiliza tanto allowed-cities como next-city para generar los caminos de cada k, tener en cuenta que esta función genera caminos que comienzan y terminan en el nodo 0 o nodo centro.

- **update-pheronomes:** Esta función se encarga de calcular, a través de funciones de ecuaciones 3.2, la cantidad actual de feronomas de cada uno de los caminos, de acuerdo a la solución generada anteriormente, a su vez, actualiza este dato en el diccionario p.

- *** update-pheronomes1:** Esta función también se encarga de calcular, a través de funciones de ecuaciones 3.6, la cantidad actual de feronomas de cada uno de los caminos, de acuerdo a la solución generada anteriormente, a su vez, actualiza este dato en el diccionario p. Adicionalmente, esta función incluye el cálculo del flujo y de la conductividad actual, según las ecuaciones ya vistas.

- **sol:** Con esta función se calcula la distancia total de la red, calculando la solución de cada iteración.

4.3. Algoritmos

Para el desarrollo de este trabajo, se implementaron dos los dos algoritmos presentados en Chen et al. (2015), el algoritmo ACO, que es un algoritmo clásico para la solución del problema CVRP basado en las colonias de hormigas; por otro lado, se implementó el algoritmo SMPNACO, que se describe como una algoritmo híbrido entre el ACO (colonia de hormigas) y el SMPN (Modelo de red basado en Physarum con una entrada y varias salidas), dando como resultado un algoritmo que amplía la búsqueda de soluciones posibles, reduciendo, según el artículo, la posibilidad de convergencia a un óptimo local.

4.3.1. Algoritmo ACO

A la relación entre los conceptos del funcionamiento teórico de una colonia de hormigas, y el funcionamiento de un problema CVRP le llamaremos: representación. En este caso, entendemos la representación de cada vehículo como una hormiga de la colonia, hormiga que al igual que los vehículos, tendrá una capacidad de carga; vemos el nodo/estación central como una fuente de alimento, y al tener varios nodos/clientes, podríamos interpretarlos como diferentes hormigueros (aunque no sea una representación exacta de la realidad); por último, podemos ver las aristas de nuestro grafo como los diferentes caminos que toman los hormigas, caminos que conectan la fuente de comida y los hormigueros entre sí, por los que, a su vez, se distribuyen feromonas. Para este caso, cada uno de los vehículos que toma un camino, deja feromonas tras de sí, marcando el camino y aumentando la probabilidad de otros vehículos de tomar ese camino. Se puede observar el pseudo-código de este algoritmo, donde se describe el paso a paso detallado, mirar **Algorithm 1**.

Tanto el generar redes de caminos, como el cálculo de la distancia total de la red y la actualización de las feronomas son tareas que se realizan con las funciones auxiliares: *walk*, *sol*, *update-pheromones*, respectivamente.

4.3.2. Algoritmo SMPNACO

La representación para este algoritmo es análoga, sin embargo, este algoritmo tiene una diferencia conceptual respecto al anterior. Su diferencia radica en la adición de dos conceptos: el flujo y la conectividad, conceptos que para nuestra representación abstracta, es como si los caminos se convirtieran en una especie de túneles dónde que tienen una conductividad, interpretada como la eficiencia (ya sea en distancia o tiempo) del camino respecto a los demás, y el flujo, la cantidad de vehículo que utilizan el camino respecto al uso de los demás caminos. Como podemos ver, ambos conceptos relativizan la eficiencia de los caminos, brindando un panorama global a un estudio local de la eficiencia de cada camino. Se puede observar el pseudo-código de este algoritmo, donde se describe el paso a paso detallado, mirar **Algorithm 2**.

Algorithm 1 Implementación ACO

```
Recibir k, nd, dem, cap, itmax
Inicializar p, O, Sbest
for la cantidad itmax do
    Inicializar ks
    while True do
        Inicializar esn
        for cada vehículo k do
            Inicializar capk, capkd, nod
            Generar redes de caminos
            Actualizar esn, capkd, nod
        end for
        Validar si se visitaron todos los nodos
        if esn es todo 1 then
            break
        end if
    end while
    Calcular distancia total de la red
    Agregar solución a so
    if si la solución es mejor que Sbest then
        Actualizar Sbest
    end if
    Actualizar Feromonas
end for
Actualizar solu con la mejor solución encontrada
Retornar ksf, so, solu
```

Algorithm 2 Implementación SMPNACO

```
Recibir k, nd, dem, cap, itmax
Inicializar p, O, Sbest
for la cantidad itmax do
    Inicializar ks
    while True do
        Inicializar esn
        for cada vehículo k do
            Inicializar capk, capkd, nod
            Generar redes de caminos
            Actualizar esn, capkd, nod
        end for
        Validar si se visitaron todos los nodos
        if esn es todo 1 then
            break
        end if
    end while
    Calcular distancia total de la red
    Agregar solución a so
    if si la solución es mejor que Sbest then
        Actualizar Sbest
    end if
    Actualizar Feromonas
    Calcular Flujo
    Actualizar Conductividad
end for
Actualizar solu con la mejor solución encontrada
Retornar ksf, so, solu
```

Al igual que en el caso anterior, tanto generar redes de caminos, como el cálculo de la distancia total de la red, y el cálculo del flujo de la red, actualización de conductividad y la actualización de las feronomas son tareas que se realizan con las funciones auxiliares: *walk*, *sol*, *update-pheromones1* , respectivamente.

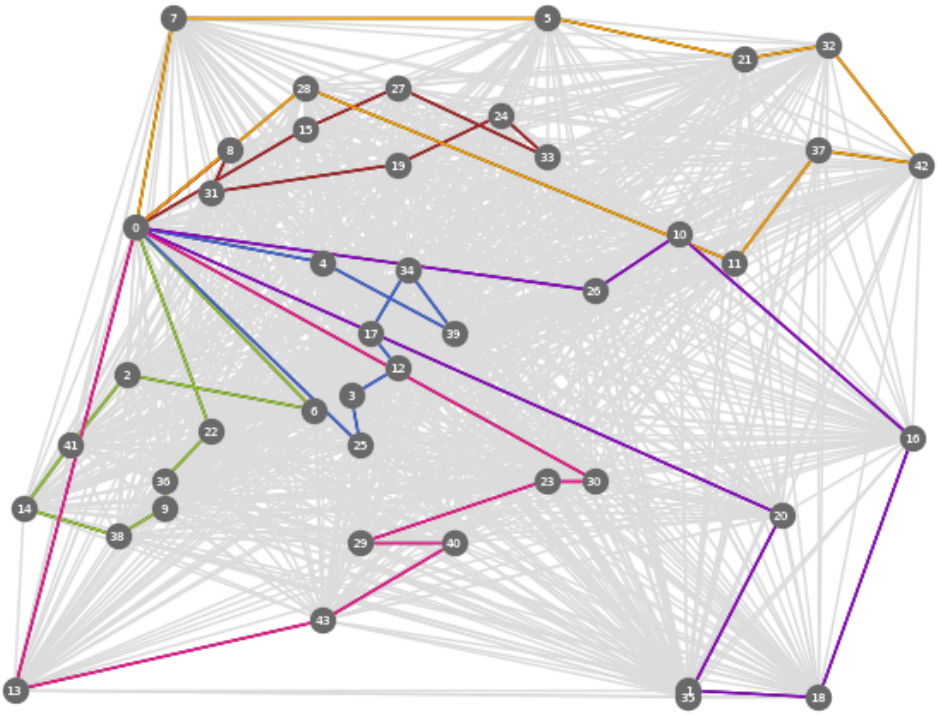
4.4. Parámetros

Los parámetros que se presentarán a continuación, son los parámetros que ya vimos en la parte del marco teórico, pues son parámentros que se utilizan para hacer el cálculo de diferentes variables.

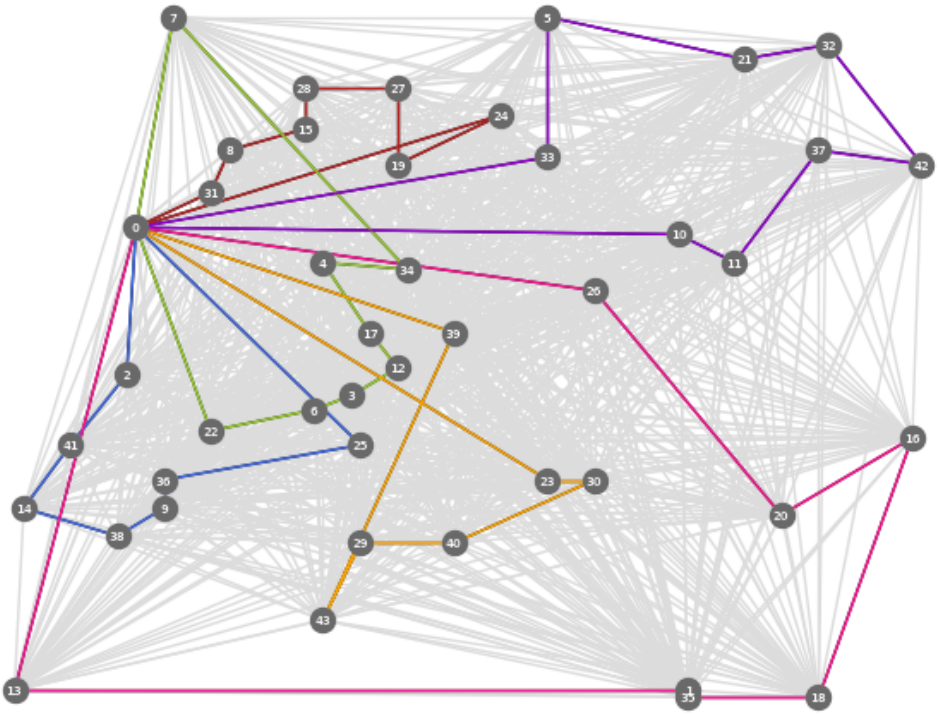
- $\alpha :=$ Este parámetro indica el peso de la variable τ en el cálculo de la probabilidad.
- $\beta :=$ Este parámetro indica el peso de la variable η en el cálculo de la probabilidad.
- $\gamma :=$ Este parámetro indica el peso de la variable g en el cálculo de la probabilidad.
- $q0 :=$ Es el *probability threshold* o el umbral de probabilidad. Este parámetro nos permite codificar la decisión de la probabilidad que se utilizará en cada iteración.
- $\rho :=$ Este parámetro mide el porcentaje de feromonas que se pierde entra iteración.
- $F :=$ Aquí se representa la cantidad de feronomas que deja una hormiga por cada camino que pasa.
- $\lambda :=$ Nos ofrece importancia acerca de la cercanía a terminar la cantidad de iteraciones estipuladas.

4.5. Salida computacional

Por último, es pertinente observar la salida computacional de nuestros algoritmos, que además de darnos la distancia total y las rutas que debe seguir cada vehículo, nos ofrece una vista gráfica de cómo sería la distribución y el recorrido que cada vehículo.



(a) Salida computacional ACO instancia A-n44-k5



(b) Salida computacional SMPNACO instancia A-n44-k5

Figura 4.1: Salida computacional algoritmos

Resultados

5.1. Experimentos

Para la parte de la evaluación de los algoritmos, se proponen diferentes experimentos con instancias de uso público para el problema CVRP (CVRPLIB-all-instances, 2024). Estas instancias nos ofrecen una descrpción general en su nombre, la primera letra en mayúscula indica la familia de instancias de la que viene, seguido a eso se encuentra una n y un número, esto indica la cantidad de nodos que tiene la instancia, por último, nos encontramos con una k que nos indica la cantidad de vehículos disponibles en la instancia; de esta maera podemos observar que si el nombre de uns instancia es, por ejemplo, A-n25-k4, se trata de ins instancia de la familia A que consta de 25 nodos y se debe solucionar con 4 vehículos.

Los experimentos con las instancias se ejecutaron en Python, en un equipo con 12,0 GB de RAM y procesador Intel(R) Core(TM) i3-7100U CPU @ 2.40GHz con un sistema operativo de 64 bits. Es importante recordar que estos experimentos se ejecutaron bajo estas condiciones teniendo en cuenta que el objetivo de este proyecto es realizar una evaluación de los dos algoritmos, no se plantea una competencia entre estos algoritmos y otras propuestas en la literatura, pues solo se pretende comparar el rendimiento del ACO con el rendimiento de SMPNACO presentados en (Chen et al., 2015) y de esta manera poner a prueba la relevancia de la implementación de representaciones del Physarum para el CRVP.

Con el objetivo de evidenciar el rendimiento de los algoritmos desde diferentes perspectivas se proponen tres experimentos:

- El primero y principal, experimentos con 15 instancias, en los que, por cada instancia, se ejecuta el algoritmo 20 veces, esto nos permite obtener una solución promedio y la mínima solución de cada una, pues recordemos que los dos algoritmos tienen factores aleatorios, aunque se calculen en las mismas condiciones, no siempre se encontrará la misma solución.
- El segundo, son experimentos de 5 instancias en los que los algoritmos se ejecutan 50 veces, esto para adquirir una noción del posible comportamiento de los algoritmos en una muestra más extensa para las instancias, se toman únicamente 5 instancias en este caso debido a las restricciones del alcance del proyecto.
- Por último, con el tercer experimento se quiso examinar más a detalle la exploración y explotación de los algoritmos, por lo que se diseñó un experimento para cada instancia en dónde se corría el código con un $itmax = 700$ de tal

manera que tuviera suficientes iteraciones para evidenciar las tendencias de los algoritmos.

Todos los experimentos que se presentarán a continuación se realizaron con los siguientes valores de parámetros: $\rho = 0,7$, $F = 20$, $\lambda = 1,05$, $\alpha = 1$, $\beta = 5$, $\gamma = 5$, $q_0 = 0,07$.

5.2. Presentación de Resultados

5.2.1. Primer experimento

Comenzamos por los resultados del primer experimento, que se pueden encontrar en el **Cuadro 5.1**. En el cuadro, puede observarse, no solamente los resultados de nuestros algoritmos ACO y SMPNACO, también se encuentran los resultados de un método adicional disponible en una librería de Python (vrpy), los cuáles nos ofrecen un punto de referencia de los métodos evaluados con otro método utilizado para resolver el problema. Esta librería utiliza el método de generación de columnas, un método heúristico conocido para solucionar problemas de programación lineal (Chabrier, 2020), en el manual de usuario de dicha librería se encuentra una explicación más detallada del cómo lo aplica (VRPy-Documentation, 2021).

Es importante aclarar que los resultados de los experimentos que se muestran, no son los únicos que se llevaron a cabo, sin embargo, sí son de los que se obtuvo un resultado para un análisis completo, recordemos que por cuestiones del alcance del proyecto, las condiciones tanto de computo como de tiempo eran limitadas, por lo que no se permitieron ejecuciones con un tiempo mayor a 750 minutos (12 horas aprox.). Se notó, que además del número de nodos (n), otro factor que aumentaba el tiempo computacional era el número de vehículos (k), por lo que a la medida que se ejecutaban los experimentos, se estableció que el número máximo de nodos (n) que se calcularían sería 50 y el número máximo de vehículos iba a ser 7.

Para la lectura clara de las tablas es importante aclarar que tanto el error como la diferencia entre los errores, son valores dados en porcentajes y el tiempo promedio de ejecución está dado en segundos, de encontrarse en (-) en los datos indica que el dato de esa columna no aplica para el caso del algoritmo. Los **Cuadro 5.1** y **Cuadro 5.2** se conforman por las siguientes columnas:

- itmax: Número máximo de iteraciones permitido para los algoritmos.
- S_{ave}: Valor promedio de la solución obtenida.
- S_{min}: Mejor solución alcanzada.
- t_{ave} **(s)**: Tiempo promedio de ejecución en segundos.
- Error: Porcentaje de desviación con respecto al óptimo conocido.
- Dif. error: Diferencia relativa de error entre ACO Y SMPNACO, dónde se toma como referencia el error de SMPNACO, lo que nos permite establecer

que si el valor es positivo SMPNACO tiene un error menor a ACO y viceversa.

- Ópt.: Mejor solución conocida.

Cuadro 5.1: Resultados Primer Experimento

	Alg.	itmax	S_{ave}	S_{min}	t_{ave}(s)	Error	Dif. error	Ópt.
	ACO		453.17	451.95	3.05	0.43 %	48.28 %	
P-n16-k8	SMPNACO	200	454.07	451.33	6.99	0.29 %		450
	vrpy		-	450	0.17	0 %	-	
	ACO		229.39	225.32	4.79	6.28 %	72.05 %	
P-n19-k2	SMPNACO	200	227.97	219.75	8.42	3.65 %		212
	vrpy		-	219	0.21	3.3 %	-	
	ACO		228.07	219.94	4.73	1.82 %	0.00 %	
P-n20-k2	SMPNACO	200	228.3	219.94	7.94	1.82 %		216
	vrpy		-	222	0.20	2.78 %	-	
	ACO		229.51	218.6	5.37	3.60 %	-6.74 %	
P-n21-k2	SMPNACO	200	232.42	219.14	8.86	3.86 %		211
	vrpy		-	224	0.12	6.16 %	-	
	ACO		234.85	227.1	6.56	5.18 %	-24.16 %	
P-n22-k2	SMPNACO	200	236.8	230.76	9.85	6.83 %		216
	vrpy		-	232	0.25	7.41 %	-	
	ACO		390.8	381.58	296.09	1.75 %	78.57 %	
E-n22-k4	SMPNACO	200	388.66	378.66	317.08	0.98 %		375
	vrpy		-	-	-	-	-	
	ACO		595.16	567.19	310.11	6.21 %	-37.34 %	
E-n30-k3	SMPNACO	200	600.14	586.93	418.99	9.91 %		534
	vrpy		-	-	-	-	-	
	ACO		888.8	816.6	11.81	4.16 %	-2.35 %	
A-n32-k5	SMPNACO	200	888.26	817.3	14.66	4.26 %		784
	vrpy		-	847	0.63	8.04 %	-	
	ACO		748.31	711.53	359.57	7.64 %	6.11 %	
A-n33-k5	SMPNACO	200	752.04	708.64	406.30	7.2 %		661
	vrpy		-	675	0.73	2.12 %	-	
	ACO		837.44	787.59	843.19	6.14 %	17.85 %	
A-n33-k6	SMPNACO	200	843.66	780.66	988.52	5.21 %		742
	vrpy		-	766	0.43	3.23 %	-	
	ACO		905.05	840.20	298.97	7.99 %	-35.25 %	
A-n34-k5	SMPNACO	200	897.89	874.03	342.5	12.34 %		778
	vrpy		-	789	0.53	1.41 %	-	
	ACO		944.78	911.2	235.59	14.04 %	17.10 %	
A-n36-k5	SMPNACO	300	948.28	894.85	277.36	11.99 %		799

Continúa en la siguiente página

	Alg.	itmax	S_{ave}	S_{min}	t_{ave}(s)	Error	Dif. error	Ópt.
	vrpy		-	806	0.52	0.88 %	-	
	ACO		550.78	523.9	298.44	14.39 %	9.16 %	
P-n40-k5	SMPNACO	300	554.84	514.29	298.09	12.79 %		456
	vrpy		-	508	0.47	11.41 %	-	
	ACO		1012.45	976.32	482.34	24.59 %	4.59 %	
A-n44-k7	SMPNACO	300	1020.67	970.15	528.11	23.72 %		784
	vrpy		-	847	0.63	8.04 %	-	
	ACO		1205.78	1154.22	698.34	17.2 %	8.64 %	
P-n50-k7	SMPNACO	300	1210.23	1145.18	729.45	16.49 %		983
	vrpy		-	1005	0.89	2.23 %	-	

El **Cuadro 5.1** nos permite observar cómo se desempeñan los algoritmos desde diferentes perspectivas. En primer lugar, veremos los resultados de nuestros dos algoritmos implementados (ACO y SMPNACO) respecto a los resultados del vrpy, podemos notar que para los casos de instancias de menor cantidad de nodos (n), aunque los errores relativos no se encuentran lejanos, se ve mayor cercanía al óptimo por parte de SMPNACO, no obstante, esto cambia cuando el número de nodos sobrepasa el 32, pues a patir de este n, se nota una dismunición significativa en el error de vrpy. Por otro lado, cabe mencionar la clara diferencia en tiempo de ejecución que tienen ACO y SMPNACO respecto a vrpy, diferencia que para este factor de efciciencia era predecible, pues se debe tener en cuenta que los algoritmos metaheurísticos pueden llegar a ser mucho más complejos computacionalmente, lo que no implica que se deban descartar.

Ahora centraremos nuestra atención en la comparación de la eficiencia de SMPNACO con respecto a ACO. Como primer punto, es importante notar que ACO tiene menores resultados en tiempo promedio de ejecución en todos los experimentos, lo cuál era de esperarse, pues tengamos en cuenta que SMPNACO debe realizar el cálculo aproximado de una ecuación diferencial cada vez que se actualizan las feronomas, lo que implica más esfuerzo computacional. Por otro lado, con los resultados en cuánto la cercanía de estos al óptimo, tenemos 8 instancias de 15 en las que SMPNACO tiene menor error respecto a ACO, es un poco más de la mitad de los casos, además del caso en el que el error es el mismo, sin embargo, no se ve alguna influencia clara de los valores de n o de k en estos resultados; por otro lado, observamos que el promedio de la diferencia relativa del error es de 8,64 %, esto, al ser positivo nos indica que en promedio SMPNACO ofrece soluciones más cercanas a la óptima, no obtante, la diferencia con el error de ACO no sobrepasa el 10 % en promedio.

5.2.2. Segundo experimento

Por otro lado, tenemos el experimento número dos, con el que se tiene como objetivo observar el comportamiento de los algoritmos al aumentar el número de

iteraciones de los experimentos, esto nos permite ver si la cantidad de repeticiones en los experimentos tiene un impacto en la eficiencia de los algoritmos. Para este caso, no se incluye más comparación externa que la mejor solución encontrada, debido a que el enfoque del análisis es la comparación entre ACO y SMPNACO.

	Alg.	itmax	S_{ave}	S_{min}	t_{ave}(s)	Error	Dif. error	Ópt.
P-n16-k8	ACO	200	453.99	451.3	2.42	0.28 %	33.33 %	450
	SMPNACO		454.36	451.9	6.87	0.42 %		
P-n19-k2	ACO	200	228.42	221.71	4.1	4.38 %	100.92 %	212
	SMPNACO		226.71	216.73	8	2.18 %		
P-n22-k2	ACO	200	235.95	222.33	5.02	2.85 %	58.33 %	216
	SMPNACO		236.63	219.95	9.36	1.80 %		
A-n32-k5	ACO	200	905.66	839.65	8.36	6.63 %	184.55 %	784
	SMPNACO		896.00	802.71	15.4	2.33 %		
A-n44-k7	ACO	300	1093.51	1046.09	123.90	11.73 %	19.57 %	937
	SMPNACO		1080.74	1028.92	143.68	9.81 %		

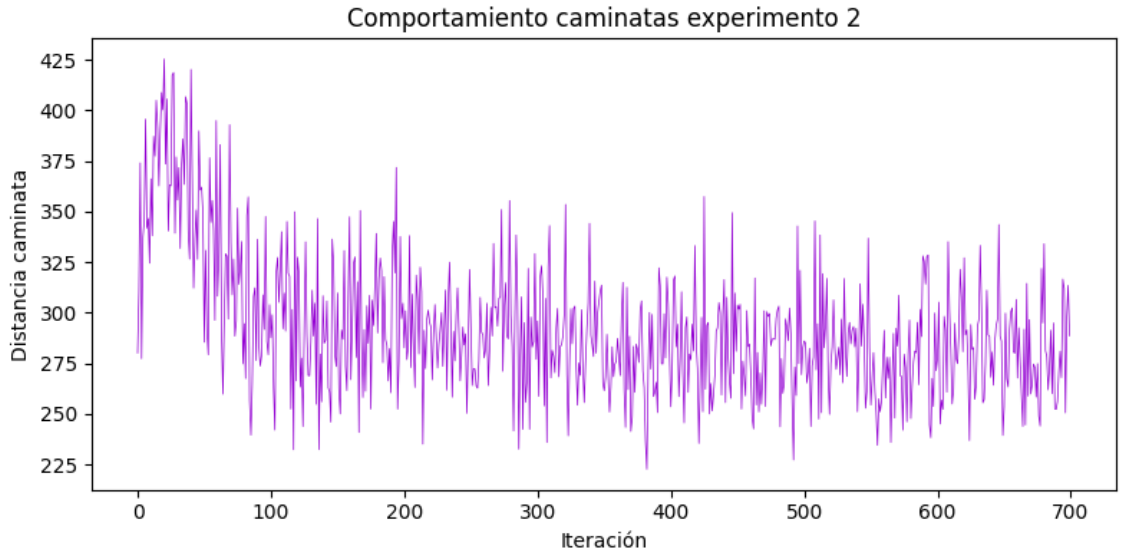
Cuadro 5.2: Resultados Segundo Experimento

Para el caso presentado en **Cuadro 5.2**, nos volvemos a encontrar con mayores tiempos computacionales, lo cuál vuelve a ser predecible. Sin embargo, sí se nota un mejor desempeño de SMPNACO, pues en este caso, en todas las instancias obtuvo un error menor al de ACO, se debe tener en cuenta que con estas mismas instancias no pasó lo mismo en el primer experimento. Por otra parte, también notamos que la diferencia relativa de los errores es mayor, pues en este caso el promedio de la misma es 79,34 %, que no solo es positivo y nos indica que en promedio el SMPNACO tiene resultados más cercanos al óptimo si no que sobrepasa el 50 %, lo que nos demuestra una mejora bastante clara.

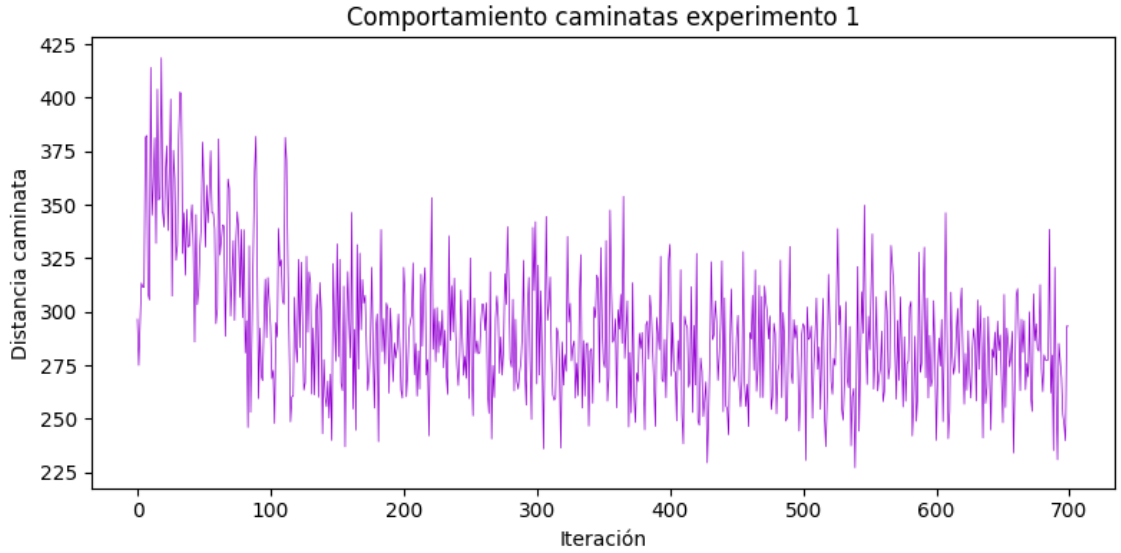
5.2.3. Tercer experimento

Para el caso del tercer experimento, en el que se buscaba observar el comportamiento de exploración y explotación se ambos algoritmos, se obtuvo resultados bastante similares entre todas las instancias. Debido a que lo que se quiere observar en este caso es el comportamiento interno de los algoritmos, se toma como muestra la salida computacional del experimento para el ACO y el SMPNACO de la instancia P-n22-k2.

En la **Fig. 5.1** se muestra como en ambos casos se evidencia una tendencia convergente, se ve claramente como ambos algoritmos comienzan con valores altos y eventualmente se van estabilizando. No obstante, se nota una curva más suavizada (ACO) respecto a la otra (SMPNACO), esto nos indica que la segunda experimenta cambios mucho más bruscos, es decir, las soluciones en cada iteración tienden a estar más alejadas de la inmediatamente anterior, cuando en el otro caso, las soluciones cambian paulatinamente. Esto podría explicarse con el hecho de que, con



(a) Comportamiento ACO con instancia P-n22-k2



(b) Comportamiento SMPNACO con instancia P-n22-k2

Figura 5.1: Comportamiento interno de los algoritmos

SMPNACO, se pretende una búsqueda más general, para evitar la caída precipitada en óptimos locales.

5.3. Conclusiones

El presente proyecto aplicado de fin de carrera evalúa la significancia de representaciones basadas en *Physarum polycephalum* en algoritmos para la solución de problemas CVRP, en términos de eficiencia computacional y calidad de las soluciones. Para evaluar esta significancia, se comparó una implementación del algoritmo de colonia de hormigas (ACO) con y sin las representaciones de *Physarum polycepha-*

lum. Es necesario aclarar que se escogió el CVRP como problema para realizar esta evaluación por la amplia disponibilidad de resultados en la literatura, resultados que llegan incluso a detallar soluciones óptimas, lo que permiten establecer un marco de comparación entre el ACO y SMPNACO. Fueron ejecutados tres experimentos, el primero evaluó la eficiencia y calidad de los algoritmos en instancias de prueba de la literatura (CVRPLIB-all-instances, 2024), el segundo comparó el desempeño de los algoritmos en respuesta a un mayor número de repeticiones, y finalmente se evaluó con caminatas aleatorias la convergencia y comportamiento de exploración y explotación del espacio solución.

Los resultados muestran que tanto el ACO como el SMPNACO no superan en eficiencia computacional y calidad las mejores soluciones reportadas para el CVRPLIB. Esto reafirma lo reportado en la literatura, en el que para un problema de estructura simple como el CVRP, los algoritmos basados en enjambres tienden a tener un menor rendimiento que otras estrategias metaheurísticas como las de búsqueda local o evolutivas. Lo anterior confirma que las implementaciones realizadas en el presente trabajo están en concordancia con la evidencia empírica de la literatura.

En cuanto a los resultados específicos de la implementación de ACO y SMPNACO, en el primer experimento vemos que en promedio el SMPNACO ofrece resultados 8.67 % más cercanos a la mejor solución registrada en comparación con el ACO, como se explica más a detalle en la sección anterior, no es una mejora mayor al 10 %. Sin embargo, se nota una diferencia del 79.34 % en el segundo experimento, donde se aumentan las repeticiones del experimento, de lo que se concluye que, a mayor cantidad de repeticiones, el SMPNACO muestra soluciones más acertadas que ACO, de lo que obtenemos una noción de la búsqueda en el campo de soluciones que tiene SMPNACO y cómo este amplía la búsqueda en dicho campo. Adicional a lo anterior, tenemos los resultados del tercer experimento, que refuerzan la hipótesis, pues el tener una curva menos suavizada indica cambios más bruscos, y, por lo tanto, una búsqueda menos localizada de la solución.

De acuerdo con lo anterior, se puede afirmar que el algoritmo SMPNACO con representación basada en el comportamiento de *Physarum polycephalum* presentado en (Chen et al., 2015), cumple con lo que indica aportar, pues se ofrece una solución y una búsqueda menos localizada. Este valor puede tener más o menos impacto dependiendo de lo que eso represente en costos, emisiones etc. Dada la operación de quien llegue a aplicar esto.

Bibliografía

Adamatzky, A. I. (2014). Route 20, Autobahn 7, and Slime Mold: Approximating the Longest Roads in USA and Germany with Slime Mold on 3-D Terrains. *IEEE Transactions on Cybernetics*, *44*(1), 126-136. https://doi.org/10.1109/TCYB.2013.2248359

Adamatzky, A. I. (2019). *Shortest Path Solvers. From Software to Wetware*. Springer Nature.

Algarín, C. A. R. (2010). Optimización por Colonia de Hormigas: Aplicaciones y Tendencias. *Revista Ingeniería Solidaría*, *6*(10), 83-89.

Awad, A., Pang, W., Lusseau, D., & Coghill, G. M. (2023). A Survey on Physarum Polycephalum Intelligent Foraging Behaviour and Bio-inspired Applications. *Artificial intelligence review*, *56*(1), 1-26. https://doi.org/10.1007/s10462-021-10112-1

Braekers, K., Ramaekers, K., & Van Nieuwenhuyse, I. (2016). The Vehicle Routing Problem: State of the Art Classification and Review. *Computers industrial engineering*, *99*, 300-313. https://doi.org/10.1016/j.cie.2015.12.007

Cardozo, J. P. O. (2013). *Solución al Problema de Ruteo de Vehículos con Capacidad Limitada "CVRP" a través de la Heurística de Barrido y la Implementación del Algoritmo Genético de Chu-Beasley* [Tesis doctoral, Universidad Tecnológica de Pereira].

Casas, A., Andrea, P., Sánchez, G., & Alejandro, J. (2022). Infoxicación e Impacto en la Sociedad. *Revista Tecnol.Investig.Academia TIA*, *10*, 66-84.

Chabrier, A. (2020). *Column Generation Techniques*. https://medium.com/@AlainChabrier/column-generation-techniques-6a414d723a64

Chen, Q., Qian, T., & Liu, K. (2015). A Logistic Distribution Routes Solving Strategy Based on the Physarum Network and Ant Colony Optimization Algorithm. *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium*

on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems.

CVRPLIB-all-instances. (2024). *CVRPLIB-all-instances*. http://vrp.galgos.inf.puc-rio.br/index.php/en/

Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation: A Publication of the IEEE Neural Networks Council*, *1*(1), 53-66. https://doi.org/10.1109/4235.585892

Le Verge-Serandour, M., & Alim, K. (2024). Physarum Polycephalum: Smart Network Adaptation. *Annual Review of Condensed Matter Physics*, *15*(1), 263-289. https://doi.org/10.1146/annurev-conmatphys-040821-115312

Nakagaki, T., Yamada, H., & Tóth, A. (2000). Maze-solving by an Amoeboid Organism. *Nature*, *407*(6803), 470. https://doi.org/10.1038/35035159

Ombuki-Berman, B., & Hanshar, F. T. (2009). Using Genetic Algorithms for Multi-depot Vehicle Routing. En *Bio-inspired Algorithms for the Vehicle Routing Problem* (pp. 77-99). Springer Berlin Heidelberg.

Osorio-Mora, A., Soto-Bustos, M., Gatica, G., Palominos, P., & Linfati, R. (2021). The Multi-depot Cumulative Vehicle Routing Problem with Mandatory Visit Times and Minimum Delayed Latency. *IEEE Access: Practical Innovations, Open Solutions*, *9*, 27210-27225. https://doi.org/10.1109/access.2021.3058242

Pinto-Santos, A. R., Díaz Carreño, J. A., & Santos-Pinto, Y. A. (2018). Infoxicación y Capacidad de Filtrado: Desafíos en el Desarrollo de Competencias Digitales. *Étic@ net*, *18*(1), 102-117. https://doi.org/10.30827/eticanet.v18i1.11884

Qian, T., Zhang, Z., Gao, C., Wu, Y., & Liu, Y. (2013). An Ant Colony System Based on the Physarum Network. En *Lecture Notes in Computer Science* (pp. 297-305). Springer Berlin Heidelberg.

Roa, J. C. P. (2024, febrero). Introducción a Métodos Metaheurísticos.

Roa, J. C. P., Escobar, J. W., & Moreno, C. A. M. (2020). An Online Real-time Matheuristic Algorithm for Dispatch and Relocation of Ambulances. *International Journal of Industrial Engineering Computations*, 443-468. https://doi.org/10.5267/j.ijiec.2019.11.003

Ruiz, S. F. Á. (2005). *Solución de Problemas de Ruteo de Vehículos Capacitados (CVRP) y con Ventanas de Tiempo (CVRPTW) Utilizando ILOG Dispatcher* [Tesis doctoral, Universidad de Los Andes].

Sanchez L. A., C. L., Tovar N. J. (2022). Problema de Ruteo de Vehículos. Importancia para la Logística y Métodos de Solución. *Documentos De Trabajo ECBTI*, *3*, 1.

Sörensen, K. (2015). Metaheuristics—The Metaphor Exposed. *International Transactions in Operational Research: A Journal of The International Federation of Operational Research Societies*, *22*(1), 3-18. https://doi.org/10.1111/itor.12001

Surekha P, D. S. S. (2011). Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms. *World Applied Programming, 1*(3), 118-131.

Tero, A., Takagi, S., Saigusa, T., Ito, K., Bebber, D. P., Fricker, M. D., Yumiki, K., Kobayashi, R., & Nakagaki, T. (2010). Rules for Biologically Inspired Adaptive Network Design. *Science (New York, N.Y.)*, *327*(5964), 439-442. https://doi.org/10.1126/science.1177894

Vidal, T. (2022). Hybrid Genetic Search for the CVRP: Open-source Implementation and SWAP* Neighborhood. *Computers operations research*, *140*(105643), 105643. https://doi.org/10.1016/j.cor.2021.105643

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A Hybrid Genetic Algorithm for Multidepot and Periodic Vehicle Routing Problems. *Operations Research*, *60*(3), 611-624. https://doi.org/10.1287/opre.1120.1048

VRPy-Documentation. (2021). *vrpy* [Manual de usuario/ documentación librería]. https://vrpy.readthedocs.io/en/latest/index.html

Watanabe, S., Tero, A., Takamatsu, A., & Nakagaki, T. (2011). Traffic Optimization in Railroad Networks Using an Algorithm Mimicking an Amoeba-like Organism, Physarum Plasmodium. *Bio Systems*, *105*(3), 225-232. https://doi.org/10.1016/j.biosystems.2011.05.001

Zhang, X., Gao, C., Deng, Y., & Zhang, Z. (2016). Slime Mould Inspired Applications on Graph-optimization Problems. En *Advances in Physarum Machines* (pp. 519-562). Springer International Publishing.