Variante Del Algoritmo De Optimización De Colonia De Hormigas (Aco) Incorporando Características Bilógicas De Las Hormigas Cultivadoras De Hongos.

Trabajo de Maestría

FRANCISCO JAVIER OBANDO VIDAL

Director: Ph. D. NÉSTOR MILCIADES DÍAZ MARIÑO
Co-Director: Mg. EMBER UBEIMAR MARTÍNEZ FLOR

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de investigación en Inteligencia Computacional (GICO)
Maestría en Computación
Popayán, Abril de 2020

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

1 INTRODUCCIÓN

En las últimas décadas, nuevas metaheurísticas que imitan comportamientos de sistemas naturales han sido propuestas con el fin de resolver diversos problemas de optimización. Debido a su dinámica y robustez, estas técnicas son capaces de encontrar la mejor solución manteniendo el equilibrio entre las partes involucradas [1].

Se han realizado diferentes estudios de investigación de los comportamientos de la naturaleza, que han sido fuente de inspiración de nuevas metaheurísticas que imitan ciertas características de esta, resolviendo problemas de optimización, este tipo de algoritmos son conocidos como Bio-inspirados que se caracterizan por estar basados en los comportamientos de la naturaleza, utilizados para resolver diversos problemas de optimización. Entre los algoritmos Bio-inspirados se destacan los de tipo inteligencia enjambre que estudian comportamiento colectivo para realizar tareas simples en espacio y tiempo, con altas respuestas de calidad en el entorno y adaptables, donde múltiples agentes interactúan resolviendo tareas específicas [2].

El algoritmo de optimización de colonia hormigas (En adelante ACO), es un algoritmo de inteligencia de enjambres, se fundamenta en el comportamiento natural de las hormigas, en sus características naturales estos insectos sociales son capaces de realizar tareas complejas debido a su interacción colaborativa y adaptiva, realizan búsquedas aleatoria de alimento con la habilidad de encontrar la ruta más corta entre el nido y el alimento, lo interesante es que esta labor la hacen a pesar de ser ciegas, por lo cual se apoyan de una sustancia química llamada feromona, la cual permite a las hormigas encontrar el camino de regreso al nido ayudando a las demás para que conozcan la ruta que han tomado; a medida que la cantidad de hormigas en pasar por una ruta aumenta hace que la feromona sea más fuerte siendo más atractiva por las hormigas obteniendo así el camino más corto, esta característica es fundamental para el planteamiento del algoritmo [3][4]. La figura 1 muestra proceso natural de la hormiga.

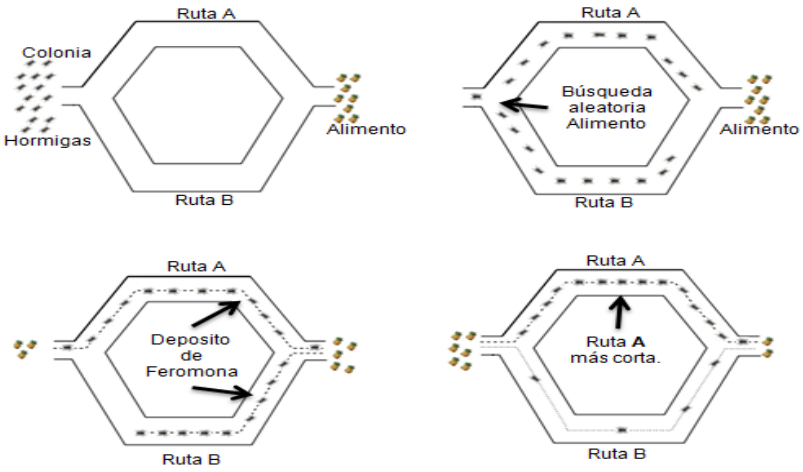

Figura 1. Comportamiento de la colonia de hormigas obteniendo la ruta más corta. Imagen adaptada de [3]

ACO se basa en una colonia de hormigas artificiales, es decir, simples agentes computacionales que trabajan cooperativamente construyendo una solución en cada iteración. Se destacan dos procesos, el primero es la información heurística que mide el camino más corto entre un nodo i hasta un nodo j y no varía durante el algoritmo. El segundo es la feromona artificial la cual mide la ruta deseable según la información que guarda la hormiga y cambia durante cada iteración del algoritmo dependiendo de la solución encontrada por las hormigas, además de una función de evaporación de la feromona artificial evitando estancamientos en óptimos locales [5][6].

La hormiga artificial busca soluciones validas, poseen una memoria de almacenamiento de información sobre las rutas que recorre la hormiga, aplicando reglas de transición en función del rastro de la feromona y de los valores heurísticos de la memoria privada de la hormiga.

Las hormigas reales y artificiales comparten ciertas similitudes como:

- Uso de colonias de individuos.
- Interactúan y colaboran para una tarea dada.
- Modifican su entorno través de comunicación basada en la feromona.
- Buscan el camino más corto desde el origen o hormiguero hasta el estado final o comida.
- Ambas aplican una estratégica de transición local estocástica para moverse al siguiente estado.

Pero estas características no permiten desarrollar algoritmos eficientes por sí solo, es por eso que la hormiga artificial del ACO pueden hacer uso de la información heurística y tiene memoria de almacenamiento, a diferencia de la hormiga natural la artificial coloca la feromona después de generar la solución y esta se deposita según la calidad de la solución encontrada, la evaporización de la feromona es un mecanismo para evitar estancamientos en óptimos locales permitiendo olvidar la historia de los caminos y así direccionar la búsqueda hacia nuevas regiones del espacio [4].

ACO es eficaz en problemas de optimización combinatoria de redes y rutas, como por ejemplo el problema del vendedor viajero (Travelling Salesman Problem) y en predicción de estructuras de proteínas. Una de las desventajas de ACO es que a pesar de encontrar buenas soluciones su tiempo de convergencia es muy lento, en ocasiones tienden a converger de forma prematura y puede tener estancamientos, y estos problemas pueden ser frecuentes dependiendo del tamaño del problema [7].

ACO no usa todas las características naturales de las hormigas, en diferentes trabajos en el campo biológico se describen diferentes castas de hormigas como por ejemplo las arrieras conocidas también como Hormigas *Atta* que se destacan por ser hormigas cultivadoras de un hongo, con el cual trabajan de forma simbiótica, donde el hongo sirve de alimento para las hormigas y estas a su vez protegen al hongo de enfermedades, con un sistema de identificación que le indica a las hormigas si el alimento que consiguen para alimentar el hongo es adecuado o dañino para este [8]. En el presente trabajo busca proponer una variante del ACO que al incluir una o más características naturales de las

hormigas *Atta* mejore al ACO tradicional en un problema de optimización combinatoria como el TSP.

A continuación, se presentan los objetivos, tanto general, como específicos, los cuales se plantearon en la propuesta de tesis de investigación que dio origen a este trabajo.

- **Objetivo general:** Proponer una variante del algoritmo de optimización de colonia de hormigas (ACO) que incorpore una o más característica del comportamiento natural de las hormigas *Atta* o arrieras
    - **Objetivo específico 1:** Caracterizar trabajos y/o investigaciones del algoritmo optimización de colonia ACO identificando diferentes propiedades usadas en las implementaciones
    - **Objetivo específico 2:** Diseñar el algoritmo de optimización de colonia de hormigas incorporando un comportamiento natural de las hormigas *Atta*.
    - **Objetivo específico 3:** Evaluar el desempeño y rendimiento de la variante del algoritmo propuesto contra el ACO tradicional en un problema de optimización.

Para alcanzar los objetivos propuestos para el proyecto, se siguió la metodología de patrón investigativo iterativo propuesto en [9], donde se implementaron las etapas de este patrón en 3 iteraciones. Se contemplan las siguientes etapas:

a) **Observación**: En esta etapa se realizaron tareas como definición del problema, revisión de las características de las hormigas *Atta* apoyado por una revisión sistemática, implementaciones de ACO para ser contrastadas con la propuesta, revisión de resultados preliminares y problemas que se solucionan con ACO.

b) **Identificación:** esta etapa se realizó la identificación de los puntos para adaptación de las características naturales de la tribu de hormigas *Atta*, diseño de algoritmo y pruebas de concepto en un problema de complejidad combinatoria en este caso TSP.

c) **Desarrollo:** En esta etapa se verifico el diseño del algoritmo, se implementó una solución en donde se propondrá un modelo y verificación de parámetros colocando a prueba el diseño del nuevo algoritmo.

d) **Pruebas:** Esta etapa incluye pruebas de la revisión sistemática en sus criterios de calidad, pruebas de implementación del algoritmo propuesto con diferentes instancias de TSP, verificación y ajustes de parámetros del algoritmo en la siguiendo lo definido en [10], análisis de resultados y presentación de resultados.

El resto del libro se estructura de la manera que a continuación se describe.

El Capítulo Dos presenta el marco conceptual y los avances existentes en las áreas que dan sustento a la propuesta de investigación con apoyo de un mapeo sistemático referente a ACO en solución al problema del viajero problemas (TSP) y las hormigas *Atta*, que dan paso para resolver el problema de investigación. En el Capítulo Tres, se abordan el diseño

propuesto donde se indican las características de las hormigas *Atta* seleccionadas que aportan a las mejoras de ACO tradicional. El Capítulo Cuatro, se dedica a describir los resultados de investigación obtenidos, desde ACO y la perspectiva del algoritmo propuesto frente al problema del vendedor viajero. El capítulo final, sintetiza las conclusiones del trabajo de investigación, y futuros trabajos en el área.

2 MARCO CONCEPTUAL

2.1. CONTEXTO GENERAL

2.1.1. HORMIGAS *ATTA*

Los insectos como las termitas, escarabajo y hormigas son agentes sociales que han evolucionado con el tiempo, en [8] se explica la evolución de estos al realizar cultivos, los cuales tienen ciertas similitudes en una evolución convergente, que se caracteriza por tener monocultivos y realizar un buen manejo de control de enfermedades generando diferentes estrategias, como aislamiento, monitoreo y administración de los cultivos evitando parásitos que infecten el cultivo.

El algoritmo de optimización de colonia de hormigas utiliza algunas características naturales como el uso de la feromona, pero existen diferentes estudios de las hormigas, como por ejemplo, en [11] se describen a las hormigas cultivadoras de hongos como insectos sociales que han desarrollado sistemas de agricultura y de salud pública similares a la sociedad humana. Estas hormigas cortadoras pertenecen a la tribu de las *Atta* que tienen como característica principal una relación cooperativa con un hongo el cual han domesticado y cultivado para alimentar a sus crías.

En [12] se describe a las hormigas *Atta* como especialistas en optimizar energía y proteger a la colonia de contaminaciones, además presentan diferentes castas o tipo de hormigas por un lado la casta obrera hormigas pequeñas dedicadas al cultivo y cuidado del hongo, esta casta no sale al exterior, ya que pueden llegar a contaminar el cultivo arriesgando su fuente de alimentación, además asisten y alimentan a las crías .

La casta recolectora, son hormigas medianas encargadas de explorar por fuera del nido, buscando y recolectando el alimento que servirá para el cultivo del hongo, por otro lado, se identifica la casta seleccionadora la cual se encarga del sistema de salud verificando si las hormigas recolectoras están infectadas y no permitirle acceso a la cámara de cultivo del hongo. Una última casta de mayor tamaño y robustas se encargan de la defensa del nido.

En [13] indican que las *Atta* recolectan distintos materiales como hojas, restos vegetales, heces de insectos sobre los cuales cultivan el hongo, el cual es altamente vulnerable a patógenos. Por lo cual las *Atta* han desarrollado un sistema sanitario para combatir y prevenir patógenos que pueden afectar el hongo, creando una cooperación con bacterias filamentosas (*Actinomycetes*) que son cultivadas como fuentes de antibióticos. Las *Atta* han evolucionado con respecto a las hormigas tradicionales logrando un gran tamaño y ser más resistentes, poseen una gran capacidad de adaptación. Cultivan el hongo en jardines subterráneos produciendo el alimento necesario donde se identifican tres transiciones principales la primera cultivar una gran variedad de hongos, la segunda cultivar hongos de calidad con una co-evolución bilateral y tercero el surgimiento de cultivadoras herbívoras.

2.1.2. ALGORITMOS BASADOS EN COLONIA DE HORMIGAS

La computación inspirada en la naturaleza o algoritmos Bio-inspirados descritos en [1] y [2] toman como base la observación de la naturaleza, esta es la fuente de inspiración para el desarrollo de nuevas metaheurísticas tomando algunas sus características. Al observar el comportamiento de la naturaleza se identifican dos características principales, las detectoras como aquellos que reciben información del entorno y las efectoras que hacen cambios de estado y realizan cambios en el entorno, estas son guiadas por reglas de comportamiento para decidir el actuar según la información recolectada, además de ser capaces de aprender y adaptarse al entorno.

En [14], [15], [16] se describe la inteligencia de enjambres ("Swarm Intelligence – SI"), como el comportamiento colectivo compuesto de agentes simples e individuales que interactúan entre sí y con el entorno, descentralizados y auto-organizados que siguen reglas locales de comportamiento, donde las interacciones entre los agentes producen una inteligencia colectiva proporcionando soluciones eficientes a problemas complejos. Las principales metaheurísticas que surgieron de la observación de la inteligencia de enjambres son optimización de colonia de abejas ("Artifical Bee Colony Optimization - ABC"), optimización por enjambre de partículas ("Particle Swarm Optimization - PSO"), optimización de colonia de hormigas ("Ant Colony Optimization - ACO"), búsqueda cuckoo ("Cuckoo Search - CS").

En [3] y [4] se presenta el algoritmo optimización de colonia de hormigas (ACO), como una metaheurística inspirada en el comportamiento real de estos insectos. Está basada en el comportamiento natural de las hormigas, las cuales son insectos sociales, capaces de realizar tareas complejas debido a su interacción colaborativa y adaptiva.

Las hormigas realizan búsquedas aleatoria de alimento con la habilidad de encontrar la ruta más corta entre el nido y el alimento, esta tarea la realizan a pesar de ser casi ciegas, por lo cual se apoyan de una sustancia química llamada feromona, la cual permite a la hormiga encontrar el camino de regreso al nido, que ayuda a otras hormigas para que conozcan la ruta que han tomado; a medida que la cantidad de hormigas en pasar por una ruta aumenta hace que la feromona sea más fuerte siendo más atractiva por las hormigas obteniendo así el camino más corto, esta característica es fundamental para el planteamiento del algoritmo.

ACO se basa en una colonia de hormigas artificiales, es decir, simples agentes computacionales que trabajan cooperativamente y se comunican a través de senderos artificiales de feromonas. Este algoritmo en cada iteración construye la solución a un problema, se destacan dos procesos:

- **Información heurística**, No varía durante el algoritmo y mide que camino es más corto entre en un nodo *i* hasta un *j*.
- **Feromona artificial,** mide la ruta deseable según la información que guarda la hormiga y cambia durante cada iteración del algoritmo, además cuenta con una función de evaporación de feromona artificial evitando estancamientos en óptimos locales.

En [17] existen ciertas similitudes y diferencias entre las hormigas reales y artificiales, la tabla 1 muestra estas diferencias.

Tabla 1. Similitudes y diferencias Hormiga reales y artificiales

Similitudes	Diferencias
Usan una colonia de individuos y colaboran para solucionar una tarea	La hormiga artificial tiene memoria que almacena el camino
Buscan el camino más corto desde el hormiguero (Estado inicial) hasta la comida(Estado Final)	La hormiga artificial deposita la feromona solo después de generar la solución.
Modifican su entorno por intermedio de una comunicación basada en feromona	Hacen uso de información heurística no se basa solo en rastros de feromona.
Usan estrategias de transición local estocástica para moverse.	Evaporación de la feromona, lo cual permite que la hormiga artificial no se estanque en óptimo locales, al hacer esto se puede olvidar lentamente la historia pasada y redireccionar la búsqueda.

La hormiga artificial puede llegar a construir soluciones poco prometedoras que pueden llegar a ser penalizadas dependiendo de la solución, La hormiga artificial busca soluciones válidas junto con una memoria que almacena información sobre el camino visitado hasta el momento, puede usarse para construir la solución y reconstruir el camino que ha seguido la hormiga. Cada movimiento de la hormiga artificial se lleva a cabo aplicando reglas de transición en función de los rastros de la feromona disponible localmente por cada hormiga y de las restricciones del problema.

En [4] Durante el proceso de construcción una hormiga se mueve de un punto a otro y va actualizando el rastro de la feromona paso a paso, una vez la hormiga ha construido la solución puede reconstruir el camino recorrido y actualizar los rastros de la feromona. El proceso de construcción acaba cuando satisface alguna condición de parada.

El pseudocódigo del algoritmo genérico de ACO se muestra en la figura2. La primera parte es la inicialización de los parámetros que tiene el algoritmo, se debe fijar el rastro inicial de la feromona, el número de hormigas de la colonia, el peso de la información heurística y memorística de las reglas de probabilidad

Uno de los procedimientos principales del ACO es la *creación de hormigas*, se encarga de generar y hacer funcionar a las hormigas artificiales, ejecuta tareas de evaporación de la feromona artificial y controla las acciones de paradas. Otro procedimiento es *actualizar memoria* en la hormiga, se encarga de especificar el estado inicial de la hormiga y almacena la memoria de esta, además calcula la probabilidad y aplicar políticas de decisión tomando el estado actual de la hormiga [20][6].

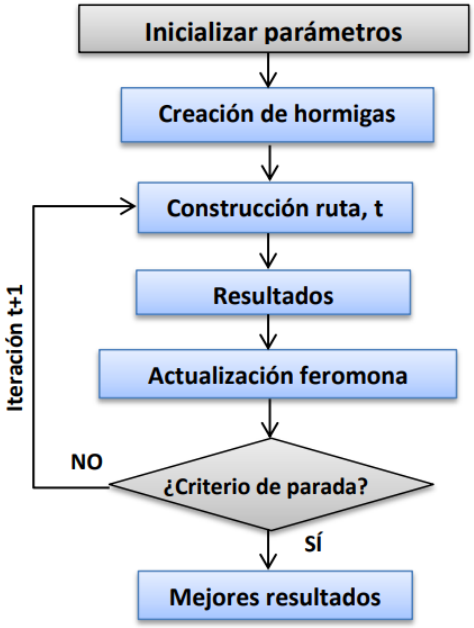

Figura 2. Algoritmo genérico ACO[18] [19].

ACO se especializa en los problemas de rutas como el problema del viajero, redes, grafos y predicción de estructuras de proteínas. Existen diferentes implementaciones de ACO por ejemplo en [20] presenta una descripción general de la optimización de ACO usando computación paralela, donde se incluye un método conceptual de una propuesta taxonómica útil para comprender y organizar trabajos relacionados, además de futuras investigaciones con ACO, además, se presenta un análisis de las diferentes implementaciones de ACO con computación paralela.

En [21] se usa una estrategia de distribución que consiste en configurar el parámetro heurístico de tal forma que sea adaptativo, esto se realiza para evitar una convergencia prematura, dado el caso si el parámetro es muy grande o que se estanque en un óptimo local si el parámetro es muy pequeño. Esta configuración permite mejor rendimiento del algoritmo ACO, además utilizan la búsqueda local perfeccionando las hormigas de la colonia, evidenciando mejores resultados que los obtenidos con un ACO tradicional.

En [22] se realiza un híbrido de ACO utilizando una red neuronal, diseñando nuevos conjuntos de reglas para actualizar la feromona y la medición de la información heurística. Por otro lado, implementa una estrategia de guía, que ubica a las hormigas en rutas correctas mientras construyen el grafo. Con esta estrategia intensifican la capacidad de búsqueda local, proporcionando un equilibrio efectivo entre la exploración y la explotación de la información.

2.1.3. DESCRIPCIÓN DEL PROBLEMA TSP

En el contexto revisado se identifica que uno de los problemas clásicos a resolver es el problema del vendedor viajero (Travelling Salesman Problem en inglés, en adelante TSP), donde Marco Dorigo en [23], implementa por primera vez ACO para dar una solución a este problema.

TSP es un problema de tipo NP-hard, el cual consiste en determinar el recorrido con menor costo posible para que un vendedor visite *n* ciudades pasando por todas ellas una vez, iniciando y finalizando por la misma ciudad. Este tipo de recorrido se conoce en teoría de grafos como circuito hamiltoniano.

Un problema de TSP se puede representar mediante un grafo G (N; A), donde *N* es el conjunto de ciudades, *A* los arcos o aristas, donde se realiza un cálculo de la distancia entre ciudades que se representa como $D = d_{ij}$. Una ruta es una sucesión de arcos (a_1, a_2, …, a_n, a_1) donde el nodo final de cada arco coincide con el nodo inicial, sin utilizar el mismo arco más de una vez, pasando por todos los arcos.

2.2. REVISIÓN SISTEMATICA

Para la revisión de los antecedentes de trabajos relacionados se propuso una revisión sistemática (en adelante RSL), para identificar las diferentes variantes de ACO aplicadas al problema del agente viajero, y a partir de esta plantear una nueva solución ACO con hormigas artificiales con capacidad de distinguir buenas soluciones que ayuden a la colonia en el proceso de cultivar el hongo para intensificar la búsqueda esperando encontrar la mejor solución en el menor tiempo.

Con la RSL se exploró el estado del arte del algoritmo de optimización de colonia de hormigas, identificando si se incluyen comportamientos de las hormigas naturales, en específico de la hormiga arriera o cultivadora de hongos (subespecie *Atta*). Se procuró analizar las diferentes variantes de ACO más citados dentro del estado del arte tomando como base la solución al problema del agente viajero o TSP.

La metodología planteada en Kitchenham [24], es usada para la revisión, permitió definir los objetivos de la RSL aportando una guía en el, seguidamente se establecieron las preguntas de investigación para enmarcar la RSL, después se realizó la búsqueda de documentos en bases de datos digitales como Scopus, ScienceDirect, ACM digital. Posteriormente se aplicaron criterios de exclusión e inclusión y finalmente se realiza la comparación de diferentes estudios, se eliminan los repetidos y se exponen los resultados del RSL, como un primer proceso se realizó un protocolo de revisión sistemática que se presenta en el anexo 1.

2.2.1. PREGUNTAS DE INVESTIGACIÓN

Para dar solución al objetivo se plantean las preguntas de investigación de la Tabla 2. de esa forma seleccionar y agrupar los diferentes estudios acerca del ACO aplicado a TSP, además para identificar la utilización previa del comportamiento natural de las hormigas cultivadoras de hongos.

Tabla 2. Preguntas para la Revisión sistemática

Identificador	Pregunta para la RSL
P1	¿Cuáles algoritmos de optimización de colonia de hormigas permite resolver el problema del vendedor viajero (TSP)?
P2.	¿Qué tipos de algoritmos de optimización de colonia de hormigas existen que permiten resolver TSP?

P3.	¿Qué algoritmos de optimización de colonias hormigas incorporan características naturales de hormigas cultivadoras de hongos?

2.2.2. CADENA DE BÚSQUEDA

Como siguiente paso se obtuvieron las palabras claves de las preguntas de la RSL como lo expone [24], con la finalidad de construir las cadenas de búsqueda (CB). Se usan conectores lógicos “AND” y “OR”, por cada pregunta como muestra la Tabla 2.

Las CB se ejecutaron sobre bases de datos digitales cómo Scopus, ScienceDirect y ACM Digital; En la figura 3 se muestra la configuración para CB1, que busca identificar cuáles han sido las soluciones de ACO para el problema de TSP.

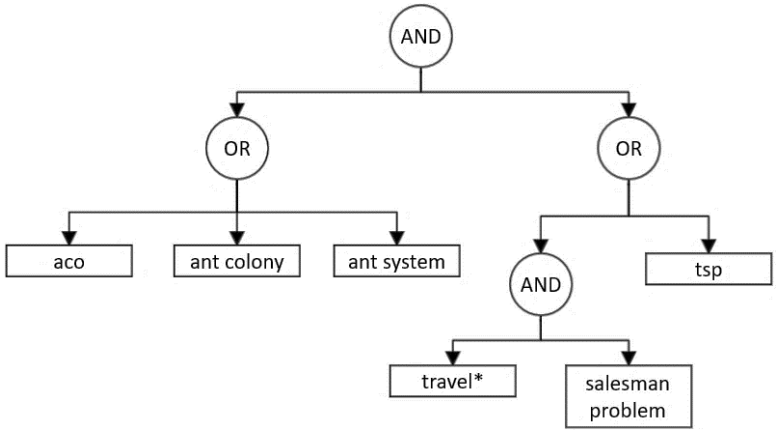

Figura 3. Cadena de búsqueda pregunta 1.

En la figura 4, se muestra la configuración de CB2 que aporta en determinar cuáles son los diferentes tipos de ACO para el problema de TSP en los últimos años.

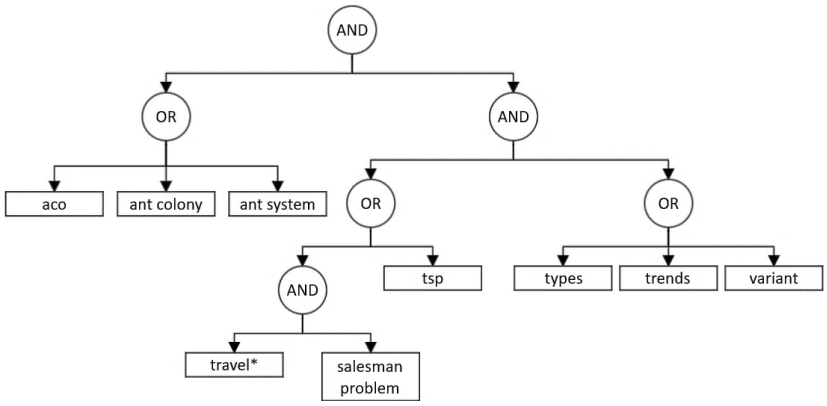

Figura 4. Cadena de búsqueda pregunta 2.

Finalmente, la figura 5 muestra la configuración de CB3 para determinar si existen soluciones de ACO que incorporen conceptos o características de hormigas cultivadoras de hongos. Cada cadena fue adaptada según las opciones avanzadas de búsqueda que brindan cada uno de los motores de las fuentes seleccionada.

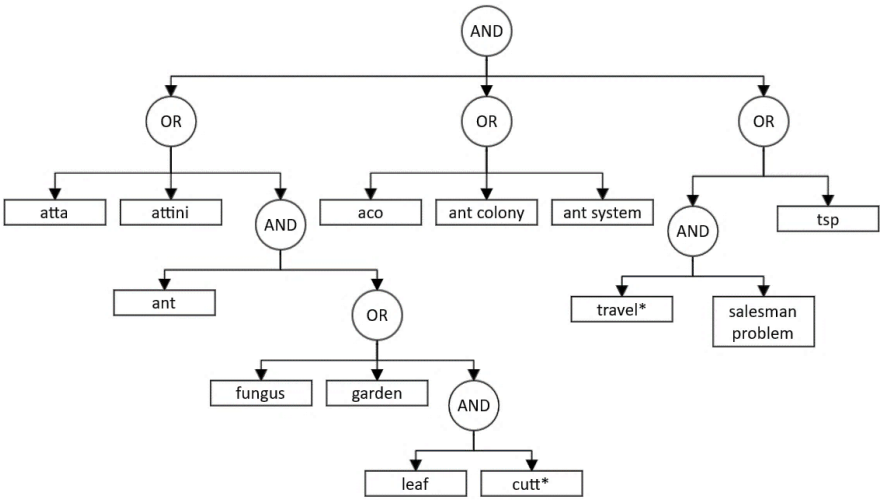

Figura 5. Cadena de búsqueda pregunta 3.

2.2.3. EJECUCIÓN DE CB

Para la búsqueda de la información, se usaron las bases de datos digitales mencionadas anteriormente, la tabla 3. Muestra la configuración para cada una de las CB por motor de búsqueda aplicando criterios de inclusión y exclusión para la RSL.

Tabla 3. Cadena de búsqueda por pregunta de RSL

CB	Fuentes	CB ajustada por motor	Resultado
CB1	*Scopus*	*TITLE-ABS-KEY (("ant colony optimization" OR aco OR "ant colony" OR "ant system") AND ("travel salesman problem" OR "travelling salesman problem"OR TSP)) AND PUBYEAR > 2014 AND PUBYEAR < 2020 AND (LIMIT-TO(LANGUAGE,"English") OR LIMIT-TO (LANGUAGE,"Spanish"))*	475
	ScienceDirect	*Title, abstract or author-specified keywords ("ant colony optimization" OR aco OR "ant colony*" OR "ant system") AND ("travel salesman problem" OR "travelling salesman problem"OR TSP) Year 2014 -2019*	66
	ACM Digital	*Por titulo: [[Publication Title: "ant colony optimization"] OR [Publication Title: aco] OR [Publication Title: "ant colony"] OR [Publication Title: "ant system"]] AND [[Publication Title: "travel salesman problem"] OR [Publication Title:*	5

CB	Fuentes	CB ajustada por motor	Resultado
		"travelling salesman problem"] OR [Publication Title: or tsp]] AND [Publication Date: (01/01/2014 TO 12/31/2019)]	
		Por Abstract: [[Abstract: "ant colony optimization"] OR [Abstract: aco] OR [Abstract: "ant colony"] OR [Abstract: "ant system"]] AND [[Abstract: "travel salesman problem"] OR [Abstract: "travelling salesman problem"] OR [Abstract: or tsp]] AND [Publication Date: (01/01/2014 TO 12/31/2019)]*	20
		Por palabras claves: [[Keywords: "ant colony optimization"] OR [Keywords: aco] OR [Keywords: "ant colony"] OR [Keywords: "ant system"]] AND [[Keywords: "travel salesman problem"] OR [Keywords: "travelling salesman problem"] OR [Keywords: or tsp]] AND [Publication Date: (01/01/2014 TO 12/31/2019)]*	7
CB2	*Scopus*	*TITLE-ABS-KEY (("ant colony optimization" OR aco OR "ant colony" OR "ant system") AND ("travel salesman problem" OR "travelling salesman problem" OR tsp) AND (type OR variant OR trend)) AND PUBYEAR > 2014 AND PUBYEAR < 2020 AND (LIMIT-TO (LANGUAGE , "English") OR LIMIT-TO (LANGUAGE , "Spanish"))*	308
	ScienceDirect	*("ant colony optimization" OR aco OR "ant colony" OR "ant system") AND ("travel salesman problem" OR "travelling salesman problem" OR tsp) AND (type OR trend OR variant) Year 2014 -2019*	22
	ACM Digital	*Por titulo: [[[Publication Title: "ant colony optimization"] OR [Publication Title: aco] OR [Publication Title: "ant colony"] OR [Publication Title: "ant system"]] AND [[Publication Title: "travel salesman problem"] OR [Publication Title: "travelling salesman problem"] OR [Publication Title: tsp]] AND [[Publication Title: variant] OR [Publication Title: trend] OR [Publication Title: types]]] OR [All:))] AND [Publication Date: (01/01/2014 TO 12/31/2019)]*	**0**
		Por Abstract: [[[Abstract: "ant colony optimization"] OR [Abstract: aco] OR [Abstract: "ant colony"] OR [Abstract: "ant system"]] AND [[Abstract: "travel salesman problem"] OR [Abstract: "travelling salesman problem"] OR [Abstract: tsp]] AND [[Abstract: variant] OR [Abstract: trends] OR [Abstract: types]]] OR [All:))] AND [Publication Date: (01/01/2014 TO 12/31/2019)]	**5**

CB	Fuentes	CB ajustada por motor	Resultado
		Por palabras claves *[[[Keywords: "ant colony optimization"] OR [Keywords: aco] OR [Keywords: "ant colony"] OR [Keywords: "ant system"]] AND [[Keywords: "travel salesman problem"] OR [Keywords: "travelling salesman problem"] OR [Keywords: tsp]] AND [[Keywords: variant] OR [Keywords: trend] OR [Keywords: type]]] OR [All:))] AND [Publication Date: (01/01/2014 TO 12/31/2019)]*	0
CB3	*Scopus*	*TITLE-ABS-KEY (atta OR Atta OR ant-fungus OR garden OR leafcutter OR fungus-garding OR leaf-cutting OR farming) AND TITLE-ABS-KEY (ant OR "ant colony" OR "ant sysyem" OR aco OR "ant colony optimization" OR "ant colony optimization algorithm") AND PUBYEAR > 2008 AND PUBYEAR < 2020 AND (LIMIT-TO (SUBJAREA , "ENGI") OR LIMIT-TO (SUBJAREA , "COMP")) AND (LIMIT-TO (LANGUAGE , "English") OR LIMIT-TO (LANGUAGE , "Spanish"))*	39
	ScienceDirect	*Title, abstract or author-specified keywords (atta OR AttaOR antfungus OR garden OR leaf-cutter OR leaf-cutting OR leafcutting OR farming) AND (ant OR "ant colony" OR "ant system" OR aco OR "ant colony optimization") Year 2009 -2019*	2
	ACM Digital	*Por titulo: [[Publication Title: atta] OR [Publication Title: attini] OR [Publication Title: antfungus] OR [Publication Title: garden] OR [Publication Title: leaf-cutter] OR [Publication Title: leaf-cutting] OR [Publication Title: leafcutting] OR [Publication Title: farming]] AND [[Publication Title: ant] OR [Publication Title: "ant colony"] OR [Publication Title: "ant system"] OR [Publication Title: aco] OR [Publication Title: "ant colony optimization"]] AND [Publication Date: (01/01/2009 TO 12/31/2019)]*	**1**
		Por Abstract: [[Abstract: atta] OR [Abstract: attini] OR [Abstract: antfungus] OR [Abstract: garden] OR [Abstract: leaf-cutter] OR [Abstract: leaf-cutting] OR [Abstract: leafcutting] OR [Abstract: farming]] AND [[Abstract: ant] OR [Abstract: "ant colony"] OR [Abstract: "ant system"] OR [Abstract: aco] OR [Abstract: "ant colony optimization"]] AND [Publication Date: (01/01/2009 TO 12/31/2019)]	**15**
		Por palabras claves: [[Keywords: atta] OR [Keywords: attini] OR [Keywords: antfungus] OR [Keywords: garden] OR [Keywords: leaf-cutter] OR [Keywords: leaf-cutting] OR [Keywords: leafcutting] OR [Keywords: farming]] AND [[Keywords: ant] OR [Keywords: "ant colony"] OR [Keywords: "ant system"] OR [Keywords: aco] OR [Keywords: "ant	**2**

CB	Fuentes	CB ajustada por motor	Resultado
		colony optimization"]] AND [Publication Date: (01/01/2009 TO 12/31/2019)]	

Se encontraron en total 976 trabajos, 576 relacionados con P1, 344 relacionados con P2 y 59 con P3. Después de obtener los estudios de cada una de las bases de datos se procede a realizar la revisión aplicando criterios de inclusión y exclusión como lo indica [24].

2.2.4. PROCESO DE REVISIÓN

Después de obtener los estudios de cada una de las bases de datos se procede a realizar la revisión aplicando criterios de inclusión y exclusión como lo indica [24], el cual se muestra en la figura 6.

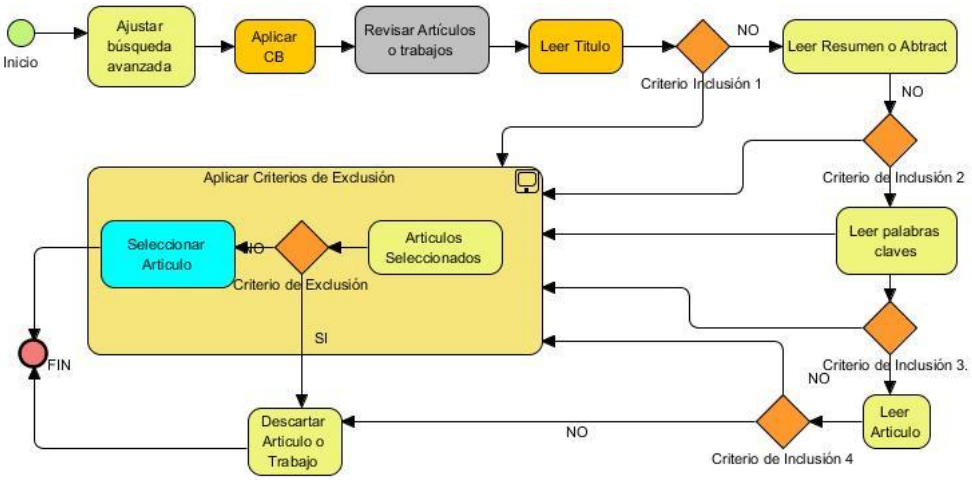

Figura 6. Proceso de la Revisión

2.2.4.1. CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN

Los criterios de inclusión de trabajos se aplicaron en el siguiente orden, Ci1: El título de los trabajos que evidencie relación con el problema de investigación. Ci2: Resumen (abstract) de los trabajos previamente seleccionados que evidencia relación con el problema seleccionado. Ci3: Palabras claves de los trabajos que evidencia el problema relacionado Ci4: Trabajos que presenten comparación de diferentes implementaciones de ACO. Los criterios de exclusión de los trabajos son los siguientes: Ce1: Más de 5 años en artículos científicos, excepto para la cadena de búsqueda CB3 que se amplió a 10 años para descartar posibles implementaciones con el tema en cuestión. Ce2: Referencias que no estén en inglés o español. Ce3: Referencias a las que no se tenga acceso completo. Ce4: Eliminar estudios repetidos. Ce5: Se excluirán aquellos artículos que no hayan sido citados excepto para la cadena de búsqueda Cb3. Ce6: Se excluyen todos los trabajos que no apliquen los criterios de inclusión.

2.2.5. EVALUACIÓN DE CALIDAD

La calidad de los trabajos seleccionados se evaluó respecto a 6 de los 11 criterios para determinar los estudios de calidad desarrollada en [25], las cuales buscan que los estudios

cumplan con tres aspectos generales que son el rigor, la credibilidad y la relevancia. Los estudios se evaluaron según cada aspecto y criterios expuestos en la tabla 4.

Tabla 4. Aspectos y criterios de calidad de evaluación.

Aspectos		Criterios
Rigor	I.	El alcance y los objetivos de la investigación son claros.
	II.	Se identifica una descripción adecuada del contexto en el que se llevó a cabo la investigación.
Credibilidad	III.	El diseño de la investigación es apropiado para abordar los objetivos de la investigación
	IV.	Utilizaron y describieron métodos apropiados de recopilación de datos.
Relevancia	V.	Se realizó una descripción adecuada de los métodos utilizados para análisis de datos.
	VI.	El estudio proporcionó hallazgos claramente establecidos con resultados creíbles y conclusiones justificadas.

En el primer criterio el Rigor se refiere si el estudio siguió un método de investigación adecuado, el segundo criterio la Credibilidad que indica si los resultados son significativos y son bien presentado. Y el tercer criterio la Relevancia, se refiere si los resultados obtenidos son útiles.

2.2.6. RECOPILACIÓN DE DATOS

Para extraer la información se aplicaron los criterios de inclusión/exclusión y los criterios de evaluación de calidad. Descartando por título, palabras claves y resumen un total de 367 estudios, los artículos repetidos aplicados a las 3 preguntas formuladas fueron en total 228 y aquellos que nos cumplían con el tema principal de ACO, TSP y no cumplía los criterios de calidad en total 295 estudios. En total se seleccionaron 84 estudios distribuidos como muestra la tabla 5 para las preguntas P1 - P2 y la tabla 6 de la pregunta P3.

Tabla 5. Estudios seleccionados por año preguntas P1 y P2.

Pregunta	2019	2018	2017	2016	2015	2014	Total
P1	6	6	5	5	6	1	**29**
P2	2	3	1	0	1	0	**7**

Tabla 6. Estudios seleccionados por año preguntas P3.

Pregunta	2019 - 2018	2017-2016	2015-2014	2013-2012	2011 - 2010	2009	Total
P3	1	1	0	0	0	0	**8**

La información de los estudios se condenso en un archivo de Excel, para mejorar la organización de los estudios primarios información, la estructura que se muestra en la Figura 4.

Titulo	Tipo de publicación	Autores	Resumen Temas importantes	Palabras Claves	Año	Idioma	Base de datos Digital	Enlace base de datos Revista Digital	Pregunta de investigación

Figura 7. Estructura del archivo de información seleccionada

2.2.7. RESULTADOS DE LA REVISIÓN SISTEMÁTICA

Los estudios de investigación identificados como fuente de información primaria son 51 siguiendo el proceso de revisión de la Figura 3, el proceso detallado se muestra en Tabla 7, para cada una de las preguntas de investigación.

Tabla 7. Proceso detallado selección de estudios

Paso	Estudios Prelimares	Actividad	Estudios Seleccionados
1	P1: 572 P2: 344 P3: 59 **Total: 976**	Identifiifcar estudios relavantes según criterios de Inclusión	P1: 188 P2: 82 P3: 20 **Total: 290**
2	P1:188 P2:82 P3:20 **Total:290**	Estudios que se les aplico el proceso de exclusión	P1: 47 P2: 18 P3: 8 **Total: 73**
3	P1: 47 P2: 22 P3: 8 **Total: 73**	Evaluación de calidad	P1: 29 P2: 7 P3: 2 **Total:38**

De la pregunta 1 se seleccionaron 29 estudios primarios de diferentes implementaciones de ACO para dar solución al problema del agente viajero, como soluciones para mejorar la evaporación y depósito de la feromona, nuevos métodos para realizar la construcción de la solución por medio de heurísticas y modelos matemáticos, soluciones combinadas con metaheurísticas para mejorar la solución, además de comparaciones de diferentes implementaciones de ACO aplicados a TSP como la librería TSPLIB, adicional ajustes de parámetros de manera automática, esta información nos brinda un panorama de cómo se ha realizado aplicado ACO para abordar el problema del agente viajero.

Por otro lado, para la pregunta 2 se seleccionaron 7 artículos que muestran nuevas tendencias, mejoras y como se han implementado ACO para TSP, como variantes con otras metaheurísticas o híbridos potenciando la profundización y diversificación en la búsqueda de la solución, además la utilización de computación paralela para optimizar la construcción de la solución o para la identificación de múltiples caminos utilizando CUDA, el uso de lógica difusa como método de control para ajustar parámetros, mejora en la población por medio de operadores genéticos del algoritmos genéticos. Y para la pregunta 3 se

seleccionaron 2 artículos que muestran como en diferentes áreas de aplicación han aplicado el concepto de castas o múltiples colonias de hormigas, que es similar a las hormigas cultivadoras de hongos.

Después de realizar la revisión de los estudios se consideró que para realizar un mejor análisis de resultados era necesario agruparlos en 5 categorías como se observa en la Tabla 8.

Tabla 8. Estudios primarios seleccionados por categoría.

Categoría	Referencia	Cantidad
Computación Paralela	[26] - [27] - [28] - [29] - [30]	5
Conceptos de hormigas cultivadoras de hongos	[31] - [32]	2
Híbrido	[33] - [34] - [35] - [36] - [37] - [38] -[39] - [40] - [41]	9
Variantes	[42] - [43] - [44] - [45] - [46] - [47] -[48] - [49] - [50] - [51] - [52] - [53] - [54] - [55] - [56] - [57] - [58] - [59] - [60] - [61] - [62] - [63]	22

2.3. ANTECEDENTES

Del resultado de la revisión de sistemática se identifica los siguientes antecedentes, los cuales son categorizados según lo descrito en la sección anterior.

2.3.1. COMPUTACIÓN PARALELA

Los estudios relacionados exponen soluciones de ACO aplicando conceptos de computación paralela enfocados a la construcción de la solución para mejorar de esa forma la convergencia de la solución.

En [26] presentan un ACO paralelo usando arquitectura CPU de una sola instrucción con múltiples datos (SIMD en su siglas en inglés), y es probado en instancias de TSP de 196 a 4461 ciudades. La propuesta toma cada hormiga creándola en un núcleo de CPU y la construcción de una ruta de cada hormiga se acelera mediante instrucciones vectoriales, apoyado por un método de selección por ruleta, que es clave en los procesos de paralelismo, ya que este proceso consume mucho tiempo por ser altamente estocástico. Para evaluar la eficiencia de la propuesta realizan una comparación con 3 algoritmos un ACO tradicional, sistema de hormigas (AS) y ACO-3opt, en 7 instancias TSP (d198, lin318, pcb442, rat783, pr1002, fl1577, pr2392, pcb3038, fnl4461), donde mantienen una solución similar, superando los algoritmos en tiempo basado en microsegundos.

En [27] realizan una comparación entre un RBPACO (Best Probability ACO), contra un RBPACO paralelo en instancias de TSP, RBPACO es un ACO basado en un controlador estocástico, donde generan grupos de tres potenciales ciudades con la probabilidad más

alta de selección, lo que garantiza que varias hormigas no produzcan soluciones idénticas en la mista iteración, de esta forma evitan estancamientos. Se enfocan en la calidad de las soluciones y en el tiempo de ejecución de los algoritmos para las instancias de TSP Eil51, Eil76 y KroA100 donde se realizan experimentos de 300 iteraciones con 10 repeticiones. El mejor resultado para RBPACO paralelo tiene un margen de error del 0.2% en comparación de RBPACO tradicional que presenta un error del 2.3% para Eil51. Por otro lado, para KroA100 RBPACO paralelo tiene un margen de error de 0.1% con respecto al RBPACO tradicional que presenta un error del 5.2%, los autores evidencian que entre más grande es el problema a resolver mejor se comporta el ACO paralelo.

En [28] Indica que los algoritmos ACO en particular no son tan fáciles de paralelizar debido que se debe tener un conocimientos global del problema para ser implementados de forma correcta, proponen un enfoque de HPC (high.perfomance computing) basado en una arquitectura de modular (Actor model), donde implementan un módulo específico para manejar de forma sincrónica el depósito de la feromona para cada hormiga artificial, distribuyendo la feromona de forma sincrónica para cada hormiga remota. Utilizan una instancia de TSP (pr152), donde comparan la versión distribuida contra un ACO secuencial, ejecutado en 20 repeticiones de 100 iteraciones cada una. Los márgenes de error son poco significativos, pero el algoritmo ACO distribuido propuesto converge más rápido que el ACO secuencial.

En [29] se presentan 3 versiones de un algoritmo de sistema de hormigas (ACS) usando unidades de procesamiento grafico (GPU) con Nvidia CUDA, donde los 2 primeros algoritmos proponen usar una matriz de feromona estándar, el primero para ACS-GPU se centra en el proceso de construcción de la solución, el segundo ACS-GPU-Alt busca acelerar el proceso usando una única ejecución del kernel y tercero ACS-GPU-SPM usa una memoria de feromonas usando selección donde solo se dejan los arcos importantes (distancia más corta) para construir buenas soluciones. Se realizan pruebas de ASC secuencial y los 3 algoritmos propuestos a 8 instancias de TSP (d197, A280, lin318, pcb442, at783, pr1002, nrw1379, pr2392), ejecutadas en 1000 iteraciones por cada algoritmo, donde se observa que los 3 convergen más rápido en tiempo que el ACS secuencial.

En [30] usan el modelo paralelo para dividir la colonia de hormigas en subgrupos realizando un ACO secuencial por cada subgrupo por medio de computación distribuida de tipo maestro Esclavo, donde cada subcolonia se ejecuta en un esclavo y luego el maestro compara cual es la mejor solución de los esclavos. Usan como instancia de prueba KroA100.

2.3.2. CONCEPTOS DE HORMIGAS CULTIVADORAS DE HONGOS

Este tema es de vital importancia, permite identificar con que características de hormigas cultivadoras de hongos se ha trabajado o conceptos similares; utilizando conceptos como castas o tipos de hormigas.

En [31], proponen una variante de ACO llamada HHACO donde realizan una comunicación heurística con una población dual, la primera es una población de hormigas que se encarga de diversificar la solución utilizando un algoritmo de sistema de hormigas y la otra se encarga de la velocidad de convergencia por medio de autoadaptación usando sistema de colonia de hormigas mejorado (IASC). Por otro lado, la propuesta se basa en una heurística de comunicación entre AS y IASC por medio de una actualización entre matrices de feromona manteniendo la individualidad entre las colonias. Se compara HHACO con otras

técnicas de ACO, donde obtienen un mejor óptimo específicamente para problemas de TSP de larga escala.

En [32], esta propuesta presenta un nuevo algoritmo multiobjetivo, donde generan 3 tipos de hormigas basadas en comportamientos socio-cognitivos. La idea se centra en generar feromonas múltiples donde cada tipo de hormiga interactúan cambiando la perspectiva, esto quiere decir que cambia las decisiones tomadas por una hormiga al depositar diferentes rastros feromona, donde cada hormiga utiliza diferentes reglas heurísticas para calcular la atractividad. Esta estrategia genera un tipo de "olor" diferente para cada hormiga lo que evita caer en óptimos locales.

2.3.3. HÍBRIDOS:

Los estudios pertenecientes a este tema tienen soluciones para mejorar ACO con otros metaheurísticas o algoritmos Bio-Inspirados como Tabú search, Cuckoo search y algoritmos genéticos, entre otros. Es importante tenerlos en cuenta para identificar las soluciones ya que para el diseño e implementación del nuevo algoritmo no se busca realizar un híbrido.

En [33] los autores proponen un hibrido entre recocido simulado (SA), ACO y un operador de mutación de un algoritmo genético (AG), con el fin de evitar el estancamientos usan SA y el operador de mutación para diversificar la población de las hormigas, creando hormigas elitistas y mejorando el proceso con una búsqueda local focalizada. El nuevo algoritmo se prueba en 24 instancias de TSPLIB entre 50 a 1650 ciudades, es comparado contra un ACO secuencial y ACO hibrido con optimización de enjambre de partículas, los resultados muestran que el nuevo algoritmo supera a los otros, en un porcentaje pequeño.

En [34] proponen una hibrido con entre ACO y un algoritmo de agrupación basado en densidad, donde sectorizan las rutas, dividiendo las rutas en subrutas con el algoritmo de agrupación basado en densidad identificando cuales son las ciudades más cercanas y usan ACO secuencial para resolver cada subruta, luego se unen las subrutas para ser optimizadas con k-opt. Utilizan 30 instancias de TSPLIB para las pruebas dividido en tres partes pequeña, mediana y gran escala, se comparan contra 8 algoritmos ACO, donde la solución propuesta reduce el tiempo de ejecución para problemas de gran escala.

En [35] proponen un ACO con múltiples estrategias para evitar estancamientos prematuros, identifican que lugares son los susceptibles a mejorar, por ejemplo para mejorar la distribución inicial de la feromona usan el método del vecino más cercano, por otro lado para mejorar el espacio de búsqueda usan un método de evolución cruzada, combinan un cruce genético con transición de estados para saltar óptimo locales. Usan 8 instancias de TSPLIB con una colonia de 35 hormigas, se realizan 20 repeticiones de 1000 iteraciones, el ACO con múltiples estrategias supera a 3 ACO usados para comparación.

En [36] se usa optimización por enjambres de partículas (PSO) para mejorar ACO, asignado a cada una partículas, mejorando el depósito de la feromona como paso de información, aquí es donde PSO identificando cuáles son los extremos locales y los extremos globales, a partir de los movimientos de cada hormiga se obtienen los extremos locales usando una lista tabú para identificar el paso de la partícula.

En [37] proponen un algoritmo hibrido bidireccional HBACO, que es comparado con otras implementaciones de ACO, BACO, es usado para problemas asimétricos que significa que

la distancia de ir de una ciudad *i* a *j* es diferente de *j* a *i*. Se cambia una ruta unidireccional por una ruta bidireccional ampliando el efecto de exploración de la hormiga, cuando la hormiga termina el camino se integran los caminos de dos vías para actualizar la feromona, haciendo que se mejoren las rutas.

En [38], proponen soluciones de un hibrido con un sistema de hormigas llamado Min Max (MMAS), además de programación linear relaxation (LP-relaxation), primero se usa LP para reducir el tamaño del problema, se mejorar el depósito de la feromona con MMAS para mantener un nivel predefinido de exploración.

En [39], proponen un algoritmo memético con Max-Min ACO (MMAS) integrando con un operador de búsquedas local el cual permite realizar un cambio dinámico permutando diferentes algoritmos de búsqueda local como 2-opt o 3-opt, los cuales son lanzados cuando se identifica que no hay una mejora en un determinada iteración, esto mejora significativamente el rendimiento de MMAS para problemas de TSP.

En [40], realizan un hibrido para el problema de planificación y de enrutamiento de servicios de mensajería, donde utilizan un sistema experto para determinar la fecha de visita a un cliente dentro de la planeación, por otro lado usan algoritmo de agrupamiento basado en centroides para determinar el grupo de vértices visitado por un vehículo, realizan este proceso dado que el espacio de búsqueda es amplio por lo cual se hace necesario agruparlo y finalmente usan ACO para ordenar los vértices para cada vehículo de cada programación, realizando una evaluación el proceso de evaluación identificando una mejora en la ruta y depositando feromona entre cada vértice dependiendo del agrupamiento., lo que permitió encontrar una secuencia para visitar cada clúster teniendo en cuenta la ventana de tiempo por cliente y vehículo.

En [41], realizan una comparación experimental entre un sistema de hormigas(AS) contra una variante ACO, con el objetivo de medir el desempeño de los algoritmo en 7 instancias de TSPLIB, con diferentes configuraciones de parámetros explicando el comportamiento que AS obtiene al modificar cada uno de los parámetros.

2.3.4. VARIANTES

Los estudios aquí expuestos muestran cómo mejorar ACO realizando variantes al ACO original, utilizan conceptos para mejorar la feromona que usa el algoritmo, nuevas propuestas para la construcción de la solución, mejoras para automatizar el refinamiento de los parámetros de ACO, conceptos que pueden aplicarse en el diseño del algoritmo a implementar.

En [42], presentan una optimización del algoritmo de colonia de hormigas llamado ICMPACO, para solucionar el problema de planificación y TSP. La variante propone una estrategia de múltiples poblaciones, con estrategia de coevolución donde dividen el problema de varios subproblemas y las hormigas de la población en dos tipos de hormigas, las primeras son las elites que definen un sistema de parámetros propios para mejorar la tasa de convergencia de ACO y segundas se llaman comunes que son utilizadas para generar nuevas soluciones por medio de una función gaussiana para evitar en caer en óptimos locales. Es probado en 8 instancias de TSPLIB donde ICMPACO no obtiene los resultados óptimos, pero obtiene resultados de calidad ya que obtiene una mejor capacidad de optimización y de estabilidad con respecto al ACO tradicional.

En [43], proponen una variante de ACO para la solución del problema TSP bi-objetivo, la propuesta llamada PM-MOACO es un algoritmo multiobjetivo utilizando una optimización en la matriz de feromona usando un modelo matemático inspirado en *Physarum*(PMM), que permite encontrar rutas más cortas entre nodos basados en la retroalimentación positiva. Además, proponen un framework para la adaptación de diferentes algoritmos entre PMM y diferentes tipos de ACO, por otro lado, realizan una comparación entre las diferentes implementaciones de ACO multiobjetivos, donde PM-MOACO obtiene resultados mejores en la mayoría de las comparaciones realizadas ya que promueve la explotación de la solución óptima.

En [44], aplican una hyperheurística la cual está diseñada para aumentar la generalidad de los sistemas de optimización de tal forma que la técnica se pueda reutilizar en otro problemas realizando cambios pequeños en las heurísticas simples y en la función de evaluación. La hyperheurística ACO (ACOhh), donde las hormigas al pasar de nodo a nodo generando una heurística simple en acciones concretas como la actualización de feromona, donde la mejora se considera como la diferencia entre la mejor calidad de la solución actual y la mejor calidad de la solución anterior. HHACOhh es comparado con diferentes metaheurísticas ejecutado en 4 instancias de TSP que tiene como ventaja la utilización de heurísticas de bajo nivel y que son fáciles de implementar.

En [45], es una variante de un algoritmo de sistema de colonias de hormigas (ACS) basado en un sistema difuso para adaptar de manera dinámica los parámetros de la feromona local, esto evita caer en óptimos locales durante la construcción de la solución. Demostraron que este enfoque permite una convergencia más rápida y con soluciones de buena calidad, concluyendo, que al adaptar el parámetro de la feromona se proporciona un control flexible de la información de la heurística equilibrando la búsqueda de exploración y explotación para encontrar buenas soluciones.

En [46], en este caso se presenta DFACO, que es un adaptación del algoritmo de optimización de colonia de hormigas voladoras, donde una hormiga deposita feromona a distancia, por lo cual, no solo los nodos en la ruta reciben feromona, sino que también sus nodos vecinos, esto ayuda equilibrar las estrategias de exploración y explotación del algoritmo, además se realiza una adaptación de la búsqueda local usando 3-opt para evitar el estancamiento y de esta forma mejorar la solución. Para evaluar el algoritmo se compara la propuesta con 5 implementaciones de ACO en 22 instancias que serán donde DFACO obtiene buenos resultados en la mayoría de las soluciones.

2.4. CONCLUSIONES

El estado del arte ayuda a identificar cuáles son aquellas características de *Atta* que pueden ser susceptibles para ser usadas en la construcción de la una nueva variante de ACO.

La RSL permitió seleccionar los estudios primarios para realizar un análisis de las diferentes implementaciones de ACO y de esta forma tener una visión general para el diseño e implementación de una nueva variante de ACO que incluya características de hormigas cultivadoras de hongos o *Atta.*

Como resultado de la revisión sistemática se obtiene:

- Un protocolo de revisión sistemática que permitió definir cuáles eran los pasos para seguir para definición de la RSL – Anexo 1.

- Estado del arte de las implantaciones de ACO, sus variantes en la construcción de la solución, adaptaciones para mejorar el depósito y evaporación de feromona, técnicas hibridas con otros métodos o metaheurísticas.

3 DISEÑO DEL ALGORITMO

En este capítulo se muestra el proceso para el diseño del algoritmo, donde se introducen los conceptos naturales de las hormigas *Atta*, los cuales se usarán como inspiración para mejorar el algoritmo de colonia de hormigas.

Se exponen las decisiones de diseño, la prueba de concepto del desarrollo del algoritmo aplicado a tres problemas tomados de la librería TSPLIB [64].

3.1. MARCO CONCEPTUAL

3.1.1. HORMIGAS *ATTA*

El algoritmo de optimización de colonia de hormigas se basa en el proceso natural de las hormigas al buscar su alimento, como se describió en el capítulo anterior. Sin embargo, aunque ACO fue inspirado por características naturales de las colonias de hormigas, existen colonias de hormigas con comportamientos más complejos. Un ejemplo de ello son las hormigas arrieras o también conocidas como *Atta*, las cuales se destacan por tener un sistema de agricultura en donde cultivan un hongo el cual usan como alimento para la colonia, además, también han desarrollado un sistema sanitario de prevención de sustancias nocivas que puedan afectar al hongo y a la colonia.

Las *Atta* están divididas en castas especializadas en tareas concretas para preservar el bienestar de la colonia. Una casta la *exploradora* es la encargada de realizar el proceso de búsqueda y recolección de hojas o restos vegetales que aporten al proceso de cultivo del hongo.

Otra casta, la *seleccionadora,* es una hormiga que se encarga del proceso de desinfección, realiza un proceso de selección que identifica que hormigas recolectoras están infectadas para evitar el ingreso a la cámara de cultivo del hongo, de esta forma previenen que el ingreso de alimento nocivo que puede impedir el crecimiento del hongo.

Finalmente, la casta *cultivadora* es una hormiga más pequeña que se encarga del proceso de cultivo del hongo, la cual ayuda al crecimiento de este, que es la principal fuente de alimento de la colonia.

La figura 8 muestra el proceso realizado por cada una de las castas de las hormigas *Atta*. En la figura 8.a), se ilustra como las hormigas exploradoras inician el proceso de búsqueda de alimento por alguna de las tres rutas; en la figura 8.b), se muestra dos partes del proceso, por un lado las hormigas exploradoras encuentran el alimento y depositan feromona para hacer atractiva la ruta más corta, por el otro las hormigas seleccionadoras verifican si las hormigas exploradoras tiene algún tipo de patógeno dañino para el hongo y no permiten el ingreso de ese alimento a la colonia; la figura 8.c), muestra como las hormigas exploradoras ya no visitan las rutas con alimento contaminado y como las hormigas seleccionadoras permiten ingresar a la colonia a las hormigas exploradoras con el alimento para iniciar el proceso de cultivo; en la figura 8.d), muestra como las hormigas cultivadoras realizan el proceso de cultivo del hongo, haciendo que este crezca con el alimento obtenido por las hormigas exploradoras.

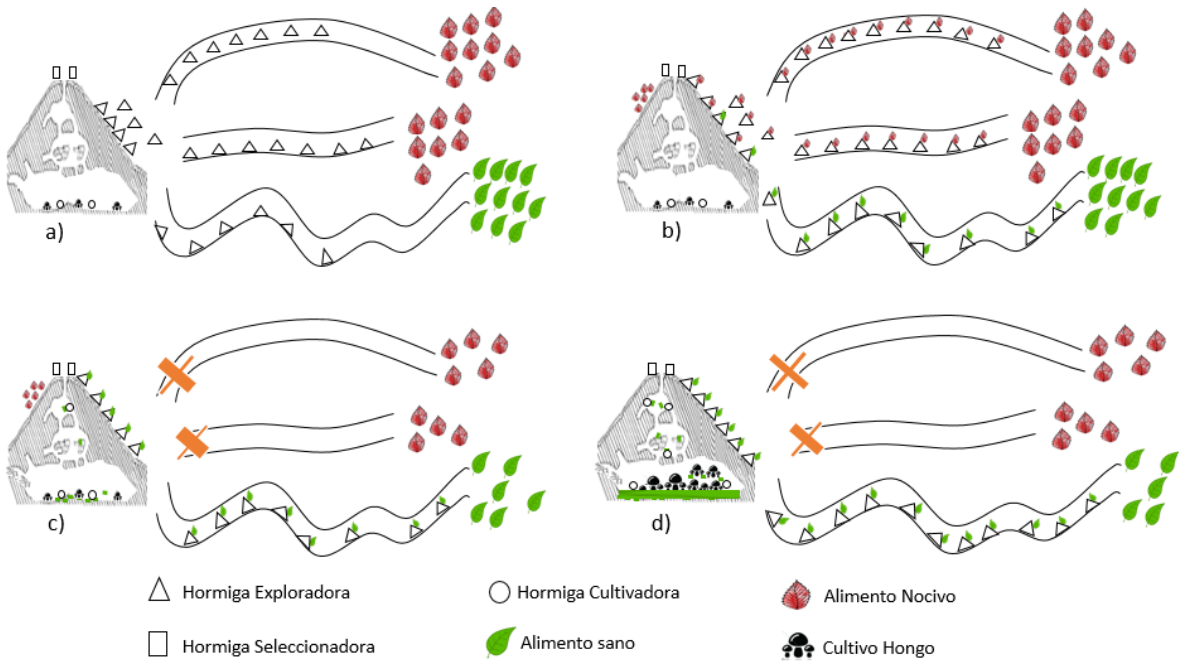

Figura 8. Proceso natural hormigas *Atta*.
Panel a) Proceso de búsqueda de la hormiga exploradora; Panel b) Hormiga exploradora depositando rastro de feromona; Panel c) Hormiga seleccionadora identificando hojas adecuadas para el cultivo; Panel d) Hormiga cultivadora realizando el proceso de cultivo.

3.1.2. DESCRIPCIÓN DEL PROBLEMA

Como se mencionó en el capítulo 2, uno de los problemas clásicos que resuelve ACO es TSP, donde se deben visitar *n* ciudades pasando por todas ellas una vez, iniciando y finalizando por la misma ciudad. En la propuesta se trabajó con instancias TSP simétricas lo que significa que la distancia entre las ciudades i y j es igual en cualquier sentido.

TSPLIB [64], es una librería de instancias de TSP curada por la Universidad de Heidelberg. Los investigadores alrededor del mundo utilizan subconjuntos de problemas tomados de esta librería, dado que contiene una variedad de ejemplos junto con las mejores soluciones encontradas hasta la fecha.

La figura 9, muestra un ejemplo de TSP simétrico llamado Burma 14, que consiste en un conjunto de 14 ciudades de Birmania, donde cada punto son sus coordenadas geográficas en latitud y longitud. Las diferentes metaheurísticas como ACO, que se aplican al problema buscan obtener la ruta más corta.

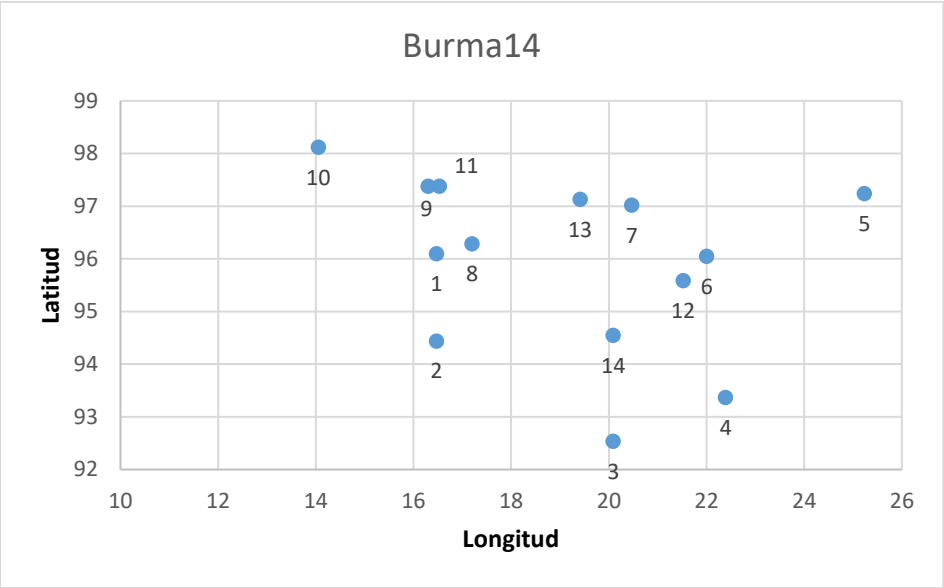

Figura 9. Ejemplo de TSP Burma14. Datos tomados de TSPLIB

Para hallar la solución se obtiene la distancia euclidiana a partir de las coordenadas de latitud y longitud de cada par de ciudades utilizando la fórmula 1.

$$\mathrm{d}(\mathrm{ciudad}_i, \mathrm{ciudad}_j) = \sqrt[2]{((\mathrm{longitud}_j - \mathrm{longitud}_i)^2 - (\mathrm{latitud}_j - \mathrm{latitud}_i)^2} \quad (1)$$

Para encontrar la ruta óptima se exploran los posibles recorridos, buscando minimizar la distancia total recorrida. La figura 10, muestra la mejor ruta para Burma14 con una distancia total recorrida de 3323.

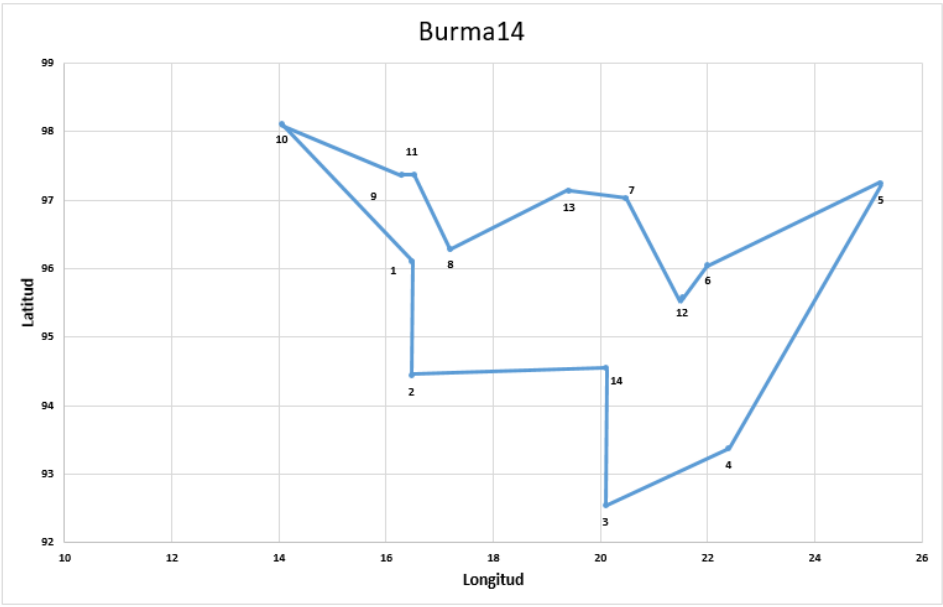

Figura 10. Mejor ruta de Burma14

3.2. DISEÑO DEL ALGORITMO

Para la propuesta del nuevo algoritmo centrado en resolver el problema de TSP, a continuación, se describen el proceso para el diseño del algoritmo adicionando características de las hormigas *Atta.*

3.2.1. DE LA HORMIGA NATURAL A LA ARTIFICIAL

Partiendo de las castas de hormigas naturales *Atta*, se realiza la propuesta de hormigas artificiales para encontrar mejores soluciones al problema de TSP. La forma en la que cooperan las castas es uno de los aspectos más importante dentro de la propuesta, a continuación, se describen las hormigas artificiales.

- **Hormiga exploradora:** Se encarga de realizar el recorrido buscando la ruta más corta (mejor solución) para el problema de TSP, sigue el rastro de feromona y selecciona los trayectos con mayor tendencia a ser seleccionada por otras hormigas.

- **Hormiga seleccionadora:** Encargada de descartar las rutas poco optimas por cada iteración, y permite que ingresen al cultivo las mejores rutas (mejor solución por iteración) las cuales serán optimizadas en el cultivo del hongo.

- **Hormiga cultivadora:** Se encarga del proceso de cultivar el hongo. Realiza un proceso de optimización basado en conocimiento para mejorar las rutas (soluciones), se realiza medición de distancias óptimas entre nodos y obtiene las distancias más cortas. Las hormigas cultivadoras reciben una ruta particionada en *n* rutas, de la cual identifican trayectos que pueden ser nocivos para el cultivo (distancia más larga entre nodos), la cual ese penalizada para que no se atractiva por el paso de las hormigas exploradoras, este concepto lo llamaremos *Mecanismo de defensa.*

Otra característica importante es el concepto del cultivo del hongo, que trabaja de forma simbiótica con las hormigas de la colonia, a continuación, se describe el proceso de cultivo de hongo.

- **Cultivo del Hongo:** El proceso de cultivo se realiza para mejorar rutas obtenidas por las hormigas seleccionadoras (mejores soluciones por iteración. El proceso de cultivo lo realizarán varias hormigas cultivadoras, encargadas de ayudar a mejorar las rutas ingresadas al cultivo. Si durante el ciclo de cultivo se encuentra una mejor solución se actualiza la solución global de lo contrario el cultivo del hongo mantiene la solución global actual.

3.2.2. PROCESO DEL ALGORITMO

Después de identificar las características que formaran parte del algoritmo se propone el siguiente proceso de ejecución del algoritmo ACO-ATTA

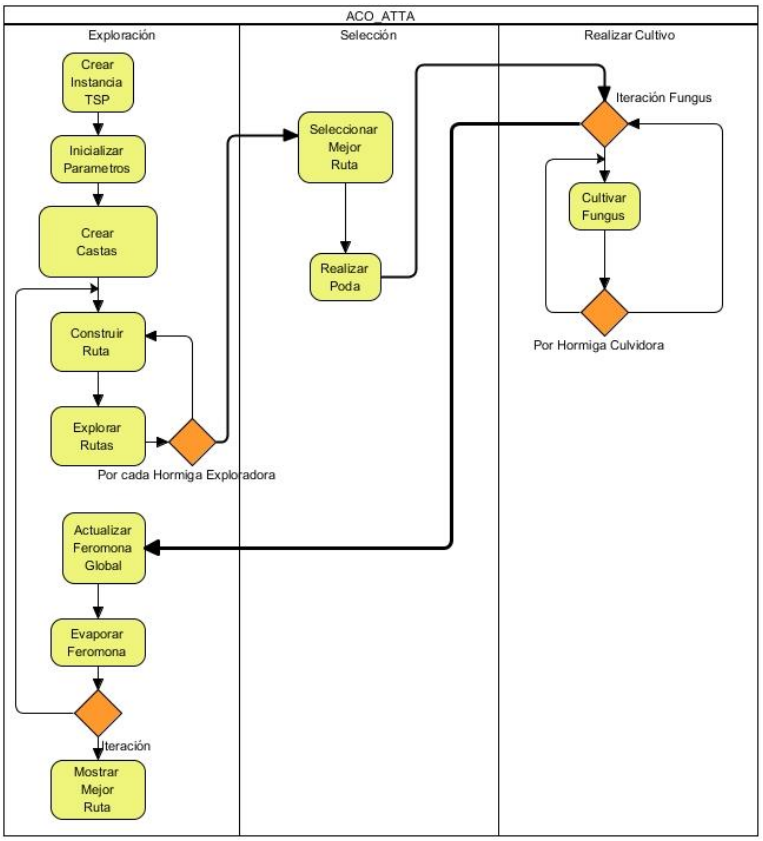

Figura 11. Proceso algoritmo Aco-*Atta*.

La figura 11, muestra cada uno de los procesos realizados por el algoritmo ACO-ATTA, el cual se subdivide en tres subprocesos que son la exploración, la selección y realizar cultivo A continuación, se describe cada uno de los pasos del algoritmo.

Paso 1. Crear instancia: se crea la definición del problema TSP, donde se leen las coordenadas de las ciudades y se carga el grafo que representa el problema.

Paso 2. Inicializar parámetros: Se configuran parámetros ACO-ATTA para el problema del viajero (TSP), como se especifica a continuación.

- ants_number(M): Cantidad de hormigas exploradoras que se van a crear para realizar la exploración de soluciones.
- max_iterations(N): Número de iteraciones que se ejecutara el algoritmo.
- Alpha(α): Factor de influencia heurístico de la feromona
- Betha(β): Factor importancia heurística
- Rho(ρ): Coeficiente de evaporación de la feromona
- Q(q_0): Cantidad de feromona a depositar.
- ants_number_sorter(M_s): Cantidad de hormigas seleccionadora a crear
- ants_number_farmer(M_f): Cantidad hormigas cultivadoras a crear
- max_iterations_fungus(N_f): Número de iteraciones que se ejecutara el algoritmo para el hongo.
- alpha_fungus(α_f): Factor de influencia heurístico de la feromona en el cultivo del hogo

- betha_fungus(β_f): Factor importancia heurística a depositar en el cultivo del hongo
- rho_fungus(ρ_f): Coeficiente de evaporación de la feromona en el cultivo del hongo
- Q_fungus(q_{0f}): Cantidad de feromona a depositar en el cultivo del hongo.
- Problem: instancia de TSP

Paso 3. **Crear Castas**: Esta tarea se encargará de crear las hormigas exploradoras, seleccionadoras y cultivadoras. Donde las primeras van a recorrer todos los nodos y a su paso dejar la feromona. Las segundas realizarían el proceso de elegir la mejor ruta por iteración y la tercera se encargará de realizar el proceso de cultivo.

Paso 4. Construir ruta: esta actividad la realiza la hormiga exploradora, quien se encarga de visitar los nodos verificando si este ha sido visitado o no, hasta terminar de recorrer el grafo.

Paso 4.1. Siguiente Nodo: La hormiga exploradora selecciona el siguiente nodo para la ruta, le asigna probabilidades a cada nodo de acuerdo al atractivo del arco que va del nodo actual a cada uno de los nodos posibles, donde a mayor distancia menor atractivo y a mayor rastro de feromona mayor atractivo (Ver Formula 2)

$$Atractivo(nodo_i, nodo_j = \left(Feromona_{ij}\right)^{\alpha} * \left(1/distancia_{ij}\right)^{\beta} \qquad (2)$$

Paso 5. Explorar ruta: La hormiga exploradora verifica si realizo todo el recorrido, si es así, continua en el paso 6, de lo contrario regresa al paso 4.

Paso 6. Actualizar Solución Global: Se verifica si la solución encontrada por la hormiga exploradora es mejor a la solución global, si es así se reemplaza de lo contrario mantiene la solución global actual.

Paso 7. Seleccionar mejor ruta: Este proceso lo realiza la hormiga seleccionadora, se encarga de tomar la solución top de cada iteración.

Paso 8. Realizar Poda: La hormiga seleccionadora se encarga del proceso de partición de la mejor ruta para ser entregada al cultivo.

Paso 9. Cultivar Fungus: Se ingresan las mejores rutas obtenidas por la hormiga seleccionadora.

Paso 9.1. La hormiga cultivadora toma la partición de la solución seleccionada. Este factor de partición que está dado por la fórmula 3,

$$factor = (1 + iteración \,//\, 20 \,\%\, 4) * 0.1 \quad (3)$$

Este factor se usa para seleccionar el inicio y final de la partición para recortar la solución top obtenida por la hormiga seleccionadora utilizando la fórmula 4. La figura 12 muestra el proceso de la partición.

$$factor_{inicio\ fin} = seleccionar(random(1,\ factor * longitud(ruta) + 1) \quad (4)$$

Figura 12. Partición de la Ruta Top.

Después de obtener la ruta particionada, la hormiga cultivadora realiza el proceso de mejora realizando los pasos 4 y 5 de ACO.

Paso 9.2. Penalizar arcos dañinos para el cultivo. La hormiga cultivadora verifica los arcos que generan distancias que superan significativamente la distancia de la mejor solución actual, y realiza un proceso de penalización para dichos arcos.

Paso 9.3. Se compara si la solución mejorada en el cultivo es mejor a la solución a global. La partición mejorada se une con las particiones inicio y fin. Si esta nueva solución es mejor, se actualiza la nueva solución global, de lo contrario el cultivo mantiene la solución global actual.

Paso 9.4. En este paso se actualiza la memoria de la feromona global., Con esto se logra que arcos identificados como perjudiciales resulten menos atractivos, lo que fomenta una rápida convergencia de la solución.

Paso 10. Evaporar la feromona. Permite evitar la convergencia hacia óptimos locales, decrece la feromona depositada por las hormigas exploradoras y cultivadoras.

Paso 11. Mostrar mejor solución global**:** Se selecciona la mejor solución de todo el proceso.

3.3. DESARROLLO DEL ALGORITMO

A partir del proceso descrito anteriormente, se procede a realizar la estructura del algoritmo mediante la representación estática que ofrece el diagrama de clases; se usó como base el repositorio de licencia BSD llamado *Swarmlib*, que provee varios algoritmos de optimización de tipo enjambres, entre ellos ACO. Se realiza una revisión de la implementación de ACO del repositorio, se realiza una modificación del algoritmo debido a que no se adaptaba en conceptos al ACO original, por lo cual, se realizan cambios en definición del problema y en la construcción de la solución mejorando a nivel conceptual Swarmlib-ACO y a partir de esta mejora se procede a realizar el desarrollo del algoritmo *ACO-Atta*, que implemente las características de las hormigas Atta.

Se definen las siguientes clases. La clase *Tsp_problem* crea una instancia del problema convirtiendo en grafo un archivo TSP de la librería TSPLIB, obteniendo información de nodos y arcos, además, provee la configuración necesaria para alojar la feromona y calcula la distancia entre los nodos. La clase *AntScout*, define a la hormiga exploradora encargada de realizar el recorrido completo del problema TSP, tiene la inteligencia para seleccionar el

siguiente nodo identificando si ya paso por este o no, deposita la feromona y realiza la optimización local con 2-opt [65].

La clase *AntSorter*, define a la hormiga seleccionadora encargada de obtener las mejores soluciones por cada iteración y realiza el proceso de poda usando una funcionalidad de la clase *PartitionTSPProblem* que recorta tramos de la solución para realizar mejoras en el cultivo. La clase *AntFarmer* se encarga de recibir las soluciones de la hormiga seleccionadora y realiza el proceso de cultivo del hongo.

Las tres clases principales para ACO-Atta, son las clases ACO_Atta, ACOSolver y ACO_Atta_Fungus. *ACO_Atta* quien es la clase encargada de crear las castas (hormigas exploradoras, seleccionadora y cultivadora), realiza la construcción de la solución lanzando el proceso que permite realizar en la exploración y cultivo. Por otro lado, la clase *ACOSolver* es donde la hormiga cultivadora realiza el recorrido, crea la solución del problema en función de los nodos, arcos y distancias, además contiene el método que lanza las hormigas seleccionadoras. *AcoAttaFungus* es la clase donde la hormiga cultivadora realiza el proceso de cultivo de las mejores soluciones, por medio de dos funciones la primera es *solver_fungus* donde las hormigas cultivadoras tomas las soluciones particionadas por la hormiga seleccionadora para iniciar el proceso de cultivo. La segunda *grow_fungus* toma la información de la hormiga cultivadora, verificando a partir del primer y último nodo que arcos son buenos para el cultivo, penalizándolos si no son buenos y actualizando la feromona para evitar el estancamiento del algoritmo.

A continuación, se presenta el diagrama de clases que soporta el modelo obtenido del proceso de la figura 13.

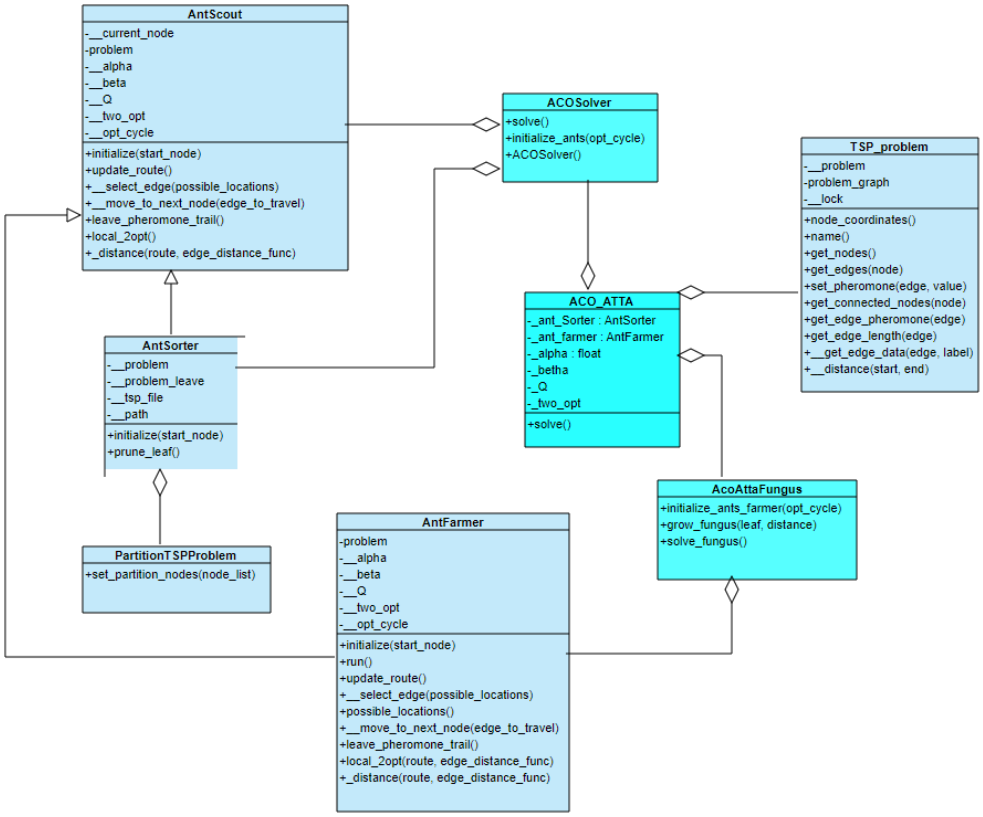

Figura 13. Diagrama de clases Aco-*Atta*.

3.4. PRUEBA DE CONCEPTO

La prueba de concepto permite verificar si la propuesta del nuevo algoritmo es viable, a partir del diagrama de clases se realiza la implementación del algoritmo usando Python en su versión 3.8 [66], apoyado en librerías como *threading* para el trabajo con múltiples hilos, la librería *tsplib95* que proporciona una lectura y escritura de archivos TSP, convirtiendo de forma rápida este tipo de archivos a grafos. Por otro lado, se utilizó un ambiente con equipo AMD-FX 9370 de 8 núcleos de 4.4 GHz con 32 GB de memoria RAM con sistema operativo Debian 8.

Se realizó una comparación entre ACO versus ACO-Atta, en 10 repeticiones 100 iteraciones cada una , tomando como instancia del problema TSP a Berlin52 que tiene un recorrido óptimo de 7542, y es una de las instancias de TSP usadas para pruebas [67]. Como parámetros para la pruebas se usaron los definidos en [68] como base para ACO y ACO-Atta, La tabla 9 muestra los parámetros usados.

Tabla 9. Configuración de parámetros ACO y ACO-Atta

Parámetros	valores
M	52, igual a la cantidad de ciudades del problema
N	100
α	1.0
β	2.0
ρ	0.05
q_0	0.9
M_s	1
M_f	7
N_f	2
α_f	1.0
β_f	2.0
ρ_f	0.05
q_{0f}	0.9

Para las pruebas se aplica el error relativo para indicar la calidad de la solución. La fórmula 5 indica la mejor solución encontrada de las 10 ejecuciones con respecto a la solución óptima, la fórmula 6 indica el promedio de las mejores soluciones de las 10 ejecuciones con respecto a la solución óptima y la fórmula 7 es el valor máximo encontrado de las 10 ejecuciones con respecto a la solución óptima.

$$\mathrm{ER}_{\mathrm{mejor}} = \frac{|\, mejor\ solución - solución\ óptima \,|}{solución\ óptima} \quad (5)$$

$$\mathrm{ER}_{\mathrm{avg}} = \frac{|\, promedio\ solución - solución\ óptima \,|}{solución\ óptima} \quad (6)$$

$$\mathrm{ER}_{\max} = \frac{|\, solución\ maxima - solución\ óptima\,|}{solución\ óptima} \quad (7)$$

Como resultado de las pruebas se identifica que ACO y ACO-Atta no alcanzan el óptimo reportado con los parámetros, sin embargo, ambos se acercan al óptimo reportando una desviación porcentual del 0.07%, como se observa en la tabla 10.

Tabla 10. Pruebas ACO y ACO-Atta para Berlin52.

	ACO-Atta			**ACO**		
1	7547			7615		
2	7566			7707		
3	7548			7653		
4	7597			7699		
5	7657			7655		
6	7584			7716		
7	7674			7674		
8	7657			7691		
9	7689			7719		
10	7662			7547		
	AVG	**MIN**	**MAX**	**AVG**	**MIN**	**MAX**
	7618,1	7547	7689	7667,6	7547	7719
	ER_{avg}	**ER_{mejor}**	**ER_{max}**	**ER_{avg}**	**ER_{mejor}**	**ER_{max}**
	1,01%	0,07%	1,95%	1,67%	0,07%	2,35%

Por otro lado, ACO-Atta encuentra mejores resultados que ACO, esto se puede observar en la tabla 10, ya que en promedio ACO-Atta reporta un error relativo del 1.01% a diferencia de ACO que reporta un error relativo del 1.67%. Se estudió la posibilidad de afinar parámetros para mejorar los resultados obtenidos, como alternativa de afinar parámetros se decide usar *hyperOpt* [69], que es una librería de Python para optimización Gaussiana en serie y en paralelo donde se define un espacio de búsqueda entre valores discretos, reales o condicionales.

Tabla 11. Configuración de HyperOpt para parámetros ACO

Parámetros ACO	**Función HyperOpt**	**Valores**
M	choice	{50, 60, 70, 80, 90, 100}
N	uniform	[0.01..2.0]
α	choice	{1, 2,…, 10}
β	uniform	[0.01..1.0]
ρ	uniform	[0.7..1.0]

HyperOpt identifica que configuración es necesaria para llegar a los parámetros óptimos que mejoren el proceso de ACO-*Atta*. *HyperOpt* provee funciones para apoyar la optimización de los parámetros, se usaron las funciones como *uniform* que retorna un valor con muestreo uniforme en el rango de valores definido. La función *choice* regresa un valor entre las opciones de una lista. Las tablas 11 y 12 muestran la configuración de los espacios definidos para los parámetros para ACO y ACO-Atta respectivamente.

Tabla 12. Configuración de HyperOpt para parámetros ACO-Atta

Parámetros	Función HyperOpt	valores
M	choice	{50, 60, 70, 80, 90, 100}
α	uniform	[0.01..2.0]
β	choice	{1, 2, …, 10}
ρ	uniform	[0.01..,.0]
q_0	uniform	[0.7..,.0]
M_s	choice	{1,2}
M_f	choice	$\{7 + i * 4 : i \in [0,..,11]\}$
α_f	uniform	[0.01..2.0]
β_f	choice	{1, 2, …, 10}
ρ_f	uniform	[0.01..1.0]
q_{0f}	uniform	[0.7..1.0]

Tabla 13. Parámetros mejorados por hyperOpt para ACO

Parámetros ACO	Valores
M	100
α	1.2966171825951867
β	1
ρ	0.4310868857634383
q_0	0.9439164375920963

Se realiza la ejecución de las configuraciones para obtener los parámetros con *hyperOpt*, ejecutándose sobre la Berlin52, obteniendo como resultados los parámetros mejorados que se muestran en la tabla 13 y 14 para ACO y ACO-Atta respectivamente.

Después de obtener los parámetros, se procede a ejecutan los algoritmos de ACO y ACO-Atta con las configuraciones de los parámetros mejoradas por *hyperOpt*, donde se evidencia que ACO-Atta alcanza mejores resultados que ACO. Los resultados se exponen en la siguiente sección.

Tabla 14. Parámetros mejorados por hyperOpt para ACO-Atta

Parámetros	Valores
M	60
α	1.1695656061998494
β	1
ρ	0.5115302498979079
q_0	0.7627058987132872
M_s	2
M_f	47
α_f	0.7052061739249137
β_f	6
ρ_f	0.4821555611187698
q_{0f}	0.7052330592158559

3.5. DISCUSIÓN DE RESULTADOS

Para las pruebas con los parámetros optimizados se usaron dos instancias de TSPLIB, Bayg29 con un óptimo de 1610 y berlin52 con un óptimo de 7542, con el fin de verificar el comportamiento de los algoritmos ACO y ACO-Atta. En la tabla 15 se presentan en forma general los resultados de las 10 ejecuciones.

Tabla 15. Resultados de ejecución Aco Versus ACO- Atta

		ACO_Atta			**ACO**		
TSP instance	Óptimo	AVG	MIN	MAX	AVG	MIN	MAX
bayg29	1610	1623,2	1610	1632	1625,9	1622	1634
Error		0,82%	0,00%	1,37%	0,99%	0,75%	1,49%
berlin52	7542	7600,1	7542	7658	7596,3	7542	7776
Error		0,77%	0,00%	1,54%	0,72%	0,00%	3,10%

En la Tabla 15 se observa que la instancia bayg29, para el caso de ACO no alcanza el óptimo, presentando un error del 0.75%, mientras que ACO-Atta alcanza el óptimo reportado en TSPLIB, además se puede observar que en promedio ACO-Atta obtiene mejores resultados con respecto a ACO. Para la instancia Berlin52, tanto ACO-Atta como ACO, llegan al óptimo reportado con los nuevos parámetros, pero, ACO-Atta obtiene resultados máximos relativamente menores a los encontrados por ACO, reportando un error del 1.54% para ACO-Atta con respecto al 3.10% que obtiene ACO.

A continuación, en la tabla 16, se presentan los 10 resultados a detalle de Bayg29 para las ejecuciones realizadas para ACO y ACO-Atta. Como se indicó con anterioridad ACO-Atta alcanza el óptimo de 1610, además de alcanzar resultados menores a los presentados por ACO, donde el óptimo máximo en las 10 ejecuciones para ACO es de 1634 comparado al de ACO-Atta de 1632.

Tabla 16. Resultados de ejecución Aco Versus ACO- Atta instancia bayg29

Ejecución	ACO-Atta			ACO		
1	**1610**			1634		
2	**1624**			**1624**		
3	**1624**			1631		
4	1627			1625		
5	**1624**			**1624**		
6	**1615**			**1624**		
7	1632			**1622**		
8	**1624**			**1624**		
9	**1624**			**1624**		
10	1628			1627		
	AVG	MIN	MAX	AVG	MIN	MAX
	1623,2	1610	1632	1625,9	1622	1634
	ER_{avg}	**ER_{mejor}**	**ER_{max}**	**ER_{avg}**	**ER_{mejor}**	**ER_{max}**
	0,82%	0,00%	1,37%	0,99%	0,75%	1,49%

Se selecciona el mejor resultado de las 10 repeticiones para ACO- Atta y para ACO, de la instancia bayg29, se grafican sus resultados para comparar la evolución de cada algoritmo en la solución del problema. En la figura 14 para bayg29, ACO-Atta alcanza el óptimo en la iteración 60, se puede observar que ACO- Atta converge mas rápido, mientras que ACO no alcanza el óptimo.

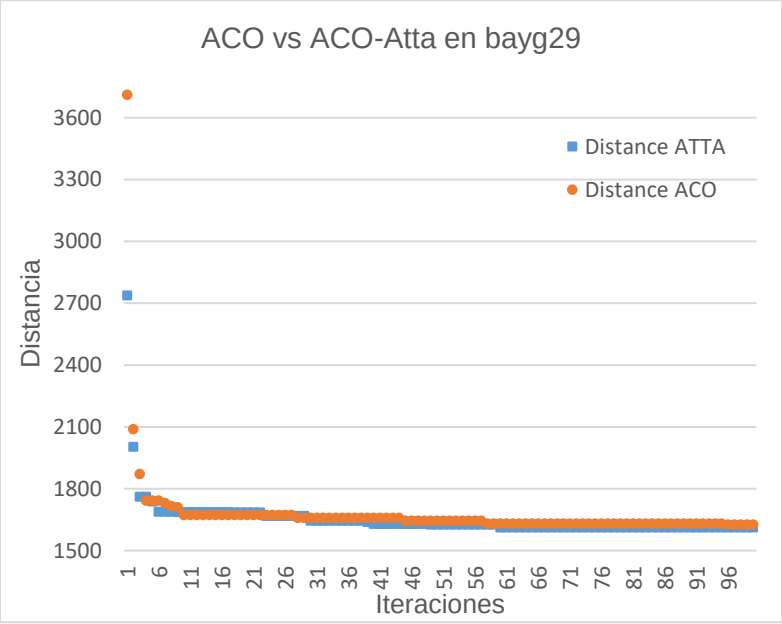

Figura 14. ACO-ATTA Vs ACO para bayg29

Para la instancia berlin52, ACO-Atta y ACO alcanzan el óptimo reportado de 7542, como se observa en la tabla 17, sin embargo en promedio ACO se comporta algo mejor que ACO-Atta con una diferencia del error promedio ($ER_{Avg)}$ para ACO del 0.72 con respecto a ACO-Atta del 0.77%. Por otro lado ACO-Atta mantiene mejores soluciones, como se observa en en la solución MAX donde obtiene un valor 7658 contra ACO con 7776, que en su error promedio maximo ($ER_{max)}$ representa para ACO-Atta un 1.54% contra ACO con 3.10%. Además, ACO-Atta alcanza en 3 ocasiones la mejor solución en la s 10 ejecuciones a diferencia de ACO, que solo lo alcanza una sola vez.

Tabla 17. Resultados de ejecución ACO Versus ACO- Atta instancia Berlin52

Ejecución	ACO-Atta			ACO		
1	7657			7543		
2	7657			7548		
3	7657			7658		
4	**7542**			7547		
5	7658			7548		
6	**7542**			7543		
7	7543			7542		
8	7548			**7542**		
9	**7542**			7716		
10	7655			7776		
	AVG	MIN	MAX	AVG	MIN	MAX
	7600,1	7542	7658	7596,3	7542	7776
	ER_{avg}	**ER_{mejor}**	**ER_{max}**	**ER_{avg}**	**ER_{mejor}**	**ER_{max}**
	0,77%	0,00%	1,54%	0,72%	0,00%	3,10%

En la figura 15, se selecciona la mejor solución de las 10 repeticiones se puede observar que ACO-Atta converge mas rápido que ACO; en este caso ACO- Atta alcanza el óptimo en a iteración 16 mientras mientras que ACO la alcanza en la iteración 44.

La tabla 18 muestra los resultados de ACO-Atta vs ACO para la mejor ejecución, donde solo se muestra las iteraciones en las que cambia la solución global. En este comparativo se puede verificar que ACO-Atta iteración a iteración obtiene mejores resultados que ACO.

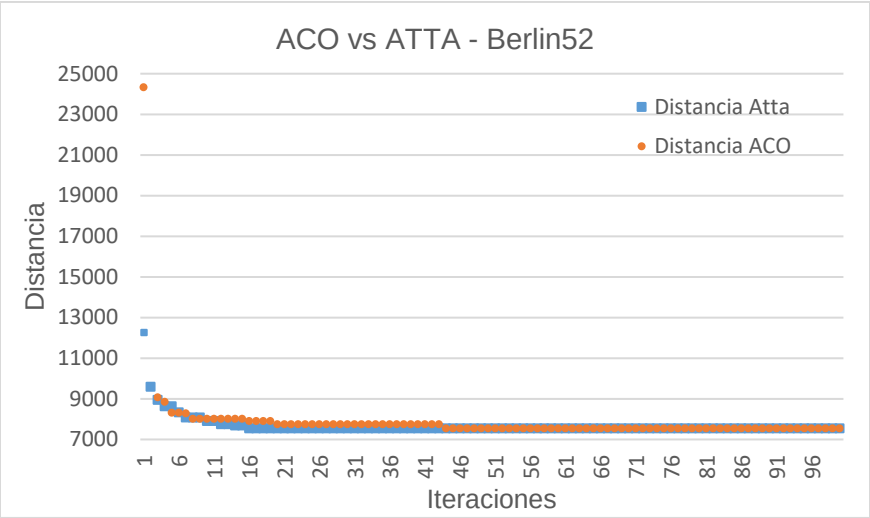

Figura 15. ACO- Atta Vs ACO para Berlin52

Tabla 18. Resultados mejor ejecución ACO Versus ACO- Atta berlin52

Iteración	Solución ACO-Atta	Solución ACO
1	**12261**	**24326**
2	9597	11594
3	8943	8978
6	8338	8193
7	**8082**	**8193**
10	**7913**	**8108**
12	7751	8108
16	**7542**	**7706**
21	7542	7547
44	**7542**	**7542**

En la figura 16, se grafican las iteraciones 1, 7, 16 y 44, de la mejor ejecución de ACO y ACO-Atta en la instancia berlin52, donde se puede observar cómo iteración a iteración ACO-Atta obtiene mejores resultados que ACO. En la figura 16.A se muestra la iteración 1, se puede observar que ACO se comporta un poco errático obteniendo un óptimo de 24326 a diferencia de ACO-Atta qué empieza a obtener una mejor solución entre tramos, reportando una solución global de 12261.

En la figura 16.B se muestra la iteración 7, mostrando un mejor comportamiento para ACO obteniendo una solución de 8193, sin embargo, ACO-Atta se comporta de mejor forma con una solución de 8082. La figura 16.C ACO-Atta obtiene la solución reportada en TSPLIB en la iteración 16, mientras que ACO en esta misma iteración alcanza una solución global de 7706. Finalmente, la figura 16.D muestra la iteración 44 donde ACO alcanza la solución reportada en TSPLIB, con una diferencia de 28 iteraciones después de que ACO-Atta encontrara el mismo resultado.

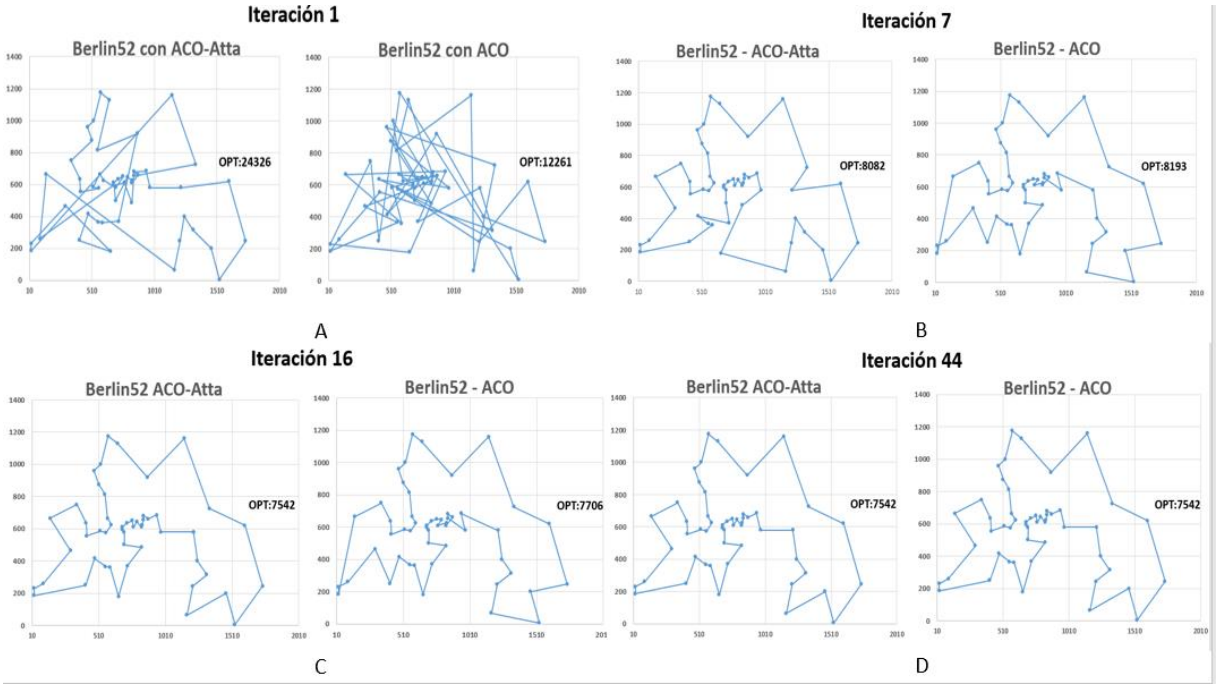

Figura 16. Mejor resultado de ACO-Atta Vs ACO para las iteraciones 1,7, 16 y 44 para berlin52

Aco-Atta obtiene buenos resultados, a continuación, las figuras 17, 18 y 19 muestran el sistema de defensa que aplica para algunos de los tramos mas largos de una ruta particionada el cual es penalizado para que dicho tramo no sea atractivo y genere mejores soluciones e la siguiente iteración.

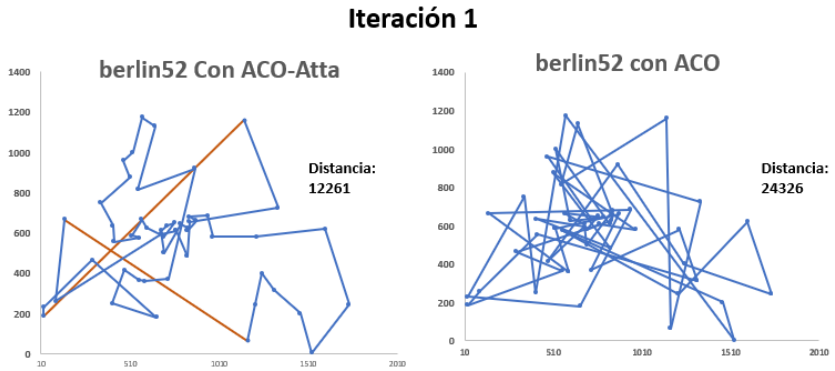

Figura 17. Mejor resultado de ACO-Atta Vs ACO para las iteraciones 1, penalizando tramos.

Como se observa en la figura 17, ACO-Atta, penaliza los tramos de color naranja haciendo que sean menos atractivos para siguiente iteración, la hormiga cultivadora pasa la información heurística para actualizar la feromona global y garantizar la selección tramos más cortos. En la figura 18, para ACO-Atta se muestra que estos tramos ya nos son seleccionados, y se generan nuevos tramos largo, los cuales serán penalizados y que sean

menos tractivos, por otro lado, para ACO tiene tramos largos lo que hace que su convergencia sea lenta.

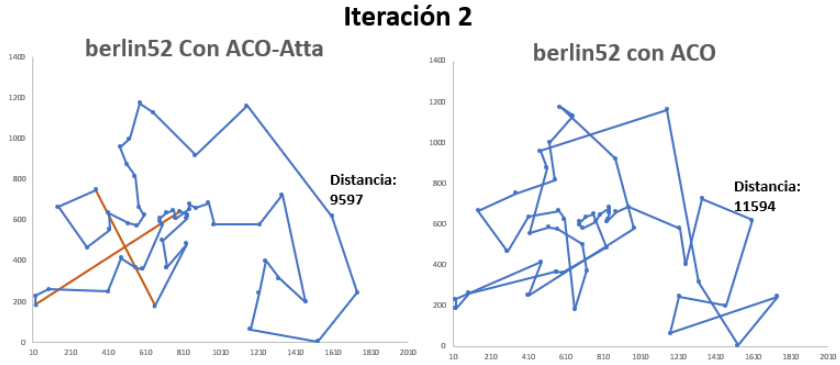

Figura 18. Mejor resultado de ACO-Atta Vs ACO para las iteraciones 2, penalizando tramos.

Finalmente, en la figura 19, se puede observar como para ACO-Atta ha solucionado la instancia de berlin52, obtenido tramos más cortos a diferencia de ACO.

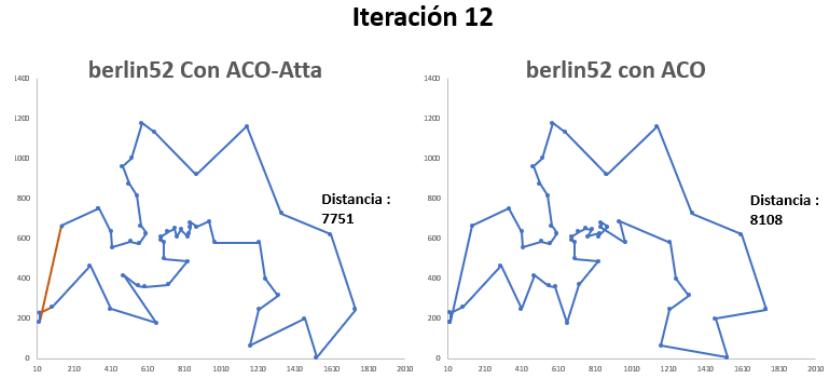

Figura 19. Mejor resultado de ACO-Atta Vs ACO para las iteraciones 12, penalizando tramos.

La tabla 19, muestra las ejecuciones por hormiga para ACO y ACO-Atta, donde se identifica cuantas computaciones por hormigas son necesarias para alcanzar la mejor solución reportada en TSPLIB. Para el caso de la instancia eil51, ACO-Atta es 10% mejor que ACO, lo que significa que ACO-Atta necesita 10% menos ejecuciones de hormiga que ACO para encontrar la mejor solución. Para la instancia bayg29, ACO-Atta es 8% mejor que ACO, lo que significa que ACO-Atta necesita 8% menos ejecuciones de hormiga que ACO para encontrar la mejor solución y para la instancia berlin52, , ACO-Atta es 44% mejor que ACO, lo que significa que ACO-Atta necesita 44% menos ejecuciones de hormiga que ACO para encontrar la mejor solución reportada en TSPLIB.

Tabla 19. Ejecuciones por hormiga ACO Vs ACO-Atta Para las instancias TSP de prueba**.**

	ACO	**ACO-Atta**
Hormigas por ejecución	$Cantidad\ de\ \mathrm{h}ormigas\ \times iteraci\mathrm{o}nes$	$(\mathrm{h}ormigas\ \mathrm{exploradoras}\ \times iteraci\mathrm{o}nes + (\mathrm{h}ormigas\ \mathrm{cultivadoras} \times 2) \times iteraci\mathrm{o}nes$
berlin52	$100\ \times 44 = 4400$	$(60 \times 16) + (47 \times 2) \times 16 =\ 2464$
	ACO-Atta 44% Mejor que ACO	
bayg29	$100 \times 100 = 10000$	$(60 \times 60) + (47 \times 2) \times 60 = 9240$
	ACO-Atta 8% Mejor que ACO	
eil51	$100 \times 58 = 5800$	$(60 \times 34) + (47 \times 2) \times 34 = 5236$
	ACO-Atta 10% Mejor que ACO	

3.6. CONCLUSIONES

El algoritmo denominado ACO-Atta fue diseñado a partir de las características de las hormigas *Atta* o arrieras, explotando la capacidad de estas para mejorar los cultivos de hongos, para el caso del algoritmo proporcionando las características necesarias para mejorar las soluciones por medio de las castas de hormigas y el concepto de hongo.

Como resultado se obtiene:

- El diseño de algoritmo con características de hormigas Atta, donde se incluye el concepto de hormiga seleccionadora para mejorar la exploración de las soluciones, evitando tramos poco prometedores para mejorar la solución. Adicional la hormiga cultivadora y el cultivo del hongo para mejorar explotación de la solución, que mejoran las soluciones globales obtenidas.

- La descripción del proceso del algoritmo partiendo del ACO tradicional, donde se mostraron 3 procesos la exploración, selección y realizar cultivo.

- Se indica el paso a paso de cómo se debe implementar el algoritmo ACO-*Atta*, entre las más importante se identifican cuáles son los parámetros necesarios para la buena ejecución del algoritmo, proceso de crear castas, construcción de la solución, proceso de selección, proceso de cultivo y evaporación de feromona.

- Implementación del algoritmo ACO-Atta – Anexo 2.

- Se realizan pruebas de concepto de ACO-*Atta* donde se ejecuta el algoritmo 10 veces con 10 iteraciones, esta fase permitió realizar ajustes de parámetros para mejorar los resultados obtenidos, se hace uso de *hyperOpt*, que obtiene los parámetros óptimos de una instancia, de esta forma mejoran los resultados del algoritmo para la instancia de prueba.

En conclusión, el diseño del algoritmo ACO-Atta obtiene mejores resultados en las instancias de prueba con respecto a ACO, consiguiendo mejores soluciones desde la primera iteración, esto se debe al proceso de cultivo, donde después de la selección de las mejores soluciones obtenidas por las hormigas exploradoras, se procede a optimizar estas soluciones en el cultivo. Además, ACO-Atta converge más rápido que ACO encontrando mejores soluciones en pocas iteraciones.

4. ANÁLISIS DE RESULTADOS

Este capítulo se centra en analizar los resultados obtenidos de una serie de experimentos realizados sobre 12 instancias de TSPLIB con el algoritmo propuesto y los resultados publicados en [70]. La prueba consiste en comparar los resultados obtenidos con ACO-Atta, contra el algoritmo base (ACO) y otros algoritmos en TSPLIB.

4.1. EJECUCIÓN DE INSTANCIAS

4.1.1. INSTANCIAS TSPLIB DE PRUEBAS

La tabla 20 muestra las 12 instancias de prueba, donde la primera columna muestra las instancias, la segunda muestra la cantidad de nodos o ciudades que contiene cada instancia y la tercera columna muestra el óptimo reportado en [64].

Tabla 20. Conjunto de Instancias TSP para pruebas ACO-Atta

	Instancia	Nodos	Óptimo
1	bayg29	29	1610
2	eil51	51	426
3	berlin52	52	7542
4	eil76	76	538
5	rat99	99	1211
6	rad100	100	7910
7	eil101	101	629
8	lin105	105	14379
9	ch130	130	6110
10	pre136	136	96772
11	u159	159	42080
12	kroA200	200	29368

4.2. RESULTADO DE PRUEBAS TSPLIB.

Se ejecutan ACO-Atta para cada una de las instancias, se mantienen las mismas condiciones presentadas en el capítulo 3 de la prueba de concepto, con los parámetros mejorados con *hyperOpt* de las tablas 13 y 14; realizando 10 ejecuciones de 100 iteraciones para cada instancia.

La tabla 21 muestra los resultados de las ejecuciones de ACO-Atta para las 12 instancias, donde la columna **Mejor** es el valor mínimo obtenido por el algoritmo en las 10 ejecuciones, la columna **Avg** se refiere al promedio de los mejores resultados de las 10 ejecuciones, la columna $\mathbf{ER_{mejor}}$ es el error relativo de la columna mejor con respecto a la solución óptima y la columna $\mathbf{ER_{Avg}}$ el error relativo del de la columna Avg con respecto a la solución óptima.

Tabla 21. Resultados ejecución de las 12 instancias con ACO-Atta

Instancia	óptimo	ACO-Atta			
		Mejor	Avg	ER_{mejor}	ER_{avg}
bayg29	1610	1610	1619,8	0,00%	0,61%
eil51	426	429	434,2	0,70%	1,92%
berlin52	7542	7542	7557,4	0,00%	0,20%
eil76	538	547	556,8	1,67%	3,49%
rat99	1211	1253	1272	3,47%	5,04%
rad100	7910	8009	8272	1,25%	4,58%
eil101	629	666	684	5,88%	8,74%
lin105	14379	14452	14564,6	0,51%	1,29%
ch130	6110	6261	6491,5	2,47%	6,24%
pre136	96772	103876	107140,1	7,34%	10,71%
u159	42080	43281	44075	2,85%	4,74%
kroA200	29368	31604	32520,8	7,61%	10,74%

Para las instancias bayg29 y berlin52, ACO-Atta obtiene las soluciones reportadas en TSPLIB como se observó en el capítulo 3, en el promedio de las 10 ejecuciones obtiene un ER_{avg} del 0,61% y 0.20% respectivamente, esto indica que ACO-Atta obtiene soluciones de buena calidad para estas instancias, La figura 20(a) y 20(b) muestra la gráfica de la mejor ruta encontrada por ACO-Atta para estas instancias.

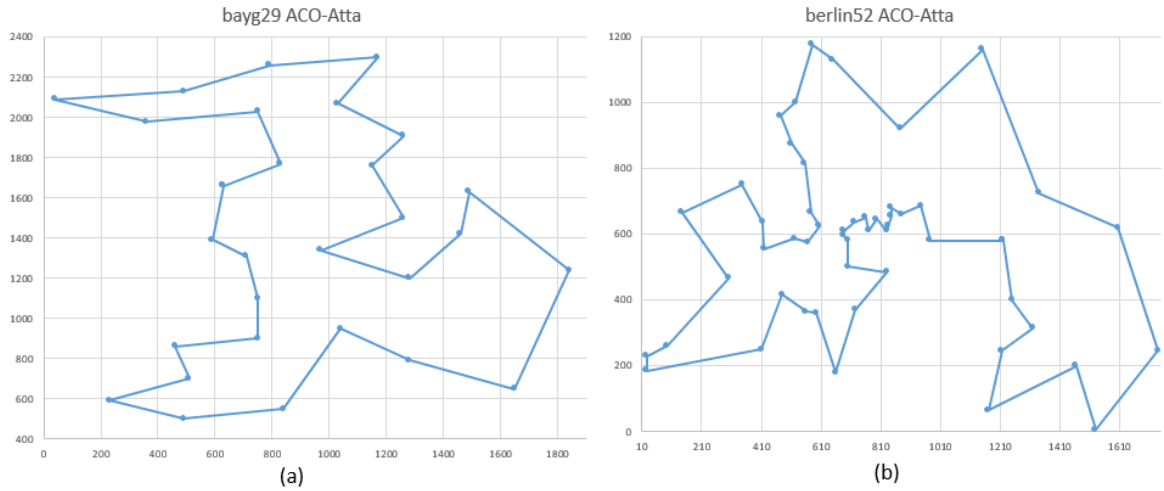

Figura 20. Mejor ruta encontrada para bayg29 (a) y berlin52(b) con ACO-Atta.

Para las instancias eil51 y lin105, ACO-Atta obtienen resultados donde se reporta un ER_{mejor} de menos del 1% acercándose a la mejor solución registrada. Para la instancia eil51, ACO-Atta obtiene una solución mínima de 429 acerándose a la solución global reportada en TSPLIB de 426, que equivale a un ER_{mejor} del 0.70% y con un ER_{avg} con un 1.92%.

Para la instancia lin105, ACO-Atta obtiene una solución mínima de 14452 respeto a la mejor solución reportada en TSPLIB de 14379, presentando ER_{mejor} del 0.51%, y un ER_{avg} del 1.92%. En la figura 21(a) y 21(b) muestra las gráficas de la mejor ruta encontrada por ACO para las instancias eil51 y lin105.

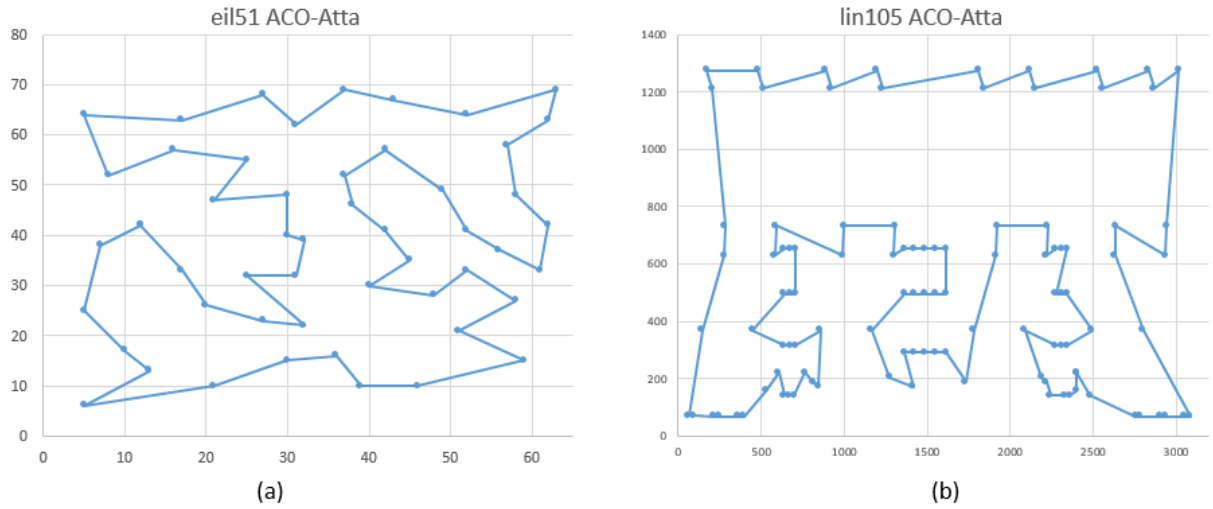

Figura 21. Mejor ruta encontrada para eil51 (a) y lin105(b) con ACO-Atta.

Para las instancias eil76 y rad100, ACO-Atta presenta un ER_{mejor} entre el 1% y 2%, para eil76, ACO-Atta reporta una solución mínima de 547 contra la mejor solución reportada en TSPLIB de 538, esto representa un ER_{mejor} de 1.67% y un ER_{avg} de 5.4%. Para la instancia rad100, ACO-Atta obtiene una solución mínima de 8009 contra la mejor solución reportada en TSPLIB de 7910, que equivale a un ER_{mejor} del 1.25% y un ER_{avg} de 4.58%. La figura 22 (a) y 22 (b), muestra las gráficas de la mejor ruta encontrada por ACO-Atta para las instancias eil75 y rad100.

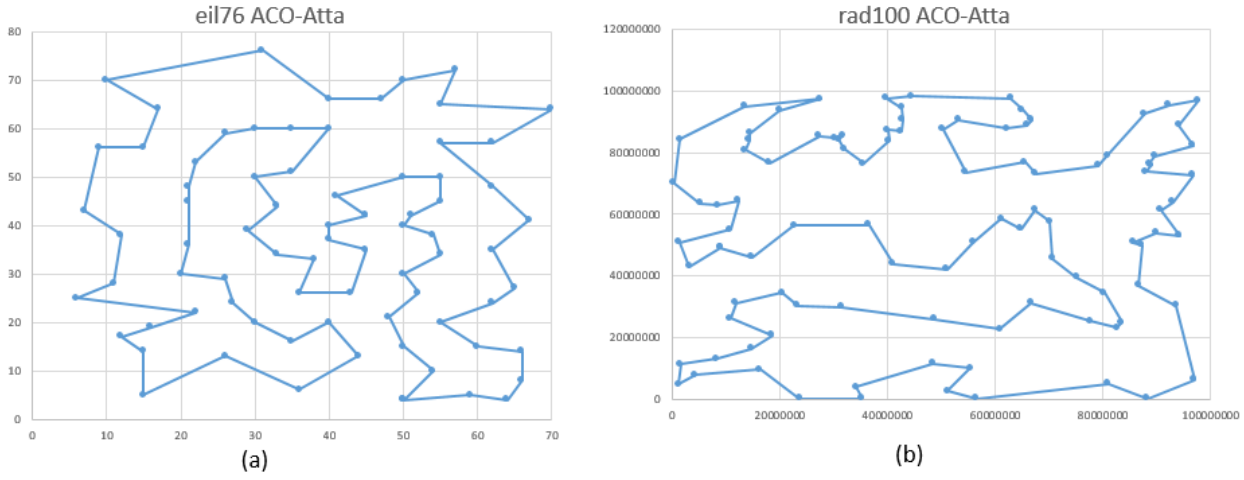

Figura 22. Mejor ruta encontrada para eil76 (a) y rad105 (b) con ACO-Atta.

Para las instancias ch130 y u159, ACO-Atta presenta un ER_{mejor} entre 2% y 3%, para ch130, ACO-Atta obtiene una solución mínima de 6261 con respecto a la mejor solución reportada en TSPLIB de 6110, esto equivale a un ER_{mejor} de 2.47% y un ER_{avg} de 6.24%. Para la instancia u159, ACO-Atta obtiene una solución mínima de 43281 con respecto a la mejor solución reportada en TSPLIB de 42080, lo que equivale a un ER_{mejor} de 2.85% y un ER_{avg} de 4.74%. La figura 23 (a) y 23 (b), muestra las gráficas de la mejor ruta encontrada por ACO-Atta para ch130 y u159.

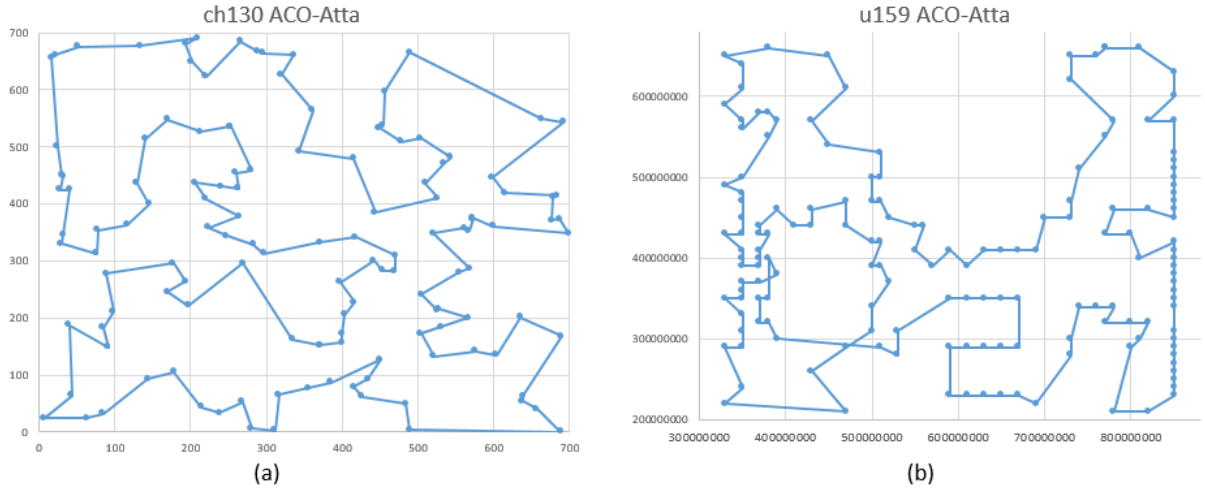

Figura 23. Mejor ruta encontrada para ch130 (a) y u159(b) con ACO-Atta

Para las instancias rat99 y eil101, ACO-Atta reportan un ER_{mejor} entre 3% y 6%, para rat99, ACO-Atta obtuvo una solución mínima de 1253 con respecto a la mejor solución reportada en TSPLIB de 1211, lo que equivale a un ER_{mejor} de 3.47% y un ER_{avg} de 5.04%. Para la instancia eil101, ACO-Atta presento una solución mínima de 666 con respecto a la mejor solución reportada en TSPLIB de 629, esto equivale a un ER_{mejor} de 5.88% y un promedio ER_{avg} de 8.74%. figura 24 (a) y 24 (b), muestra las gráficas de la mejor ruta encontrada por ACO-Atta para rat99 y eil101.

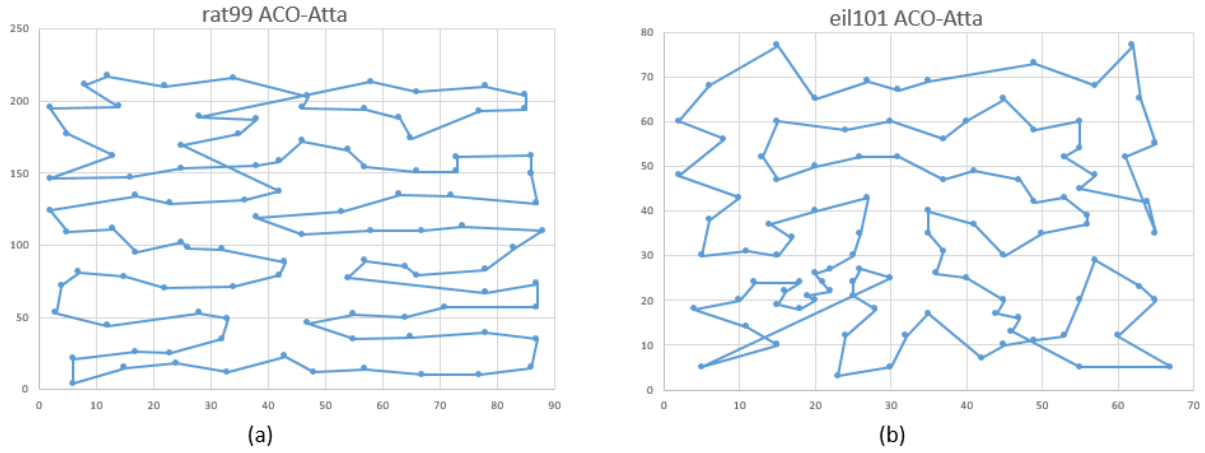

Figura 24. Mejor ruta encontrada para rat99 (a) y eil101 (b) con ACO-Atta

Por ultimo las instancias pr136 y kroA200, ACO-Atta presenta un ER_{mejor} superior del 6%, para la instancia pre136, ACO-Atta obtuvo una solución mínima de 103876 con respecto a la mejor solución reportada en TSPLIB de 96772, lo que equivale a un ER_{mejor} de 7.34% y un error promedio ER_{avg} de 10.71%. Ahora para la instancia kroA200, ACO-Atta presento una solución mínima de 31604 con respecto a la mejor solución reportada en TSPLIB de 29368, lo que equivale a un ER_{mejor} de 7.61% y un ER_{avg} de 10.74%. figura 25 (a) y 25 (b), muestra las gráficas de la mejor ruta encontrada por ACO-Atta para pre136 y kroa200.

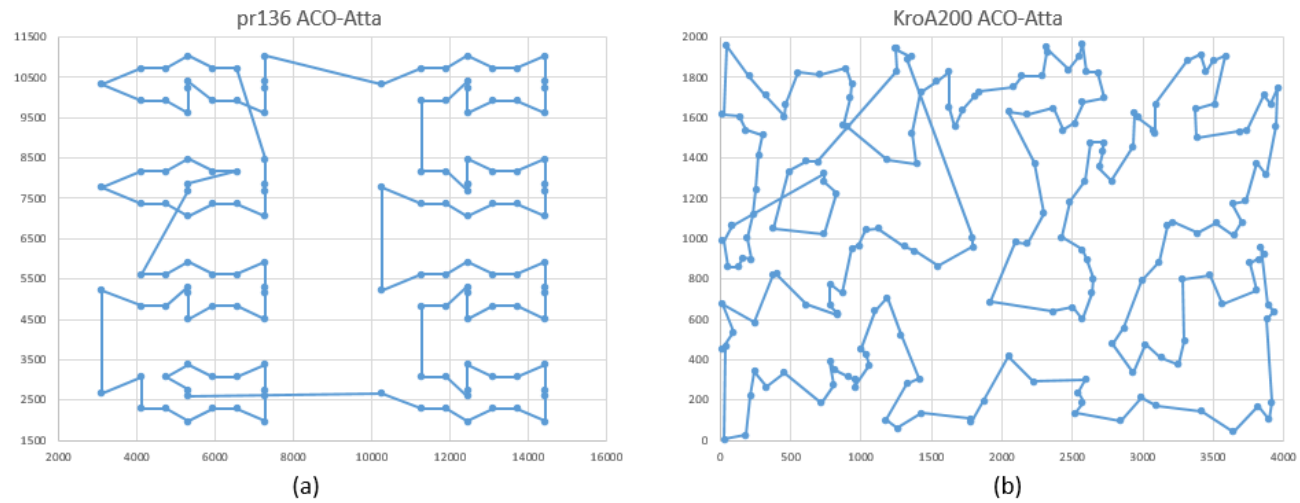

Figura 25. Mejor ruta encontrada para pre136 (a) y kroA200 (b) con ACO-Atta

A continuación, la tabla 22 muestra los puntos de las mejores rutas de obtenidas por ACO-Atta para cada una de las instancias seleccionadas.

Tabla 22. Mejores rutas obtenidas por ACO-Atta.

Instancia		Mejor	ACO-Atta Mejor Ruta
1	bayg29	1610	[2, 20, 10, 4, 15, 18, 14, 17, 22, 11, 19, 25, 7, 23, 8, 27, 16, 13, 24, 1, 28, 6, 12, 9, 26, 3, 29, 5, 21, 2]
2	eil51	429	[14, 25, 13, 41, 19, 40, 42, 44, 15, 45, 33, 39, 10, 30, 9, 49, 5, 38, 11, 32, 1, 22, 2, 16, 50, 34, 21, 29, 20, 35, 36, 3, 28, 31, 8, 26, 7, 43, 24, 23, 48, 6, 27, 51, 46, 12, 47, 37, 17, 4, 18, 14]
3	berlin52	7542	[12, 28, 27, 26, 47, 13, 14, 52, 11, 51, 33, 43, 10, 9, 8, 41, 19, 45, 32, 49, 1, 22, 31, 18, 3, 17, 21, 42, 7, 2, 30, 23, 20, 50, 29, 16, 46, 44, 34, 35, 36, 39, 40, 37, 38, 48, 24, 5, 15, 6, 4, 25, 12]
4	eil76	547	[69, 36, 47, 48, 29, 45, 27, 52, 34, 46, 8, 35, 7, 26, 67, 76, 75, 4, 30, 2, 68, 6, 51, 17, 40, 12, 58, 72, 39, 9, 32, 44, 3, 16, 63, 33, 73, 62, 28, 74, 21, 61, 22, 64, 42, 41, 43, 1, 56, 23, 49, 24, 18, 50, 25, 55, 31, 10, 38, 65, 66, 11, 59, 14, 53, 19, 54, 13, 57, 15, 5, 37, 20, 70, 60, 71, 69]
5	rat99	1253	[72, 81, 80, 71, 70, 69, 78, 77, 68, 67, 66, 65, 64, 74, 73, 82, 83, 91, 92, 93, 94, 95, 86, 87, 88, 79, 89, 90, 99, 98, 97, 96, 84, 85, 76, 75, 59, 58, 57, 56, 55, 46, 47, 38, 48, 39, 40, 41, 32, 31, 30, 29, 37, 28, 19, 20, 21, 22, 13, 12, 11, 10, 1, 2, 3, 4, 14, 5, 6, 7, 8, 9, 18, 17, 16, 15, 23, 24, 25, 26, 27, 36, 35, 33, 42, 43, 34, 44, 45, 54, 53, 52, 51, 50, 49, 60, 61, 62, 63, 72]

Instancia		Mejor	ACO-Atta Mejor Ruta
6	rad100	8009	[80, 81, 91, 64, 36, 57, 47, 99, 53, 79, 6, 94, 98, 66, 88, 84, 31, 16, 45, 52, 11, 61, 5, 22, 41, 34, 7, 42, 24, 25, 43, 40, 55, 96, 39, 100, 44, 29, 35, 23, 2, 89, 58, 76, 59, 95, 77, 93, 28, 37, 19, 56, 46, 50, 73, 54, 38, 70, 72, 17, 92, 27, 10, 33, 3, 4, 32, 78, 20, 74, 26, 9, 12, 14, 85, 82, 75, 21, 49, 13, 67, 97, 86, 63, 15, 69, 8, 60, 1, 62, 87, 18, 71, 68, 83, 51, 90, 65, 48, 30, 80]
7	eil101	666	[101, 53, 58, 40, 21, 73, 72, 74, 22, 75, 56, 4, 54, 80, 68, 12, 26, 28, 27, 69, 1, 50, 76, 77, 3, 79, 33, 81, 9, 51, 20, 66, 71, 65, 35, 34, 78, 29, 24, 55, 25, 39, 67, 23, 41, 15, 57, 2, 87, 42, 43, 38, 14, 44, 86, 16, 61, 85, 91, 100, 37, 98, 92, 59, 93, 99, 96, 95, 97, 13, 94, 6, 89, 52, 18, 83, 60, 5, 84, 17, 45, 8, 46, 47, 36, 49, 64, 63, 90, 32, 10, 62, 11, 19, 48, 82, 7, 88, 31, 70, 30, 101]
8	lin105	14452	[84, 83, 82, 78, 71, 68, 67, 64, 72, 77, 79, 86, 80, 76, 73, 81, 75, 74, 69, 70, 63, 62, 105, 59, 56, 55, 50, 48, 45, 49, 40, 104, 44, 47, 51, 54, 57, 58, 53, 52, 46, 43, 41, 42, 37, 36, 17, 16, 18, 25, 26, 27, 24, 19, 12, 20, 23, 28, 33, 32, 31, 30, 29, 22, 21, 103, 15, 11, 10, 7, 6, 2, 1, 3, 8, 9, 5, 4, 13, 14, 34, 35, 38, 39, 60, 61, 65, 66, 87, 88, 94, 95, 100, 99, 98, 90, 89, 93, 102, 101, 97, 96, 92, 91, 85, 84]
9	ch130	6261	[41, 39, 71, 130, 50, 2, 118, 80, 46, 20, 35, 54, 17, 31, 27, 19, 100, 15, 29, 24, 116, 95, 79, 12, 87, 81, 103, 77, 94, 89, 110, 98, 68, 48, 25, 113, 32, 36, 84, 119, 111, 123, 101, 82, 9, 57, 56, 65, 52, 75, 74, 99, 73, 92, 38, 106, 4, 44, 42, 51, 60, 120, 53, 58, 49, 72, 91, 6, 102, 55, 122, 10, 14, 96, 67, 13, 23, 40, 47, 22, 37, 93, 33, 21, 18, 108, 8, 126, 114, 3, 83, 30, 59, 121, 78, 90, 125, 85, 66, 28, 115, 62, 105, 112, 117, 128, 16, 45, 5, 11, 76, 109, 61, 129, 124, 64, 69, 86, 88, 7, 26, 97, 70, 63, 43, 104, 107, 127, 34, 1, 41]
10	pre136	103876	[89, 82, 93, 94, 111, 112, 123, 116, 104, 113, 114, 103, 115, 124, 133, 125, 126, 134, 127, 128, 122, 135, 121, 117, 105, 110, 109, 120, 118, 106, 108, 107, 119, 132, 131, 136, 130, 129, 100, 87, 83, 71, 76, 75, 72, 74, 73, 85, 98, 97, 99, 96, 95, 86, 84, 64, 63, 65, 62, 61, 66, 54, 50, 37, 42, 41, 38, 51, 53, 40, 39, 52, 18, 30, 5, 6, 1, 7, 8, 2, 16, 20, 32, 27, 28, 31, 29, 19, 17, 15, 9, 10, 3, 11, 12, 4, 13, 22, 34, 23, 24, 33, 21, 14, 25, 26, 43, 44, 55, 48, 36, 45, 46, 35, 47, 56, 68, 57, 58, 67, 59, 60, 49, 88, 77, 78, 70, 79, 80, 69, 81, 90, 102, 91, 92, 101, 89]
11	u159	43281	[58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 36, 37, 35, 34, 33, 32, 31, 30, 29, 28, 26, 27, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 11, 12, 10, 9, 8, 148, 147, 146, 145, 144, 143, 142, 141, 139, 140, 138, 137, 136, 135, 133, 134, 132, 131, 128, 127, 126, 125, 1, 159, 3, 2, 4, 5, 6, 153, 152, 7, 151, 150, 149, 155, 154, 156, 157, 158, 129, 124, 130, 123, 122, 121, 120, 119, 118, 117, 116, 115, 114, 113, 112, 111, 110, 109, 108, 93, 92, 91, 90, 89, 94, 88, 87, 96, 95, 107, 106, 105, 104, 103, 102, 101, 100, 99, 98, 97, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 62, 63, 61, 60, 59, 58]
12	KroaA200	31604	[51, 194, 122, 116, 188, 44, 63, 16, 118, 124, 138, 9, 78, 82, 7, 199, 26, 61, 136, 32, 24, 159, 174, 121, 172, 46, 12, 147, 40, 132, 85, 145, 191, 27, 198, 123, 15, 79, 160, 162, 64, 20, 55, 42, 135, 186, 127, 112, 120, 47, 31, 67, 177, 13, 65, 80, 77, 158, 167, 30, 68, 169, 35, 2, 181, 125, 161, 151, 187, 157, 107, 109, 6, 54, 75, 155, 183, 22, 134, 8, 17, 25, 143, 90, 34, 103, 146, 129, 114, 98, 58, 171, 141, 200, 148, 88, 28, 39, 38, 71, 130, 72, 83, 62, 185, 168, 173, 23, 144, 150, 91, 94, 95, 50, 139, 86, 5, 196, 178, 152, 56, 105, 43, 137, 133, 176, 113, 195, 182, 76, 70, 102, 154, 21, 140, 164, 89, 41, 59, 73, 3, 189, 131, 180, 142, 69, 108, 192, 14, 36, 57, 74, 156, 100, 33, 45, 197, 81, 97, 104, 165, 166, 96, 126, 87, 11, 52, 84, 170, 48, 153, 66, 119, 175, 10, 92, 19, 99, 149, 106, 93, 163, 4, 101, 60, 128, 193, 53, 1, 117, 115, 111, 29, 184, 37, 179, 190, 49, 18, 110, 51]

4.3. COMPARACIÓN CON DIFERENTES ALGORITMOS

Para validar la eficacia de ACO-Atta, se comprara con otros algoritmos, para ellos se toma como referencia los resultados presentados en [70], donde toman un conjunto de instancias simétricas de TSPLIB y realizan comparaciones entre diferentes implementaciones de ACO.

En las pruebas presentadas en [70], se realizaron 30 ejecuciones de 1000 iteraciones, mientras para ACO-Atta se realizaron 10 ejecuciones de 100 iteraciones, Como se mencionó en el capítulo 3 en el ítem de pruebas de concepto.

Las comparaciones se realizaron contra un ACO tradicional que es presentado en [70] para indicar si se la propuesta supera los resultados presentados para ACO. Por otro lado se usa Neuro-Immune [71] que es una metaheurística basada en una red neuronal con conceptos del sistema inmune ubicando cada ciudad como una celda para la red neuronal definiendo vecindades para establecer la ruta de TSP y HyMSPO [72], que es una metaheurística utilizando una mejora de optimización con enjambre de partículas con expansión de vecindades lo que permite mejorar los resultados, estas propuestas presentan técnicas que se utilizan en TSP los cuales son repetitivos en el estado del arte y vale la pena la comparación con ellos.

Por otro lado se compara con ACE [73], que es un algoritmo de optimización por colonia de hormigas extendido, que usa el concepto de castas, por una parte presentan la casta de las recolectoras que solo usan el concepto de depósito de feromona y la casta de exploradoras que pueden usar alguna heurística definida para mejorar la búsqueda, además ACE implementan una política de regulación para controlar el tipo de hormiga que creara durante el proceso de búsqueda, en este caso los resultados superan a ACO en varias ocasiones y MGACACO [70], es un hibrido entre la exploración de un algoritmo genético con la capacidad estocástica de ACO, además de utilizar el método de optimización caótica para evitar caer en un óptimo local para mejorar la convergencia del algoritmo, que supera los resultados de los algoritmos anteriormente mencionados.

Las configuraciones de los parámetros de ACO tradicional, Neuro-Immune, HyMSPO y ACE se toman de la referencia de MGACACO [70] y los parámetros de ACO-Atta de la tabla 13 y 14 del capítulo 3. Las configuraciones de los parámetros por cada algoritmo se muestran en la tabla 23.

Tabla 23. Parámetros de prueba ACO-Atta y MGACACO

Parámetros	**MAGCACO** [70]	**ACO-Atta**
N	1000	100
M	200	60
α	1	1.16956560619985
β	2	1
ρ	0.05	0.5115302498979079
q_0	0.1	0.76270589871329

Parámetros	**MAGCACO** [70]	**ACO-Atta**
M_s	-	1
M_f	-	47
α_f	-	0.70520617392491
β_f	-	6
ρ_f	-	0.48215556111877
q_{0f}	-	0.70523305921586

La tabla 24 muestra los mejores resultados obtenidos por cada uno de los algoritmos, para 11 instancias de TSPLIB y la tabla 25 muestra el cálculo de los errores relativos con respecto a los mejores resultados de cada algoritmo y al mejor resultado presentado en TSPLIB.

Tabla 24. Comparación de ACO-Atta con otros algoritmos.

Instancia	óptimo	**ACO** [70]		**Neuro-Immune** [71]		**HyMSPO** [72]		**ACE** [73]		**MGACACO** [70]		**ACO-Atta**	
		Mejor	Avg	Mejor	Avg	Mejor	Avg	Mejor	Avg	Mejor	Avg	Mejor	Avg
eil51	426	431,02	437,79	433	442,72	430,16	437,79	427,12	429,68	426	427,21	429	434,2
berlin52	7542	7584,36	7602,83	7544	8044	-	-	-	-	7544,37	7544,37	7542	7557,4
eil76	538	552,426	569,865	552	563,2	544,672	551,241	540,538	545,69	539,153	540,302	547	556,8
rat99	1211	1262,53	1294,32	-	-	-	-	-	-	1216,56	1223,86	1253	1272
rad100	7910	7986,3	8003,61	7937	8342,8	7935,46	7968,25	7912,65	7935,46	7910	7934,69	8009	8272
eil101	629	681,562	693,472	640,05	665,93	635,68	650,43	631,76	636,04	630,01	634,68	666	684
lin105	14379	14464,3	14483,8	14379	16031,3	14395	14414,5	14380,1	14393	14381,8	14394,1	14452	14564,6
ch130	6110	6148,32	6171,66	6203	6332,43	6179,03	6199,35	6112,42	6120,68	6113,26	6123,92	6261	6491,5
pre136	96772	96989,5	67279,3	-	-	-	-	-	-	96945,1	67142,6	103876	107140,1
u159	42080	42489,2	42503,9	-	-	-	-	-	-	42489,2	42413,5	43281	44075
KroA200	29368	29505,3	29521,8	30144	33228,3	29404,2	29476,3	29376,2	29402,9	29380,2	29434,4	31604	32520,8

Tabla 25. Comparación de ACO-Atta con otros algoritmos – cálculo de errores.

Instancia	óptimo	**ACO** [70]		**Neuro-Immune** [71]		**HyMSPO** [72]		**ACE** [73]		**MGACACO** [70]		**ACO-Atta**	
		ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}
eil51	426	1,178%	2,77%	1,64%	3,92%	0,98%	2,77%	0,26%	0,86%	0,0%	0,3%	0,70%	1,92%
berlin52	7542	0,562%	0,81%	0.03%	6.66%	-	-	-	-	0,0%	0,0%	0,00%	0,20%
eil76	538	2,681%	5,92%	2,60%	4,68%	1,24%	2,46%	0,47%	1,43%	0,2%	0,4%	1,67%	3,49%
rat99	1211	4,255%	6,88%	-	-	-	-	-	-	0,5%	1,1%	3,47%	5,04%
rad100	7910	0,965%	1,18%	0,34%	5,47%	0,32%	0,74%	0,03%	0,32%	0,0%	0,3%	1,25%	4,58%
eil101	629	8,356%	10,25%	1,76%	5,87%	1,06%	3,41%	0,44%	1,12%	0,2%	0,9%	5,88%	8,74%
lin105	14379	0,593%	0,73%	0,00%	11,49%	0,11%	0,25%	0,01%	0,10%	0,0%	0,1%	0,51%	1,29%

Instancia	óptimo	ACO [70]		Neuro-Immune [71]		HyMSPO [72]		ACE [73]		MGACACO [70]		ACO-Atta	
		ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}	ER_{mejor}	ER_{avg}
ch130	6110	0,627%	1,01%	1,52%	3,64%	1,13%	1,46%	0,04%	0,17%	0,1%	0,2%	2,47%	6,24%
pre136	96772	0,225%	0,52%	-	-	-	-	-	-	0,2%	0,4%	7,34%	10,71%
u159	42080	0,972%	1,01%	-	-	-	-	-	-	1,0%	0,8%	2,85%	4,74%
Kroa200	29368	0,468%	0,52%	2,64%	13,14%	0,12%	0,37%	0,03%	0,12%	0,0%	0,2%	7,61%	10,74%

Para la instancia eil51, se puede observar que ACO-Atta supera los resultados de ACO tradicional, Neuro-inmuner y HyMSPO. ACO-Atta obtiene una solución mínima de 429 mientras que ACO tradicional alcanza una solución mínima de 431.02 con un ER_{mejor} del 1.78% y con un ER_{avg} de 2.77%, Neuro-inmuner alcanza una solución mínima de 433 con un ER_{mejor} del 1.64% y con un ER_{avg} de 3.92% y HyMSPO alcanza una solución mínima de 430 con un ER_{mejor} del 0.98% y con un ER_{avg} de 2.77%. Sin embargo, ACO-Atta no logra superar a ACE y MGACACO, que obtienen mejores soluciones reportando un ER_{mejor} por debajo 0.3%. La figura 26 muestra el comparativo entre ACO-Atta y los demás algoritmos.

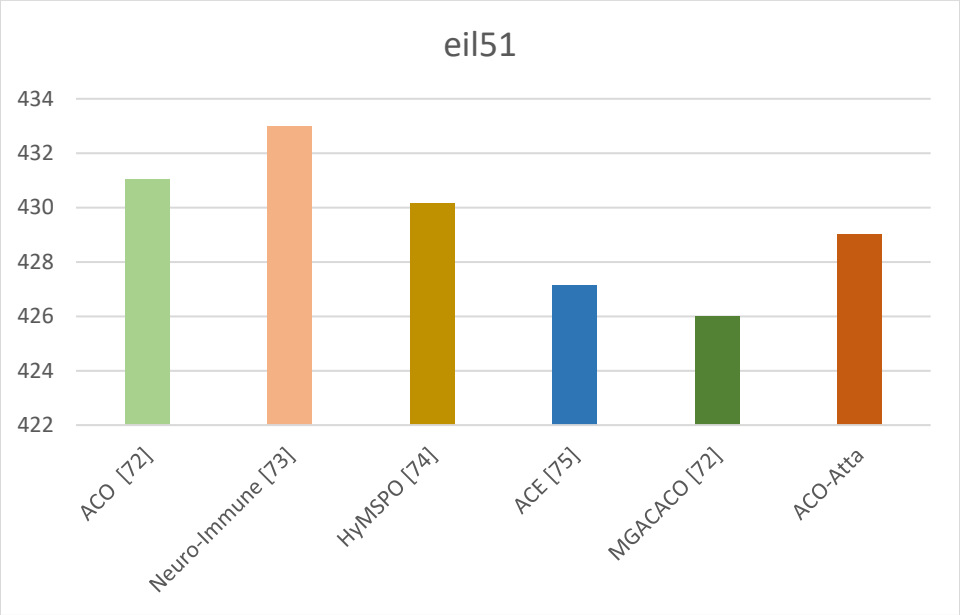

Figura 26. Comparación ACO-Atta con otros algoritmos para la instancia eil51.

Para la instancia berlin52, se puede observar que ACO-Atta supera los resultados de ACO tradicional, Neuro-inmuner y MGACACO. ACO-Atta obtiene la mejor solución reportada en TSPLIB de 7542, mientras que ACO tradicional alcanza una solución mínima de 7584.36 con un ER_{mejor} del 0.56% y con un ER_{avg} de 0.81%, Neuro-inmuner alcanza una solución mínima de 7544 con un ER_{mejor} del 0.03% y con un ER_{avg} de 6.662% y MGACACO alcanza una solución mínima de 7544.37 con un ER_{mejor} del 0.98% y con un ER_{avg} de 0.03%. La figura 27 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia berlin52.

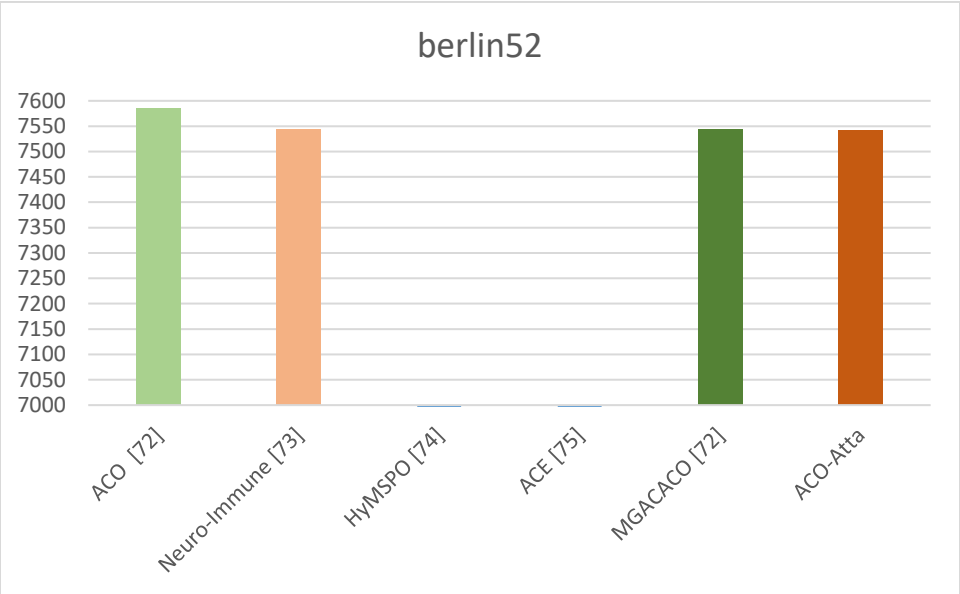

Figura 27. Comparación ACO-Atta con otros algoritmos para la instancia berlin52.

Para la instancia eil76, se observar que ACO-Atta supera los resultados de ACO tradicional y Neuro-inmuner, ACO-Atta obtiene una solución mínima de 547 mientras que ACO tradicional alcanza una solución mínima de 552.42 con un ER_{mejor} del 2.681% y con un ER_{avg} de 5.92%. Neuro-inmuner alcanza una solución mínima de 552 con un ER_{mejor} del 2.60 % y con un ER_{avg} de 4.68%. Sin embargo, ACO-Atta no logra superar a HyMSPO, ACE y MGACACO, que obtienen mejores soluciones reportando un ER_{mejor} por debajo 1.25%. La figura 28 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia eil76.

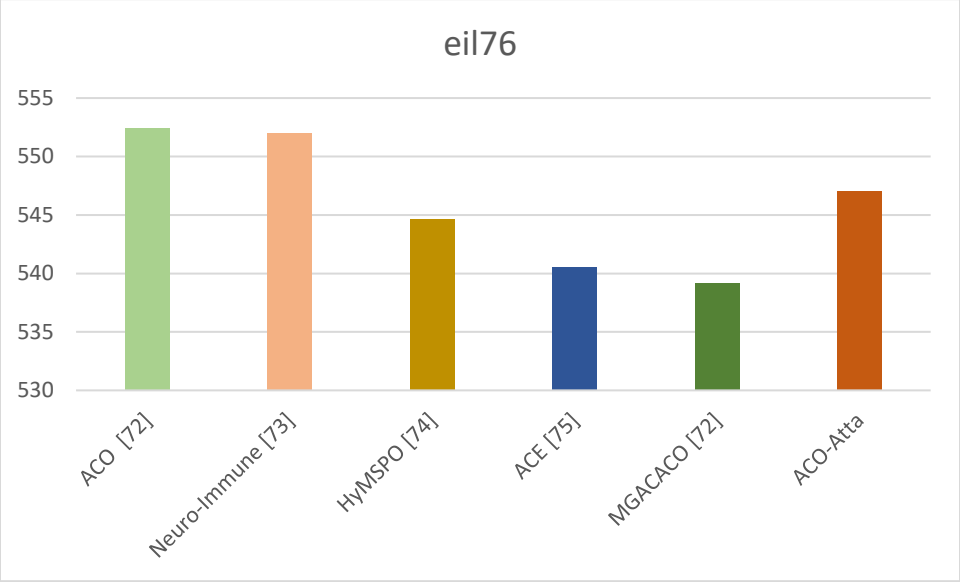

Figura 28. Comparación ACO-Atta con otros algoritmos para la instancia eil76.

Para la instancia rat99, se observa que ACO-Atta supera los resultados de ACO tradicional, obteniendo una solución mínima de 1553 mientras que ACO tradicional alcanza una

solución mínima de 1262.53 con un ER_{mejor} del 4.25% y con un ER_{avg} de 6.88%. Por otro lado, MGACACO supera a ACO-Atta, alcanzando una solución mínima de 1216.56 con un ER_{mejor} del 0.5 % y con un ER_{avg} de 1.1%. La figura 29 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia rat99.

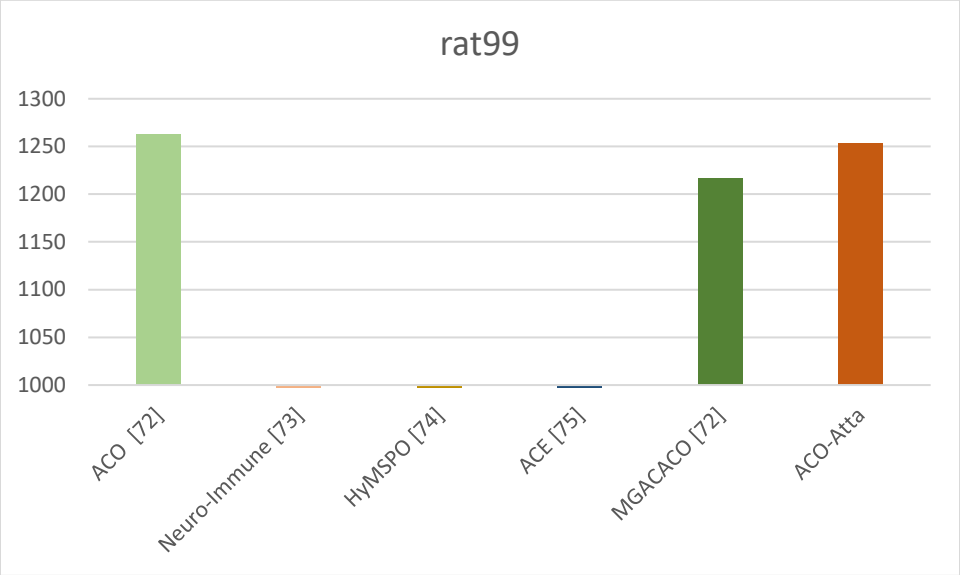

Figura 29. Comparación ACO-Atta con otros algoritmos para la instancia rat99.

Para la instancia rad100, se observa que ACO-Atta no logra supera los resultados de los demás algoritmos, en este caso ACO tradicional, obtuvo una solución mínima de 7986.3 con un ER_{mejor} del 0.96% y con un ER_{avg} de 1.18%, mientras que ACO-Atta alcanza una solución mínima de 8009. los demás algoritmos obtienen mejores resultados reportando ER_{mejor} menores del 0.35%, donde se destaca MGACACO que obtiene la solución reportada en TSPLIB de 7910. La figura 30 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia ras100.

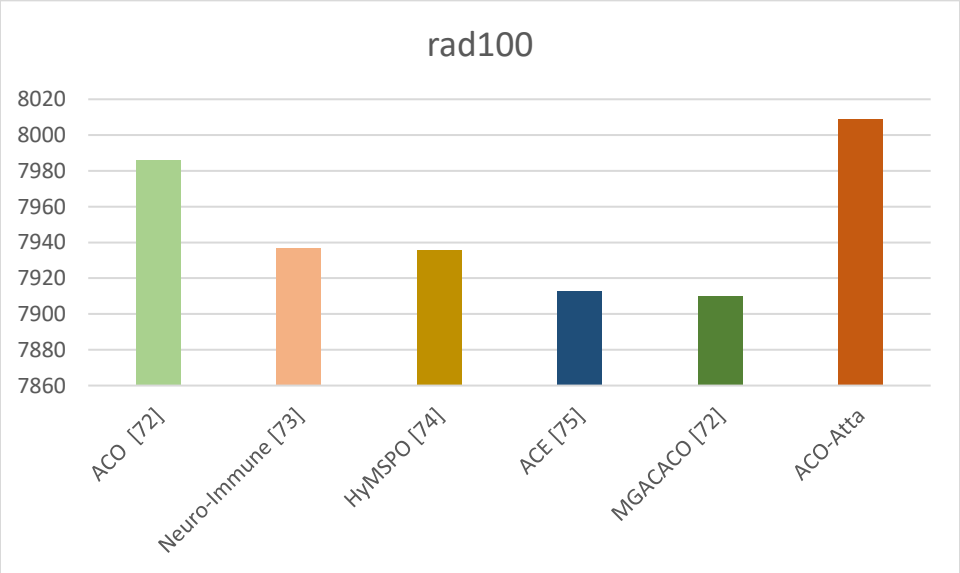

Figura 30. Comparación ACO-Atta con otros algoritmos para la instancia rad100.

Para la instancia eil101, se observa que ACO-Atta supera solo los resultados de ACO tradicional, obteniendo una solución mínima de 666 mientras que ACO tradicional alcanza una solución mínima de 681.56 con un ER_{mejor} del 8.356% y con un ER_{avg} de 10.25%. Además, se observa que los demás algoritmos superan a ACO-Atta, reportando ER_{mejor} menores al 1.80%. La figura 31 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia eil101.

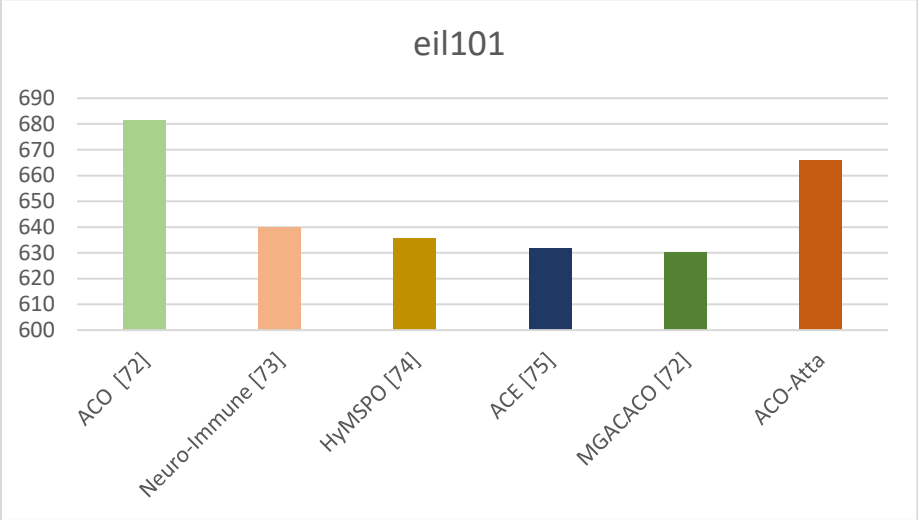

Figura 31. Comparación ACO-Atta con otros algoritmos para la instancia eil101.

Para la instancia lin105, se observa que ACO-Atta supera solo los resultados de ACO tradicional al igual que la instancia anterior, ACO-Atta obtiene una solución mínima de 14452 mientras que ACO tradicional alcanza una solución mínima 14464 con un ER_{mejor} del 0.593% y con un ER_{avg} de 0.73%. Además, se observa que los demás algoritmos superan a ACO-Atta, reportando ER_{mejor} menores al 0.11%. La figura 32 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia lin105.

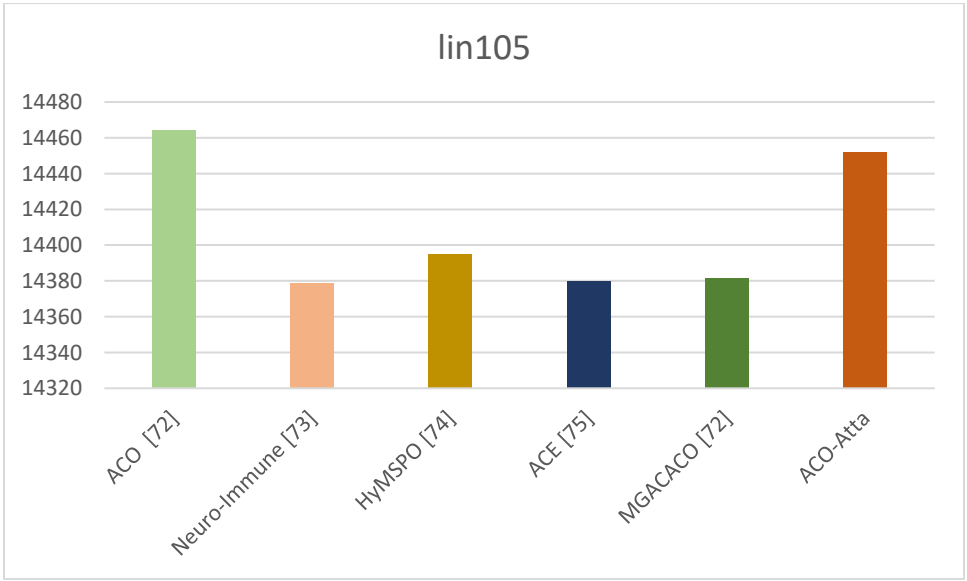

Figura 32. Comparación ACO-Atta con otros algoritmos para la instancia lin105.

Para la instancia ch130, se observa que ACO-Atta no logra supera a los resultados demás algoritmos, en este caso ACO tradicional, obtuvo una solución mínima de 6148.32 con un ER_{mejor} del 0.62% y con un ER_{avg} de 1.01%, mientras que ACO-Atta alcanza una solución mínima de 6261, con un ER_{mejor} del 2.47% y con un ER_{avg} de 6.24%. Los demás algoritmos obtienen mejores resultados reportando ER_{mejor} menores del 1.53%, donde se destaca MGACACO que obtiene la solución reportada en TSPLIB de 6113.26. La figura 33 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia rad100.

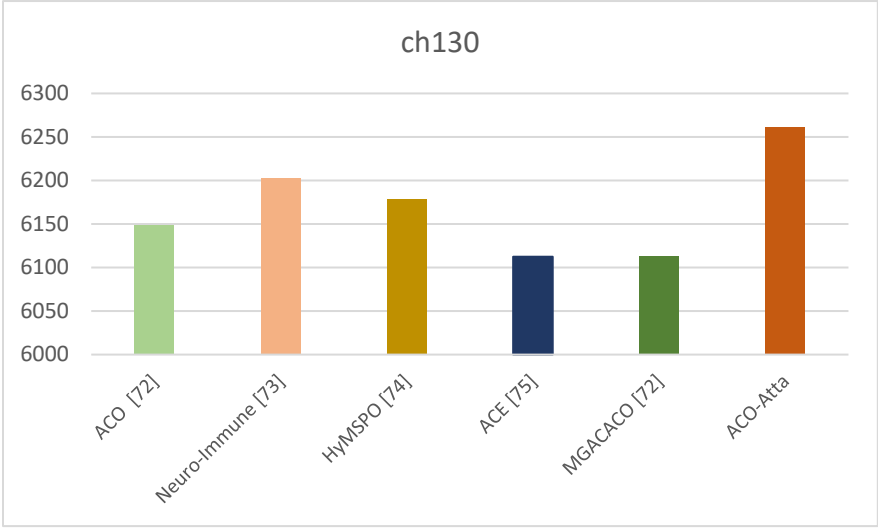

Figura 33. Comparación ACO-Atta con otros algoritmos para la instancia ch130.

Para la instancia pre136, se observa que ACO-Atta no alcanza a superar a los demás algoritmos, en este caso ACO tradicional, obtuvo una solución mínima de 96772 con un ER_{mejor} del 0.225 % y con un ER_{avg} de 0.52%, mientras que ACO-Atta alcanza una solución mínima de 103876, con un ER_{mejor} del 7.34% y con un ER_{avg} de 10.71%. Solo MGACACO presenta mejores resultados para esta instancia, obtiene una solución mínima de 96945.1 reportando ER_{mejor} de 0.2%. La figura 34 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia pre136.

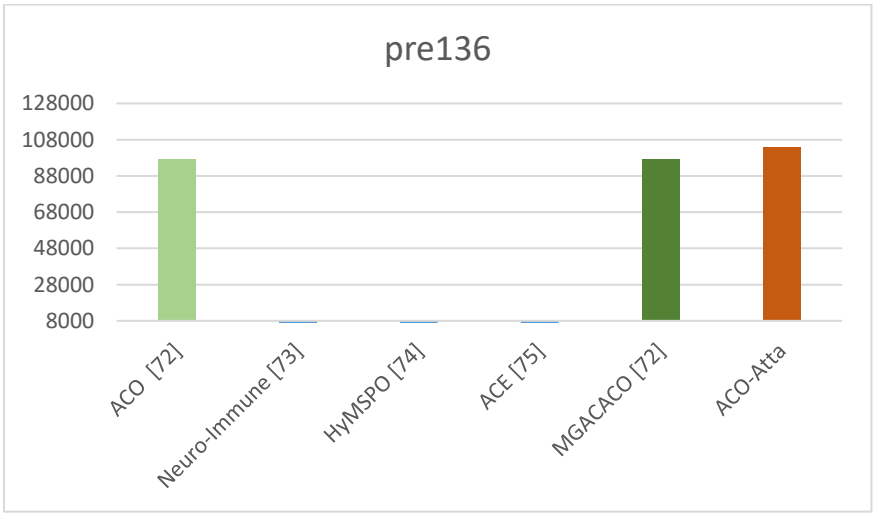

Figura 34. Comparación ACO-Atta con otros algoritmos para la instancia pre136.

Para la instancia u159, se observa que ACO-Atta no alcanza a superar a los demás algoritmos, en este caso ACO tradicional, obtuvo una solución mínima de 42489 con un ER_{mejor} del 0.9 % y con un ER_{avg} de 1.01%, mientras que ACO-Atta alcanza una solución mínima de 43281, con un ER_{mejor} del 2.85% y con un ER_{avg} de 4.74%. Solo MGACACO presenta mejores resultados para esta instancia, obtiene una solución mínima de 42489.2 reportando ER_{mejor} de 1.0%. La figura 35 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia u159.

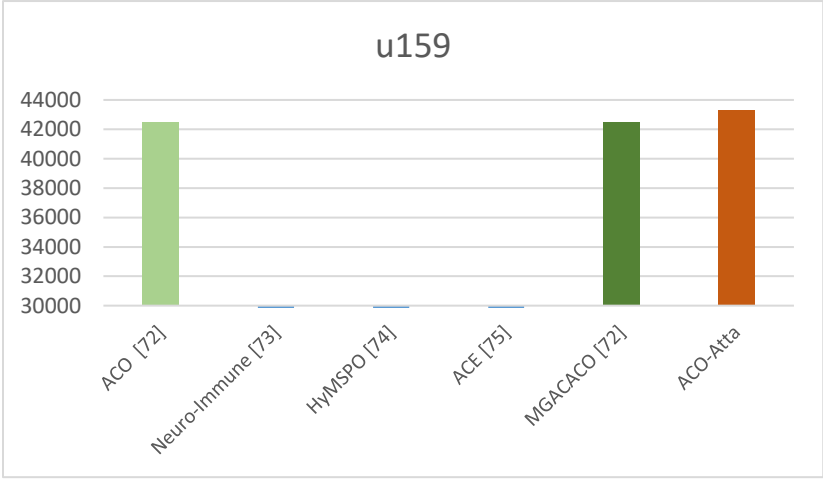

Figura 35. Comparación ACO-Atta con otros algoritmos para la instancia u159.

Para la instancia KroA200, se observa que ACO-Atta no alcanza a superar a los demás algoritmos, en este caso ACO tradicional, obtuvo una solución mínima de 29505.3 con un ER_{mejor} del 0.468 % y con un ER_{avg} de 0.52%, mientras que ACO-Atta alcanza una solución mínima de 31604, con un ER_{mejor} del 7.61% y con un ER_{avg} de 10.74%. donde HyMPSO, ACE y MGACACO presenta mejores resultados para esta instancia, reportando ER_{mejor} menores al 0.13%. La figura 36 muestra el comparativo entre ACO-Atta y los demás algoritmos para la instancia u159.

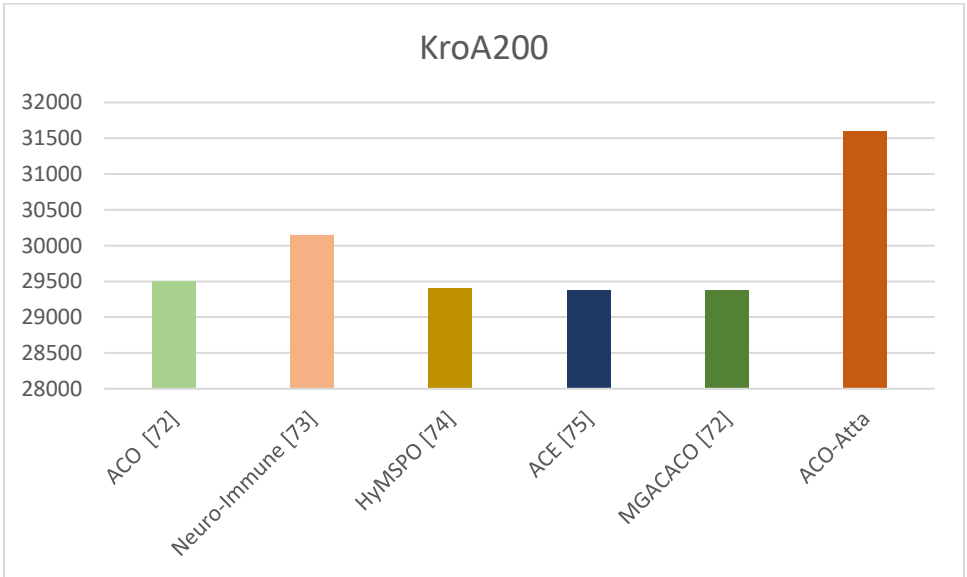

Figura 36. Comparación ACO-Atta con otros algoritmos para la instancia kroA200.

4.4. CONCLUSIONES

En la serie de experimentos de prueba de ACO-Atta, se observa que los mejores resultados obtenidos en las ejecuciones de las 12 instancias corresponden a instancias de corta escala (menores 100 de ciudades), obteniendo o acercándose a la solución reportada en TSPLIB, sin embargo, para las instancias de larga escala el algoritmo le cuesta encontrar soluciones de buena calidad.

Los datos proporcionados en la tabla 21, donde se muestran el ciclo hamiltoniano de cada una de las instancias, es importante debido qué, en el estado del arte pocos estudios brindan esta información, la cual se considera relevante para las diferentes comparaciones y análisis de las rutas, de esta forma identificar los tramos donde ACO-Atta le cuesta más trabajo mejorar la calidad de la solución y asi forma realizar ajustes para mejoras.

En las pruebas realizadas de ACO-Atta a las 12 instancias de TSP, se muestra que en 6 instancias logra superar al ACO tradicional para instancias de TSPLIB de corta escala (menores de 100 ciudades), sin embargo, en las instancias de larga escala ACO-Atta es superado por ACO.

Como resultado se obtiene:

- Resultados de los experimentos de 12 instancias de TSPLIB - Anexo 3.

- Herramienta en Excel que permite extraer los puntos de cada instancia según las mejores rutas obtenidas y de esta forma graficar la ruta de la instancia según la longitud y latitud de sus puntos – Anexo 3(en la carpeta de cada instancia)

5. CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se resume de forma general los resultados obtenidos de cada una de las actividades realizadas, y se enuncian algunas recomendaciones para trabajos futuros.

5.1. CONCLUSIONES GENERALES

5.1.1. REVISIÓN SISTEMÁTICA DE LITERATURA

Este tipo de diseño La revisión sistemática de literatura permitió ampliar el panorama de las diferentes implementaciones de ACO, entre los estudios seleccionados se evidencia que algunos estudios trabajan con el concepto de castas de hormigas o múltiples colonias, sin embargo, no se evidencia el uso de algoritmos basados en ACO que incorporen características de las hormigas *Atta* en el área de computación, lo que permitió la continuidad de este trabajo.

Del estado de arte seleccionado por la RSL, se identifica como las diferentes implementaciones o variantes de ACO realizan pruebas sobre instancias de TSPLIB, con un método de comparación por resultado. Esto brinda

5.1.2. DISEÑO DEL ALGORITMO

Fue desarrollado una variante del algoritmo de optimización de colonia de hormigas - ACO llamada ACO-Atta la cual incorpora conceptos de las hormigas arrieras o *Atta*, dónde se mantiene el concepto de hormiga exploradora quien se encarga que se explorar las posibles soluciones de una instancia de TSP, encontrando las mejores soluciones apoyados del concepto de feromona. Otro concepto es la hormiga seleccionadora, encargada de seleccionar las mejores soluciones de cada iteración, las cuales son proporcionadas a la hormiga cultivadora que encarga de ayudar a mejorar la soluciones por medio del hongo en el proceso de cultivo de este.

ACO-Atta es implementado de forma secuencial para que sea comparado contra el ACO tradicional y así se justo con las pruebas realizadas presenta soluciones de calidad.

5.1.3. ANÁLISIS DE RESULTADOS

Entre las comparaciones se puede evidenciar que ACO-Atta converge más rápido que las otras propuestas, en las experimentaciones realizadas con 100 iteraciones ACO-Atta se acerca en varias instancias a la mejor solución, donde otros algoritmos necesitan de 1000 a 3000 iteraciones para alcanzar una buena solución.

Se identifica que ACO converge lento en comparación a ACO-Atta, donde para algunas instancias obtiene resultados de efectividad entre un 8% al 40%, lo que iednica que la propuesta cumple con el diseño propuesto.

5.2. TRABAJO FUTURO

- Dados los resultados de ACO-Atta para las instancias de larga escala, es necesario realizar un ajuste en los parámetros para obtener mejores resultados, cuando se usó *HyperOpt*, los parámetros fueron ajustados sobre la instancia berlin52, por tanto, como un nuevo experimento se puede llegar a ejecutar *HyperOpt* para instancias de larga escala como ch130 o KroA200 para superar los resultados de ACO.

- Para reducir el tiempo de ejecución de ACO-Atta, se puede llegar a usar computación paralela con una estrategia **BLOCK$_{ant}$**, presentada en [74] que puede ser utilizada para mejorar la *construcción de la solución* de ACO-Atta y de esta forma mejorar los tiempos de ejecución y resultados de obtenidos en las instancias de *TSP.* La propuesta consiste en asignar por cada hormiga exploradora un bloque de hilos CUDA, donde la probabilidad de transición puede ser calculada en paralelo, la condición de transición pude ser asignada al valor heurístico y valor de la feromona. Además, cada hilo verificaría si una ciudad ha sido visitada o no, adicional se puede modificar a la hormiga seleccionadora con una memoria donde se pude colocar una bandera para indicar si la solución ya fue seleccionada. Finalmente, intensificar en CPU las soluciones obtenidas por la hormiga seleccionadora para que sean mejoradas por medio de las hormigas cultivadoras en el cultivo. La figura 37 muestra el proceso que podría llegar a implementarse utilizando computación paralela.

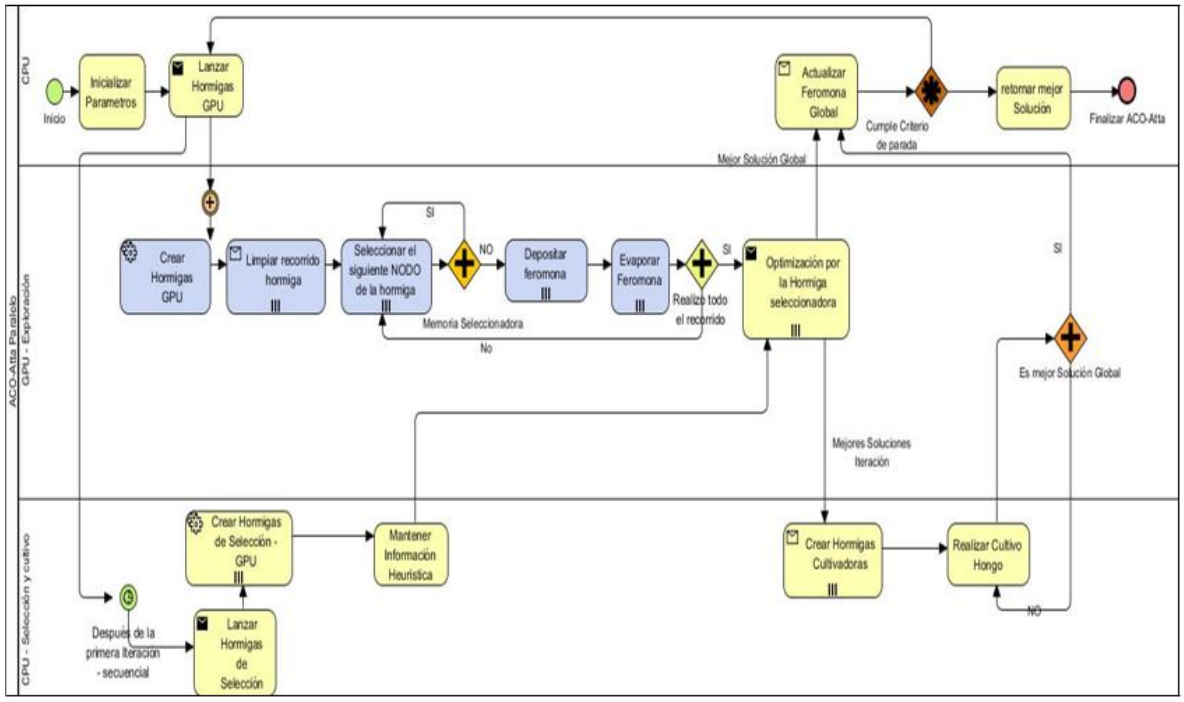

Figura 37. Propuesta ACO-Atta paralelo.

6. REFERENCIAS Y BIBLIOGRAFÍA

[1] S. Binitha and S. S. Sathya, “A Survey of Bio inspired Optimization Algorithms,” *Int. J. Soft Comput. Eng.*, vol. 2, no. 2, pp. 137–151, 2012.

[2] M. Mukhopadhyay, “A brief survey on bioinspired optimization,” 2014.

[3] O. Cordon, F. Herrera, and T. Stiitzle, “A Review on the Ant Colony Optimization Metaheuristic-Basis Models and New Trends.” p. 35, 2002.

[4] M. S. R. Monteiro, D. B. M. M. Fontes, and F. a. C. C. Fontes, “Ant Colony Optimization: a literature survey,” *FEP Work. Pap.*, no. December, 2012.

[5] B. Chandra Mohan and R. Baskaran, “A survey: Ant Colony Optimization based recent research and implementation on several engineering domain,” *Expert Systems with Applications*, vol. 39, no. 4. pp. 4618–4627, 2012.

[6] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, 2005.

[7] U. Jaiswal and S. Aggarwal, “Ant Colony Optimization,” *Int. J. Sci. Eng. Res.*, vol. 2, no. 7, 2011.

[8] U. G. Mueller, N. M. Gerardo, D. K. Aanen, D. L. Six, and T. R. Schultz, “The evolution of agriculture in insects,” *Annu. Rev. Ecol. Evol. Syst*, vol. 36, pp. 563–95, 2005.

[9] K. Pratt, “Design Patterns for Research Methods: Iterative Field Research,” *AAAI Spring Symp. Exp. Des. Real …*, no. 1994, 2009.

[10] R. Velásquez, K. Candoti, and D. Mavares, “Metodología de ajuste de parámetros en algoritmos de optimización metaheurísticos,” vol. 6, no. 2, 2015.

[11] H. Fernandez-Marin and W. T. Wcislo, “Hormigas cultivadoras de hongos,” *Investigacion Y Ciencia*, pp. 12–13, Oct-2010.

[12] M. Herrera and N. Valenciaga, “Peculiaridades de las bibijaguas (Attini: Acromyrmex y Atta) que hacen difícil su control,” *Rev. Cuba. Cienc. Agrícola*, vol. 45, no. 3, 2011.

[13] S. Nygaard *et al.*, “Reciprocal genomic evolution in the ant-fungus agricultural symbiosis,” *Nat. Commun.*, vol. 7, pp. 1–9, 2016.

[14] E. A. K. Mishra, M. N. Das, and T. C. Panda, “Swarm Intelligence Optimization: Editorial Survey,” *Int. J. Emerg. Technol. Adv. Eng.*, vol. 3, no. 1, pp. 217–230, 2013.

[15] D. Martens, B. Baesens, and T. Fawcett, “Editorial survey: Swarm intelligence for data mining,” *Mach. Learn.*, vol. 82, no. 1, pp. 1–42, 2011.

[16] S. Roy, S. Biswas, and S. Sinha Chaudhuri, “Nature-Inspired Swarm Intelligence and Its Applications,” *Int. J. Mod. Educ. Comput. Sci.*, vol. 6, no. 12, pp. 55–65, 2014.

[17] C. Blum, “Ant colony optimization: Introduction and recent trends,” *Phys. Life Rev.*, vol. 2, no. 4, pp. 353–373, 2005.

[18] T. Revuelta Martinez and M. Vázquez Pérez, “Desarrollo y aplicación de un algoritmo ACO para el problema ATSP,” 2015.

[19] F. Luis, A. Arito, G. Leguizamón, and M. Errecalde, “Algoritmos de Optimizaci´on basados en Colonias de Hormigas aplicados al Problema de Asignaci´on Cuadr´atica y otros problemas relacionados,” 2010.

[20] M. Pedemonte, S. Nesmachnow, and H. Cancela, “A survey on parallel ant colony optimization,” *Applied Soft Computing Journal*. 2011.

[21] Z. Hlaing and M. Khine, “An ant colony optimization algorithm for solving traveling salesman problem,” in *International Conference on Information Communication and Management*, 2011, vol. 16, pp. 54–59.

[22] M. M. Kabir, M. Shahjahan, and K. Murase, “A new hybrid ant colony optimization algorithm for feature selection,” *Expert Syst. Appl.*, vol. 39, no. 3, pp. 3747–3763, Feb. 2012.

[23] M. Dorigo and G. Di Caro, “Ant colony optimization: A new meta-heuristic,” *Proc. 1999 Congr. Evol. Comput. CEC 1999*, vol. 2, no. February, pp. 1470–1477, 1999.

[24] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic literature reviews in software engineering - A systematic literature review,” *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[25] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 833–859, 2008.

[26] Y. Zhou, F. He, N. Hou, and Y. Qiu, “Parallel ant colony optimization on multi-core SIMD CPUs,” *Futur. Gener. Comput. Syst.*, vol. 79, pp. 473–487, 2018.

[27] G. Dong, W. Li, J. Shen, Y. Wang, X. Fu, and W. W. Guo, “Solving traveling salesman problems with ant colony optimization algorithms in sequential and parallel computing environments: A normalized comparison,” *Int. J. Mach. Learn. Comput.*, vol. 8, no. 2, pp. 98–103, 2018.

[28] M. Starzec, G. Starzec, A. Byrski, and W. Turek, “Distributed ant colony optimization based on actor model,” *Parallel Comput.*, vol. 90, p. 102573, 2019.

[29] R. Skinderowicz, “The GPU-based parallel Ant Colony System,” *J. Parallel Distrib. Comput.*, vol. 98, pp. 48–60, 2016.

[30] T. P. Hong, L. I. Huang, and W. Y. Lin, “A different perspective on parallel sub-ant-colonies,” *12th Int. Conf. Adv. Mob. Comput. Multimedia, MoMM 2014*, pp. 322–325, 2014.

[31] M. Xu, X. You, and S. Liu, “Dual Population Ant Colony Optimization Algorithm,” vol. 5, 2017.

[32] M. Sekara *et al.*, “Multi-pheromone Ant Colony Optimization for socio-cognitive simulation purposes,” *Procedia Comput. Sci.*, vol. 51, no. 1, pp. 954–963, 2015.

[33] A. M. Mohsen, “Annealing Ant Colony Optimization with Mutation Operator for Solving TSP,” *Comput. Intell. Neurosci.*, vol. 2016, 2016.

[34] E. Liao and C. Liu, “A hierarchical algorithm based on density peaks clustering and ant colony optimization for traveling salesman problem,” *IEEE Access*, vol. 6, pp. 38921–38933, 2018.

[35] Y. Xiao, J. Jiao, J. Pei, K. Zhou, and X. Yang, “A Multi-strategy Improved Ant Colony Algorithm for Solving Traveling Salesman Problem,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 394, no. 4, 2018.

[36] M. Yu, “A solution of TSP based on the ant colony algorithm improved by particle swarm optimization,” *Discret. Contin. Dyn. Syst. - Ser. S*, vol. 12, no. 4–5, pp. 979–987, 2019.

[37] H. E. Tseng, C. C. Chang, S. C. Lee, and Y. M. Huang, “Hybrid bidirectional ant colony optimization (hybrid BACO): An algorithm for disassembly sequence planning,” *Eng. Appl. Artif. Intell.*, vol. 83, no. May, pp. 45–56, 2019.

[38] S. Al-Shihabi, M. Arafeh, and M. Barghash, “An improved hybrid algorithm for the set covering problem,” *Comput. Ind. Eng.*, vol. 85, pp. 328–334, 2015.

[39] M. Mavrovouniotis, F. M. Müller, and S. Yang, “An ant colony optimization based memetic algorithm for the dynamic travelling salesman problem,” *GECCO 2015 - Proc. 2015 Genet. Evol. Comput. Conf.*, no. Cci, pp. 49–56, 2015.

[40] E. López-Santana, W. C. Rodríguez-Vásquez, and G. Méndez-Giraldo, “A hybrid expert system, clustering and ant colony optimization approach for scheduling and routing problem in courier services,” *Int. J. Ind. Eng. Comput.*, vol. 9, no. 3, pp. 369–396, 2018.

[41] K. Thirugnanasambandam, R. S. Raghav, D. Saravanan, U. Prabu, and M. Rajeswari, “Experimental analysis of ant system on travelling salesman problem dataset TSPLIB,” *EAI Endorsed Trans. Pervasive Heal. Technol.*, vol. 5, no. 19, pp. 1–15, 2019.

[42] W. Dend, J. Xu, and H. Zhao, “An improved ant colony optimization algorithm based on immunization strategy,” *Adv. Mater. Res.*, vol. 490–495, pp. 66–70, 2019.

[43] Z. Zhang, C. Gao, Y. Lu, Y. Liu, and M. Liang, “Multi-objective ant colony optimization based on the physarum-inspired mathematical model for bi-objective traveling salesman problems,” *PLoS One*, vol. 11, no. 1, pp. 1–23, 2016.

[44] Z. A. Aziz, “Ant Colony Hyper-heuristics for Travelling Salesman Problem,” *Procedia Comput. Sci.*, vol. 76, no. Iris, pp. 534–538, 2015.

[45] A. El Afia, S. Bouzbita, and R. Faizi, “The effect of updating the local pheromone on ACS performance using fuzzy logic,” *Int. J. Electr. Comput. Eng.*, vol. 7, no. 4, pp. 2161–2168, 2017.

[46] F. Dahan, K. El Hindi, H. Mathkour, and H. Alsalman, “Dynamic flying ant colony optimization (DFACO) for solving the traveling salesman problem,” *Sensors (Switzerland)*, vol. 19, no. 8, 2019.

[47] R. Sagban, K. R. Ku-Mahamud, and M. S. Abu Bakar, “ACO ustic: A nature-inspired exploration indicator for ant colony optimization,” *Sci. World J.*, vol. 2015, 2015.

[48] E. N. Kencana, I. Harini, and K. Mayuliana, “The Performance of Ant System in Solving Multi Traveling Salesmen Problem,” *Procedia Comput. Sci.*, vol. 124, pp. 46–52, 2017.

[49] H. Eldem and E. Ülker, “The application of ant colony optimization in the solution of 3D traveling salesman problem on a sphere,” *Eng. Sci. Technol. an Int. J.*, vol. 20, no. 4, pp. 1242–1248, 2017.

[50] Y. Oonsrikaw and A. Thammano, “Modified ant colony optimization algorithm for multiple-vehicle traveling salesman problems,” *Int. J. Networked Distrib. Comput.*, vol. 7, no. 1, pp. 29–36, 2018.

[51] H. Zhang and X. You, “Multi-Population Ant Colony Optimization Algorithm Based on Congestion Factor and Co-Evolution Mechanism,” *IEEE Access*, vol. 7, pp. 158160–158169, 2019.

[52] Y. Luo, P. Guo, and M. Zhang, “A Framework of Ant Colony P System,” *IEEE Access*, vol. 7, pp. 157655–157666, 2019.

[53] Y. Yan, H. suk Sohn, and G. Reyes, “A modified ant system to achieve better balance between intensification and diversification for the traveling salesman problem,” *Appl. Soft Comput. J.*, vol. 60, pp. 256–267, 2017.

[54] T. Thepphakorn, P. Pongcharoen, and C. Hicks, “An ant colony based timetabling tool,” *Int. J. Prod. Econ.*, vol. 149, pp. 131–144, 2014.

[55] X. C. Han, H. W. Ke, Y. J. Gong, Y. Lin, W. L. Liu, and J. Zhang, “Multimodal optimization of traveling salesman problem: A niching ant colony system,” *GECCO 2018 Companion - Proc. 2018 Genet. Evol. Comput. Conf. Companion*, pp. 87–88, 2018.

[56] D. R. D. S. Alves, M. T. R. S. Neto, F. D. S. Ferreira, and O. N. Teixeira, “SIACO: A novel algorithm based on ant colony optimization and game theory for travelling salesman problem,” *ACM Int. Conf. Proceeding Ser.*, pp. 62–66, 2018.

[57] S. Bouzbita, A. El Afia, and R. Faizi, “Parameter adaptation for ant colony system algorithm using hidden markov model for tsp problems,” *ACM Int. Conf. Proceeding Ser.*, no. 1, pp. 1–6, 2018.

[58] S. Chowdhury, M. Marufuzzaman, H. Tunc, L. Bian, and W. Bullington, “A modified Ant Colony Optimization algorithm to solve a dynamic traveling salesman problem: A case study with drones for wildlife surveillance,” *J. Comput. Des. Eng.*, vol. 6, no. 3, pp. 368–386, 2019.

[59] G. Wu, R. Mallipeddi, and P. N. Suganthan, “Ensemble strategies for population-

based optimization algorithms – A survey," *Swarm Evol. Comput.*, vol. 44, no. August 2018, pp. 695–711, 2019.

[60] R. Skinderowicz, "Population-Based Ant Colony Optimization for Sequential," vol. 1, no. April, pp. 471–472, 2015.

[61] J. Ning, Q. Zhang, C. Zhang, and B. Zhang, "A best-path-updating information-guided ant colony optimization algorithm," *Inf. Sci. (Ny).*, vol. 433–434, pp. 142–162, 2018.

[62] S. Jayasankari and O. P. Uma Maheswari, "Multi improved ant colony optimization (IACO) based cluster head selection and information gathering in wireless sensor networks (WSNs)," *Int. J. Adv. Sci. Technol.*, vol. 28, no. 17, pp. 326–337, 2019.

[63] S. Alaiso, J. Backman, and A. Visala, *Ant colony optimization for scheduling of agricultural contracting work*, vol. 4, no. PART 1. IFAC, 2013.

[64] U. Heidelberg, "TSPLIB - Discrete and Combinatorial Optimization." [Online]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

[65] A. Sathyan, N. Boone, and K. Cohen, "Comparison of Approximate Approaches to Solving the Travelling Salesman Problem and its Application to UAV Swarming," *Int. J. Unmanned Syst. Eng.*, vol. 3, no. 1, pp. 1–16, 2015.

[66] PSF, "Python Software Foundation." [Online]. Available: http://www.python.org.

[67] C. Gavidia-Calderon and C. Beltrán Castañon, "Isula: A java framework for ant colony algorithms," *SoftwareX*, vol. 11, p. 100400, 2020.

[68] H. E. Tito Chura, C. A. Silva Delgado, E. E. Alfaro Gonzales, and E. Fajardo Espinoza, "Aplicación Del Algoritmo De Colonia De Hormigas Al Problema Del Agente Viajero," *Cienc. Desarro.*, no. 20, pp. 98–102, 2019.

[69] pypi, "hyperopt," 2020. [Online]. Available: https://pypi.org/project/hyperopt/.

[70] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang, and D. Wu, "A novel collaborative optimization algorithm in solving complex optimization problems," *Soft Comput.*, vol. 21, no. 15, pp. 4387–4398, 2017.

[71] R. Pasti and L. N. De Castro, "A neuro-immune network for solving the traveling salesman problem," *IEEE Int. Conf. Neural Networks - Conf. Proc.*, no. January, pp. 3760–3766, 2006.

[72] Y. Marinakis and M. Marinaki, "A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 432–442, 2010.

[73] J. B. Escario, J. F. Jimenez, and J. M. Giron-Sierra, "Ant colony extended: Experiments on the travelling salesman problem," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 390–410, 2015.

[74] K. C. Wei, C. C. Wu, and C. J. Wu, "Using CUDA GPU to accelerate the ant colony optimization algorithm," *Parallel Distrib. Comput. Appl. Technol. PDCAT Proc.*, pp.

90–95, 2014.