METAHEURÍSTICA ACO-FUNGUS PARA LA OPTIMIZACIÓN DEL PROBLEMA DEL EMPAREJAMIENTO DE LA TRIPULACIÓN BAJO INCIDENTE OPERATIVO

Trabajo de Grado

Andrés Felipe Guerrero Muñoz
Julián Yesid Montero Muñoz

Director del Proyecto:
Ember Ubeimar Martínez Flor
Magíster en Computación

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Línea de Investigación Ciencia de los datos
Popayán, 05 de septiembre del 2021

TABLA DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE FIGURAS

GLOSARIO

En esta sección se define la terminología que se usa en el planteamiento del problema de emparejamiento de la tripulación.

Crew members: miembros de la tripulación, generalmente se dividen en dos grupos según su función: los miembros de la tripulación de cabina son el piloto (capitán), el copiloto (primer oficial) y el ingeniero de vuelo, todos los cuales están calificados para volar uno o más tipos de aeronaves. Los miembros de la tripulación de cabina son el capitán de cabina y los auxiliares de vuelo.

Tramo aéreo (Vuelo): Un segmento de vuelo sin escalas. Cada tramo tiene la siguiente información: número de vuelo, aeropuerto de origen, aeropuerto de destino, hora de salida y hora de llegada.

Deadhead: Tramo aéreo en el que un miembro de la tripulación aún en servicio vuela como pasajero con fines de reubicación para regresar a su base.

Tiempo de vuelo: Tiempo total desde que la aeronave despega de un origen hasta que aterriza en un destino.

Servicio o duty: Secuencia de tramos aéreos (vuelos) consecutivos (y / o puntos muertos) que comprende un día de trabajo para un solo miembro de la tripulación, durante un turno o jornada en el día, puede tener máximo 4 vuelos y tiene un tiempo máximo de 14 horas en un periodo de 24 horas. Dos deberes consecutivos deben comenzar y terminar en el mismo aeropuerto. Los deberes están separados por escalas.

Tiempo de servicio: Tiempo total desde que la tripulación entra en servicio (Brief), usualmente una hora antes de despegar el primer vuelo hasta finalizar el Debrief.

Brief: Tiempo de reuniones informativas transcurrido antes del inicio de cada servicio (duty), usualmente una hora. Tiempo que se dedica a las instrucciones y discusiones de la tripulación con el objetivo de transformar a un grupo de personas en un equipo eficaz.

Debrief: Tiempo para informar transcurre después del último vuelo del servicio (duty), usualmente media hora. Brinda a los miembros de la tripulación una comprensión de los eventos que ocurrieron y sus implicaciones.

Tiempo de vuelo de un servicio: Suma de todos los tiempos de cada uno de los vuelos de un servicio, tiene un tiempo máximo de 8 horas.

Emparejamiento: Una secuencia de tareas y escalas para un miembro de la tripulación no especificado que comienza y termina en una base. En problemas de corta y mediana distancia los emparejamientos suelen durar de 1 a 5 días; en problemas de larga distancia se permiten emparejamientos más largos.

Tiempo muerto: Tiempo en el que una tripulación aún en servicio viaja como pasajero para regresar a su base.

Rest: Tiempo en el cual la tripulación está en descanso en medio de dos servicios, también llamada conexión nocturna.

Connection time: Periodo de tiempo entre dos tramos de vuelo consecutivos, generalmente las aerolíneas consideran un tiempo mínimo de 30 minutos y un máximo de 3 horas.

Tiempo de un emparejamiento de la tripulación: Tiempo total que está constituido por uno o más servicios, incluidos los tiempos de descanso.

Forrajeo: todo comportamiento asociado a la obtención y el consumo de alimento que obliga al animal a buscar o cazar.

Castas: en los insectos sociales, son los miembros de una determinada especie que cumplen diferentes funciones dentro de la comunidad.

Comportamiento eusocial: sistema de organización dentro del cual un gran número de individuos coexisten en grandes colonias donde se presenta un solapamiento de generaciones, división de labores y cuidados parentales compartidos.

CAPÍTULO 1. INTRODUCCIÓN

Uno de los mayores desafíos con los que diariamente se enfrentan las compañías aéreas es la de abordar toda su planificación operativa de manera óptima, con el objetivo de la reducción al máximo de los costos asociados a la operación de la aerolínea.

Uno de los procesos de planificación donde las aerolíneas invierten sus recursos (tiempo y dinero), para proveer el servicio de transporte a sus pasajeros es la programación de la tripulación (en inglés crew scheduling). La programación de la tripulación es el proceso en el cual se busca identificar secuencias de vuelos disponibles y asignar el personal idóneo para cada tramo de vuelo. El problema de la programación de la tripulación se divide en dos subprocesos: el emparejamiento de la tripulación y la asignación de la tripulación.

El proceso del emparejamiento de la tripulación consiste en encontrar una secuencia de vuelos óptimos para una tripulación que hasta el momento se considera anónima, cumpliendo con normas y restricciones impuestas por entes gubernamentales, sindicatos y de las mismas aerolíneas. Entre las restricciones más importantes se encuentran el regreso a la base, el número de vuelos por emparejamiento, duración del emparejamiento entre otras. Se considera un emparejamiento factible si se cumplen con las normas y restricciones impuestas.

Teniendo en cuenta la realidad de la operación aérea donde hay factores externos que hacen que en la planificación se presenten interrupciones, el proceso del emparejamiento de la tripulación es afectada por incidentes operativos entre los que se encuentran: el mantenimiento no programado de las aeronaves, retraso, cancelación en los vuelos, aspectos meteorológicos entre otros. Estas interrupciones generan sobrecostos para las compañías aéreas como para sus pasajeros, según [29] un estudio realizado estima que dichas interrupciones generan a la industria un alto porcentaje de pérdidas al año, debido a que se deben aumentar los gastos en las tripulaciones, combustible y mantenimiento. Además de que crean problemas de conexiones perdidas, disgusto entre los viajeros llevando a posibles compensaciones y asistencias a los pasajeros en caso de cancelación o gran retraso de los vuelos.
Lo que se busca es generar emparejamientos factibles y cuando se presente un incidente reorganizar los emparejamientos de manera eficiente para no generar cuantiosas pérdidas para las aerolíneas.

Para dar solución al problema del emparejamiento de la tripulación, bajo incidente operativo se plantea una solución basada en el algoritmo de optimización de colonia de hormigas ACO, el cual se basa en el proceso natural de las hormigas en la búsqueda de su alimento. Sin embargo, aunque ACO fue inspirado por características naturales de las colonias de hormigas, existen estudios que describen comportamientos más complejos como por ejemplo el de las hormigas cortadoras de hojas (*hormigas Atta*) [32] [33] [37]. Estos trabajos destacan entre muchos otros aspectos de su comportamiento biológico en tener un sistema de agricultura donde cultivan un hongo el cual es la fuente primaria de alimento para la colonia, lo que significa que las hormigas cortadoras no se alimentan de las hojas que cortan, la fuente de proteínas de la colonia proviene de los hongos junto con los fragmentos de las plantas cultivan dentro del hormiguero [37].

En este trabajo se adoptan roles, procesos y actividades propias de las hormigas cortadoras de hojas cultivadoras. El algoritmo propuesto incorpora aspectos biológicos y de comportamiento. Las hormigas poseen una de las estructuras sociales más complejas de grupo por lo que han desarrollado la capacidad de la división del trabajo dentro de las colonias, lo que se expresa en la existencia de diferentes castas morfológicas o grupos de hormigas que son de formas muy diferentes entre sí [37]. Las actividades de una colonia de hormigas se dividen en tres etapas: corte y transporte de material vegetal, cultivo del hongo y disposición final de desechos [33]. La etapa de corte y transporte de material lo realiza una casta de hormiga que es la encargada de llevar todo el material recolectado (hojas, flores) al nido. Para la etapa del cultivo del hongo se requiere cosechar el material vegetal alrededor de la colonia para luego transportarlo a cámaras subterráneas; en el interior del hormiguero el material vegetal es limpiado, raspado y cortado en pedazos pequeños que son masticados e impregnados de saliva y otros fluidos para obtener una pulpa que se distribuye como medio de crecimiento para el hongo. La etapa final que desarrollan las hormigas es la disposición final de los desechos cuando el sustrato vegetal va perdiendo su valor nutrimental y el hongo finaliza su ciclo de vida [33].

1.1 PLANTEAMIENTO DEL PROBLEMA

El transporte aéreo es uno de los más efectivos a la hora de reducir distancias desde un punto geográfico a otro, por lo tanto, se ha convertido a través de los años en uno de los más atractivos a la hora de viajar, debido a que el usuario siempre busca practicidad y comodidad a pesar de los costos que implica hacer uso de él. Por lo cual desde hace algunos años se ha incrementado notablemente el surgimiento de nuevas aerolíneas y el mejoramiento de la capacidad de las que ya existían. Sin embargo, esto involucra un problema que va más allá de una competencia comercial de oferta y demanda; puesto que este hecho trae problemas de logística que a su vez genera un exceso en los costos de planificación y operación [1]. A lo cual las compañías aéreas sin importar el tamaño siempre se encuentran con la necesidad y búsqueda de la reducción de costos, para así poder tener un margen de ganancia, siendo competitivas y accesibles para el usuario en el mercado del transporte aéreo. Actualmente las aerolíneas deben enfrentar diferentes problemas para poder ser viables, entre los problemas que enfrentan las aerolíneas se encuentran los problemas de logística y planeación.

Uno de los problemas con mayor impacto para las compañías aéreas está en sus procesos de planificación. En el proceso de la planificación habitualmente las aerolíneas diferencian cuatro etapas principales: programación de los vuelos, asignación de las flotas, rutas de aeronaves y programación de la tripulación.

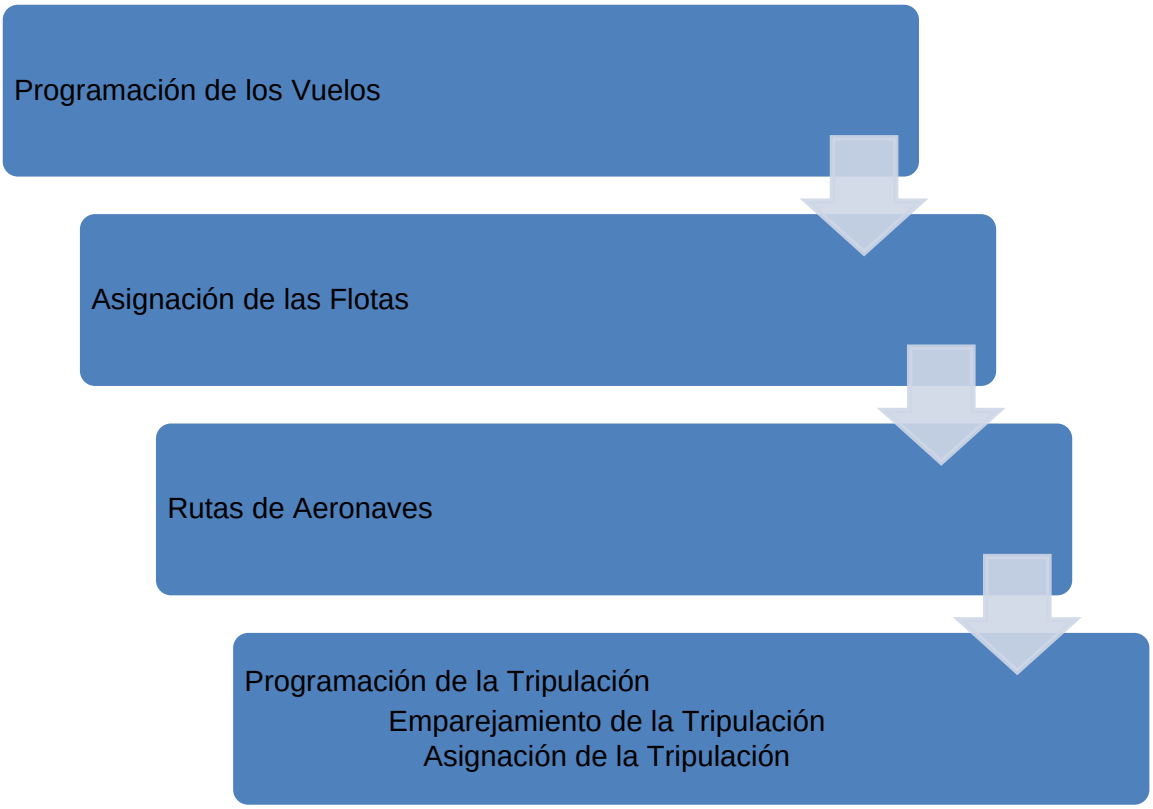


Figura 1. Proceso de planeación de vuelos de una aerolínea.

En la programación de los vuelos, la creación de un horario de vuelo es la primera tarea en el proceso de programación de la aerolínea. La planificación del horario de la aerolínea es uno de los pasos de mayor importancia debido a que afecta cada etapa siguiente de la planificación y tiene el impacto más alto en la demanda de pasajeros. La finalidad de esta etapa de planificación es crear un horario que se muestra al público con la información correcta y completa sobre los vuelos presentados, como aeropuertos de salida y llegada, horarios de salida y llegada, y frecuencias de vuelo [2].

La siguiente etapa consiste en la asignación de las flotas, está habitualmente se establece meses antes de ejecutar los vuelos, y su resultado impacta las dos siguientes etapas (rutas de aeronaves y la programación de la tripulación) [3]. Es común que las aerolíneas tengan varios tipos de aviones para la asignación de las flotas. La finalidad de esta etapa es la de asignar tipos de aeronaves a los tramos de vuelo para que las ganancias sean máximas logrando minimizar dos tipos de costos: costos operativos y costos de derrame (pérdida de pasajeros) esto sucede cuando la aeronave llena su cupo quedando pasajeros sin oportunidad de viajar [4], [5].

Finalizada la asignación de flotas y teniendo un cronograma de estas, la etapa de la programación de la ruta consiste en encontrar los vuelos a diferentes destinos, de tal forma que cumpla con los mantenimientos y recorridos establecidos [6], donde cada aeronave está sujeto a condiciones de vuelo y requisitos de mantenimiento con el objetivo de maximizar las ganancias de acuerdo con la asignación de vuelos encontrados, lo que resulta en una utilización igual de la aeronave y una planificación operativa más fácil [2],[7].

La etapa final de la planificación es la programación de la tripulación considerando una tripulación tradicional [2], tiene por objetivo encontrar un conjunto de emparejamientos de vuelos de mínimo costo para que a cada tramo de vuelo se le asigne una tripulación calificada [1]. Normalmente este proceso se realiza una vez dado un horario de vuelo y una asignación de flota a las distintas rutas que la aerolínea dispone. Esta programación representa uno de los mayores costos operativos en una aerolínea [8], [9] y con el aumento significativo en el número de flotas hace que el número de tripulantes de cabina y tripulación de tierra aumenten y por consiguiente sus costos. En consecuencia, el problema de la programación de la tripulación se divide en dos subproblemas que son secuenciales y están interconectados: el problema del emparejamiento de la tripulación (o de vuelo [14]) y el problema de la asignación de la tripulación [1], [2], [11].

El problema del emparejamiento de la tripulación tiene como objetivo encontrar una secuencia de vuelos que inician y finalizan en una misma base o aeródromo de la tripulación, de tal forma que en una secuencia la ciudad de destino coincide con la ciudad de origen del siguiente vuelo. Cada emparejamiento tiene un costo asociado. El objetivo es encontrar un subconjunto de estas parejas con un costo mínimo y que a su vez cubran todas las rutas programadas exactamente una vez. Al igual que en las etapas previas, hay que tener en cuenta las normas y restricciones, maximizando la utilización de la tripulación (horas de vuelo) [15], [16].

El emparejamiento (Pairing) hace referencia a una lista de vuelos ya definidos en fases anteriores a los cuales se les debe determinar una tripulación que hasta este punto se manejan como tripulaciones anónimas [17], las cuales deben cubrir todos los vuelos planificados. Un emparejamiento está conformado por un conjunto de vuelos que son establecidos a una tripulación, la cual solo opera un determinado tipo de flota o aeronave, para un determinado periodo o tiempo que dura el emparejamiento, por lo general se manejan tiempos o planificaciones de vuelos de 3 a 5 días para cada emparejamiento, tiempo en los cuales la tripulación y la aeronave van a estar en servicio. El conjunto de vuelos establecidos a una tripulación para un día determinado se le conoce como servicio o duty y el conjunto de los servicios se establece como el emparejamiento o secuencia de vuelos que una tripulación debe realizar [1], [3].

El problema del emparejamiento de la tripulación considera una serie de restricciones, entre las que se encuentran reglas de trabajo dadas por las regulaciones del gobierno, políticas de la empresa, acuerdos sindicales, entre otras [11], [12], [13]. Se establece que para cada emparejamiento el tiempo de vuelo total no puede exceder el tiempo de vuelo máximo que las autoridades reguladoras permiten, dependiendo si es un vuelo doméstico o uno internacional. El número de vuelos en un emparejamiento no puede exceder un número máximo de vuelos permitidos. En un tramo de vuelo, la ciudad de llegada de un vuelo es la misma ciudad de partida de su vuelo inmediatamente posterior, el tiempo de espera, es decir, la cantidad de tiempo entre dos vuelos consecutivos debe restringirse dentro del límite inferior y superior (usualmente de 30 minutos a 3 horas) [5]. Si el tiempo de espera excede el tiempo máximo permitido, los miembros de la tripulación deben ser enviados a descansar en el hotel.

Figura 2. Componentes de un emparejamiento

Como cada emparejamiento contiene una secuencia de vuelos, se considera un emparejamiento factible si cumple con las restricciones y si sus servicios están conectados lo que significa que se debe comenzar y terminar en la misma ciudad de origen, es decir la ciudad donde están alojados los miembros de la tripulación. De lo contrario, los miembros de la tripulación son llevados a sus respectivas ciudades de origen como pasajeros normales después del último vuelo de ese emparejamiento [3], generando gastos adicionales y un mayor costo monetario en la operación. Por lo anterior el problema del emparejamiento de la tripulación, representa un problema de optimización complejo, debido al número de variables ya que depende del tamaño de la aerolínea, el número y tipo de flotillas disponibles, sus capacidades operativas y de personal para abordar las distintas rutas que manejan ya sean vuelos nacionales o internacionales [10].

Para plantear una solución ajustada a la realidad operativa de las aerolíneas para el problema del emparejamiento de la tripulación se hace necesario establecer la adición de incidentes operativos (mantenimiento no programado, cancelaciones de vuelos programados, fallas técnicas en alguna aeronave, problemas inesperados con los miembros de la tripulación y problemas con el estado del tiempo [3]) que en operación normal se dan y son difíciles de prever alterando el normal funcionamiento de la operatividad y la planificación.

Se ha considerado tomar en cuenta una falla técnica que lleve a una aeronave a un mantenimiento no programado lo que conlleva a que se vuelva a reprogramar y ajustar toda la planificación, tanto de vuelos como el de la tripulación, generando retrasos y pérdidas monetarias para las compañías aéreas como para los pasajeros.

Como el emparejamiento de la tripulación está supeditado por aspectos multifactoriales que necesitan atenderse con el fin de generar una organización óptima por parte de las aerolíneas de todo el mundo, así como reducir gastos económicos internos en las compañías aéreas [15]. Esto hace necesario encontrar un emparejamiento de la tripulación confiable cumpliendo todas las restricciones que el entorno presente.

Desde el ámbito de la computación el problema del emparejamiento de la tripulación se ha venido trabajando desde diferentes enfoques o tipos de algoritmos. Uno de estos enfoques son los algoritmos metaheurísticos en especial los algoritmos de comportamiento biológico como el algoritmo ACO tradicional (optimización por colonia de hormigas), inspirada en el comportamiento biológico y social de las hormigas reales, con características naturales que permiten adaptar el problema al tener un conjunto finito de componentes y parámetros controlables, con hormigas artificiales que permiten un mejor control de la información que se desea manejar.

En este trabajo se toma como base el algoritmo propuesto por el estudiante de Maestría en Computación de la Universidad del Cauca Francisco Javier Obando Vidal en su trabajo de maestría denominado Variante del algoritmo de optimización de colonia de hormigas (Atta). Algoritmo que incorpora las características biológicas de las hormigas *Atta* cultivadoras de hongos, la adaptación e instanciación requirió realizar cambios en el algoritmo por lo que la solución propuesta debe ser considerada como una versión mejorada del algoritmo inicial.
Por lo anterior surge la siguiente pregunta de investigación. ¿Cómo adaptar el algoritmo ACO-FUNGUS para incorporar incidentes operacionales relacionados con el mantenimiento no programado en la solución del problema de emparejamiento de la tripulación en una aerolínea regional?

1.2 JUSTIFICACIÓN DEL PROBLEMA

Este proyecto surge de la necesidad que existe en la mejora continua del proceso de planificación del emparejamiento de la tripulación, aunque este es solo un proceso de los muchos que se tienen para prestar el servicio aéreo por parte de las aerolíneas, la reducción de los costos en la operación siempre va a hacer unos de los retos a cumplir y más aún cuando por factores económicos muchas de las aerolíneas dejan de operar sus aviones. El factor económico siempre es un factor de trascendencia para cualquier empresa sin importar su tamaño, aunque en las

crisis siempre las empresas pequeñas son las más afectadas como es el caso de las aerolíneas que operan en Colombia donde la más grande aerolínea que presta sus servicios es considerada de tamaño medio. Son diversos los factores que influyen a que una aerolínea pierda sus recursos; la mala administración del dinero, la mala gestión y manejo del tiempo, de recursos como el personal que labora y de la infraestructura (Flotas aviones). Según estadísticas de la Aerocivil entidad que regula la operación aérea en Colombia las compensaciones y pagos hechas por las aerolíneas que operan de forma regular en Colombia generan pérdidas por aspectos como:

- Vuelos cancelados
- Vuelos anticipados
- Vuelos demorados
- Sobreventa
- Equipaje
- Denegación de embarque

Aunque muchos de los anteriores aspectos no sean de fácil predicción como vuelos cancelados, accidentes operativos como el mantenimiento no programado de las aeronaves, cierre de aeropuertos, vuelos demorados. A los que le influyen factores externos que hacen que la operación se ralentice y no se haga una gestión o no se tenga una reacción oportuna, hacen que la incorporación de la tecnología sea necesaria para la optimización de procesos y recursos. Para el proceso del emparejamiento de la tripulación se hace necesario la incorporación de la tecnología y esto es debido a que se necesita tener una programación de los emparejamientos óptimos y que sean de fácil manejo cuando se presente algún incidente operativo. Se necesita realizar un emparejamiento de la tripulación óptimo porque se requiere cubrir los vuelos programados, cumpliendo las restricciones impuestas con el fin de maximizar el tiempo de vuelo de los recursos (tripulación y aeronaves) y de fácil manejo ante cualquier eventualidad en la operación diaria, donde se hace necesario la incorporación de un sistema de monitoreo de la operación y realice ajustes a la planeación cuando se presente un accidente operacional. Se evidencia la necesidad de la incorporación de soluciones tecnológicas que permitan responder oportunamente y de manera eficiente a los problemas de planificación y operación de las aerolíneas colombianas.

1.3 OBJETIVOS

1.3.1. Objetivo General

Proponer una variación del algoritmo ACO-FUNGUS para solucionar el problema del emparejamiento de la tripulación en la operación de una aerolínea regional incorporando los efectos generados por mantenimiento no programado.

1.3.2. Objetivos Específicos

1. Estudiar los algoritmos más utilizados en la solución del problema de emparejamiento de la tripulación de aerolíneas regionales para identificar el alcance, estrategias de solución, manejo de restricciones, la incorporación de incidentes operacionales y reducción de complejidad propuestos en trabajos científicos.

2. Diseñar e implementar una variante del algoritmo ACO-FUNGUS para la solución del problema del emparejamiento de la tripulación de una aerolínea regional, incorporando los efectos generados por mantenimiento no programado.

3. Evaluar la solución propuesta a través de métricas de desempeño comparando los resultados de la evaluación con la mejor solución del estado del arte, para determinar la mejor solución cuando se incorpora los efectos generados por mantenimiento no programado, utilizando un conjunto real de datos, bajo las mismas restricciones.

CAPÍTULO 2. MARCO CONCEPTUAL

El objetivo de este capítulo es presentar los conceptos más relevantes para el desarrollo del presente trabajo, con el fin de facilitar una mejor comprensión de los capítulos y secciones posteriores.

2.1. MARCO TEÓRICO

2.1.1. Emparejamiento de la tripulación

El problema del emparejamiento de la tripulación tiene como objetivo encontrar un conjunto secuencial de vuelos factibles que minimice los costos de operación de la tripulación que incluye pago básico por cada emparejamiento, pago mínimo básico por cada vuelo, turno de trabajo, bono de horas extras, pago adicional por el tiempo de espera, los gastos de hotel y la tripulación que vuela como pasajeros [11], [12]. Maximizando la utilización de la tripulación (horas de vuelo) teniendo en cuenta las diferentes restricciones que estos conjuntos de vuelos presentan [15], [16].

El emparejamiento hace referencia a una lista de vuelos ya definidos en fases anteriores a los cuales se les debe determinar una tripulación que hasta este punto se manejan como tripulaciones anónimas [17], las cuales deben cubrir los vuelos planificados. Un emparejamiento está conformado por un conjunto de vuelos que son establecidos a una tripulación, la cual solo opera un determinado tipo de flota o aeronave, para un determinado periodo o tiempo que dura el emparejamiento, por lo general se manejan tiempos o planificaciones de vuelos de 3 a 5 días para cada emparejamiento, tiempo en los cuales la tripulación y la aeronave van a estar en servicio. El conjunto de vuelos establecidos a una tripulación para un día determinado se le conoce como servicio o duty y el conjunto de los servicios se establece como el emparejamiento o secuencia de vuelos que una tripulación debe realizar [1], [3].

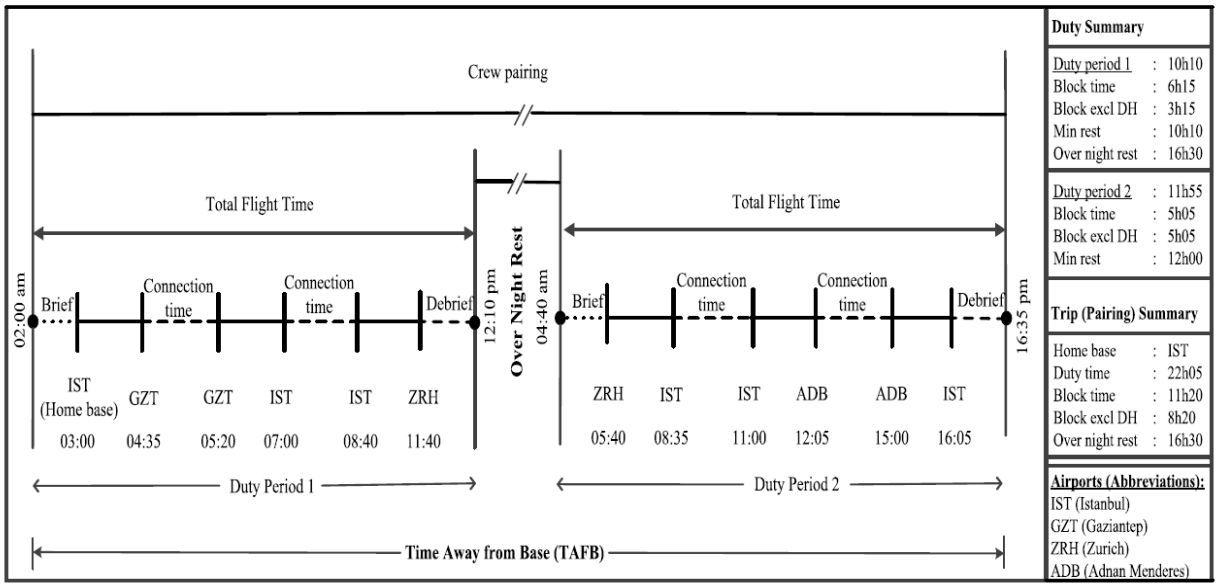


Figura 3. Ejemplo de un emparejamiento con sus componentes. Adaptada de [11]

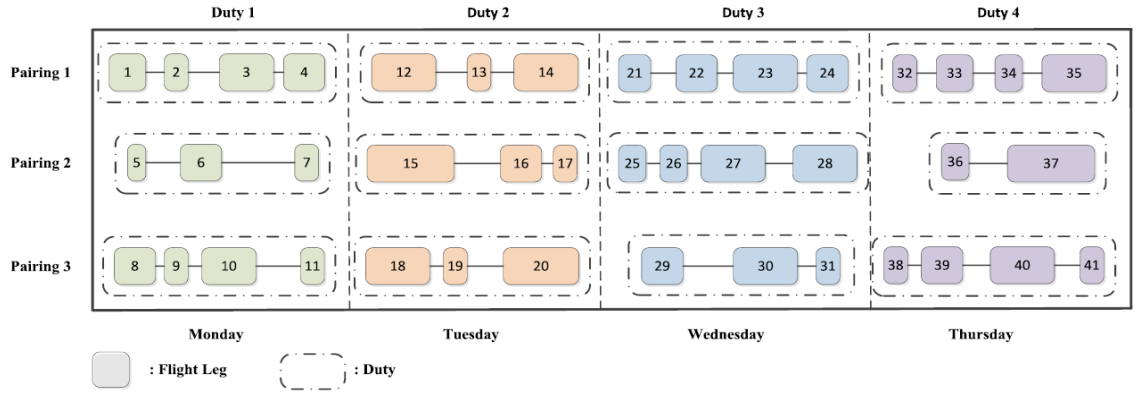


Figura 4. Ejemplo de tres emparejamientos con sus respectivos servicios y vuelos que cubrir en la semana. Adaptada de [11]

2.1.2 Regulaciones para el emparejamiento de la tripulación

En el emparejamiento de la tripulación se requieren considerar varias restricciones legales (gobierno, políticas de la empresa, acuerdos sindicales) y de seguridad [11], [12], [13] para hacer que un emparejamiento sea factible.

Restricciones de ciudad de inicio y ciudad de fin: El primer vuelo de un emparejamiento debe comenzar sólo desde una base de la tripulación y el último vuelo del emparejamiento debe terminar en la misma base de inicio. De lo contrario, los miembros de la tripulación son llevados a sus respectivas ciudades de origen como pasajeros normales después del último vuelo de ese emparejamiento [3], generando gastos adicionales y un mayor costo monetario en la operación.

Conexión de servicio: Asegura que en un par de servicio (duty) dos vuelos de servicio (duty) consecutivos deben estar ubicados en el mismo aeropuerto.

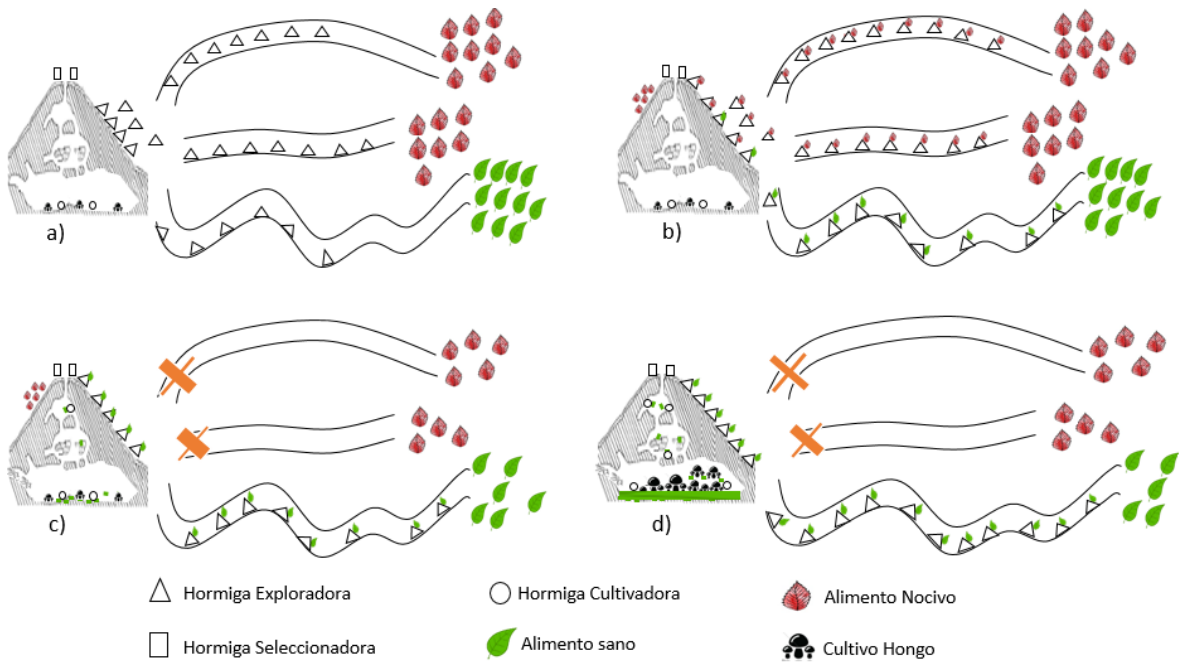
Tiempo de conexión: En un servicio, dos tramos de vuelo siempre están asegurados por un intervalo de tiempo (Tc), llamado tiempo de conexión o tiempo de asiento. Está restringido por un límite máximo y mínimo (usualmente de 30 minutos a 3 horas). El tiempo de descanso representa el tiempo que le toma a la tripulación cambiar el avión para su próximo vuelo, si es necesario o un breve descanso. Si el tiempo de espera excede el tiempo máximo permitido, los miembros de la tripulación deben ser enviados a descansar en el hotel.

Restricciones de deberes: El número de deberes en un emparejamiento está restringido por un límite máximo. Además, el número de vuelos en servicio, el tiempo transcurrido de servicio y el tiempo de vuelo de servicio deben estar restringidos por límites mínimos y máximos, y estos límites varían según el tiempo de inicio del servicio.

Restricciones especiales: Algunas reglas especiales, como el emparejamiento a partir de una base de la tripulación, no pueden utilizarse durante la noche en los mismos aeropuertos de la ciudad, etc., para optimizar la utilización de la tripulación.

Período de descanso nocturno: Se proporciona al final de un deber. Este descanso nocturno siempre es más largo que el tiempo de descanso y está restringido por un límite mínimo y máximo.

2.1.3 Hormigas Atta



a) Proceso de búsqueda de la hormiga exploradora; b) Hormiga exploradora depositando rastro de feromona; c) Hormiga seleccionadora identificando hojas adecuadas para el cultivo; d) Hormiga cultivadora realizando el proceso de cultivo

Figura 5 Descripción hormigas Atta. Adaptada de [39]

Las hormigas arrieras o cortadoras de hojas también conocidas como *Atta* son agentes sociales con un comportamiento altamente evolucionado, la división de castas, la construcción y mantenimiento de los hormigueros su comunicación bioquímica la formación compleja de cámaras subterráneas interconectadas, son algunas de las características y comportamientos biológicos de la hormiga Atta. Una de la principal característica por la que se destacan es por su sistema de agricultura, donde cultivan un hongo dentro de sus nidos en cámaras especiales el cual usan como alimento para la colonia, desarrollan un sistema sanitario de prevención de sustancias que puedan afectar al hongo y a la colonia, asociado a bacterias que mantienen sus cultivos libres de otros hongos parásitos, realizan un buen manejo de control de enfermedades, generando diferentes estrategias como aislamiento, monitoreo y administración de los cultivos evitando parásitos que infectan el cultivo [33].

La actividad de una colonia de hormigas arrieras puede dividirse en tres etapas: corte y transporte de material vegetal, cultivo del hongo y disposición final de desechos [33]. Cada etapa es realizada por un tipo de hormiga en especial (castas).

Las hormigas han desarrollado una increíble capacidad de división del trabajo, la cual se expresa en la existencia de diferentes “castas morfológicas”, o grupos de hormigas que son de formas muy diferentes entre sí. Esta diferencia morfológica dentro de una misma colonia representa una ventaja para la vida en sociedad preservando el bienestar de la colonia [31], pues cada tamaño se dedica a la tarea que su forma le permite realizar más eficientemente [37]. Entre estas castas se encuentran:

La casta *exploradora (forrajeras)* es la hormiga encargada de realizar el proceso de búsqueda y recolección (forrajeo) de fragmentos de hojas o restos vegetales apetecibles para la colonia. El material es transportado al nido y luego procesado como alimento para los hongos el cual constituye el alimento básico de la colonia [30]. Todo este proceso lo realizan siguiendo una pista química llamada feromona. La cantidad de hormigas que se reclutan para realizar todo este proceso varía mucho según la calidad de las hojas disponibles, además de la especie y la ubicación de la colonia. La calidad de las hojas es compleja de medir debido a que existen muchas variables, entre las que se incluyen "la ternura de las hojas, la composición de nutrientes, la presencia y cantidad de sustancias químicas secundarias de las plantas" [30].

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

La casta *cultivadora* es una hormiga más pequeña que se encarga del proceso de cultivo del hongo, la cual ayuda al crecimiento de este, siendo la principal fuente de alimento de la colonia. La hormiga se encarga de triturar los fragmentos o material forrajeado el cual moldean en gránulos húmedos donde añaden gotitas fecales. Luego arrancan hebras sueltas de hongos de parches densos y las plantan en la superficie de la pila recién hecha. Las hormigas se mueven y mantienen el jardín pinchando delicadamente las pilas con sus antenas, lamiendo las superficies y arrancando las esporas e hifas de especies de moho no deseadas [37].

En síntesis, el proceso del cultivo del hongo comienza con limpiar el piso de la cámara donde se va a establecer el cultivo o jardín fungoso y la ejecución sistemática de una serie de acciones de comportamiento relacionadas con el mantenimiento del jardín [37] [30].

1. Llevar el pedazo de hoja al nido.
2. Examinar, lamer y limpiar los fragmentos de hoja al agregar sustancias antibióticas.
3. Cortar la hoja en pedazos pequeños.
4. Preparar la pulpa o papilla con la hoja.
5. Colocar la pulpa o papilla sobre el jardín fungoso.
6. Podar el jardín.
7. Cortar el hongo del jardín.
8. Sembrar el hongo en la pulpa de la hoja.

Desde un punto de vista ecológico, la simbiosis entre las hormigas cortadoras de hojas y el hongo que cultivan es de tipo nutricional [30].

2.1.4 Descripción de la solución en el contexto natural de las hormigas

Partiendo del comportamiento natural de las hormigas Atta cultivadora de hongos, de la forma simbiótica de trabajo, de sus castas, de las funciones y actividades que éstas adoptan dentro de una colonia de hormigas, se adapta la propuesta de hormigas artificiales para encontrar mejores soluciones al problema del emparejamiento de la tripulación.

- ***Hormiga exploradora:*** En el ámbito natural es la hormiga encargada de realizar la búsqueda de alimento para la colonia, lleva fragmentos de hojas, flores y/o frutos de diversas plantas. Se comunican con el resto de las hormigas exploradoras por medio de un rastro de feromona que van dejando a su paso.

  La hormiga artificial exploradora en el algoritmo propuesto es la encargada de realizar una búsqueda de soluciones de manera aleatoria. Se encarga de realizar el recorrido por los nodos de manera óptima, haciendo uso del rastro de la feromona.

- ***Hormiga seleccionadora:*** En el ámbito natural es la casta de hormiga encargada de detectar, seleccionar y limpiar el material forrajeado antes de que entre a la cámara de cultivo.

  La hormiga seleccionadora en el algoritmo propuesto se encarga de descartar las rutas poco óptimas por cada iteración, y permite que ingresen al cultivo las mejores soluciones (mejor solución por iteración local). Una vez que las soluciones encontradas pasan por la limpieza y cumplen con las restricciones, son llevadas al cultivo por lo tanto las soluciones encontradas son aptas para cultivar el hongo.

- ***Hormiga cultivadora (Jardinera):*** En el ámbito natural es la encargada de cultivar el hongo, alimentan al resto de la colonia realizando un buen manejo de control de enfermedades generando diferentes estrategias, como aislamiento, monitoreo y administración de los cultivos evitando parásitos que infectan al cultivo.

  En la solución propuesta es la encargada del proceso de monitoreo del cultivo el cual se define dos aspectos fundamentales que son el registro de la ejecución y el de incidentes, estos dos procesos se realizan con el fin de simular la correcta operación del cronograma de los emparejamientos (soluciones). El registro de incidentes es el encargado de simular un incidente operativo y de las actividades posteriores que se generan una vez ocurrido el incidente como aislar nodos infectados y reorganizar si es posible.

- ***Cultivo del Hongo:*** El cultivo se realiza con el fin de que las mejores soluciones que se encuentran de cada iteración global sean cultivadas, en este proceso de cultivo se pueden dar otros procesos como el de selección o clasificación de la hoja, en nuestro problema seleccionamos tres tipos de hojas que dependen del número de bases de donde parten las hormigas o empieza un emparejamiento. Junto con la hormiga cultivadora se encarga de monitorear y administrar el cultivo identificando trayectos que pueden ser nocivos para el cultivo.

CAPÍTULO 3. ESTADO DEL ARTE

El objetivo de este capítulo es el estudio del estado del arte de las metaheurísticas utilizadas para abordar el problema del emparejamiento de la tripulación, mediante una revisión de la literatura a partir de bases de datos reconocidas. Con la revisión se busca encontrar tendencias de investigación de cuáles son los algoritmos utilizados y cuales presentan mejores resultados de solución. Posteriormente se presenta el proceso y criterios de selección de los artículos, la metodología de revisión, los resultados y el análisis.

3.1 REVISIÓN LITERARIA

El objetivo de este estudio es examinar el estado del arte de las metaheurísticas utilizadas para abordar el problema del emparejamiento de la tripulación, analizando las propuestas existentes. Teniendo en cuenta que un estudio de tipo mapeo sistemático permite encontrar tendencias de investigación, se hizo un estudio de este tipo siguiendo los lineamientos propuestos en [28]. Para dirigir la revisión se hace indispensable plantear las siguientes preguntas de investigación

- ¿Qué metaheurísticas se utilizan para resolver el problema del emparejamiento de la tripulación?
- ¿Qué tipo de datos se utilizaron para validar las metaheurísticas?

En la Tabla 1 se muestra la cadena de búsqueda básica formada de un conjunto de palabras clave utilizadas para encontrar artículos referentes a nuestro tema en estudio.

Tabla 1 - Cadena de búsqueda básica

Cadena de búsqueda básica
(Algorithm OR metaheuristic OR metaheuristic hybrid) AND (optimization) AND (crew pairing OR crew scheduling)

Fuente: Los autores

Las fuentes con las que se llevó a cabo el mapeo sistemático fueron: I) bases de datos científicas: ScienceDirect, IEEE Xplore, Scopus y II) literatura gris: documentos entregados por expertos. Además, la ventana de tiempo establecida para este mapeo fue desde el 2009 hasta el 2019.

A continuación, se ejecutó la cadena de búsqueda en cada una de las bases de datos científicas con el fin de encontrar los estudios relevantes (candidatos potenciales a convertirse en estudios primarios). Para determinar si un estudio era relevante, se hizo un análisis del título, resumen, técnica utilizada y palabras clave de cada estudio obtenido como resultado de la búsqueda. Dicho análisis se centró en determinar si cumplía con los siguientes criterios de inclusión:

- Estudios enmarcados en las preguntas planteadas.
- Estudios publicados en revistas nacionales o internacionales o en eventos indexados.
- Artículos publicados en los últimos 9 años.
- Utilización de una metaheurística.
- Utilización de datos reales
- Estudios que describan con suficiencia la metaheurística, la función de evaluación, las restricciones y los resultados obtenidos con las escalas y las unidades de medición que fueron utilizadas.

Como criterios de exclusión se consideraron:

- Estudios que no describieron con suficiencia la metaheurística, la función de evaluación, las restricciones y los resultados obtenidos con las escalas y unidades de medición que fueron utilizadas.
- Estudios que aborden el problema con datos simulados.
- Estudios que no tengan repositorio de código y conjunto de datos accesibles.

Con los estudios relevantes seleccionados, nuevamente se aplicaron criterios de inclusión y exclusión, pero esta vez analizando cada estudio en su totalidad. Los estudios seleccionados luego de este análisis se convirtieron en estudios primarios.

La selección de estudios siguió un procedimiento iterativo e incremental. Este procedimiento se implementó buscando extrayendo y visualizando los resultados de cada fuente de búsqueda iterativa. De esta forma, el informe de revisión creció y evolucionó cada vez más hasta que se completó, obteniendo así el informe final.

En la Tabla 2 se muestra el consolidado de los estudios analizados y seleccionados en cada una de las iteraciones de la revisión. En total se encontraron 1212 estudios, los cuales se obtuvieron como resultado de ejecutar la cadena de búsqueda en las bases de datos científicas. Siguiendo la metodología planteada anteriormente se seleccionaron 21 estudios relevantes y 15 estudios primarios.

Tabla 2 - Estudios analizados y seleccionados

Fuente	**Estudios encontrados**	**Estudios relevantes**	**Estudios primarios seleccionados**
1- Science Direct	1132	12	9
2- IEEE Xplore	56	5	3
3- Scopus	24	4	3
Total	1212	21	15

3.2 OPTIMIZACIÓN POR ALGORITMOS HEURÍSTICOS

A continuación, en la Tabla 3 se clasifican los estudios de acuerdo con el tipo de metaheurística que proponen los autores en los estudios primarios encontrados.

Tabla 3 - Tipo de Metaheurística

ID	**Tipo de Metaheurística**
E1	Genetic algorithm
E2	Ant colony optimization algorithm
E3	Particle swarm optimization algorithm
E4	knowledge based random algorithm
E5	Variable neighborhood search algorithm
E6	Genetic algorithm
E7	Genetic algorithm
E8	Branch-and-price heuristic
E9	Algorithm Column generation
E10	compact mixed-integer programming model
E11	Algorithm Column generation
E12	Constraint programming and Ant colony optimization algorithm
E13	Genetic algorithm NSGA II (Non-Dominated sorting Genetic Algorithm
E14	Mixed Integer Programming
E15	Branch and price Heuristic variant retrospective branching

A continuación, en la Tabla 4 se muestran las diferentes estrategias de validación que se utilizaron en cada metaheurística.

Tabla 4 - Estrategias de validación

ID	Estrategia de Validación
E1	Se hizo un estudio de caso con datos de una aerolínea
E2	Se realizó un experimento controlado con datos de compañías aéreas.
E3	Se realizó un experimento con datos sintéticos.
E4	Se realizó un experimento con datos sintéticos.
E5	Se realizó un experimento con datos sintéticos.
E6	Se realizó un experimento con datos sintéticos.
E7	Se hizo un estudio de caso con datos de una aerolínea.
E8	Se realizó un experimento con datos de una aerolínea norteamericana.
E9	Se realizó un experimento con datos de una aerolínea francesa.
E10	Se realizó un experimento con datos de una aerolínea.
E11	Se realizó un experimento con datos reales públicos de una aerolínea.
E12	Se realizó un experimento con datos tomados de la biblioteca OR
E13	Se realizó un experimento con datos de una aerolínea aérea
E14	Se realizó un experimento con datos de una aerolínea aérea
E15	Se realizó un experimento con datos de una aerolínea aérea

3.3 TRABAJOS RELACIONADOS

De la forma como las empresas del transporte aéreo aumenta su demanda de clientes también deben invertir en su compañía extendiendo tanto la logística como el número de flotas que conforma para poder abordar la cantidad de vuelos que se le presenta, esto conlleva a que el emparejamiento de la tripulación que es uno de los principales problemas de la planificación de la industria también incremente su complejidad, de acuerdo con [11], las empresas dedican buena parte de sus recursos para solucionar esta problemática y alquilan software a gran costo de fuentes externas. Se utilizan rigurosos modelos matemáticos y algoritmos para resolver estos problemas.

De este modo a continuación se hace un recuento de las investigaciones más recientes que han trabajado el problema del emparejamiento de la tripulación con diferentes algoritmos de optimización y sus resultados obtenidos:

En 2009 [8] se desarrolló un algoritmo genético para optimizar el costo total del emparejamiento de la tripulación donde se desea integrar con el siguiente problema que es la asignación de la tripulación, debido al tamaño del problema se prefirió resolver el mismo en dos subproblemas, la inicialización de la población para el algoritmo se fija de dos formas: inicialización basada por programación lineal y basada en heurística, se representa el individuo por medio de una matriz donde las columnas son los emparejamientos y las filas son los días de la semana, con valores 0 si al miembro de la tripulación no se le asigna una tarea durante ese día o -1 si no está disponible. Este caso de estudio utilizó como conjunto de datos tres instancias que son proporcionadas por la compañía aérea "Air-Algérie". Los autores muestran que los resultados en el modelo integrado fueron mejor en una de tres instancias, aunque no existe un punto de referencia donde puedan validar este modelo tras haber pocos estudios sobre esta integración, lo evaluaron con las soluciones hechas por la compañía aérea.

En 2011 [13] se desarrolló un algoritmo de optimización de colonia de hormigas (ACO) para encontrar el conjunto óptimo de emparejamientos, teniendo en cuenta las restricciones que conlleva este problema, se usaron 28 conjuntos de datos de prueba de casos reales de compañías aéreas. Se realizó un experimento controlado, esta prueba computacional el algoritmo ACO se modelo basado en el problema del vendedor ambulante y se comparó con un algoritmo genético (GA), ambos se ejecutan en condiciones experimentales idénticas. Los autores

concluyen que los resultados obtenidos por el algoritmo basado en ACO tiene un rendimiento notable para minimizar los costos totales de la tripulación al programar efectivamente los vuelos para reducir la estadía en hoteles, los tiempos muertos que se refiere cuando la tripulación viajan como pasajeros para volver a su origen desde donde inició el emparejamiento y el tiempo de espera de las compañías aéreas, el ACO obtuvo el costo promedio más bajo que el GA en 22 de los 28 conjuntos de datos disponibles

En 2013 [12] se desarrolló un algoritmo híbrido de optimización de enjambre de partículas sincronizado con una heurística de búsqueda local para optimizar el conjunto secuencial de vuelos, el algoritmo crea una lista de prioridades en un vector que tiene el número de vuelos en total y en el que cada elemento muestra la prioridad del vuelo correspondiente, además, proporciona toda la información, como las ciudades de salida, llegada y los tiempos necesarios para verificar la legalidad de los emparejamientos y por lo tanto la viabilidad de la solución. Se realizó un experimento controlado utilizando como conjunto de datos 20 problemas generados aleatoriamente que consisten en 25, 50, 100 y 150 tramos de vuelo, se compara con dos algoritmos de la literatura, un algoritmo genético híbrido y un algoritmo de colonia de hormigas, los autores concluyen que los resultados obtenidos por el algoritmo de optimización de enjambre de partículas son significativamente mejores que los del algoritmo genético híbrido así mismo que con el algoritmo de colonia de hormigas en términos de soluciones promedio en cuanto a tiempo de procesamiento.

En 2013 [15] se desarrollaron dos algoritmos, uno basado en conocimiento y un algoritmo de generación de columnas con el objetivo de optimizar el número de emparejamientos de la tripulación. El primer algoritmo genera un espacio de solución reducido a través del conocimiento recibido y lo utiliza como entrada inicial del algoritmo de generación de columnas, este último representa el individuo como un gen donde se simboliza un tramo de vuelo, por lo tanto, toda la información (ciudades, horarios de salida y llegada) podría conocerse fácilmente para verificar la legalidad de un emparejamiento. Se realizó un experimento controlado sobre un conjunto de datos hipotéticos generados que consisten en 25, 60, 100, 150 y 200 tramos de vuelo respectivamente. Los autores concluyen que es importante la entrada inicial ya que afecta el rendimiento de la técnica de generación de columnas resultando con tiempo de procesamiento mucho menor.

En 2017 [22] se implementó un algoritmo basado en la búsqueda de vecindario variable con el objetivo de minimizar el número de tripulaciones de cabina necesarias para cubrir todos los vuelos sujetos a un conjunto de restricciones. Los autores realizan dos procedimientos en este problema, el primero llamado shake recibe una solución inicial y comienza a formar subgrupos de vuelos y unirlos hasta que se pueda formar un emparejamiento, el segundo llamado “descenso de vecindario variable” toma dos emparejamientos e intercambia algunos vuelos entre los diferentes emparejamientos para buscar una solución óptima. Se realizó un experimento controlado donde se compara con un algoritmo llamado multi-start de la literatura sobre un conjunto de datos de 10 instancias sintéticas que tienen una solución inicial, estas instancias tienen diferente número de vuelos para analizar el comportamiento de los algoritmos. Los autores concluyen que el algoritmo propuesto mejora la solución inicial en 8 de 10 instancias y obteniendo mejores resultados en el número total de emparejamiento que el algoritmo multi-start, así como en el rendimiento en tiempo de procesamiento.

En 2017 [23] se implementó un algoritmo genético para tener el menor número de emparejamientos de tripulación con el fin de bajar los costos relacionados con estos, a diferencia de otros artículos relacionados, este artículo toma una población inicial de un emparejamiento factible y busca generar un subconjunto aleatorio basado en el conocimiento para poder cubrir todos los tramos de vuelos del emparejamiento generado, y luego este subconjunto continúa renovándose. Cada subconjunto tiene un costo relacionado y la función objetivo se centra en encontrar el menor costo de todos los diferentes subconjuntos. Se realizó un experimento controlado en el que se compara el algoritmo propuesto con dos algoritmos genéticos de la literatura para esto se utiliza el mismo conjunto de datos que es proporcionado por una aerolínea de Turquía con 591, 906 y 1002 tramos de vuelo mensuales, los autores concluyen que el algoritmo propuesto genera mejores resultados en cuanto al costo total de la tripulación, así como el menor número de emparejamientos frente a los otros dos algoritmos.

En 2018 [1] se implementaron tres tipos de algoritmos: un algoritmo genético, un algoritmo genético híbrido y un algoritmo memético para optimizar el costo total del conjunto secuencial de vuelos donde cada emparejamiento es representado como un cromosoma el cual tiene un costo asociado, en este caso de estudio se comparan los tres algoritmos utilizando como conjunto de datos los suministrados por una compañía aérea de Turquía, los resultados obtenidos muestran que el algoritmo memético obtiene el menor tiempo de procesamiento para todas las instancias frente a los algoritmos genéticos. Luego se compara el algoritmo memético con dos algoritmos

genéticos de la literatura y nuevamente resulta el algoritmo memético obteniendo el menor costo, aunque esta vez no muy significativo. Por lo tanto, los autores concluyen que se obtiene un horario de la tripulación con un costo más bajo utilizando el enfoque del algoritmo memético.

En [25] se implementó la heurística Branch-and-price para obtener el menor número de emparejamientos de tripulación teniendo como una nueva restricción el idioma de la tripulación. Se resuelve por medio de dos subproblemas, uno encontrando los emparejamientos sin la restricción del idioma y otro con la restricción del idioma, finalmente se incrusta un algoritmo de rolling-horizon para descomponer el primer subproblema en ventanas de tiempo más pequeñas y así encontrar soluciones a estas para posteriormente unificarlas, en este experimento se compara con el problema del emparejamiento de la tripulación básica sin restricciones de idioma, utilizando un conjunto de datos de una aerolínea norteamericana encontrados en la literatura. Los autores concluyen que la heurística propuesta con la restricción del idioma obtiene en promedio entre el 34% y 97% mayores tiempos de ejecución a comparación con el problema sin esta restricción, sin embargo, la heurística obtiene emparejamientos de mejor calidad sin violar restricciones en un 40% a comparación con el problema del emparejamiento de la tripulación básico.

En [26] se implementó un algoritmo basado en generación de columnas para abordar e integrar el problema de ruta de aeronaves y el problema del emparejamiento de la tripulación con el fin de obtener el menor número de emparejamientos y reducir los costos de la tripulación, para resolver estos problemas se empezó con un problema maestro el cual es una relajación lineal del problema original, se genera solo un número pequeño de columnas para tener una solución factible para el problema relajado, después hay un subproblema que permite identificar variables adicionales que no han sido incluidas en el problema maestro y estas ayudan a mejorar el valor de la función objetivo, en este experimento se utiliza un conjunto de datos real proporcionado por una aerolínea francesa, los autores concluyen que este método puede resolver instancias más grandes a diferencia de otros métodos.

En [27] se desarrolló un modelo de solución buscando unificar dos problemas de manera simultánea como lo son el enrutamiento de aeronaves y el emparejamiento de la tripulación, este modelo utiliza la técnica Reformulation-Linearization (RLT) y una característica es que incluye un número polinomial de variables y restricciones, teniendo como función objetivo mantener la tripulación inicial en la misma flota durante todo el emparejamiento, en este experimento se utiliza

un conjunto real de datos proporcionados por diferentes aerolíneas de Europa, los autores concluyen que este modelo ofrece con éxito soluciones rentables obteniendo en un 71% la flota al final del emparejamiento con su tripulación inicial y obteniendo un modelo con tiempos computacionales muy moderados.

En [5] se implementó un algoritmo basado en generación de columnas para integrar el problema del emparejamiento de la tripulación y la lista de la tripulación, se resolvió por medio de relajación lineal seguido de encontrar una solución entera utilizando dos estrategias de ramificación las cuales son fijación de columnas y fijación heurística entre tareas, calculando una puntuación en cada nodo de ramificación para luego comparar y seleccionar la estrategia de ramificación con el mayor puntaje, se utilizó un conjunto real de datos proporcionados por una aerolínea estadounidense, los autores concluyen que los tiempos de ejecución en cada una de las 7 instancias fueron razonables y se puede lograr un alto nivel de satisfacción de la tripulación cuando se construyen cronogramas de trabajo a través de un enfoque integrado.

CAPÍTULO 4. FORMULACIÓN Y DESARROLLO DEL ALGORITMO PROPUESTO

En este capítulo se da a conocer el proceso de diseño del algoritmo metaheurístico Aco Fungus adaptado al problema del emparejamiento de la tripulación. Describiendo cada proceso de desarrollo y presentando el algoritmo propuesto.

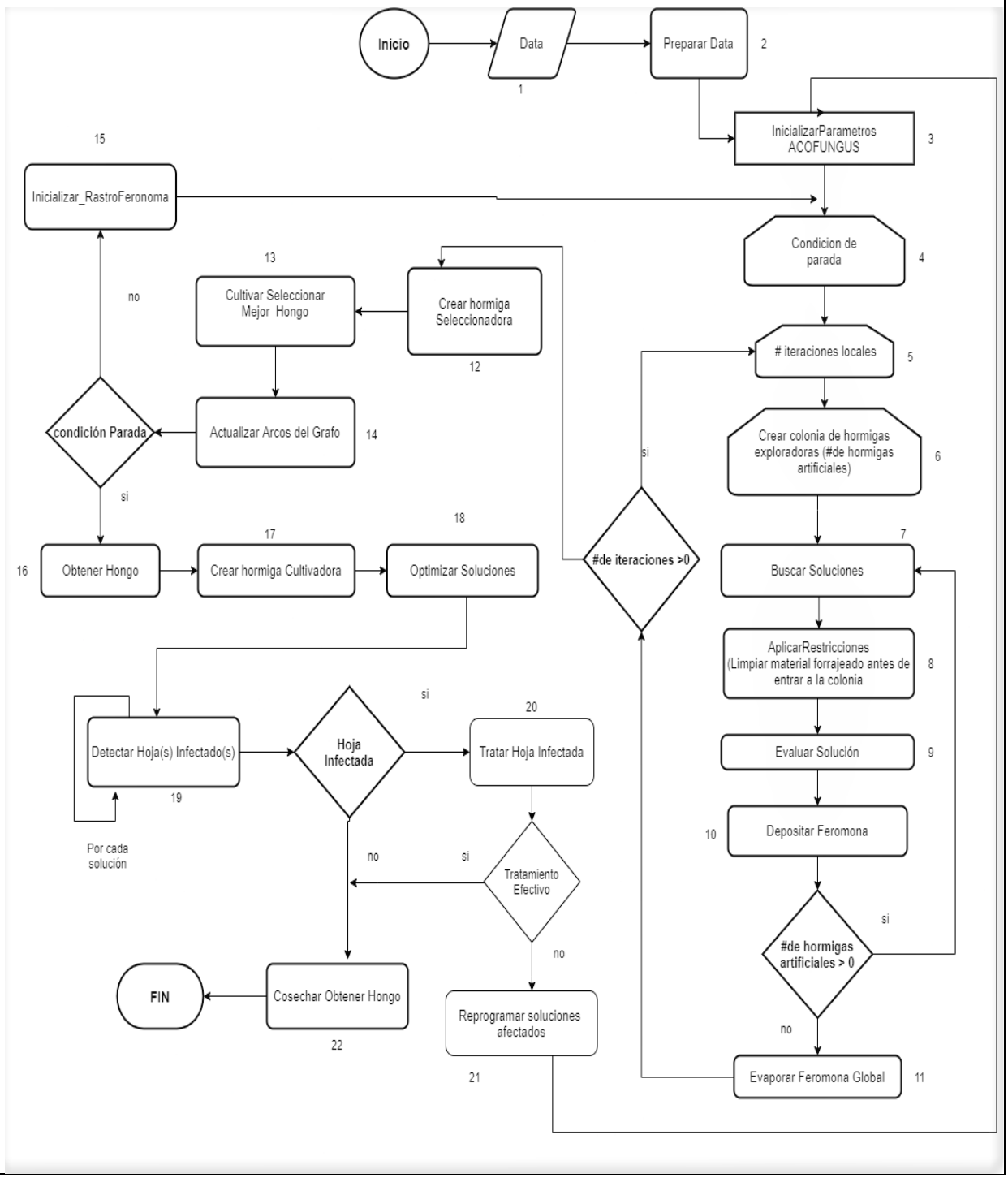
4.1 DISEÑO ALGORITMO ACO-FUNGUS

Para la construcción de la propuesta se toma como base el proceso desarrollado por las hormigas Atta descrito en el capítulo 2 sección 2.2.3, y el framework de la metaheurística ACO propuesta en [38].

4.1.1 Diagrama de flujo de la solución propuesta

Proceso	**Elaborado Por**: Julian Montero	**Fecha**
Variante Algoritmo ACO Fungus	**Estatus**: Propuesto	**Elaboración**: agosto 2021

FLUJOGRAMA DE ACTIVIDADES



Validado Por:	**Fecha Presentación:** septiembre 2021

Figura 6: Representa el diagrama de flujo propuesto para la variante de algoritmo Aco - Fungus. *Fuente: Autores.*

4.1.2 Descripción de las actividades y operaciones del algoritmo

Tabla 5 – Descripción de actividades algoritmo ACO-FUNGUS

Descripción de actividades algoritmo ACO-FUNGUS		
No Proceso	**Nombre Proceso**	**Objetivo**
1	**Data**	Es el conjunto de datos o instancias. Sirve como parámetro de entrada para el algoritmo.
2	**Preparar data**	Se configura o se transforma la data de acuerdo con las necesidades del problema.
3	**Inicialización parámetros de entrada Aco-Fungus**	Se crean e inicializan todos los parámetros necesarios para que el algoritmo realice las operaciones necesarias. Los parámetros necesarios se describen a continuación: ● Creación de un grafo de conexión. Cada vértice va junto a una matriz o lista de adyacencias junto con el peso de cada arista a la cual cada hormiga puede acceder para la construcción de una solución. ● Se define el parámetro alfa el cual se define como parámetro de influencia de la feromona. ● Se define un parámetro beta el cual se define como un parámetro de importancia heurística.

		● Se define un número de hormigas que harán el recorrido en el grafo. ● Se define un número de iteraciones, el cual define la cantidad de veces que cada hormiga artificial va a recorrer el grafo. ● Se define la tasa de evaporación para la feromona. ● Se define un aprendizaje Q, como estrategia para compensar los parámetros alfa y beta. ● Se inicializa y deposita cierta cantidad igual de feromona para cada arista o conexión esto con el fin de que todos los posibles caminos que pueda escoger una hormiga tengan la misma probabilidad de elección.
4	**Condición de parada:**	La condición de parada se ajusta a cada problema. Se toma como una iteración global al no cumplirse la condición. Y está dada si existen aristas activas en la representación del grafo.
5	**Número de iteraciones locales**	Se define como número de iteraciones locales al número de veces que cada hormiga va a recorrer el grafo en búsqueda de soluciones.
6	**Creación colonia de hormigas exploradoras artificiales**	Se crea un número determinado de hormigas exploradoras artificiales.
7	**Buscar soluciones**	En este proceso se envía a cada una de las hormigas exploradoras artificiales a recorrer el grafo. Las cuales tienen como objetivo recorrer un grafo y buscar soluciones óptimas al problema.

8	**Aplicar restricciones**	En este proceso se definen y aplican todo el conjunto de restricciones del problema a resolver. Se penaliza al conjunto de restricciones que no se cumplan. Cada una de las restricciones trae consigo una penalización que sirve como argumento para decidir la cantidad de feromona que deposita cada hormiga según el valor de penalización.
9	**Evaluar solución**	Se evalúa la calidad de la solución generada por cada hormiga al haberle aplicado las restricciones del problema.
10	**Depositar feromona**	Cada hormiga deposita cierta cantidad de feromona. La cantidad de feromona a depositar dependerá de que tan buena sea la solución encontrada, si la hormiga ha encontrado un camino óptimo donde se cumplan las restricciones dejará un rastro de feromona mayor con el fin de que nuevas hormigas tengan esta información y así más probabilidad de ir por ese recorrido.
11	**Evaporar feromona**	Es el proceso por el cual se evitan los óptimos locales, se evapora cierta cantidad de feromona de cada vértice por cada iteración local, la tasa de evaporación se define como parámetro de entrada.
12	**Crear Hormiga Seleccionadora**	Se crea una hormiga seleccionadora artificial. La cual tiene como objetivo de elegir la o las mejores soluciones por iteración.
13	**Cultivar hongo**	Una vez que las soluciones pasan por la limpieza y cumplen con las restricciones son llevadas al cultivo.
14	**Actualizar arcos del grafo**	Consiste en actualizar los arcos del grafo. La actualización elimina arcos del grafo y la elección de

		los arcos se da por la mejor solución de una iteración global.
15	**Inicializar rastro feromona**	Se realiza de nuevo el proceso de inicializar y depositar cierta cantidad igual de feromona para cada arista o conexión esto con el fin de que todos los posibles caminos que pueda escoger una hormiga tengan la misma probabilidad de elección. Esta inicialización se hace cada vez que no se cumpla la condición de parada se toma como una iteración global del algoritmo.
16	**Obtener hongo**	Terminada el número de iteraciones se procede a obtener todas las soluciones que se han cultivado.
17	**Crear hormiga Cultivadora**	Se crea una hormiga cultivadora artificial, encargada de monitorear el cultivo y optimizar las soluciones.
18	**Optimizar soluciones**	Proceso por el cual se busca optimizar las soluciones que se encuentran cultivadas.
19	**Detectar hojas infectadas**	Una vez el hongo ha sido cultivado se debe de tener un monitoreo de este, con el objetivo de detectar hojas infectadas.
20	**Tratar hoja infectada**	Proceso por el cual se aíslan las hojas que se ven afectadas por la infección del cultivo y se busca la agregación de las hojas afectadas a otras soluciones siempre y cuando se cumplan con las restricciones del problema.

21	**Reprogramar soluciones afectadas**	Si se han detectado hojas infectadas y no se logró un tratamiento efectivo en el proceso de tratar hoja infectada, todo el hongo se deja de cultivar y se envía a buscar nuevas soluciones y seguir todos los procesos anteriormente descritos.
22	**Obtener hongo final**	El hongo contendrá todos los emparejamientos una vez se hayan reprogramado los vuelos afectados si es que los hay. Este es el resultado final si en el cultivo no se encuentran hojas infectadas.

4.1.3 Descripción de las condiciones de inicio y parada

Las condiciones de Inicio están dadas por cada problema que se quiera optimizar. En general la(s) condiciones de inicio se establecen en tener la data necesaria a optimizar y de la creación e inicialización de los parámetros necesarios para que el algoritmo realice las operaciones correspondientes.

Las condiciones de terminación están dadas por varios factores si en la representación de grafo no existen aristas activas por la que la hormiga pueda recorrer o si el algoritmo no puede generar soluciones óptimas después de cierto número de iteraciones.

4.1.4 Pseudocódigo del Algoritmo Propuesto

```
Algoritmo AcoFungus
        paso 1 Inicialización
        crear_inicializar_parametros_de_entrada
        inicializar_rastro_feronoma

        paso 2 Iteración
        Mientras criterio_de_terminacion Hacer
                Para 0 Hasta n_iteraciones Con Paso 1 Hacer
                        S = generar_Soluciones()
                        evaporar_feromona_manera_global
                Fin Para
                crear_hormiga_seleccionadora
                seleccionar_mejor_solucion(S)
                cultivar_hongo_mejor_solucion
                actualizar_numero_de_vertices
                inicializar_rastro_feronoma
        Fin Mientras
        paso 3 Cosechar
        solución = cosechar_Mejores_Soluciones_generadas()
        mostrar solución

FinAlgoritmo


Función generar_soluciones()
        n_ants = crear_hormigas_exploradoras
        S = caminos_generado_por_hormigas_exploradoras
        Para n Hasta n_ants Con Paso 1 Hacer
                nodoIncial = Elegir_nodo_incial
                M = camino_generado_por_hormiga[n]
                Mientras nodoSiguiente <! nodoIncial Hacer
                        P = calcular_probabilidades_transicion(nodo)
                        nodoSiguiente = aplicar_criterio_Eleccion(P)
                        M = actualizar_camino_generado_por_hormiga[n](nodoSiguiente)
                Fin Mientras
                P = evaluar_aplicar_Restricciones(M)
                depositar_feromona_arcos_visitados(M, P)
                S = actualizar_camino_generado_por_hormiga(M)
        Fin Para
FinFuncion retornar S


Funcion cosechar_Mejores_Soluciones_generadas()
        cosecha =  cosechar_mejores_soluciones_generadas[]
        crear hormiga cultivadora
        cosecha_optima = optimizar soluciones()
        Mientras cosecha_optima == Verdadero Hacer
                verificar_infeccion_en_cultivo
                Si verificar_infeccion_en_cultivo == Verdadero Entonces
                        registrar_infeccion
                        aislar_nodos_hongo_infectados
                        solucionar_infeccion_del_cultivo
                        Si solucionar_infeccion == Verdadero Entonces
                                obtener_cosecha_de_hongo
                        SiNo
                                retornar Paso 2 Iteracion
                        Fin Si
                SiNo
                        ObtenerHongo
                Fin Si
        Fin Mientras retornar cosecha_hongo
Fin Funcion
```

.

Figura 7: Representa el pseudocódigo propuesto para la variante del algoritmo Aco – Fungus.

Fuente: Autores

4.1.5 Descripción Pseudocódigo

Tabla 6 – Descripción pseudocódigo

Descripción pseudocódigo		
Proceso o función	***Objetivo***	***Fórmula***
Inicializar *Parámetros*	Se crean e inicializan todos los parámetros de entrada necesarios para el algoritmo descrito anteriormente.	
Inicializar Rastro feromona	Se inicializa en todas las aristas una cantidad pequeña de feromona. Para toda conexión i j existente se inicializa un rastro de feromona (θ).	$\forall ij \ C_{xy} = \theta$ Donde C_{xy} = conjunto de todas las conexiones XY existentes. θ = porcentaje inicial de feromona
Criterio de terminación	En general el criterio de terminación está dado si en la representación de grafo no existen aristas activas por la que la hormiga pueda recorrer.	$C_{xy} \neq 0$
Crear hormigas exploradoras	Se crean el número de hormigas exploradoras que van a recorrer el	

	grafo. El número de hormigas a crear depende de la inicialización que se le da al inicio del algoritmo en los parámetros de entrada.	
Caminos generados por hormigas exploradoras	Lista que va a contener todas las soluciones que genera cada hormiga. Se utiliza como parámetro de entrada para procesos posteriores.	
Elegir nodo inicial	Es el proceso por el cual se sitúa a cada hormiga en un vértice o nodo de inicio, si se tiene más de un nodo de inicio esta asignación se realiza de forma aleatoria para el conjunto que contiene los vértices o nodos de inicio Cni.	$aleatorio\ Cni$
Camino generado por hormiga	Lista en la que se ingresan todos los nodos que la hormiga ha visitado durante la búsqueda de la solución.	

Calcular probabilidades de transición de nodo	Es una función que calcula la probabilidad de elección de un nodo alcanzable por la hormiga. Recibe como parámetro una lista de adyacencia la cual es la lista de nodos que la hormiga puede visitar. Viene dada por la siguiente ecuación:	$P_{xy}^{k} = \{\frac{[r_{xy(t)}]^{\propto}[n_{xy(t)}]^{\beta}}{\sum_{y \in N_x^k} [r_{xy(t)}]^{\propto}[n_{xy(t)}]^{\beta}} \; si\ y \in N_k(r)$ Donde: t = se refiere a un tiempo t o iteración. P_{rs}^{k} = probabilidad de la hormiga k, de ir a x → y . $\propto$ = parámetro de influencia de la feromona. β = parámetro de influencia de la heurística. $N_k\ (x)$ = son todos los nodos alcanzables por la hormiga k desde el nodo x. $r_{xy(t)}$ = representa el rastro de feromona entre los vértices x e y. $n_{xy(t)}$ = representa el valor de la función heurística elegida, deseabilidad o visibilidad y viene dada por la por: $\frac{1}{d}$ Donde d representa el peso de la arista.
Aplicar criterio de elección	Una vez generadas las probabilidades a los nodos que la hormiga pueda acceder, se elige el nodo siguiente generando un número	$aleatorio\ P_{xy}^{k}$

	de forma aleatoria dentro de las probabilidades generadas.	
Actualizar camino generado	Se agrega a la lista de nodos recorrido por la hormiga k, el nodo elegido de forma aleatoria.	
Condición de parada:	La condición de parada verifica si el nodo elegido es igual al nodo inicial. Si la condición es verdadera sale del ciclo, en caso contrario sigue buscando nodos a las que la hormiga k pueda elegir y agregándolos a la lista de nodos visitados.	$nodoInicial \,!=\, nodoActual$
Evaluar aplicar restricciones	Una vez que se la hormiga k genera una solución esta es evaluada. La evaluación se da de acuerdo con el cumplimiento de las restricciones del problema. Si alguna restricción no se cumple esta es penalizada. Recibe	

	como parámetro de entrada la solución generada por la hormiga y genera todo el proceso de evaluación y penalización fundamental para procesos posteriores como el de la actualización del rastro de feromona en cada arista visitada.	
Depositar feromona en arcos visitados	Se deposita feromona dependiendo de la calidad de la solución que genera cada hormiga. La cantidad depositada en cada arco no siempre va a hacer la misma ya que depende principalmente de la evaluación y la penalización generada al no cumplir con las restricciones en las transiciones de nodo a nodo.	La cantidad de feromona a depositar está dada por la siguiente ecuación. $\forall_{xy} \{ r_{xy}^{k} = \frac{\sigma}{d_k}$ $d_{k = \sum} \quad pesoArista + evaluacionArista$ Donde: r_{xy}^{k} = rastro de feromona a depositar en el arco de xy por la hormiga k. σ = factor de aprendizaje se ajusta para que la influencia de las estrategias de alfa y beta sean compensadas. La actualización de la feromona se da por la siguiente ecuación. $r_{xy(t)} = r_{xy(t-1)} + r_{xy}^{k}$

		Donde: $r_{xy(t)}$ = Cantidad de feromona actualizada en un tiempo t. $r_{xy(t-1)}$ = cantidad de feromona en un tiempo t-1 r_{xy}^{k} = cantidad de feromona a depositar en un tiempo t.
Actualizar camino generado por hormiga	Se agrega la solución generada por la hormiga k a una lista donde se encuentran todas las soluciones generadas por las hormigas k_n.	
Condición parada iteraciones locales	La condición de parada se da al realizar el número total de iteraciones locales que se definen al inicio de la ejecución del algoritmo.	
Evaporar feromona manera global:	Se evapora la feromona de manera global lo que significa que la evaporación se hace para todos los arcos. La evaporación global se realiza para cada iteración local y cuando todas las hormigas han	La evaporación global se da por la siguiente ecuación. $$r_{xy(t)} = (1 - \theta)\, r_{xy(t-1)}$$ Donde $r_{xy(t)}$ = rastro de feromona actualizado en el arco xy. θ = coeficiente de evaporación $0 < \theta < 1$.

	generado una solución al problema planteado.	$r_{xy(t-1)}$ = rastro de feromona en un tiempo t -1.
Crear hormiga seleccionadora	Se crea una hormiga seleccionadora que es la encargada de seleccionar la mejor solución generada por las hormigas.	
Seleccionar mejor solución	Se selecciona la solución generada por la hormiga que tenga la mejor evaluación. Recibe como parámetro la lista de las soluciones generadas por las hormigas. Y se obtiene como salida la mejor solución generada.	
Cultivar hongo mejor solución	Es el proceso por el cual se lleva la mejor solución que generó la hormiga seleccionadora a un cultivo. Aquí se mantienen todas las mejores soluciones generadas por las hormigas dependiendo de las ***n*** iteraciones	

	globales que se tengan.	
Actualizar número de arcos	Es el proceso por el cual se actualizan los arcos del grafo principal. Tiene como entrada la mejor solución generada por las hormigas durante las iteraciones locales. Estos arcos se desactivan con el propósito de que en nuevas iteraciones las hormigas no puedan acceder o recorrer. La salida que genera es un grafo actualizado sin los arcos de la mejor solución ya cultivados.	
Inicializar rastro de feromona:	Se vuelve a inicializar los arcos del grafo con un rastro de feromona aristas una cantidad pequeña de feromona. Esta inicialización se da por terminada cada iteración global.	Para toda conexión XY existente se inicializa un rastro de feromona (θ). $$\forall xy \;\; C_{xy} = \theta$$ Donde C_{xy} = conjunto de todas las conexiones xy existentes. θ = porcentaje inicial de feromona.

COSECHAR MEJORES SOLUCIONES		
Crear hormiga cultivadora	***Crear hormiga cultivadora:*** Se crea una hormiga cultivadora que es la encargada de optimizar las mejores soluciones de las iteraciones globales y de monitorear el estado del cultivo de soluciones.	
Optimizar soluciones	La hormiga cultivadora artificial de acuerdo con la evaluación de cada solución verifica que soluciones se pueden optimizar. Recibe como parámetro el cultivo de las mejores soluciones encontradas por las hormigas exploradoras y seleccionada por la hormiga seleccionadora. La optimización consiste en tratar de unir dos soluciones en una sola cumpliendo con las restricciones del problema. Retorna el conjunto de soluciones optimizadas.	
Cosecha óptima criterio de parada	Se establece que una cosecha es óptima cuando no se presenta ningún	

	inconveniente en el cultivo de las mejores soluciones. Se define dependiendo del problema que puede afectar a estas soluciones. Si existe un problema en el cultivo se notifica y se procede a solucionarlo.	
Verificar infección en el cultivo	Es el proceso por el cual la hormiga cultivadora se encarga de monitorear las soluciones si presentan algún inconveniente en el proceso de cosecha ya sea por agentes internos o externos.	
Aislar nodos hongos infectados	Una vez detectado una infección por parte de la hormiga cultivadora se procede a aislar los nodos a partir del nodo infectado.	
Solucionar infección del cultivo	Consiste en reorganizar los nodos infectados en las soluciones más cercanas cumpliendo con las restricciones del problema. Si la reorganización de los nodos no se cumple se procede a recoger todas las soluciones del cultivo y se procede a iniciar el proceso de nuevas búsquedas con	

	los nodos faltantes por cosechar dentro del cultivo.	
Obtener hongo	Es el proceso por el cual la hormiga cultivadora ha monitoreado las soluciones y las retorna.	
Mostrar solución	Se encarga de visualizar la o las mejores soluciones al problema planteado.	

4.1.6 Mejoras a la solución original

A continuación, se describe las mejoras realizadas al algoritmo ACO original:

- Estrategia para la actualización de rastro de feromona en los arcos y la cantidad de feromona que las hormigas depositan. La actualización del rastro de feromona se realiza una vez que cada hormiga ha encontrado una solución. A diferencia del algoritmo original donde la actualización del rastro de feromona se realiza por iteración local, en donde primero las *k* hormigas encuentran una solución y después se realiza el proceso de actualizar el rastro de feromona.

  Ahora la cantidad de depósito en cada arco de la solución encontrada por la hormiga depende del cumplimento de las restricciones en cada transición, cada restricción está penalizada si no se cumple. La cantidad se establece por la sumatoria del peso de cada arco y de la suma de las cantidades de penalización en cada transición.

- Estrategia en la búsqueda de las soluciones, se da por medio de una búsqueda local las hormigas recorren los arcos haciendo transiciones entre los vértices hasta encontrar un criterio de parada dependiendo del problema puede adaptarse varios criterios de parada. La estrategia de búsqueda se complementa con la de la de la actualización de rastro de

feromona y por la cantidad de iteraciones locales que se definan, al ser un problema adaptado se tienen dos tipos de iteraciones una local y una global.

- Iteraciones locales y globales, se definieron estos dos conceptos como complemento a la estrategia de búsqueda. Las iteraciones locales se definen o entran como parámetro de entrada al algoritmo, es el número de veces que cada hormiga va a recorrer el grafo, al cambiar la estrategia de rastro de feromona cambia el comportamiento del algoritmo debido a que se necesitan menos iteraciones locales para que las hormigas generen una solución. Las iteraciones globales están dadas por dos criterios y se considera una iteración global cuando, han pasado el número de iteraciones locales y cuando se cumple una condición de parada global que también es ajustable a cada problema.

- Estrategia de elección de las mejores soluciones. Una vez que se ha generado una solución por parte de las hormigas se elige la mejor solución de acuerdo con el valor de la función de evaluación. Los arcos que se encuentran en está mejor solución son desactivados o eliminados del grafo para que no vuelvan a ser recorridos por las hormigas y pasados a un cultivo.

- La condición de parada está dada por la estrategia de elección de la mejor solución, esto se da porque al desactivar nodos las hormigas van quedando con menor espacio de búsqueda. La condición de parada está dada por el número de arcos que estén activos.

- La estrategia de un cultivo, en el planteamiento se genera el concepto de cultivo para monitorear las soluciones generadas el modelo está basado en posibles reprogramaciones de esas soluciones. El cultivo crece o se mantiene por cada solución generada por las hormigas. La optimización que se plantea busca hacer la unión de dos soluciones si es posible.

CAPÍTULO 5. INSTANCIACIÓN DEL ALGORITMO ACO FUNGUS PARA EL EMPAREJAMIENTO DE LA TRIPULACIÓN

En este capítulo se describen todos los procesos para la instanciación del algoritmo Aco - Fungus para el problema del emparejamiento de la tripulación. Describiendo cada proceso de desarrollo de acuerdo con el algoritmo propuesto.

5.1 DESCRIPCIÓN DE PROCESOS

Tabla 7 - Instanciación algoritmo ACO – FUNGUS

para el emparejamiento de la tripulación

DESCRIPCIÓN DE PROCESOS		
No Proceso	**Nombre Proceso**	**Objetivo**
1	***Data***	Para el problema del emparejamiento de la tripulación el algoritmo tiene como entrada una lista de todos los vuelos programados en un horizonte de tiempo no mayor a un mes. Esta lista debe ser proveída por una aerolínea, donde se establece la información relevante de cada vuelo (arista), la cual debe corresponder con los siguientes parámetros: ● Identificación del vuelo ● Aeropuerto de salida ● Fecha de salida ● Hora de salida ● Aeropuerto de llegada ● Fecha de llegada ● Hora de llegada

2	**Preparar data**	Para el problema del emparejamiento de la tripulación inicialmente se crea una lista de adyacencia con los vuelos (arista) para cada vértice (aeropuertos y bases) ordenados de forma cronológica. El peso de cada arista representa el tiempo de vuelo de cada tramo se define como la diferencia que hay entre la hora de llegada y la hora de salida del vuelo (tramo de vuelo). $$peso = Hllegada - Hsalida$$ Esta lista de adyacencia se considera como parámetro de entrada para el algoritmo.
3	**Inicialización parámetros de entrada Aco-Fungus**	La inicialización de algunos parámetros está basada en datos hallados en el estado del arte. Los parámetros necesarios se describen a continuación: ● Creación de un grafo de conexión. Para nuestro problema cada arista representa un tramo de vuelo. Los aeropuertos y bases de dónde despegan y aterrizan los vuelos representan los vértices. Cada vértice va acompañado de una matriz de adyacencias a la cual cada hormiga puede acceder para la construcción de un emparejamiento (secuencia de vuelos). ● Se define el parámetro alfa el cual se define como parámetro de influencia de la feromona. ● Se define un parámetro beta el cual se define como un parámetro de importancia heurística. ● Se define un número de hormigas que harán el recorrido en el grafo.

		● Se define un número de iteraciones, el cual define la cantidad de veces que cada hormiga artificial va a recorrer el grafo. ● Se define la tasa de evaporación para la feromona ● Se define un aprendizaje (***Q***), como estrategia para compensar los parámetros alfa y beta. ● Se inicializa y deposita cierta cantidad igual de feromona para cada arista o conexión esto con el fin de que todos los posibles caminos que pueda escoger una hormiga tengan la misma probabilidad de elección.
4	**Condición de parada**	Para el problema del emparejamiento esta condición deja de ser verdadera cuando todos los vuelos programados han sido cubiertos por algún emparejamiento.
5	**Número de iteraciones locales**	Se define como número de iteraciones locales al número de veces que cada hormiga va a recorrer el grafo en búsqueda de soluciones. Se recibe como parámetro de entrada.
6	**Creación colonia de hormigas exploradoras artificiales**	Creación de hormigas exploradoras artificiales. Las cuales tienen como objetivo recorrer un grafo y buscar soluciones óptimas al problema.
7	**Buscar soluciones (emparejamiento)**	Para nuestro problema la búsqueda de soluciones son los emparejamientos, el concepto de emparejamiento se define como la secuencia de vuelos que una hormiga recorre cumpliendo ciertas restricciones dadas por el problema del emparejamiento.

8	**Aplicar restricciones**	Para nuestro problema en el proceso de búsqueda de soluciones o emparejamientos la hormiga debe aplicar un conjunto de restricciones como por ejemplo el número máximo de vuelos en cada emparejamiento, el número máximo de vuelos en cada servicio, el número máximo de servicios, el tiempo máximo de vuelo en cada emparejamiento, el tiempo máximo de vuelo de cada servicio y la más importante llegar al aeropuerto base de donde salió. $$minimizar \sum_{p \epsilon P} C_p Y_p$$ Donde: C_p = Es el costo asociado a un emparejamiento. Sujeto a variables de restricciones del problema. $p \epsilon P$ = p es un emparejamiento del conjunto P
9	**Evaluar solución**	Se evalúa la calidad de la solución generada por cada hormiga y se le aplica una penalización a las transiciones que no cumplan las restricciones.
10	**Depositar feromona**	Cada hormiga deposita cierta cantidad de feromona. La cantidad de feromona a depositar dependerá de que tan buena sea la solución encontrada, si la hormiga ha encontrado un camino óptimo donde se cumplan las restricciones dejará un rastro de feromona mayor con el fin de que nuevas hormigas tengan esta información y así más probabilidad de ir por ese recorrido.
11	**Evaporar feromona**	Es el proceso por el cual se evitan los óptimos locales, se evapora cierta cantidad de feromona de cada

		vértice por cada iteración local, la tasa de evaporación se define como parámetro de entrada.
12	**Crear Hormiga Seleccionadora**	Se crea una hormiga seleccionadora artificial. La cual tiene como objetivo de elegir la o las mejores soluciones (emparejamientos) por iteración.
13	**Cultivar hongo**	Una vez que las hojas (secuencia de ciudades visitadas) pasan por la limpieza y cumplen con las restricciones, son llevadas al cultivo por lo tanto los emparejamientos son aptos para cultivar el hongo. Estos emparejamientos podrían modificarse en futuras iteraciones incluyendo algunos vuelos siempre y cuando las restricciones del emparejamiento se sigan cumpliendo.
14	**Inicializar rastro feromona**	Se realiza de nuevo el proceso de inicializar y depositar cierta cantidad igual de feromona para cada arista o conexión esto con el fin de que todos los posibles caminos que pueda escoger una hormiga tengan la misma probabilidad de elección. Esta inicialización se hace cada vez que no se cumpla la condición de parada se toma como una iteración global del algoritmo.
15	**Obtener hongo**	Terminada el número de iteraciones se procede a obtener todos los emparejamientos que se han cultivado en el hongo, esta lista de emparejamientos puede verse afectada por emparejamientos que por algún motivo no puede llevarse a cabo como la cancelación de un vuelo.
16	**Crear hormiga Cultivadora**	Se crea una hormiga cultivadora artificial, encargada de monitorear el cultivo y optimizar los emparejamientos.

17	**Optimizar Emparejamientos**	Proceso por el cual se busca optimizar las soluciones que se encuentran cultivadas. La optimización se lleva a cabo buscando que emparejamientos se pueden enlazar formando solo uno.
18	**Detectar hojas infectadas**	Una vez el hongo ha sido cultivado se debe de tener un monitoreo de este, con el objetivo de detectar hojas infectadas. Una hoja infectada para nuestro problema se define como un incidente operativo (cancelación de vuelo), lo que llevaría a afectar una o varias secuencias de vuelos (emparejamientos) ya programados.
19	**Tratar hoja infectada**	Para nuestro problema se aíslan las hojas(vuelos) afectados, esta afectación se da por un incidente operativo que retrasa la ejecución normal de los vuelos y el cronograma que se tiene para la tripulación.
20	**Reprogramar soluciones afectadas**	Si se han detectado hojas infectadas y no se logró un tratamiento efectivo en el proceso de tratar hoja infectada, todo el hongo se deja de cultivar y se envía a buscar nuevas soluciones y seguir todos los procesos anteriormente descritos.
21	**Obtener hongo final**	El hongo contendrá todos los emparejamientos una vez se hayan reprogramado los vuelos afectados si es que los hay. Este es el resultado final si en el cultivo no se encuentran hojas infectadas.

5.2 IMPLEMENTACIÓN DE LA SOLUCIÓN

A partir del proceso descrito anteriormente se realiza una estructuración del algoritmo mediante un diagrama de clases donde se identifican todo el conjunto de clases que representan el modelo para la solución al problema del emparejamiento de la tripulación en el cual se encapsulan las funcionalidades.

5.2.1 Diagrama de clases

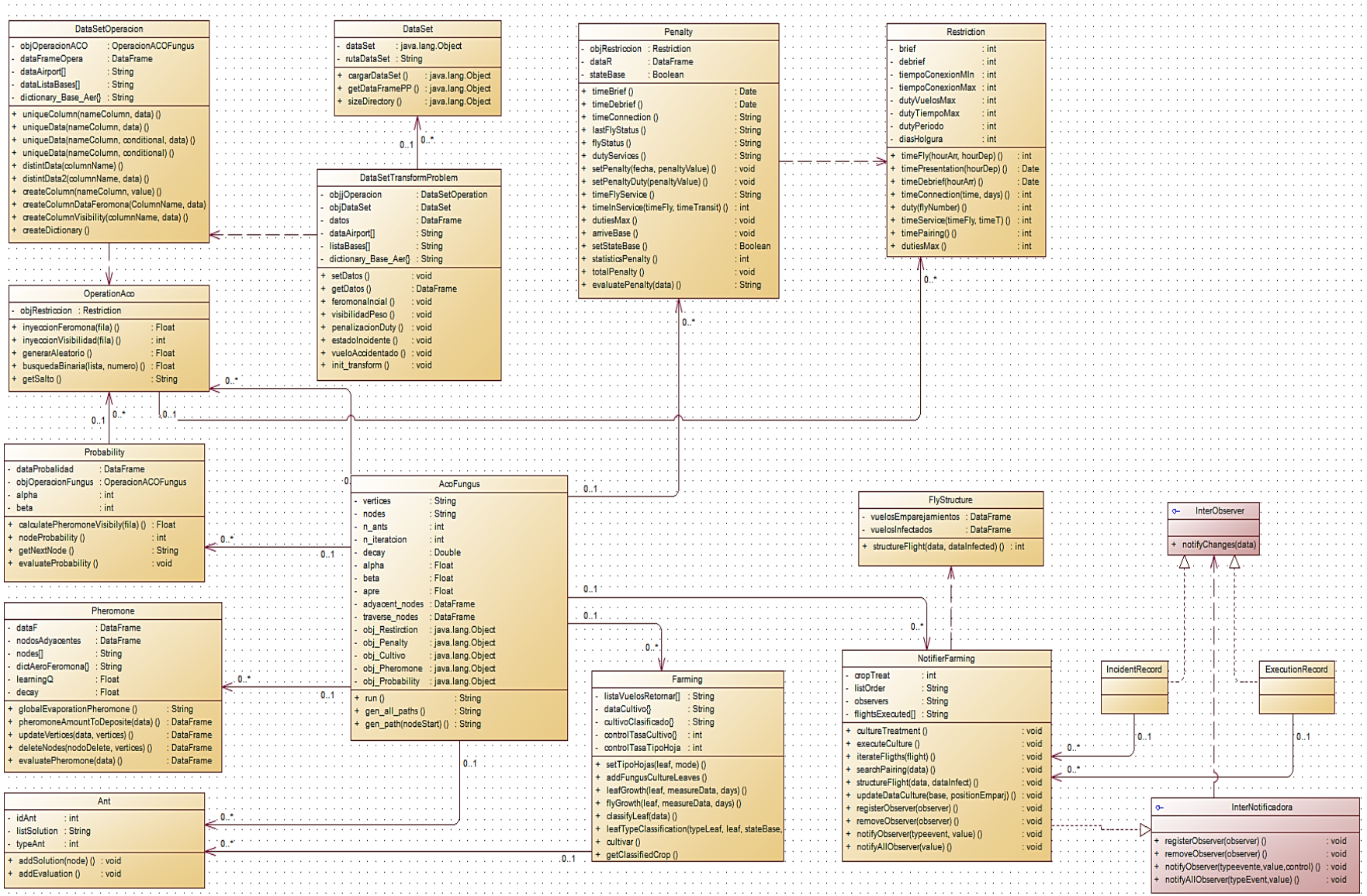

DataSetOperacion
- objOperacionACO : OperacionACOFungus
- dataFrameOpera : DataFrame
- dataAirport[] : String
- dataListaBases[] : String
- dictionary_Base_Aer{} : String
+ uniqueColumn(nameColumn, data) ()
+ uniqueData(nameColumn, data) ()
+ uniqueData(nameColumn, conditional, data) ()
+ uniqueData(nameColumn, conditional) ()
+ distintData(columnName) ()
+ distintData2(columnName, data) ()
+ createColumn(nameColumn, value) ()
+ createColumnDataFeromona(ColumnName, data)
+ createColumnVisibility(columnName, data) ()
+ createDictionary ()
DataSet
- dataSet : java.lang.Object
- rutaDataSet : String
+ cargarDataSet () : java.lang.Object
+ getDataFramePP () : java.lang.Object
+ sizeDirectory () : java.lang.Object
Penalty
- objRestriccion : Restriction
- dataR : DataFrame
- stateBase : Boolean
+ timeBrief () : Date
+ timeDebrief () : Date
+ timeConnection () : String
+ lastFlyStatus () : String
+ flyStatus () : String
+ dutyServices () : String
+ setPenalty(fecha, penaltyValue) () : void
+ setPenaltyDuty(penaltyValue) () : void
+ timeFlyService () : String
+ timeInService(timeFly, timeTransit) () : int
+ dutiesMax () : void
+ arriveBase () : void
+ setStateBase () : Boolean
+ statisticsPenalty () : int
+ totalPenalty () : void
+ evaluatePenalty(data) () : String
Restriction
- brief : int
- debrief : int
- tiempoConexionMin : int
- tiempoConexionMax : int
- dutyVuelosMax : int
- dutyTiempoMax : int
- dutyPeriodo : int
- diasHolgura : int
+ timeFly(hourArr, hourDep) () : int
+ timePresentation(hourDep) () : Date
+ timeDebrief(hourArr) () : Date
+ timeConnection(time, days) () : int
+ duty(flyNumber) () : int
+ timeService(timeFly, timeT) () : int
+ timePairing() () : int
+ dutiesMax () : int
DataSetTransformProblem
- objjOperacion : DataSetOperation
- objDataSet : DataSet
- datos : DataFrame
- dataAirport[] : String
- listaBases[] : String
- dictionary_Base_Aer{} : String
+ setDatos () : void
+ getDatos () : DataFrame
+ feromonaIncial () : void
+ visibilidadPeso () : void
+ penalizacionDuty () : void
+ estadoIncidente () : void
+ vueloAccidentado () : void
+ init_transform () : void
OperationAco
- objRestriccion : Restriction
+ inyeccionFeromona(fila) () : Float
+ inyeccionVisibilidad(fila) () : int
+ generarAleatorio () : Float
+ busquedaBinaria(lista, numero) () : Float
+ getSalto () : String
Probability
- dataProbalidad : DataFrame
- objOperacionFungus : OperacionACOFungus
- alpha : int
- beta : int
+ calculatePheromoneVisibily(fila) () : Float
+ nodeProbability () : int
+ getNextNode () : String
+ evaluateProbability () : void
AcoFungus
- vertices : String
- nodes : String
- n_ants : int
- n_iteratcion : int
- decay : Double
- alpha : Float
- beta : Float
- apre : Float
- adyacent_nodes : DataFrame
- traverse_nodes : DataFrame
- obj_Restirction : java.lang.Object
- obj_Penalty : java.lang.Object
- obj_Cultivo : java.lang.Object
- obj_Pheromone : java.lang.Object
- obj_Probability : java.lang.Object
+ run () : String
+ gen_all_paths () : String
+ gen_path(nodeStart) () : String
FlyStructure
- vuelosEmparejamientos : DataFrame
- vuelosInfectados : DataFrame
+ structureFlight(data, dataInfected) () : int
InterObserver
+ notifyChanges(data)
Pheromone
- dataF : DataFrame
- nodosAdyacentes : DataFrame
- nodes[] : String
- dictAeroFeromona{} : String
- learningQ : Float
- decay : Float
+ globalEvaporationPheromone () : String
+ pheromoneAmountToDeposite(data) () : DataFrame
+ updateVertices(data, vertices) () : DataFrame
+ deleteNodes(nodoDelete, vertices) () : DataFrame
+ evaluatePheromone(data) () : DataFrame
Farming
- listaVuelosRetornar[] : String
- dataCultivo{} : String
- cultivoClasificado{} : String
- controlTasaCultivo{} : int
- controlTasaTipoHoja : int
+ setTipoHojas(leaf, mode) ()
+ addFungusCultureLeaves ()
+ leafGrowth(leaf, measureData, days) ()
+ flyGrowth(leaf, measureData, days) ()
+ classifyLeaf(data) ()
+ leafTypeClassification(typeLeaf, leaf, stateBase,
+ cultivar ()
+ getClassifiedCrop ()
NotifierFarming
- cropTreat : int
- listOrder : String
- observers : String
- flightsExecuted[] : String
+ cultureTreatment () : void
+ executeCulture () : void
+ iterateFligths(flight) () : void
+ searchPairing(data) () : void
+ structureFlight(data, dataInfect) () : void
+ updateDataCulture(base, positionEmparj) () : void
+ registerObserver(observer) () : void
+ removeObserver(observer) () : void
+ notifyObserver(typeevent, value) () : void
+ notifyAllObserver(value) () : void
IncidentRecord
ExecutionRecord
Ant
- idAnt : int
- listSolution : String
- typeAnt : int
+ addSolution(node) () : void
+ addEvaluation () : void
InterNotificadora
+ registerObserver(observer) () : void
+ removeObserver(observer) () : void
+ notifyObserver(typeevente,value,control) () : void
+ notifyAllObserver(typeEvent,value) () : void

Figura 8: Representa el diagrama de clases propuesto para la implementación de la variante del algoritmo Aco – Fungus adaptado al problema del emparejamiento de la tripulación. Fuente: Autores

5.2.2 Descripción del diagrama de clases

Tabla 8 – Descripción diagrama de clases

DESCRIPCIÓN DEL DIAGRAMA DE CLASES	
Nombre Clase(s)	**Función**
DataSet, DataSetTransformProblem y DataSetOperation	Este conjunto de clases gestiona las acciones de los datos. En la clase *DataSet* se realiza el cargue de los datos para nuestro problema los datos vienen en formato .csv. Las clases *DataSetTransformProblem y DataSetOperation* se comunican para transformar la data, crea la lista de adyacencia para cada estación, crea la lista de nodos bases e inicializa rastro de feromona para los arcos de la lista de adyacencia.
Restriction y Penalty	Estas clases son las encargadas de validar y penalizar las soluciones generadas por las hormigas. En la clase *Restriction* se establecen todas las restricciones del problema que se entran a validar. La clase *Penalty* es la encargada de generar la penalización correspondiente si no se cumple alguna restricción.
Ant	Es la clase encargada de gestionar todo lo relacionado con las hormigas artificiales de la creación de cada hormiga según su tipo.
Pheromone	Es la clase encargada de gestionar todo lo relacionado con la feromona contiene métodos de evaporación de feromona , la cantidad de feromona a dejar , actualizar el rastro de feromona.

OperationAco, Probability	Estas clases son las encargadas de realizar cálculos para otras clases. *OperationAco* se implementa para gestionar acciones como inyección de la feromona, cálculo de la visibilidad y el que elige el próximo salto o nodo siguiente para la hormiga. La clase *Probability* es la encargada de gestionar todas las operaciones para que la hormiga pueda elegir un nodo.
Farming	Esta clase es la encargada del cultivo de las soluciones. En la clase *Farming* se inicializa una hormiga seleccionadora la encarga de seleccionar la mejor solución que generan las hormigas exploradoras y la que se agrega al cultivo. Dependiendo del problema clasifica las soluciones de acuerdo con su origen.
InterNotificadora, interObserver	Son interfaces que se generan para que otras clases implementen sus funciones.
IncidentRecord, ExecutionRecord	Son clases que se encargan de registrar la ejecución y los posibles incidentes del cultivo de soluciones. La clase *IncidentRecord* es la encargada de registrar una anomalía en el cultivo de soluciones. La clase *ExecutionRecord,* se encarga de registrar la ejecución de las soluciones que se encuentran en el cultivo.
NotifierFarming	Son las clases encargadas de gestionar todo lo relacionado con el cultivo de soluciones actúa como observador del cultivo junto con las clases *IncidentRecord y ExecutionRecord.*
FlyStructure	Esta clase es la encargada de gestionar todo lo relacionado con la optimización de las soluciones y cuando se presenta un incidente en el cultivo de soluciones. Para nuestra solución se encarga de verificar la estructura de los vuelos (secuencias) junto con su evaluación verificando si se pueden optimizar.

CAPÍTULO 6. EVALUACIÓN DEL ALGORITMO

En este capítulo se presenta el ajuste de los parámetros del algoritmo propuesto ACO – Fungus, para garantizar el mejor desempeño del algoritmo para solucionar el problema del emparejamiento de la tripulación. Además, se presentan las diferentes instancias experimentales que permiten analizar el comportamiento del algoritmo propuesto frente a otros enfoques que plantean el mismo problema del emparejamiento de la tripulación, se defienden métricas de evaluación que posteriormente se comparan y se concluye el comportamiento o la eficiencia del algoritmo propuesto.

6.1 AJUSTE DE PARÁMETROS

En la tabla 9 se presentan diferentes valores de parámetros de ACO propuestos en la literatura para solucionar el problema del emparejamiento de la tripulación, los cuales se utilizarán como base para el ajuste de parámetros del algoritmo propuesto. Por lo tanto, se obtiene un total de 288 escenarios o configuraciones que corresponden a: $k * (\alpha{:}\beta) * \rho * Q * Iteraciones$. Del número total de escenarios se eligió al alzar el 15% que equivalen 43 escenarios y por cada escenario se ejecutaron 3 réplicas, teniendo un total de 129 corridas experimentales. Ver Figura 9.

Tabla 9 – Pool de parámetros

POOL DE PRUEBA PARÁMETROS	
Ants (k)	[50, 100, 150, 200]
Alpha(α) : Beta(β)	[1:1, 1:2, 1:5, 0.7:1, 1:0.7, 2:1]
Tasa Evaporación (ρ)	[0.9 , 0.99]
Aprendizaje (Q)	[0.8, 1]
Iteraciones	[5, 10, 20]

	AJUSTE DE PARAMETROS																		
ID Prueba	Emparejamientos Generados	Ejecucion Total	Tiempo (M)	Tiempo (Horas)	Iteraciones Globales	Iteracions Totales	#hormigas	Iteraciones	α	β	ρ	Q	No Vuelos sin cubrir	%Sin Cobertura	No Vuelos Cubiertos	%Cobertura	Evaluacion Función	PromedioX Base	Promedio Vuelos	TiempoVal (S)
P9.0S3V2	218	8944	149,07	2,48	148	740	150	5	1	2	0.01	1	70	9,94	634	90,06	1349839	1,47	4,32	8400,55
P10.0S3V2	184	9648	160,80	2,68	209	1045	150	5	2	1	0.01	1	133	18,89	571	81,11	1417185	0,88	2,92	9029,45
P11.0S3V2	42	4352	72,53	1,21	34	170	150	5	1	2	0	1	265	37,64	439	62,36	1325894	1,24	13,91	3123,1
P12.0S3V2	47	3229	53,82	0,90	23	115	150	5	1	0.7	0.01	1	190	26,99	514	73,01	1585593	2,04	23,96	2223,41
P13.0S3V2	55	4360	72,67	1,21	39	195	150	5	1	0.7	0.01	1	187	26,56	517	73,44	1489485	1,41	14,33	3303,62
P14.0S3V2	50	4040	67,33	1,12	41	205	150	5	1	2	0	1	195	27,70	509	72,30	1457676	1,22	12,80	3178,82
P15.0S3V2	55	3373	56,22	0,94	52	520	50	10	1	1	0	1	209	29,69	495	70,31	1599744	1,06	9,92	2654,97
P16.0S3V2	50	8815	146,92	2,45	46	460	150	10	1	1	0	1	184	26,14	520	73,86	1439570	1,09	11,63	6768,58
P17.0S3V2	55	5087	84,78	1,41	48	240	150	5	1	1	0	1	210	29,83	494	70,17	1352798	1,15	10,94	3982,34
P18.0S3V2	57	5023	83,72	1,40	32	160	200	5	1	1	0	1	190	26,99	514	73,01	1712522	1,78	18,19	3789,29
P19.0S3V2	45	11984	199,73	3,33	41	410	200	10	1	1	0	1	262	37,22	442	62,78	1519443	1,10	11,73	8992,99
P20.0S3V2	49	9247	154,12	2,57	24	240	200	10	2	1	0	1	227	32,24	477	67,76	1451096	2,04	21,13	6373,61
P21.0S3V2	51	4568	76,13	1,27	68	680	50	10	1	1	0	1	191	27,13	513	72,87	1601961	0,75	8,01	3695,84
P22.0S3V2	49	6447	107,45	1,79	58	290	150	5	1	1	0	1	193	27,41	511	72,59	1636421	0,84	9,43	5361,71
P23.0S3V2	49	3228	53,80	0,90	19	95	150	5	0	0.7	0.01	1	189	26,85	515	73,15	1237951	2,58	28,84	2257,19
P24.0S3V2	48	3254	54,23	0,90	24	120	150	5	0.7	0	0.01	1	190	26,99	514	73,01	1509757	2,00	22,54	2325,83
P25.0S3V2	44	3309	55,15	0,92	21	105	150	5	1	1	0	1	239	33,95	465	66,05	1428443	2,10	23,00	2337,08
P26.0S3V2	42	6945	115,75	1,93	52	260	150	5	5	1	0	1	306	43,47	398	56,53	1621706	0,81	8,96	5602,59
P27.0S3V2	0	0	0,00	0,00	43	215	150	5	1	1	0,3	1	0	0,00	704	100,00	1537856	1,16	12,60	3593,3
P28.0S3V2	53	4142	69,03	1,15	36	180	150	5	1	1	0,3	1	216	30,68	488	69,32	1617583	1,47	15,19	3161,15
P29.0S3V2	38	7292	121,53	2,03	27	270	150	10	1	5	0,3	1	336	47,73	368	52,27	1010159	1,41	14,15	4913,68
P30.0S3V2	39	10739	178,98	2,98	54	540	150	10	1	5	0,1	1	331	47,02	373	52,98	1039461	0,72	7,48	8814,39
P31.0S3V2	53	22016	366,93	6,12	41	820	150	20	1	1	0,1	1	244	34,66	460	65,34	1520889	1,29	12,56	15967,25
P32.0S3V2	52	3343	55,72	0,93	28	140	150	5	1	1	0,1	1	198	28,13	506	71,88	1645148	1,86	20,21	2630,2
P33.0S3V2	55	3386	56,43	0,94	60	300	100	5	1	1	0,1	1	182	25,85	522	74,15	1686364	0,92	9,38	2790,74
P34.0S3V2	50	2376	39,60	0,66	27	135	100	5	1	1	0	1	218	30,97	486	69,03	1516725	1,85	19,15	1695,85
P35.0S3V2	46	3687	61,45	1,02	45	225	100	5	2	1	0,1	1	293	41,62	411	58,38	1555585	1,02	11,38	2874,71
P37.0S3V2	49	1911	31,85	0,53	59	295	50	5	1	1	0,1	1	209	29,69	495	70,31	1512649	0,83	8,58	1506,82
P38.0S3V2	48	1630	27,17	0,45	42	210	50	5	1	1	0	1	199	28,27	505	71,73	1619537	1,14	13,17	1235,61
P39.0S3V2	50	2343	39,05	0,65	20	100	100	5	1	1	0	1	239	33,95	465	66,05	1433771	2,50	24,85	1528,07
P40.0S3V2	49	2628	43,80	0,73	29	145	100	5	1	1	0,1	1	237	33,66	467	66,34	1437905	1,69	17,14	1855,8
P41.0S3V2	51	3251	54,18	0,90	45	225	100	5	1	0.7	0.1	1	170	24,15	534	75,85	1619014	1,16	12,93	2621,46
P42.0S3V2	64	4379	72,98	1,22	74	370	100	5	1	1	0,1	1	165	23,44	539	76,56	1653986	0,86	7,82	3636,34
P44.0S3V2	52	2425	40,42	0,67	28	140	100	5	1	1	0	1	187	26,56	517	73,44	1537608	1,86	19,68	1739,03
P45.0S3V2	50	8240	137,33	2,29	62	620	100	10	1	1	0,1	1	219	31,11	485	68,89	1577832	0,81	8,45	6617,77
P46.0S3V2	57	6346	105,77	1,76	44	220	200	5	1	1	0,1	1	189	26,85	515	73,15	1477994	1,30	13,07	5022,37

Figura 9. Resultados configuración de parámetros

La configuración del emparejamiento de la tripulación para el ajuste de parámetros consiste en construir secuencias de vuelos (emparejamiento) para 704 vuelos programados con un horizonte de tiempo de 3 semanas. Cada emparejamiento es asignado a una tripulación, se considera como mínimo 2 vuelos para que se considere un emparejamiento con un horizonte de tiempo de 1 semana.

De los experimentos de ajuste de parámetros, se obtiene que el mayor porcentaje de vuelos cubiertos con una cantidad menor de emparejamientos construidos se logra al variar los parámetros α y β. La configuración que obtuvo mejores resultados fue la combinación de $\alpha = 1$ y $\beta = 0.7$, generando un valor promedio de 75.56% de vuelos cubiertos con respecto a otras configuraciones y un promedio de generación del número de emparejamientos de 56.6, con respecto a otras configuraciones como por ejemplo $\alpha = 1$ y $\beta = 5$, obteniendo un 52.98 % de vuelos cubiertos y una generación de 38 emparejamientos, aunque el valor del número de emparejamientos es menor que la anterior configuración el valor de vuelos sin cubrir es mucho mayor con un promedio de 47.37% equivalentes a 333.5 vuelos.

Con respecto a los valores [0.9 , 0.99] correspondiente al parámetro de tasa de evaporación de la feromona (ρ) se observa en los experimentos que la incidencia en el cambio del valor con respecto a vuelos cubiertos y al número de emparejamientos generados se observa una ligera tendencia a mejorar con un valor de 0.9 de tasa de evaporación, obteniendo un 7.9 % más de cobertura de vuelos y un 10 % más de emparejamientos generados con respecto a una tasa de evaporación de [0.99].

Para el número de hormigas (K) y el número de iteraciones se observan ligeras variaciones, en la mejor réplica se obtuvo al considerar un total de 100 hormigas con 5 iteraciones esta combinación genera un mejor valor para el porcentaje de vuelos cubiertos con un 76.20 %, con respecto a una combinación de 50 hormigas y 5 iteraciones que tiene un porcentaje de cobertura del 70.31% en el peor de los casos. En cuanto a la función de evaluación la combinación de 100 hormigas y 5 iteraciones genera un 8.71% más en el valor de la función de evaluación con respecto al valor de la función de evaluación que genera una combinación de 200 hormigas y 5 iteraciones. El valor es comparado con la siguiente mejor cobertura.

En cuanto al tiempo de cómputo al variar los distintos valores de hormigas e iteraciones el comportamiento es exponencial. Teniendo en cuenta el valor de los resultados de porcentaje de

vuelos cubiertos, número de emparejamientos generados y el valor de la función de evaluación tomada como un conjunto (total de emparejamientos) para la mejor de las réplicas se obtiene un tiempo de cómputo de 54.18 minutos con respecto a otras configuraciones.

Aunque en muchas configuraciones se obtienen mejores valores para la función de evaluación y generan un número menor de emparejamientos el porcentaje de cobertura es muy bajo y esto es dado a que mayores números de vuelos cubiertos mayor va a hacer el valor de la función de evaluación. Por lo anterior se optó por una configuración en donde prevaleciera la mayor cobertura de vuelos posibles con un número de emparejamientos generados no tan bajos.

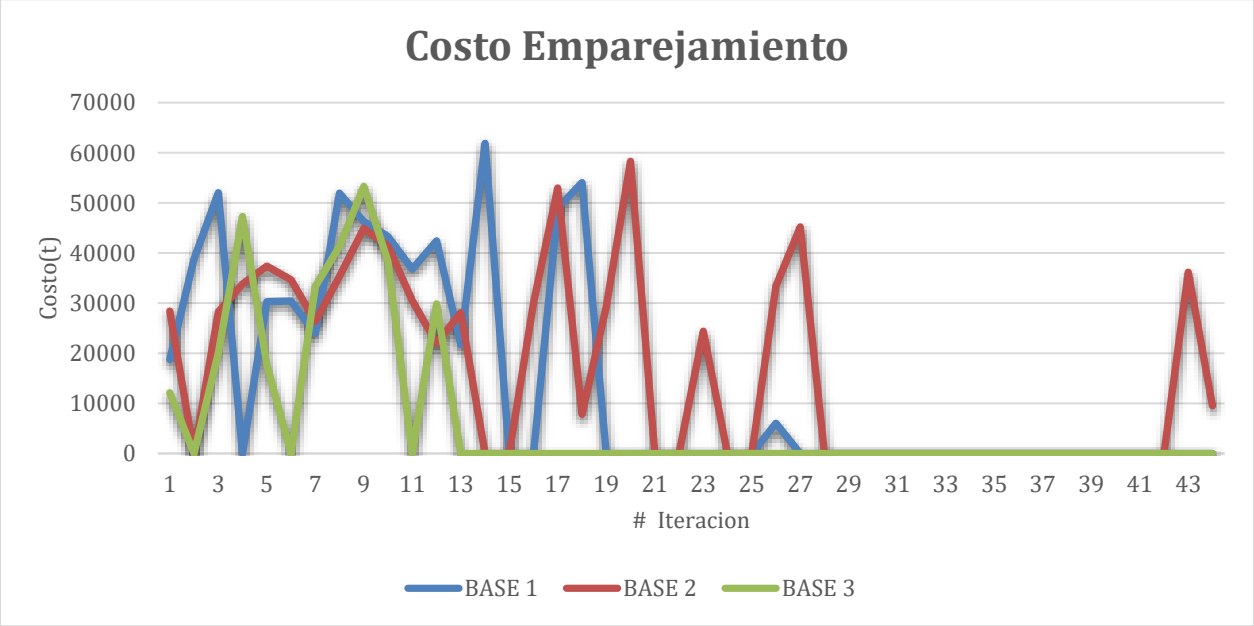
De acuerdo con lo anterior, los valores para los parámetros sugeridos para ACO- Fungus propuesto son:

Tabla 10 - Parámetros Instancias

PARÁMETROS ACO - FUNGUS	
Ants (k)	100
Alpha(α) : Beta(β)	[1:0.7]
Tasa Evaporación (ρ)	[0.9]
Aprendizaje (Q)	[1]
Iteraciones	[5]

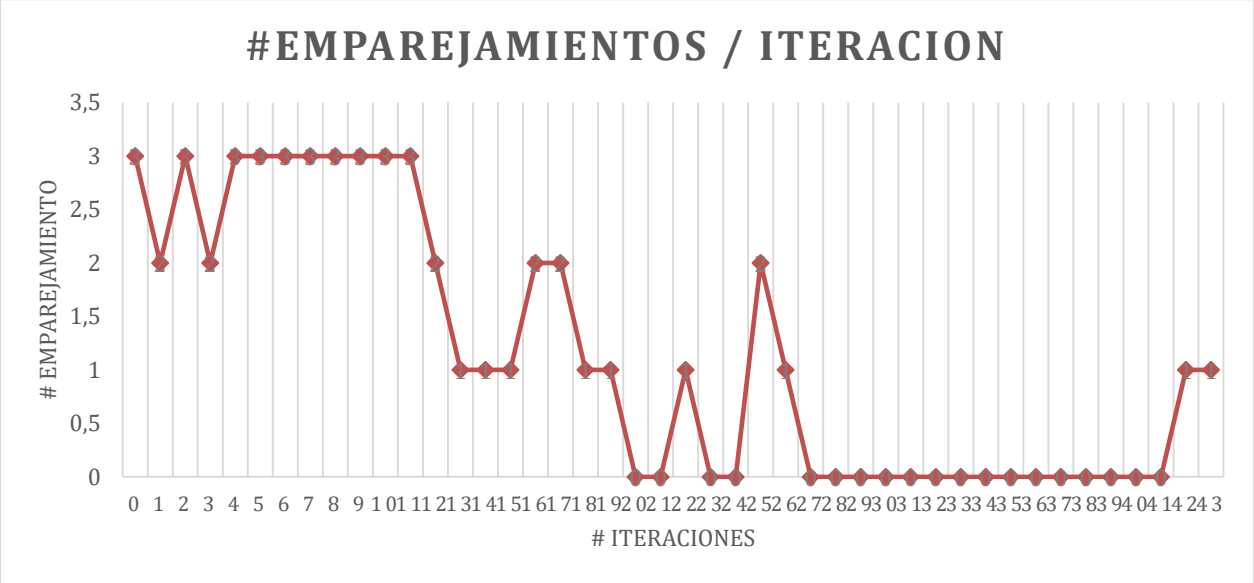
Con los parámetros de tabla 10 se obtuvieron los siguientes resultados para una de las mejores replicas.

Figura 10 - Emparejamientos Costo x Iteración



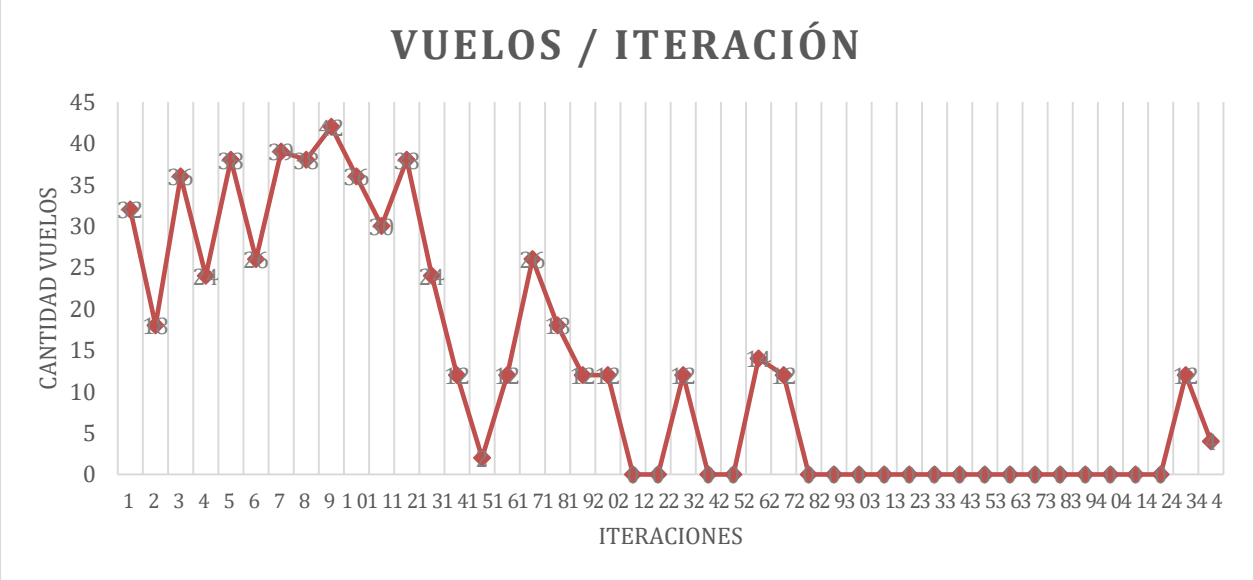
La figura 10 muestra el costo acumulado de la función de evaluación por iteración global. El costo está asociado a la suma de la función de evaluación de los emparejamientos generados, es decir si en una iteración se generaron 2 de los 3 posibles emparejamientos que se pueden generar por la existencia de 3 bases de donde pueden partir las hormigas y construir soluciones. Se suma el valor de la función para cada emparejamiento generado. El costo representa tiempo de servicio y penalizaciones.

Figura 11 - Emparejamientos x Iteración



La figura 11 muestra los mejores emparejamientos por iteración generados por el algoritmo, el número máximo de emparejamientos por iteración son 3 debido al problema las hormigas pueden salir de 3 bases distintas cada base genera un emparejamiento.

Figura 12 - Número Vuelos Cubiertos x Iteración



La figura 12 muestra la cantidad de vuelos cubiertos por iteración, con un total de 44 iteraciones globales que el algoritmo género y de un total de 704 vuelos programados se obtuvo un total de 534 vuelos que fueron cubiertos por algún emparejamiento lo que equivale al 75.85% de cobertura y un total de 51 emparejamientos con un promedio de vuelos por emparejamiento de 10.4 vuelos.

Figura 13 – Tiempo Ejecución x Iteración

La gráfica 13 muestra el tiempo de ejecución por iteración, es decir muestra cuanto tiempo de cómputo le tomó al algoritmo generar soluciones. Se puede observar que a mayor iteración el tiempo de cómputo disminuye y esto es debido a que en cada iteración se van eliminados nodos que hacen que las hormigas ya no puedan recorrer disminuyendo su espacio de búsqueda.

Igualmente se testearon valores iniciales apropiados de la feromona inicial, con el fin de obtener mejores resultados en el menor número de iteraciones posibles. Para esto se testearon valores iniciales de rastro de feromona $\tau_0 = 0.001$ y $\tau_0 = 0.01$ para todos los arcos. Ejecutando el algoritmo Aco – Fungus un determinado número de veces, se observó que para el rastro de feromona inicial a $\tau_0 = 0.001$ el algoritmo converge a un error ya que el valor de algunos arcos que son fuertemente penalizados el valor del rastro de feromona tendía a ser un valor demasiado pequeño que generaba error de cómputo.

Tabla 11 - Parámetros Instancias

Parámetros	Instancia Pequeña	Instancia Grande
Hormigas exploradoras	100	150
Iteraciones locales	5	10
Alpha	1	1
Betha	0.7	0.7
Tasa evaporación (1 -p)	0.9	0.9
Q aprendizaje	1	1
Hormiga seleccionadora	1	1
Hormiga cultivadora	1	1

6.2 DISEÑO DEL EXPERIMENTO

Una vez establecida la configuración de los parámetros del Aco – Fungus, se planifican los experimentos para establecer el desempeño del algoritmo bajo diferentes escenarios de ejecución. El dataset utilizado para probar nuestra propuesta de variante del algoritmo Aco Fungus fue tomado de [5]. El dataset contiene 7 instancias con un rango de vuelos que va de 1013 para la primera instancia hasta 7765 vuelos correspondiente a la última instancia. Las estaciones o aeropuertos varían entre 26 en la primera instancia y 54 en la última instancia. Se incluyen el número las bases de inicio para cada secuencia de vuelos, en nuestro problema cada estación representa un nodo o vértice. Cada instancia cuenta con 3 bases que significan el punto de partida de la secuencia de vuelos para una tripulación y a la cual debe regresar. El dataset mencionado anteriormente tiene un horizonte de planeación de vuelos mensual, cada vuelo describe la siguiente información:

- Número de vuelo
- Aeropuerto de salida
- Fecha de salida
- Hora de salida
- Aeropuerto de llegada
- Fecha de llegada
- Hora de llegada

Tabla 12 – Descripción DataSet

	Vuelos programados	**Promedio Vuelos diarios**	**Estaciones**	**Bases**
I1-727	1013	33	23	3
I2-DC9	1500	48	32	3
I3 -D94	1854	60	38	3
I4 -D95	5613	181	46	3
I5 - 757	5743	185	31	3
I6 - 319	5886	190	49	3
I7 - 320	7765	250	51	3

La Tabla 12 describe las 7 instancias con sus respectivas características, vuelos programados en un horizonte de tiempo de un mes, el promedio de vuelos diarios el número de estaciones y el número de bases.

Para la comparación de los resultados frente a otras propuestas se toma como instancia para la prueba la instancia I1 con 1013 vuelos y 26 estaciones con un programa de vuelos de 1 mes. La comparación del algoritmo propuesto se realizó con [35]. El algoritmo fue evaluado con distintas densidades de datos para la evaluación del comportamiento ver tabla 12. Dadas las mismas condiciones de [35] en cuanto a las restricciones del problema.

Tabla 13 – Descripción Instancias

Instancia	Vuelos Programados	Estaciones	Bases
I1 – 727	1013	23	3
I 1.1 - 727	235	16	3
I 1.2 - 727	477	17	3
I 1.3 - 727	704	17	3

Para [35] estudian la regularidad del horario de vuelo para todas las instancias de *Tabla 13*, en primer lugar, consideran solo los vuelos que operan durante la primera semana del mes (que es similar a las otras semanas).

Los parámetros se miden basados en dos enfoques. Se tiene la evaluación de las soluciones que genera el algoritmo, con este aspecto se mide cuantos emparejamientos realizó el algoritmo, costo del emparejamiento que para nuestra solución viene dada por el costo de tiempo, cuántos vuelos deja de cubrir, cuántas iteraciones se necesitaron para encontrar soluciones. En la parte computacional se establece la medición del tiempo promedio en que le toma a una hormiga encontrar una solución, recursos computacionales utilizados.

Se toman algunas consideraciones para las restricciones del problema:

- Tiempo mínimo de conexión y un tiempo máximo de 30 y 210 minutos respectivamente.
- Vuelos máximos por Duty o servicio 4,
- Tiempo máximo de un servicio de 480 minutos, en un periodo máximo de 1140 minutos.
- Máximo servicios o dutys por emparejamiento 4.

- Para nuestro problema no se toman en cuenta las soluciones de vuelos muertos o sin conexiones, solo los emparejamientos y soluciones que parten y regresan a la misma base.
- Mínimo 2 vuelos pueden conformar un emparejamiento.

Los experimentos se llevaron a cabo en un PC con sistema operativo Windows 10, con procesador Intel Core i5 2 núcleos a 1.70GHZ, memoria RAM de 12 GB.

La implementación del algoritmo se realizó en Python en su versión 3.8.5. Para la evaluación, la inicialización de algunos parámetros se basa en datos hallados en el estado del arte.

6.3 COMPARACIÓN DE RESULTADOS

Tabla 14 - Aco - Fungus VS Generación de columnas

	Generación de columnas			Aco - Fungus		
Instancias	I1.1	I 1.2	I 1.3	I1.1	I 1.2	I 1.3
No semanas	1	2	3	1	2	3
No Flights(vuelos)	231	459	645	235	477	704
No Vuelos sin cubrir	0	0	0	60	-	170
No Iteraciones	194	314	456	12500	-	22000
CPU tiempo (min)	0.9	32	193	22.4	-	54.18

La tabla 14 describe el comportamiento generado por los algoritmos en distinta densidad de datos ver Tabla 11, las instancias están dadas por el número de vuelos de acuerdo con el número de semanas establecidas. Se puede observar que para las tres instancias (I 1.1, I 1.2, I1.3), el algoritmo de generación de columnas es mucho mejor que el algoritmo propuesto, esto se evidencia en el número de vuelos sin cubrir el algoritmo generación de columnas cubre todos los vuelos de la instancia. Aco – Fungus deja de cubrir un 26% de los vuelos en cualquiera de las

instancias. El número de iteraciones generadas para dar solución al problema con el método de generación de columnas se necesitan mucho menos iteraciones en contraparte Aco – Fungus necesita un % porcentaje muy alto de número de iteraciones para generar las soluciones, lo que significa que se genere un tiempo usó mucho mayor de CPU, en la última instancia Aco -Fungus es un 56% más eficiente en el uso de CPU que en enfoque de generación de columnas. En conclusión, los resultados generados por Aco - Fungus el porcentaje promedio de vuelos sin cubrir es del 26% de los vuelos en las tres instancias, este porcentaje se mantiene en todas las instancias siempre se dejan de cubrir un número significativo de vuelos, generando tiempos de ejecución mucho mayores y número de iteraciones mayores.

Tabla 15 - Aco - Fungus vs Column Generation and DCA

		Column generation and DCA	
	Aco - Fungus	w/o HLA	With HLA
No Iteraciones	60000	1295	1622
No Column generated	-	18667	24169
CPU tiempo(min)	126.36	2	5
SubProblem CPU tiempo (min)		31	20
Total, tiempo CPU (min)	126.36	33	25

En la tabla 15 se muestran los resultados de la ejecución de la instancia I1 -727 ver tabla 11. Se miden el número de iteraciones y el tiempo de ejecución. En [35] se establecen dos variantes del algoritmo de generación de columna y una de ellas es el incluir un DCA o (Dynamic Constraint Agregation). En los resultados se observa que esta variante es mejor que el algoritmo propuesto Aco Fungus. El número de iteraciones que necesita el algoritmo Aco - Fungus es mucho mayor que las iteraciones que necesita el algoritmo Column generation and DCA, para generar soluciones, se necesita un promedio de 31.1% más de iteraciones. Y un 26.1% más de tiempo en CPU.

Tabla 16 - Aco - Fungus vs Column Generation

		Column Generation	
	Aco - Fungus	w/o HLA	With HLA
No Iteraciones	60000	5240	5640
No Column Generated	-	111269	101254
CPU tiempo(min)	126.36	487	551
SubProblem CPU tiempo (min)	-	2380	511
Total, tiempo CPU (min)	126.36	2867	1062

En la tabla 16 se muestra el resultado de la ejecución de la instancia I1 – 727. Los resultados muestran que el algoritmo Aco -Fungus mejora en cuanto a su tiempo de utilización de la CPU de 126.36 minutos significativamente más pequeño que los algoritmos HLA. El uso de la CPU es un 80% más eficiente en Aco – Fungus. El número de iteraciones que se necesitan para establecer los emparejamientos. Aco - Fungus aún sigue siendo mayor en comparación con los algoritmos HLA.

Tabla 17 – Estadísticas de Emparejamientos

	Aco - Fungus	Column Generation Sequential
No Vuelos	1013	1013
No vuelos sin cubrir	230	0
No Emparejamiento	85	162
Promedio Vuelos x Emparejamiento	12.9	6
Conc	0	22
Broken	0	46
Cost	-	460,632

En la tabla 17 se muestran las estadísticas para los emparejamientos generados por los dos algoritmos. Se obtiene que el algoritmo de generación de columnas en algunas variables es mucho mejor que el Aco-Fungus. El enfoque de generación de columnas cubre todos los vuelos programados, Aco-Fungus deja un porcentaje de 22% de vuelos sin cubrir. El número de emparejamientos que genera el enfoque de generación de columnas es mucho mayor un 47% más a comparación de Aco – Fungus. Aunque la calidad de los emparejamientos en Aco – Fungus es mejor en cuanto al promedio de vuelos por emparejamiento con un 40 % más de vuelos por emparejamiento que el enfoque de generación de columnas. En cuanto los costos de emparejamiento el enfoque de los dos algoritmos es distinto y esto es debido a que Aco – Fungus se enfoca en el costo del tiempo se intenta es maximizar el costo de tiempo en que la tripulación mantiene en vuelo, el algoritmo de generación de columnas se enfoca en los dos al tiempo le da un porcentaje que significa dinero y a un costo monetario asociado a cada emparejamiento.

CAPÍTULO 7. CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se dan las conclusiones del trabajo desarrollado y los trabajos futuros donde se puede seguir investigando tanto el comportamiento de las hormigas para dar solución al problema del emparejamiento como a otros problemas.

7.1 CONCLUSIONES

- Se propone un algoritmo basado en ACO, ACO – FUNGUS, donde se incluye las características naturales de las hormigas Atta: cultivo del hongo, sistema de recolección y selección de material para el cultivo, estrategias de saneamiento del cultivo y los roles de las hormigas: exploradora, seleccionadora y cultivadora denominada castas cada una con funciones distintas dentro de la colonia de hormigas. Este conjunto de actividades y roles permite tener un algoritmo de dos fases: construcción de soluciones parciales y la administración de un cultivo que es la solución del problema.

- El algoritmo propuesto incorpora restricciones de: tiempo mínimo y máximo de conexión, vuelos máximos por servicio (dutys), tiempo máximo de un servicio, número máximo de servicios(dutys) por emparejamiento y la de mayor importancia la restricción de retorno a base. Estas restricciones se consideran tanto en la formación de los emparejamientos como en la solución del incidente operacional.

- El algoritmo propuesto permite la instanciación o solución de otros problemas de planeación donde se necesite una rápida solución si se presenta una novedad en tiempo de operación. Problemas de optimización de tiempo en el ruteo de mensajería partiendo de distintas bases. Problemas de optimización de otro tipo de tripulaciones, variaciones del problema del viajero partiendo de distinta ciudad.

- El algoritmo propuesto obtuvo mejores resultados en la calidad de los emparejamientos debido a que se obtiene un promedio mayor de vuelos por emparejamiento comparado con el algoritmo de generación de columnas. Sin embargo, no logra el 100% de

cubrimiento de los vuelos programados, esto si los logra el algoritmo de Generación de columnas (Column Generation) la calidad de sus emparejamientos es menor además no incorpora el mismo número de restricciones.

- Se propone un mecanismo de monitorización que aborda los incidentes operacionales y genera una solución viable al incidente, esto permite tener una planeación real y coherente durante la ejecución de la operación aérea.

7.2 TRABAJOS FUTUROS

Durante el estudio para el desarrollo del presente trabajo de grado surgieron nuevos cuestionamientos que pueden ser tomados como posibles futuras líneas de trabajo de investigación.

- Paralelización del algoritmo.
- Mejorar el método de monitorización y notificación de incidentes.
- Aplicar el algoritmo a otro tipo de problema de optimización donde la representación de la solución no sean grafos.
- Integración con las otras fases del proceso de planeación.
- Prueba de otras estrategias de búsqueda local.
- Prueba de estrategias para el manejo de un conjunto de restricciones.

REFERENCIAS BIBLIOGRÁFICAS

[1] M. Deveci, «Evolutionary algorithms for solving the airline crew pairing problem,» *Computers & Industrial Engineering,* pp. 389-406, 2018.

[2] T. Grosche, computation intelligence in integrated airline scheduling, 2009.

[3] K. Nabil, «The integrated aircraft routing problem with optional flights and delay considerations,» *Transportation Research Part E,* pp. 355-375, 2018.

[4] K. Nabil, «An integrated flight scheduling and fleet assignment problem under uncertainty,» *Computers and Operations Research,* 2017.

[5] S. Mohammed, «Airline crew scheduling: models, algorithms, and data sets,» *Association of European Operational Research Societies,* 2015.

[6] C. Valentina, «Optimal Solutions to a Real-World Integrated Airline Scheduling Problem,» *Transportation Science,* pp. 1-19, 2016.

[7] E. Omar, «Particle swarm optimization algorithm for solving airline crew scheduling problem,» de *Conferencia Internacional 2014*, 2014.

[8] S. Nadia, «Genetic algorithm-based approach for the integrated airline crew-pairing and rostering problem,» *European Journal of Operational Research,* pp. 674-683, 2009.

[9] Y. Lijima, «column generation heuristics to airline crew scheduling problem for fair working time,» *IEEE International Conference on Systems, Man, and Cybernetics,* 2016.

[10] R. Camila, «Diseño de un algoritmo de generación de columnas para la programación de tripulación en logística aeroportuaria» *Universidad Industrial de Santander*, 2018.

[11] D. Muhammet, «A survey of the literature on airline crew scheduling,» *Engineering Applications of Artificial Intelligence,* pp. 54-69, 2018.

[12] A. Azadeha, «A hybrid meta-heuristic algorithm for optimization of crew scheduling,» *Applied Soft Computing,* pp. 158-164, 2013.

[13] D. Guang-Feng, «Ant colony optimization-based algorithm for airline crew scheduling problem,» *Expert Systems with Applications,* pp. 5787-5793, 2011.

[14] K. Arayikanon, «Solving cockpit crew scheduling problem of a low-cost airline using metaheuristics,» *AIP Conference Proceedings 2044,* 2018.

[15] A.-K. Ayyuce, «Crew pairing optimization based on hybrid approaches,» *Computers & Industrial Engineering,* pp. 87-96, 2013.

[16] H. Rieske, «Optimization Model for an Airline Crew Rostering Problem: Case of Garuda Indonesia,» pp. 218-234, 2013.

[17] F. Diana, «A Mathematical Programming Approach to Airline Crew Pairing Optimization, »

[18] V. Limlawan, «A hybrid particle swarm optimization and an improved heuristic algorithm for an airline crew rostering problem,» p. 456–462, 2014.

[19] Z. Mingyu, «A Heuristic Algorithm for Solving Crew Rostering Problem,» *Applied Mechanics and Materials,* pp. 2854-2858, 2012.

[20] Z. Tian, «Research on Optimization of Crew Scheduling of the Passenger Dedicated Line Based on a Genetic Ant Colony Algorithm,» *Critical Issues in Transportation Systems Planning, Development, and Management.,* 2009.

[21] D. Berna, «A Hybrid Approach of Heuristic and Exact Method for Crew Pairing Problem,» *International Conference on Computers & Industrial Engineering,* 2010.

[22] A. Alba, «A variable neighborhood search approach for the crew pairing problem,» *Electronic Notes in Discrete Mathematics,* pp. 87-94, 2017.

[23] D. Nihan, «Novel search space updating heuristics-based genetic algorithm for optimizing medium-scale,» *International Journal of Computational Intelligence Systems,* vol. 10, p. 1082–1101, 2017.

[24] A. Chakrabarti, «DRM, a Design Research Methodology», pp. 13-42, 2009.

[25] F. Quesnel, «A branch-and-price heuristic for the crew pairing problem with language constraints», 2019.

[26] A. Parmentier, «Aircraft routing and crew pairing: Updated algorithms at Air France»,2019.

[27] B. Mohamed, «Robust integrated maintenance aircraft routing and crew pairing», Journal of Air Transport Management, pp 15-31, 2018.

[28] K. Petersen, «Guidelines for conducting systematic mapping studies in *software* engineering: An update," *Information and Software Technology* », vol. 64, pp. 1-18, 2015.

[29] R. Monje, «Análisis y predicción de los retrasos de vuelo. Estudio del Aeropuerto de Seattle-Tacoma », 2015.

[30] M. Lugo, E. Crespo, «Hongos asociados con dos poblaciones de Acromyrnex Lobicornis(Formicidae) de San Luis Argentina », 2013.

[31] D. Villanueva , « Aislamiento y extracción de ADN del complejo de hongos simbiontes asociados a la hormiga arriera (*atta cephalotes)* », Universidad del Quindio.

[32] Y. Perez, « Comportamiento y selección de sitios de forrajeo de *atta cephalotes* (l.) en la estación primates- municipio de colosó – sucre (Colombia) », Universidad de Sucre, 2017.

[33] J. Lezaun (2020, 08). Hormiga arriera, una plaga evolucionada, eusocial y polimórfica. Agribusiness & Marketing Consultant South America Region [Online]. Available: https://www.croplifela.org/es/plagas/listado-de-plagas/.

[34] F. Quesnel, «A new heuristic branching scheme for the crew pairing problem with base constraints», *Computers and Operations Research, 2016*

[35] M. Saddoune, «Aircrew pairings with possible repetitions of the same flight number », *Computers & Operations Research, 2010.*

[36] J. Folgarait Patricia, «Un Mundo De Hormigas», Centro de Estudios e Investigaciones Universidad Nacional de Quilmes Buenos Aires.»

[37] F. Fernandez, « Hormigas cortadoras de hojasde Colombia: Acromyrmex & Atta(Hymenoptera: Formicidae)», FAUNA DE COLOMBIA - Monografía No. 5.

[38] 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.

[39] F.Obando, «Algoritmo Aco-Atta», Universidad del Cauca, 2020.