PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMO BIOINSPIRADO PARA EL PROBLEMA DE PROGRAMACIÓN DE DESPACHOS (TIMETABLING) DE UN CENTRO DE DISTRIBUCIÓN CON MÚLTIPLES ESTACIONES DE CARGA

Tesis para obtener el título profesional de Ingeniero Informático

AUTOR:

Jorge Miguel Baca Sáenz

ASESOR:

Rony Cueva Moscoso

Lima, 2026

Informe de Similitud

Yo, ...Rony Cueva Moscoso.....,

docente de la Facultad de .....Ciencias e Ingeniería..... de la Pontificia Universidad Católica del Perú, asesor(a) de la tesis/el trabajo de investigación titulado **Algoritmo bioinspirado para el problema de programación de despachos (timetabling) de un centro de distribución con múltiples estaciones de carga**, del/de la autor(a)/ de los(as) autores(as)

Jorge Miguel Baca Sáenz .....,

.....,

.....,

dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de 21%. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el 12/10/2025.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: ...Lima, 09 de febrero del 2026.....

Apellidos y nombres del asesor / de la asesora: Cueva Moscoso Rony	
DNI: 09942265	Firma
ORCID: 0000-0003-4861-571X	

Dedicatoria

A Dios, por acompañarme y cuidarme en este camino y a lo largo de mi vida.

A mi familia, por ser mi soporte y ánimos para continuar.

A mis amigos, quienes me ayudaron e hicieron este camino más divertido, y en especial a mi asesor y gran amigo, por su apoyo incondicional en este largo camino.

Resumen

Uno de los principales problemas dentro de la cadena de suministros empresarial es el manejo de varios factores como los horarios de despacho, la optimización de la atención de pedidos realizados para un horario específico, las rutas que van a tomar múltiples vehículos de despacho, entre otros; de manera que se puede satisfacer las necesidades de los clientes de las organizaciones. Dentro de las actividades que se pueden optimizar se encuentra la planificación de despachos (timetabling dispatch problem), realizada usualmente de forma manual con herramientas como hojas de cálculo, lo que invierte tiempo y no facilita el ajuste de un plan de despacho, ni garantiza un resultado óptimo o libre errores. En el presente estudio, se ha desarrollado un algoritmo de búsqueda de colonia de virus (VCS) para la asignación de horarios aplicado al despacho en una planta con múltiples bahías. El enfoque se orienta a la búsqueda de maximizar el despacho de productos en las diferentes bahías, para lo cual se tiene que priorizar las órdenes de despacho y minimizar el tiempo muerto, considerando las distintas restricciones como son las órdenes especiales que necesitan ser cargadas a una hora particular.

Tabla de Contenido

Índice de Tablas
Capítulo 1. Generalidades 1
1.1. Problemática 1
1.1.1. Árbol de problemas 1
1.1.2. Descripción 1
1.1.3. Problema seleccionado 4
1.2. Objetivos 4
1.2.1. Objetivo general 5
1.2.2. Objetivos específicos 5
1.2.3. Resultados esperados 5
1.2.4. Mapeo de objetivos, resultados y verificación 6
1.3. Métodos y procedimientos 8
1.3.1. Algoritmo de búsqueda de colonia de virus 9
1.3.2. C++ 9
1.3.3. Git 9
1.3.4. Método Taguchi 9
1.3.5. Qt Creator 10
1.3.6. Pseudocódigo 10
1.3.7. Pruebas de ANOVA 10
1.3.8. Pruebas estadísticas no paramétricas 10
1.3.9. Rstudio 10
1.3.10. Scrum 11
Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros 12
2.1. Objetivo 12
2.2. Desarrollo del marco 12
2.2.1. Bahía 12
2.2.2. Logística 13
2.2.3. Problema generalizado de asignación de horarios 13
2.2.4. Clase de problemas NP 13
2.2.5. Clase de problemas NP-difícil 13
2.2.6. Algoritmos metaheurísticos 14
2.2.7. Algoritmos bioinspirados 14
2.2.8. Algoritmo genético 14

Capítulo 3. Estado del arte 15
3.1. Introducción 15
3.2. Metodología de revisión 15
3.3. Objetivos de la revisión 15
3.4. Preguntas de revisión 16
3.5. Estrategias de búsqueda 17
3.5.1. Motores de búsqueda 17
3.5.2. Cadenas de búsqueda a utilizar 17
3.6. Documentos encontrados 19
3.7. Criterios de inclusión y exclusión 19
3.8. Formulario de extracción 22
3.9. Resultados de la revisión 24
3.9.1. Respuesta de la pregunta P1 24
3.9.2. Respuesta de la pregunta P2 24
3.9.3. Respuesta de la pregunta P3 30
3.9.4. Respuesta de la pregunta P4 32
3.10. Conclusiones 35
Capítulo 4. Parámetros, restricciones y función objetivo 36
4.1. Introducción 36
4.2. Resultados alcanzados 36
4.2.1. Lista de definición de parámetros y restricciones del problema de planificación. 36
4.2.1.1. Parámetros del problema 36
4.2.1.2. Restricciones del problema 37
4.2.2. Formulación de la función objetivo para el problema de planificación 38
4.2.2.1. Validaciones realizadas 38
4.3. Discusión 39
Capítulo 5. Estructuras de datos 40
5.1. Introducción 40
5.2. Resultados alcanzados 40
5.2.1. Diseño de estructuras de datos para el algoritmo genético adaptado al problema de planificación 40
5.2.2. Diseño de estructuras de datos que sirven de soporte para el algoritmo búsqueda de colonia de virus 40
5.2.3. Estructuras de datos auxiliares 41

5.3. Discusión 41
Capítulo 6. Adaptación del algoritmo genético para el problema de planificación 43
6.1. Introducción 43
6.2. Resultados alcanzados 43
6.2.1. Algoritmo genético aplicado al problema de planificación 43
6.2.1.1. Pseudocódigo del algoritmo genético 43
6.2.1.2. Construcción de la población inicial 44
6.2.1.3. Método de recombinación 45
6.2.1.4. Método de mutación 46
6.2.1.5. Adaptación de los genes 46
6.2.1.6. Validación de la solución 47
6.2.1.7. Cálculo del fitness 48
6.2.2. Validaciones realizadas 49
6.3. Discusión 49
Capítulo 7. Diseño del algoritmo de búsqueda de colonia de virus 50
7.1. Introducción 50
7.2. Resultados alcanzados 50
7.2.1. Algoritmo de búsqueda de colonia de virus aplicado al problema de planificación 50
7.2.1.1. Pseudocódigo del algoritmo Búsqueda de Colonia de Virus 50
7.2.1.2. Construcción de la población inicial 51
7.2.1.3. Difusión de virus 51
7.2.1.4. Infección de células huésped 52
7.2.1.5. Respuesta inmune 52
7.2.1.6. Selección de sobrevivientes 53
7.2.2. Validaciones realizadas 53
7.3. Discusión 53
Capítulo 8. Codificación del algoritmo genético 55
8.1. Introducción 55
8.2. Resultados alcanzados 55
8.2.1. Codificación del algoritmo genético adaptado al problema de planificación 55
8.2.1.1. Función principal del algoritmo genético 55
8.2.1.2. Construcción de la población inicial 56
8.2.1.3. Método de recombinación 57

8.2.1.4. Método de mutación 57
8.2.1.5. Adaptación de genes 58
8.2.1.6. Validación de la solución 58
8.2.1.7. Cálculo del fitness 59
8.2.2. Validaciones realizadas 59
8.3. Discusión 59
Capítulo 9. Codificación del algoritmo de búsqueda de colonia de virus 60
9.1. Introducción 60
9.2. Resultados alcanzados 60
9.2.1. Algoritmo de búsqueda de colonia de virus aplicado al problema de planificación 60
9.2.1.1. Función principal del algoritmo de búsqueda de colonia virus 61
9.2.1.2. Construcción de la población inicial 61
9.2.1.3. Difusión de virus 62
9.2.1.4. Infección de células huésped 62
9.2.1.5. Respuesta inmune 63
9.2.1.6. Selección de sobrevivientes 63
9.2.2. Validaciones realizadas 63
Capítulo 10. Calibración de variables 65
10.1. Introducción 65
10.2. Resultados alcanzados 65
10.2.1. Calibración de variables 65
10.2.2. Validaciones realizadas 66
10.3. Discusión 66
Capítulo 11. Experimentación numérica 67
11.1. Introducción 67
11.2. Resultados alcanzados 67
11.2.1. Informe de experimentación numérica 67
11.2.1.1. Recolección de datos 67
11.2.1.2. Prueba de normalidad 68
11.2.1.3. Prueba de igualdad de varianzas 69
11.2.1.4. Prueba T de Welch 69
11.2.1.5. Validaciones realizadas 70
11.2.2. Discusión 70
Capítulo 12. Referencias 71

Capítulo 13. Anexos 76
13.1. Anexo A – Formulario de extracción de datos 76
13.2. Anexo B – Plan de proyecto 76
13.2.1. Justificación 76
13.2.2. Viabilidad 76
13.2.2.2. Viabilidad temporal 77
13.2.2.3. Viabilidad económica 77
13.2.2.4. Viabilidad de datos 77
13.2.3. Alcance 77
13.2.4. Limitaciones 78
13.2.5. Riesgos 78
13.2.6. EDT 79
13.2.7. Lista de Tareas 80
13.2.8. Cronograma del proyecto 83
13.2.9. Recursos del proyecto 86
13.2.9.1. Capacitación 86
13.2.9.2. Materiales 86
13.2.9.3. Estándares 86
13.2.9.4. Equipamiento 87
13.2.9.5. Herramientas 87
13.2.9.6. Costeo del proyecto 87
13.3. Anexo C – Acta de conformidad de parámetros y restricciones 88
13.4. Anexo D – Pruebas de la función objetivo 88
13.5. Anexo E – Acta de conformidad de la función objetivo 89
13.6. Anexo F - Estructuras auxiliares 89
13.7. Anexo G - Acta de conformidad de las estructuras bioinspiradas 89
13.8. Anexo H - Acta de conformidad de la adaptación del algoritmo genético 90
13.9. Anexo I - Acta de conformidad del algoritmo búsqueda de colonia de virus 90
13.10. Anexo J – Código del algoritmo genético 90
13.11. Anexo K - Pruebas del algoritmo genético 90
13.12. Anexo L – Código del algoritmo búsqueda de colonia de virus 90
13.13. Anexo M - Pruebas del algoritmo búsqueda de colonia de virus 90
13.14. Anexo N – Acta de conformidad de la calibración de variables 91

13.15. Anexo O – Acta de conformidad de la experimentación numérica 91

Índice de Tablas

Tabla 1: Árbol de problemas. Fuente: Elaboración propia. 1

Tabla 2: Árbol de objetivos. Fuente: Elaboración propia. 4

Tabla 3: Medios de verificación e indicadores objetivamente verificables del primer objetivo específico. Fuente: Elaboración propia. 6

Tabla 4: Medios de verificación e indicadores objetivamente verificables del segundo objetivo específico. Fuente: Elaboración propia. 6

Tabla 5: Medios de verificación e indicadores objetivamente verificables del tercer objetivo específico. Fuente: Elaboración propia. 7

Tabla 6: Medios de verificación e indicadores objetivamente verificables del cuarto objetivo específico. Fuente: Elaboración propia. 8

Tabla 7: Herramientas, métodos y procedimientos. Fuente: Elaboración propia. 8

Tabla 8: Elementos PICOC. Fuente: Elaboración propia. 17

Tabla 9: Palabras clave PICOC. Fuente: Elaboración propia. 18

Tabla 10: Cantidad de resultados de búsqueda. Fuente: Elaboración propia. 19

Tabla 11: Artículos relevantes para la Revisión Sistemática. Fuente: Elaboración propia. 20

Tabla 12: Formulario de extracción. Fuente: Elaboración propia. 22

Tabla 13: Causas de los problemas considerados por cada estudio. Fuente: Elaboración propia. 24

Tabla 14: Soluciones, variables y restricciones consideradas por estudio. Fuente: Elaboración propia. 25

Tabla 15: Características y conceptos de cada estudio relacionados al problema de asignación de horarios. Fuente: Elaboración propia. 30

Tabla 16: Artículos que reportan el uso de algoritmos metaheurísticos y bioinspirados. Fuente: Elaboración propia. 32

Tabla 17: Parámetros identificados. Fuente: Elaboración propia. 36

Tabla 18: Resultados de la calibración de variables del algoritmo genético. Fuente: Elaboración propia 65

Tabla 19: Resultados de la calibración de variables del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia 66

Tabla 20: Riesgos del proyecto. Fuente: Elaboración propia. 78

Tabla 21: Lista de tareas del proyecto. Fuente: Elaboración propia. 80

Tabla 22: Cronograma del proyecto. Fuente: Elaboración propia. 83
Tabla 23: Requerimientos de capacitación. Fuente: Elaboración propia. 86
Tabla 24: Requerimientos de materiales. Fuente: Elaboración propia. 86
Tabla 25: Requerimientos de estándares. Fuente: Elaboración propia. 86
Tabla 26: Requerimientos de equipamiento. Fuente: Elaboración propia. 87
Tabla 27: Requerimiento de herramientas. Fuente: Elaboración propia. 87
Tabla 28: Costeo del proyecto. Fuente: Elaboración propia. 87
Tabla 29: Resultados de la prueba de la función objetivo. Fuente: Elaboración propia 88
Tabla 30: Representación de la estructura auxiliar "Horario". Fuente: Elaboración propia. 89

Índice de Ilustraciones

Ilustración 1:Representación de "Equipamiento de un muelle de carga". Adaptado de (Solumat, 2022). 12
Ilustración 2: Representación de un cromosoma. Fuente: Elaboración propia. 40
Ilustración 3: Representación de un virus. Fuente: Elaboración propia 41
Ilustración 4: Pseudocódigo del algoritmo genético. Fuente: Elaboración propia. 43
Ilustración 5: Construcción de la población inicial del algoritmo genético. Fuente: Elaboración propia. 44
Ilustración 6: Recombinación de cromosomas en el algoritmo genético. Fuente: Elaboración propia. 45
Ilustración 7:Proceso de recombinación del algoritmo genético. Fuente: Elaboración propia 45
Ilustración 8: Método de mutación del algoritmo genético. Fuente: Elaboración propia. 46
Ilustración 9: Proceso de mutación en el algoritmo genético. Fuente: Elaboración propia 46
Ilustración 10: Adaptación de cromosomas en el algoritmo genético. Fuente: Elaboración propia 46
Ilustración 11: Proceso de adaptación en el algoritmo genético. Fuente: Elaboración propia. 47
Ilustración 12: Validación del cromosoma en el algoritmo genético. Fuente: Elaboración propia. 47
Ilustración 13: Cálculo del fitness en el algoritmo genético. Fuente: Elaboración propia. 48
Ilustración 14: Pseudocódigo del algoritmo Búsqueda de Colonia de Virus. Fuente: Elaboración propia. 50
Ilustración 15: Inicialización de población de virus. Fuente: Elaboración propia. 51

Ilustración 16: Difusión de virus del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia. 51
Ilustración 17: Infección de células huésped en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia. 52
Ilustración 18: Respuesta inmune en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia. 52
Ilustración 19: Selección de virus sobrevivientes en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia. 53
Ilustración 20: Código de la función principal del algoritmo genético. Fuente: Elaboración propia. 55
Ilustración 21: Código de la construcción de la población inicial del algoritmo genético. Fuente: Elaboración propia. 56
Ilustración 22: Método de recombinación en el algoritmo genético. Fuente: Elaboración propia. 57
Ilustración 23: Método de mutación del algoritmo genético. Fuente: Elaboración propia 57
Ilustración 24: Método de adaptación de genes del algoritmo genético. Fuente: Elaboración propia. 58
Ilustración 25: Método de validación de la solución en el algoritmo genético. Fuente: Elaboración propia. 58
Ilustración 26: Método de cálculo del fitness en el algoritmo genético. Fuente: Elaboración propia. 59
Ilustración 27: Función principal del algoritmo VCS. Fuente: Elaboración propia. 61
Ilustración 28: Método de construcción de la población inicial en el algoritmo VCS. Fuente: Elaboración propia. 61
Ilustración 29: Método de difusión de virus del algoritmo VCS. Fuente: Elaboración propia. 62
Ilustración 30: Método de infección celular en el algoritmo VCS. Fuente: Elaboración propia. 62
Ilustración 31: Método de respuesta inmune en el algoritmo VCS. Fuente: Elaboración propia. 63
Ilustración 32: Método de selección de virus sobrevivientes en el algoritmo VCS. Fuente: Elaboración propia. 63
Ilustración 33: Promedio de fitness de los resultados obtenidos por cada algoritmo. Fuente: Elaboración propia 68
Ilustración 34: Diferencia entre las observaciones obtenidas de ambos algoritmos. Fuente: Elaboración propia 68
Ilustración 35: Prueba de normalidad de las observaciones del algoritmo genético. Fuente: Elaboración propia 68
Ilustración 36: Prueba de normalidad de las observaciones del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia 69
Ilustración 37: Prueba de homogeneidad de varianzas para las observaciones de ambos algoritmos. Fuente: Elaboración propia 69
Ilustración 38: Prueba de diferencia de medias para las observaciones de ambos algoritmos. Fuente: Elaboración propia 70
Ilustración 39: Estructura de Distribución del Trabajo. Fuente: Elaboración

propia. 80

Capítulo 1. Generalidades

1.1. Problemática

A continuación, se presenta el contexto de la problemática, sus causas y consecuencias; para poder así dilucidar la necesidad que se ha identificado y avalar la envergadura del presente proyecto.

1.1.1. Árbol de problemas

En esta sección se presenta la Tabla 1, donde se representa el árbol de problemas construido con el objetivo de identificar el problema de esta investigación y organizar la información obtenida.

Tabla 1: Árbol de problemas. Fuente: Elaboración propia.

<table>
<tr><td>Las soluciones no son utilizables en un entorno real</td><td>Insatisfacción por resultados que no cumplen las expectativas</td><td colspan="2">Las soluciones que pudiesen ser generadas pueden demorar varios minutos en ejecutarse</td></tr>
<tr><td colspan="4">Las pocas soluciones algorítmicas existentes para el problema de asignación de horarios de despacho en planta con múltiples bahías tienen resultados deficientes.</td></tr>
<tr><td>Las soluciones algorítmicas actuales no consideran diferentes variables y restricciones observadas en un entorno real.</td><td>Alto nivel de complejidad computacional necesario para resolver la asignación de despachos, calificándolos como NP-difíciles.</td><td>Los operadores de almacén realizan la asignación de horarios de despacho de manera manual.</td><td>Existen nuevos algoritmos con buenos resultados que todavía no han sido aplicados al problema.</td></tr>
</table>

1.1.2. Descripción

La cadena de suministros es el conjunto de actividades, instalaciones y medios de producción que permiten obtener materia prima, transformarla y fabricar el producto que se transporta y entrega al consumidor final (Roldan, s. f.). Con el paso del tiempo, las cadenas de suministro de las empresas se han hecho cada vez más complejas debido a

la integración de varios factores como son los horarios de despacho, pedidos realizados para un horario específico, cantidad de vehículos de despacho, entre otros; de manera que se puede satisfacer las necesidades del cliente de una mejor manera y con una menor pérdida de recursos.

Por otro lado, al agregar complejidad a la cadena de suministros, también se corre el riesgo de agregar vulnerabilidades que pueden ser difíciles de solucionar; como se ha podido observar en la reciente coyuntura producida por la pandemia, donde la distribución de suministros se ha visto muy afectada debido a las distintas restricciones aplicadas por los gobiernos, los cuales reducían o suspendían en su totalidad muchas actividades de dispensación de abastos (El Comercio, 2021).

Ante esta problemática, las empresas se enfrentan a la tarea de optimizar las actividades de distribución, de manera que se pueda realizar un abastecimiento que se adecue a las distintas restricciones existentes en el mercado y cumpla con las expectativas de los clientes. De esta manera, es esencial investigar si es posible optimizar estas actividades para alcanzar un nivel adecuado de despachos y evitar cuellos de botella.

Dentro de las actividades que se pueden optimizar se encuentra la planificación de despachos. Esta tarea se realiza usualmente de forma manual con herramientas como hojas de cálculo, invirtiendo tiempo y esfuerzo en la realización y ajuste de un plan de despacho, el cual no garantiza un resultado óptimo y puede presentar errores. Los errores que puede presentar la planificación de despachos es causa directa de un incremento de costos para la empresa y reflejo de una gestión logística ineficiente (de Man & Strandhagen, 2018).

La logística de despachos en varias bahías supone un esfuerzo por mantener un horario ordenado, sin solapamientos entre tareas, tareas específicas que deben ser realizadas en un tiempo específico o la carga excesiva de trabajo en una bahía de despacho, puesto que cualquiera de estos acontecimientos son un costo que la empresa debe solventar. El caso de solapamiento de tareas implica que se debe elegir entre una tarea u otra, mientras que, en el caso de la carga excesiva de tareas en una bahía, los retrasos generan un incumplimiento del horario establecido. Estos problemas no se pueden solucionar solamente con el incremento de la capacidad de despacho. De igual manera, el no realizar una tarea con restricciones de tiempo en un horario determinado puede generar una pérdida de clientes para la empresa (Sartori et al., 2021).

Algunas soluciones propuestas para la gestión de la planificación de despachos como son los ERP’s muestran variadas desventajas para alinear su uso con las necesidades

inherentes de cada empresa en particular. Los módulos relacionados a la logística proveen funcionalidades como la programación de despachos, no obstante, estas funcionalidades no son suficientes para cubrir los requerimientos de las empresas (de Man & Strandhagen, 2018), lo que lleva a estas a realizar ajustes posteriores. Estas falencias generan perjuicios en el proceso de despacho, puesto que una entrega fuera de hora, como es una entrega temprana o tardía, pueden generar multas hacia la empresa. En este tipo de problemas que tienen un alto nivel de complejidad computacional, donde no es asequible recorrer todas las combinaciones posibles, se utilizan algoritmos metaheurísticos, los cuales son eficientes, prácticos y permiten obtener soluciones de calidad en un tiempo razonable, mediante estrategias de alto nivel que sirven para explorar espacios de búsqueda de forma guiada (Blum & Roli, 2003).

Un algoritmo que ha sido ampliamente utilizado y estudiado como una solución para este problema es el algoritmo genético (GA), el cual ha sido aplicado en su forma base o modificado para poder obtener un mejor rendimiento. Este algoritmo ha demostrado un gran nivel de ajuste a distintos objetivos, brindando resultados satisfactorios (Weise, 2009).

Por otro lado, el algoritmo de búsqueda de colonia de virus (VCS), presentado en 2015, tiene resultados prometedores en comparación con otros algoritmos de optimización bastante utilizados como son la colonia artificial de abejas (ABC), la búsqueda Cuckoo (CS), la optimización de migración de animales (AMO), la optimización de lobo gris (GWO), entre otros (Li et al., 2016).

Debido a que el problema presentado es clasificado como NP-hard, gran cantidad de soluciones utilizan el algoritmo genético (GA), a pesar de que este algoritmo fue presentado hace bastante tiempo y actualmente existen soluciones que buscan mejorar los resultados obtenidos (Rose & Coenen, 2015). Es por eso que se propone utilizar el algoritmo bioinspirado búsqueda de colonia de virus en esta investigación, de forma que se pueda probar que el algoritmo antes mencionado obtiene mejores soluciones que el algoritmo genético.

A partir de la problemática mostrada, este proyecto de tesis propone el uso del algoritmo bioinspirado búsqueda de colonia de virus para dar solución al problema de asignación de horarios aplicado al despacho con múltiples bahías, mientras que se utilizará al algoritmo genético como elemento comparativo respecto a la solución propuesta.

1.1.3. Problema seleccionado

El problema seleccionado es la asignación de horarios aplicado al despacho en una planta con múltiples bahías. El enfoque se orienta a la búsqueda de maximizar el despacho de productos en las diferentes bahías de despacho, para lo cual se tiene que priorizar las órdenes de despacho y minimizar el tiempo sin carga de trabajo en las bahías de carga, tomando en cuenta distintas restricciones como son las órdenes especiales que necesitan ser cargadas a una hora particular. Se sugiere emplear el algoritmo de Búsqueda de Colonia de Virus para abordar esta problemática, adaptándolo al contexto específico, con el propósito de lograr una planificación efectiva para los usuarios y alineada con los objetivos de la empresa.

1.2. Objetivos

A continuación, se presentan el objetivo general y los objetivos específicos de esta investigación.

Tabla 2: Árbol de objetivos. Fuente: Elaboración propia.

OBJETIVO GENERAL	Implementar el algoritmo de búsqueda de colonia de virus (VCS) para realizar la planificación de horarios de despachos en una planta con múltiples bahías.			
OBJETIVOS ESPECÍFICOS	Identificar y definir parámetros, restricciones y relaciones mediante la función objetivo, la cual se utilizará para evaluar las posibles soluciones.	Adaptar un algoritmo de menor complejidad (búsqueda de colonia de virus) para resolver el problema de asignación de horarios.	Desarrollar un software para la ejecución de los algoritmos VCS y GA.	Desarrollar la comparación entre el algoritmo VCS y GA como solución del presente problema.

CAUSAS	Las soluciones actuales no consideran diferentes variables y restricciones observadas en un entorno real.	Alto nivel de complejidad computacional necesario para resolver la asignación de despachos, calificándolos como NP-difíciles.	Los operadores de almacén realizan la asignación de horarios de despacho de manera manual.	Existen nuevos algoritmos con buenos resultados que todavía no han sido aplicados al problema.

1.2.1. Objetivo general

El objetivo general de esta investigación es implementar el algoritmo de búsqueda de colonia de virus (VCS) para realizar la planificación de despachos en una planta con múltiples bahías.

1.2.2. Objetivos específicos

Los objetivos específicos, que se han definido en base al objetivo general, para la presente investigación son los siguientes:

O1. Identificar y definir parámetros y restricciones, además de su relación mediante la función objetivo, la cual se utilizará para evaluar las posibles soluciones.

O2. Adaptar el algoritmo búsqueda de colonia de virus para resolver el problema de asignación de horarios.

O3. Desarrollar un software para la ejecución de los algoritmos búsqueda de colonia de virus (VCS) y el algoritmo genético (GA).

O4. Desarrollar la comparación entre el algoritmo de búsqueda de colonia de virus (VCS) y el algoritmo genético (GA) como solución del presente problema, de forma que se pueda probar el algoritmo de búsqueda de colonia de virus y verificar sus resultados contra uno de los algoritmos más utilizados en la resolución del problema investigado.

1.2.3. Resultados esperados

Los resultados esperados, los cuales son detallados a continuación, se especifican como resultados a conseguir para lograr obtener el objetivo general.

- RE.1.1: Enumeración de los parámetros y restricciones que definen el problema a resolver.

RE.1.2: Elaboración de la función objetivo para el problema de planificación.
RE.2.1: Creación de estructuras de datos destinadas a respaldar el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.

- RE.2.2: Algoritmo de búsqueda de colonia de virus adaptado al problema en cuestión.
- RE.3.1: Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.
- RE.3.2: Adaptación del algoritmo genético al problema de planificación.
- RE.3.3: Codificación del algoritmo genético adaptado al problema de planificación.
- RE.4.1: Calibración de variables.
- RE.4.2: Informe de experimentación numérica.

1.2.4. Mapeo de objetivos, resultados y verificación

En esta sección se van a dar a conocer los medios de verificación de completitud de los resultados esperados:

Tabla 3: Medios de verificación e indicadores objetivamente verificables del primer objetivo específico. Fuente: Elaboración propia.

Objetivo 1: Identificar y definir parámetros, restricciones y su relación utilizando la función objetivo, la cual servirá para evaluar las posibles soluciones.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R1.1 Lista de definición de parámetros y restricciones del problema en cuestión.	Documento que contiene la definición, descripción y formulación de los parámetros y restricciones.	Aprobación del documento por parte de un especialista en logística.
R1.2 Formulación de la función objetivo para el problema de planificación.	Documento que contiene la definición y formulación de la función objetivo.	Aprobación del documento por parte de un especialista en algoritmia.

Tabla 4: Medios de verificación e indicadores objetivamente verificables del segundo objetivo específico. Fuente: Elaboración propia.

Objetivo 2: Adaptar el algoritmo de búsqueda de colonia de virus para resolver el problema de asignación de horarios.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R2.1 Diseño de estructuras de datos que tienen como fin el soporte para el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.	Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo de búsqueda de colonia de virus.	Aprobación del documento por parte de un especialista en algoritmia.

R2.2 Algoritmo de búsqueda de colonia de virus adaptado al problema en cuestión.	Documento que contiene el pseudocódigo del algoritmo de búsqueda de colonia de virus diseñado para la resolución del problema de planificación.	Aprobación del diseño por parte de un especialista en algoritmia.

Tabla 5: Medios de verificación e indicadores objetivamente verificables del tercer objetivo específico. Fuente: Elaboración propia.

Objetivo 3: Desarrollar un software para la ejecución de los algoritmos VCS y GA.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R3.1 Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.	Código fuente del algoritmo de búsqueda de colonia de virus.	Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo ejecutadas con éxito.
R3.2 Adaptación del algoritmo genético al problema de planificación.	- Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo genético. - Documento que incluye el pseudocódigo del algoritmo genético esquematizado para la resolución del problema planteado.	Aprobación de la adaptación por parte de un especialista en algoritmia.
R3.3 Codificación del algoritmo genético adaptado al problema de planificación.	Código fuente del algoritmo genético.	Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo ejecutadas con éxito.

Tabla 6: Medios de verificación e indicadores objetivamente verificables del cuarto objetivo específico. Fuente: Elaboración propia.

Objetivo 4: Desarrollar la comparación entre el algoritmo VCS y GA como solución del presente problema.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R4.1 Calibración de variables	Documento que contiene la calibración de variables para la ejecución de los algoritmos.	Aprobación del documento por parte de un especialista en algoritmia.
R4.2 Informe de experimentación numérica.	Informe que contiene el análisis de los resultados de la experimentación numérica realizada.	Aprobación del documento por parte de un especialista en algoritmia.

1.3. Métodos y procedimientos

En esta sección se describirán las herramientas, métodos y procedimientos que se emplearán durante el presente proyecto.

En el siguiente acápite se presentarán las herramientas que se utilizarán en el presente proyecto, así como los distintos métodos y procedimientos a emplear.

Tabla 7: Herramientas, métodos y procedimientos. Fuente: Elaboración propia.

RESULTADOS ESPERADOS	**HERRAMIENTAS Y METODOS**
R1.1 Lista de definición de parámetros, variables y restricciones del problema en cuestión.	No aplica
R1.2 Formulación de la función objetivo para el problema de planificación.	No aplica
R2.1 Diseño de estructuras de datos que tienen como fin el soporte para el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.	No aplica
R2.2 Algoritmo de búsqueda de colonia de virus adaptado al problema en cuestión.	Pseudocódigo
R2.3 Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.	NetBeans C++ Git Buenas prácticas de Scrum
R3.1 Adaptación del algoritmo genético al problema de planificación.	Pseudocódigo
R3.2 Codificación del algoritmo genético adaptado al problema de planificación.	NetBeans C++ Git Buenas prácticas de Scrum

R4.1 Calibración de variables	NetBeans C++ RStudio Método Taguchi
R4.2 Informe de experimentación numérica.	RStudio Pruebas de ANOVA Pruebas estadísticas no paramétricas

1.3.1. Algoritmo de búsqueda de colonia de virus

Algoritmo desarrollado en 2015 por Mu Dong Li, Hui Zhao, Xing Wei Wen y Tong Han, que simula el comportamiento de infección y difusión entre virus y células anfitrionas. Mientras tanto, el comportamiento de evolución siempre ocurre durante el proceso de adaptación del ambiente de cambio celular (Li et al., 2016).

Este algoritmo será utilizado en el presente proyecto para intentar solucionar el problema propuesto.

1.3.2. C++

Lenguaje de programación multiparadigma derivado del lenguaje C, como es descrito en el ISO/IEC9899:2011. Algunas de las facilidades que se proveen sobre el lenguaje C son tipos de datos adicionales, clases, plantillas, excepciones, espacios de nombre, sobrecarga de operadores, sobrecarga de funciones, referencias, operadores de manejo de espacio libre y librerías adicionales (ISO/IEC 14882, s. f.).

Se utilizará este lenguaje de programación para la codificación de los algoritmos y la interfaz del proyecto.

1.3.3. Git

Sistema de control de versionamiento gratis y de código abierto diseñado para manejar eficientemente proyectos sin importar su tamaño. La ramificación y la capacidad de trabajar con múltiples flujos paralelos son sus principales características (Git, s. f.).

En el presente proyecto se utilizará este sistema para el versionamiento del código fuente de los algoritmos y la interfaz de usuario.

1.3.4. Método Taguchi

Herramienta eficiente para el diseño y optimización de procesos y productos, debido a que se centra en la identificación y evaluación de las variables con mayor influencia en salida del proceso, en la reducción de los efectos de los factores no controlables y en la reducción de la variación del desempeño del proceso (Zapata Gómez & Sarache Castro, 2014).

Se utilizará este método para la elección y calibración de variables para el proyecto de investigación.

1.3.5. Qt Creator

Entorno de desarrollo integrado multiplataforma de código abierto de Apache, el cual permite desarrollar distintas aplicaciones en diversos lenguajes de programación como son C++ y Python (The Qt Company, s. f.)

Se utilizará este programa en el proyecto de tesis para la codificación de los algoritmos Genético y Búsqueda de Colonia de Virus, así como la interfaz de usuario.

1.3.6. Pseudocódigo

Herramienta de programación en el cual se escriben las instrucciones en lenguaje natural. El pseudocódigo se construye con instrucciones no específicas de algún lenguaje en específico, pero que reflejan un comportamiento similar. Al tener un programa en pseudocódigo se puede convertir con mayor facilidad a un lenguaje de programación (Duque et al., 2017).

Se utilizará esta herramienta para el diseño del flujo de los algoritmos seleccionados.

1.3.7. Pruebas de ANOVA

El análisis de varianza (ANOVA) es uno de los métodos más eficientes para el análisis de data experimental. Este método de considerable complejidad y sutileza, con una amplia cantidad de variaciones, cada una de las cuales se van a emplear en un contexto experimental particular (Armstrong et al., 2002).

En el presente proyecto se van a utilizar las pruebas de ANOVA, condicionado a las características de los datos, para la realización de una experimentación numérica y la realización de la comparación de los resultados de los algoritmos.

1.3.8. Pruebas estadísticas no paramétricas

Las pruebas no paramétricas engloban una serie de pruebas no estadísticas que tienen como denominador común la ausencia de asunciones acerca de la ley de probabilidad que sigue la población de la que ha sido extraída. Por esta razón es común referirse a ellas como pruebas de distribución libre (Berlanga-Silvente & Rubio-Hurtado, 2012).

Este tipo de pruebas se van a utilizar en el proyecto, condicionado a las características de los datos, para la realización de la comparación de los resultados de algoritmos.

1.3.9. Rstudio

Entorno de desarrollo integrado para el lenguaje de programación estadístico R. Incluye una consola, editor de texto con resaltado sintáctico que soporta ejecución de código

directo, así como instrumentos de gráficos, historial, debugging y manejo del espacio de trabajo (RStudio, s. f.).

Se utilizará este entorno de desarrollo para realizar análisis estadísticos utilizando los resultados obtenidos a partir de los algoritmos codificados.

1.3.10. Scrum

Scrum es un marco de trabajo liviano que ayuda a personas, equipos y organizaciones a generar el mayor valor posible a través de soluciones adaptativas para problemas complejos (Scrum, s. f.).

Para el desarrollo de este proyecto de tesis se utilizarán buenas prácticas que se pueden obtener a partir de la metodología, tales como los eventos de scrum como son el sprint, el planeamiento del sprint, la revisión del sprint y la retrospectiva del sprint; con el objetivo de generar valor y así poder completar la investigación.

Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros

En el presente capítulo se presenta el marco legal, regulatorio y conceptual utilizado en la presente investigación.

2.1. Objetivo

El siguiente marco conceptual tiene por objetivo presentar conceptos importantes relacionados a la problemática de asignación de horarios aplicado al despacho. Las definiciones presentadas serán puestas en contexto a partir de ejemplos expuestos para facilitar la comprensión y vinculación con el problema.

2.2. Desarrollo del marco

A continuación, se presentarán distintos conceptos relacionados al problema tratado en esta investigación.

2.2.1. Bahía

También conocido como muelle de carga, las bahías son equipamientos industriales diseñados para facilitar el trasiego de materiales entre naves industriales y vehículos de transporte de mercancía (INSHT, 2016), el cual puede contar con elementos adicionales que facilitan el trabajo que se realiza en el lugar. Estos son lugares estratégicos en la logística de los almacenes, siendo que la agilidad en el envío mejora el servicio al cliente y la operación en el resto de las zonas (Garcia-Sabater, 2020).

Ilustración 1:Representación de "Equipamiento de un muelle de carga". Adaptado de (Solumat, 2022).

Dependiendo del almacén, se pueden tener múltiples muelles de carga, lo que origina un esfuerzo adicional en la planificación de los despachos. Teniendo en cuenta la importancia de las bahías en la cadena de suministro, es importante velar por su buena gestión.

2.2.2. Logística

La logística es el movimiento de los bienes correctos en la cantidad adecuada hacia el lugar preciso en el momento apropiado (Benjamín & Fincowsky, 2004), siendo así que requiere una gran cantidad de planificación y evaluación. Esta se da dentro y fuera de la organización, pues se encarga de la función operativa que comprende desde la obtención y administración de materias primas hasta el manejo de productos terminados, su empaque y distribución a los clientes (apud Ferrel et al, 2004, pag 282).

Un ejemplo de logística aplicado al tema de estudio es la planificación de los horarios de despacho en la planta, la cual permite la distribución de los productos terminados.

2.2.3. Problema generalizado de asignación de horarios

También conocido como timetabling problem, este problema es aquel donde se asigna un conjunto de recursos que son asignados dentro de un conjunto de periodos de tiempo sujetos a un conjunto de restricciones (Cooper et al., 1996). Este problema es un derivado de los problemas de cobertura de conjuntos (set covering problem), el cual busca encontrar el mínimo número de conjuntos que incorpore (cubra) todos los elementos dados (Gass & Fu, 2013). Este tipo de problemas es clasificado como NP-dificil (Ahmadizar & Farahani, 2012; Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Hsu et al., 2021; S. Nguyen, 2017) por su alto nivel de complejidad computacional.

2.2.4. Clase de problemas NP

Referido al nombre Nondeterministic Polynomial time, la clase de problemas NP es una clase de problemas de decisión que pueden ser resueltos por una máquina de Turing no determinista en tiempo polinómico (Cook, s. f.). Dentro de esta clase de problemas se encuentran distintos problemas de búsqueda y optimización para los que se desea saber si existe cierta solución o si existe una mejor solución que las conocidas.

2.2.5. Clase de problemas NP-difícil

Dentro de la clase de problemas NP se encuentra la clase NP-hard (NP-difícil). Los problemas NP-difíciles son problemas en los que no se conoce a la fecha un algoritmo determinístico que obtenga una solución en tiempo polinomial a partir del tamaño del problema (Guturu & Dantu, 2008). Esta complejidad hace que sea difícil encontrar una solución a partir de algoritmos tradicionales o técnicas matemáticas, puesto que, en el peor de los casos, el tiempo de procesamiento tendrá un incremento exponencial. Por esta razón se utilizan diversas técnicas, como son las heurísticas, metaheurísticas y estrategias de optimización global, las cuales permiten obtener una solución aceptable en un tiempo razonable (Guturu & Dantu, 2008).

2.2.6. Algoritmos metaheurísticos

Según Blum y Roli, los algoritmos metaheurísticos suelen ser estrategias de alto nivel que guían a una heurística subyacente más específica de un problema para incrementar su rendimiento, con el objetivo de evitar las desventajas de la mejora iterativa y la descendencia múltiple al permitir que la búsqueda local escape de los óptimos locales (apud Stützle, 1999).

En este tipo de algoritmos es importante mantener un balance entre la búsqueda de regiones con resultados de alta calidad y el desperdicio de tiempo en áreas ya exploradas o con soluciones que no representan una mejora en la solución (Blum & Roli, 2003).

Dentro de las estrategias de búsqueda utilizadas en este tipo de algoritmos se pueden encontrar distintas filosofías como son los algoritmos bioinspirados, los cuales nacen a partir de la imitación de comportamientos encontrados en la naturaleza (Fister Jr. et al., 2013).

2.2.7. Algoritmos bioinspirados

Son algoritmos que se encuentran basados en sistemas biológicos, como por ejemplo el algoritmo genético, que se basa en la evolución, o el algoritmo de polinización de flores, el cual se basa en el sistema de polinización que se encuentra en la naturaleza (Fister Jr. et al., 2013).

2.2.8. Algoritmo genético

El algoritmo genético es una subclase de los algoritmos evolutivos, donde los elementos del espacio de búsqueda (llamados cromosomas) son cadenas binarias o cualquier otro tipo de arreglos de tipos elementales, los cuales son usados en operaciones de reproducción u operadores genéticos como son la recombinación, la mutación y la selección (Weise, 2009) para buscar mejores soluciones.

Este algoritmo ha sido usado satisfactoriamente para resolver problemas como son los problemas en ingeniería química, medicina, minado de datos, análisis de datos, geometría, física, economía y finanzas, redes y comunicaciones, ingeniería eléctrica, diseño de circuitos, así como el tema del presente proyecto de investigación (asignación de horarios) (Weise, 2009).

Capítulo 3. Estado del arte

3.1. Introducción

La revisión de literatura es un paso que se da antes de comenzar a realizar una investigación, con el cual se aproxima al conocimiento de un tema y ayuda a identificar lo que se conoce y se desconoce actualmente de un tema en específico, dando como resultado un resumen conciso, objetivo y lógico que resume diferentes investigaciones y artículos (Goris & Adolf, 2015).

El propósito de realizar una revisión de literatura de tipo conceptual-empírica es hacer uso de la crítica y los estudios anteriores de manera ordenada, precisa y analítica (Goris & Adolf, 2015), los cuales serán la base de la investigación reflejada en este documento. El presente capítulo busca reflejar el proceso de la revisión de literatura realizada para el trabajo de tesis propuesto.

3.2. Metodología de revisión

La revisión que se va a utilizar en este trabajo es la revisión sistemática (systematic review), la cual es una revisión de una pregunta formulada claramente, donde se usan métodos sistemáticos y reproducibles para identificar, seleccionar y evaluar críticamente toda la información relevante, y recopilar y analizar datos provenientes de los estudios incluidos en la revisión (Librarian, s. f.).

3.3. Objetivos de la revisión

A continuación, se indican los objetivos por los cuales se realiza la presente revisión conceptual-empírica de literatura:

- ➢ Entender conceptos pertenecientes al ámbito de las soluciones del problema generalizado de asignación de horarios (O1).

- ➢ Conocer qué investigaciones se han realizado con respecto a la solución del problema de asignación de horarios en el despacho de una planta de producción (O2).

- ➢ Identificar los conocimientos obtenidos por otros investigadores en la búsqueda de soluciones del problema de asignación de horarios de despacho o similares; de manera específica, las variables y restricciones que se han identificado (O3).

- ➢ Conocer qué formas de comparación y verificación se realizaron a los resultados de otras investigaciones relacionadas (O4).

3.4. Preguntas de revisión

Una revisión sistemática comienza con el planteamiento de una pregunta clara, específica y estructurada que determinará los términos que se van a utilizar en la búsqueda en las bases de datos y artículos para responder a dicha pregunta (Moreno et al., 2018). Debido a esto, se han formulado las siguientes preguntas de investigación:

P1. ¿Cuáles son las causas de los problemas de asignación de horarios en el despacho de una planta de producción?
P2. ¿Qué soluciones se están dando a los problemas de asignación de horarios en el despacho de productos y cuáles fueron las variables y restricciones presentadas?
P3. ¿Qué características y conceptos se han investigado acerca de la forma de asignar horarios para el despacho de productos?
P4. ¿Qué algoritmos metaheurísticos o bioinspirados se están utilizando para la solución del problema generalizado de asignación de horarios aplicado al despacho y cómo se han evaluado, comparado y verificado?

Para poder definir y estructurar de una manera concreta los elementos de las preguntas a responder se ha utilizado el método PICOC (Petticrew & Roberts, s. f.), donde se busca diferentes evidencias y salidas de una búsqueda. Este método describe 5 elementos principales a partir de los cuales se estructuran los resultados de la investigación, los cuales son la población sobre la cual se va a investigar, la intervención realizada sobre la población, la comparación con otras intervenciones realizadas, la salida o resultado que se quiere lograr y el contexto en el cual se realiza la investigación. Los elementos antes mencionados son descritos en la Tabla 8:

Tabla 8: Elementos PICOC. Fuente: Elaboración propia.

Elemento	Responde a	Descripción
Population o población	¿Quién/es?	Problema generalizado de asignación de horarios aplicado al despacho.
Intervention o intervención	¿Qué o cómo?	Aplicación de algoritmos bioinspirados.
Comparison o comparación	¿Comparado con qué?	Aplicación de algoritmos metaheurísticos y bioinspirados más usados en problemas similares.
Outcome o salida	¿Qué se busca lograr o mejorar?	Investigaciones sobre soluciones al problema generalizado de asignación de horarios aplicado al despacho.
Context o contexto	¿En qué tipo de organización / circunstancias?	Esta investigación toma lugar en el ámbito académico y de planta industrial.

3.5. Estrategias de búsqueda

A continuación, se describen los factores más importantes de la estrategia de exploración a utilizar.

3.5.1. Motores de búsqueda

Los motores de búsqueda con accesos universitarios seleccionados en base al campo de investigación (Computer Science) son los siguientes:

- Scopus
- IEEE Xplore Digital Library

3.5.2. Cadenas de búsqueda a utilizar

A partir de los elementos PICOC, se definieron palabras clave y sinónimos auxiliares para la construcción de la cadena de búsqueda. De esta manera, se tiene la Tabla 9 con las

palabras clave para los siguientes elementos PICOC:

Tabla 9: Palabras clave PICOC. Fuente: Elaboración propia.

Valor	Palabras clave
Population o población	- timetabling - problem - assignment - schedule - dispatching
Intervention o intervención	- algorithm - method
Comparison o comparación	- metaheuristic - bioinspired
Outcome o salida	- solution
Context o contexto	- industry

Es así que se propone las siguientes cadenas de búsqueda para las preguntas planteadas, en donde se ha prescindido de las palabras clave que hacen referencia a los valores de salida y contexto, puesto que devuelven resultados muy generales; mientras que se filtraron documentos fuera del alcance de esta investigación:

- Scopus: TITLE-ABS-KEY ((“timetabling* problem” OR “schedule assignment” OR “scheduling*”) AND (“dispatch*” OR “ship*” OR “post goods”) AND (“algorithm*” OR “method*”) AND (“metaheuristic*” OR “bioinspired*") AND NOT ("economic" OR "energy" OR "power" OR "learning" OR "rout*" OR "dispatch* rule*" OR "spatial" OR "berth*"))
- IEEE: (((“timetabling problem" OR "schedule assignment" OR "scheduling”) AND (“dispatch*" OR "ship*" OR "post goods”) AND (“algorithm" OR "method”) AND (“metaheuristic" OR "bioinspired") AND NOT (“economic” OR “energy” OR “power” OR “learning” OR “rout*” OR “dispatch rule*” OR “spatial” OR “berth" OR "unit

commitment")))

3.6. Documentos encontrados

En la Tabla 10, se presenta la cantidad de documentos encontrados usando las cadenas de búsqueda descritas anteriormente.

Tabla 10: Cantidad de resultados de búsqueda. Fuente: Elaboración propia.

Motor	**Resultados**	**Artículos relevantes / primarios**
Scopus	37	12
IEEE Xplore Digital Library	14	2
Total	**51**	**14**

La búsqueda de artículos primarios terminó con 14 artículos relevantes, los cuales fueron filtrados por distintos criterios descritos a continuación.

3.7. Criterios de inclusión y exclusión

En la presente revisión se incluyen las investigaciones que satisfacen con las siguientes pautas de inclusión:

- El estudio aborda un tema relacionado al problema generalizado de asignación de horarios (CI1).

- El estudio aborda el tema de la optimización del rendimiento de las soluciones del problema generalizado de asignación de horarios (CI2).

- El estudio reporta el uso de métodos de comparación de la eficiencia de la solución investigada en el problema generalizado de asignación de horarios (CI3).

En el caso de que no se encuentren suficientes investigaciones en los últimos años, se

tomará en cuenta a las investigaciones con más de cinco años de antigüedad.
En la presente revisión se excluyen los estudios que cumplen con las siguientes normas de exclusión:

- La investigación aborda el tema del problema generalizado de asignación de horarios desde un punto de vista de una evaluación comparativa (benchmarking), más que para proponer alguna solución a los problemas relacionados (CE1).

- El estudio se encuentra escrito en un idioma distinto al español o el inglés, esto debido a que un estudio escrito en un idioma distinto no podrá ser comprendido (CE2).

- El estudio fue realizado con 10 años de antigüedad o más, esto debido a que se busca ejecutar una revisión del contenido más actualizado posible (CE3).

- El estudio no se encuentra en el ámbito de la informática y/o computer science (CE4).

- El estudio se enfoca en aspectos no relacionados a lo que se quiere investigar como, por ejemplo: tiempos de ejecución, análisis de datos, entre otros (CE5).

La Tabla 11 presenta los resultados de aplicar las normas de inclusión y exclusión sobre los resultados de la búsqueda sistemática, de donde se han logrado obtener 14 artículos relevantes.

Tabla 11: Artículos relevantes para la Revisión Sistemática. Fuente: Elaboración propia.

ID	Título	Autor	Año de publicación
S1	A hybrid metaheuristic method for dispatching automated guided vehicles in container terminals	L.Q. Song, S.Y. Huang	2013

S2	A metaheuristic algorithm for project selection and scheduling with due windows and limited inventory capacity	C. Garcia	2014
S3	A novel hybrid genetic algorithm for the open shop scheduling problem	F. Ahmadizar, M.H. Farahani	2012
S4	Application to Order Consolidation and Dynamic Selection of Transshipment Points for Time-Critical Freight Logistics	S. Salhi, B. Gutierrez, N. Wassan, S. Wu , R. Kaya	2020
S5	Cloud Scheduling Using Improved Hyper Heuristic Framework	A. Jain, A. Upadhyay	2019
S6	Comparing four metaheuristics for solving a constraint satisfaction problem for ship outfitting scheduling	C.D. Rose, J.M.G. Coenen	2015
S7	Cross-dock scheduling considering time windows and deadline for truck departures	A. Golshahi-Roudbaneh, M. Hajiaghaei-Keshteli, M.M. Paydar	2021
S8	Effective Metaheuristic Algorithms for Bag-of-Tasks Scheduling Problems under Budget Constraints on Hybrid Clouds	L. Ma, C. Xu, H. Ma, Y. Li, J. Wang, J. Sun	2021
S9	Fast Scheduling of Semiconductor Manufacturing Facilities Using Case-Based Reasoning	J. Lim, M. -J. Chae, Y. Yang, I. -B. Park, J. Lee, J. Park	Feb. 2016
S10	Improvement of container scheduling for Docker using Ant	C. Kaewkasi, K. Chuenmuneewong	2017

	Colony Optimization		
S11	Integrated capacitated transportation and production scheduling problem in a fuzzy environment	A.M. Fathollahi-Fard, M. Hajiaghaei-Keshteli	2018
S12	Optimization, dispatching rules and hyper-heuristics: A comparison in dynamic single machine scheduling	S. Nguyen	29 Oct.-1 Nov. 2017
S13	Production Scheduling with Stock- and Staff-Related Restrictions	C.S. Sartori, V. Gandra, H. Çalık, P. Smet	2021
S14	Scheduling of collaborative operations of yard cranes and yard trucks for export containers using hybrid approaches	H.-P. Hsu, H.-H. Tai, C.-N. Wang, C.-C. Chou	2021

3.8. Formulario de extracción

En la Tabla 12 se presenta el formulario de extracción de datos, el cual ayudará a describir cómo es que cada fuente seleccionada ayudará a contestar las preguntas de investigación planteadas.

Tabla 12: Formulario de extracción. Fuente: Elaboración propia.

Campo	Descripción	Pregunta
Id	E[número] P.ej: E001	General
Fecha de extracción		General
Título		General
Autores		General
Tipo de fuente	Revista, informe, tesis, caso de estudio,	General

	congreso o capítulo de libro	
Año de publicación		General
Base de datos de extracción	Scopus/ Science Direct / IEEE / ACM	General
Abstract		
Link de consulta		
Información relevante		
Causas identificadas del problema	Qué causas se atribuyen al problema generalizado de asignación de horarios	P1
Soluciones encontradas en la solución del problema	Soluciones encontradas por distintos autores para el problema generalizado de asignación de horarios	P2
Variables encontradas en la solución del problema	Qué variables se utilizan en las soluciones encontradas	P2
Restricciones encontradas en la solución del problema	Qué restricciones se utilizan en las soluciones encontradas	P2
Características consideradas en el diseño de la solución del problema	Qué características del problema se han enfatizado y/o tomado en cuenta en el diseño de la solución del problema tratado	P3
Conceptos considerados en el diseño de la solución del problema	Qué conceptos del problema se han enfatizado y/o tomado en cuenta en el diseño de la solución del problema tratado	P3
Algoritmos utilizados en la solución del problema	Qué algoritmos se han utilizado en la investigación para resolver el problema y evaluar los resultados	P4
Formas de evaluación, comparación y verificación de los algoritmos utilizados en la solución del problema	Qué formas de evaluación, comparación y verificación se han utilizado con los algoritmos propuestos en la investigación	P4

Para visualizar la tabla completa véase el Anexo A ubicado al final del documento.

3.9. Resultados de la revisión

3.9.1. Respuesta de la pregunta P1

En la Tabla 13 se aprecian los resultados obtenidos de los documentos primarios seleccionados para responder a la primera pregunta de investigación planteada: "¿Cuáles son las causas de los problemas de asignación de horarios en el despacho de una planta de producción?".

Tabla 13: Causas de los problemas considerados por cada estudio. Fuente: Elaboración propia.

ID	Causas de los problemas de asignación de horarios
S2	- Poca capacidad para abarcar la totalidad de tareas existentes en un horizonte de planeamiento.
S12	- Cambios dinámicos - Interacción con otras decisiones de planeamiento - Múltiples objetivos en conflicto
S13	- Capacidad limitada de producción (No se puede atender toda la demanda)

La Tabla 13 permite conocer distintas causas del problema generalizado de asignación de horarios, cada uno tomando en cuenta el entorno en el cual se encuentra el problema referido. Una causa común que mencionan Garcia (2014) y Sartori (et al., 2021) es la incapacidad de atender la demanda acumulada, de manera que se tiene que priorizar y organizar los trabajos a realizar; mientras que S. Nguyen (2017) propone causas como los cambios dinámicos del entorno, la interacción con otras actividades de planeamiento y los múltiples objetivos en conflicto como las causas del problema generalizado de asignación de horarios.

3.9.2. Respuesta de la pregunta P2

En la Tabla 14 se muestran los resultados obtenidos de los artículos relevantes obtenidos para dar respuesta a la segunda pregunta de investigación que se ha planteado: "¿Qué soluciones se están dando a los problemas de asignación de horarios en el despacho de productos y cuáles fueron las variables y restricciones presentadas?".

Tabla 14: Soluciones, variables y restricciones consideradas por estudio. Fuente: Elaboración propia.

ID	2.1. Soluciones encontradas en el texto	2.2. Variables presentadas en el texto	2.3. Restricciones
S1	-Algoritmo genético/genetic algorithm (GA) -Aneación simulada/simulated annealing (SA) -Búsqueda tabú/tabu search (TS) -Algoritmo glotón/greedy algorithm (G)	-Tamaño de ventana de tiempo -Promedio de valor de retraso por tarea	-Tamaño de la asignación por vehículo -Trabajos independientes
S2	-GA -SA -TS -G -Ramificación y acotación/branch and bound (BB) -Programación entera/integer programming (IP)	-Conjunto de tareas -Ganancia por tarea -Tiempo de procesamiento de tarea -Tamaño de la tarea -Ventana de tiempo -Capacidad de inventario -Penalidad por prontitud o tardanza	-Costo de guardar la tarea en el inventario

S3	-GA -SA -TS -BB -Colonia de Hormigas/ant colony optimization (ACO) -Enjambre de partículas/particle swarm optimization (PSO) -Algoritmo genético hibridado con búsqueda rayo, técnica genética egoísta, entre otros.	-Operación -Tiempo de procesamiento de la operación -Tiempo para completar la operación -Conjunto de espacios por máquina -Conjunto de espacios por operación	
S4	-Algoritmo de búsqueda en una vecindad grande/Large Neighbourhood Search (LNS) basado en GRASP -LNS basado en un GA -Programación entera lineal/Integer linear programming (ILP)	-Número de embarcaciones -Volumen del pedido -Peso del pedido -Ventana de tiempo para la entrega del pedido -Conjunto de potenciales puntos de traslado	-Capacidad multidimensional -Compatibilidad de productos (fragilidad, tamaño) -Heterogeneidad de vehículos -Posibilidad de combinar 2 pedidos -Posibilidad de separar un pedido en varios
S5	-ACO -PSO -Algoritmo FCFS.		
S6			-Colisiones -Disponibilidad de recursos -Ventana de tiempo

S7	-BB -Búsqueda de colonia de virus/virus colony search (VCS) -Optimización de olas de agua/water wave optimization (WWO) -Búsqueda en una vecindad modificada variable/modified variable neighborhood search (MVNS) -GA -Colonia de abejas artificial/artificial bee colony (ABC)	-Número de camiones -Número de productos -Ventana de tiempo -Tiempo máximo de salida	-Fecha límite de salida para productos perecibles
S8	-GA -G -PSO -ACO -Algoritmo de enjambre de hormigas caóticas basado/chaotic ant swarm algorithm (CAS) basado en GA -PSO basado en GA -Optimización lobo gris/grey wolf optimization (GWO) -Algoritmo de polinización de flores/flower pollination algorithm (FPA) -GWO basado en FPA	-Ventana de tiempo -Costo de ejecución de tareas -Capacidad de memoria -Inicio de operación -Fin de operación -Tiempo para completar la operación -Costo del tiempo	

S9	-GA	-Número de máquinas -Tipo de trabajos -Tipo de operaciones -Número de operaciones	-Tiempo de residencia en el almacén -Re-entradas -Variaciones en las actividades
S10	-Algoritmo de embalaje/binpacking algorithm (BA) -Algoritmo todos contra todos/Round-robin algorithm (RR) -G	-Recursos del nodo -Memoria disponible del nodo -Memoria total del nodo -CPU disponible del nodo -CPU total del nodo -Peso de la memoria	
S11	-Imperialism Competitive Algorithm (ICA) -GA -Branch and cut algorithm (BC) -PSO	-Penalidad de entrega temprana de orden -Penalidad de salida temprana de orden -Penalidad de tardanza de orden -Duración del vuelo -Penalidad de tardanza en salida de vuelo -Tiempo de procesamiento de orden	-Salidas ordenadas -Máxima capacidad de transporte
S12	-TS -GA -Algoritmo memético/memetic algorithm (MA)	-Tardanza ponderada total -Ventana de tiempo	
S13		-Días específicos de pedidos -Ventanas de tiempo de tareas	-Retrasos en los pedidos -Capacidad de stock

S14	-ACO -SA -PSO -IP -ILP -SA basado en GA -Programación entera mixta/mixed integer programming (MIP) -Programación lineal entera mixta/mixed integer linear programming (MILP) -Método de ramificación y precio /branch and price method (BP)	-Conjunto de contenedores -Conjunto de camiones -Duración del trabajo de carga -Inicio del trabajo de carga -Fin del trabajo de carga -Interrupción del trabajo -Relocalización de contenedores -Velocidad de trabajo	-El número de contenedores asignados es igual a N -La carga de contenedores está ordenada

La Tabla 14 muestra distintas soluciones que se han estudiado para solucionar el problema generalizado de asignación de horarios, así como distintas variables y restricciones presentadas en las distintas investigaciones.

Dentro de las soluciones más utilizadas en la literatura citada en los artículos relevantes se pueden encontrar los algoritmos genético (Ahmadizar & Farahani, 2012; Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Garcia, 2014; Golshahi-Roudbaneh et al., 2021; J. Lim et al., 2016; L. Q. Song & S. Y. Huang, 2013; Ma et al., 2021; S. Nguyen, 2017), optimización de enjambre de partículas (Ahmadizar & Farahani, 2012; Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Hsu et al., 2021; Jain & Upadhyay, 2019; Ma et al., 2021), genético híbrido (Ahmadizar & Farahani, 2012; Hsu et al., 2021; Ma et al., 2021; Salhi et al., 2020), aneación simulada (Ahmadizar & Farahani, 2012; Garcia, 2014; Hsu et al., 2021; L. Q. Song & S. Y. Huang, 2013) y búsqueda tabú (Ahmadizar & Farahani, 2012; Garcia, 2014; L. Q. Song & S. Y. Huang, 2013; S. Nguyen, 2017); mientras que las restricciones en las que más concuerdan los autores son el costo de inventario (Garcia, 2014; Hsu et al., 2021; J. Lim et al., 2016), la cantidad de productos/actividades asignadas a un vehículo (Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; L. Q. Song & S. Y. Huang, 2013) y la variación de actividades o la capacidad de combinar o separar pedidos (J. Lim et al., 2016; Salhi et al., 2020). Por otro lado, las variables en las que más coinciden los autores son la ventana de tiempo (Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Garcia,

2014; Golshahi-Roudbaneh et al., 2021; Hsu et al., 2021; L. Q. Song & S. Y. Huang, 2013; Ma et al., 2021; Salhi et al., 2020; Sartori et al., 2021) y el número de trabajos que se pueden realizar paralelamente (Garcia, 2014; Golshahi-Roudbaneh et al., 2021; Hsu et al., 2021; J. Lim et al., 2016; Salhi et al., 2020) .

En conclusión, algunos de los algoritmos más utilizados conocidos a través de la literatura son los algoritmos metaheurísticos, lo que refiere buenos resultados de estas soluciones para este tipo de problemas.

3.9.3. Respuesta de la pregunta P3

En la Tabla 15 se muestran los resultados obtenidos de los artículos relevantes obtenidos para dar respuesta a la tercera pregunta de investigación que se ha planteado: "¿Qué características y conceptos se han investigado acerca de la forma de asignar horarios para el despacho de productos?".

Tabla 15: Características y conceptos de cada estudio relacionados al problema de asignación de horarios. Fuente: Elaboración propia.

ID	3.1. Características halladas/investigadas	3.2. Conceptos hallados/investigados
S1		-Retraso en el trabajo: Tiempo fuera del plazo estimado que demora realizar un trabajo
S2		-Horizonte de planeamiento: Tiempo definido desde el inicio del planeamiento hasta el final de la última ventana de tiempo de un trabajo. -Fechas más temprana y alejada de embarque permitido: Fechas límite para la realización de un trabajo sin consecuencias posteriores. -Tiempo de ventaja: Tiempo definido desde el inicio del horizonte de planeamiento hasta el inicio de la primera ventana de tiempo de un trabajo.

		-Ventana de tiempo: tiempo definido entre el inicio y el final planificado de un trabajo.
S3	-NP-hard	
S6		-Restricciones duras/hard constraints: Restricciones que no se deben dejar de lado (ejm: continuidad del trabajo -Restricciones suaves/soft constraints: Restricciones que pueden ser dejadas de lado en caso sea necesario (ejm: ventana de tiempo)
S7		-Método Taguchi: método estadístico utilizado para optimizar procesos en relación con sus especificaciones.
S11	-NP-hard	
S12	-NP-hard	-Hyper-heurística: método de búsqueda heurística que busca automatizar el proceso de seleccionar, combinar, generar o adaptar distintas heurísticas simples para solucionar eficientemente problemas de búsqueda computacional.
S14	-NP-hard	

La Tabla 15 permite observar distintas características y conceptos investigados y aplicados en los distintos proyectos de investigación realizados. La característica que destaca en este tipo de proyectos es el hecho de tener un problema que sea clasificado como NP-hard (Ahmadizar & Farahani, 2012; Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Hsu et al., 2021; S. Nguyen, 2017). El término NP-Hard indica que no se pueden encontrar soluciones óptimas en tiempos polinomiales, por lo que el uso de heurísticas es propicio para el problema en cuestión.

Por otro lado, los conceptos hallados en estas investigaciones incluyen conceptos propios del problema como el retraso de un trabajo (L. Q. Song & S. Y. Huang, 2013), horizonte de planeamiento, fechas de embarque permitido, tiempo de ventaja, ventana de tiempo(Garcia, 2014); conceptos que conciernen a las entradas de algoritmos como

las restricciones duras y restricciones blandas (Rose & Coenen, 2015); conceptos relacionados a la optimización del proceso de pruebas como el método Taguchi (Golshahi-Roudbaneh et al., 2021); y conceptos aplicados para la búsqueda de soluciones como son las hyper-heurísticas (S. Nguyen, 2017).

Gracias a estos resultados se puede verificar que el presente proyecto es clasificado como np-hard, lo que indica que es propicio el uso de algoritmos metaheurísticos para buscar soluciones cercanas a la óptima en tiempos razonables. Por otro lado, los conceptos encontrados sirven para definir la problemática, así como la solución, verificación y comparación con otras soluciones.

3.9.4. Respuesta de la pregunta P4

En la Tabla 16 se muestran los resultados obtenidos de los artículos relevantes obtenidos para dar respuesta a la cuarta pregunta de investigación que se ha planteado: "¿Qué algoritmos metaheurísticos o bioinspirados se están utilizando para la solución del problema generalizado de asignación de horarios aplicado al despacho y cómo se han evaluado, comparado y verificado?".

Tabla 16: Artículos que reportan el uso de algoritmos metaheurísticos y bioinspirados. Fuente: Elaboración propia.

ID	4.1. Algoritmos utilizados en la solución	4.2. Forma de evaluación, comparación y verificación de la solución
S1	-Algoritmo híbrido (GA combinado con SA y TS)	-Se evaluaron los algoritmos independientes (GA, SA, TS, G) y las siguientes combinaciones (GA+SA, GA+TS y GA+SA+TS) en un todos contra todos, mientras se revisaban los mejores resultados de las variables antes mencionadas.
S2	-G -Meta-RaPS (algoritmo glotón con randomizado controlado en una heurística de priorización)	-Se evaluaron los algoritmos antes mencionados (G y Meta-RaPS) variando los siguientes factores de diseño: -Número de trabajos -Pico de demanda múltiple -Tiempo de ventaja múltiple -Límite de inventario múltiple

S3	-GA hibridado con operadores específicos del problema	-Testeado con los benchmarks propuestos por Taillard, Bucker y Guéret y Prins.
S4	-Algoritmo glotón basado en GRASP -ILP	-Se obtuvieron datos de ejemplo de una compañía, con los cuales se probaron el algoritmo GRASP y ILP, mientras que también se puso a prueba el algoritmo GRASP contra el método manual.
S5		-Comparación de algoritmos tradicionales con algoritmos metaheurísticos a partir de 3 diferentes datasets que incrementan en tamaño.
S6	-GA -SA -PSO -Algoritmo genético hibridado con aneación simulada (GSA)	-Resolución del problema por cada uno de los algoritmos seleccionados, analizando la calidad de cada uno de los horarios obtenidos.
S7	- Evolución diferencial/Differential Evolution (DE) - SA - Algoritmo de keshtel/keshtel algorithm (KA) -Algoritmo híbrido basado en KA y SA	-Cada algoritmo fue utilizado en 10 conjuntos de datos generados de manera aleatoria, para luego transformar los resultados a través de un RPD (Robust Parameter Design), donde el menor valor indica un mejor resultado.
S8	-Metaheurísticas basadas en GA	-Se realizaron pruebas entre los distintos IGA (improved GA) variando distintos parámetros (5 corridas por IGA), para luego realizar una prueba de ANOVA.
S9	-Razonamiento basado en el caso/case based reasoning (CBR) (Modelo formal propuesto bajo la utilización de redes Petri)	-Se prepararon 12 datasets con diferentes configuraciones, probandolas con el algoritmo propuesto y el GA.
S10	-Ant Colony Optimization	-1 sola estructura de 12 servidores NGINX distribuidos en 5 nodos, desplegados con los algoritmos greedy y ACO para observar su comportamiento.

S11	-KA -VCS	-Se utilizó un conjunto de parámetros elegidos a través del método Taguchi, para luego utilizar este conjunto en los siguientes algoritmos: GA, KA, PSO y VCS. Luego se utilizó un test de ANOVA para poder determinar el mejor
S13	-Late Acceptance Hill-Climbing (LAHC) heuristic -Programación entera lineal	-Se utilizaron 10 sets de datos con los cuales se compararon los resultados obtenidos por ambos algoritmos
S14	-MILP -Algoritmos basado en GA -PSO -Algoritmos basados en PSO (SubGroupsPSO)	-Se realizaron experimentos con conjuntos de datos con parámetros variados, para luego comparar los resultados

La Tabla 16 permite conocer los algoritmos utilizados a lo largo de los últimos años, los cuales han sido evaluados y/o comparados de las diferentes formas.

Los algoritmos más utilizados en las soluciones de las investigaciones reunidas son los algoritmos genéticos y genéticos hibridados (Ahmadizar & Farahani, 2012; Garcia, 2014; Hsu et al., 2021; L. Q. Song & S. Y. Huang, 2013; Ma et al., 2021; Rose & Coenen, 2015), seguidos de los algoritmos basados en aneación simulada (Golshahi-Roudbaneh et al., 2021, p.; L. Q. Song & S. Y. Huang, 2013; Rose & Coenen, 2015) y los algoritmos basados en el algoritmo de Keshtel (Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Golshahi-Roudbaneh et al., 2021). Por otro lado, la evaluación de los algoritmos se ha realizado en su mayoría a través de una comprobación directa, comparando los resultados obtenidos de distintos datasets o combinaciones de variables -los cuales pueden ser obtenidos a través del método Taguchi (Fathollahi-Fard & Hajiaghaei-Keshteli, 2018)- o usar benchmarks ya propuestos (Ahmadizar & Farahani, 2012), para luego verificar los resultados a través de un test de anova (Fathollahi-Fard & Hajiaghaei-Keshteli, 2018; Ma et al., 2021).

Estos resultados servirán para determinar un algoritmo con el cual comparar la solución propuesta en esta investigación, mientras que sirve como guía para realizar las comprobaciones y verificaciones del caso.

3.10. Conclusiones

La presente revisión de literatura ha permitido tener una visión clara y objetiva de la investigación realizada en la búsqueda de soluciones para el problema generalizado de asignación de horarios.

Se han resuelto las preguntas de investigación planteadas, al mismo tiempo que se ha verificado la falta de investigaciones que tomen en cuenta soluciones del problema de asignación de horarios aplicado al despacho utilizando algoritmos recientes, puesto que la mayoría de literatura propone soluciones a partir de algoritmos bastante utilizados adaptando las restricciones a un problema específico. Es así que el algoritmo genético (GA) surge como uno de los algoritmos más utilizados, gracias a su adaptabilidad y capacidad de generar buenos resultados. Por otro lado, se ha observado que pocas investigaciones utilizan un método formal para obtener datos de pruebas (como el método Taguchi). A partir de lo mencionado, se tiene por finalidad cubrir el vacío identificado.

Capítulo 4. Parámetros, restricciones y función objetivo

4.1. Introducción

En este capítulo se desarrolla el objetivo 1, el cual se compone de los resultados esperados R1.1 y R1.2. El primero de estos resultados esperados radica en la determinación de los parámetros y restricciones para el problema presentado de planificación, los cuales sirven como base para el segundo resultado esperado, el cual busca formular una función objetivo acorde al problema, de manera que se evalúen las distintas soluciones que se puedan encontrar.

4.2. Resultados alcanzados

En este acápite se presenta en detalle los pasos realizados para el logro de los resultados esperados R1.1 ("Lista de definición de parámetros y restricciones del problema en cuestión") y R1.2 ("Formulación de la función objetivo para el problema de planificación").

4.2.1. Lista de definición de parámetros y restricciones del problema de planificación.

El contenido de la presente sección está dividido en dos partes complementarias que sirven para lograr el resultado esperado R1.1. Los parámetros del problema y sus restricciones permitirán validar si una propuesta de planificación es adecuada y puede ser aceptada como una solución.

4.2.1.1. Parámetros del problema

En base al análisis del problema que se busca resolver, se han identificado el conjunto de parámetros presentado a continuación, el cual se puede apreciar en la Tabla 17.

Tabla 17: Parámetros identificados. Fuente: Elaboración propia.

Abreviatura	Nombre	Descripción
HP	Horizonte de planificación	El lapso en el cual se realizará la planificación.
HT	Número de horas de trabajo por día	Número de horas de trabajo por día. Para efectos del problema se tomará un día completo (24 horas) sin interrupciones.

B	Número de bahías	Cantidad de bahías en las que se puede realizar un despacho.
P	Número de pedidos	Número de pedidos a entregar.
FMD	Fecha máxima de despacho	Fecha máxima donde se puede realizar un despacho de un pedido.
TC	Tiempo de carga	Tiempo que se demora la realización del proceso de carga de un pedido.
TiD	Tipo de despacho	Despacho nacional o internacional.
TuD	Turnos de despacho	Turno en el que se realiza el despacho.

4.2.1.2. Restricciones del problema

Luego de determinar los parámetros necesarios para la solución del problema propuesto, se han identificado las condiciones que estos deben cumplir, con la finalidad de que puedan ser útiles para su uso en la solución propuesta.

1. El número de horas de trabajo por día no puede ser mayor a 24 horas:

   HT <= 24

2. La cantidad de turnos de despacho no puede ser menor a 1 y no puede ser mayor a el número de horas de trabajo por día:

   HT >= TuD >= 1

3. La cantidad de bahías no puede ser menor o igual a 0:

   B > 0

4. La sumatoria de Tiempos de Carga en cada bahía no puede ser mayor a las Horas de Trabajo por día multiplicado por el Horizonte de Planificación.

   $$\forall \text{ b } \in \text{B: } (\sum_{p=1}^{P'} T\text{□}_p)\text{□} \leq (HT * HP)\text{□}$$

5. La hora de carga de un pedido (HC) no puede ser menor que 0 o mayor que el horizonte de planificación.

   $$\forall \text{ p} \in \text{pedidos: } H\text{□}_p < HP$$

6. Un pedido con un turno especificado solo puede ser colocado en ese turno.

7. La fecha de despacho de un pedido no puede ser mayor a la fecha máxima de despacho de este.

4.2.2. Formulación de la función objetivo para el problema de planificación

El objetivo principal del problema de planificación de horarios es minimizar el tiempo en que las bahías no están despachando pedidos, es decir, se espera que todas las bahías estén despachando pedidos de manera continua. Al mismo tiempo, se espera que la carga de trabajo para todas las bahías sea proporcional. De esta manera, se puede cumplir con las exigencias de los clientes, los cuales tienden a recibir sus pedidos en un horario establecido, mientras que la empresa utiliza sus bahías de una forma más eficiente.

Para formalizar lo expresado anteriormente, se ha formulado la siguiente función objetivo:

$$\text{Minimizar} \sum_{B} (H_{max} + \text{TCPB} - H_{max.} + \text{HCPB}) - \sum_{p'} T_{p'}$$

Donde TCPB es el tiempo promedio de los tiempos de carga (TC) asignados a una bahía, HCPB representa el promedio de las horas de carga y tiempos muertos en una bahía, y $H_{max.}$ representa la hora de carga máxima (o última) de un pedido en una bahía.

Esta función objetivo va a ser utilizada en los dos algoritmos que se van a implementar para realizar la puntuación de las soluciones generadas.

En el Anexo D – Pruebas de la función objetivo se presenta la evolución de esta función objetivo a partir de distintas validaciones realizadas.

4.2.2.1. Validaciones realizadas

Se ha realizado la evaluación de distintos resultados para el problema propuesto, utilizando la función objetivo presentada y validando que los resultados obtenidos son acordes a lo esperado. Esto quiere decir que, cuando se presenta una solución mejor que otra, el valor que devuelve la función objetivo disminuye, mientras que, en caso contrario, aumenta. En el anexo D se describe con mayor profundidad las pruebas efectuadas.

El presente capítulo ha sido verificado por el especialista en logística, el cual ha brindado su aprobación referente a la información presentada. En el Anexo C – Acta de conformidad de parámetros y restricciones y Anexo E – Acta de conformidad de la función objetivo se adjuntan las actas de conformidad.

4.3. Discusión

En este acápite se han obtenido los resultados esperados R1.1 y R1.2 en los ítems 4.2.1. y 4.2.2. respectivamente.

El ámbito de estos resultados concede, a partir del planteamiento inicial de los parámetros, restricciones y la función objetivo, sentar las bases para el diseño e implementación de las estructuras de datos y los algoritmos a utilizar en esta investigación.

De esta manera, este trabajo de investigación mantiene coherencia con trabajos anteriores como los listados en el Estado del Arte, los cuales tienen como objeto de estudio el uso de uno o más algoritmos metaheurísticos para la solución de problemas de similar complejidad, iniciando la investigación con la determinación de parámetros, restricciones y la función objetivo para el problema central del estudio.

Las conclusiones alcanzadas fueron realizadas para solucionar el problema de planificación del estudio; no obstante, estos pueden ser modificados para buscar soluciones de problemas de planificación en un contexto distinto, como problemas de planificación de citas, puesto que estos escenarios comparten similitudes desde su definición.

Para finalizar, cabe resaltar que los resultados alcanzados en este acápite tienen como enfoque la revisión individual de una solución; lo que quiere decir que, para procesar un conjunto de datos, se debe tratar cada potencial solución de manera individual, obteniendo así una planificación para cada una de las potenciales soluciones. Otro tipo de acercamiento a distintas soluciones no están contempladas como parte de esta investigación.

Capítulo 5. Estructuras de datos

5.1. Introducción

En este epígrafe se procederá a realizar tareas relacionadas a los resultados esperados R2.1 y R3.2, donde se realiza el bosquejo de las estructuras de datos necesarias para resolver el problema de planificación utilizando el algoritmo genético (GA) y búsqueda de colonia de virus (VCS). Se presentan también las estructuras auxiliares que dan soporte a los algoritmos presentados.

5.2. Resultados alcanzados

En este punto se presentan los resultados obtenidos al trabajar los resultados esperados R2.1: "Diseño de estructuras de datos que sirven de soporte para el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación" y R3.2: "Adaptación del algoritmo genético al problema de planificación", trabajo que servirá para la codificación de los algoritmos antes mencionados.

5.2.1. Diseño de estructuras de datos para el algoritmo genético adaptado al problema de planificación

En el algoritmo genético usado para la resolución de este problema, la estructura cromosoma es la representación de un posible resultado. Para lograr esta representación, se ha utilizado un arreglo unidimensional, donde cada bloque representa a un pedido que debe ser despachado en una bahía. Teniendo en consideración lo mencionado anteriormente, se ha definido la siguiente estructura para representar los cromosomas:

Pedido 1	Pedido 2	Pedido 3	Pedido 4	…	Pedido n
Bahía 2	Bahía 7	Bahía 5	Bahía 1	…	Bahía 3

Ilustración 2: Representación de un cromosoma. Fuente: Elaboración propia.

El arreglo presentado en la Ilustración 2 muestra los distintos pedidos y su asignación a distintas bahías, siendo la cantidad de pedidos la cantidad máxima entre el total de pedidos que se puede entregar y el total de pedidos en espera.

Este arreglo proporciona una asociación entre las bahías y los pedidos, donde cada pedido es asignado a una bahía, lo que permite obtener un horario de despacho para cada bahía en su posterior evaluación.

5.2.2. Diseño de estructuras de datos que sirven de soporte para el algoritmo búsqueda de colonia de virus

Para el algoritmo de búsqueda de colonia de virus, la estructura que define una posible

solución es una representación de un virus, la cual también se puede representar en forma de un arreglo, de manera similar al algoritmo genético.

Pedido 1	Pedido 2	Pedido 3	Pedido 4	…	Pedido n
Bahía 2	Bahía 7	Bahía 5	Bahía 1	…	Bahía 3

Ilustración 3: Representación de un virus. Fuente: Elaboración propia

Como se puede observar en la Ilustración 3, se utiliza la misma estructura para representar un virus y para representar un cromosoma, a pesar de ser utilizados en algoritmos distintos. De esta manera, se mantienen los atributos mencionados anteriormente, de manera que se pueden obtener soluciones a partir de los mismos métodos en ambos algoritmos.

5.2.3. Estructuras de datos auxiliares

A parte de las estructuras base para cada uno de los algoritmos, como son el cromosoma para el algoritmo genético y el virus para el algoritmo de búsqueda de colonia de virus, se han planteado otras estructuras auxiliares para la implementación de ambos algoritmos, las cuales se presentan en el Anexo F - Estructuras auxiliares. La conformidad con el presente capítulo se presenta en el acta adjunta en el Anexo G - Acta de conformidad de las estructuras bioinspiradas.

5.3. Discusión

En este capítulo se han completado tareas relacionadas a los resultados esperados R3.2 y se ha alcanzado el resultado esperado R2.1, habiendo así definido las estructuras bioinspiradas necesarias para el almacenamiento y actualización de datos utilizados en los algoritmos implementados en esta investigación.

El alcance de estos resultados nos permite representar soluciones en ambos algoritmos, donde su diseño permite tener flexibilidad al momento de armar posibles soluciones dentro de los algoritmos, lo que permite obtener una mejor calidad de las soluciones.

A comparación de otras investigaciones, se ha procedido con la definición y explicación de las estructuras a utilizar, puesto que esta sección es omitida en obras recaudadas en el estado del arte, donde luego de explicar el problema, sus restricciones y función objetivo, pasan a explicar la adaptación de los algoritmos utilizados, lo que dificulta la comprensión en la mayoría de los casos.

Cabe resaltar que las estructuras presentadas están planteadas para el problema presentado en esta investigación, pero pueden ser modificadas y utilizadas en otro tipo de problemas similares. Como ejemplo, un problema de planificación de citas puede

utilizar las mismas estructuras con distintas nomenclaturas en sus variables.

Capítulo 6. Adaptación del algoritmo genético para el problema de planificación

6.1. Introducción

En este capítulo se completará el resultado esperado R3.2, el cual presenta el pseudocódigo del algoritmo genético a utilizar en la problemática de planificación seleccionado en esta investigación. Se presenta el detalle de las funciones de creación de la población inicial, generación de una nueva población, recombinación, mutación de los genes, adaptación, validación de la posible solución y el cálculo de la función fitness.

6.2. Resultados alcanzados

En este capítulo se muestra el trabajo realizado con le objetivo completar el resultado esperado R3.2: “Adaptación del algoritmo genético al problema de planificación”.

6.2.1. Algoritmo genético aplicado al problema de planificación

Se presenta el boceto del algoritmo genético adaptado para el problema seleccionado en esta investigación, dividido en los principales métodos que componen al algoritmo.

6.2.1.1. Pseudocódigo del algoritmo genético

```
Algoritmo AlgoritmoGenetico:
    pedidosPlanificacion = FiltrarPedidos(pedidos)
    cromosomaInicial = Cromosoma(pedidos, horizontePlanificacion, cantidadHorasDia)
    generacion = Generacion()
    generacion.inicializarPoblacion(cromosomaInicial, tamañoPoblacion)
    generacion.calcularFitnessPoblacion()
    mejorCromosoma = generación.obtenerMejorCromosoma()
    contadorGeneraciones = 0
    Mientras contadorGeneraciones < generacionMaxima:
        generacion.obtenerNuevaGeneracion(numeroParticipantesRuleta)
        generacion.calcularFitnessPoblacion()
        Si generacion.obtenerMejorFitness() < mejorCromosoma.obtenerFitness():
            mejorCromosoma = generacion.obtenerMejorCromosoma()
        FinSi
    FinMientras
FinAlgoritmo
```

Ilustración 4: Pseudocódigo del algoritmo genético. Fuente: Elaboración propia.

En la Ilustración 4 se pone a disposición el pseudocódigo de la función principal del algoritmo genético, donde recibe los pedidos que se tienen para realizar la planificación (pedidos), el horizonte de planificación (horizontePlanificacion), la cantidad de horas por día de despacho (cantidadHorasDia), el tamaño de la población (tamañoPoblacion), la cantidad máxima de generaciones con las que se va a trabajar (generacionMaxima, actúa como condición de parada del algoritmo) y el número de participantes de la ruleta (numeroParticipantesRuleta).

En primer lugar, se filtran los pedidos que van a ser utilizados en la planificación (línea 2), puesto que, si la cantidad de pedidos excede la capacidad de las bahías en el horizonte de planificación estimado, no todos los pedidos podrán ser entregados, por lo que se priorizan los pedidos con una fecha de despacho máxima más cercana a la fecha de inicio de la planificación.

Seguido de la filtración de los pedidos a utilizar en la planificación, se inicializa el cromosoma inicial (línea 3), lo que permite crear la población inicial del algoritmo (línea 5). Realizada la generación de la población inicial, se procede a calcular el fitness de cada individuo de la población (línea 6), lo que, a su vez, permite obtener el mejor cromosoma (que representa a la mejor solución obtenida) de la generación (línea 7).

A continuación, se inicia con el ciclo evolutivo generando una nueva población de individuos (línea 10), luego se calcula el fitness de la población (línea 11) y se verifica si se encontró una mejor solución (línea 12). En caso sea así, se procede a guardar esta solución (línea 13). Este ciclo evolutivo se repite hasta alcanzar el tope de generaciones permitidas (línea 9). Finalmente, llegado el tope de generaciones permitidas, la mejor solución obtenida está guardada en la variable “mejorCromosoma”.

6.2.1.2. Construcción de la población inicial

```
1  Algoritmo Generacion::inicializarPoblacion:
2      numeroGeneracion = 0
3      Para i = 0 Hasta tamañoPoblacion:
4          nuevoCromosoma = cromosoma
5          mezclarGenes(nuevoCromosoma)
6          listaCromosomas.agregar(nuevoCromosoma)
7      FinPara
8  FinAlgoritmo
```

Ilustración 5: Construcción de la población inicial del algoritmo genético. Fuente: Elaboración propia.

En la Ilustración 5 se presenta la generación de la población inicial, la cual toma como argumento el cromosoma creado inicialmente en el acápite anterior, para luego copiarlo en un nuevo individuo (línea 4), obtener genes distintos al original (línea 5) y agregar el nuevo cromosoma a la lista de cromosomas a utilizar en el algoritmo (línea 6). Todo lo explicado anteriormente se ejecuta hasta obtener el número definido de individuos en una población, número marcado por la variable tamañoPoblación (línea 3).

6.2.1.3. Método de recombinación

```
1  Algoritmo Cromosoma::recombinacion:
2      rnd = obtenerEnteroAleatorio(tamañoGenes)
3      hijo1 = padre1[:rnd] + padre2[rnd:]
4      hijo2 = padre2[:rnd] + padre1[rnd:]
5      hijos = [hijo1, hijo2]
6      Retornar hijos
7  FinAlgoritmo
```

Ilustración 6: Recombinación de cromosomas en el algoritmo genético. Fuente: Elaboración propia.

La función presentada en la Ilustración 6 sirve para la recombinación de los cromosomas, donde se presenta una recombinación (o crossover) de punto, donde se obtiene un punto aleatorio entre el inicio y el final de los genes (línea 2) para partir los cromosomas en dos partes. A partir de este punto aleatorio, el primer hijo que se obtiene es la parte inicial del primer padre, mientras que la parte restante la hereda del segundo padre (línea 3). De manera similar, el segundo hijo obtenido obtiene su primera parte heredándola del segundo padre, mientras que la parte restante se obtiene del primer padre (línea 4). Obtenidos los dos hijos, estos se juntan en una tupla (línea 5) y se retornan (línea 6) para poder ser utilizados en la siguiente población.

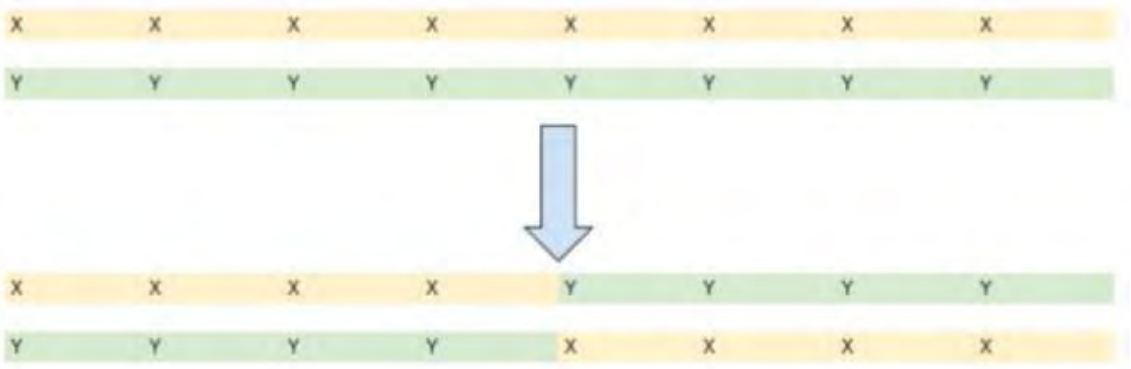

Ilustración 7:Proceso de recombinación del algoritmo genético. Fuente: Elaboración propia

La Ilustración 7 muestra gráficamente el proceso de recombinación utilizado en el algoritmo genético.

6.2.1.4. Método de mutación

```
1  Algoritmo Generacion::mutacion:
2      nuevoCromosoma = cromosoma
3      Si obtenerDoubleAleatorio < probabilidadMutacion:
4          indice1 = obtenerEnteroAleatorio(tamañoGenes)
5          indice2 = obtenerEnteroAleatorio(tamañoGenes - indice1) + indice1
6          genesAuxiliar = cromosoma.obtenerGenes()[indice1:indice2]
7          revertir(genesAuxiliar)
8          nuevoCromosoma.genes = cromosoma.obtenerGenes()[:indice1] +
9              genesAuxiliar + cromosoma.obtenerGenes()[indice2:]
10     FinSi
11     Retornar nuevoCromosoma
12 FinAlgoritmo
```

Ilustración 8: Método de mutación del algoritmo genético. Fuente: Elaboración propia.

La función de mutación presentada en la Ilustración 8 es una mutación por inversión, donde primero se verifica la probabilidad de mutación (línea 3). En caso la probabilidad de mutación obtenida sea menor que la probabilidad de mutación establecida para el algoritmo se procede a obtener dos puntos distintos aleatorios (líneas 4 y 5), a partir de los cuales se obtiene una parte del cromosoma original delimitada por los puntos obtenidos anteriormente (línea 6), la cual es invertida (línea 7) y reemplazada en su posición original (línea 8). Finalmente se retorna al algoritmo genético el nuevo cromosoma generado (línea 11) con una parte de sus genes invertidos.

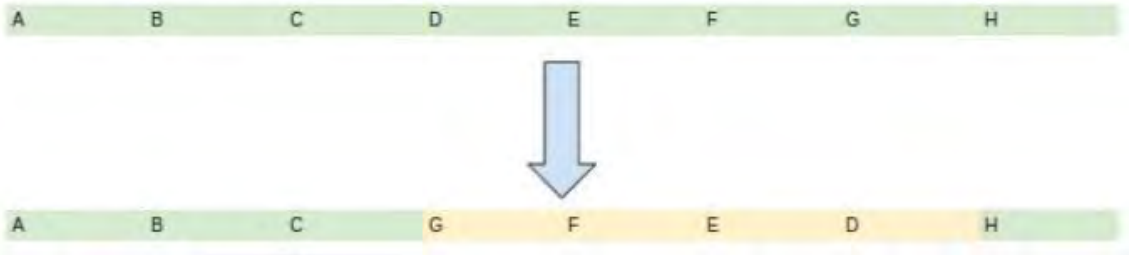

Ilustración 9: Proceso de mutación en el algoritmo genético. Fuente: Elaboración propia

La Ilustración 9 muestra gráficamente el proceso de mutación utilizado en el algoritmo genético.

6.2.1.5. Adaptación de los genes

```
1  Algoritmo Generacion::mutacion:
2      nuevoCromosoma = cromosoma
3      Si obtenerDoubleAleatorio > probabilidadMutacion:
4          indice1 = obtenerEnteroAleatorio(tamañoGenes)
5          indice2 = obtenerEnteroAleatorio(tamañoGenes)
6          nuevoCromosoma.obtenerGenes()[indice1] = cromosoma.obtenerGenes()[indice2]
7      FinSi
8      Retornar nuevoCromosoma
9  FinAlgoritmo
```

Ilustración 10: Adaptación de cromosomas en el algoritmo genético. Fuente: Elaboración propia

La función de adaptación presentada en la Ilustración 10 nos muestra una adaptación de un cromosoma, donde primero se primero se verifica la probabilidad de mutación (línea 3). En caso la probabilidad de mutación obtenida sea mayor que la probabilidad de mutación establecida para el algoritmo se procede a obtener dos puntos distintos aleatorios (líneas 4 y 5), a partir de los cuales se realiza una copia entre estos espacios de los genes (línea 6). Finalmente se retorna al algoritmo genético el nuevo cromosoma generado (línea 8) con una de las bahías asignadas distinta al gen original.

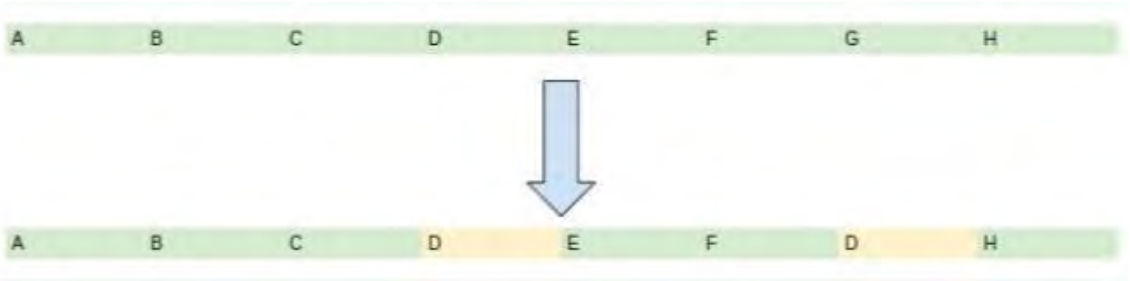

Ilustración 11: Proceso de adaptación en el algoritmo genético. Fuente: Elaboración propia.

La Ilustración 11 muestra gráficamente el proceso de adaptación en el algoritmo genético.

6.2.1.6. Validación de la solución

```
1  Algoritmo Cromosoma::esValido:
2      Para bahía en bahías:
3          horario = Horario()
4          pedidosObligatorios = FiltrarPedidosObligatorios(pedidos)
5          Para pedido en pedidosObligatorios:
6              estaLleno = horario.asignarPedido(pedido)
7              Si no estaLleno:
8                  pedido.horaCarga = horario.horarioAsignado(pedido)
9              Sino
10                 Retornar Falso
11             FinSi
12         FinPara
13         pedidosNormales = FiltrarPedidosNormales(pedidos)
14         Para pedido en pedidosNormales:
15             estaLleno = horario.asignarPedido(pedido)
16             Si no estaLleno:
17                 pedido.horaCarga = horario.horarioAsignado(pedido)
18             Sino
19                 Retornar Falso
20             FinSi
21         FinPara
22     FinPara
23     Retornar Verdadero
24 FinAlgoritmo
```

Ilustración 12: Validación del cromosoma en el algoritmo genético. Fuente: Elaboración propia.

Para validar si una solución es factible o no, se utiliza el método de validación mostrado

en la Ilustración 12, donde se verifica para todas las bahías (línea 2) que los pedidos asignados que tengan un turno, los cuales se almacenan en la variable pedidosObligatorios (línea 4), puedan ser asignados a un horario que cumpla con las especificaciones del pedido. Para esto se usa la estructura auxiliar horario (línea 3), la cual sirve para verificar si un espacio en el horario que cumpla con las especificaciones está disponible. En caso ningún espacio del horario que cumpla con las especificaciones de turnos y fechas de entrega esté disponible, la solución se descarta (línea 10). El mismo proceso se sigue con los pedidos que no tienen una especificación de turno, los cuales se almacenan en la variable pedidosNormales (línea 13). Si se da el caso que todos los pedidos pueden ser asignados a un periodo de tiempo, la solución es aceptada y se retorna el valor de verdadero (línea 23), para su posterior uso en el algoritmo.

6.2.1.7. Cálculo del fitness

```
1  Algoritmo Cromosoma::calcularFitness:
2      horario = CrearHorario(cromosoma.obtenerGenes())
3      Si no esValido(horario):
4          Retornar MaximoDouble - 1
5      FinSi
6      valorFitness = 0
7      promedioTrabajoTotal = obtenerPromedioHoras(horario)
8      promedioCarga = obtenerPromedioCarga(horario)
9      Para bahía en horario:
10         totalHorasCarga = horario.obtenerTotalHorasCarga()
11         totalHorasTrabajo = horario.obtenerTotalHorasTrabajo()
12         valorFitness += totalHorasTrabajo +
13             absoluto(promedioTrabajoTotal - totalHorasTrabajo) +
14             absoluto(promedioCarga - totalHorasCarga)
15     FinPara
16     Retornar valorFitness
17 FinAlgoritmo
```

Ilustración 13: Cálculo del fitness en el algoritmo genético. Fuente: Elaboración propia.

El cálculo del fitness, en el algoritmo propuesto, se realiza a partir de la función objetivo definida en el capítulo 4, donde podemos observar que primero se calcula el promedio del total de tiempo de carga por bahía (línea 8) y el promedio de tiempo de horas de trabajo por bahía (línea 7), para luego realizar la sumatoria de la cantidad total de horas de trabajo (línea 11), junto con el valor absoluto de la diferencia entre el promedio de horas de trabajo de todas las bahías y las horas de trabajo de una sola bahía (línea 13), agregado al valor absoluto de la diferencia entre el promedio de horas de carga de todas las bahías y las horas de carga asignadas a una sola bahía (línea 14). Finalmente, se retorna el valor del fitness obtenido (línea 16) para ser utilizado en el algoritmo. Cabe resaltar que si se encuentra que el cromosoma no es válido (línea 3), el valor del fitness

toma un valor muy alto, el cual va a hacer que el algoritmo no tome en cuenta la solución.

6.2.2. Validaciones realizadas

Este acápite ha sido verificado por el versado en algoritmos, quien ha expresado su aprobación en cuanto a todo lo descrito en este capítulo. En el Anexo H - Acta de conformidad de la adaptación del algoritmo genético se pueden observar las pruebas.

6.3. Discusión

En este epígrafe se completaron las tareas designadas para obtener el resultado esperado R3.2: “Adaptación del algoritmo genético al problema de planificación”, el cual busca adaptar el algoritmo genético al problema presentado en esta investigación.

El trabajo realizado permite entender el funcionamiento general del algoritmo genético, así como el diseño base del algoritmo para su posterior implementación. Una consideración importante es la verificación del uso de las estructuras desarrolladas en el capítulo anterior, las cuales sirvieron para poder trabajar con facilidad y limpieza este capítulo. Finalmente, con la finalización de este capítulo se ha sentado la base para la implementación del algoritmo genético.

Dentro de las investigaciones recopiladas en el Estado del Arte, se ha logrado identificar que la mayoría de estos detallan la adaptación de los algoritmos que utilizan, además de las modificaciones realizadas para optimizar las soluciones obtenidas. En el caso de esta investigación, se realizará una calibración de variables en un capítulo posterior.

De igual manera, todo el trabajo realizado en este capítulo puede ser adaptado y utilizado en problemas similares, utilizando la adaptación del algoritmo genético como se presenta o modificando alguna de sus partes, como puede ser la recombinación, utilizando otro tipo como la recombinación uniforme o de dos puntos.

Capítulo 7. Diseño del algoritmo de búsqueda de colonia de virus

7.1.Introducción

En este acápite se expone el resultado esperado R2.2, donde se busca diseñar un algoritmo de búsqueda de colonia de virus adaptado al problema presentado en esta investigación. Para esto, se presenta un pseudocódigo general, tomando en cuenta los métodos principales del algoritmo como son la inicialización de la población, la difusión de los virus, la infección de células huésped, la respuesta inmune y la selección de virus sobrevivientes.

7.2. Resultados alcanzados

En el presente acápite se busca desarrollar el resultado esperado R2.2., donde se busca diseñar el algoritmo de búsqueda de colonia virus aplicado al problema de planificación presentado en esta investigación.

7.2.1. Algoritmo de búsqueda de colonia de virus aplicado al problema de planificación

A continuación, se presenta el pseudocódigo del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación que se busca resolver en esta investigación.

7.2.1.1. Pseudocódigo del algoritmo Búsqueda de Colonia de Virus

```
Algoritmo BusquedaColoniaVirus:
    generacion = 0
    pedidosPlanificacion = FiltrarPedidos(pedidos)
    virusInicial = Virus(pedidosPlanificacion, horizontePlanificacion, cantidadHorasDia)
    generacion = Generacion()
    generacion.inicializarPoblacion(VirusInicial, tamañoPoblacion)
    generacion.calcularFuncionObjetivoPoblacion()
    mejorVirus = generación.obtenerMejorVirus()
    contadorGeneraciones = 0
    Mientras contadorGeneraciones < generacionMaxima:
        difusion()
        infeccionCelular()
        respuestaInmune()
        generacion.calcularFuncionObjetivoPoblacion()
        Si generacion.obtenerMejorFuncionObjetivo() < mejorVirus.obtenerFuncionObjetivo():
            mejorVirus = generacion.obtenerMejorVirus()
        FinSi
        contadorGeneraciones = contadorGeneraciones + 1
    FinMientras
FinAlgoritmo
```

Ilustración 14: Pseudocódigo del algoritmo Búsqueda de Colonia de Virus. Fuente: Elaboración propia.

En la Ilustración 14se presenta el pseudocódigo base del algoritmo de búsqueda de colonia de virus, el cual utiliza los pedidos que se deben entregar para obtener una lista de pedidos que se pueden entregar, dependiendo de la capacidad de las bahías (línea

3). A continuación, se utilizan estos pedidos para crear un virus inicial (línea 4), el cual servirá para crear una población inicial (línea 6), con la cual se puede empezar a trabajar en el algoritmo.

Construida la población inicial, se toma el mejor virus (línea 8) como referencia para las siguientes generaciones.

Finalmente, para todas las generaciones (hasta alcanzar el límite generacional) se realizan los procesos de difusión de los virus (línea 11), infección celular (línea 12) y respuesta inmune (línea 13). Después de realizar estos procedimientos y calcular la función objetivo de todos los virus en la población (línea 14), se busca un mejor virus (línea 15) para reemplazar el antiguo mejor virus (línea 16) y buscar en la siguiente generación (línea 18).

7.2.1.2. Construcción de la población inicial

```
1  Algoritmo BCV::inicializarPoblacion:
2      Para i = 0 Hasta tamañoPoblacion:
3          nuevoVirus = virus
4          mezclar(nuevoVirus)
5          listaVirus.agregar(nuevoVirus)
6      FinPara
7  FinAlgoritmo
```

Ilustración 15: Inicialización de población de virus. Fuente: Elaboración propia.

La inicialización de la población se realiza tomando un virus inicial, el cual es copiado hacia un nuevo virus (línea 3), para luego barajar el virus (línea 4) y agregar el nuevo virus a la población (línea 5).

7.2.1.3. Difusión de virus

```
1  Algoritmo BCV::difusion:
2      Para i en len(poblacion):
3          sigma = (log(generacion + 1) / maxGeneracion) * (virus[i] - mejorVirus)
4          virus'[i] = Gaussiano(Gaussiano(mejorVirus, sigma))
5      FinPara
6      Evaluacion(virus')
7      Actualizar(virus, virus')
8  FinAlgoritmo
```

Ilustración 16: Difusión de virus del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia.

Para el proceso de difusión de virus, se realiza el cálculo de una sigma (línea 3), la cual nos servirá para crear nuevos virus (línea 4), hasta completar una población (línea 2). Luego de crear una nueva población de virus, se evalúan (línea 6) para verificar que son soluciones aptas para el problema y actualizar la población (línea 7).

7.2.1.4. Infección de células huésped

```
1  Algoritmo BCV::infeccionCelular:
2      promedio = calcularPromedio(virus)
3      sigma = sigmaPoblacion * ( 1 - (generacion+1) / maxGeneracion )
4      virus' = []
5      Para i en len(poblacion):
6          nuevoVirus = promedio + sigma * arregloNormal(0,1, tamañoGenes)
7          regularizar(nuevoVirus)
8          virus'[i] = nuevoVirus
9      FinPara
10     Actualizar(poblacion, virus')
11 FinAlgoritmo
```

Ilustración 17: Infección de células huésped en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia.

Para el proceso de infección celular, se calcula el promedio de todos los espacios en un virus (línea 2), los cuales representan las bahías en el problema investigado, y una sigma (línea 3), variables que serán utilizadas para la creación de un nuevo virus (línea 6). Luego, se ajusta el nuevo virus creado para que cumpla con los requerimientos del problema (línea 7), como utilizar una numeración cardinal de las bahías y no utilizar más bahías de las existentes. Realizado el paso anterior, se realiza una actualización de la población (línea 10).

7.2.1.5. Respuesta inmune

```
1  Algoritmo BCV::respuestaInmune:
2      Para i en len(poblacion):
3          pr  = (tamañoGenes - i + 1) / tamañoGenes
4          v = virus'[i]
5          Para j en len(tamañoGenes):
6              Si obtenerDoubleAleatorio > pr:
7                  Mientras (x1 = obtenerDoubleAleatorio()) != i
8                  Mientras (x2 = obtenerDoubleAleatorio()) != i
9                  v[j] = virus'[x1][j] - (virus'[x2][j] - virus'[i][j]) * obtenerAleatorio()
10             FinSi
11         FinPara
12     FinPara
13     Actualizar(poblacion, virus')
14 FinAlgoritmo
```

Ilustración 18: Respuesta inmune en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia.

La respuesta inmune en el algoritmo de búsqueda de colonia de virus, mostrada en la Ilustración 18, muestra como para toda la población (línea 2) se calcula una proporción (línea 3) y se toma un virus de la nueva población (línea 4), donde, para cada espacio en el virus (línea 5), si se obtiene un número aleatorio mayor a la proporción calculada anteriormente (línea 6), se obtienen dos índices (líneas 7 y 8). Estos índices sirven para actualizar un espacio del virus escogido en anteriormente (línea 8). Finalizadas las

instrucciones iterativas, se realiza una actualización de la población (línea 13) y finaliza el método.

7.2.1.6. Selección de sobrevivientes

```
1  Algoritmo BCV::Actualizar:
2      nuevaPoblación = []
3      Para i en len(poblacion):
4          Si virus[i].obtenerFuncionObjetivo() < virus'[i].obtenerFuncionObjetivo():
5              nuevaPoblación[i] = poblacion[i]
6          Sino:
7              nuevaPoblación[i] = virus'[i]
8          FinSi
9      FinPara
10     poblacion = nuevaPoblación
11 FinAlgoritmo
```

Ilustración 19: Selección de virus sobrevivientes en el algoritmo de búsqueda de colonia de virus. Fuente: Elaboración propia.

La Ilustración 19nos muestra la selección de virus sobrevivientes en el algoritmo de búsqueda de colonia de virus, para lo cual se verifica en toda la población (línea 3) si un virus tiene mejor valor de función objetivo que su contraparte en la nueva población (línea 4). En caso esto sea cierto, el virus de la población anterior va a ingresar a la nueva población resultante de la selección de sobrevivientes (línea 5), en caso contrario, el virus de la antigua población calculada antes de la selección de sobrevivientes va a ingresar a esta nueva población (línea 7). Finalmente, la población se actualiza con la población resultante de la selección de nuevos virus (línea 10).

7.2.2. Validaciones realizadas

Este acápite ha sido verificado por el versado en algoritmos, quien ha expresado su beneplácito en cuanto a al contenido de este capítulo. En el anexo Anexo I - Acta de conformidad del algoritmo búsqueda de colonia de virus se puede observar la evidencia.

7.3. Discusión

En este capítulo se alcanzó el resultado esperado R2.2, relacionado al diseño de un algoritmo de búsqueda de colonia de virus adaptado al problema de planificación presentado en esta investigación.

Este resultado ayuda a comprender el funcionamiento general del algoritmo de búsqueda de colonia de virus, así como el diseño base del algoritmo para su posterior implementación. Una consideración importante es la verificación del uso de las estructuras desarrolladas en el capítulo anterior, las cuales sirvieron para trabajar con

facilidad y limpieza este capítulo. Gracias a lo presentado anteriormente, se ha presentado la base para la implementación del algoritmo de búsqueda de colonia de virus. Dentro de las investigaciones recopiladas en el Estado del Arte, se ha logrado identificar que la mayoría de estos detallan la adaptación de los algoritmos que utilizan, además de las modificaciones realizadas para optimizar las soluciones obtenidas. En el caso de esta investigación, se realizará una calibración de variables en un capítulo posterior.

De igual manera, todo el trabajo realizado en este capítulo puede ser adaptado y utilizado en problemas similares, como por ejemplo la actualización de la población de virus, realizando una actualización que tome los mejores virus de cada población obtenida para formar una nueva población.

Capítulo 8. Codificación del algoritmo genético

8.1. Introducción

En el presente capítulo se desarrolla el R3.3, el cual está relacionado al desarrollo del código fuente del algoritmo genético. Esta codificación se realizó tomando en cuenta los diseños del algoritmo previamente realizados, como son el y las estructuras de datos pseudocódigo del algoritmo genético.

De la misma manera que en el Capítulo 6, se mostrarán las funciones principales del mismo, considerando esta vez distintos aspectos técnicos y anotaciones sobre situaciones observadas en el proceso de desarrollo.

En el anexo Anexo J – Código del algoritmo genético se presenta el código del algoritmo desarrollado en el lenguaje C++.

8.2. Resultados alcanzados

En el siguiente apartado se muestra en detalle el trabajo realizado para obtener los resultados esperados R3.3: “Codificación del algoritmo genético adaptado al problema de planificación”.

8.2.1. Codificación del algoritmo genético adaptado al problema de planificación

De la misma manera que en el capítulo 6, la presentación del código del algoritmo genético se distribuye a partir de las funciones principales que lo componen, de manera que se pueda explicar lo realizado en cada uno de ellos.

8.2.1.1. Función principal del algoritmo genético

```
while (i < generaciones) {
    try {
        generation.generateNewGeneration();
        generation.calculateAllFitness();
        if (generation.getBestFitness() < menor) {
            timetable = generation.getBestChromosome();
            menor = generation.getBestFitness();
        }
        i++;
    } catch (string exception) {
        cout << "Error: " << exception << endl;
    }
}
```

Ilustración 20: Código de la función principal del algoritmo genético. Fuente: Elaboración propia.

La Ilustración 20 muestra la función principal del algoritmo genético, donde se itera hasta

alcanzar el número de generaciones que es definida dentro de los parámetros iniciales del algoritmo, generando nuevas poblaciones y obteniendo distintas soluciones a lo largo de estas iteraciones. Con esto se busca obtener las soluciones con un menor fitness, siendo que un menor fitness implica una mejor solución a partir de lo definido en la función objetivo. Cabe resaltar que solo se utiliza el número de generaciones como condición de parada del algoritmo, puesto que la aleatoriedad de este permite obtener soluciones dispersas en el espacio de búsqueda, lo que evita óptimos locales.

8.2.1.2. Construcción de la población inicial

```
/*
    Inicializa la población del algoritmo genético
*/
void Generation::initPopulation(Chromosome chromosome, int populationSize) {
    this->generationNumber = 0;
    for (int i = 0; i < populationSize; i++) {
        Chromosome newChromosome = Chromosome(chromosome);
        shuffle(newChromosome.getGenes().begin(), newChromosome.getGenes().end(), RandomUtils::dre);
        this->chromosomesList.push_back(newChromosome);
    }
}
```

Ilustración 21: Código de la construcción de la población inicial del algoritmo genético. Fuente: Elaboración propia.

En la Ilustración 21 se muestra el método de construcción de la población inicial en dónde, a partir de un cromosoma inicial, se realiza una mezcla de los genes para crear nuevos individuos y agregarlos a la población inicial. También se inicializa el contador de generaciones en cero.

8.2.1.3. Método de recombinación

```
int i = 0;
while (!i) {
    i = RandomUtils::get_next_int(parent1.getGenes().size());
}

vector<int> firstC1(parent1.getGenes().begin(), parent1.getGenes().begin() + i);
vector<int> firstC2(parent2.getGenes().begin(), parent2.getGenes().begin() + i);
vector<int> lastC1(parent1.getGenes().begin() + i, parent1.getGenes().end());
vector<int> lastC2(parent2.getGenes().begin() + i, parent2.getGenes().end());
firstC1.insert(firstC1.end(), lastC2.begin(), lastC2.end());
firstC2.insert(firstC2.end(), lastC1.begin(), lastC1.end());
Chromosome child1(parent1), child2(parent2);
child1.setGenes(firstC1);
child2.setGenes(firstC2);

children.push_back(child1);
children.push_back(child2);
return children;
```

Ilustración 22: Método de recombinación en el algoritmo genético. Fuente: Elaboración propia.

La Ilustración 22 muestra la recombinación de cromosomas, en donde se obtienen dos cromosomas hijos a partir d la partición de 2 cromosomas padres en un punto aleatorio.

8.2.1.4. Método de mutación

```
vector<int> Generation::inversionMutation(vector<int> genes) {
    int index1 = RandomUtils::get_next_int(genes.size()),
        index2 = RandomUtils::get_next_int(genes.size() - index1) + index1;
    vector<int> chromosomeMid(genes.begin() + index1, genes.begin() + index2);
    reverse(chromosomeMid.begin(), chromosomeMid.end());
    vector<int> result(genes.begin(), genes.begin() + index1);
    result.insert(result.end(), chromosomeMid.begin(), chromosomeMid.end());
    result.insert(result.end(), genes.begin() + index2, genes.end());
    return result;
}
```

Ilustración 23: Método de mutación del algoritmo genético. Fuente: Elaboración propia

La Ilustración 23 muestra la función de mutación utilizada en el algoritmo, la cual es una mutación de inversión en donde se obtiene una parte de los genes, los cuales se invierten y se vuelven a ingresar en la misma posición para crear un nuevo individuo.

8.2.1.5. Adaptación de genes

```
for (uint i = 0; i < c.getGenes().size(); i++) {
    prob = RandomUtils::get_next_double();
    if (prob >= mutationProb) {
        int size = newChromosome.getGenes().size();
        int change = RandomUtils::get_next_int(size);
        newChromosome.getGenes()[change] = newChromosome.getGenes()[RandomUtils::get_next_int(size)];
    }
}
```

Ilustración 24: Método de adaptación de genes del algoritmo genético. Fuente: Elaboración propia.

En la Ilustración 24 se muestra una adaptación de genes dónde se copia asignación de una bahía a otro pedido de manera que se le entrega mayor aleatorización al algoritmo. Este método fue agregado puesto que los resultados obtenidos en un inicio no se acercaban a las planificaciones requeridas.

8.2.1.6. Validación de la solución

```
copy_if(timetable[i].begin(), timetable[i].end(), back_inserter(obligatorio), [&](Pedido &ped) { return ped.turno != -1; });
for (uint j = 0; j < obligatorio.size(); j++) {
    // se verifican todos los espacios correspondientes a un turno
    int maxDate = obligatorio[j].fechaMaximaDespacho != -1 ? obligatorio[j].fechaMaximaDespacho * dailyHours : deadline;
    for (int t = obligatorio[j].turno; t * timestamp < maxDate; t += 4) {
        for (int k = timestamp * t; k < timestamp * (t + 1) - obligatorio[j].tiempoCarga; k++) {
            bool lleno = false;
            for (int m = 0; m < obligatorio[j].tiempoCarga; m++)
                lleno = lleno || schedule[k + m] != nullptr;
            if (!lleno) {
                for (int m = 0; m < obligatorio[j].tiempoCarga; m++)
                    schedule[k + m] = &obligatorio[j];
                obligatorio[j].horaCarga = k;
                goto JUMP;
            }
        }
    }
JUMP:;
    if (obligatorio[j].horaCarga == -1)
        return false;
}

// filtro para pedidos normales
copy_if(timetable[i].begin(), timetable[i].end(), back_inserter(no_obligatorio), [&](Pedido &ped) { return ped.turno == -1; });
for (uint j = 0; j < no_obligatorio.size(); j++) {
    // se verifican todos los espacios correspondientes
    int maxDate = no_obligatorio[j].fechaMaximaDespacho != -1 ? no_obligatorio[j].fechaMaximaDespacho * dailyHours : deadline;
    for (int k = 0; k < maxDate - no_obligatorio[j].tiempoCarga; k++) {
        bool lleno = false;
        for (int m = 0; m < no_obligatorio[j].tiempoCarga; m++) {
            lleno = lleno || schedule[k + m] != nullptr;
        }
        if (!lleno) {
            for (int m = 0; m < no_obligatorio[j].tiempoCarga; m++)
                schedule[k + m] = &no_obligatorio[j];
            no_obligatorio[j].horaCarga = k;
            break;
        }
    }

    if (no_obligatorio[j].horaCarga == -1) {
        return false;
    }
}
```

Ilustración 25: Método de validación de la solución en el algoritmo genético. Fuente: Elaboración propia.

La Ilustración 25 muestra la implementación de la validación planteada, a partir de la cual se puede verificar que una solución propuesta cumple con los requerimientos para poder

ser usada.

8.2.1.7. Cálculo del fitness

```
fitnessValue = 0;
double check = 0, mean = getMeanHours(), mwh = getMeanWeightHours();
for (int i = 0; i < baysNumber; i++) {
    double ac = 0;
    if (!timetable[i].size()) {
        fitnessValue += ac + abs(mean - ac) + abs(mwh - 0);
        continue;
    }
    double totalChargingHours = accumulate(timetable[i].begin(), timetable[i].end(), 0, [&](int &i, Pedido &p) { return i + p.tiempoCarga; });
    Pedido p = timetable[i][timetable[i].size() - 1];
    ac = p.horaCarga + p.tiempoCarga;
    fitnessValue += ac + abs(mean - ac) + abs(mwh - totalChargingHours);
}
```

Ilustración 26: Método de cálculo del fitness en el algoritmo genético. Fuente: Elaboración propia.

En la Ilustración 26 se identifica la implementación de la función objetivo en el lenguaje de programación C++, la cual permite comparar distintas soluciones y verificar cuáles son mejores para el problema planteado.

8.2.2. Validaciones realizadas

Se realizaron distintas pruebas unitarias a los distintos métodos implementados en el presente capítulo, para así verificar el correcto funcionamiento de cada uno de estos. En el anexo Anexo K - Pruebas del algoritmo genético, se presenta un informe de las pruebas realizadas.

8.3. Discusión

En el presente acápite se realizaron los trabajos relacionados al resultado esperado 3.3, en el cual se busca realizar la codificación del algoritmo genético adaptado al problema de planificación presentado en esta investigación. Este desarrollo se ha basado en los logros obtenidos de los resultados esperados anteriores cómo son el pseudocódigo y las estructuras de datos del algoritmo genético.

Dentro de las investigaciones recopiladas en el estado del arte, se han logrado identificar que la mayoría de estas no presentan una implementación en algún lenguaje de programación. En el caso de esta investigación, no solo se presenta la implementación en el lenguaje C++, sino que también se mostrará la realización de una calibración de variables en el capítulo posterior.

Finalmente, todo el trabajo realizado en este capítulo puede ser adaptado y utilizado en problemas similares, teniendo en consideración el uso de estructuras y métodos que tienen por objetivo solucionar el problema planteado en esta investigación.

Capítulo 9. Codificación del algoritmo de búsqueda de colonia de virus

9.1. Introducción

En el presente epígrafe se desarrolla el R3.1, el cual está vinculado con el desarrollo del código fuente del algoritmo de búsqueda de colonia virus. Este código se realizó tomando en cuenta los diseños del algoritmo previamente bosquejados, como son las estructuras de datos y el pseudocódigo del algoritmo de búsqueda de colonia de virus.

De la misma manera que en el Capítulo 7, se presentarán los métodos principales del algoritmo, considerando distintos aspectos técnicos y algunas anotaciones sobre situaciones presentadas en el desarrollo.

En el anexo L se presenta el código del algoritmo desarrollado en el lenguaje de programación C++.

9.2. Resultados alcanzados

En el siguiente acápite se muestra en detalle el trabajo realizado para obtener el resultado esperado R3.1: “Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación”, implementando así el algoritmo de búsqueda de colonia virus aplicado al problema de planificación presentado en esta investigación.

9.2.1. Algoritmo de búsqueda de colonia de virus aplicado al problema de planificación

A continuación, se presenta la implementación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación que se busca resolver en esta investigación en el lenguaje de programación C++, con el apoyo de una librería de terceros para el trabajo con arreglos, llamada NumCpp (Pilger, s. f.). Cabe resaltar que se utiliza el mismo método de validación y función objetivo que en la implementación del algoritmo genético.

9.2.1.1. Función principal del algoritmo de búsqueda de colonia virus

```
while (i < generaciones) {
    try {
        difusion(i);
        infeccionCelular(i);
        respuestaInmune();
        if (bestVirusG.getFitnessValue() < menor) {
            bestVirus = bestVirusG;
            menor = bestVirus.getFitnessValue();
        }
        i++;
    } catch (string exception) {
        cout << "Error: " << exception << endl;
    }
}
```

Ilustración 27: Función principal del algoritmo VCS. Fuente: Elaboración propia.

La Ilustración 27 presenta la implementación de la función base del algoritmo de búsqueda de colonia virus el cual utiliza una lista de pedidos que se pueden entregar, dependiendo de la capacidad de las bahías, y retorna una planificación consecuente.

9.2.1.2. Construcción de la población inicial

```
for (int i = 0; i < tamGeneracion; i++) {
    Virus nuevo = inicial;
    nc::random::shuffle(nuevo.getGenes());
    Vpop.push_back(nuevo);
}
```

Ilustración 28: Método de construcción de la población inicial en el algoritmo VCS. Fuente: Elaboración propia.

En la Ilustración 28 se muestra la inicialización de la población, que se realiza tomando un virus inicial, el cual es copiado hacia un nuevo virus, para luego mezclar la información que contiene y, finalmente, agregar este a la nueva población.

9.2.1.3. Difusión de virus

```
for (Virus v : Vpop) {
    auto sigma = (nc::log1p(generacion + 1) / maxGeneration) *
        (v.getGenes() - bestVirus.getGenes()).astype<double>();
    vector<double> g;
    for (uint i = 0; i < v.getGenes().size(); i++)
        g.push_back(nc::random::normal((double)bestVirus.getGenes()[i], abs(sigma[i] + DBL_MIN)));
    nc::NdArray<double> gauss(g);
    auto nuevosGenes = gauss + (nc::random::uniform(0.0, 1.0) *
        bestVirus.getGenes().astype<double>() -
        nc::random::uniform(0.0, 1.0) * v.getGenes().astype<double>());
    Virus nuevoVirus = bestVirus;
    nuevoVirus.setGenes(nuevosGenes.astype<int>());
    nuevoVirus.clipValues();
    Vpopp.push_back(nuevoVirus);
}
actualizarPoblacion();
```

Ilustración 29: Método de difusión de virus del algoritmo VCS. Fuente: Elaboración propia.

Este proceso, mostrado en la Ilustración 29, tiene por finalidad crear una nueva población de virus para así obtener distintas soluciones que se acerquen a un resultado óptimo.

9.2.1.4. Infección de células huésped

```
sort(Vpop.begin(), Vpop.end());
vector<int> popFitness;
for (Virus v : vector<Virus>(Vpop.begin(), Vpop.begin() + nBest))
    popFitness.push_back(v.getFitnessValue());
double factor = nBest * nc::log1p(nBest + 1) - nc::log1p(nc::arange(1, nBest + 1).prod()[0]);
double peso = nc::log1p(nBest + 1) / factor;
peso /= nBest;
double mean = peso * nc::sum(nc::NdArray<int> { popFitness })[0];
double gSigma = sigma * (1 - (generacion + 1) / maxGeneration);
for (uint i = 0; i < Vpop.size(); i++) {
    auto newVirus = mean + gSigma * nc::random::normal({ 1, (uint)Vpop[i].getGenes().size() }, 0.0, 1.0);
    Virus newV = Vpop[i];
    newV.setGenes(newVirus.astype<int>());
    newV.clipValues();
    Vpopp.push_back(newV);
}
actualizarPoblacion();
```

Ilustración 30: Método de infección celular en el algoritmo VCS. Fuente: Elaboración propia.

De manera similar a la difusión de virus, como es mostrado en la Ilustración 30, en este proceso se crea una nueva población, la cual es utilizada para realizar una búsqueda de nuevas soluciones que se acerquen al resultado óptimo. La imagen muestra como se realizan procedimientos adicionales para poder trabajar fácilmente la población de virus, como es el ordenamiento de la población; así como otros cálculos requeridos.

9.2.1.5. Respuesta inmune

```
uint tamGenes = Vpop[0].getGenes().size();
for (uint i = 0; i < Vpop.size(); i++) {
    double pr = (tamGenes - i + 1) / tamGenes;
    Virus newV = Vpop[i];
    //          newV.setFitnessValue(DBL_MAX);
    for (uint j = 0; j < tamGenes; j++)
        if (RandomUtils::get_next_double() < pr) {
            uint index1, index2;
            while ((index1 = RandomUtils::get_next_int(Vpop.size())) == i) { }
            set<uint> indexSet { (uint)index1, i };
            while (indexSet.count((index2 = RandomUtils::get_next_int(Vpop.size()))) != 0) { }
            auto genes = newV.getGenes();
            genes[j] = Vpop[index1].getGenes()[j] -
                (int)(Vpop[index2].getGenes()[j] - Vpop[i].getGenes()[j]) *
                RandomUtils::get_next_double();
        }
    newV.clipValues();
    Vpopp.push_back(newV);
}
actualizarPoblacion();
```

Ilustración 31: Método de respuesta inmune en el algoritmo VCS. Fuente: Elaboración propia.

El método presentado en la Ilustración 31 tiene por objetivo modificar una población de virus ya existente, de manera que está aleatorización produzca nuevos resultados que se acerquen a la solución esperada.

9.2.1.6. Selección de sobrevivientes

```
vector<Virus> newVpop;
for (unsigned int i = 0; i < Vpop.size(); i++) {
    if (Vpop[i] < Vpopp[i])
        newVpop.push_back(Vpop[i]);
    else
        newVpop.push_back(Vpopp[i]);
}
Vpopp.clear();
Vpop = newVpop;
```

Ilustración 32: Método de selección de virus sobrevivientes en el algoritmo VCS. Fuente: Elaboración propia.

Se muestra en la Ilustración 32 la selección de virus sobrevivientes en él algoritmo de búsqueda de coronavirus, para lo cual se toman dos poblaciones de virus que compiten para sobrevivir, para lo cual, entre dos virus, sobrevive el que tenga un menor valor de la función objetivo, lo que significa una mejor solución.

9.2.2. Validaciones realizadas

Se realizaron distintas pruebas unitarias a los distintos métodos implementados en el presente capítulo coma para así verificar el correcto funcionamiento de cada 1 de estos

puntos en el anexo M se presenta un informe de las pruebas realizadas. Discusión

En el presente acápite se realizaron los trabajos relacionados al resultado esperado 3.1, en el cual se busca realizar la codificación del algoritmo de búsqueda de colonia de virus aplicado a la planificación de horarios de despacho en una planta con múltiples bahías. Este desarrollo se ha basado en los logros obtenidos de los resultados esperados anteriores como son el pseudocódigo y las estructuras de datos del presente algoritmo.

Dentro de las investigaciones recopiladas en el estado del arte, se ha logrado identificar que la mayoría de estas no presentan una implementación algún lenguaje de programación. En el caso de investigación, no solamente se presenta la implementación en el lenguaje de C++, sino que también se mostrará la realización de una calibración de variables en el capítulo en un capítulo posterior. Finalmente, todo el trabajo realizado en este capítulo puede ser adaptado utilizado en problemas similares teniendo en consideración el uso de estructuras y métodos que tienen por objetivo solucionar el problema planteado en esta investigación, como son la función objetivo y las validaciones realizadas con los virus.

Capítulo 10. Calibración de variables

10.1. Introducción

En este capítulo se desarrolla el resultado 4.1, relacionado a la calibración de variables para los algoritmos desarrollados para esta investigación, con el fin de obtener los mejores valores de los parámetros de los algoritmos, de manera que se obtengan mejores soluciones.

10.2. Resultados alcanzados

En el presente acápite se presentan los procedimientos realizados para realizar el R4.1: "Calibración de variables"

10.2.1. Calibración de variables

Para el algoritmo búsqueda de colonia de virus se calibraron las variables del número de generaciones, el tamaño de la población, lambda y sigma; mientras que para e algoritmo genético se realizó la calibración de las variables del número de generaciones, el tamaño de la población y la taza de mutación.

Se realizó la calibración utilizando cuatro conjuntos de datos con ventanas de planificación de un día, tres días, una semana y dos semanas.

Para cada conjunto de valores de las variables presentadas se realizó una muestra de 10 observaciones, tomando como el valor de fitness como el dato representativo y el fitness promedio de la población como se evidencia a continuación.

Para el algoritmo genético, se puede observar que las mejores soluciones tienen entre 50 y 100 generaciones, una media de 100 individuos por generación y una taza de mutación de 0.7 en su mayoría.

Tabla 18: Resultados de la calibración de variables del algoritmo genético. Fuente: Elaboración propia

Fitness	Ventana de planificación	Número de generaciones	Tamaño de la población	Taza de mutación
148.571	1	50	100	0.7
375.857	3	100	200	0.7
1079.29	7	50	100	0.6
2078.57	15	100	150	0.7

Por otra parte, el algoritmo búsqueda de colonia de virus tiene un número de generaciones con media de 100, una media de 100 individuos por generación, una lambda de 0.4 en la mayoría de los casos y una sigma entre 1.5 y 2.

Tabla 19: Resultados de la calibración de variables del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia

Fitness	Ventana de planificación	Número de generaciones	Tamaño de la población	Lambda	Sigma
149.429	1	100	200	0.5	2
409.429	3	50	150	0.4	1.5
1111.29	7	100	150	0.4	1.5
2089.43	15	150	100	0.4	2

Para la presente investigación, se utilizarán los siguientes valores a partir de la aproximación con las mejores soluciones obtenidas: para el algoritmo genético se utilizará 100 generaciones, 100 individuos por generación y una taza de mutación de 0.7; mientras que para el algoritmo búsqueda de colonia de virus, se utilizará el valor de 100 generaciones, 100 individuos por generación, una lambda de 0.4 y una sigma de 2.

10.2.2. Validaciones realizadas

El presente epígrafe ha sido verificado por el versado en algoritmos, quien ha expresado su aprobación respecto a lo presentado. Se adjunta evidencia en el anexo N.

10.3. Discusión

En este epígrafe se ha realizado el resultado esperado R4.1, relacionado a la calibración de variables para los algoritmos presentados en esta investigación.

El alcance de este resultado permite obtener variables con las que se obtengan mejores soluciones para ambos algoritmos, de manera que se puedan comparar las mejores soluciones que puedan ser utilizadas en el ámbito industrial.

Respecto a las investigaciones de mayor antigüedad y otros casos de estudio seleccionados en el Estado del Arte, las variables utilizadas en el algoritmo genético varían según el problema. Por esta razón, se realizó la calibración de las principales variables antes mencionadas.

Capítulo 11. Experimentación numérica

11.1. Introducción

En este acápite se desarrolla el resultado 4.2, el cual refiere a la experimentación numérica, a partir de la cual se podrá verificar la hipótesis que se puede obtener a partir de los datos recolectados, de forma que se puede verificar que un algoritmo devuelve mejores soluciones que otro; tomando como métrica los valores de la función objetivo de las soluciones obtenidas.

11.2. Resultados alcanzados

En este capítulo se muestra el detalle del procedimiento realizado para lograr el resultado esperado R4.2: “Informe de experimentación numérica”.

11.2.1. Informe de experimentación numérica

El presente capítulo se distribuye en la recolección de datos y en las pruebas de normalidad, igualdad de varianzas y la prueba T de Welch, las cuales conceden analizar las observaciones obtenidas de las soluciones de los algoritmos desarrollados. Cabe resaltar que, para todas las pruebas, se utiliza un nivel de confianza de 0.05.

11.2.1.1. Recolección de datos

Para la recolección de datos, se estableció una ventana de planificación de un mes y siete bahías de despacho. Con estos datos se realizaron diez planificaciones para cada uno de los treinta distintos conjuntos de pedidos, de manera que se puedan comparar los mejores resultados.

Se pueden observar los resultados en las imágenes a continuación.

Ilustración 33: Promedio de fitness de los resultados obtenidos por cada algoritmo. Fuente: Elaboración propia

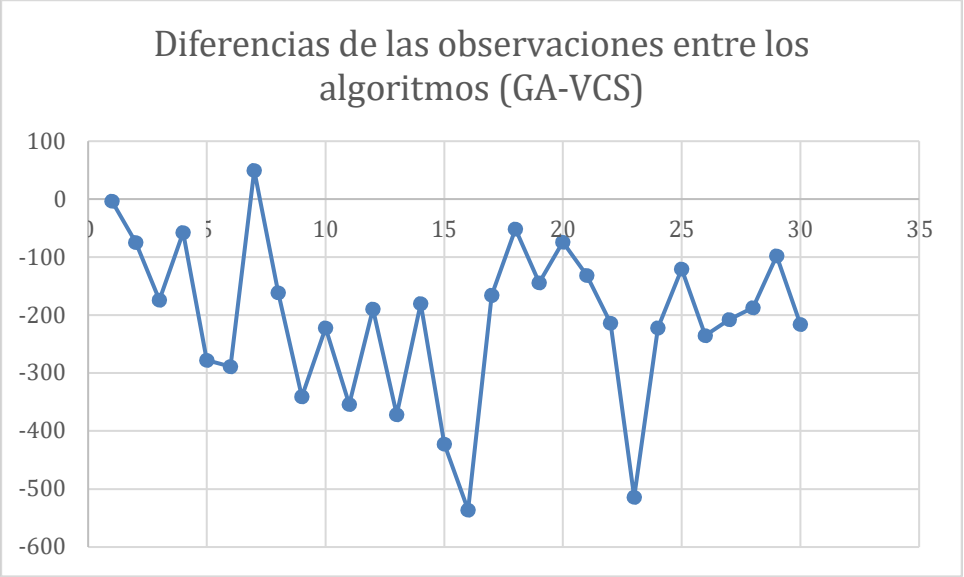

Ilustración 34: Diferencia entre las observaciones obtenidas de ambos algoritmos. Fuente: Elaboración propia

A partir de los datos obtenidos, se va a realizar una comparación de medias, teniendo en cuenta que, empíricamente, las medias de los resultados del algoritmo genético son menores que las medias de los resultados del algoritmo búsqueda de colonia de virus.

11.2.1.2. Prueba de normalidad

Se realizó la prueba de normalidad de los datos, de manera que se verifique que estas siguen una distribución normal. Esta prueba nos permitirá saber qué tipo de prueba debemos realizar para la comparación de medias.

Para la prueba mencionada, se realiza una prueba para las observaciones de cada algoritmo, pero las hipótesis se mantienen sin cambios para ambos casos, puesto que se mantiene el mismo objetivo.

H_0: La muestra sigue una distribución normal

H_1: La muestra no sigue una distribución normal

Para el algoritmo genético se obtuvo el siguiente resultado:

```
> normalityTest(~Fitness, test="shapiro.test", data=ga)

        Shapiro-Wilk normality test

data:  Fitness
W = 0.9554, p-value = 0.2353
```

Ilustración 35: Prueba de normalidad de las observaciones del algoritmo genético. Fuente: Elaboración propia

Para el algoritmo búsqueda de colonia de virus se obtuvo el siguiente resultado:

```
> normalityTest(~Fitness, test="shapiro.test", data=vcs)

        Shapiro-Wilk normality test

data:  Fitness
W = 0.94275, p-value = 0.1079
```

Ilustración 36: Prueba de normalidad de las observaciones del algoritmo búsqueda de colonia de virus. Fuente: Elaboración propia

Para ambos conjuntos de datos, se puede observar que los p-valores son mayores al nivel de confianza, de manera que no se puede comprobar la falsedad de la hipótesis nula; por lo que se puede afirmar que las observaciones de ambos algoritmos siguen una distribución normal.

11.2.1.3. Prueba de igualdad de varianzas

Para verificar que la varianza entre las muestras es homogénea, se realiza una prueba de homogeneidad de varianzas. Para esto se formulan las siguientes hipótesis:

H_0: La muestra tiene varianzas homogéneas

H_1: La muestra no tiene varianzas homogéneas

A partir de estas hipótesis, se realiza la prueba de Bartlett para ambos conjuntos de datos:

```
> bartlett.test(Fitness ~ Algoritmo, data=resultados)

        Bartlett test of homogeneity of variances

data:  Fitness by Algoritmo
Bartlett's K-squared = 6.3553, df = 1, p-value = 0.0117
```

Ilustración 37: Prueba de homogeneidad de varianzas para las observaciones de ambos algoritmos. Fuente: Elaboración propia

Se puede observar que el p-valor es menor al nivel de confianza (establecido en 0.05), por lo que se niega la hipótesis nula, de manera que se verifica que las varianzas de ambos conjuntos de datos no son homogéneas. A partir de los datos recolectados, se procede a utilizar una prueba t de Welch, de manera que se verifique la hipótesis de que las medias de las soluciones obtenidas del algoritmo genético son menores a las medias de las soluciones obtenidas del algoritmo búsqueda de colonia de virus.

11.2.1.4. Prueba T de Welch

Para realizar la prueba T de Welch, se han planteado las siguientes hipótesis:

H_0: Las medias de las soluciones obtenidas por el algoritmo genético es menor a las medias de las soluciones obtenidas por el algoritmo búsqueda de colonia de virus

H_1: Las medias de las soluciones obtenidas por el algoritmo genético no es menor a las medias de las soluciones obtenidas por el algoritmo búsqueda de colonia de virus

Estas hipótesis se pueden traducir de la siguiente manera:

H_0: La diferencia entre las medias de las soluciones obtenidas por el algoritmo genético y las medias de las soluciones obtenidas por el algoritmo búsqueda de colonia de virus es menor que cero.

H_1: La diferencia entre las medias de las soluciones obtenidas por el algoritmo genético y las medias de las soluciones obtenidas por el algoritmo búsqueda de colonia de virus no es menor que cero.

A partir de las hipótesis planteadas, se procede a realizar la prueba:

```
> t.test(Fitness~Algoritmo, alternative='less', conf.level=.95,
+        var.equal=FALSE, data=resultados)

        Welch Two Sample t-test

data:  Fitness by Algoritmo
t = -8.1687, df = 48.34, p-value = 5.824e-11
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
      -Inf -163.8467
sample estimates:
 mean in group ga mean in group vcs
         2934.861          3141.034
```

Ilustración 38: Prueba de diferencia de medias para las observaciones de ambos algoritmos. Fuente: Elaboración propia

De manera que, a partir de un p-valor menor que el valor de confianza, no se puede negar la hipótesis nula, de manera que se puede afirmar que las medias de las soluciones obtenidas con el algoritmo genético son menores a las medias de las soluciones obtenidas con el algoritmo búsqueda de colonia de virus.

11.2.1.5. Validaciones realizadas

Este acápite ha sido revisado por el especialista en algoritmia, quien ha expresado su aprobación respecto a la información presentada en el presente capítulo; aprobación plasmada en la evidencia presentada en el anexo O.

11.2.2. Discusión

El alcance del resultado esperado R4.2. ha permitido identificar que el algoritmo genético presenta mejores soluciones respecto al algoritmo búsqueda de colonia de virus. Este resultado resuelve que el algoritmo genético sigue siendo una solución fiable en cuanto al tipo de problemas como el presentado en esta investigación se refiere.

Se debe analizar la posible razón de este resultado, como puede ser la estrategia utilizada para abordar el problema; puesto que esta puede afectar a la obtención de resultados, lo cual podría diferir de los resultados obtenidos en esta investigación.

Capítulo 12. Referencias

Ahmadizar, F., & Farahani, M. H. (2012). A novel hybrid genetic algorithm for the open shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *62*(5-8), 775-787. Scopus. https://doi.org/10.1007/s00170-011-3825-1

Armstrong, R. A., Eperjesi, F., & Gilmartin, B. (2002). The application of analysis of variance (ANOVA) to different experimental designs in optometry. *Ophthalmic & Physiological Optics: The Journal of the British College of Ophthalmic Opticians (Optometrists)*, *22*(3), 248-256. https://doi.org/10.1046/j.1475-1313.2002.00020.x

Benjamín, E., & Fincowsky, F. (2004). *Enrique, Franklin-Organizacion de empresas* (3ra ed.). https://naghelsy.files.wordpress.com/2016/01/enrique-franklin-organizacion-de-empresas.pdf

Berlanga-Silvente, V., & Rubio-Hurtado, M.-J. (2012). Clasificación de pruebas no paramétricas. Cómo aplicarlas en SPSS. *REIRE Revista d'Innovació i Recerca en Educació*, *5*(2), 101-113. https://doi.org/10.1344/reire2012.5.2528

Blum, C., & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Comput. Surv.*, *35*(3), 268-308. https://doi.org/10.1145/937503.937505

Cook, S. (s. f.). *THE P VERSUS NP PROBLEM*. 12.

Cooper, T. B., Kingston, J. H., Cooper, T. B., & Kingston, J. H. (1996). The

complexity of timetable construction problems. *In Burke and Ross [BR96*, 283-295.

de Man, J. C., & Strandhagen, J. O. (2018). Spreadsheet Application still dominates Enterprise Resource Planning and Advanced Planning Systems. *IFAC-PapersOnLine*, *51*(11), 1224-1229. https://doi.org/10.1016/j.ifacol.2018.08.423

DEFINICIÓN DE LOGÍSTICA - Promonegocios.net. (s. f.). Recuperado 26 de abril de 2022, de https://www.promonegocios.net/distribucion/definicion-logistica.html

Duque, D. F., Saint-Priest Velásquez, Y., Segovia, P., & Loaiza, D. F. (2017). *Algoritmos y programación en pseudocódigo*.

El Comercio. (2021, febrero 1). *Lima inicia cuarentena ante el avance de la segunda ola de la pandemia*. El Comercio Perú; NOTICIAS EL COMERCIO PERÚ. https://elcomercio.pe/videos/pais/lima-inicia-cuarentena-ante-el-avance-de-la-segunda-ola-de-la-pandemia-cuarnetena-2021-nnav-agafp-noticia/

Equipamiento de un muelle de carga—Solumat. (s. f.). [Figura] Recuperado 22 de agosto de 2022, de https://www.solumat.com.co/blog/equipamiento-muelle-carga/

Fathollahi-Fard, A. M., & Hajiaghaei-Keshteli, M. (2018). Integrated capacitated transportation and production scheduling problem in a fuzzy environment. *International Journal of Industrial Engineering and Production Research*, *29*(2), 197-211. Scopus. https://doi.org/10.22068/ijiepr.29.2.197

Fister Jr., I., Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). A Brief Review

of Nature-Inspired Algorithms for Optimization. *arXiv:1307.4186 [cs]*. http://arxiv.org/abs/1307.4186

Garcia-Sabater, J. P. (2020). *Los almacenes como nodos de la red logística* [Nota Técnica].

Gass, S. I., & Fu, M. C. (Eds.). (2013). Set-covering Problem. En *Encyclopedia of Operations Research and Management Science* (pp. 1393-1393). Springer US. https://doi.org/10.1007/978-1-4419-1153-7_200755

Git. (s. f.). Recuperado 8 de junio de 2022, de https://git-scm.com/

Goris, G., & Adolf, S. J. (2015). Usefulness and types of literature review. *Ene*, *9*(2), 0-0. https://doi.org/10.4321/S1988-348X2015000200002

Guturu, P., & Dantu, R. (2008). An Impatient Evolutionary Algorithm With Probabilistic Tabu Search for Unified Solution of Some NP-Hard Problems in Graph and Set Theory via Clique Finding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(3), 645-666. https://doi.org/10.1109/TSMCB.2008.915645

Hsu, H.-P., Tai, H.-H., Wang, C.-N., & Chou, C.-C. (2021). Scheduling of collaborative operations of yard cranes and yard trucks for export containers using hybrid approaches. *Advanced Engineering Informatics*, *48*. Scopus. https://doi.org/10.1016/j.aei.2021.101292

INSHT. (2016). *Muelles de carga y descarga: Seguridad* [Norma Técnica de Prevención].

ISO/IEC 14882. (s. f.). *ISO/IEC 14882:2017*. ISO. Recuperado 9 de junio de 2022, de https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standar

d/06/85/68564.html

Li, M. D., Zhao, H., Weng, X. W., & Han, T. (2016). A novel nature-inspired algorithm for optimization: Virus colony search. *Advances in Engineering Software*, *92*, 65-88. https://doi.org/10.1016/j.advengsoft.2015.11.004

Librarian, H. S. F. (s. f.). *LibGuides: Systematic Reviews: What is a systematic review?* Recuperado 23 de mayo de 2022, de https://libguides.library.curtin.edu.au/c.php?g=863554&p=6191897

Moreno, B., Muñoz, M., Cuellar, J., Domancic, S., Villanueva, J., Moreno, B., Muñoz, M., Cuellar, J., Domancic, S., & Villanueva, J. (2018). Revisiones Sistemáticas: Definición y nociones básicas. *Revista clínica de periodoncia, implantología y rehabilitación oral*, *11*(3), 184-186. https://doi.org/10.4067/S0719-01072018000300184

Petticrew, M., & Roberts, H. (s. f.). *Systematic Reviews in the Social Sciences*. 354.

Roldan, P. N. (s. f.). *Cadena de suministro—Definición, qué es y concepto*. Economipedia. Recuperado 23 de mayo de 2022, de https://economipedia.com/definiciones/cadena-de-suministro.html

Rose, C. D., & Coenen, J. M. G. (2015). Comparing four metaheuristics for solving a constraint satisfaction problem for ship outfitting scheduling. *International Journal of Production Research*, *53*(19), 5782-5796. Scopus. https://doi.org/10.1080/00207543.2014.998786

RStudio. (s. f.). Recuperado 8 de junio de 2022, de https://www.rstudio.com/products/rstudio/

S. Nguyen. (2017). Optimization, dispatching rules and hyper-heuristics: A

comparison in dynamic single machine scheduling. *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, 4796-4801. https://doi.org/10.1109/IECON.2017.8216827

Sartori, C. S., Gandra, V., Çalık, H., & Smet, P. (2021). Production Scheduling with Stock- and Staff-Related Restrictions. En *Lect. Notes Comput. Sci.: Vol. 13004 LNCS* (p. 162). Scopus. https://doi.org/10.1007/978-3-030-87672-2_10

Weise, T. (2009). *Global Optimization Algorithms—Theory and Application*. 820.

Welcome to Apache NetBeans. (s. f.). Recuperado 8 de junio de 2022, de https://netbeans.apache.org/

Pilger, D. (s. f.). *NumCpp*. Recuperado 1 de octubre de 2022, de https://dpilger26.github.io/NumCpp/doxygen/html/index.html

The Qt Company. (s. f.). *Qt Creator*. Recuperado 29 de septiembre de 2022, de https://www.qt.io/product/development-tools

What is Scrum? (s. f.). Scrum.Org. Recuperado 8 de junio de 2022, de https://www.scrum.org/resources/what-is-scrum

Zapata Gómez, A., & Sarache Castro, W. A. (2014). Mejoramiento de la calidad del café soluble utilizando el método Taguchi. *Ingeniare. Revista chilena de ingeniería*, *22*(1), 116-124. https://doi.org/10.4067/S0718-33052014000100011

Capítulo 13. Anexos

13.1. Anexo A – Formulario de extracción de datos

FormularioExtracciónDatos.xlsx

13.2. Anexo B – Plan de proyecto

13.2.1. Justificación

Las cadenas de suministro han enfrentado distintos problemas a lo largo de la historia, en especial en lo acontecido a raíz de la pandemia, donde la distribución de suministros se ha visto muy afectada debido a las distintas restricciones aplicadas por los gobiernos, los cuales reducían o suspendían en su totalidad muchas actividades de dispensación de abastos, lo que generó insatisfacción en los clientes por la demanda no atendida (Ec, 2021).

Ante esta problemática, las empresas se enfrentan a la tarea de optimizar las actividades de distribución, dentro de ellas la programación de horarios de despacho, de manera que se pueda realizar un abastecimiento que se adecue a las distintas restricciones existentes en el mercado y cumpla con las expectativas de los clientes. De esta manera, es esencial investigar si es posible optimizar la programación de horarios de despacho en una planta con múltiples bahías para alcanzar un nivel adecuado de despachos y evitar cuellos de botella de paquetes o camiones en las bahías de distribución.

Por consiguiente, el objetivo de este proyecto es adaptar un algoritmo bioinspirado a la problemática presentada, de manera que se obtenga una solución cercana a la óptima; logrando así una mejora de una de las actividades necesarias para la distribución de productos como es el despacho desde una planta de producción con múltiples bahías.

13.2.2. Viabilidad

En los siguientes acápites se va a tratar la viabilidad del presente proyecto en los aspectos técnicos, temporal, económica y de datos.

13.2.2.1. Viabilidad técnica

Las herramientas que se utilizarán para el desarrollo del proyecto son open source, como los IDE's NetBeans y Rstudio, y los lenguajes de programación C++ y R para la implementación de los algoritmos y el análisis de los datos recogidos a lo largo del proyecto. Estas herramientas cuentan con una gran comunidad de programadores e investigadores, por lo que se encuentra una gran cantidad de documentación y soporte

para resolver dudas que se puedan suscitar.

En la misma línea, se cuenta con el apoyo de una persona especialista en el área de algoritmia, quien cuenta con una vasta experiencia y conocimientos relacionados a los algoritmos bioinspirados, los cuales serán de ayuda en la implementación de los algoritmos que se van a implementar.

Gracias a lo mencionado anteriormente, se espera que no se presenten obstáculos que puedan perjudicar el desarrollo del proyecto.

13.2.2.2. Viabilidad temporal

El tiempo límite del proyecto se estima en 114 días. Se espera efectuar las actividades definidas dentro de los plazos estimados, las cuales permitirán obtener los objetivos específicos definidos, así como los resultados esperados definidos.

13.2.2.3. Viabilidad económica

No se requiere una inversión monetaria para el desarrollo del proyecto gracias al uso de herramientas open source en todas las fases del proyecto en las que son requeridas.

13.2.2.4. Viabilidad de datos

Los datos que se van a utilizar a lo largo del proyecto van a ser proporcionados por el especialista en algoritmia referido en la realización del proyecto. Por lo mencionado anteriormente se espera que no se encuentren obstáculos en cuanto a los datos requeridos.

13.2.3. Alcance

El proyecto presentado pertenece al área de ciencias de la computación, específicamente en relación con la algoritmia, y tiene como objetivo desarrollar un algoritmo bioinspirado llamado Búsqueda de Colonia de Virus, el cual será utilizado para solucionar el problema de planificación de horarios de despacho en una planta con múltiples bahías.

En primer lugar, se definen los parámetros, restricciones, la función objetivo y sus relaciones. El alcance del proyecto considera dentro de las variables a estudiar la cantidad de unidades a despachar, el número de bahías, el tipo de despacho, horarios de atención, turnos de despacho y tiempos de demora promedio de despacho. Otros parámetros que pueden se pueden presentar en un problema relacionado no se consideran dentro del alcance.

A partir de la revisión del Estado del Arte, se ha elegido al algoritmo genético como el algoritmo más representativo para abordar problemas similares. Por lo tanto, se utilizará este algoritmo para poder realizar la comparación con el algoritmo seleccionado durante el desarrollo del proyecto.

A continuación, se diseñarán los componentes necesarios que formarán parte de cada uno de los algoritmos.

Luego, se implementará una interfaz que permitirá la interacción con los usuarios, donde se permitirá el ingreso de datos y la exportación de los resultados obtenidos en forma gráfica o en archivos una vez finalice el proceso.

Finalmente, se llevará a cabo la comparación entre los resultados de los algoritmos a través de una experimentación numérica, con la finalidad de analizar la eficiencia en la generación de horarios frente al algoritmo genético.

El alcance del presente proyecto no contempla el desarrollo de un sistema de información.

13.2.4. Limitaciones

El proyecto de investigación tiene como limitaciones la disponibilidad de tiempo del especialista en logística, quien se encarga de la aprobación del documento de definición, descripción y formulación de parámetros y restricciones; y el especialista en algoritmia, el cual se encarga de la aprobación de los documentos relacionados con los algoritmos a utilizar.

Por otra parte, otra limitación presentada es el hardware utilizado para la ejecución de los algoritmos en cuestión, puesto que una infraestructura de poca capacidad puede no ser suficiente para realizar esta ejecución, retrasando los tiempos esperados de respuesta.

13.2.5. Riesgos

A continuación, se presentan los riesgos identificados para el presente proyecto.

Tabla 20: Riesgos del proyecto. Fuente: Elaboración propia.

Descripción	Síntomas	Probabilidad	Impacto	Severidad	Mitigación	Contingencia

El especialista en logística no se encuentra disponible para la validación del documento respectivo.	El especialista no tiene disponibilidad para atender el proyecto por diversos factores, como pueden ser personales o profesionales.	0,5	Alto	Media	Planificar con anticipación la participación del especialista en logística, de manera que se conozca de manera previa los momentos de su participación en el proyecto.	El asesor de la tesis asume el rol de especialista en logística, puesto que tiene los conocimientos necesarios para realizar esta función.
El especialista en algoritmia no se encuentra disponible para la validación del documento respectivo.	El especialista no tiene disponibilidad para atender el proyecto por diversos factores, como pueden ser personales o profesionales.	0,5	Alto	Media	Planificar con anticipación la participación del especialista en algoritmia, de manera que se conozca de manera previa los momentos de su participación en el proyecto.	Un profesor de la universidad del área de ciencias de la computación (con afinidad en algoritmia) asume el rol de especialista en algoritmia, puesto que tiene los conocimientos necesarios para realizar esta función.
El equipamiento a utilizar no se encuentra disponible.	El equipamiento utilizado puede sufrir algún accidente y dejar de funcionar.	0.2	Alto	Alto	El equipamiento a utilizar tiene un plan de tratamiento para prevenir accidentes.	Se cuenta con equipamiento alternativo para realizar el proyecto.

13.2.6. EDT

En la se puede observar la distribución del trabajo que se ha realizado para el presente proyecto.

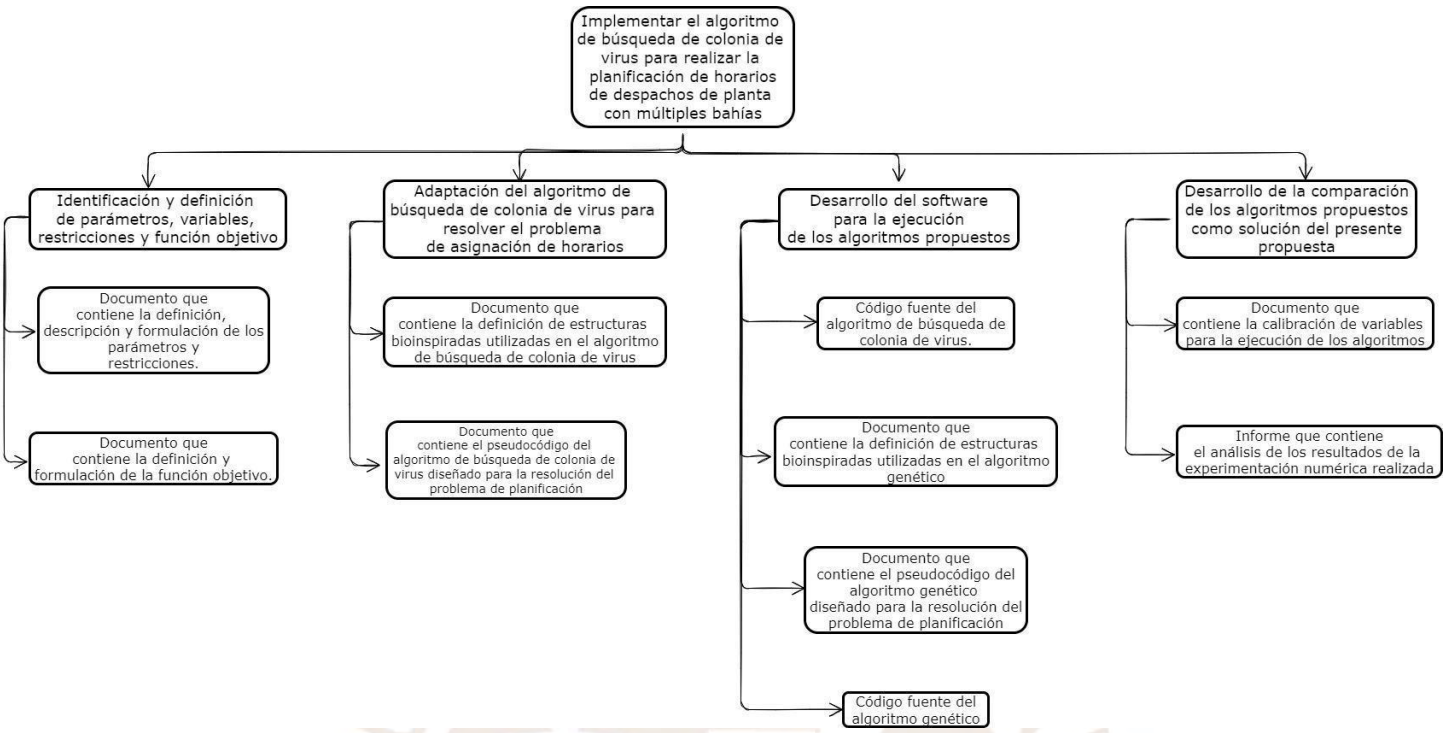

Ilustración 39: Estructura de Distribución del Trabajo. Fuente: Elaboración propia.

13.2.7. Lista de Tareas

En la Tabla 18 se definen las tareas que se van a realizar para lograr los objetivos propuestos.

Tabla 21: Lista de tareas del proyecto. Fuente: Elaboración propia.

Tarea	Duración estimada (días)	Esfuerzo asociado (horas/persona)	Costo estimado (soles)
Objetivo 1: Identificar y definir parámetros, variables, restricciones y su relación mediante la función objetivo, la cual se utilizará para evaluar las posibles soluciones.			
R1.1 Lista de definición de parámetros, variables y restricciones del problema en cuestión.			
Documento que contiene la definición, descripción y formulación de los parámetros y restricciones.	7	21	420
Sesión con el asesor.	1	1	50

Validación del documento de definición por parte del especialista en logística.	1	2	100
R1.2 Documento que contiene la definición y formulación de la función objetivo.			
Documento que contiene la definición y formulación de la función objetivo.	7	21	420
Sesión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100
Objetivo 2: Adaptar el algoritmo de búsqueda de colonia de virus para resolver el problema de asignación de horarios.			
R2.1 Diseño de estructuras de datos que tienen como fin el soporte para el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.			
Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo de búsqueda de colonia de virus.	7	21	420
Sesión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100
R2.2 Algoritmo de búsqueda de colonia de virus adaptado al problema en cuestión.			
Documento que contiene el pseudocódigo del algoritmo de búsqueda de colonia de virus diseñado para la resolución del problema de planificación.	7	21	420
Sesión con el asesor.	1	1	50
Validación del documento de definición del pseudocódigo por parte del especialista en algoritmia.	1	2	100
Objetivo 3: Desarrollar un software para la ejecución de los algoritmos VCS y GA.			
R3.1 Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.			

Código fuente del algoritmo de búsqueda de colonia de virus.	14	42	840
Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo.	5	15	300
Sesión con el asesor.	1	1	50
R3.2 Adaptación del algoritmo genético al problema de planificación.			
Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo genético.	7	21	420
Documento que contiene el pseudocódigo del algoritmo genético diseñado para la resolución del problema de planificación.	7	21	420
Sesión con el asesor.	1	1	50
Validación del documento de definición de estructuras por parte del especialista en algoritmia.	1	2	100
Validación del documento de pseudocódigo por parte del especialista en algoritmia.	1	2	100
R3.3 Codificación del algoritmo genético adaptado al problema de planificación.			
Código fuente del algoritmo genético.	14	42	840
Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo.	5	15	300
Sesión con el asesor.	1	1	50
Objetivo 4: Desarrollar la comparación entre el algoritmo VCS y GA como solución del presente problema.			
R4.1 Calibración de variables			
Documento que contiene la calibración de variables para la ejecución de los algoritmos.	10	30	600

Sesión con el asesor.	1	1	50
Validación del documento de calibración de variables por parte del especialista en algoritmia.	1	2	100
R4.2 Informe de experimentación numérica.			
Informe que contiene el análisis de los resultados de la experimentación numérica.	7	21	420
Sesión con el asesor.	1	1	50
Validación del informe de experimentación numérica por parte del especialista en algoritmia.	1	2	100

13.2.8. Cronograma del proyecto

En la Tabla 19 se define el cronograma de tareas a realizar a lo largo del proyecto para lograr los objetivos propuestos.

Tabla 22: Cronograma del proyecto. Fuente: Elaboración propia.

Semana	Tarea
1	Objetivo 1: Identificar y definir parámetros, restricciones y su relación mediante la función objetivo, la cual se utilizará para evaluar las posibles soluciones.
	R1.1 Lista de definición de parámetros y restricciones del problema en cuestión.
	Documento que contiene la definición, descripción y formulación de los parámetros y restricciones.
	Reunión con el asesor.
	Validación del documento de definición por parte del especialista en logística.
	R1.2 Formulación de la función objetivo para el problema de planificación.
	Documento que contiene la definición y formulación de la función objetivo.

	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
Exposición 1	
2	Levantamiento de observaciones.
	Objetivo 2: Adaptar el algoritmo de búsqueda de colonia de virus para resolver el problema de asignación de horarios.
	R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.
	Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo de búsqueda de colonia de virus.
	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
	R2.2 Algoritmo de búsqueda de colonia de virus adaptado al problema en cuestión.
	Documento que contiene el pseudocódigo del algoritmo de búsqueda de colonia de virus diseñado para la resolución del problema de planificación.
	Reunión con el asesor.
	Validación del documento de definición del pseudocódigo por parte del especialista en algoritmia.
Exposición 2	
3	Levantamiento de observaciones.
	Objetivo 3: Desarrollar un software para la ejecución de los algoritmos VCS y GA.
	R3.1 Codificación del algoritmo de búsqueda de colonia de virus adaptado al problema de planificación.
	Código fuente del algoritmo de búsqueda de colonia de virus.
Exposición 3	
4	Levantamiento de observaciones.
	Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo.
	Reunión con el asesor.

	R3.2 Adaptación del algoritmo genético al problema de planificación.
	Documento que contiene la definición de estructuras bioinspiradas utilizadas en el algoritmo genético.
	Documento que contiene el pseudocódigo del algoritmo genético diseñado para la resolución del problema de planificación.
	Reunión con el asesor.
Exposición 4	
5	Levantamiento de observaciones.
	Validación del documento de definición de estructuras por parte del especialista en algoritmia.
	Validación del documento de pseudocódigo por parte del especialista en algoritmia.
Entregable parcial	
6	R3.3 Codificación del algoritmo genético adaptado al problema de planificación.
	Código fuente del algoritmo genético.
	Pruebas unitarias realizadas sobre las funciones implementadas en el algoritmo.
	Reunión con el asesor.
7	Objetivo 4: Desarrollar la comparación entre el algoritmo VCS y GA como solución del presente problema.
	R4.1 Calibración de variables
	Documento que contiene la calibración de variables para la ejecución de los algoritmos.
	Reunión con el asesor.
Exposición 5	
8	Levantamiento de observaciones.
	Validación del documento de calibración de variables por parte del especialista en algoritmia.
	R4.2 Informe de experimentación numérica.
	Informe que contiene el análisis de los resultados de la experimentación numérica.
9	Reunión con el asesor.
	Validación del informe de experimentación numérica por parte del especialista en algoritmia.
Exposición 6	

10	Levantamiento de observaciones
	Correcciones finales
	Reunión con el asesor
Entregable final	

13.2.9. Recursos del proyecto

En la se definen los recursos que se van a utilizar a lo largo del proyecto.

13.2.9.1. Capacitación

En el siguiente acápite se describen las capacitaciones requeridas para el presente proyecto.

Tabla 23: Requerimientos de capacitación. Fuente: Elaboración propia.

Persona involucrada	Necesidad de capacitación
Jorge Miguel Baca Sáenz	Capacitación en diseño de algoritmos, pruebas unitarias y calibración de variables.
Rony Cueva Moscoso	Ninguna

13.2.9.2. Materiales

En el siguiente acápite se describen los materiales que serán requeridos para el presente proyecto.

Tabla 24: Requerimientos de materiales. Fuente: Elaboración propia.

Material requerido	Objetivo de uso
Útiles de escritorio	Apoyo en los cálculos a realizar, diseño y pruebas.

13.2.9.3. Estándares

En el siguiente acápite se describen los estándares que se van a utilizar en el presente proyecto.

Tabla 25: Requerimientos de estándares. Fuente: Elaboración propia.

Estándar utilizado	Objetivo de uso
Guía PMBOK	Se empleará como base metodológica del proyecto, gracias a la estructura jerárquica de tareas para la obtención de los entregables.
SCRUM	Se emplearán algunas buenas prácticas del marco de trabajo para obtener consistencia en aspectos como la colaboración eficaz y la entrega iterativa de resultados.

13.2.9.4. Equipamiento

En el siguiente acápite se describe el equipamiento necesario para el desarrollo del proyecto.

Tabla 26: Requerimientos de equipamiento. Fuente: Elaboración propia.

Equipamiento requerido	Objetivo de uso
Computadora personal	Se empleará para la codificación de los algoritmos e interfaces propuestas, así como las pruebas necesarias. También se utilizará para la calibración de variables y la experimentación numérica.

13.2.9.5. Herramientas

En el siguiente acápite se describen las herramientas que se van a utilizar en el desarrollo del proyecto.

Tabla 27: Requerimiento de herramientas. Fuente: Elaboración propia.

Herramientas requeridas	Objetivo de uso
C++	Se empleará como lenguaje de programación para el desarrollo de los algoritmos e interfaces.
NetBeans IDE	Se empleará en entorno de desarrollo integrado para la codificación con el lenguaje antes mencionado.
Git	Se empleará como herramienta para el control de versiones del código fuente.
Rstudio	Se empleará el entorno de desarrollo integrado para el análisis de datos para el informe de experimentación numérica.

13.2.9.6. Costeo del proyecto

En la Tabla 25 se define el costo del proyecto.

Tabla 28: Costeo del proyecto. Fuente: Elaboración propia.

Ítem	Descripción	Unidad	Cantidad	Valor Unidad (S/)	Monto Parcial (S/)	Monto Total (S/)
0	Costo total del proyecto					7070
1	Estudiantes o tesistas					6000
1.1	Jorge Miguel Baca Sáenz	Horas	300	20	6000	
2	Otros participantes					
2.1	Especialista en logística		Horas	2	50	100
2.2	Especialista en algoritmia		Horas	14	50	700
3	Materiales e insumos					
3.1	Útiles de escritorio		Unidad	5	5	25

4	Bienes y equipo	Unidad 1	Cantidad 1	Unidad 2	Cantidad 2	
4.1	Computadoras	Equipo	1	Horas		
					Total	13895

13.3. Anexo C – Acta de conformidad de parámetros y restricciones

En el siguiente enlace se puede verificar el acta de conformidad de los parámetros y restricciones:
https://docs.google.com/document/d/1FTiqGxICPaNvNlTc7HTukGn1kyxM9uafH7aFpJ1xRIA/edit?usp=sharing

13.4. Anexo D – Pruebas de la función objetivo

Para realizar las pruebas de la función objetivo, se tenía la siguiente función de minimización:

$$\text{[illegible]} \sum_{hp=1}^{HP} \sum (HT - \sum_{p=0}^{p'} \quad + \text{[illegible]}s(PP\text{[illegible]} - \sum^{p'} T\text{[illegible]}))$$

Para la primera función objetivo, se realizaron pruebas con la función objetivo con los siguientes parámetros:

Tabla 29: Resultados de la prueba de la función objetivo. Fuente: Elaboración propia

Cantidad de pedidos	Cantidad de días	Cantidad de bahías
15	1	3
35	2	3
65	3	3

Para el primer set de datos, una primera distribución obtuvo un puntaje de 10, mientras que una distribución más equitativa entre las bahías obtuvo un puntaje de 6.
Para el segundo set de datos, una primera distribución desigual de pedidos en las bahías obtuvo un puntaje de 30, mientras que una distribución más equitativa obtuvo un puntaje de 18.
Finalmente, para el tercer set de datos, con una distribución desigual se obtuvo un puntaje de 18.33, mientras que para una distribución equitativa devuelve un puntaje de 14.33.
Para efectos del caso, luego de analizar los resultados obtenidos, se tomó en consideración las horas en las que no se realizaba un despacho, lo cual no está representado dentro de la función objetivo original. Para esto, se replanteó la función objetivo a la siguiente fórmula:

$$\text{Maximizar } \sum_{B} (H_{max} + H_{min}) + \sum_{p'} (TP - T_{p'})$$

A partir de la nueva función objetivo, se realizan las pruebas con los mismos datos, obteniendo los siguientes resultados.

Para el primer set de datos, una primera distribución obtuvo un puntaje de 74.66, mientras que una distribución más equitativa entre las bahías obtuvo un puntaje de 66.

Para el segundo set de datos, una primera distribución desigual de pedidos en las bahías obtuvo un puntaje de 150, mientras que una distribución más equitativa obtuvo un puntaje de 136.66.

Finalmente, para el tercer set de datos, con una distribución desigual se obtuvo un puntaje de 239.33, mientras que para una distribución equitativa devuelve un puntaje de 204.66. Con estas pruebas podemos asegurar que la función objetivo formulada nos ayuda a verificar que una solución retorna mejores resultados que otras.

13.5. Anexo E – Acta de conformidad de la función objetivo

En el siguiente enlace se puede verificar el acta de conformidad de la función objetivo: https://docs.google.com/document/d/15ANBCHUJuFuV5DFC2G-bu9YjNsp5s0ewVaPIFbV2RRY/edit?usp=sharing

13.6. Anexo F - Estructuras auxiliares

Para representar la solución que se va a presentar, se utiliza la estructura auxiliar “Horario”, el cual es un arreglo de arreglos con los pedidos ordenados a partir de la hora de despacho asignada, de manera que se puede verificar y utilizar la tabla como guía para realizar los despachos en las bahías.

Tabla 30: Representación de la estructura auxiliar "Horario". Fuente: Elaboración propia.

Bahía 1	Bahía 2	Bahía 3
P4	P2	P5
P7	P1	P6
P9	P8	P3

13.7. Anexo G - Acta de conformidad de las estructuras bioinspiradas

En el siguiente enlace se puede verificar el acta de conformidad de las estructuras bioinspiradas utilizadas en la investigación: https://docs.google.com/document/d/1SL60MB8ZNcvIuz6inoH42MFRiLYUlZJtQ2TaEU0fR_s/edit?usp=sharing

13.8. Anexo H - Acta de conformidad de la adaptación del algoritmo genético

En el siguiente enlace se puede verificar el acta de conformidad de la adaptación del algoritmo genético para el problema presentado en esta investigación:
https://docs.google.com/document/d/1HmF895bbSEr9fkzK7yVe_KsNWY0z45AVaEy4blhMntQ/edit?usp=sharing

13.9. Anexo I - Acta de conformidad del algoritmo búsqueda de colonia de virus

En el siguiente enlace se puede verificar el acta de conformidad de los parámetros y restricciones:
https://docs.google.com/document/d/1o7e5xU_pW8WRHd0yliL-68pitZQw6jheE690Hy82268/edit?usp=sharing

13.10. Anexo J – Código del algoritmo genético

En el siguiente enlace se puede verificar la implementación del algoritmo genético en el lenguaje de programación C++:
https://drive.google.com/file/d/1VuYYtcQ0Nu6N7B-b1bu6GdT_a5u5bvHo/view?usp=sharing

13.11. Anexo K - Pruebas del algoritmo genético

En el siguiente enlace se puede verificar las pruebas realizadas con la implementación del algoritmo genético:
https://docs.google.com/spreadsheets/d/1kJ1a1448Vg69D_jMwtEKRfSo0bMyDajJTBDL0Vll0uc/edit?usp=sharing

13.12. Anexo L – Código del algoritmo búsqueda de colonia de virus

En el siguiente enlace se puede verificar las pruebas realizadas con la implementación del algoritmo búsqueda de colonia de virus:
https://drive.google.com/file/d/1zr21Y5U_Ff_qS7FU9H8kfXhHG9sKyoLp/view?usp=share_link

13.13. Anexo M - Pruebas del algoritmo búsqueda de colonia de virus

En el siguiente enlace se puede verificar las pruebas realizadas con la implementación del algoritmo búsqueda de colonia de virus:
https://docs.google.com/spreadsheets/d/1vd5VyBJNWVjKRlz1PlIjyNohWOO8G6UVNNDRQ1iMKlU/edit?usp=share_link

13.14. Anexo N – Acta de conformidad de la calibración de variables

En el siguiente enlace se puede verificar el acta de conformidad la calibración de variables:

https://docs.google.com/document/d/199LsjXp79unc9yR-YzmqeZEhHktMWTfUwJqtMfp_1Hw/edit?usp=share_link

13.15. Anexo O – Acta de conformidad de la experimentación numérica

En el siguiente enlace se puede verificar el acta de conformidad de la experimentación numérica:

https://docs.google.com/document/d/1XgRcLXgf316c9OmoGnMkXzOR6og7vVQRO8LADMMXXW0/edit?usp=share_link