

Facultad de Ingeniería Informática
Universidad Tecnológica de La Habana
José Antonio Echeverría
cujae

HERRAMIENTA PARA LA GENERACIÓN DE RUTAS EN EL PROBLEMA DEL TRANSPORTE OBRERO: TransO

Trabajo de diploma para optar por el título de Ingeniería en Informática

Autor: Roberto José Nerey Reyes
rnerrey@ceis.cujae.edu.cu

Tutor: Dr. C. Isis Torres Pérez
Facultad Ingeniería Informática (CUJAE)
itorres@ceis.cujae.edu.cu

Consultante: Dr. C. Alejandro Rosete Suárez
Facultad Ingeniería Informática (CUJAE)
rosete@ceis.cujae.edu.cu

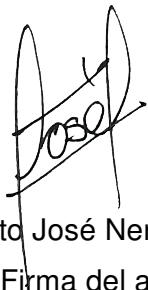
La Habana, Cuba

31 de mayo de 2017

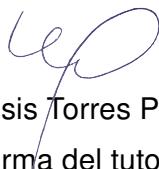
Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al Programa de Informática del Complejo de Investigaciones Tecnológicas Integradas (CITI) y a la Facultad Ingeniería Informática de La Universidad Tecnológica de La Habana José Antonio Echeverría para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los 3 días del mes de mayo del año 2017.



Roberto José Nerey Reyes
Firma del autor



Dr. C. Isis Torres Pérez
Firma del tutor

Opinión del usuario del trabajo de diploma

El Trabajo de Diploma, titulado "*Herramienta para la generación de rutas en el Problema del Transporte Obrero: TransO*", fue realizado en la entidad Complejo de Investigaciones Tecnológicas Integradas (CITI). Se considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface:

Total Parcialmente en _____ %

Por medio de la presente se reconoce la utilidad que tiene el desarrollo de una herramienta que permite resolver el Problema del Transporte Obrero utilizando algoritmos heurísticos. Es importante destacar que los resultados del trabajo que se presenta tienen un aporte significativo pues permite realizar la planificación de rutas a través de un sistema que garantiza la obtención de soluciones viables y que contribuye en gran medida a eliminar el despilfarro de recursos materiales y humanos. Además, puede dar respuestas ante eventualidades determinadas. Esta solución ayuda a planificar los recursos destinados a este servicio facilitando el trabajo y el apoyo a la toma de decisiones en disimiles situaciones.

Por tanto es voluntad de esta institución avalar la utilidad e importancia de la propuesta realizada y documentada en esta tesis de grado. Y para que así conste se firma a los 29 días del mes de Mayo del año 2017.

MSc. Reinaldo Díaz Castro

Nombre del representante de la entidad



Firma

Vice Director de Servicios TIC

Cargo



Dedicatoria

A todas esas grandes personas que me han hecho llegar a este punto de mi vida.

A mis padres, que me han dado la vida y han soñado mucho con este momento.

A mi esposa por todo su apoyo incondicional.

Agradecimiento

A las primeras personas que quiero agradecerles por estar donde estoy hoy es a mis padres, porque ellos siempre se han sacrificado mucho por mí y han sido mi apoyo en todo momento.

A mi otra persona importante, que lucha conmigo todos los días, mi esposa, quien desde que me levanto hasta que me acuesto vive luchando conmigo.

A Yerza y Conrado, grandes profesores que desde que los conocí me exigieron que no podía quedarme como Técnico Medio sino que tenía que esforzarme en coger la Universidad, a ellos muchas Gracias.

A mi tutora Isis, le agradezco que siempre intentara sacar lo mejor de mí, para que diera todo lo que tengo para que este trabajo saliera con la mayor calidad. Gracias por exigirme, darme la oportunidad de hacer este trabajo y confiar en mí.

A la profesora Margarita (Mayi), por ayudarnos tanto y apoyarnos en los últimos años, por transmitirme sus conocimientos y por alentarme a seguir mejorando cada día más.

Al profesor Alejandro Rosete, por mostrarme lo que es necesario para ser un ingeniero de verdad, por confiar en mí y enseñarme tantas cosas.

Al profesor Eduardo Sánchez, por ayudarme cuando no tenía ningún conocimiento de los temas necesarios para los trabajos venideros del proyecto, por atenderme todas mis dudas cuando iba a preguntarle sin ningún reproche.

Al personal de transporte y aseguramiento del CITI, por brindarme la información necesaria para los casos de estudio.

Al Complejo de Investigaciones Tecnológicas Integradas, por proveerme de los recursos para realizar esta tesis.

A las profesoras de Ingeniería del Software Vanessa, Claudia, Anabel y Laura por aclararme las dudas sobre la asignatura cuando lo necesité.

Al profesor Wenny, por todos los conocimientos de Latex que me ha aportado.

A la profesora Lisandra, por todos los conocimientos de minería de datos que me ha aportado.

Al profesor Yusniel Doral, por toda la recopilación de datos de los recorridos aportados para los experimentos de la herramienta.

A Sirenia y Martín, por hacer de nuestra facultad un lugar organizado y agradable, además de ser personas que puedes conversar en todo momento.

A todos los profesores que me dieron clases durante mis cinco años, gracias por formarme como profesional, con ustedes aprendí mucho.

A todos los miembros del proyecto de Optimización, por siempre ser tan buenos compañeros y profesores.

A los miembros del proyecto de GIS Adrián, Michel y Erick que siempre me ayudaron con las dudas que me surgían sobre esos temas.

A Ligia y Calixto, por ser siempre como unos padres para mí y tratarme como parte de su familia.

A Lían, por ser amigo y como un hermano en todo momento tanto las cosas fáciles de la vida como en las cosas difíciles.

A todos mis amigos del primer año de la Universidad de Camagüey, Yasmani, Lino, Danisely, entre otros, los quiero mucho y sé que siempre han estado ahí para mí cuando lo he necesitado.

A todos mis compañeros de año, en especial a Silva, Leandry, Amanda, Sergio, Raidel, Oleida, Geisy, Alex, Yilian, Ronny, Yamel, Jessi, Dariel, Amed, Alejandro, David, Paifer y todos los demás, de una forma u otra aprendí mucho con ustedes y siempre me divertí.

A toda mi familia y amigos, todos tienen un lugar en mi corazón y han sido parte de la persona que soy hoy, gracias por ser parte de vida.

Resumen

El Problema del Transporte Obrero que se puede presentar en cualquier entidad es un ejemplo práctico de un Problema de Planificación de Rutas de Vehículos (VRP). En este problema una entidad determinada tiene la necesidad de brindar como servicio a sus trabajadores la transportación desde diferentes puntos de la ciudad a sus instalaciones y viceversa. La gestión y planificación del parque automotor de la entidad se torna engorroso pues se necesitan coordinar los recorridos que cumplan con las restricciones que se presentan a la hora de realizar la tarea requerida.

El Proyecto Optimización y Metaheurísticas, que se desarrolla en el Complejo de Investigaciones Tecnológicas Integradas (CITI), dispone de una línea de investigación dirigida a resolver diferentes variantes VRP utilizando algoritmos heurísticos. Una de sus tareas es el diseño de modelos de optimización que reflejen las características que se presentan en el Problema del Transporte Obrero y para su uso se decide implementar una herramienta que ilustra como interactuar con estos modelos. En concreto se desarrolla una herramienta que resuelve el Problema del Transporte Obrero a partir de fusionar diferentes variantes clásicas VRP como: el Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos, el Problema de Planificación de Rutas de Vehículos Abierto, el Problema de Planificación de Rutas de Vehículos con restricciones de Distancia y Capacidad y el Problema de Planificación de Rutas de Vehículos con Flota Heterogénea. La fusión de algunas de estas variantes permite que se consideren las distintas características que se presentan en el problema.

La herramienta, denominada **TransO**, sigue un flujo secuencial donde el usuario transita desde la carga, edición y selección de los datos del problema, la asignación y configuración de los elementos involucrados en el proceso y la ejecución de los algoritmos. Además, hace uso de mapas para visualizar tanto la información geográfica de entrada como los recorridos que se obtienen. Otra facilidad que brinda, consiste en la salva de distintas configuraciones, en diferentes formatos, en cada fase por donde el usuario transita para usos posteriores. Por último, los recorridos obtenidos son de gran valor práctico para aquellas entidades donde se presente esta problemática;

facilitando el trabajo y el apoyo a la toma de decisiones en disimiles situaciones.

Abstract

The Worker Transport Problem that can occur in any entity is a practical example of a Vehicle Routing Planning Problem (VRP). In this problem a specific entity has the need to provide as a service to its workers the transportation from different parts of the city to its facilities and vice versa. The management and planning of the car park of the entity becomes cumbersome because it is necessary to coordinate the routes that comply with the restrictions that are presented when performing the required task.

The project Optimization and Metaheuristics, which is developed in the Integrated Technological Research Complex (CITI), has a line of research aimed at solving different VRP variants using heuristic algorithms. One of its tasks is the design of optimization models that reflect the characteristics that are presented in the Worker Transport Problem and for its use it is decided to implement a tool that illustrates how to interact with these models. In particular, a tool is developed that solves the Problem of Transport Workers by merging different classic VRP variants such as: The Multi-Depots Vehicle Routing Problem, the Open Vehicle Routing Problem, the Distance and Capacity Vehicle Routing Problem and the Heterogeneous Fleet Vehicle Routing Problem. The fusion of some of these variants allows to consider the different characteristics that are presented in the problem.

The tool, called TransO, follows a sequential flow where the user moves from loading, editing and selecting the problem data, assigning and configuring the elements involved in the process and executing the algorithms. In addition, it makes use of maps to visualize as much the geographic information of entrance as the routes that are obtained. Another facility that provides, consists of the saving of different configurations, in different formats, in each phase where the user transits for later uses. Finally, the routes obtained are of great practical value for those entities where this problem is present; Facilitating the work and the support to the decision making in dissimilar situations.

Índice general

Introducción	1
1. Fundamentos teóricos	8
1.1. Introducción	8
1.2. Transporte Obrero	8
1.2.1. Caracterización del Problema del Transporte Obrero	10
1.2.2. Herramientas empleadas para resolver el Problema del Trans- porte Obrero	12
1.3. Problemas de planificación de rutas de vehículos	13
1.3.1. Variantes de planificación de rutas de vehículos	15
1.3.1.1. Problema de planificación de rutas de vehículos con múltiples depósitos	16
1.3.1.2. Problema de planificación de rutas de vehículos con flota heterogénea	17
1.3.1.3. Problema de planificación de rutas de vehículos con restricciones de capacidad y distancia	18
1.3.1.4. Problema de planificación de rutas de vehículos abierto	19
1.3.2. Métodos para resolver problemas de planificación de rutas de vehículos	20
1.3.2.1. Metaheurísticas	23
1.3.2.2. Bibliotecas que implementan métodos heurísticos . .	26
1.3.2.3. Biblioteca de algoritmos metaheurísticos: BiCIAM . .	29
1.3.3. Herramientas desarrolladas para la resolución de problemas de planificación de rutas de vehículos	30
1.4. Conclusiones parciales	32

2. Modelado y análisis del Problema del Transporte Obrero	34
2.1. Introducción	34
2.2. Modelo del dominio	34
2.2.1. Reglas generales del negocio	37
2.3. Captura de requisitos	38
2.3.1. Lista de requisitos candidatos	38
2.3.2. Diagrama de casos de uso del sistema	40
2.3.3. Descripción de los actores del sistema	41
2.3.4. Descripción de los casos de uso del sistema	41
2.4. Diagrama de paquetes y sus relaciones	47
2.5. Requisitos no funcionales	48
2.6. Conclusiones parciales	48
3. Herramienta para la generación de rutas en el Problema del Transporte Obrero	50
3.1. Introducción	50
3.2. Vista de la arquitectura	50
3.2.1. Capa Presentación	52
3.2.2. Capa Lógica del Negocio	52
3.2.2.1. Descripción del paquete cujae.inf.citi.om.problem . . .	53
3.2.2.2. Descripción del paquete cujae.inf.citi.om.optimization .	58
3.2.2.3. Descripción del paquete cujae.inf.citi.om.externalService	63
3.2.2.4. Descripción del paquete cujae.inf.citi.om.factoryPattern	64
3.2.3. Capa Acceso a Datos	67
3.3. Diseño de subsistema	69
3.3.1. Subsistema BiCIAM	70
3.3.2. Subsistema BHCVRP	71
3.3.3. Subsistema LMOVVRP	73
3.3.4. Subsistema Geotools	74
3.3.5. Subsistema Server	75
3.4. Mecanismos de diseño	76
3.5. Patrones de diseño	78
3.5.1. Patrón Singleton	79

3.5.2. Patrón Factory Method	80
3.5.3. Otros patrones de diseño	80
3.6. Tratamiento de errores	82
3.7. Modelo de despliegue	83
3.8. Herramientas y tecnologías	85
3.8.1. Lenguaje de programación JAVA	85
3.8.2. Entorno de Desarrollo Integrado: NetBeans	86
3.9. Estudio de factibilidad de la propuesta de solución	87
3.10. Conclusiones parciales	88
4. Validación de la solución propuesta	91
4.1. Introducción	91
4.2. Pruebas	91
4.2.1. Diseño de los casos de prueba	93
4.2.2. Descripción de las instancias	95
4.2.3. Resultados obtenidos en las pruebas	95
4.2.4. Análisis de los resultados para la variante desde los depósitos a la entidad	96
4.2.5. Análisis de los resultados para la variante desde la entidad a los depósitos	99
4.3. Caso de estudios	100
4.3.1. Escenario A	100
4.3.2. Escenario B	102
4.3.3. Escenario C	105
4.4. Conclusiones parciales	108
Conclusiones	110
Recomendaciones	112
Referencias	114
A. Comparación de herramientas empleadas para resolver el problema del Transporte Obrero.	125

B. Comparación de bibliotecas de clases que implementan métodos heurísticos.	127
C. Interfaces gráficas de la herramienta TransO	129
D. Diagramas de secuencia para la comunicación entre TransO y los subsistemas	133
E. Estudio de factibilidad económica de la propuesta de solución.	137

Índice de tablas

2.1. Requisitos candidatos.	39
2.2. Descripción de los actores del sistema.	41
2.3. Descripción del caso de uso Ingresar datos del problema.	42
2.4. Descripción del caso de uso Cargar datos del problema.	42
2.5. Descripción del caso de uso Generar recorridos.	43
2.6. Descripción del caso de uso Geocodificar dirección.	43
2.7. Descripción del caso de uso Gestionar datos del problema.	43
2.8. Descripción del caso de uso Asignar vehículos a depósitos.	44
2.9. Descripción del caso de uso Construir matriz de costo.	44
2.10. Descripción del caso de uso Construir matriz de costo con las distan- cias clásicas.	44
2.11. Descripción del caso de uso Construir matriz de costo con el servicio web.	45
2.12. Descripción del caso de uso Asignar puntos a depósitos.	45
2.13. Descripción del caso de uso Exportar recorridos.	45
2.14. Descripción del caso de uso Visualizar información.	46
2.15. Descripción del caso de uso Salvar configuraciones.	46
2.16. Descripción del caso de uso Configurar parámetros de ejecución.	46
3.1. Descripción del paquete cujae.inf.citi.om.problem.	56
3.2. Descripción del paquete cujae.inf.citi.om.problem.assignment.	57
3.3. Descripción del paquete cujae.inf.citi.om.problem.distance.	58
3.4. Descripción del paquete cujae.inf.citi.om.optimization.designAspects. .	61
3.5. Descripción del paquete cujae.inf.citi.om.optimization.variants.	62
3.6. Descripción del paquete cujae.inf.citi.om.externalService.	64
3.7. Descripción del paquete cujae.inf.citi.om.factoryPattern.interfaces. . .	66

3.8. Descripción del paquete cujae.inf.citi.om.factoryPattern.methods.	67
3.9. Descripción del paquete cujae.inf.citi.om.dataAccess.file.	69
3.10. Descripción del paquete cujae.inf.citi.om.dataAccess.manual.	69
3.11. Descripción del paquete cujae.inf.citi.om.dataAccess.database.	69
3.12. Descripción de los errores comunes en el sistema.	82
3.13. Descripción del diagrama de despliegue.	84
4.1. Datos brindados por el personal de Transporte del CITI.	95
4.2. Resultado de las pruebas de software.	95
4.3. Descripción de los recorridos obtenidos en el escenario A.	101
4.4. Descripción de los recorridos obtenidos en el escenario B.	104
4.5. Descripción de los recorridos obtenidos en el escenario C.	107

Índice de figuras

1.1. Representación gráfica de un VRP.	14
1.2. Clasificación de los algoritmos aproximados.	21
1.3. Diagrama de clases de BiCIAM [1].	30
2.1. Modelo del dominio asociado al problema del Transporte Obrero.	35
2.2. Diagrama de Caso de Uso del Sistema.	40
2.3. Diagrama de paquetes y sus relaciones.	47
3.1. Diagrama de estructuración en capas lógicas.	51
3.2. Diagrama de clases de los paquetes cujae.inf.citi.om.problem.data y cujae.inf.citi.om.problem.solution.	54
3.3. Diagrama de clases del paquete cujae.inf.citi.om.problem.assignment. .	55
3.4. Diagrama de clases del paquete cujae.inf.citi.om.problem.distance. .	55
3.5. Diagrama de clases del paquete cujae.inf.citi.om.optimization.	60
3.6. Diagrama de clases del paquete cujae.inf.citi.om.externalService. . . .	63
3.7. Diagrama de clases del paquete cujae.inf.citi.om.factoryPattern.	65
3.8. Diagrama de clases del paquete cujae.inf.citi.om.dataAccess.	68
3.9. Diagrama de clases para la comunicación con el subsistema BiCIAM. .	70
3.10. Arquitectura de la biblioteca de clases BHCVRP.	71
3.11. Diagrama de clases para la comunicación con el subsistema BHCVRP.	72
3.12. Arquitectura de la biblioteca de clases LMOVVRP.	73
3.13. Diagrama de clases para la comunicación con el subsistema LMOVVRP.	74
3.14. Arquitectura de la biblioteca de clases Geotools [2].	75
3.15. Vista estática del mecanismo de diseño empleado.	77
3.16. Diagrama de secuencia del mecanismo de diseño para la carga dinámica.	78

3.17. Implementación del patrón Singlenton en la clase Problem_TO.	80
3.18. Diagrama de despliegue.	84
4.1. Esquema de casos de prueba.	94
4.2. Comportamiento de los métodos de asignación en la variante MD. . .	97
4.3. Comportamiento del método de asignación Trayectoria Secuencial en la variante MD.	98
4.4. Comportamiento del método de asignación Mejor Asignación en la variante MD.	98
4.5. Comportamiento de los algoritmos en la variante FH.	99
4.6. Solución obtenida en el escenario A en la variante MD.	102
4.7. Solución obtenida en el escenario B en la variante MD.	105
4.8. Solución obtenida en el escenario C en la variante MD.	108

Introducción

Este trabajo se desarrolla en el Complejo de Investigaciones Tecnológicas Integradas (CITI), donde se aborda el Problema del Transporte Obrero como una aplicación real de un Problema de Planificación de Rutas de Vehículos (*Vehicle Routing Problem*, VRP) [3]. La planificación de rutas de vehículos en el caso del Transporte Obrero es una situación común a la que se enfrenta cualquier entidad a la hora de brindar un servicio de transportación a sus trabajadores; cuyo objetivo principal es reducir los costos de esta actividad [4]. Los ahorros en que se puede incurrir justifican en gran medida la utilización de técnicas de Investigación de Operaciones como facilitadoras de la planificación, dado que se estima que los costos del transporte representan entre el 10% y el 20% del costo final de los bienes [4].

Concretamente, los problemas VRP son uno de los problemas de optimización combinatoria más estudiados en las últimas décadas [3]. Los problemas de planificación de rutas de vehículo datan del año 1959, donde Dantzig y Ramser [5] lo introducen y describen como una variante generalizada del Problema del Viajante de Comercio (*Traveling Salesman Problem*, TSP) [6]. Los autores diseñaron una aplicación real para la distribución de gasolina a las estaciones de servicio y se propuso una formulación matemática. Cinco años después, Clarke y Wright proponen el primer algoritmo que resultó efectivo para resolver este problema, el Algoritmo de Ahorros [7]. De esta manera se inician grandes investigaciones y trabajos en el área de planificación de rutas de vehículos [6]. Por un lado, hacia modelos que incorporen cada vez más características de la vida real y por otro lado, en la búsqueda de algoritmos que permitan resolver los problemas de manera eficiente [3].

A grandes rasgos un problema VRP consta de una flota de vehículos que tiene que ser dispuesta desde un depósito para servir a un conjunto de clientes dispersos geográficamente. El objetivo es determinar el conjunto de rutas que minimice el

tiempo total o la distancia total del recorrido realizado por la flota satisfaciendo las restricciones presentes en el problema [3, 8, 9, 10, 11].

Existen diferentes situaciones de la vida real que pueden ser extendidas a diferentes variantes que atienden diversas problemáticas reales del mundo de la logística y el transporte [12, 13, 14]. Ejemplos de aplicaciones prácticas del VRP son visibles en los recorridos de los ómnibus urbanos, la distribución del periódico y del correo o la recolección de los desperdicios sólidos. Estas situaciones cotidianas cada una con sus características puede ser modelada a partir de las distintas variantes VRP. A continuación se mencionan algunas de ellas:

- Problema de Planificación de Rutas de Vehículos con Capacidades (*Capacitated VRP, CVRP*) [15, 16].
- Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (*Multi-Depot VRP, MDVRP*) [17, 18].
- Problema de Planificación de Rutas de Vehículos con Flota Heterogénea (*Heterogenous Fleet VRP, HFVRP*) [19, 17].
- Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo (*VRP with Time Windows, VRPTW*) [13, 14].
- Problema de Planificación de Rutas de Vehículos Abierto (*Open VRP, OVRP*) [19, 20].
- Problema de Planificación de Rutas de Vehículos con restricciones de Capacidad y Distancia (*Distance and Capacity VRP, DCVRP*) [21].

Un VRP es un problema de optimización combinatoria con categoría computacional NP-duro [13]. Estos problemas no han podido ser reducidos a problemas NP y se consideran muy difíciles de resolver [22]. En estos problemas se trata de hallar la mejor solución entre un número finito de posibilidades. Para resolver estos problemas y obtener una mayor eficiencia en los resultados se aplican los algoritmos aproximados dentro de los que están comprendidos las heurísticas y las metaheurísticas [23, 24].

Los algoritmos metaheurísticos son algoritmos de propósito general, y se definen como un proceso iterativo que combinan distintos aspectos para explorar el espacio de búsqueda. Los valiosos resultados alcanzados con estos algoritmos en los

problemas de optimización combinatoria han propiciado el desarrollo de numerosas bibliotecas que implementan algoritmos metaheurísticos. Una de estas bibliotecas es BiCIAM [25, 26], desarrollada en la Facultad de Ingeniería Informática. Esta biblioteca fue implementada en el lenguaje de programación Java y contiene algoritmos metaheurísticos basados en un punto y en poblaciones de puntos.

Por otra parte, en noviembre del 2010 se inaugura el CITI con el propósito de desarrollar tecnologías integradas de un amplio espectro de las ciencias técnicas, en interés de la seguridad y el orden interior. Con este fin se realiza un trabajo coordinado entre la CUJAE, el MININT y otras instituciones, vinculando las necesidades de superación científica de especialistas con las soluciones concretas y ágiles, mediante la ejecución de proyectos por grupos de trabajo flexibles. La entidad cuenta con 137 trabajadores de los cuales 101 se les brinda un servicio de transportación desde diferentes puntos de la ciudad hacia las instalaciones del centro y viceversa. Para llevar a cabo este servicio de transportación la empresa dispone de diferentes medios de transporte con distintas capacidades. Además, estos vehículos inician su recorrido desde diferentes áreas de estacionamiento dependiendo del lugar de parqueo del vehículo. Este lugar está determinado por la ubicación geográfica donde reside el conductor del vehículo.

Para identificar las características presentes en el problema del Transporte Obrero se realizó un estudio en el propio CITI. Para cualquier entidad con esta necesidad, la gestión y planificación de su parque automotor se torna engorrosa cuando se desea brindar a sus trabajadores un servicio con calidad, pues en la planificación se tienen que considerar todos los recursos destinados a este servicio. Específicamente, en países en vía de desarrollo como Cuba, las empresas realizan la planificación de los recorridos de forma manual incurriendo en las siguientes situaciones:

- Se produce un desgaste intelectual en la planificación debido a que se realiza de forma manual, siendo agotador por no decir imposible la planificación de este servicio cuando involucra un número considerable de trabajadores y vehículos.
- Una vez que se obtiene una planificación permanece estática, pues se torna engorroso realizar cambios en la misma, debido a que se realiza de forma manual y requiere de tiempo su confección.

- Existe una alta probabilidad de que se despilfarren recursos materiales y humanos (combustible, personal, vehículos), pues está presente el factor humano que puede no ser adecuado a la hora de planificar determinados recorridos donde intervengan una cantidad considerable de recursos.
- Esta forma de planificar puede no ser idónea para dar respuesta inmediata ante una eventualidad determinada, pues determinadas afectaciones como la entrada de un nuevo trabajador o la rotura de un vehículo implican determinar nuevos recorridos. Siendo en estos momentos la solución más adoptada el traslado por sus propios medios de los trabajadores involucrados en dicha situación.

A partir de la situación antes descrita se identifica el siguiente **problema a resolver** que en la actualidad no existe una herramienta que realice la planificación de los recorridos del Transporte Obrero de forma rápida y eficiente, disminuyendo el consumo de recursos y respondiendo de forma oportuna ante eventualidades que se puedan presentar.

Para dar solución a este problema se enfocó el **objeto de estudio** en los problemas de planificación de rutas de vehículos, servicios integrados y los algoritmos heurísticos. El **campo de acción** está enfocado en las variantes de planificación de rutas de vehículos, los servicios web y las metaheurísticas.

Con el fin de resolver estos problemas se definió el siguiente **objetivo general**: desarrollar una herramienta para la generación de rutas de vehículos para resolver el Problema del Transporte Obrero.

Para dar cumplimiento al objetivo general se definen los siguientes **objetivos específicos** con sus respectivas **tareas**:

- Identificar las características presentes en el Problema del Transporte Obrero que permita su modelado y resolución.
 1. Realizar una revisión bibliográfica sobre el Problema del Transporte Obrero y las herramientas empleadas para resolverlo.
 2. Fundamentar el marco teórico sobre los Problemas de Planificación de Rutas de Vehículos, sus variantes y métodos de solución para este tipo de problemas.

3. Realizar un análisis acerca de las herramientas desarrolladas para la resolución Problemas de Planificación de Rutas de Vehículos.
- Implementar una arquitectura flexible que permita interactuar con modelos de optimización en la resolución del Problema del Transporte Obrero.
 1. Diseñar una arquitectura que contemple las características de distintas variantes de planificación de rutas de vehículos para la generación de rutas en el Problema del Transporte Obrero.
 2. Implementar una herramienta para la generación de rutas del Transporte Obrero.
 3. Incorporar a la herramienta la entrada de datos desde diferentes fuentes de datos.
 4. Desarrollar una interfaz de usuario que permita la gestión de los datos con los cuales se desea trabajar.
 5. Incorporar a la herramienta distintos métodos para el cálculo de distancias entre dos puntos.
 6. Incorporar métodos de asignación para resolver el problema del Transporte Obrero cuando se trata como una variante del Problema de Planificación de Rutas de Múltiples Depósitos.
 7. Desarrollar una interfaz de usuario que permita en dependencia de la experiencia del usuario la utilización de diferentes metaheurísticas que se encuentran en BiCIAM, con diferentes parámetros y configuraciones.
 8. Implementar las clases necesarias para la comunicación con la biblioteca de clases BiCIAM para la utilización de algoritmos metaheurísticos.
 9. Exportar los recorridos obtenidos con la herramienta en formato pdf que facilite su uso por los directivos.
 10. Realizar la salva de las distintas configuraciones, en cada fase por donde el usuario transita para usos posteriores.
 11. Visualizar la información mediante mapas.
 12. Confeccionar un tutorial para el uso de la herramienta.

- Validar el funcionamiento de la herramienta mediante la ejecución de diferentes experimentos.
 1. Definir distintos escenarios de pruebas para la validación de la herramienta.
 2. Realizar las pruebas para identificar los mejores algoritmos y métodos en la solución del Problema del Transporte Obrero.
 3. Analizar los resultados obtenidos para cada tipo de prueba realizada.
 4. Obtener los datos necesarios para modelar el caso de estudio con sus distintos escenarios.
 5. Obtener los resultados arrojados por los experimentos utilizando el algoritmo que mejor rendimiento obtuvo en las pruebas realizadas.

A partir de lo descrito anteriormente se puede resumir que el **valor práctico** de este trabajo está relacionado con el aporte significativo de realizar la planificación de rutas a través de una herramienta que garantice la obtención de soluciones viables y no permita el despilfarro de recursos materiales y humanos. Además pueda dar respuestas ante eventualidades determinadas. Esta solución ayuda a planificar los recursos destinados a este servicio, utilizando distintos algoritmos metaheurísticos en la resolución de diferentes variantes VRP.

Para alcanzar los objetivos trazados, la estructura de este trabajo se ha ordenado en cuatro capítulos principales, a los que precede esta introducción. Además, se completa con otras secciones de caracteres complementarios pero también importantes, como son las conclusiones y la bibliografía consultada.

En el **Capítulo 1:** "Fundamentos teóricos" se realiza una revisión del estado del arte de los problemas de planificación de rutas de vehículos y sus variantes, así como la forma de modelar y solucionar estos problemas. Además, se hace una caracterización de los Problemas del Transporte Obrero y las herramientas específicas para dar solución a este tipo de problemas. En el **Capítulo 2:** "Modelado y análisis del Problema del Transporte Obrero" se dedica a describir el modelado y análisis del Problema del Transporte Obrero mediante el uso de una herramienta de ingeniería de software asistida por computadora la cual utiliza un conjunto de artefactos propuestos por RUP. El **Capítulo 3:** "Herramienta para la generación de rutas en el Problema del

"Transporte Obrero" se describe la solución propuesta mediante la vista de la arquitectura de la herramienta. Otros elementos como el consumo de diferentes servicios web, el trabajo con los subsistemas, la comunicación con BiCIAM y la estimación de esfuerzo y costo de la implementación. Por último, el **Capítulo 4**: "Validación de la solución propuesta" presenta la validación de la herramienta a partir del conjunto de pruebas ejecutadas y los resultados obtenidos. Además, se hace un análisis de los resultados obtenidos en la herramienta mediante diferentes escenarios de un mismo caso de estudio.

Capítulo 1

Fundamentos teóricos

1.1. Introducción

En el presente capítulo se tiene como objetivo establecer las bases teóricas de la investigación, sintetizando los aspectos que se encuentran relacionados con el Problema del Transporte Obrero. Se realiza una caracterización de este problema y la relación que tiene con los Problemas de Optimización Combinatoria, específicamente con el Problema de Planificación de Rutas de Vehículos y sus variantes. Otros temas que se abordan son los métodos heurísticos existentes para darle solución a este tipo de problema y las herramientas que los resuelven.

1.2. Transporte Obrero

El transporte es una actividad del sector terciario, entendida como el desplazamiento de objetos o personas de un lugar (punto de origen) a otro (punto de destino) en un vehículo (medio o sistema de transporte) que utiliza una determinada infraestructura (red de transporte). Esta ha sido una de las actividades terciarias que mayor expansión ha experimentado a lo largo de los últimos dos siglos debido a la industrialización; al aumento del comercio y de los desplazamientos humanos tanto a escala nacional como internacional; y a los avances técnicos que se han producido y que repercuten en mayor rapidez, capacidad, seguridad y con menor coste de los transportes.

Un término muy utilizado dentro del contexto del transporte es el de movilidad obligada, que no es más que el desplazamiento entre el lugar de residencia y el trabajo o

centro de estudios. La movilidad obligada ocupa el 54 % de los desplazamientos diarios, siendo un 40 % dedicado al sector de la población que trabaja y el restante 14 % al sector estudiantil. En el caso particular del sector empresarial estas cifras están influenciadas por el aumento del tránsito en la ciudad y sus alrededores pues afecta directamente la llegada puntual de los empleados a sus puestos de trabajo. De igual manera, la producción diaria se ha visto perjudicada por las ausencias operativas. Muchas empresas buscan mejorar este tema para que sus trabajadores sean más eficientes y por ende no disminuya la producción. La falta de soluciones efectivas a los problemas del transporte público, ha llevado a las empresas y compañías a tomar medidas y a recurrir a alternativas para atenuar la crisis del transporte.

Una de las alternativas o medidas más usadas por las empresas es brindarles a los trabajadores un servicio de transporte de personal o transporte obrero como comúnmente es llamado. Este servicio consiste en realizar un conjunto de recorridos desde diferentes puntos de la ciudad o del territorio hacia las instalaciones de la entidad y viceversa. Aunque la mayoría de estos recorridos se realizan en medios de transporte privado, es también muy frecuente que se lleven a cabo de forma colectiva en buses y caravanas especiales para este propósito.

En Cuba y otros países es habitual que algunas empresas dediquen determinados vehículos para transportar a los trabajadores desde diferentes puntos hacia el centro de trabajo. Normalmente, los trabajadores que se benefician con este tipo de servicio reciben una identificación dada por la empresa que les permite hacer uso de estos servicios durante su permanencia en la misma, a un costo muy bajo (subsidiado) o sin ninguno costo. No obstante, existen empresas que no cuentan con la solvencia suficiente para brindar este tipo de servicio y sus empleados o trabajadores se ven obligados a usar los vehículos del transporte público de la ciudad.

En el mundo empresarial está cada vez más extendida la práctica de ofrecer a sus empleados una serie de beneficios e incentivos sociales, entendidos como una forma de mejorar la motivación del personal, aumentando de esta forma la productividad. Empresas de todos los sectores y regiones están viendo beneficios reales al facilitar a los trabajadores un servicio de transportación, no solo para ellas sino también para sus trabajadores o empleados al aumentar la calidad de vida en el trabajo. Estas medidas aportan disimiles ventajas tanto para la empresa como para los trabajadores.

Algunas de ellas se mencionan a continuación:

- La imagen empresarial se mejora pues se incluye un servicio que favorece el bienestar de los trabajadores y el medioambiente.
- Se reduce el estrés laboral ocasionado por las cogeraciones de tráfico, aumentando considerablemente la productividad y motivación del personal.
- Aumenta la puntualidad de la plantilla reduciendo el absentismo laboral.
- Se consigue un mayor y mejor aprovechamiento de la jornada laboral.
- Existe una disminución real del riesgo de accidentes de tráfico de camino al trabajo o a su regreso a casa.
- Los trabajadores no incurren en gastos asociados a la transportación.

1.2.1. Caracterización del Problema del Transporte Obrero

Para identificar las características presentes en un problema de este tipo se realizó un estudio en la entidad cubana Complejo de Investigaciones Tecnológicas Integradas (CITI). El CITI desarrolla tecnologías integradas de un amplio espectro de las ciencias técnicas, en interés de la seguridad y el orden interior, a través de un trabajo coordinado entre la CUJAE, el MININT y otras instituciones, vinculando las necesidades de superación científica de especialistas con las soluciones concretas y ágiles, mediante la ejecución de proyectos por grupos de trabajo flexibles. La empresa cuenta con una plantilla de 137 trabajadores de los cuales 101 reciben el servicio de transportación. Además la empresa consta de 5 vehículos destinados para este servicio con una capacidad total de 110 asientos.

Para hacer frente al transporte de los trabajadores de esta organización, debemos aclarar que existen características específicas para este tipo de prestación de servicios, que cualquier empresa interesada en implementar, debe seguir y cumplir. Las características identificadas en el estudio realizado son:

- Cada punto de recogida solo debe ser visitado una sola vez en el recorrido hacia el centro de trabajo o viceversa.

- Cada vehículo debe estar destinado a un solo recorrido, aunque existen casos excepcionales donde se requiere que un vehículo realice más de un recorrido.
- Se debe satisfacer la demanda en todos los puntos de recogida, es decir, en el recorrido todos los trabajadores deben ser transportados.
- No se considera factible transportar trabajadores de pie o más de una persona por asiento.
- No pueden existir más recorridos que vehículos disponibles.
- La duración máxima del tiempo de viaje en cada recorrido no debe exceder de un tiempo determinado.

Contemplar todas estas características a la hora de planificar este tipo de servicio es muy engorroso. Muchas empresas que brindan este servicio realizan la planificación de forma manual lo que conlleva a:

- Se produce un desgaste intelectual en la planificación debido a que se realiza de forma manual, siendo agotador por no decir imposible la planificación de este servicio cuando involucra un número considerable de trabajadores y vehículos.
- Una vez que se obtiene una planificación permanece estática, pues se torna engorroso los cambios en la misma, debido a que se realiza de forma manual y requiere de tiempo su confección.
- Existe una alta probabilidad de que se despilfarren recursos materiales y humanos (combustible, personal, vehículos), pues está presente el factor humano que puede no ser adecuado a la hora de planificar determinados recorridos donde intervengan una cantidad considerable de recursos.
- Esta forma de planificar puede no ser idónea para dar respuesta inmediata ante una eventualidad determinada, pues determinadas afectaciones como la entrada de un nuevo trabajador o la rotura de un vehículo implican determinar nuevos recorridos. Siendo en estos momentos la solución más adoptada el traslado por sus propios medios de los trabajadores involucrados en dicha situación.

1.2.2. Herramientas empleadas para resolver el Problema del Transporte Obrero

La importancia alcanzada por este problema en el sector empresarial ha motivado el desarrollo de múltiples herramientas para gestionar rutas para la transportación de trabajadores. La mayoría de estas herramientas son software propietario y en nuestro país se dificulta mucho la obtención de dichas herramientas. Vale aclarar que las características presentes en el Problema del Transporte Obrero coinciden o son muy similares a las características que se presentan en el Problema del Transporte Escolar. Es por esta razón que herramientas diseñadas específicamente para dar solución a cada problema pueden ser utilizadas en ambos casos. A continuación se describen cuatro herramientas específicas, sus características y funcionalidades para gestionar las rutas de transportación de trabajadores.

- **Novatrans:** es una herramienta que incorpora tecnología de localización en tiempo real, que garantiza una forma de control, gestión y regulación del servicio. Mediante este software se puede controlar cuál es el conductor encargado de cada vehículo y las horas que dedica a cada recorrido. La herramienta gestiona de forma absoluta y en tiempo real los recorridos que siguen los vehículos. Por otro lado, Novatrans también dispone de un sistema de avisos automáticos y control de gastos, que permiten a los administradores del sistema estar al corriente de todos los cambios y modificaciones que se deban realizar a la flota de vehículos [27].
- **TRACEUS:** es una herramienta web desarrollada por Nunsys. Esta herramienta permite la gestión y optimización de las rutas escolares, de esta forma, la ubicación del vehículo, el número de niños que son trasladados o las demoras y atascos que pueden existir en el trayecto, pueden ser verificadas por maestros y padres de familia [28].
- **SIGTEBAL:** es una herramienta construida a partir del diseño de una base de datos geográfica de la red vial regional de Mallorca a la cual se le incorpora información correspondiente al sistema de rutas de transporte escolar (paradas, itinerarios, colegios). Tanto el diseño como la implementación de la herramienta están orientados al personal técnico de la Consejería de Educación del Go-

bierno Balear para facilitar las tareas de gestión de rutas escolares. El proceso analítico y de planificación de rutas se realiza a partir de la implementación de diversos métodos de optimización [29].

- **ONTRACK CORPORATE:** es una herramienta que genera rutas con las características y restricciones que le imponga el usuario. La herramienta monitorea en tiempo real la ubicación y estado de los vehículos. Además, emplea un sistema de notificaciones al teléfono celular para mantener informado a los trabajadores sobre la llegada de su vehículo de recogida [30].

En el **Apéndice A** se presenta una tabla con las comparación de las herramientas descritas anteriormente. Para la comparación se consideran los siguientes criterios: lenguaje de programación en que fue desarrollada la herramienta, algoritmos utilizados para la optimización de los recorridos, uso de mapas para la visualización de la información, así como otras funcionalidades que le aporten un valor agregado.

Por último, a partir del análisis realizado a las herramientas mencionadas anteriormente se puede concluir que:

- El total de las herramientas estudiadas están implementadas en los lenguajes de programación C++ o Php.
- Una gran parte de estas herramientas no utilizan métodos de optimización a la hora de generar los recorridos, en otros casos se desconoce.
- Es un factor común que todas estas herramientas empleen mapas para visualizar los recorrido.
- En todos los casos es necesario pagar la licencia para la explotación de la herramienta.

1.3. Problemas de planificación de rutas de vehículos

Los Problema de Planificación de Rutas de Vehículos son considerados problemas de optimización combinatoria en el cual el número de soluciones factibles aumenta de forma exponencial con respecto al número de clientes a visitar. El mismo

fue introducido como una variante generalizada del Problema del Viajante de Comercio (*Traveling Salesman Problem*, TSP) [31]. El TSP es un problema NP-duro [32] ampliamente estudiado. Este tipo de problemas se caracteriza por ser fácilmente descriptible pero difícil de resolver. Se puede decir que los problemas NP-duro son los problemas más difíciles de la clase NP puesto que pueden ser resueltos por métodos matemáticos exactos pero no es posible resolverlos en tiempo polinomial [16]. Con el fin de resolver dichos problemas, en ocasiones es necesario comprometer algunos requisitos de optimalidad y construir una estructura de control que aunque no garantice encontrar la respuesta óptima casi siempre encuentre una buena solución [23].

El Problema de Planificación de Rutas de Vehículos más básico consiste en la existencia de un depósito central que dispone de una flota de vehículos homogénea y debe atender a un conjunto de clientes dispersos geográficamente [8]. Su objetivo es encontrar las rutas a realizar por cada uno de los vehículos de manera que se satisfagan los requerimientos de los clientes, las restricciones operativas y se minimice el costo total de transportación [6]. A continuación en la Figura 1.1 se muestra un ejemplo gráfico de un VRP.

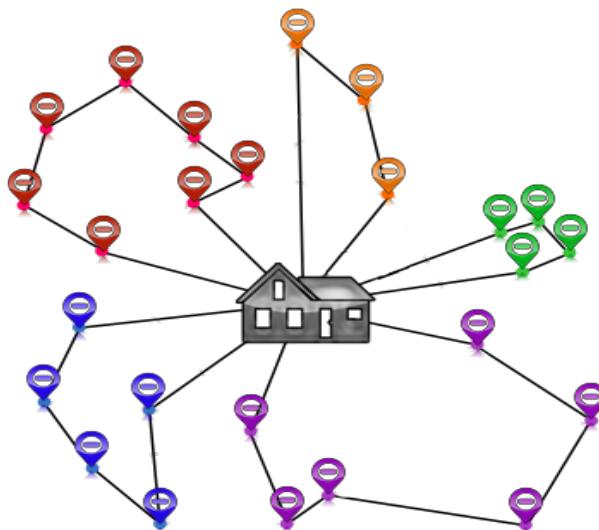


Figura 1.1: Representación gráfica de un VRP.

El problema data del año de 1959 y fue introducido por Dantzig y Ramser, quienes describieron una aplicación real de la entrega de gasolina a las estaciones de servicio y propusieron una formulación matemática [3]. Cinco años después, Clarke y Wright propusieron el primer algoritmo que resultó efectivo para resolver este problema, el

Algoritmo de Ahorros [7, 14, 33]. De esta manera se inician grandes investigaciones y trabajos en el área planificación de rutas de vehículos [6, 7]. Por un lado, hacia modelos que incorporen cada vez más características de la vida real, y por otro lado, en la búsqueda de algoritmos que permitan resolver los problemas de manera eficiente [3].

Los modelos básicos que se proponen son usados para resolver situaciones prácticas y pueden ser extendidos a diferentes variantes que atienden diversas problemáticas reales del mundo de la logística y el transporte [12, 13, 14]. Ejemplos de aplicaciones prácticas del VRP son visibles en los recorridos de los ómnibus urbanos, la distribución del periódico y del correo o la recolección de los desperdicios sólidos. Estas situaciones cotidianas cada una con sus características puede ser modelada a partir de las distintas variantes VRP.

1.3.1. Variantes de planificación de rutas de vehículos

Los Problemas de Planificación de Rutas de Vehículos son usados para resolver situaciones prácticas y pueden ser extendidos a diferentes variantes que atienden diversas problemáticas reales del mundo de la logística y el transporte. Estas situaciones incluyen la adición de variables y restricciones y pueden ser modeladas a partir de las diferentes variantes VRP existentes o de la fusión entre ellas. A continuación se explican algunas de las variantes de Problemas de Planificación de Rutas de Vehículos abordadas en la literatura.

La versión clásica de un VRP se denomina Problema de Planificación de Rutas de Vehículos con Capacidades (*Capacitated VRP*, CVRP) [15, 16]. En este problema se dispone de una flota homogénea de m vehículos con cierta capacidad máxima para repartir una determinada cantidad de mercancía desde un depósito central a una serie de clientes. El objetivo del problema es diseñar m rutas que partan del depósito, visiten varios clientes satisfaciendo su demanda y vuelvan al depósito, sin exceder la capacidad máxima de los vehículos con un coste total mínimo. Con el fin de ajustarse mejor a los problemas de la vida real, a partir de la variante clásica surgen numerosas variantes con restricciones adicionales. Algunas de las más importantes se mencionan a continuación:

- Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (*Multi-*

Depot VRP, MDVRP) [17, 18, 34].

- Problema de Planificación de Rutas de Vehículos con Flota Heterogénea (*Heterogenous Fleet VRP, HFVRP*), [17, 19].
- Problema de Planificación de Rutas de Vehículos con Ventanas de Tiempo (*VRP with Time Windows, VRPTW*) [13, 14].
- Problema de Planificación de Rutas de Camiones y Remolques (*Truck and Trailer Routing Problem, TTRP*) [10, 35].
- Problema de Planificación de Rutas de Vehículos con restricciones de Capacidad y Distancia (*Distance and Capacity Vehicle Routing Problem, DCVRP*) [21].
- Problema de Planificación de Rutas de Vehículos Abierto (*Open Vehicle Routing Problem, OVRP*) [19].

Como resultado de la creciente aparición de problemas de planificación de rutas de vehículos de la vida práctica cada vez más complejos, que abarcan un gran número de restricciones y características a tener en cuenta, se hace necesario crear nuevas variantes como resultado de fusionar dos o más de ellas. Esto tiene como ventaja que se reduzcan los límites en los que una misma variante de VRP pueda abarcar un problema de la vida real, ya que las variantes más puras (como las mencionadas anteriormente), dan solución solo a los problemas de planificación de rutas de vehículos que presenten las características propias de cada una de ellas por si solas. Por la importancia que tiene para el trabajo realizado se describen en más detalles las variantes: Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos [34], Problema de Planificación de Rutas de Vehículos con Flota Heterogénea [17, 19], Problema de Planificación de Rutas de Vehículos con restricciones de Capacidad y Distancia [21] y Problema de Planificación de Rutas de Vehículos Abierto [19].

1.3.1.1. Problema de planificación de rutas de vehículos con múltiples depósitos

El Problema de Planificación de Rutas de Vehículos con Múltiples Depósitos (*Multi-Depot VRP, MDVRP*) [17, 18] es una extensión del VRP clásico (CVRP) que

incorpora varios depósitos con ubicaciones diferentes. Cada depósito cuenta con una flota limitada de vehículos con capacidad restringida, utilizada para repartir o recoger los productos requeridos por los clientes, cuya localización y demanda es también conocida de antemano.

Este problema aparece por primera vez en [36], y más tarde es formalizado por [37]. Luego en [38] los autores propusieron un algoritmo basado en ramificación y acotamiento para dar solución al problema. Desde entonces hasta la actualidad, se han registrado más de 116 trabajos sobre esta variante del problema y sus combinaciones, tal y como es descrito por [39] en su trabajo de revisión literaria sobre los MDVRPs y métodos de solución.

Según se explica en [34], si los clientes están agrupados alrededor de los depósitos, entonces el problema puede modelarse como CVRPs independiente. Sin embargo, si los clientes y los depósitos están mezclados, el problema de planificación debe ser resuelto en dos fases. Inicialmente se asignan los clientes a los depósitos y luego se determinan los recorridos a realizar por cada depósito para visitar a los clientes asignados.

El MDVRP se puede identificar en una gran variedad de contextos y su correcta solución permite una reducción considerable en los costos de distribución. Estudios de casos documentados incluyen entrega de alimentos procesados, distribución de productos químicos, reparto de bebidas gaseosas, distribución de productos en empresas multinacionales, distribución de gases industriales, despacho de vehículos cisterna con derivados del petróleo, distribución de alimentos lácteos, entre otros.

En el caso concreto del Problema del Transporte Obrero esta variante se encuentra presente debido a que el problema puede ser enfocado desde los puntos de estacionamiento de los conductores de los vehículos hacia el centro o la entidad que brinda el servicio de transportación. De igual forma cada estacionamiento dispone de un vehículo con una capacidad determinada para realizar la recogida de los trabajadores.

1.3.1.2. Problema de planificación de rutas de vehículos con flota heterogénea

El Problema de Planificación de Rutas de Vehículos con Flota Heterogénea (*Heterogenous Fleet VRP*, HFVRP), [17, 19] es una variante donde los vehículos de la

flota presentan características y/o capacidades diferentes. La capacidad heterogénea de la flota de vehículos se debe considerar para determinar las rutas a construir, ya que un vehículo más grande puede realizar una ruta más larga o que tenga mayor concentración de demanda. La cantidad de vehículos de cada tipo es limitada y existe un costo fijo o un costo por unidad de distancia recorrida asociado a cada tipo de vehículo. En [16] está disponible una recopilación de las diferentes variantes de este problema y de aquellos autores que las han abordado hasta esa fecha.

Para dar solución a este problema, en la literatura se han identificado diferentes formas como en [40] donde se desarrolla una heurística para resolver el Problema de Planificación de Rutas de Vehículos Heterogéneo con tamaño de la flota fijo. Otro trabajo es el presentado en [41] con dos algoritmos meméticos (también conocidos como algoritmos genéticos híbridos) que resuelven con éxito este problema. Además, en [42] se desarrolló un algoritmo basado en el algoritmo Búsqueda Tabú para dar solución a un problema con características similares a este.

Entre las actividades que se ajustan en este modelo, se pueden mencionar las rutas escolares y las rutas del transporte público. Vale mencionar que en el problema que se aborda en este trabajo, los vehículos que componen la flota pueden, y generalmente sucede así, presentar capacidades diferentes.

1.3.1.3. Problema de planificación de rutas de vehículos con restricciones de capacidad y distancia

El Problema de Planificación de Rutas de Vehículos con restricciones de Capacidad y Distancia (*Distance and Capacity Vehicle Routing Problem*, DCVRP) [21] es una extensión del VRP clásico (CVRP) donde existe una restricción adicional. Esta restricción consiste en que cada ruta no puede exceder una longitud máxima de recorrido permitida, o el equivalente a un máximo tiempo de duración del recorrido.

Este problema aparece por primera vez en [13] donde se hace una extensa revisión de los métodos heurísticos para VRP, centrándose especialmente en CVRP y DCVRP. Dado que hay varios depósitos y un conjunto de clientes dispersos geográficamente, el objetivo de DCVRP es determinar la ruta de entrega de costo mínimo, bajo restricciones de capacidad y distancia. Existen dos variantes que derivan directamente de este tipo de problema. La primera variante, DCVRP Fix es una ex-

tensión del DCVRP simétrico tradicional, con restricciones de servicio y tiempo de viaje adicionales, minimización del número de vehículos y aplicación flexible a problemas simétricos y asimétricos. La segunda variante, DCVRP Flex es una relajación de DCVRP Fix para permitir la asignación flexible de depósitos de inicio y final. Esto permite a los vehículos la libertad de iniciar y finalizar su recorrido en diferentes depósitos, al tiempo que permite visitas intermedias a cualquier depósito (para reabastecer mercancías) durante el recorrido.

Para dar solución a este problema, en la literatura se han identificado diferentes formas como los modelos de red y las formulaciones de programación entera mixta. Se pueden mencionar disímiles actividades que se ajustan a esta variante, siempre y cuando requieran establecer limitaciones en la utilización de los vehículos en cada uno de los recorridos. En el caso concreto del Problema del Transporte Obrero esta variante se encuentra presente debido a que se puede establecer para cada vehículo una duración en tiempo o distancia.

1.3.1.4. Problema de planificación de rutas de vehículos abierto

El Problema de Planificación de Rutas de Vehículos Abierto (*Open Vehicle Routing Problem*, OVRP) [19] es una variante VRP, donde los vehículos no están obligados a regresar al depósito después de cumplir con un conjunto de nodos cliente haciéndoles sus respectivas entregas [43].

Este problema se define en los años ochenta donde recibió muy poca atención. Sin embargo, a partir del año 2000 se han realizado varias investigaciones exitosas, como la realizada por [44], utilizando la Búsqueda Tabú y el algoritmo Recocido Determinístico. Para la solución de Problemas de Planificación de Rutas de Vehículos Abierto, OVRP, en la literatura se han propuesto algunos algoritmos heurísticos como [45] que propone una heurística para resolver este problema. Más tarde en [46] se diseña un algoritmo de Búsqueda Tabú como método de solución a un problema de ruteo de vehículos abierto y en [47] se propone el algoritmo de Branch and Cut para el Problema de Vehículos Abierto Capacitado COVRP [47]. Por otra parte en [43] se propone una metaheurística para examinar soluciones de vecino más cercano y en cuanto a métodos exactos en [47] se propone un algoritmo de ramificación y corte para solucionar un OVRP.

De las actividades de entrega que se ajustan en este modelo, encontramos las rutas de distribución de diarios, la generación de rutas de entregas para aviones y las rutas escolares. Vale mencionar que en el problema que se aborda en este trabajo, los vehículos que componen la flota inician recorrido desde depósitos designados los cuales tienen una ubicación en dependencia de los lugares de residencia de los conductores.

1.3.2. Métodos para resolver problemas de planificación de rutas de vehículos

En el lenguaje coloquial, optimizar significa poco más que mejorar; sin embargo, en el contexto científico la optimización es el proceso de tratar de encontrar la mejor solución posible para un determinado problema. En un problema de optimización existen diferentes soluciones y un criterio para discriminar entre ellas. De forma más precisa, estos problemas se pueden expresar como encontrar el valor de unas variables de decisión para los que una determinada función objetivo alcanza su valor máximo o mínimo. El valor de las variables en ocasiones está sujeto a unas restricciones.

La idea intuitiva de problema difícil de resolver queda reflejada en el término científico NP-duro utilizado en el contexto de la complejidad algorítmica. En términos sencillos podemos decir que un problema de optimización duro es aquel para el que no podemos garantizar el encontrar la mejor solución posible en un tiempo razonable. La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, promovió el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas. Estos métodos, en los que la rapidez del proceso es tan importante como la calidad de la solución obtenida, se clasifican según [23] como se muestra en la Figura 1.2.

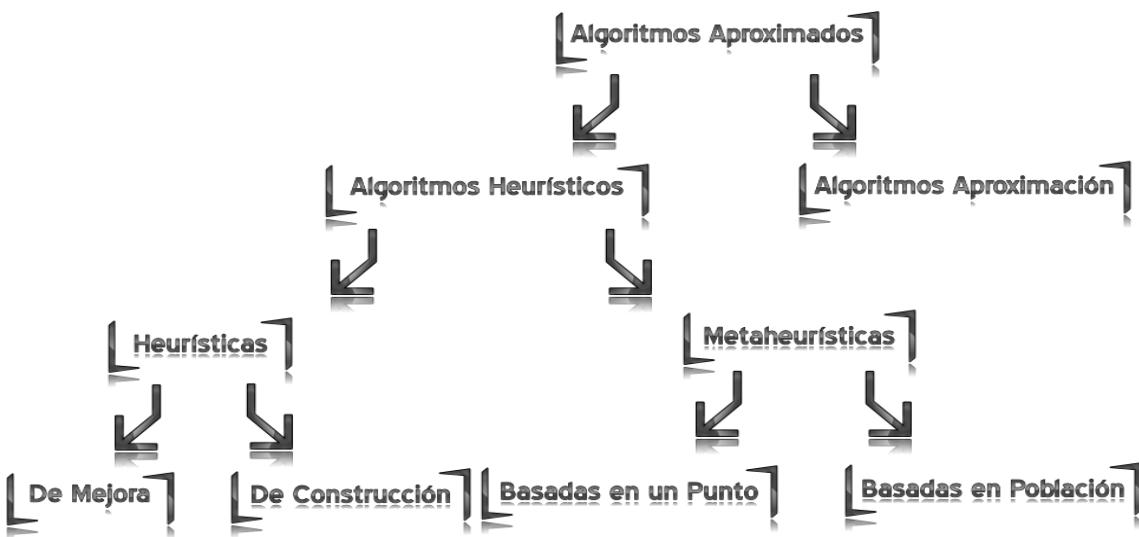


Figura 1.2: Clasificación de los algoritmos aproximados.

Los métodos aproximados se limitan a proporcionar una buena solución no necesariamente óptima, están compuestos fundamentalmente por dos subconjuntos de algoritmos que se pueden identificar como: algoritmos de aproximación y algoritmos heurísticos. A diferencia de los algoritmos heurísticos, que encuentran soluciones cercanas al óptimo en un tiempo polinomial, los algoritmos aproximados tienden a generar soluciones de poca calidad y en tiempos de ejecución poco razonable. Los algoritmos heurísticos se pueden clasificar en dos familias: heurísticas y metaheurísticas. En el caso de las heurísticas están hechas a medida y diseñadas para resolver un problema y/o instancia específica y las metaheurísticas son algoritmos de propósito general que se pueden utilizar para resolver cualquier problema de optimización [23].

Un método heurístico es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución [48]. La palabra heurística es un concepto que viene desde la Grecia clásica y proviene de la palabra griega *heuriskein* cuyo significado es encontrar o descubrir. Según la historia se deriva de *eureka*, famosa exclamación atribuida a Arquímedes. De esta forma podemos relacionar la heurística con la tarea de resolver problemas inteligentemente utilizando la información disponible [48]. Aplicado el término heurística en la Investigación Operacional, este toma un significado más exigente, para ilustrarlo se presenta la siguiente definición:

Se califica de heurístico a un procedimiento para el que se tiene un alto grado de confianza en que encuentra soluciones de alta calidad con un coste computacional razonable, aunque no se garantice su optimalidad o su factibilidad, e incluso, en algunos casos, no se llegue a establecer lo cerca que se está de dicha situación. Se usa el calificativo heurístico en contraposición a exacto [48].

La idea más genérica del término heurística está relacionada con la tarea de resolver inteligentemente problemas reales usando el conocimiento disponible [49]. Existen muchas heurísticas de naturaleza muy diferente, por lo que es complicado dar una clasificación general. Además, ellas son diseñadas para un problema en específico sin posibilidad de generalización o aplicación a problemas similares [50]. Algunos casos especiales de heurística son:

- *Heurísticas de construcción*: consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración [50].
- *Heurísticas de mejora*: parten de una solución y hacen intercambios de clientes en las rutas para mejorar las soluciones, es decir, reparan las rutas previamente construidas para conseguir una mejora [24].

Con frecuencia las heurísticas se basan en ideas bastante simples, de sentido común. Estas ideas deben ajustarse al problema específico de interés. Por lo general, cada método se diseña para abordar un tipo específico de problema en vez de una variedad de aplicaciones [51]. Durante muchos años esta situación derivó en que para desarrollar un método heurístico, un equipo de Investigación de Operaciones necesitaba comenzar desde cero. Luego, era necesario ajustarse al problema bajo consideración, siempre que no existiera un algoritmo disponible para encontrar una solución óptima. Este panorama ha cambiado en años relativamente recientes [51].

En la búsqueda de mejores soluciones y de mayor calidad, la investigación en este campo ha centrado su atención en el diseño de técnicas de propósito general para orientar la construcción de soluciones en las distintas heurísticas. Estas técnicas se llaman comúnmente metaheurísticas y son estrategias para diseñar y/o mejorar los procedimientos orientados a obtener un alto rendimiento [48].

Las heurísticas de construcción pueden usarse para ofrecer la solución final del problema o bien son aplicables dentro de un contexto de uso de las metaheurísticas. Un caso concreto es en la obtención de la solución inicial dentro de una metaheurísticas [51]. Los procedimientos constructivos empiezan desde una solución vacía. Son métodos donde la solución se crea seleccionando nodos sucesivamente con algún criterio hasta que construyen la solución factible. Las heurísticas de construcción se dividen en cuatro grupos Heurísticas basadas en ahorros, Heurísticas de inserción o basadas en costes, Métodos basados en ruta primero - asignar después y Métodos basados en asignar primero - ruta después. La mayoría de las heurísticas de construcción existentes están pensadas para la variante más simple de planificación de rutas de vehículos, es decir, CVRP. Por esta razón, es necesario realizar adaptaciones a estos métodos constructivos para su utilización en otras variantes.

1.3.2.1. Metaheurísticas

El término metaheurística fue introducido por Fred Glover en 1986 [52]. Las metaheurísticas son métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria. Estos algoritmos son procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explorar el espacio de búsqueda [50].

Las metaheurísticas son ampliamente reconocidas como una de las mejores aproximaciones para atacar los problemas de optimización combinatoria [48]. A diferencia de los métodos exactos las metaheurísticas permiten hacer frente a los problemas de gran tamaño mediante la búsqueda de soluciones satisfactorias en un plazo de tiempo razonable. La mayoría de las metaheurísticas tienen como objetivo los problemas de optimización combinatoria, pero se pueden aplicar a cualquier problema que se formule en términos heurísticos, por ejemplo, en resolución de ecuaciones *booleanas*. El uso de las metaheurísticas en muchas aplicaciones muestra su eficiencia y eficacia para resolver problemas grandes y complejos [52].

Los algoritmos metaheurísticos se pueden clasificar también según distintos criterios. Cada uno de estos criterios brinda una idea de la filosofía en que se basan cada uno de ellos [53]:

- *Inspiradas o no en la naturaleza:* Esta clasificación es intuitiva basada en los

orígenes y no es muy significativa, dada la dificultad en clasificar algunas metaheurísticas sobre todo nuevas metaheurísticas híbridas. Algunos ejemplos son los algoritmos evolutivos o genéticos [54], los sistemas artificiales inmuno-lógicos de la biología [55], las colonias de hormigas y abejas [50], la optimización de enjambre de partículas o inteligencia de enjambre en diferentes especies y el recocido simulado de la física [56].

- *Basadas en poblaciones o en una única solución:* algoritmos basados en una sola solución, también llamados basados en trayectorias (por ejemplo, Búsqueda Local (BL) [57], Recocido Simulado (RS) [56], Búsqueda Tabú (BT) [55, 58]), manipulan y transforman la solución describiendo una trayectoria en el espacio de soluciones durante el proceso de búsqueda. En el caso de los basados en poblaciones (por ejemplo, Enjambre de Partículas (EP) [56], Algoritmos Evolutivos (AE) [54]) toda una población de soluciones va evolucionando. Las metaheurísticas basadas en una sola solución están orientadas a la explotación, pero no tienen el poder de intensificar la búsqueda en determinadas regiones. Las basadas en poblaciones están orientadas a la exploración, que permiten una mayor diversificación en el espacio de soluciones.
- *Mono-objetivo o Multi-objetivo:* las metaheurísticas mono-objetivo son aquellas que buscan dar solución a una única función objetivo, mientras que las metaheurísticas multi-objetivo buscan dar solución a dos o más funciones objetivo. En este último caso no se obtiene una sola solución si no un conjunto de soluciones que en conjunto forman la solución y se les conoce como soluciones no dominadas del Frente de Pareto [58]. Entre los algoritmos mono-objetivos basados en un punto se encuentran: Escalador de Colinas (EC) [59], Búsqueda Aleatoria (BA) [58], entre otros y para los multi-objetivo están: Escalador de Colina Estocástico Multi-objetivo (ECMO) [60], Escalador de Colina Estocástico Multi-objetivo con Reinicio (ECMOR) [60], Escalador de Colina Estocástico Multi-objetivo por mayor Distancia (ECMOD) [60], Búsqueda Tabú Multi-objetivo (BTMO) [61], Recocido Simulado Multi-objetivo (RSMO) [61]. Algunos de los mono-objetivos basados en poblaciones de puntos son: Estrategias Evolutivas (EE) [62], entre otros, y de los multi-objetivos se tiene: *Non-dominated Sorting Genetic Algorithm II* (NSGAII por sus siglas en inglés) [63].

Algunas metaheurísticas mantienen en cada instante de ejecución un único estado actual, y lo cambian en cada iteración por uno nuevo. Este paso básico se conoce como transición de estado, movimiento o actualización del estado. El movimiento es hacia arriba o hacia abajo dependiendo de si los valores que da la función objetivo se incrementa o se decrementa. Alternativamente, el nuevo estado puede derivar del estado actual por un mutador proporcionado por el usuario; en este caso, el nuevo estado se conoce como vecino del estado actual. Generadores y mutadores son habitualmente procedimientos probabilísticos. El conjunto de todos los nuevos estados dados por el mutador es el vecindario del estado actual.

Las metaheurísticas más sofisticadas mantienen, en vez de un único estado actual, un conjunto de varios estados candidato. Así, el paso básico añade o elimina estados de este conjunto. En este caso, los procedimientos dados por el usuario seleccionan estados para ser descartados, y generan nuevos estados a añadir. El último estado puede ser generado como combinación o cruce de dos o más estados del conjunto.

Se puede percibir que la representación de la solución está estrechamente relacionada con los operadores de vecindad. Cuando se define una representación se debe tener en cuenta cómo será evaluada la solución y cómo los operadores funcionaran sobre esta. Por tanto, los operadores de vecindad juegan un papel fundamental dentro de los algoritmos metaheurísticos, debido a que son los responsables de definir los espacios de búsquedas de las soluciones. Los operadores de vecindad o de búsqueda son modificaciones que se realizan sobre una solución (unario) o varias soluciones (binario o n -arios) [64].

Una metaheurística puede guardar información del óptimo actual, escogiendo el estado óptimo entre todos los óptimos actuales obtenidos en varias etapas del algoritmo. Dado que el número de candidatos puede ser muy grande, normalmente, las metaheurísticas están diseñadas de manera que puedan ser interrumpidas por un tiempo máximo especificado por el usuario. En cambio, otras metaheurísticas dan sólo una garantía probabilística pobre de poder alcanzar el óptimo, de manera que cuando el tiempo máximo se aproxima a infinito, la probabilidad de examinar cada candidato tiende a 1. Este es un campo en investigación, con un gran número de publicaciones en revistas, un gran número de investigadores y usuarios, además de

un gran número de aplicaciones y bibliotecas que implementan este tipo de métodos.

1.3.2.2. Bibliotecas que implementan métodos heurísticos

El uso de una biblioteca de clases que implemente algoritmos heurísticos o metaheurísticos para la solución de problemas de optimización, resulta fundamental, en tanto garanticen la reutilización de los algoritmos ya existentes [65]. Principalmente el desarrollo de este tipo de bibliotecas se ha centrado en los algoritmos metaheurísticos, ya que son capaces de encontrar buenas soluciones (e incluso la solución óptima) con un tiempo y consumo de recursos razonables. Por esta razón, existen diferentes bibliotecas de clases que implementan algoritmos metaheurísticos de forma extensible. A continuación se describen brevemente algunas de ellas:

- **MALLBA**: biblioteca de clases desarrollada en el lenguaje C++, bajo la Licencia Pública General (GNU-GPL). Surge con el objetivo de proporcionar una biblioteca de esqueletos algorítmicos para optimización combinatoria (incluyendo métodos de resolución genérica, exacta, heurístico y métodos híbridos). En su diseño se contempló que para cada método de resolución existan implementaciones en secuencial y en paralelo [66].
- **METSlip**: esta biblioteca implementa algunos de los algoritmos metaheurísticos básicos como: Búsqueda Local Aleatoria [58], Búsqueda con Vecindad Variable [67], Búsqueda Local Iterada [58], Recocido Simulado (con enfriamiento lineal, exponencial o personalizado) [68], Búsqueda Tabú [55, 58]. Está diseñada para facilitar la implementación y adaptación de modelos sobre los que se pueden aplicar los algoritmos, los cuales una vez creados se les puede aplicar cualquiera de los algoritmos que tiene implementados [69].
- **MOMHLib++**: biblioteca de clases orientada a objeto. Esta biblioteca fue desarrollada con el lenguaje de programación C++ y publicada bajo Licencia Pública General. Su diseño está orientado a la implementación de metaheurísticas multi-objetivos y permite la incorporación de nuevas metaheurísticas [70].
- **Open Metaheuristics**: biblioteca de clases desarrollada con los lenguajes de programación C y C++ y publicada bajo Licencia Pública General. Uno de sus

principales objetivos es permitir pruebas experimentales a los algoritmos metaheurísticos a través de aproximación estadística. Además, se centra en la simplicidad por su forma para asimilar el diseño e implementar una metaheurística creando una interfaz propia. El diseño del código está separado en tres componentes: algoritmos, problemas y comunicación [71].

- **BiCIAM:** es una biblioteca de clases que implementa un modelo unificado de algoritmos metaheurísticos propuesto por [25]. Se implementó utilizando el lenguaje de programación Java. En sus inicios solamente contemplaba los algoritmos metaheurísticos basados en un punto como: Búsqueda Aleatoria [58], Escalador de Colinas Clásico [58], Búsqueda Tabú [55, 58] y Recocido Simulado [68]. Desde entonces se han agregado nuevos algoritmos como: Estrategias Evolutivas [56], Algoritmo Genético [61] y Algoritmo de Estimación de Distribuciones [72]. Luego en [73] se le incorporó un algoritmo que permitía la colaboración y competencia de los diferentes generadores metaheurísticos que implementa, llamado Multigenerador. Posteriormente se creó otra versión de BiCIAM [26] donde se realizaron cambios en la arquitectura y en las estructuras de datos utilizadas, y se incorporaron los algoritmos Escalador de Colinas con Reinicio [61] y Enjambre de Partículas (PSO, por sus siglas en inglés) [68]. La versión más reciente de BiCIAM realizada por [61] cuenta con la unificación de la versión mono-objetivo de [26] y la multi-objetivo de [74], las cuales fueron creadas en paralelo. Además de la incorporación del algoritmo multi-objetivo NSGAII [61].
- **Opt4J:** es una biblioteca de clases implementada en Java de código abierto para la computación evolutiva. Contiene un conjunto de algoritmos de optimización (multi-objetivo) tales como algoritmos evolutivos (incluyendo SPEA2 y NSGA2), evolución diferencial, optimización de enjambre de partículas y recocido simulado. Los puntos de referencia incluidos incluyen ZDT, DTLZ, WFG y el problema de la mochila. El objetivo de Opt4J es simplificar la optimización evolutiva de los problemas definidos por el usuario, así como la implementación de algoritmos arbitrarios de optimización metaheurística. Para ello, Opt4J se basa en una implementación basada en módulos y ofrece una interfaz gráfica de usuario para la configuración, así como una visualización del proceso de

optimización [75].

- **ECJ:** es un sistema de investigación implementado en Java. Fue diseñado para ser altamente flexible, con casi todas las clases (y todas sus configuraciones) determinada dinámicamente en tiempo de ejecución por un archivo de parámetros proporcionado por el usuario. Todas las estructuras del sistema están dispuestas para ser fácilmente modificables. El ECJ se desarrolla en el Laboratorio de Cálculo Evolutivo de ECLab de la Universidad George Mason y está diseñado para colaborar con MASON, un sistema multi-agente también desarrollado en este laboratorio.
- **JMetal:** es una biblioteca de algoritmos evolutivos, entre los que se encuentran implementaciones algoritmos genéticos simples, estrategias evolutivas, etc. A diferencia del MOE Framework, esta biblioteca no provee de mecanismos de ejecución paralela para los distintos algoritmos. Esta biblioteca está implementada utilizando el lenguaje Java. Aunque la misma no es orientada a algoritmos genéticos mono-objetivos tiene implementadas algunas metaheurísticas para ese tipo de algoritmos, así como distintas codificaciones de soluciones y operadores genéticos de cruzamiento, mutación, y selección de individuos [76].

En el **Apéndice B** se presenta una tabla con las comparaciones entre diferentes bibliotecas existentes en la literatura. Para la comparación se utilizaron los siguientes criterios: lenguaje de programación, algoritmos implementados, tipo de licencia y extensibilidad.

Para finalizar el análisis realizado a las bibliotecas antes mencionadas se puede concluir que:

- Existen bibliotecas implementadas en Java y otras en C++. Estas últimas tienen el inconveniente de que no pueden ser desplegadas en cualquier plataforma.
- Todas las bibliotecas presentan alguna licencia de software libre.
- En todas las bibliotecas se garantiza la extensibilidad.

Como resultado de lo expuesto anteriormente se puede decir que de las 8 bibliotecas que se compararon, 4 de estas constituyen propuestas validas a ser utilizadas

por estar implementada en el lenguaje de programación Java, lo que facilita su despliegue en cualquier plataforma. De estas 4 se puede utilizar 1 (BiCIAM) debido a que los VRPs utilizan permutaciones en su representación, por ende los algoritmos poblacionales tienden a destruir sus representaciones. Es por esta razón que en este trabajo se utilizan algoritmos basados en trayectoria. Se seleccionó la biblioteca de algoritmos metaheurísticos BiCIAM [25, 73] debido a que es una biblioteca de código abierto sin restricciones para su uso. Además, es una herramienta implementada en la facultad y ampliamente usada en proyectos del CITI por su flexibilidad. Por último, y no menos importante se tiene conocimiento profundo de su funcionamiento por los años de experiencia en el uso de la misma.

1.3.2.3. Biblioteca de algoritmos metaheurísticos: BiCIAM

BiCIAM es una biblioteca de clases que implementa un modelo unificado de algoritmos metaheurísticos. Su primera versión fue desarrollada en el 2009 [73] y disponía de los siguientes algoritmos basados en trayectoria: Búsqueda Aleatoria [58], Escalador de Colinas Clásico [59], Búsqueda Tabú [65, 77] y Recocido Simulado [59, 77]. Posteriormente, se han agregado nuevos algoritmos, en estas versiones basadas en poblaciones, como: Estrategias Evolutivas [62], Algoritmo Genético [78, 79] y Algoritmo de Estimación de Distribuciones [80]. Además, incorpora un algoritmo llamado Multigenerador que permite la colaboración y competencia de los diferentes generadores metaheurísticos que implementa la biblioteca [81]. BiCIAM propone una arquitectura basada en dos capas como se muestra en la Figura 1.3.

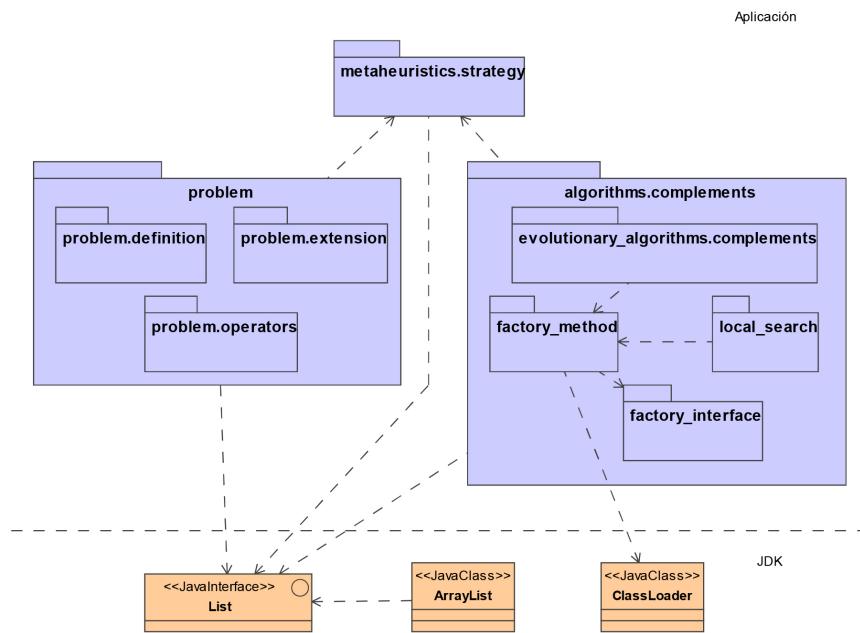


Figura 1.3: Diagrama de clases de BiCIAM [1].

En la capa JDK se encuentran las clases e interfaces que brinda la plataforma de Java. Por otra parte, la capa Aplicación contiene las clases y paquetes que manejan toda la lógica de la BiCIAM. Ejemplo de esto es el paquete **problem** que contiene las clases y paquetes que abstraen el problema del algoritmo, permitiendo que estas clases sean empleadas para definir cualquier problema. Otro paquete que compone la arquitectura de BiCIAM es **algorithms.complements** que agrupa las clases que son reutilizadas por otros paquetes y no son propiamente parte de la lógica del modelo, si no que garantizan su implementación. Por último, se encuentra el paquete **metaheuristics.strategy** que contiene la clase *Strategy* encargada de implementar el modelo unificado para llevar a cabo el proceso de búsqueda. Además, en este paquete se agrupan todos los generadores metaheurísticos implementados en BiCIAM.

1.3.3. Herramientas desarrolladas para la resolución de problemas de planificación de rutas de vehículos

Debido a la gran demanda y necesidad de resolver Problemas de Planificación de Rutas de Vehículos, se han desarrollado múltiples herramientas que brindan este servicio. La mayoría de estas herramientas aplican solamente una metaheurística y no especifican que variantes de problemas de planificación de rutas de vehículos pue-

de solucionar. Aunque abarcan un buen número de restricciones posibles. En este acápite se describen algunas de estas herramientas, sus características y funcionalidades.

ArcGIS Network Analyst: es una de las herramientas del Sistema de Información Geográfico ArcGIS. Esta herramienta proporciona análisis espacial basado en redes como las de reparto, logísticas, servicios más cercanos, áreas de atención de un servicio, entre otros. Utilizando esta extensión se puede modelar de forma muy dinámica y realista las condiciones de funcionamiento de cualquier red, incluso redes de tráfico, como calles de único sentido, restricciones de giros o altura, límites de velocidad, entre otras [82]. También se puede construir redes fácilmente a partir de datos geográficos utilizando un modelo de datos de red sofisticado.

CaSVRP: es una solución informática para resolver problemas de planificación de rutas de vehículos. CaSVRP es una capa de servicio que contiene diferentes variantes clásicas VRP (CVRP [4], MDVRP [18], VRPTW [14], HFVRP [19]) y fusiones de ellas. Está diseñada de forma robusta para permitir la incorporación de nuevas variantes. Esta solución consta de un módulo de optimización donde el problema de planificación de rutas de vehículos en sí es modelado como un problema de optimización. Para la resolución de estos problemas CaSVRP se apoya en la biblioteca de clases BiCIAM que le proporciona diferentes algoritmos metaheurísticos [83].

TransCAD: es un sistema de información geográfica (SIG) diseñado especialmente para profesionales de transporte con el objeto de almacenar, mostrar y analizar datos de transporte. En esta herramienta se combina las propiedades de un SIG y las capacidades de modelación del transporte, puede usarse para todos los modos de transporte y a cualquier escala geográfica o nivel de detalle. Además, contiene varios módulos que brindan disímiles servicios como: el análisis de transporte de pasajeros, entre otros [84].

TourSolver: es un software propietario de optimización de rutas orientado a la evaluación, planificación y optimización de planes de rutas. Entendiendo esto último como la planificación del orden de visita a los clientes para reducir al máximo los costos de operación de la empresa [85].

Después de haber analizado algunas de las herramientas que fueron diseñadas e implementadas para resolver Problemas de Planificación de Rutas de Vehículos

se puede decir que de algunas se desconocen las variantes que tuvieron en cuenta a la hora de proponer una solución. Por otro lado muchas de estas herramientas están pensadas para el transporte logístico y no para el transporte público y menos para el transporte obrero. Además, aunque abarcan un buen número de restricciones posibles son de software propietario.

1.4. Conclusiones parciales

En el presente capítulo se han expuesto los conceptos principales sobre el Problema del Transporte Obrero; así como el análisis de herramientas que resuelven este tipo de problemas. Se realizó un estudio de los Problemas de Planificación de Rutas de Vehículos y las variantes que por sus características se relacionan más con el Problema del Transporte Obrero problemas. Por último, se abordaron los métodos utilizados y herramientas desarrolladas para dar solución a los Problemas de Planificación de Rutas de Vehículos.

Al concluir la revisión y el análisis de la bibliografía se formulan las siguientes conclusiones:

- ✓ Los Problemas de Planificación de Rutas de Vehículos están presentes en muchos sectores de la sociedad por lo que la resolución de forma eficiente de estos problemas es de interés para la mejora de diversos servicios. Existen diferentes variantes VRP, pero muchas de estas variantes VRP por si solas no pueden resolver problemas de la vida real. Los Problemas de Planificación de Rutas de Vehículos son problemas de optimización combinatoria que pertenecen a la clase de problemas NP-duro y una alternativa para su resolución es mediante algoritmos heurísticos y metaheurísticos. Las metaheurísticas son algoritmos aproximados de optimización y búsqueda, que tiene un propósito general y son de gran utilidad para resolver problemas de optimización.
- ✓ El Problema del Transporte Obrero es una aplicación real de los Problemas de Planificación de Rutas de Vehículos, debido a las características que exhibe este problema se encuentran presentes muchas variantes de planificación de rutas de vehículos. Para dar solución a este problema se pueden utilizar cuatro variantes clásicas como son Problema de Planificación de Rutas de Vehículos

con Múltiples Depósitos, Problemas de Planificación de Rutas de Vehículos con Flota Heterogénea, Problemas de Planificación de Rutas de Vehículos Abierto y Problemas de Planificación de Rutas de Vehículos con restricciones de Distancia y Capacidad.

- ✓ Debido a la gran utilidad de los Problemas de Planificación de Rutas de Vehículos y sus aplicaciones en la vida real como es el caso del Problema del Transporte Obrero, existen diferentes herramientas para resolver este tipo de problemas. Un factor común es que todas estas herramientas emplean mapas para visualizar los recorridos. La mayoría de estas herramientas son propietarias y algunas no ejecutan algoritmos de optimización para la planificación de rutas y en otras se desconoce la utilización de estos algoritmos.
- ✓ Se decidió utilizar la biblioteca de algoritmos metaheurísticos BiCIAM debido a que los VRPs utilizan permutaciones en su representación y por ende los algoritmos poblacionales tienden a destruir sus representaciones. Es por esta razón que en este trabajo se utilizan algoritmos basados en trayectoria. Además, es una biblioteca de clases de última generación que implementa un modelo unificado de algoritmos metaheurísticos. La cual está realizada sobre el lenguaje de programación Java e implementada en la facultad y ampliamente usada en proyectos del CITI por su flexibilidad. Por último, y no menos importante se tiene conocimiento profundo de su funcionamiento por los años de experiencia en el uso de la misma.

Capítulo 2

Modelado y análisis del Problema del Transporte Obrero

2.1. Introducción

En este capítulo se aborda los aspectos relacionados con el modelado y análisis del Problema del Transporte Obrero. Mediante el uso de la herramienta de modelación MagicDraw y haciendo uso de los artefactos propuestos por RUP (por sus siglas en inglés de Rational Unified Process) se describe este problema. Se muestra el modelo del dominio y el diagrama de casos de uso del sistema y se describen los casos de uso mediante diagramas organizados por paquetes. Además, se comentan las reglas a tomar en cuenta para llevar a cabo el mismo. De igual forma se presentan los elementos necesarios para la modelación requerida, mostrando los paquetes y las relaciones entre ellos. Además de los requisitos funcionales y no funcionales necesarios para el desarrollo de la propuesta de solución de este trabajo.

2.2. Modelo del dominio

En el capítulo anterior se hace una descripción sobre los Problemas de Planificación de Rutas de Vehículos; así como de una de sus aplicaciones más comunes en la vida real: el Problema del Transporte Obrero. A continuación, se realiza una descripción del modelo del dominio asociado a esta problemática. El modelo del dominio es un artefacto que propone RUP para describir los conceptos fundamentales del negocio cuando ya los desarrolladores tienen conocimiento de ello [86]. En es-

te trabajo se decide realizar el modelo del dominio pues se desea desarrollar una herramienta informática para resolver problemas relacionados con el Problema del Transporte Obrero. Además, se dispone de un conocimiento previo debido a que ya se han realizado trabajos dentro del contexto de los Problemas de Planificación de Rutas de Vehículos, las restricciones y particularidades que este posee.

En la Figura 2.1 se muestra el modelo del dominio que expone los elementos asociados al Problema del Transporte Obrero y su representación como un problema de optimización.

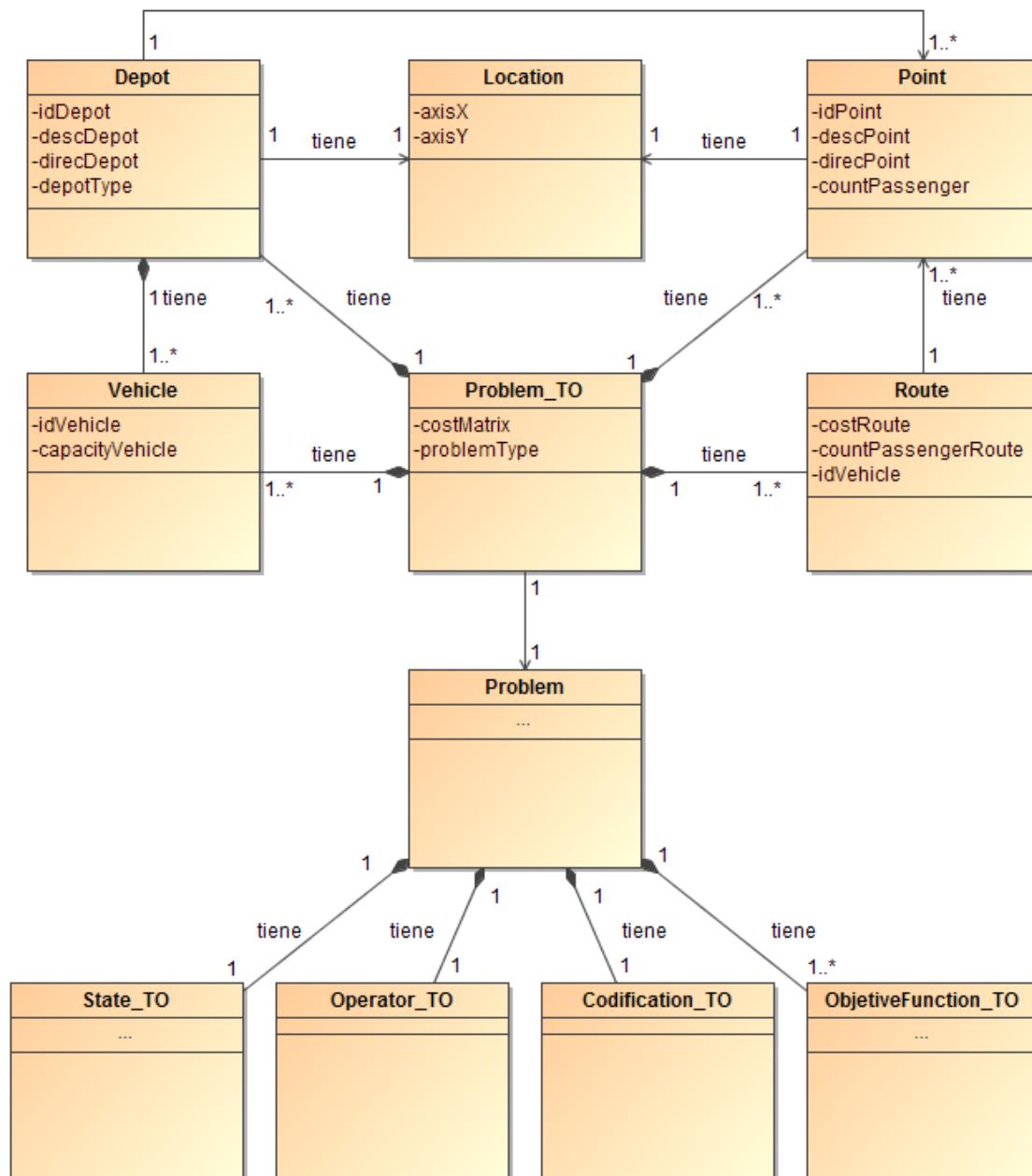


Figura 2.1: Modelo del dominio asociado al problema del Transporte Obrero.

A continuación se definen los elementos y conceptos principales:

- **Location:** representa la ubicación geográfica para un punto de recogida o para un depósito. Esta ubicación geográfica se interpreta como las coordenadas geográficas (x, y).
- **Vehicle:** representa el medio de transportación a partir de su identificador (matrícula del vehículo) y la capacidad de transportación del mismo.
- **Point:** representa un punto de concentración de trabajadores para su recogida en el servicio de transportación. De cada punto se conoce su identificador, una descripción, la dirección y la cantidad de trabajadores.
- **Depot:** representa un depósito, es decir el área de estacionamiento desde donde parten los vehículos de la flota. Esta entidad se conoce por su identificador, su localización, la descripción, la dirección y el tipo de depósito.
- **Route:** representa un recorrido de un vehículo en el problema del Transporte Obrero. Esta entidad se compone del listado de puntos que visita, el costo del recorrido, la cantidad de trabajadores a los que presta servicio y el vehículo con que se realiza dicho recorrido.
- **Problem_TO:** esta clase representa el problema a solucionar, mediante el conjunto de elementos que lo caracterizan. Contiene los depósitos, los puntos de recogida, los vehículos y las rutas. Además contiene el tipo de problema a resolver y una matriz de costo para almacenar la información de las distancias entre los puntos. Esta entidad está asociada a la clase Problem que equivale al Problema del Transporte Obrero tratado como un problema de optimización a la hora de utilizar los algoritmos metaheurísticos de BiCIAM.
- **Problem:** clase que representa un problema de optimización mediante el conjunto de elementos que lo caracterizan: la codificación, el estado, la función objetivo y el operador.
- **State_TO:** Clase que representa un estado o solución del problema de optimización.

- **ObjetiveFuntion_TO:** Clase que representa la función objetivo del problema de optimización.
- **Codification_TO:** Clase que representa la codificación de la solución del problema de optimización.
- **Operator_TO:** Clase que representa los operadores del problema de optimización.

Las entidades *Problem*, *Codification_TO*, *ObjetiveFuntion_TO*, *Operator_TO* y *State_TO* son clases necesarias para la comunicación con BiCIAM, las cuales deben ser redefinidas para cada problema de optimización a resolver.

2.2.1. Reglas generales del negocio

Las reglas del negocio describen las principales políticas, normas, y restricciones que deben estar presentes en un producto de software cumpliendo con las exigencias del usuario [87]. En el área en que está presente los Problemas de Planificación de Rutas de Vehículos y en específico las aplicaciones en la vida real como el Problema del Transporte Obrero, se consideran las reglas de negocio que a continuación se mencionan:

- En un recorrido cada punto solo puede ser visitado una vez.
- Cada vehículo utilizado en la transportación del personal (ómnibus) es asignado a un solo recorrido.
- A cada conductor se le puede asignar un solo vehículo.
- En todos los puntos se debe cumplir con la recogida de todos los trabajadores asignados a dicho punto.
- No pueden existir más recorridos que vehículos disponibles en la flota de vehículo.
- No se debe exceder la capacidad de los vehículos en cada recorrido.

Después de haber identificado las reglas que se tienen en cuenta para el Problema del Transporte Obrero. Se puede decir que el cumplimiento de cada una de ellas que en su conjunto dan lugar a que se obtenga una solución factible o lo que es igual que cumplan con todas las restricciones que se presentan para este tipo de problema.

2.3. Captura de requisitos

En la ingeniería de software, la captura de requisitos es la necesidad de documentar sobre el contenido, forma o funcionalidad de un producto o servicio. En la ingeniería clásica, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto [87]. Establecen qué debe hacer el sistema, pero no cómo hacerlo. La fase de captura y registro de requisitos puede estar precedida por una fase de análisis conceptual del proyecto, a continuación se describe la lista de requisitos candidatos.

2.3.1. Lista de requisitos candidatos

Los requisitos funcionales especifican acciones que el sistema debe ser capaz de ejecutar, sin que se tengan en consideración las restricciones físicas. Estos especifican las entradas y salidas del sistema [88]. En la Tabla 2.1 se muestra la relación de los Requisitos Candidatos asociados al software propuesto. La columna Estado indica si el requisito en cuestión va a ser considerado como parte del desarrollo de la herramienta, mientras que la tercera columna Prioridad muestra la importancia que tiene el requisito.

Tabla 2.1: Requisitos candidatos.

Requisito	Estado	Prioridad
Cargar desde diferentes fuentes de datos.	Aprobado	Importante
Insertar puntos de recogida, depósitos, vehículos.	Aprobado	Importante
Modificar puntos de recogida, depósitos, vehículos.	Aprobado	Importante
Eliminar puntos de recogida, depósitos, vehículos.	Aprobado	Importante
Seleccionar los datos del problema con los que el usuario desea trabajar.	Aprobado	Crítico
Calcular matriz de costo.	Aprobado	Crítico
Asignar trabajadores a puntos de recogida.	Propuesto	Secundario
Visualizar la asignación de trabajadores a puntos de recogida.	Propuesto	Secundario
Asignar vehículos a depósitos.	Aprobado	Crítico
Asignar puntos de recogida a depósitos.	Aprobado	Crítico
Visualizar la asignación de puntos a depósitos mediante el uso de mapas.	Aprobado	Importante
Salvar configuraciones en distintas etapas del proceso.	Aprobado	Importante
Obtener recorridos mediante algoritmos metaheurísticos.	Aprobado	Crítico
Visualizar los recorridos obtenidos mediante el uso de mapas.	Aprobado	Importante
Permitir configurar parámetros de ejecución a usuarios expertos.	Aprobado	Importante
Exportar recorridos en formato PDF.	Aprobado	Importante

2.3.2. Diagrama de casos de uso del sistema

Como resultado del análisis del modelo del dominio asociado al problema y de la captura de requisitos del sistema se diseña el diagrama de caso de uso del sistema que se presenta en la Figura 2.2. En este diagrama se muestra la relación entre los actores y los casos de uso del sistema.

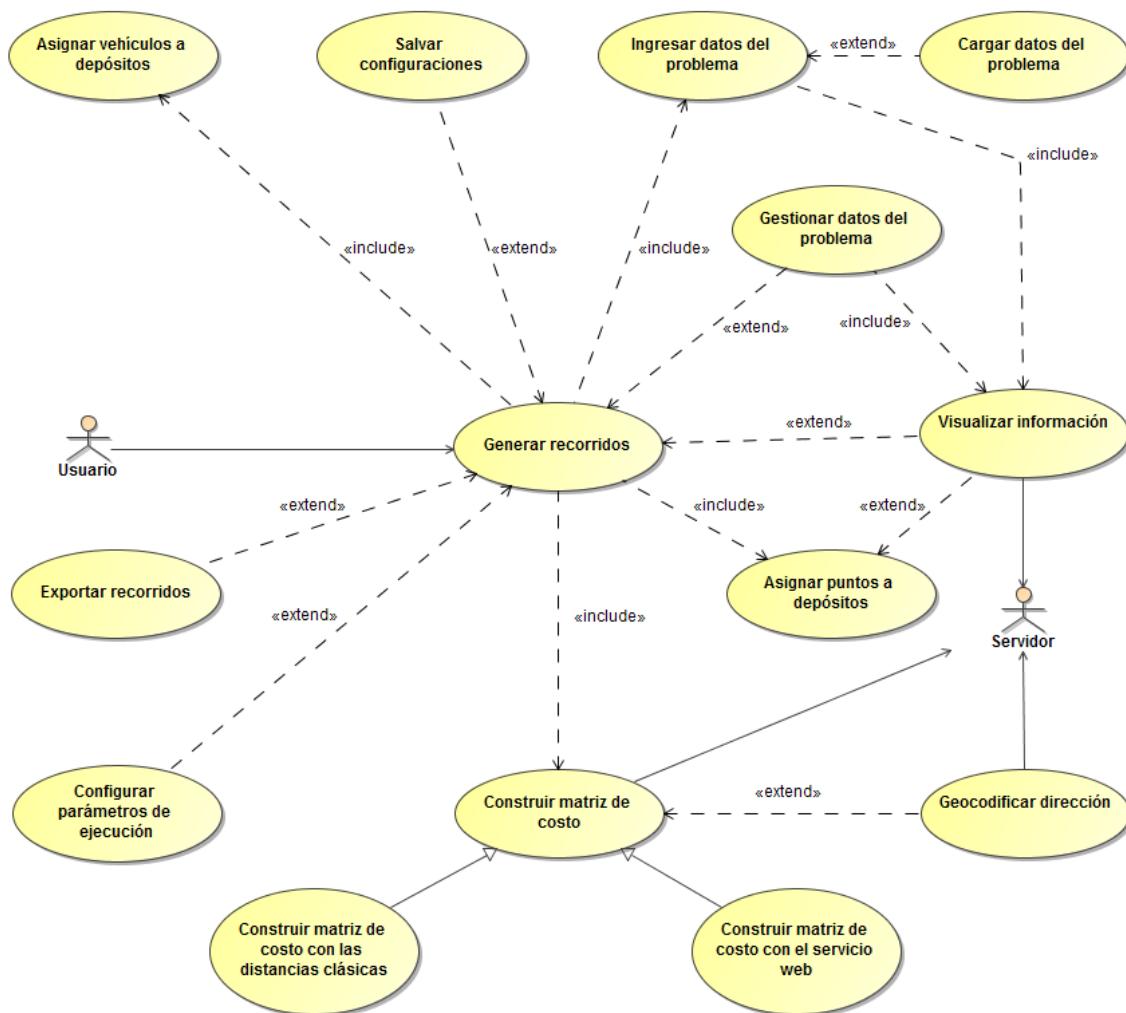


Figura 2.2: Diagrama de Caso de Uso del Sistema.

En el diagrama de caso de uso del sistema existen 14 casos de uso que agrupan a los requisitos candidatos aprobados para esta herramienta. El diagrama consta de 2 actores en el cual uno es el Usuario que se beneficia con la obtención de los recorridos en su Problema del Transporte Obrero y el otro es el Servidor de donde se consultan los servicios necesarios para dar cumplimiento al requisito de Construir matriz de costo y Visualizar información. Además el usuario es el que desencadena el trabajo con el CUS principal: Generar recorridos y de ahí se derivan los restantes

casos de uso del sistema a partir de relaciones de tipo **<include>** o **<extend>**.

2.3.3. Descripción de los actores del sistema

Un actor es una persona (usuario) o sistema que interactúa con el sistema en desarrollo, con el propósito de lograr un beneficio [89]. A continuación en la Tabla 2.2 se mencionan la descripción de los actores del sistema:

Tabla 2.2: Descripción de los actores del sistema.

Actor	Descripción
Usuario	Este usuario representa al directivo de la entidad que tenga la necesidad de generar los recorridos para el servicio de transporte obrero o al personal encargado de estas funciones como es el caso del jefe de transporte. Este usuario puede realizar en el sistema una gestión de los datos obteniendo soluciones al Problema del Transporte Obrero y visualizar en un mapa la información. Además, tendrá la facilidad de exportar en un formato PDF los resultados obtenidos por la herramienta.
Servidor	Es el servidor donde se encuentran alojados los servicios de geocodificación, matriz de costo y mapas que se utilizan para el trabajo con información geográfica real.

2.3.4. Descripción de los casos de uso del sistema

A continuación se presenta una descripción de los casos de uso del sistema diseñado, donde se establecen las condiciones que se deben cumplir para su ejecución y los resultados que se obtienen de su realización:

Tabla 2.3: Descripción del caso de uso Ingresar datos del problema.

Nombre del caso de uso	Ingresar datos del problema
Resumen	Este caso de uso es iniciado por el Usuario y consiste en la inserción de los datos del problema manualmente. El usuario tiene la facilidad de seleccionar en un mapa donde desea colocar los puntos de recogidas. Además tiene una relación de tipo <ex-tend> con el CUS Cargar datos del problema por si el Usuario decide utilizar los datos almacenados en determinadas fuente de datos.

Tabla 2.4: Descripción del caso de uso Cargar datos del problema.

Nombre del caso de uso	Cargar datos del problema
Precondiciones	Si la fuente de datos seleccionada para ingresar los datos del problema es un fichero debe cumplir con la estructura establecida y en el caso de una base de datos el esquema debe cumplir con el diseño definido.
Resumen	Este caso de uso consiste en la carga de los datos del problema que se desea resolver. Los datos a cargar pueden estar almacenados en un fichero texto, un estándar xml o en una base de datos. Además este CUS es el responsable de cargar las configuraciones guardadas previamente por el usuario para facilitar el trabajo con la herramienta.

Tabla 2.5: Descripción del caso de uso Generar recorridos.

Nombre del caso de uso	Generar recorridos
Precondiciones	Se requiere que se hayan ingresados los datos del problema. Además de cumplir con las restricciones que se han tenido en cuenta a la hora de la implementación de la herramienta.
Resumen	Este caso de uso es el más importante de la herramienta y es iniciado por el Usuario. Este caso de uso es el responsable de obtener los recorridos para el Problema del Transporte Obrero. En su funcionamiento se utilizan otros casos de uso del sistema que permiten la gestión de los datos del problema, la asignación de los recursos y la construcción de la matriz de costo.

Tabla 2.6: Descripción del caso de uso Geocodificar dirección.

Nombre del caso de uso	Geocodificar dirección
Resumen	Este caso consiste en consumir del Servidor el servicio web de geocodificación de direcciones para así obtener las coordenadas geográficas reales de los puntos de recogida y los depósitos para su posterior uso a la hora de ubicar en el mapa y para la construcción de la matriz de costo.
Requisitos especiales	Debe existir la biblioteca OpenLSCore para el tratamiento con la información devuelta por el servicio.

Tabla 2.7: Descripción del caso de uso Gestionar datos del problema.

Nombre del caso de uso	Gestionar datos del problema
Resumen	El CU consiste en la gestión de los datos del problema como sería la inserción, la modificación, la eliminación y la selección de los datos con los cuales el usuario desea trabajar.

Tabla 2.8: Descripción del caso de uso Asignar vehículos a depósitos.

Nombre del caso de uso	Asignar vehículos a depósitos
Resumen	Este caso de uso consiste en asignar en función del criterio del usuario los vehículos del problema a cada uno de los depósitos que equivale a decir a cada uno de los conductores.

Tabla 2.9: Descripción del caso de uso Construir matriz de costo.

Nombre del caso de uso	Construir matriz de costo
Resumen	Este CU consiste en calcular y construir la matriz de costo la cual se puede calcular mediante dos formas la primera mediante uno de los métodos de cálculo de distancia implementado en la herramienta o la segunda que es mediante un servicio web brindado por la infraestructura de datos espaciales publicada en el servidor del CITI la cual sería la más conveniente debido al trabajo con los sistemas espaciales. La relación con el CU Generar recorridos es de tipo <include> debido a que siempre es necesario calcular la matriz de costo. Y tiene una relación <extend> con el CU Geocodificar dirección debido a que no siempre es necesario geocodificar ya que las coordenadas pueden ser reales.

Tabla 2.10: Descripción del caso de uso Construir matriz de costo con las distancias clásicas.

Nombre del caso de uso	Construir matriz de costo con las distancias clásicas
Resumen	Este CU es el encargado de construir y calcular la matriz de costo mediante uno de los métodos de cálculo de distancias implementados en la herramienta, el cual hereda del CU Construir matriz de costo.

Tabla 2.11: Descripción del caso de uso Construir matriz de costo con el servicio web.

Nombre del caso de uso	Construir matriz de costo con el servicio web
Resumen	Este CU es el encargado de construir y calcular la matriz de costo mediante un servicio web brindado por la infraestructura de datos espaciales publicada en el servidor del CITI, el cual hereda del CU Construir matriz de costo.
Requisitos especiales	El servidor donde está publicado el servicio tiene que estar disponible.

Tabla 2.12: Descripción del caso de uso Asignar puntos a depósitos.

Nombre del caso de uso	Asignar puntos a depósitos
Resumen	Este CU consiste en asignar en función del criterio del usuario los puntos de recogida a los depósitos, esta asignación se puede realizar de dos formas la primera sería la asignación manual dependiendo de la conveniencia del usuario y la segunda mediante la utilización de los métodos de asignación implementados en la herramienta. La relación con el CU Generar recorridos es de tipo <include>debido a que siempre es necesario asignar los puntos a los depósitos.

Tabla 2.13: Descripción del caso de uso Exportar recorridos.

Nombre del caso de uso	Exportar recorridos
Precondiciones	Deben existir los recorridos que se desean exportar.
Resumen	Este caso de uso le brinda al usuario la posibilidad de exportar en formato pdf los recorridos generados para el Problema del Transporte Obrero.

Tabla 2.14: Descripción del caso de uso Visualizar información.

Nombre del caso de uso	Visualizar información
Resumen	Este caso de uso es el encargado de visualizar mediante mapas la información tanto de entrada como de salida del problema. En otras palabras las ubicaciones de los puntos de recogidas, de los depósitos, el resultado de la asignación de los puntos de recogidas a los depósitos y los recorridos obtenidos para el problema.
Requisitos especiales	Debe existir la biblioteca Geotools para el tratamiento con mapas.

Tabla 2.15: Descripción del caso de uso Salvar configuraciones.

Nombre del caso de uso	Salvar configuraciones
Resumen	Este CU es una opción que se le brinda al usuario para la salva en formato xml de la información seleccionada y transformada en determinadas fases del proceso de generación de recorridos para el Problema del Transporte Obrero. Estas salvas pueden ser utilizadas posteriormente para continuar el proceso desde la fase en que fue realizada.

Tabla 2.16: Descripción del caso de uso Configurar parámetros de ejecución.

Nombre del caso de uso	Configurar parámetros
Resumen	En este caso de uso el usuario experto en temas de optimización tendrá la posibilidad de configurar los parámetros de ejecución de los algoritmos de optimización que utiliza la herramienta para la generación de rutas en el Problema del Transporte Obrero.

2.4. Diagrama de paquetes y sus relaciones

Un diagrama de paquetes en el Lenguaje Unificado de Modelado representa las dependencias entre los paquetes que componen un modelo. Es decir, muestra cómo un sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones. Los paquetes están pensados para organizar buscando maximizar la coherencia interna dentro del paquete y minimizar el acoplamiento externo entre los paquetes. La Figura 2.3 muestra la propuesta de diagrama de paquetes y sus relaciones para la herramienta que resuelva el Problema del Transporte Obrero.

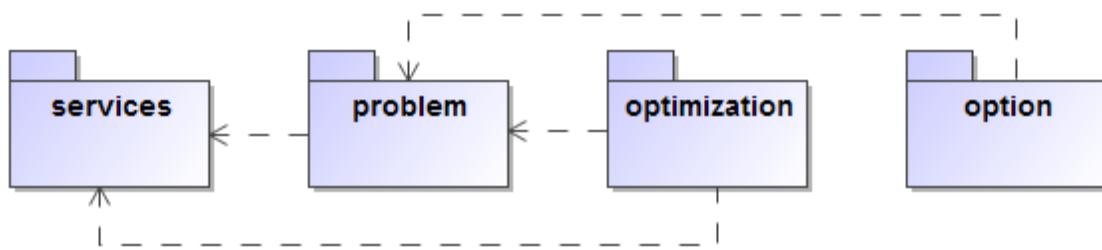


Figura 2.3: Diagrama de paquetes y sus relaciones.

A continuación se describen cada uno de los paquetes propuestos en el diagrama de la Figura 2.3:

- **problem:** es el responsable de modelar los datos y el comportamiento de un problema de planificación de rutas de vehículo.
- **services:** es el responsable de establecer la comunicación con los servicios web que se consumen por la herramienta. Todos estos servicios web son brindados por una infraestructura de datos espaciales publicada en el servidor del CITI.
- **optimization:** es el responsable de modelar el problema como un problema de optimización y obtener la solución mediante el uso de algoritmos metaheurísticos.
- **option:** es el responsable de encapsular la lógica para salvar las configuraciones en cada fase por donde transite el Usuario y de exportar los recorridos obtenidos.

2.5. Requisitos no funcionales

Un requisito no funcional es en la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento [86].

A continuación se mencionan los requisitos no funcionales que se consideraron para desarrollar la herramienta:

- Requerimientos de Software
 - 1. Para el despliegue y uso de la herramienta es necesario disponer de la máquina virtual de Java versión 8 o superior.
 - 2. La herramienta debe ser usada bajo una versión del sistema operativo Windows 7 o superior.
 - 3. Para el consumo de los servicios web que brinda la infraestructura de datos espaciales se requiere una conexión con el servidor CITI.
- Restricciones en el diseño y la implementación
 - 1. La herramienta debe ser desarrollada en el lenguaje de programación multiplataforma: Java.
 - 2. Se requiere el uso de una biblioteca que facilite el trabajo con mapas para la visualización de la información.
 - 3. Se requiere una biblioteca que facilite el trabajo con algoritmos metaheurísticos para la optimización de los recorridos que se desean obtener. Por el conocimiento que existe en su uso se exige utilizar BiCIAM.

2.6. Conclusiones parciales

En el presente capítulo se realiza una descripción del modelo del dominio asociado a la problemática. Además, se define mediante la captura de requisitos las condiciones necesarias que debe cumplir la herramienta a desarrollar. A partir de los requisitos identificados se determina un modelo de casos de uso del sistema a través del cual se describen las funcionalidades necesarias y las relaciones entre las

mismas. Por último, se describe el diagrama de paquetes y sus relaciones que se propone para la herramienta, que muestra cómo el sistema está dividido en agrupaciones lógicas y las dependencias entre esas agrupaciones.

En este capítulo se identificaron aquellos aspectos que podían ser utilizados para el desarrollo de la herramienta por tanto, se concluye que:

- ✓ La modelación del Problema del Transporte Obrero mediante el artefacto modelo del dominio con la descripción de las principales entidades y las reglas del negocio identificadas permiten obtener un modelo de la herramienta informática para la generación de rutas en este tipo de problema.
- ✓ El proceso de captura de requisitos permitió identificar 16 requisitos candidatos de los cuales 14 se van implementar en esta primera versión de la herramienta. Estos requisitos agrupados en casos de uso brindan las funcionalidades necesarias para en su conjunto resolver el Problema del Transporte Obrero.
- ✓ El diagrama de paquetes y sus relaciones que se propone para la herramienta agrupa por un criterio determinado como son elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos.
- ✓ Los requisitos no funcionales impuestos tanto de software, diseño y de implementación en su conjunto identifican las propiedades o cualidades que el producto debe tener para un desarrollo exitoso.

Capítulo 3

Herramienta para la generación de rutas en el Problema del Transporte Obrero

3.1. Introducción

En este capítulo se describe de manera detallada los procesos que intervienen en el desarrollo de una herramienta para la generación de rutas en el Problema del Transporte Obrero denominada **TransO** como propuesta de solución al Problema del Transporte Obrero esta herramienta facilita el entendimiento y aplicación de modelos de optimización a este problema de la vida real. Para la descripción de la propuesta se hace uso de los artefactos que propone RUP y se emplea la herramienta CASE MagicDraw para modelar cada uno de los artefactos. Por último, se describen las herramientas y tecnologías empleadas en la solución y se presenta un estudio de factibilidad del proyecto.

3.2. Vista de la arquitectura

En este epígrafe se presenta la vista de la arquitectura teniendo en cuenta la estructuración en capas de los paquetes y subsistemas. Para la descripción de los paquetes y subsistemas de la solución se presentan los diagramas de clases. El enfoque utilizado está basado en responsabilidad para mostrar los distintos niveles del trabajo siguiendo el principio de que las clases, subsistemas, paquetes que existen en

la aplicación, pueden ser ubicadas horizontalmente en un nivel específico de abstracción, junto a otras cuyas responsabilidades se encuentran al mismo nivel. Además de agrupan los elementos que tienen que ver con las interfaces de usuario que brinda el sistema, así como cierta lógica de la manipulación y generación de dichas interfaces de usuario, todo al mismo nivel. La vista de la arquitectura se encuentra proyectada como muestra la Figura 3.1 y se compone de tres capas:

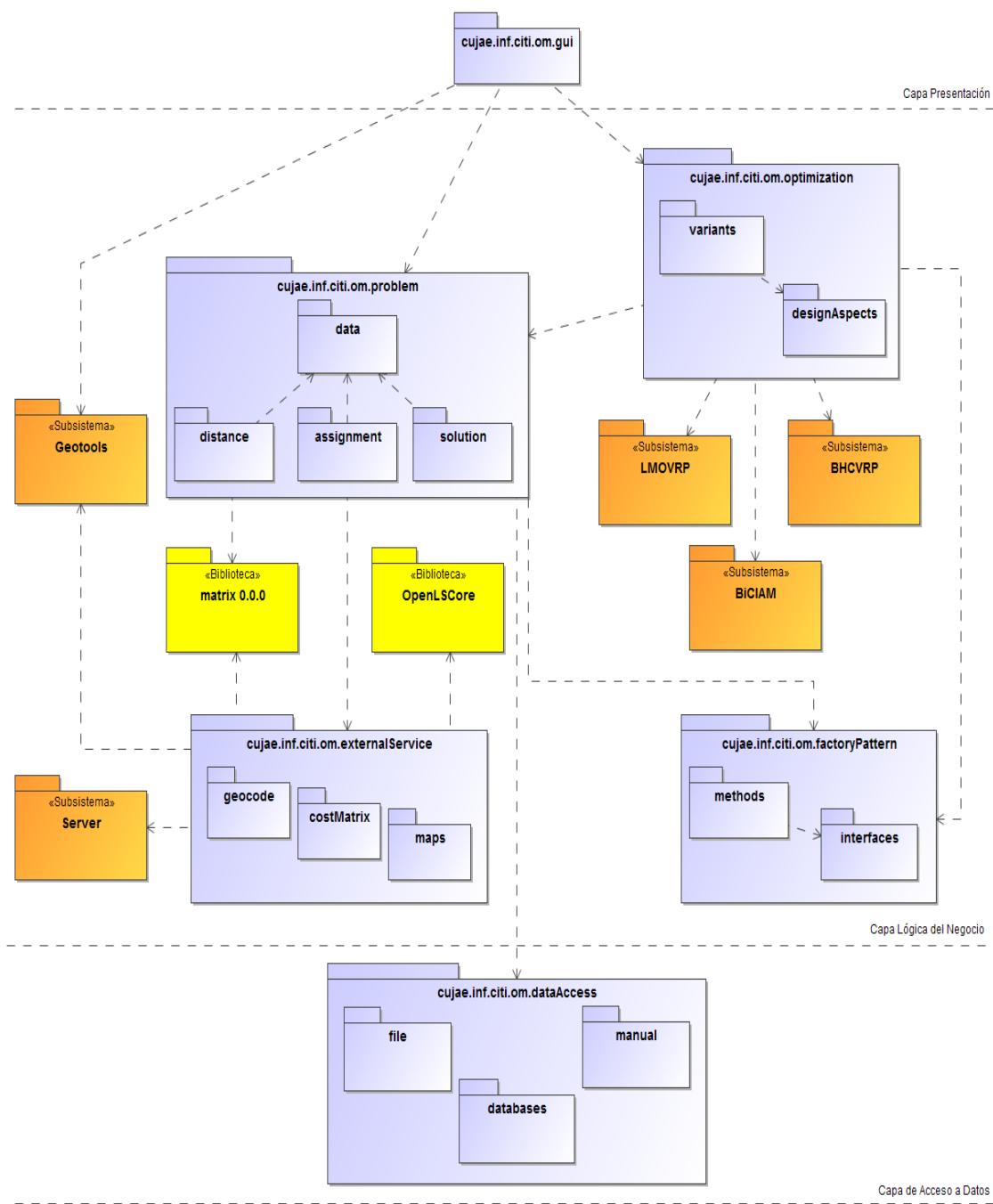


Figura 3.1: Diagrama de estructuración en capas lógicas.

Para un mejor entendimiento de la arquitectura propuesta se hace uso del color naranja para representar los subsistemas y el color amarillo para representar las bibliotecas de clases para el tratamiento de distintos tipos de datos que se requieren en la herramienta y el color azul para resaltar los demás paquetes. A continuación en los acápite siguientes se describen las capas que son propias del enfoque:

3.2.1. Capa Presentación

La **Capa Presentación** contiene el paquete **cujae.inf.citi.om.gui** que agrupa los elementos que tienen que ver con las interfaces de usuario, permitiendo la interacción del usuario con la herramienta. Como puede resultar obvio, incorpora además la lógica para la manipulación y generación de dichas interfaces visuales. **TransO** sigue un flujo secuencial donde el usuario transita desde la carga, edición y selección de los datos del problema, la asignación y configuración de los elementos involucrados en el proceso y la ejecución de los algoritmos. De manera general, las interfaces gráficas de **TransO** se diseñaron de manera intuitiva, de forma tal, que el usuario en todo momento conozca las acciones que puede y debe realizar. En el **Apéndice C** se muestran las interfaces gráficas que se emplean en cada paso a realizar.

3.2.2. Capa Lógica del Negocio

La **Capa Lógica del Negocio** contiene la mayoría de los paquetes, debido a que es la capa responsable de modelar el Problema del Transporte Obrero y las correspondientes variantes de optimización de este problema. Dentro de esta capa se ubican los subsistemas **BiCIAM** [1], **BHCVRP** [90], **LMOVRP** [64], **Geotools** [2, 91] y **Server** que no son propios de **TransO**, pero es de vital importancia para su funcionamiento. En el acápite 3.3: "Diseño de subsistema" se explica cada uno de ellos. Además, esta capa contiene las bibliotecas **OpenLSCore** y **matrix** en su versión 0.0.0. Ambas bibliotecas proporcionan un conjunto de clases necesarias para el funcionamiento de **TransO**. En el caso de **OpenLSCore** sus clases son utilizadas en el paquete **cujae.inf.citi.om.externalService.geocode** que es responsable de consumir el servicio de Geocodificación que brinda la infraestructura de datos espaciales publicada en el servidor del CITI. Mientras que **matrix** encapsula las clases para el manejo de una matriz utilizada en el paquete **cujae.inf.citi.om.problem.data** y

en el servicio para la conformación de la matriz de costo que contiene el paquete **cujae.inf.citi.om.externalService.costMatrix**. Para cada uno de los paquetes contenidos en esta capa se presenta su diagrama UML y una breve descripción de los elementos participantes.

3.2.2.1. Descripción del paquete cujae.inf.citi.om.problem

El paquete **cujae.inf.citi.om.problem** es el responsable de modelar los datos y el comportamiento de un problema de planificación de rutas de vehículo. Este paquete agrupa a los siguientes paquetes:

- **cujae.inf.citi.om.problem.data**: contiene las clases que modelan un problema de planificación de rutas de vehículos.
- **cujae.inf.citi.om.problem.solution**: contiene las clases que modelan la solución de un problema de planificación de rutas de vehículos.
- **cujae.inf.citi.om.problem.assignment**: contiene las clases que implementan los métodos para la asignación de puntos de recogidas a depósitos, necesarios en la variante donde se consideran múltiples depósitos.
- **cujae.inf.citi.om.problem.distance**: contiene las clases que implementan diferentes métodos para el cálculo de distancia entre dos puntos, necesarios para conformar la matriz de costos.

En los diagramas de clases que se presentan en las Figuras 3.2 - 3.4 se muestran cada una de las vistas del paquete **cujae.inf.citi.om.problem** y la relación que se establece entre los elementos de cada uno de los paquetes que contiene.

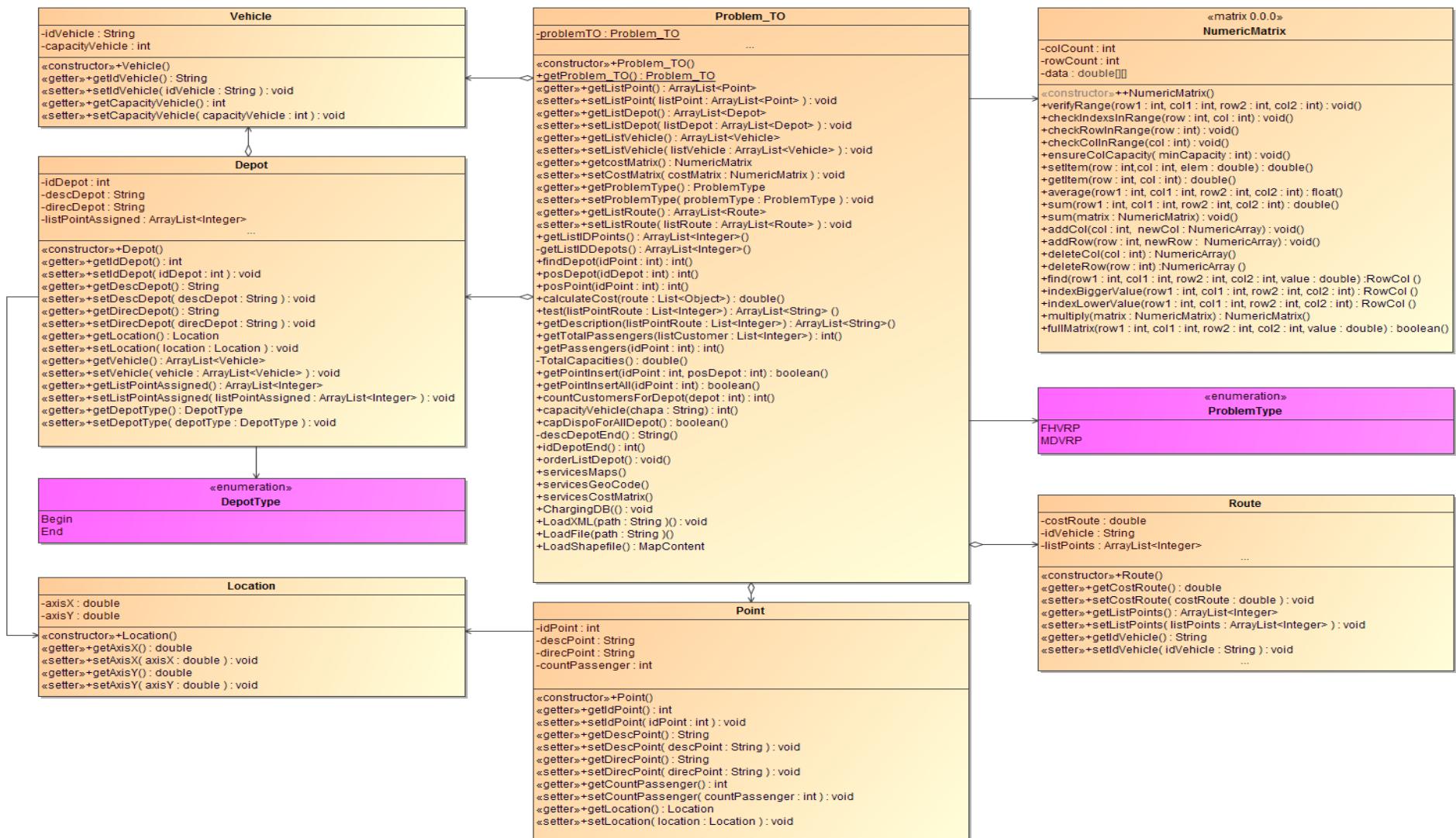


Figura 3.2: Diagrama de clases de los paquetes cujae.inf.citi.om.problem.data y cujae.inf.citi.om.problem.solution.

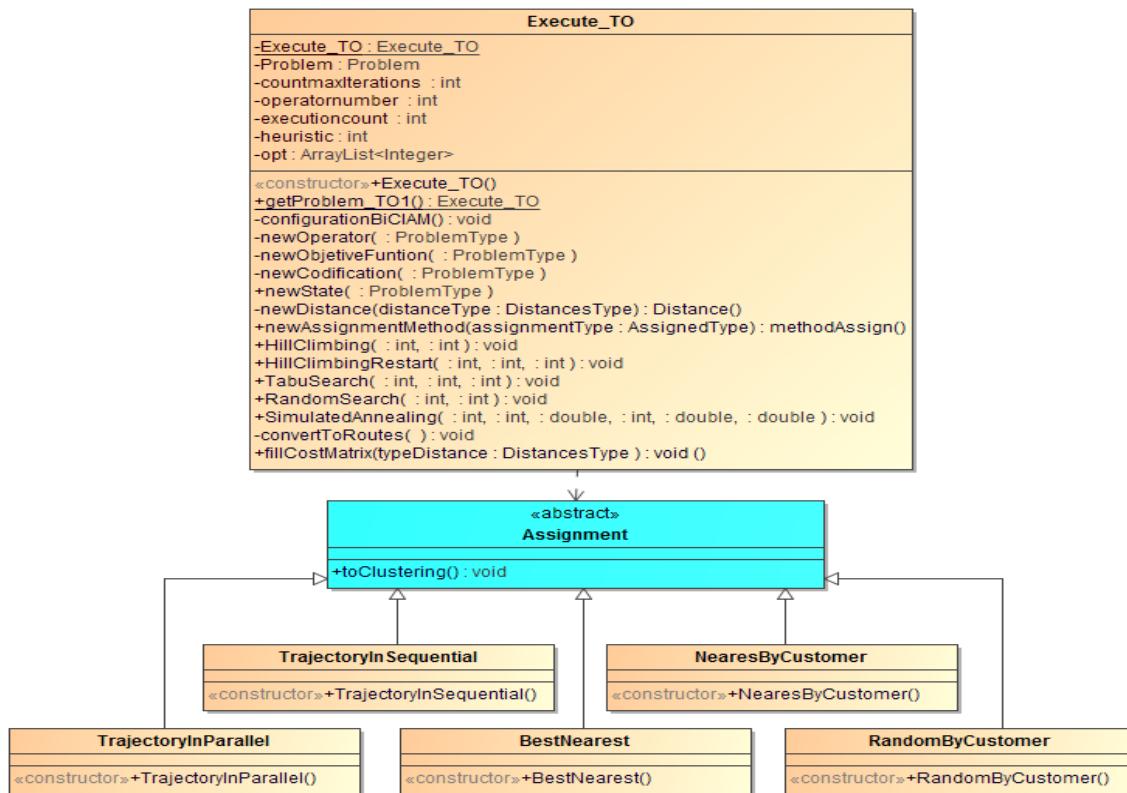


Figura 3.3: Diagrama de clases del paquete cujae.inf.citi.om.problem.assignment.

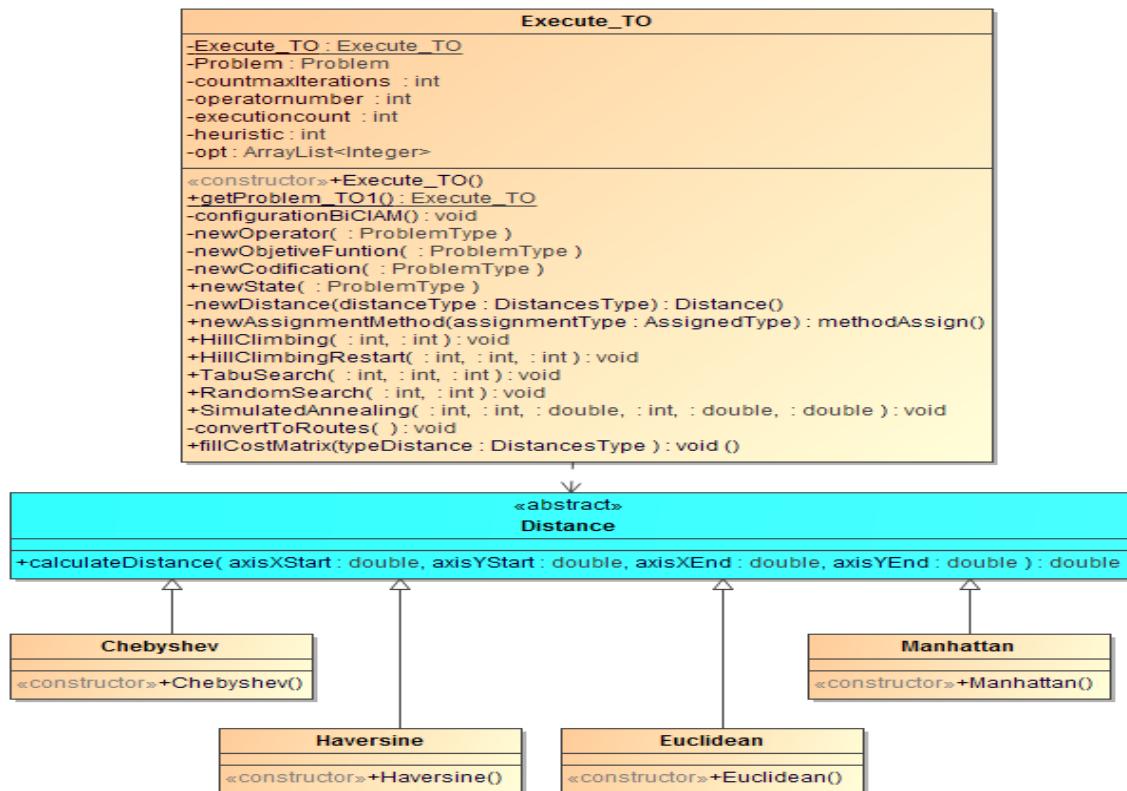


Figura 3.4: Diagrama de clases del paquete cujae.inf.citi.om.problem.distance.

Los elementos participantes en cada uno de los artefactos presentados se describen en las Tabla 3.1 - Tabla 3.3:

Tabla 3.1: Descripción del paquete cujae.inf.citi.om.problem.

Clases	Descripción
ProblemType	Enumerado que indica las variantes de planificación de rutas de vehículos a resolver para el Problema del Transporte Obrero.
Point	Clase que modela los datos de un punto de recogida.
Depot	Clase que modela los datos de un depósito.
DepotType	Enumerado que indica si el depósito es considerado de origen o destino.
Vehicle	Clase que modela los datos de un vehículo.
Location	Clase que modela la ubicación geográfica de un punto de recogida o de un depósito.
Problem_TO	Clase que controla el problema a solucionar mediante el conjunto de elementos que lo caracterizan. Además, esta clase establece una relación con la biblioteca matrix en su versión 0.0.0 que permite el trabajo con las clases que modelan una matriz de costo.
Route	Clase que modela los datos de una ruta de vehículos.

Tabla 3.2: Descripción del paquete cujae.inf.citi.om.problem.assignment.

Clases	Descripción
Assignment	Clase abstracta que define como asignar los puntos de recogidas a depósitos.
BestNearest	Esta clase modela como asignar los puntos de recogidas a los depósitos considerando la menor distancia al depósito que corresponda siempre y cuando no se incumpla la restricción de capacidad.
NearestByCustomer	Esta clase modela como asignar los puntos de recogidas a los depósitos considerando los puntos de recogidas más cercanos a él siempre y cuando no se incumpla la restricción de capacidad.
RandomByCustomer	Esta clase modela como asignar los puntos de recogidas a los depósitos de forma aleatoria siempre y cuando no se incumpla la restricción de capacidad.
TrajectoryInParallel	Esta clase modela como asignar los puntos de recogidas a los depósitos considerando el mejor punto de recogida que sea más cercano al último punto insertado siempre y cuando no se incumpla la restricción de capacidad.
TrajectoryInSequential	Esta clase modela como asignar los puntos de recogidas a los depósitos considerando todos los puntos de recogida que le corresponde, para ello se selecciona el mejor punto de recogida que sea más cercano al último punto insertado en ese depósito siempre y cuando no se incumpla la restricción de capacidad.

Tabla 3.3: Descripción del paquete cujae.inf.citi.om.problem.distance.

Clases	Descripción
Distance	Clase abstracta que define como calcular la distancia entre dos puntos.
Euclidean	Esta clase modela como calcular la distancia entre dos puntos siguiendo la distancia Euclidean, es la distancia ordinaria (que se mediría con una regla) entre dos puntos de un espacio, la cual se deduce a partir del teorema de Pitágoras.
Manhattan	Esta clase modela como calcular la distancia entre dos puntos siguiendo la distancia Manhattan, es la distancia entre dos puntos medida sobre ejes a ángulos rectos, es decir; la distancia que se recorrería para llegar de un punto a otro si se siguiera una trayectoria de rejilla (o de cuadrícula).
Chebyshev	Esta clase modela como calcular la distancia entre dos puntos siguiendo la distancia Chebyshev, es la distancia entre dos puntos medida en un espacio vectorial donde la distancia es la mayor de sus diferencias a lo largo de cualquiera de sus dimensiones o coordenadas.
Haversine	Esta clase modela como calcular la distancia entre dos puntos siguiendo la distancia Haversine, es la distancia de círculo máximo entre dos puntos de un globo sabiendo su longitud y su latitud. Es un caso especial de una fórmula más general de trigonometría esférica.

3.2.2.2. Descripción del paquete cujae.inf.citi.om.optimization

El paquete **cujae.inf.citi.om.optimization** es el responsable de modelar las variantes de planificación de rutas de vehículos para el Problema del Transporte Obrero y obtener la solución mediante el uso de algoritmos metaheurísticos. Este paquete agrupa a los siguientes paquetes:

- **cujae.inf.citi.om.optimization.designAspects:** contiene las clases que se comunican con **BiCIAM** [1] para definir en un problema de planificación de rutas de vehículos mediante los distintos aspectos del diseño de un algoritmo metaheurístico como la codificación, la función objetivo, el estado y los operadores del problema con los cuales se trabaja.

- **cujae.inf.citi.om.optimization.variants:** contiene las clases que implementan los distintos aspectos del diseño de un algoritmo metaheurístico según las características consideradas en el problema de planificación de rutas de vehículos incluidas en la herramienta.

La Figura 3.5 muestra un diagrama con la relación que se establece entre los elementos de ambos paquetes.

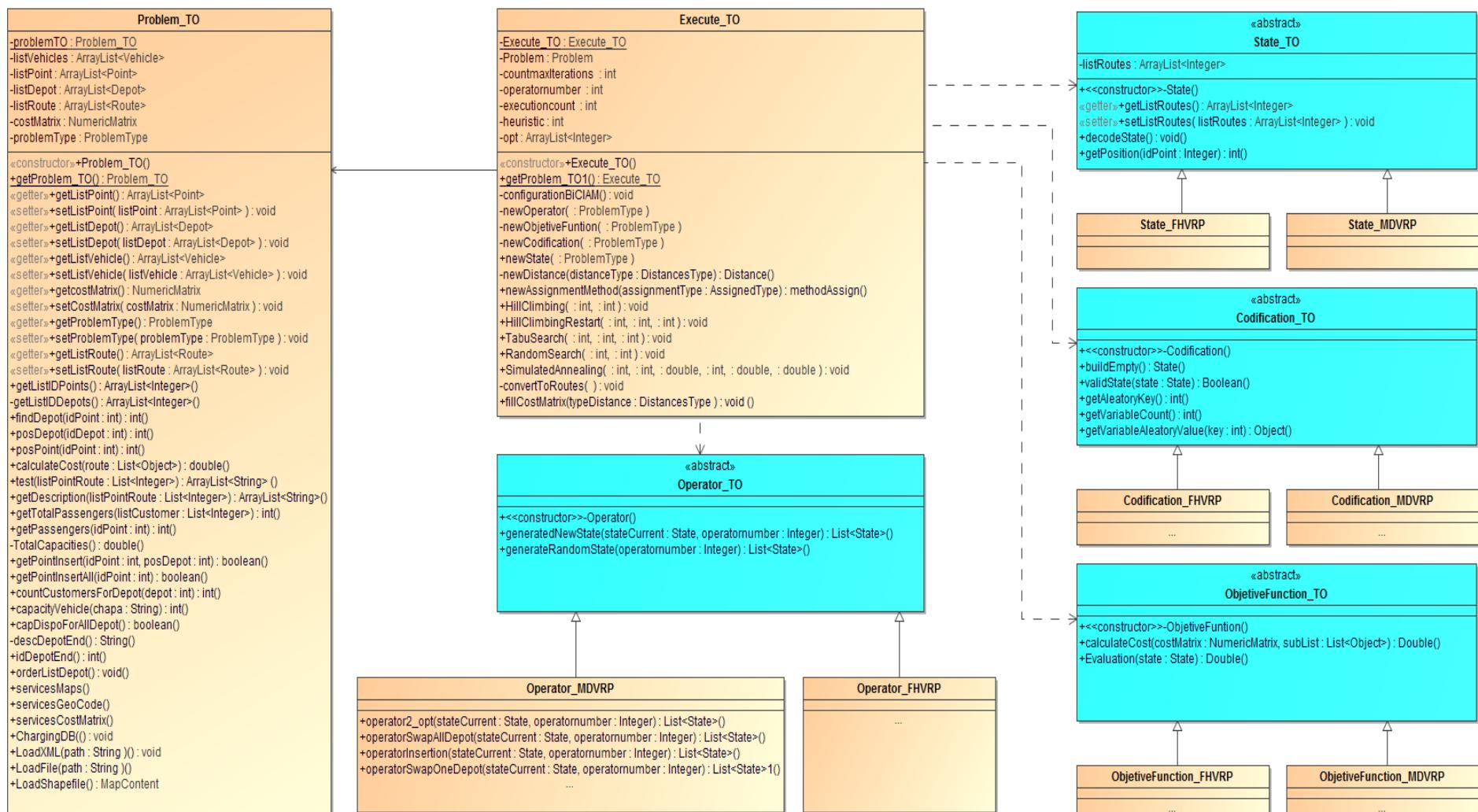


Figura 3.5: Diagrama de clases del paquete cujae.inf.citi.om.optimization.

Los elementos participantes en este artefacto se describen a continuación en las Tabla 3.4 y Tabla 3.5:

Tabla 3.4: Descripción del paquete cujae.inf.citi.om.optimization.designAspects.

Clases	Descripción
State_TO	Clase abstracta que hereda de la clase <i>State</i> de BiCIAM para definir el comportamiento de un estado de solución en el Problema del Transporte Obrero.
Codification_TO	Clase abstracta que hereda de la clase <i>Codification</i> de BiCIAM para definir el comportamiento de la codificación de un estado de solución en el Problema del Transporte Obrero.
Operator_TO	Clase abstracta que hereda de la clase <i>Operator</i> de BiCIAM para definir como construir la solución inicial y como generar una nueva solución en el Problema del Transporte Obrero.
ObjetiveFunction_TO	Clase abstracta que hereda de la clase <i>Objetive-Function</i> de BiCIAM para definir como evaluar una solución en el Problema del Transporte Obrero.
Execute_TO	Clase que controla el proceso de optimización para el Problema del Transporte Obrero.

Tabla 3.5: Descripción del paquete cujae.inf.citi.om.optimization.variants.

Clases	Descripción
State_MDVRP	Clase que hereda de <i>State_TO</i> y redefine el comportamiento de un estado de solución cuando en el Problema del Transporte Obrero se considera entre sus principales características los múltiples depósitos.
State_FHVRP	Clase que hereda de <i>State_TO</i> y redefine el comportamiento de un estado de solución cuando en el Problema del Transporte Obrero se considera entre sus principales características donde la flota de vehículo es heterogénea.
Codification_MDVRP	Clase que hereda de <i>Codification_TO</i> y redefine el comportamiento de la codificación de un estado de solución cuando en el Problema del Transporte Obrero se considera entre sus principales características los múltiples depósitos.
Codification_FHVRP	Clase que hereda de <i>Codification_TO</i> y redefine el comportamiento de la codificación de un estado de solución cuando en el Problema del Transporte Obrero se considera entre sus principales características donde la flota de vehículo es heterogénea.
Operator_MDVRP	Clase que hereda de <i>Operator_TO</i> e implementa los operadores para construir la solución inicial y para generar una nueva solución cuando en el Problema del Transporte Obrero se considera entre sus principales características los múltiples depósitos.
Operator_FHVRP	Clase que hereda de <i>Operator_TO</i> e implementa los operadores para construir la solución inicial y para generar una nueva solución cuando en el Problema del Transporte Obrero se considera entre sus principales características donde la flota de vehículo es heterogénea.
ObjetiveFunction_MDVRP	Clase que hereda de <i>ObjetiveFunction_TO</i> y redefine como evaluar una solución cuando en el Problema del Transporte Obrero se considera entre sus principales características los múltiples depósitos.
ObjetiveFunction_FHVRP	Clase que hereda de <i>ObjetiveFunction_TO</i> y redefine como evaluar una solución cuando en el Problema del Transporte Obrero se considera entre sus principales características donde la flota de vehículo es heterogénea.

3.2.2.3. Descripción del paquete cujae.inf.citi.om.externalService

Otro paquete relevante para la arquitectura es **cujae.inf.citi.om.externalService** que contiene los paquetes donde se establece la comunicación con los servicios web que son consumidos por **TransO**. Todos estos servicios web son brindados por la infraestructura de datos espaciales disponibles en el servidor del CITI. Los paquetes son:

- **cujae.inf.citi.om.externalService.geocode**: contiene las clases necesarias para el consumo del servicio web de Geocodificación.
- **cujae.inf.citi.om.externalService.costMatrix**: contiene las clases necesarias para el consumo del servicio web de Conformación de la Matriz de Costo.
- **cujae.inf.citi.om.externalService.maps**: contiene las clases necesarias para el consumo del servicio web de Mapas.

La Figura 3.6 muestra un diagrama con la relación que se establece entre los elementos de ambos paquetes.

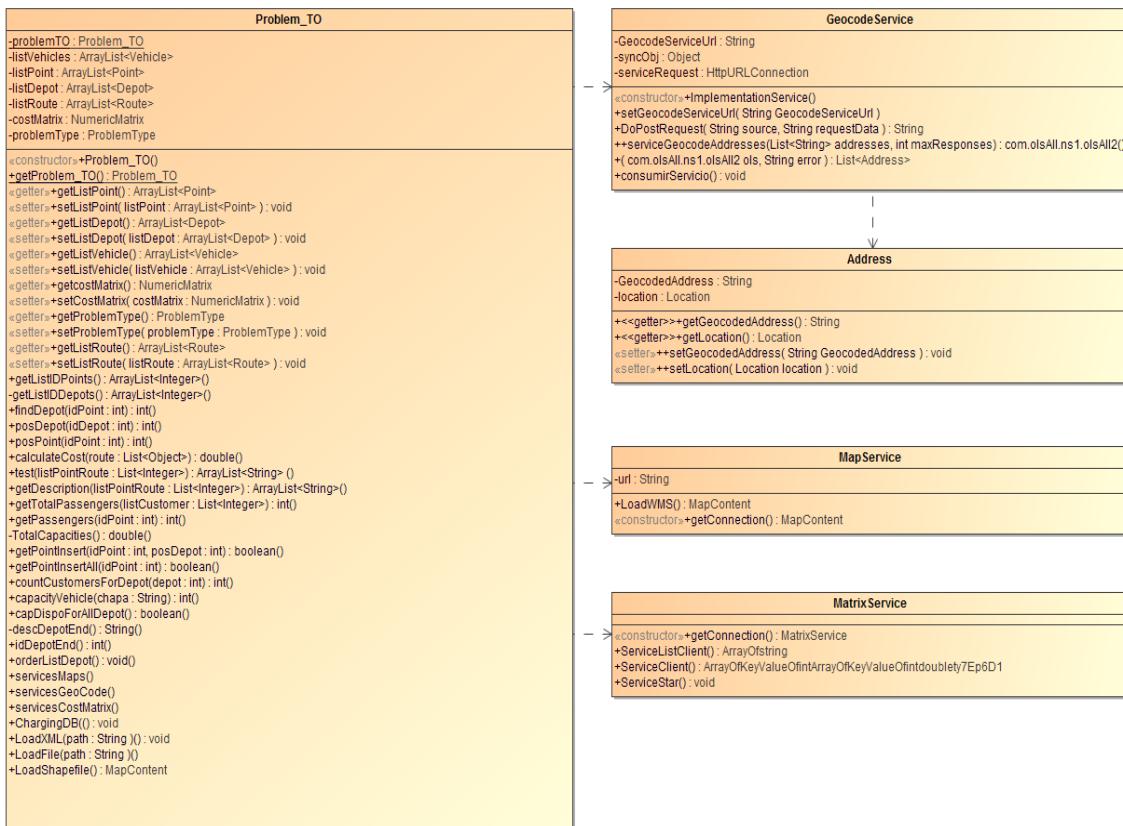


Figura 3.6: Diagrama de clases del paquete cujae.inf.citi.om.externalService.

Los elementos participantes en este artefacto se enumeran en la Tabla 3.6:

Tabla 3.6: Descripción del paquete cujae.inf.citi.om.externalService.

Clases	Descripción
Address	Clase que modela la información de una dirección requerida en el servicio con una estructura determinada y que se requiere para realizar la geocodificación.
GeocodeService	Clase que permite consumir el servicio web Geocodificación. Este servicio web está definido bajo el estándar OpenLS por lo que se incluye en el proyecto la biblioteca OpenLSCore para facilitar la comunicación.
MatrixService	Clase que permite consumir el servicio web Conformación de matriz de costos y establecer la comunicación con el mismo a través de los parámetros que se envían y de los resultados que se obtienen. Además, esta clase establece una relación con la biblioteca matrix en su versión 0.0.0 que permite el trabajo con las clases que modelan una matriz de costo.
MapService	Clase que permite consumir el servicio web de mapas y establecer la comunicación con el mismo.

3.2.2.4. Descripción del paquete cujae.inf.citi.om.factoryPattern

Por último la **Capa Lógica del Negocio** contiene uno de los paquetes fundamentales para la herramienta como es el paquete **cujae.inf.citi.om.factoryPattern** para garantizar la implementación de la lógica de negocio. Este paquete se compone de dos paquetes:

- **cujae.inf.citi.om.factoryPattern.interfaces**: contiene las interfaces y los enumerados para la creación de métodos para el cálculo de distancias, métodos de asignación de puntos de recogidas a depósitos, así como para la creación de los distintos aspectos de diseño a considerar según la variante de planificación de ruta a resolver.
- **cujae.inf.citi.om.factoryPattern.methods**: contiene las clases que implementan los métodos para la creación de métodos para el cálculo de distancias,

métodos de asignación de puntos de recogidas a depósitos, así como para la creación de los distintos aspectos de diseño a considerar según la variante de planificación de ruta a resolver declarados en las interfaces del paquete **cujae.inf.citi.om.factoryPattern.interfaces**.

La Figura 3.7 muestra un diagrama con la relación que se establece entre los elementos de ambos paquetes.

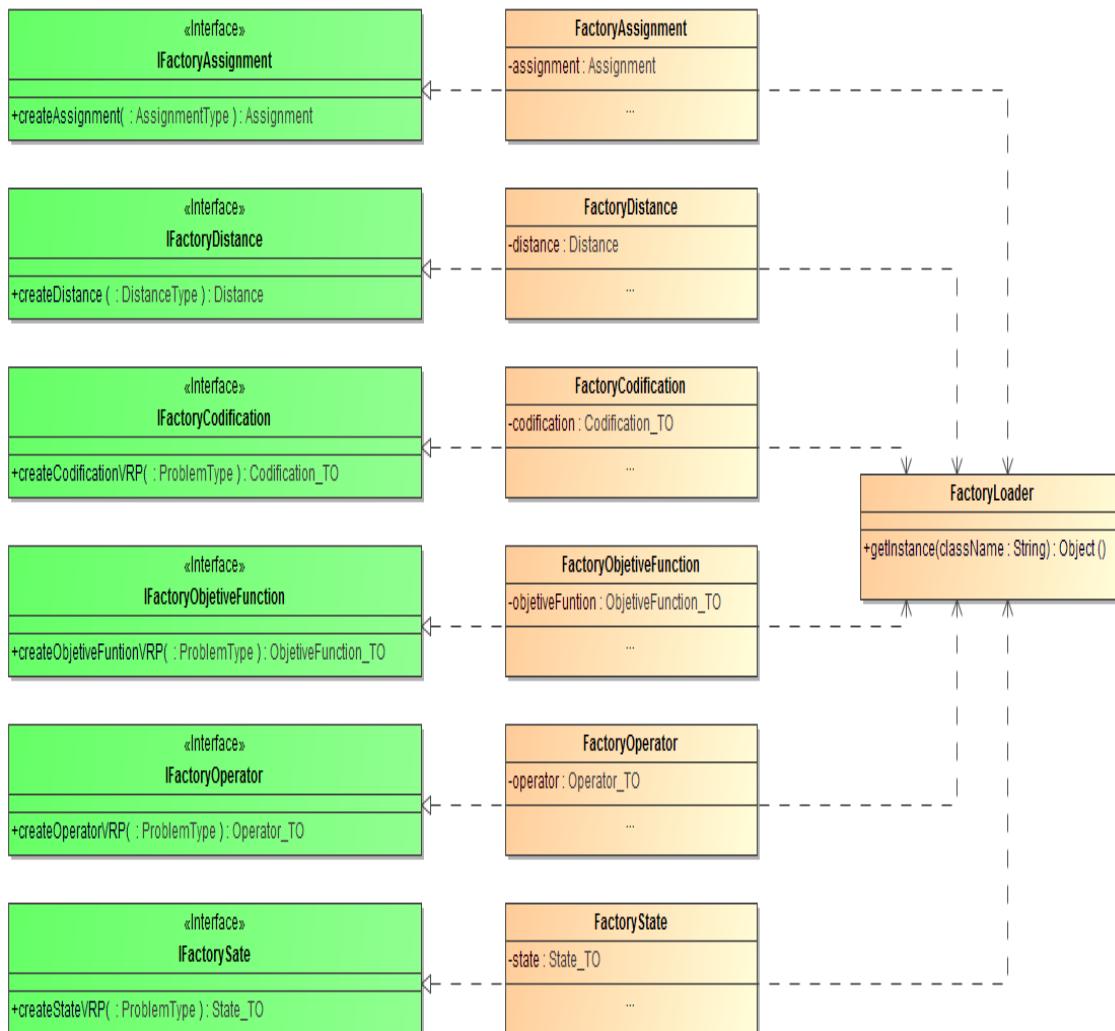


Figura 3.7: Diagrama de clases del paquete **cujae.inf.citi.om.factoryPattern**.

Los elementos participantes en este artefacto se listan en las Tabla 3.7 - Tabla 3.8:

Tabla 3.7: Descripción del paquete cujae.inf.citi.om.factoryPattern.interfaces.

Clases	Descripción
IFactoryDistance	Interfaz que declara los métodos para la creación de métodos para el cálculo de distancias entre dos puntos.
IFactoryAssignment	Interfaz que declara los métodos para la creación de métodos de asignación de puntos de recogidas a los depósitos.
IFactoryState	Interfaz que declara los métodos para la creación de un estado de solución en dependencia de la variante de planificación de rutas de vehículos seleccionada.
IFactoryCodification	Interfaz que declara los métodos para la creación de una codificación en dependencia de la variante de planificación de rutas de vehículos seleccionada.
IFactoryObjectiveFunction	Interfaz que declara los métodos para la creación de una función objetivo en dependencia de la variante de planificación de rutas de vehículos seleccionada.
IFactoryOperator	Interfaz que declara los métodos para la creación de la solución inicial o nuevas soluciones en dependencia de la variante de planificación de rutas de vehículos seleccionada.

Tabla 3.8: Descripción del paquete cujae.inf.citi.om.factoryPattern.methods.

Clases	Descripción
FactoryDistance	Clase que implementa la interfaz <i>IFactoryDistance</i> y construye un método para el cálculo de la distancia entre dos puntos.
FactoryAssignment	Clase que implementa la interfaz <i>IFactoryAssignment</i> y construye un método para la asignación de puntos a depósitos.
FactoryState	Clase que implementa la interfaz <i>IFactoryState</i> y construye un estado de solución en dependencia de la variante de planificación de rutas de vehículos seleccionada.
FactoryCodification	Clase que implementa la interfaz <i>IFactoryCodification</i> y construye una codificación en dependencia de la variante de planificación de rutas de vehículos seleccionada.
FactoryObjetiveFunction	Clase que implementa la interfaz <i>IFactoryObjetiveFunction</i> y construye una función objetivo en dependencia de la variante de planificación de rutas de vehículos seleccionada.
FactoryOperator	Clase que implementa la interfaz <i>IFactoryOperator</i> y construye la solución inicial o nuevas soluciones en dependencia de la variante de planificación de rutas de vehículos seleccionada.
FactoryLoader	Clase que implementa un método genérico para la carga dinámica de cualquier objeto.
DistanceType	Enumerado que indican los tipos de métodos para calcular distancia entre dos puntos en TransO .
AssignmentType	Enumerado que indican los tipos de métodos para asignar puntos de recogidas a depósitos en TransO .

3.2.3. Capa Acceso a Datos

La última capa denominada **Capa de Acceso a Datos** contiene al paquete **cujae.inf.citi.om.dataAccess** que encapsula la lógica para acceder a distintas fuentes de datos. Específicamente, **TransO** permite la entrada de datos desde diferentes formatos de ficheros. Además, existe la posibilidad de ingresar los datos manualmente o mediante diferentes gestores de bases de datos.

En particular este paquete contiene los siguientes elementos:

- **cujae.inf.citi.om.dataAccess.file**: contiene las clases necesarias para la carga de información desde un fichero de texto o un fichero xml.
- **cujae.inf.citi.om.dataAccess.manual**: contiene las clases necesarias para la carga de un *shapefile* en caso de que el usuario no se encuentre conectado a la red donde pueda acceder al servicio web de mapas.
- **cujae.inf.citi.om.dataAccess.database**: contiene las clases necesarias para establecer la comunicación con una base de datos relacional que pueda estar definida en distintos gestores de bases de datos.

La Figura 3.8 muestra un diagrama con la relación que se establece entre los elementos de estos paquetes.



Figura 3.8: Diagrama de clases del paquete cujae.inf.citi.om.dataAccess.

Los elementos participantes en este artefacto se enumeran en las Tabla 3.9 -

Tabla 3.11:

Tabla 3.9: Descripción del paquete cujae.inf.citi.om.dataAccess.file.

Clases	Descripción
LoadFile	Clase que permite acceder a la información que se encuentra almacenada en el fichero de texto.
LoadXML	Clase que permite acceder a la información que se encuentra almacenada en el fichero xml.

Tabla 3.10: Descripción del paquete cujae.inf.citi.om.dataAccess.manual.

Clases	Descripción
LoadShapefile	Clase que permite acceder al mapa almacenado en el proyecto en caso que no exista comunicación con el servicio web de mapa.

Tabla 3.11: Descripción del paquete cujae.inf.citi.om.dataAccess.database.

Clases	Descripción
ConfigurationProperties	Clase que configura la cadena de conexión correspondiente para cada gestor de base de datos relacional.
ConnectionDatabase	Clase para realizar la conexión a los distintos gestores de base de datos relacional.
DataBaseType	Enumerado que indica los tipos de gestores de bases de datos relacionales con que la herramienta se puede conectar.
ManagerDatabase	Clase que implementa distintos procedimientos para acceder a los datos almacenados en la base de datos relacional.

3.3. Diseño de subsistema

Los subsistemas de implementación proporcionan una forma de organizar los artefactos del modelo de implementación en trozos más manejables. Pueden estar formados por componentes, interfaces y otros subsistemas recursivamente. Además, pueden implementar las interfaces que representan la funcionalidad que exportan en forma de operaciones.

El sistema diseñado interactúa con distintos subsistemas para realizar sus funciones. Cada uno de estos subsistemas es descrito a continuación:

3.3.1. Subsistema BiCIAM

BiCIAM es una biblioteca de clases que implementa un modelo unificado de algoritmos metaheurísticos. Esta biblioteca implementa un modelo unificado que constituye un mecanismo general que representa el comportamiento de los algoritmos metaheurísticos a partir de la secuencia de pasos que tienen en común. La arquitectura actual de BiCIAM [1] se muestra en la Figura 1.3. La versión actual de **BiCIAM** incorpora algoritmos basados en trayectoria (actualmente son los que utilizan **Trans-O**) y algoritmos poblaciones y permite que los usuarios resuelvan tanto problemas mono-objetivo como multi-objetivo. En el diagrama de la Figura 3.9 se muestra los elementos que participan en la comunicación con este subsistema.

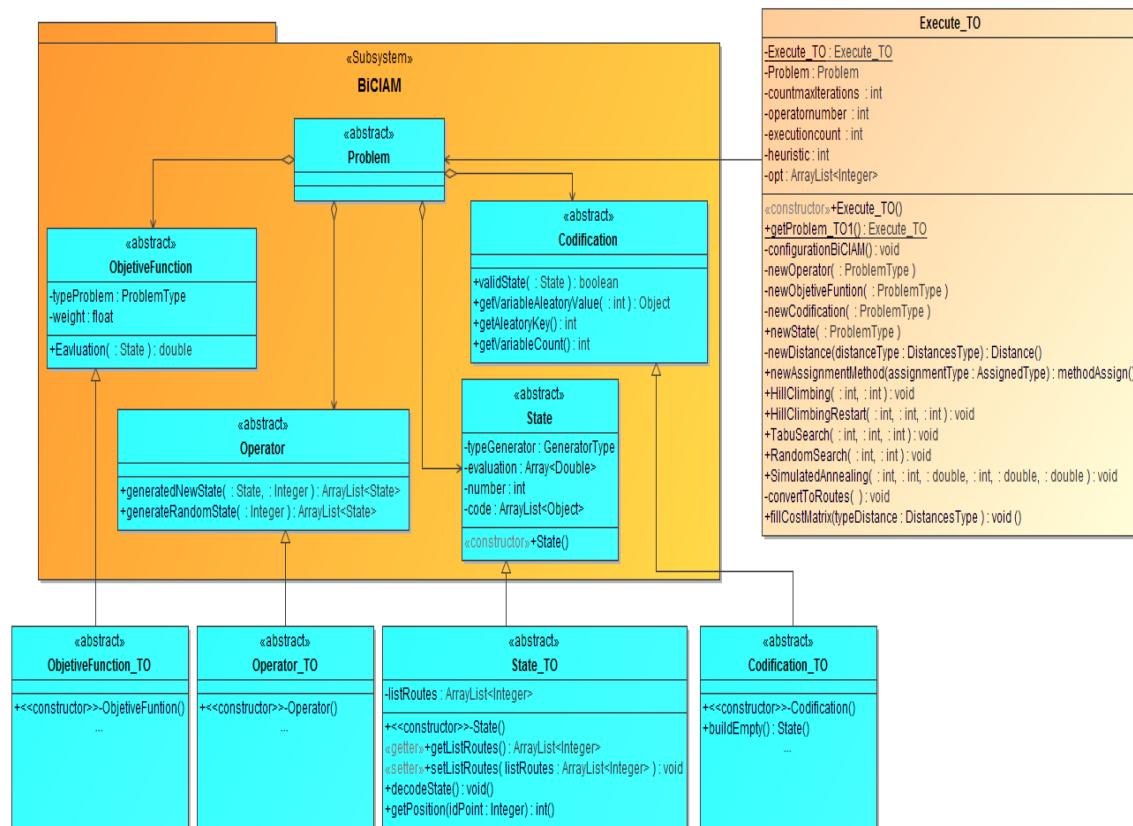


Figura 3.9: Diagrama de clases para la comunicación con el subsistema BiCIAM.

Para la comunicación con **BiCIAM**, esta biblioteca propone las clases *ObjectiveFunction*, *Codification*, *State*, *Operator* y *Problem* que constituyen los aspectos de

diseño para trabajar con un algoritmo metaheurístico y de esta forma se debe definir el comportamiento para el problema a resolver. La clase *Execute_TO* es la controladora de redefinir las clases para que cada variante se pueda modelar como un problema de optimización. Además de contener la instancia que modela el problema como un Problema de Planificación de Rutas de Vehículos.

Una vista dinámica de la comunicación entre **TransO** y **BiCIAM** se muestra en el **Apéndice D**, donde a través de un diagrama de secuencia se representan las clases involucradas en la comunicación y las llamadas entre ellas.

3.3.2. Subsistema BHCVRP

BHCVRP es una biblioteca para la generación de soluciones en variantes VRP utilizando heurísticas de construcción [90]. Esta biblioteca brinda diferentes métodos constructivos que han sido adaptados a diferentes variantes VRP, entre ellas las variantes contempladas en los modelos diseñados para el Problema del Transporte Obrero. Además la misma puede ser utilizada en el contexto de las metaheurísticas como mecanismo para generar la solución inicial. A continuación en la Figura 3.10 se muestra la arquitectura de la biblioteca de clases de **BHCVRP**.

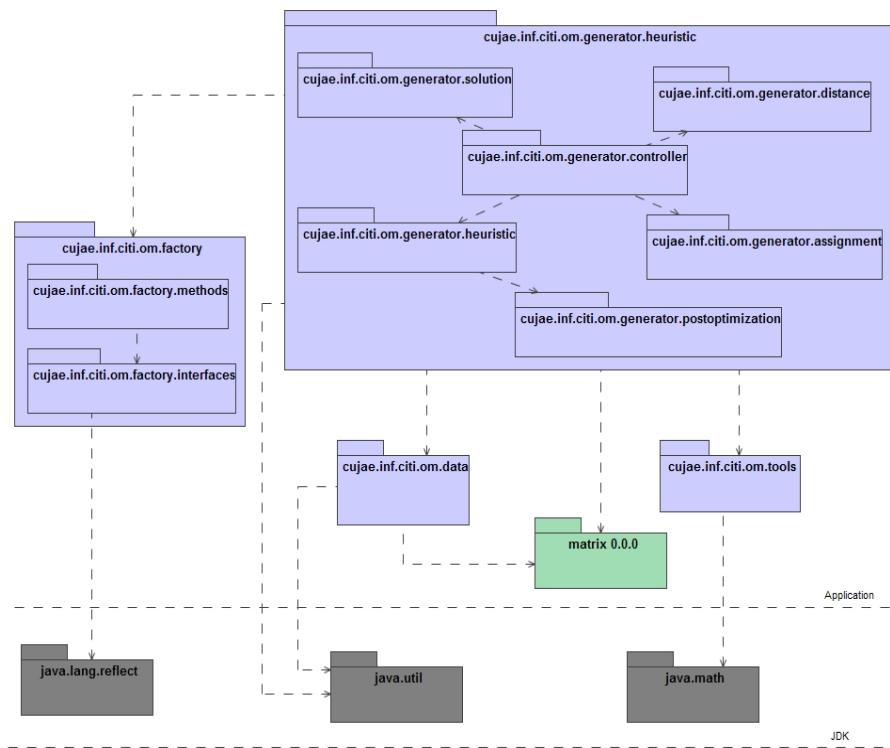


Figura 3.10: Arquitectura de la biblioteca de clases BHCVRP.

La biblioteca **BHCVRP** se compone de dos capas la primera **Capa Application** contiene la mayoría de los paquetes, debido a que es la capa responsable de modelar los problemas de planificación de rutas de vehículos y de definir y ejecutar las heurísticas de construcción. En la **Capa JDK** se ubican los paquetes: **java.lang**, **java.math** y **java.util**, los mismos no pertenecen a la herramienta, pero son necesarios para su funcionamiento. En el diagrama de la Figura 3.11 se muestran los elementos que participan en la comunicación con este subsistema.

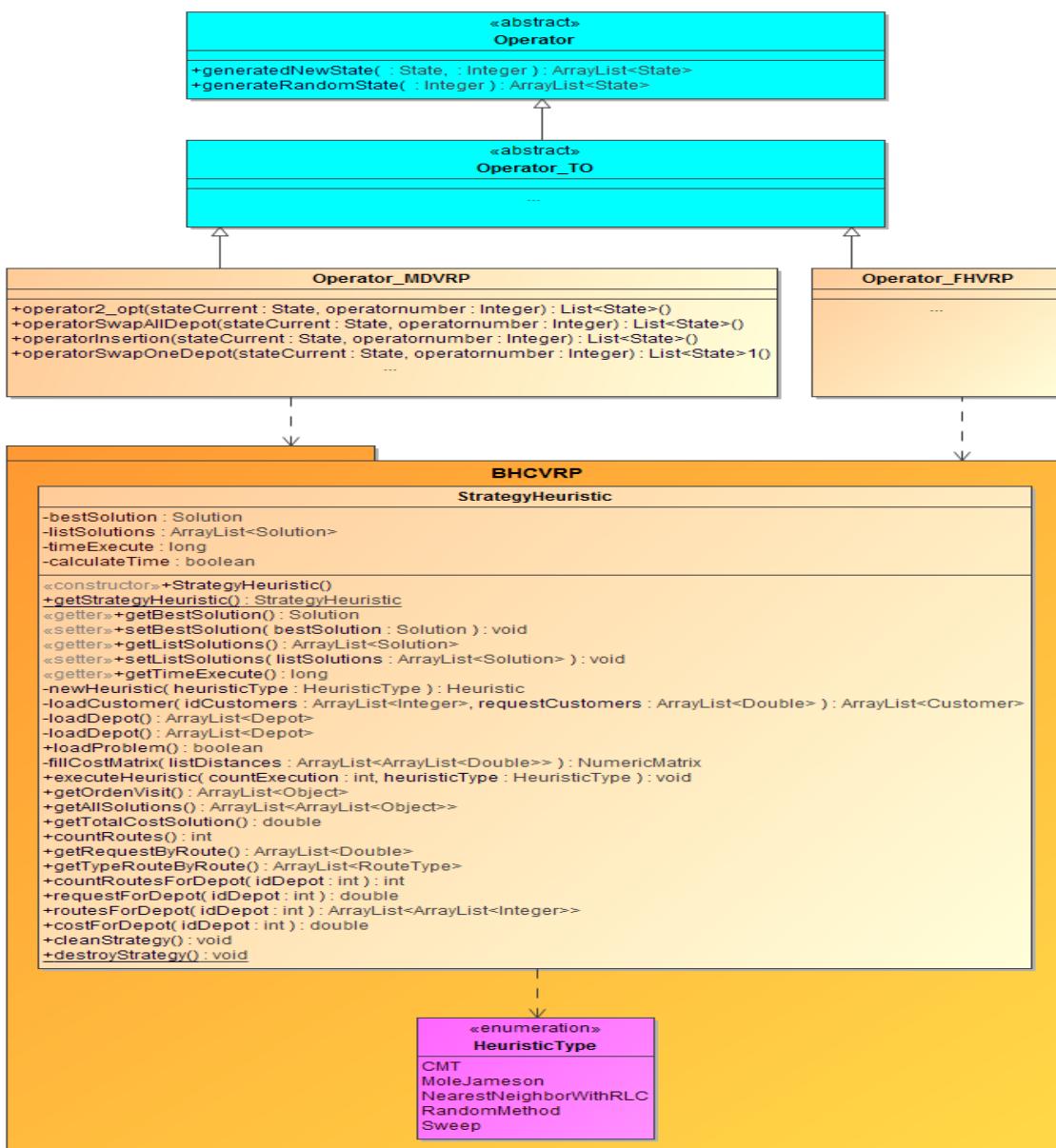


Figura 3.11: Diagrama de clases para la comunicación con el subsistema BHCVRP¹.

¹Por cuestiones de legibilidad a los métodos **loadProblem()** y **loadDepot()** hubo que ocultarle los parámetros de entrada.

Una vista dinámica de la comunicación entre **TransO** y **BHCVRP** se muestra en el **Apéndice D**, donde mediante un diagrama de secuencia se representan las clases involucradas en la comunicación y las llamadas entre ellas.

3.3.3. Subsistema LMOVVRP

En el caso del subsistema **LMOVVRP**, es una biblioteca que brinda diferentes algoritmos (operadores de mutación) para ser utilizados en el proceso de resolución de las metaheurísticas. Estos operadores son utilizados en cada variante a la hora de generar nuevas soluciones. La versión actual de **LMOVVRP** incorpora operadores para las variantes CVRP, TTRP, HFVRP, VRPTW, OVRP y parcialmente para MDVRP [64]. De igual manera contiene diferentes operadores que pueden ser configurados en la ejecución. A continuación en la Figura 3.12 se muestra la arquitectura de la biblioteca de clases de **LMOVVRP**.

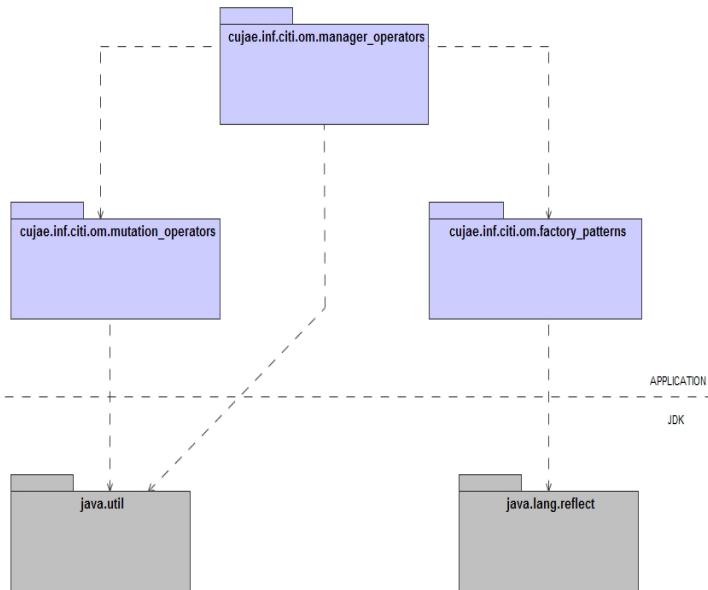


Figura 3.12: Arquitectura de la biblioteca de clases LMOVVRP.

La biblioteca **LMOVVRP** se compone de dos capas: la **Capa Application** contiene la mayoría de los paquetes, debido a que es la capa responsable de definir y ejecutar los operadores de mutación. En la **Capa JDK** se ubican los paquetes: **java.lang.reflect** y **java.util**, los mismos no pertenecen al componente, pero son necesarios para su funcionamiento. En el diagrama de la Figura 3.13 se muestra los elementos que participan en la comunicación con este subsistema.

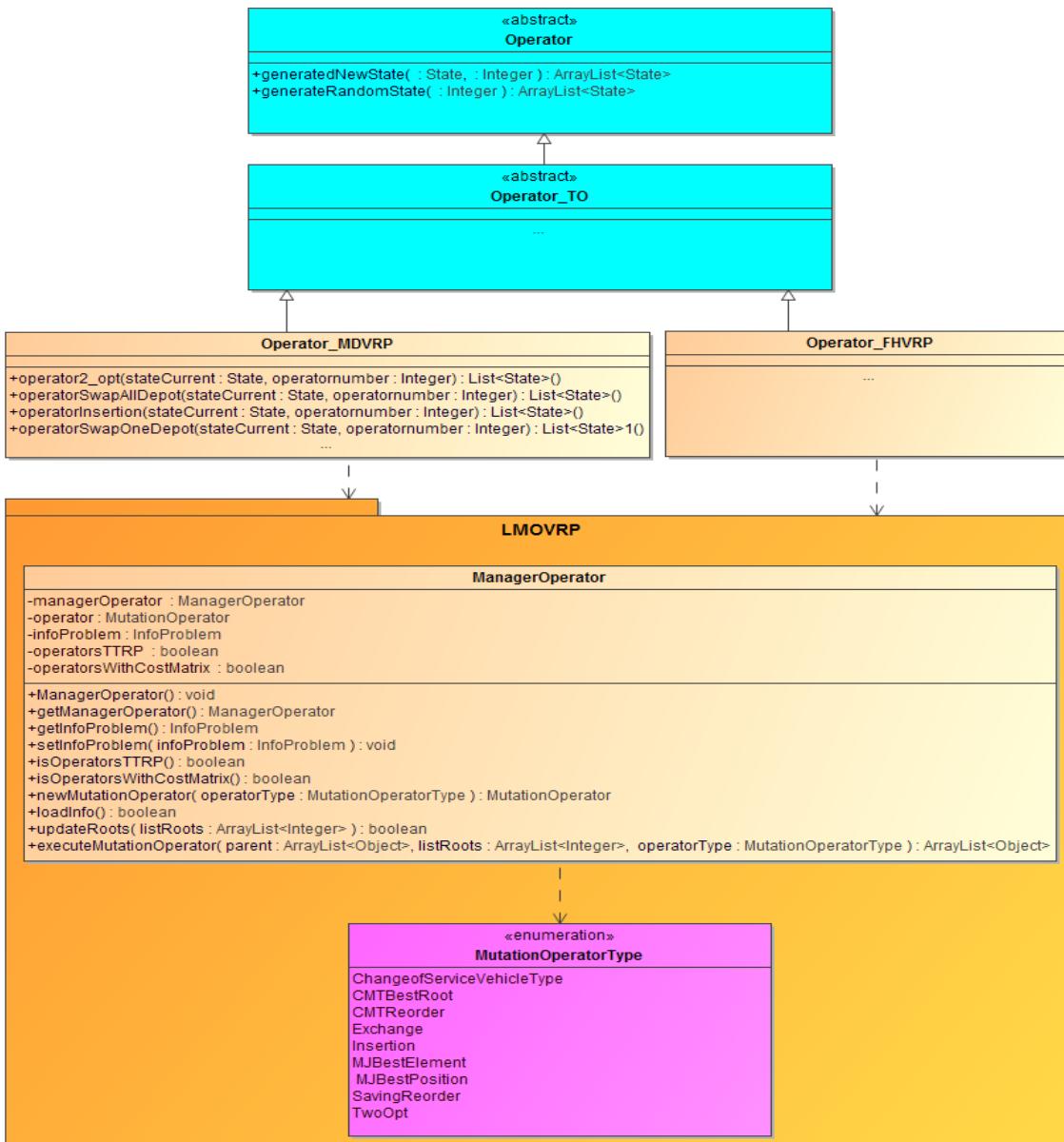


Figura 3.13: Diagrama de clases para la comunicación con el subsistema LMOVVRP².

Una vista dinámica de la comunicación entre **TransO** y **LMOVVRP** se muestra en el **Apéndice D**, donde mediante un diagrama de secuencia se representan las clases involucradas en la comunicación y las llamadas entre ellas.

3.3.4. Subsistema Geotools

Geotools es una biblioteca que provee estándares y métodos para la manipulación de datos espaciales, esencialmente para la implementación de Sistema de Información Geográfica (GIS). Las estructuras de datos de **Geotools** están basadas

²Por cuestiones de legibilidad al método **loadInfo()** hubo que ocultarle los parámetros de entrada.

en especificaciones de Open Geospatial Consortium (OGC) [2, 91]. Esta biblioteca es utilizada en las implementaciones de servicios web y aplicaciones escritorio [91].

Geotools define interfaces para el trabajo con elementos y estructuras espaciales y accede a los datos GIS en varios formatos de archivos y bases de datos espaciales como son: wms, wfs, shapefile e imágenes (JPEG, TIFF, GIF, PNG) [91]. En la Figura 3.14 se muestra la arquitectura en capas de **Geotools**:

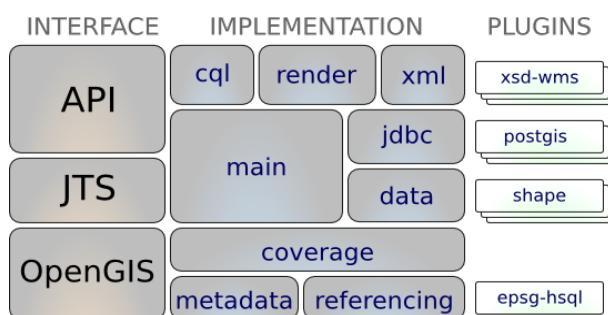


Figura 3.14: Arquitectura de la biblioteca de clases Geotools [2].

La biblioteca de clases **Geotools** se compone de tres capas la primera **Capa Interface** contiene tres paquetes importantes que son **OpenGIS** que define las interfaces comunes que implementan distintos conceptos espaciales, **JTS** que define e implementa los elementos con datos geométricos y **API** que define las interfaces para trabajar con información espacial. La segunda **Capa Implementation** se compone de nueve paquetes los cuales implementan los métodos de renderizado, el acceso a bases de datos espaciales, los formatos xml espaciales, el lenguaje *query* común para filtrar los datos, el acceso a la información rasterizada y la transformación de coordenadas. Por último la **Capa Plugins** que contienen los formatos de mapas que maneja la biblioteca: archivos *shapefiles*, servicios WMS y WFS, acceso por base de datos, imágenes con distintos formatos (jpeg, tiff, png).

3.3.5. Subsistema Server

Server es una infraestructura de datos espaciales que se encuentra disponible en el CITI. Contiene una colección de servicios web basados en datos geográficos que permite analizar y desplegar en todas sus formas la información geográficamente referenciada. En este caso los servicios que se utilizan de este servidor son:

- Servicio de geocodificación.

- Servicio de mapas.
- Servicio de cálculo de matriz de costo.

El modo en el que se realiza la conectividad con el servicio, su vínculo con el cálculo, el análisis y la gestión de los datos que retorna el mismo es la clave para el trabajo con información real en todo momento. De forma tal que a partir de la información de los clientes y depósitos y tras un proceso de construcción de un vínculo con los servicios web se obtenga la información necesaria para seguidamente la herramienta pueda dar solución al Problema del Transporte Obrero. Esto mejora notablemente la eficiencia de la gestión de la información.

3.4. Mecanismos de diseño

La capa de servicio está diseñada de forma robusta para permitir la incorporación de nuevos elementos de optimización a partir de nuevas variantes de problemas de planificación de rutas de vehículos. De forma similar sucede con los métodos para el cálculo de las distancias entre dos puntos y para la asignación de puntos a los depósitos. Es por esta razón que se decide diseñar e implementar un mecanismo de diseño que garantice la carga dinámica de las clases involucradas en los casos mencionados haciendo uso del patrón de diseño *Factory Method*.

Para la implementación del mecanismo de diseño que garantiza la carga dinámica de clases se utiliza la clase *FactoryLoader*. Esta clase contiene un método estático (*getInstance()*), al que se le pasa por parámetro el nombre de la clase que se desea instanciar. El comportamiento de este método se basa en verificar que exista una clase con el mismo nombre que el parámetro que se le pasa y devuelva una instancia de dicha clase. Esta forma dinámica de cargar clases en tiempo de ejecución se realiza utilizando el API (o *Application Program Interface* por su nombre en inglés): *Java Reflection*. Esta API Java permite a los programas examinar y hacer cambios a su estructura y comportamiento en tiempo de ejecución. Además, posibilita obtener la clase de un objeto, examinar los métodos de una clase, invocar métodos descubiertos en tiempo de ejecución y explorar la jerarquía de la herencia. Por ejemplo, con la utilización de *Java Reflection* el método *createAssignment* de la clase *FactorAssignment* solo debe invocar al método *getInstance()*, pasándole como parámetro

el nombre de la clase concreta que se desea instanciar, obteniéndose finalmente una instancia de la misma. Esto es aplicable a otros objetos como: *Distance*, *ObjectiveFunction*, *State*, *Operator*, *Codification*.

A continuación en la Figura 3.15 se muestra la vista estática del mecanismo de diseño empleado en uno de los casos donde se utiliza:

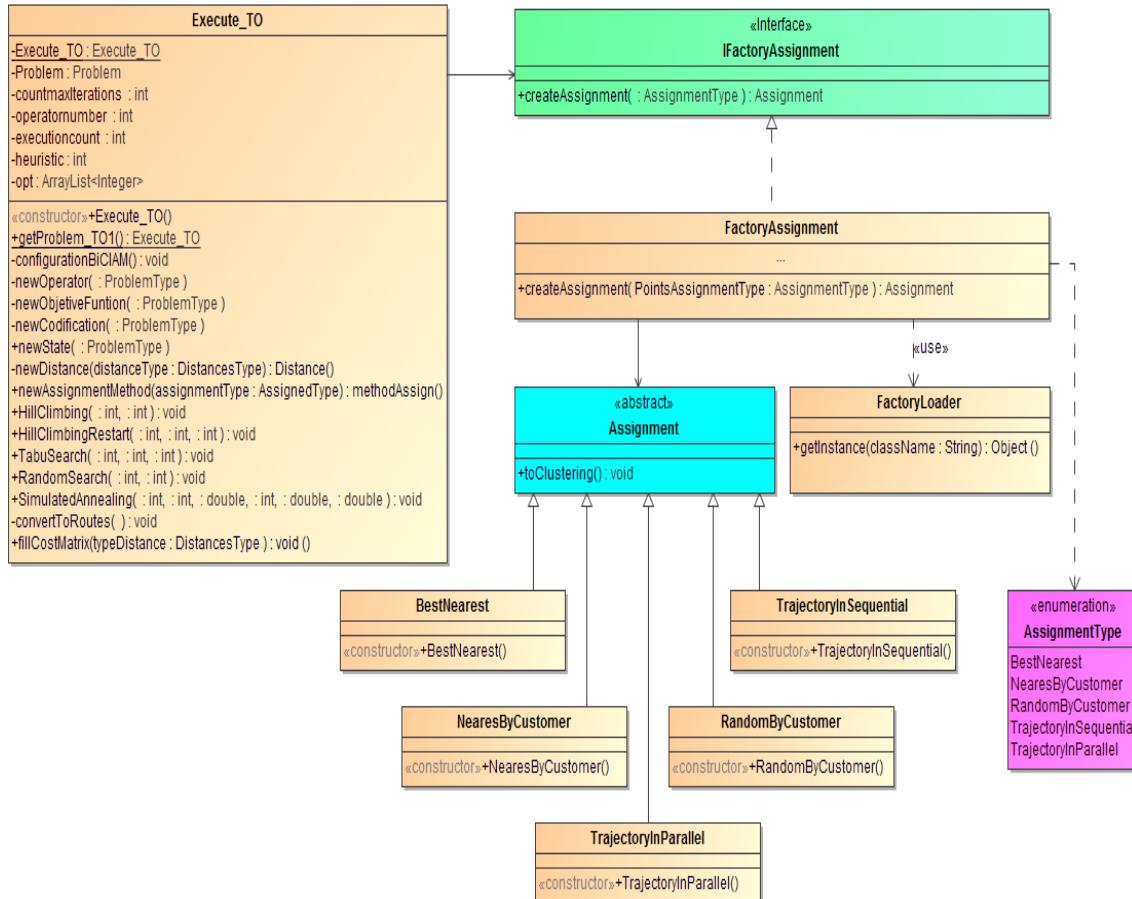


Figura 3.15: Vista estática del mecanismo de diseño empleado.

Cada uno de los elementos utilizados en la vista estática se describe en las Tabla 3.1, Tabla 3.2, Tabla 3.7 y Tabla 3.8.

Además, en el siguiente diagrama de secuencia de la Figura 3.16 se presenta el funcionamiento del mecanismo de diseño para la carga dinámica. En este diagrama en particular se muestran las colaboraciones de los elementos internos del mecanismo respondiendo a las llamadas de los clientes del mismo para el caso de la carga de métodos de asignación de puntos de recogidas a depósitos. Para los otros casos el funcionamiento es similar.

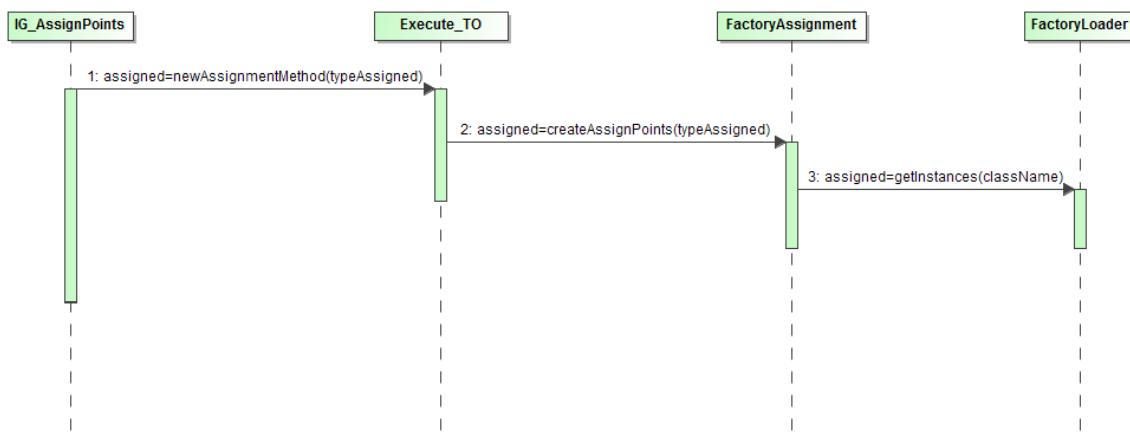


Figura 3.16: Diagrama de secuencia del mecanismo de diseño para la carga dinámica.

3.5. Patrones de diseño

Durante el diseño Orientado a Objetos es frecuente encontrarse repetidamente con ciertos tipos de problemas; para analizar, compartir y documentar el conocimiento sobre estos problemas se han desarrollado los patrones de diseño. Un patrón de diseño es un modelo formal aplicable a diferentes dominios; es decir, ayudan a seguir pautas comunes en la solución de problemas diferentes, pero semejantes en su estructura. Los patrones de diseño resultan ser una solución a un problema de diseño, pero para que esta solución se considere un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias [92].

Dentro de la diversidad de grupos de patrones de diseño que existen, dos de los más utilizados son los patrones GoF y los patrones GRASP. Los patrones GoF (*Gang of Four*, en español, la pandilla de los cuatro) se dan a conocer por los años noventa con la publicación del libro: "Design Patterns: Elements of Reusable Object Oriented Software". Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. El grupo de GoF clasificaron los patrones en tres grandes categorías basados en su propósito [93]:

- **Patrones estructurales:** permiten crear grupos de objetos para ayudar a reali-

zar tareas complejas.

- **Patrones de comportamiento:** permiten definir la comunicación entre los objetos de un sistema y el flujo de la información entre los mismos.
- **Patrones creacionales:** estos patrones crean objetos evitando que se tenga que instanciar directamente, proporcionando a los programas una mayor flexibilidad para decidir qué objetos usar.

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Por lo que en la elaboración de la herramienta se decidió hacer uso de los patrones creacionales *Factory Method* y *Singleton* [92, 94, 95].

3.5.1. Patrón Singleton

El patrón *Singlenton* es un patrón creacional que garantiza que solamente se cree una instancia de una clase proporcionando un punto de acceso global a ella y a los objetos que utilizan la instancia. En otras palabras provee un mecanismo para limitar el número de instancias de una clase, por lo que el mismo objeto es siempre compartido por distintas partes del código [92, 94, 95]. En el aspecto de la implementación de este patrón existen diferentes maneras de lograr su objetivo. Una de las vías más utilizadas es crear el *Singleton* con un método estático. En la herramienta existe la necesidad de implementar este patrón en la clase *Problem_TO* y *Execute_TO* para garantizar los aspectos antes mencionados. En la Figura 3.17 se muestra la implementación de este patrón en la clase *Problem_TO* de la herramienta **TransO**.

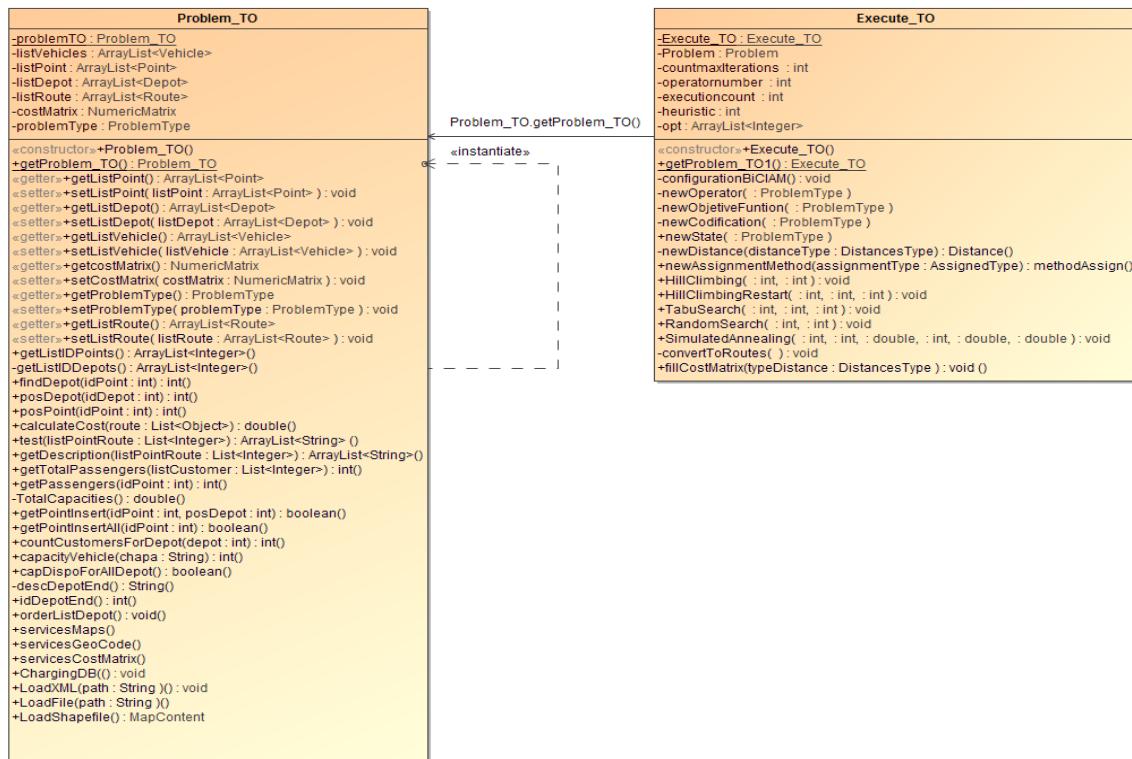


Figura 3.17: Implementación del patrón Singlenton en la clase Problem_TO.

3.5.2. Patrón Factory Method

El patrón *Factory Method* es un patrón de creación que define una clase *interface* para crear un objeto. Con este fin delega la responsabilidad a las subclases de decidir qué clase instanciar. El funcionamiento de este patrón está incorporado en el mecanismo de diseño para la carga dinámica que se explicó en el acápite 3.4: "Mecanismos de diseño". En la Figura 3.15 se muestra la implementación de este patrón en la herramienta.

3.5.3. Otros patrones de diseño

Dentro de los patrones más usados además de los antes mencionados GoF también se encuentra el grupo de los patrones GRASP, este nombre es un acrónimo de *General Responsibility Assignment Software Patterns*. El nombre se eligió para indicar la importancia de captar estos principios cuando se desea diseñar eficazmente el software orientado a objetos. Estos patrones son los que describen los principios fundamentales de la asignación de responsabilidades en objetos [92, 95]. Existen diferencias significativas entre estos grupos debido a que los GoF dan una solución

implementable con su propio diagrama de clases que muestra la forma en que deben ser usados; mientras los GRASP no implementan las soluciones, más bien llevan a pensar en el diseño, a nivel de principios generales. Los patrones pertenecientes a este grupo son: Experto, Creador, Bajo Acoplamiento, Alta Cohesión, Controlador, Polimorfismo, Fabricación Pura, Indirección y No Hables con Extraños. A continuación se describen los que se ponen en práctica en la implementación de **TransO**.

- **Patrón Controlador:** este patrón permite resolver el tema de quién es el responsable de atender un evento del sistema. La solución consiste en asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Por ejemplo, en **TransO** se pone en práctica en la clase *Problem_TO* que es la clase controladora de la lógica de la herramienta y en la clase *Execute_TO* que es la clase controladora del proceso de optimización.
- **Patrón Experto:** con este patrón se pretende resolver el tema de delegar responsabilidades a cada objeto. La solución es asignar una responsabilidad al experto en información, es decir, la clase que tiene la información necesaria para llevar a cabo la responsabilidad. Por ejemplo, en **TransO** se aplica en la clase *Problem_TO* que es la responsable de todos datos de un problema de planificación de rutas de vehículos y de brindar información necesaria respecto a ellos.
- **Patrón de Alta Cohesión:** este patrón es usado si se presentan problemas para manejar la complejidad. La solución es asignar responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. En **TransO** se emplea en la clase *Point* es la responsable de los datos de un punto de recogida.
- **Patrón Creador:** el problema de saber a quién le corresponde crear una nueva instancia de una clase es abordado con este patrón. El mismo propone como solución la creación de una nueva instancia por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, almacena o maneja varias instancias de la clase y contiene o agrega la clase. La clase controladora *Problem_TO* de **TransO** contiene una lista de *Depot* que demuestra el uso de este patrón.

- **Patrón de Bajo Acoplamiento:** este patrón permite resolver el problema de dar soporte a las bajas dependencias y al incremento de la reutilización. La solución es realizar un diseño donde las clases estén menos ligadas entre sí. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Un ejemplo en **TransO** se evidencia en la clase *State_TO* una clase base creada para agrupar los métodos comunes de las sub-clases *State_MDVRP* y *State_FHVRP*.

3.6. Tratamiento de errores

Cuando se está desarrollando un software es importante tomar acciones para minimizar la ocurrencia de errores. Estos pueden afectar la confiabilidad de los datos almacenados y por tanto disminuir la confianza del usuario en el sistema.

Durante la implementación se concibieron un conjunto de validaciones a nivel de interfaz de usuario para minimizar la cantidad de errores que pueden pasar a las capas inferiores de procesamiento. Los errores son mostrados al usuario mediante cuadros de diálogos brindados por el IDE y algunos cuadros de información que se agregaron a la aplicación. A continuación en la Tabla 3.12 se muestran los errores más comunes mostrados, así como una breve descripción del significado del mismo.

Tabla 3.12: Descripción de los errores comunes en el sistema.

Error	Descripción
Error, el fichero no es correcto	Es lanzado cuando el fichero a cargar no cumple con la estructura correcta.
Fallo de conexión con la Base de Datos	Es lanzado cuando los datos para la conexión con el servidor de bases de datos no son correctos o el servidor no se encuentra disponible.

Adicionalmente en la herramienta se tuvieron en cuenta una serie de avisos para alertar al usuario de las restricciones que se presenta en el Problema del Transporte Obrero y que pueden ocasionar la obtención de soluciones no factibles en el problema. El tratamiento dado en estos casos puede variar, pues existen restricciones que

se tratan de forma flexible como el incumplimiento de las capacidades de los vehículos. En este caso la herramienta admite que el usuario continúe la planificación de forma normal. Sin embargo, otras restricciones no permiten continuar con el proceso de resolución pues invalidan el proceso de planificación, por ejemplo, la existencia de más depósitos que vehículos disponibles o la no selección de un depósito final o inicial en dependencia de la variante de solución seleccionada. Todas estas alertas ayudan al usuario a tener presente las restricciones que se deben considerar a la hora de planificar los recorridos del transporte obrero.

3.7. Modelo de despliegue

Un diagrama de despliegue muestra la forma en que se distribuye físicamente el sistema. Las unidades de hardware que tienen capacidad de procesamiento se representan a través de nodos. En estos nodos es donde se ubican las distintas aplicaciones para su ejecución. Además, se identifican dispositivos que carecen de procesamiento y las relaciones entre los nodos a través de los protocolos de comunicación [96].

Es importante aclarar que el trabajo que se está realizando es una aplicación escritorio para resolver problemas de planificación de rutas de vehículos. Como parte de su funcionamiento se propone el consumo de varios servicios web como son Geocodificación, Calculo de matriz de costo y Mapas por lo que para su uso solo se necesita una PC conectada a la red donde se encuentran disponibles estos servicios. A continuación se muestra en la Figura 3.18 el diagrama de despliegue de la aplicación.

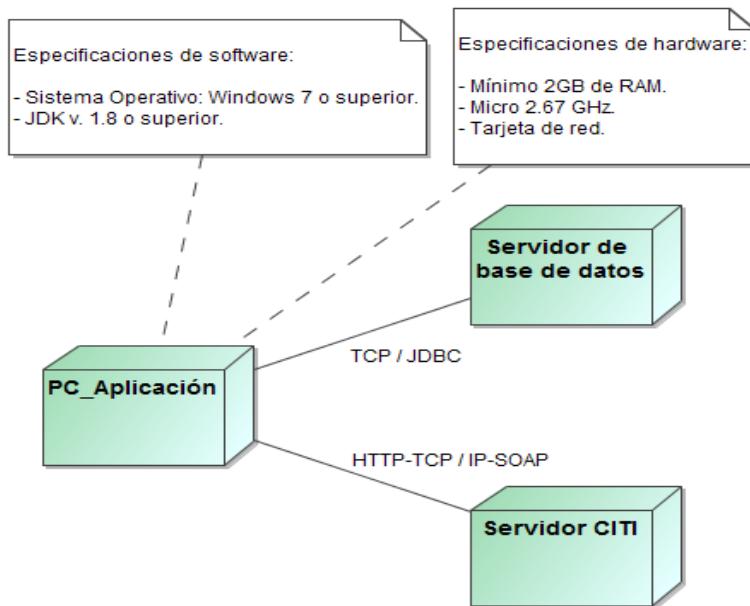


Figura 3.18: Diagrama de despliegue.

En la Tabla 3.13 se muestra la descripción de los nodos que componen el diagrama de despliegue anteriormente expuesto:

Tabla 3.13: Descripción del diagrama de despliegue.

Clases	Descripción
PC_Aplicación	En este nodo se encuentra la herramienta escritorio usada para generar las rutas en el Problema del Transporte Obrero. Es fundamental que este nodo se encuentre conectado a la red para consumir los servicios web que brinda la infraestructura de datos espaciales publicada en el servidor del CITI en los casos que lo requiera.
Servidor de bases de datos	Es el servidor donde se encuentra alojado el gestor de bases de datos con el cual se debe comunicar la aplicación.
Servidor CITI	Es el servidor donde se encuentran alojados los servicios ofrecidos por la infraestructura de datos espaciales publicada en el servidor del CITI.

3.8. Herramientas y tecnologías

Para alcanzar un desarrollo exitoso del proyecto se realizó un estudio de las diferentes herramientas y tecnologías disponibles que pueden ser empleadas para dar solución a las problemáticas existentes que influyen en el desarrollo de la herramienta. Durante la investigación se tuvieron en cuenta las características de cada una de las herramientas, así como las ventajas que se obtendrían de cada una de ellas. A continuación se describe la tecnología utilizada:

3.8.1. Lenguaje de programación JAVA

Se selecciona Java como lenguaje de programación para la implementación de la herramienta debido a que la biblioteca BiCIAM utilizada para la resolución del problema se encuentra desarrollada en este lenguaje. El uso de Java para una mejor integración con esta biblioteca; así como las siguientes características del lenguaje son las razones que sustentan esta decisión. Además existe una experiencia previa en este lenguaje debido que es el lenguaje utilizado para el desarrollo de los distintos productos de software en el proyecto.

Características:

- **Orientado a objetos:** brinda gran soporte a las técnicas de desarrollo de la programación orientada a objetos potenciando la reutilización de componentes de software.
- **Sólido:** el código no se quiebra fácilmente ante errores de programación.
- **Portable:** al poseer una arquitectura neutral es altamente portable, pero esta característica puede verse de otra manera: los tipos de datos nativos se presentan igualmente implementados en todas las máquinas virtuales por lo que las operaciones aritméticas funcionaran de la misma manera.
- **Multi-hilos:** puede aplicarse a la realización de aplicaciones en las que ocurra más de un proceso. Se apoya en un sistema de gestión de eventos basado en el paradigma de condición y monitores que soporta la conducta en tiempo real e interactivo en programas.

- Además, posee soporte para las tres características propias del paradigma de la orientación a objetos: encapsulación, enlace dinámico y polimorfismo.

3.8.2. Entorno de Desarrollo Integrado: NetBeans

Se consideró realizar la herramienta utilizando Eclipse o NetBeans. Ambos IDE permiten el desarrollo de aplicaciones con el lenguaje de programación Java. Fue seleccionado NetBeans como entorno de desarrollo para el proyecto, debido a las facilidades que brinda en el trabajo con interfaces visuales.

El IDE NetBeans no requiere un *kit* de desarrollo de software adicional. Las aplicaciones pueden instalar módulos dinámicamente. Algunas aplicaciones pueden incluir un módulo de actualización para permitir a los usuarios descargar actualizaciones de firma digital y nuevas características directamente dentro de la aplicación en ejecución. La plataforma ofrece servicios reusables comunes para las aplicaciones de escritorio, permitiendo a los desarrolladores centrarse en la lógica de sus aplicaciones. El *framework* está simplificando el desarrollo de aplicaciones para escritorio con Java Swing. El paquete del IDE NetBeans para Java SE contiene lo que se necesita para empezar a desarrollar plugins y aplicaciones basadas en la plataforma NetBeans; no se requiere un SDK adicional. Algunas de las características de la aplicación son:

Características:

- El IDE es libre, código abierto, multiplataforma con soporte integrado para el lenguaje de programación Java.
- Gestión de la interfaz de usuario (menús, barras de herramientas, personalización de botones y campos, entre otros).
- Gestión de configuración de usuario.
- Gestión de almacenamiento (guardar o cargar algún tipo de dato).
- Gestión de ventanas.
- Marco Asistente (soporta diálogos paso a paso).
- Biblioteca visual de NetBeans integrada.

3.9. Estudio de factibilidad de la propuesta de solución

El estudio de factibilidad nos permite determinar si un proyecto es viable desde distintos puntos de vista como los son el técnico, legal, organizacional y económico. En este acápite se detallara los estudios de factibilidad que se realizaron para determinar la viabilidad del proyecto considerando para ello los diferentes estudios de factibilidad. Debido a la inexactitud de los sistemas para el cálculo del costo de software surgen métodos de estimación aproximados, además de existir diferentes factores que deben analizarse. Los costos del desarrollo de un sistema se encuentran en la ingeniería aplicada para lograr su despliegue [97].

En el siguiente acápite se realiza un análisis del esfuerzo y costo de implementación del sistema mediante el método de estimación basado en casos de uso. A través de este se estima el esfuerzo, costo y tiempo de desarrollo del proyecto. Los detalles acerca de la estimación se encuentran a continuación.

Factibilidad técnica

En el caso de la factibilidad técnica del proyecto se puede afirmar que es factible en este aspecto. Esta afirmación se sustenta en que el proyecto se realiza sobre un IDE de desarrollo libre. Además, presenta nivel medio de requerimientos para su posterior uso ya que es una herramienta que genera rutas dependiendo de datos obtenidos del usuario y los resultados que devuelve los obtiene utilizando bibliotecas que se integran a la misma.

Factibilidad legal

El proyecto cuenta con soporte legal y no viola ninguna ley o normativa referente a las tecnologías de la información y las telecomunicaciones ni ninguna firma digital a ningún nivel.

Factibilidad organizacional

En este sentido el trabajo a realizar constituye una herramienta de generación de rutas para el Problema del Transporte Obrero. El impacto del proceso de desarrollo

del mismo no afecta de forma alguna a la organización ya que las responsabilidades en cada fase están correcta y claramente distribuidas de manera tal que no entren en conflicto con los otros proyectos que desarrolla el grupo.

Factibilidad económica

La fase de diseño ha sido desarrollada mediante el uso de los artefactos definidos por la metodología RUP. Se ha definido la arquitectura de la herramienta, patrones de diseño, entre otros. Estos en conjunto marcan las pautas a seguir para la fase de implementación, así como los elementos más importantes a probar en la fase de prueba para validar la solución propuesta.

El capítulo termina con dos puntos esenciales para un proceso de desarrollo de software: la selección de tecnologías en base a los requerimientos establecidos por la plataforma sobre la cual se va a desarrollar y la estimación de esfuerzo y costo usando el método de estimación basado en casos de uso que se detalla en el **Apéndice E**. En dicho análisis se obtienen resultados satisfactorios con respecto a lo esperado, ya que la aplicación es desarrollada por una persona para la culminación de su carrera.

En resumen al estimar el esfuerzo y costo según el conocido método de análisis de puntos de caso de uso se ha llega a la conclusión que el esfuerzo para desarrollar el proyecto es de aproximadamente **22.8 meses** y el costo total asciende a aproximadamente **5953.066 pesos**.

3.10. Conclusiones parciales

En el presente capítulo se exhibe la vista de la arquitectura utilizando un enfoque basado en responsabilidad, así como las bibliotecas y subsistemas que no son propios de la herramienta pero son de vital importancia para su funcionamiento. También se presenta un mecanismo para la carga dinámica utilizando el patrón *Factory Method*. Además, se hace uso de otros patrones para garantizar la extensibilidad y reusabilidad de la herramienta. Por último, se realiza un análisis de esfuerzo y costo de implementación del sistema mediante el método de estimación basado en casos de uso.

En este capítulo se identificaron aquellos aspectos relevantes en el diseño e implementación de la solución propuesta dando lugar a la creación de la herramienta **TransO** por tanto, se concluye que:

- ✓ La vista de la arquitectura utilizando un enfoque basado en responsabilidad permite resaltar los distintos niveles del diseño del trabajo, teniendo en cuenta la estructuración en capas de los paquetes y los subsistemas.
- ✓ Existe una necesidad real de utilizar los subsistemas presentados pues constituyen elementos de vital importancia para el funcionamiento de **TransO**. En el caso del subsistema **Geotools** es necesario para el trabajo con mapas que representa uno de los requisitos fundamentales de la herramienta. El subsistema **Server** es requerido a la hora de utilizar información real en el trabajo con los datos, siendo necesario el consumo de diferentes servicios web que se encuentran disponibles en él. Además, para facilitar la generación de rutas en la herramienta se utilizan los subsistemas **BiCIAM**, **BHCVRP** y **LMOVRP** debido a que encapsulan diferentes algoritmos metaheurísticos, heurísticas de construcción y operadores de mutación respectivamente para la obtención de buenas soluciones.
- ✓ Para garantizar la extensibilidad, reusabilidad y el diseño de la aplicación se utilizaron los patrones *Singleton* y *Factory Method*. Además de otros patrones del grupo GRASP.
- ✓ Se consideró diseñar e implementar un mecanismo para la carga dinámica utilizando el patrón *Factory Method* debido a que la herramienta en tiempo de ejecución requiere cargar aquellas funcionalidades configurables por el usuario durante el proceso de planificación.
- ✓ Para alcanzar un desarrollo exitoso del proyecto se realizó un estudio de las diferentes herramientas y tecnologías disponibles que pueden ser empleadas para dar solución a la problemática existente, de las cuales se seleccionaron Java y NetBeans debido al conocimiento previo obtenido durante la carrera, además de ser utilizadas en los proyectos del CITI.

-
- ✓ Se realiza un análisis de esfuerzo y costo de implementación del sistema mediante el método de estimación basado en casos de uso, arribando a la conclusión que el esfuerzo para desarrollar el proyecto es de aproximadamente **22.8 meses** y el costo total asciende a aproximadamente **5953.066 pesos**. El desarrollo del sistema es perfectamente factible, al no requerirse costos en su elaboración y ser sencilla la instalación y mantenimiento.

Capítulo 4

Validación de la solución propuesta

4.1. Introducción

En este capítulo se describe la validación realizada sobre la herramienta para la generación de rutas en el Problema del Transporte Obrero: **TransO**. Inicialmente se realizan un conjunto de pruebas que permiten validar el correcto funcionamiento de la aplicación e identificar los algoritmos más convenientes para cada variante en el proceso de planificación. Además, se hace uso de pruebas estadísticas no paramétricas para el análisis de los resultados obtenidos. Por último, se realiza un caso de estudio en la entidad CITI con diferentes escenarios que simulan situaciones reales.

4.2. Pruebas

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software y comprobar el comportamiento de la herramienta contra el resultado esperado, a partir de un conjunto finito de casos de prueba, específicamente seleccionados dentro de un dominio usualmente infinito de ejecuciones posibles. De las pruebas realizadas el programador puede ser capaz de comprobar que la aplicación funciona como fue diseñada y que los requisitos con los que debe cumplir están implementados correctamente [98]. A continuación se describen las principales tipos pruebas que se pueden realizar a cualquier tipo de software.

- Pruebas unitarias
 - ✓ Pruebas unitarias

- Pruebas de integración
 - ✓ Prueba de Integración
 - ✓ Prueba de Regresión
 - ✓ Pruebas de Humo (*Smoke Testing o Ad Hoc*)
- Pruebas del sistema
 - ✓ Pruebas del Sistema
 - ✓ Pruebas de Desempeño
 - ✓ Pruebas de Carga
 - ✓ Pruebas de Stress
 - ✓ Pruebas de Volumen
 - ✓ Pruebas de Recuperación y Tolerancia a fallas
 - ✓ Prueba de Múltiples Sitios
 - ✓ Prueba de Compatibilidad y Conversión
 - ✓ Pruebas de Integridad de Datos y Base de Datos
 - ✓ Pruebas de Seguridad y Control de Acceso
- Pruebas de validación a sistemas a la medida
 - ✓ Pruebas del Ciclo del Negocio
 - ✓ Pruebas de GUI
 - ✓ Pruebas de Configuración
 - ✓ Prueba de Estilo
 - ✓ Prueba de Aceptación
 - ✓ Prueba de Instalación
 - ✓ Pruebas Funcionales
 - ✓ Prueba de Documentación y Procedimiento
 - ✓ Prueba de Usabilidad
 - ✓ Prueba de Campo

- Pruebas de validación a aplicaciones genéricas

- ✓ Pruebas Alfa

- ✓ Pruebas Beta

Para la realización de las pruebas al sistema se decide utilizar las Pruebas de validación específicamente las Pruebas Funcionales. Este tipo de pruebas están basadas en técnicas de caja negra, donde se verifica la aplicación (y sus procesos internos) mediante la interacción con la aplicación vía Interfaz Gráfica de Usuario (GUI) y se analizan las salidas (resultados). Además, se asegura el trabajo apropiado de los requisitos funcionales, incluyendo la navegación, entrada de datos, procesamiento y obtención de resultados [99].

4.2.1. Diseño de los casos de prueba

Los diseños de casos de prueba son un conjunto de entradas con datos, condiciones de ejecución y resultados esperados, que sirven para identificar y comunicar las condiciones que se llevarán a cabo en las pruebas. Son necesarios para verificar la aplicación exitosa y aceptable de los requisitos del producto [99].

En la Figura 4.1 se muestra el diseño de caso de prueba para validar el comportamiento de la herramienta de manera general y los escenarios que deben ser evaluados tanto para la variante desde los depósitos a la entidad (Variante MD) como para la variante desde la entidad a los depósitos (Variante FH).

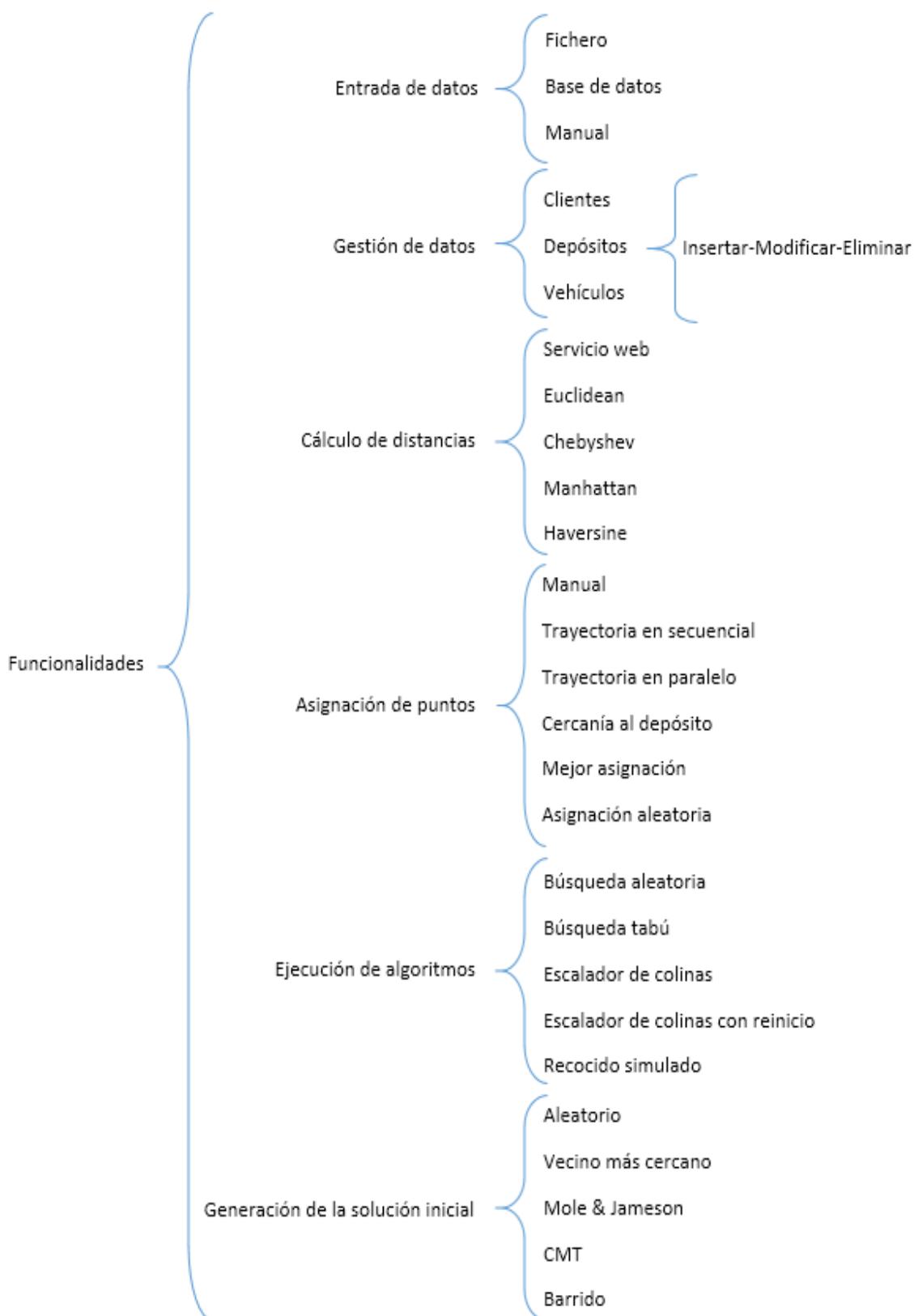


Figura 4.1: Esquema de casos de prueba.

En la figura anterior se muestran las distintas combinaciones que se consideraron

para validar el funcionamiento de la herramienta para las dos variantes implementadas. En ambas variantes se prueban las distintas combinaciones entre los diferentes tipos de funcionalidades: Entrada de datos, Gestión de datos, Cálculo de distancia, Ejecución de algoritmos metaheurísticos y Generación de solución inicial utilizando heurísticas de construcción. Adicionalmente, la variante desde los depósitos hacia la entidad (Variante MD) realiza la funcionalidad de Asignación de puntos a depósitos.

4.2.2. Descripción de las instancias

Para la ejecución de las pruebas se utilizan los datos brindados por el personal de Transporte del CITI. A continuación en la Tabla 4.1 se muestran dichos datos.

Tabla 4.1: Datos brindados por el personal de Transporte del CITI.

Cant. Puntos	Cant. Trabajadores	Cant. Depósitos	Cant. Vehículos
58	101	6	5

A partir de los datos recopilados se puede decir que los puntos de recogidas se encuentran distribuidos en los 15 municipios de la provincia de La Habana. Además, el promedio de trabajadores que se recogen en cada punto equivale a 2. Mientras que las capacidades de los vehículos oscilan entre 8 y 42.

4.2.3. Resultados obtenidos en las pruebas

Las pruebas realizadas se ejecutaron en un entorno virtual que satisface los requisitos no funcionales definidos de hardware y software. A continuación en la Tabla 4.2 se muestra un resumen de las pruebas de software realizadas:

Tabla 4.2: Resultado de las pruebas de software.

Variante	Pruebas		
	Total	Satisfactorias	No satisfactorias
Variante MD	20250	20057	193
Variante FH	3375	3298	77
Total	23625	23355	270

Se detectaron un total de 270 pruebas no satisfactorias que representa 1.14%

del total de pruebas realizadas. Es importante mencionar, que como resultado de este estudio todas las funcionalidades que presentaron anomalías fueron corregidas. Además, estas pruebas permitieron identificar los algoritmos más convenientes para cada variante en el proceso de planificación.

4.2.4. Análisis de los resultados para la variante desde los depósitos a la entidad

En los gráficos de las Figuras 4.2 - 4.4 se muestran los resultados obtenidos después de haber ejecutado el modelo desde los depósitos a la entidad (variante MD) haciendo uso de todas las combinaciones posibles entre los algoritmos metaheurísticos, los métodos de asignación y las heurísticas de construcción incorporadas a la herramienta **TransO**. En todos los casos se realizan 5 ejecuciones y 1000 iteraciones. Los datos con los cuales se ejecutaron las pruebas se encuentran descritos en la sección 4.2.2: "Descripción de las instancias".

A partir de los resultados alcanzados utilizando los cinco métodos de asignación implementados en la herramienta **TransO**, el método de asignación de puntos de recogidas a los depósitos denominado Trayectoria en Secuencial es el que presenta mejores resultados. Luego de aplicar la prueba estadística no paramétrica de Friedman [100] se obtiene que los métodos: Mejor Asignación y Trayectoria Secuencial superan a los restantes métodos significativamente, lo que se puede apreciar en la Figura 4.2.

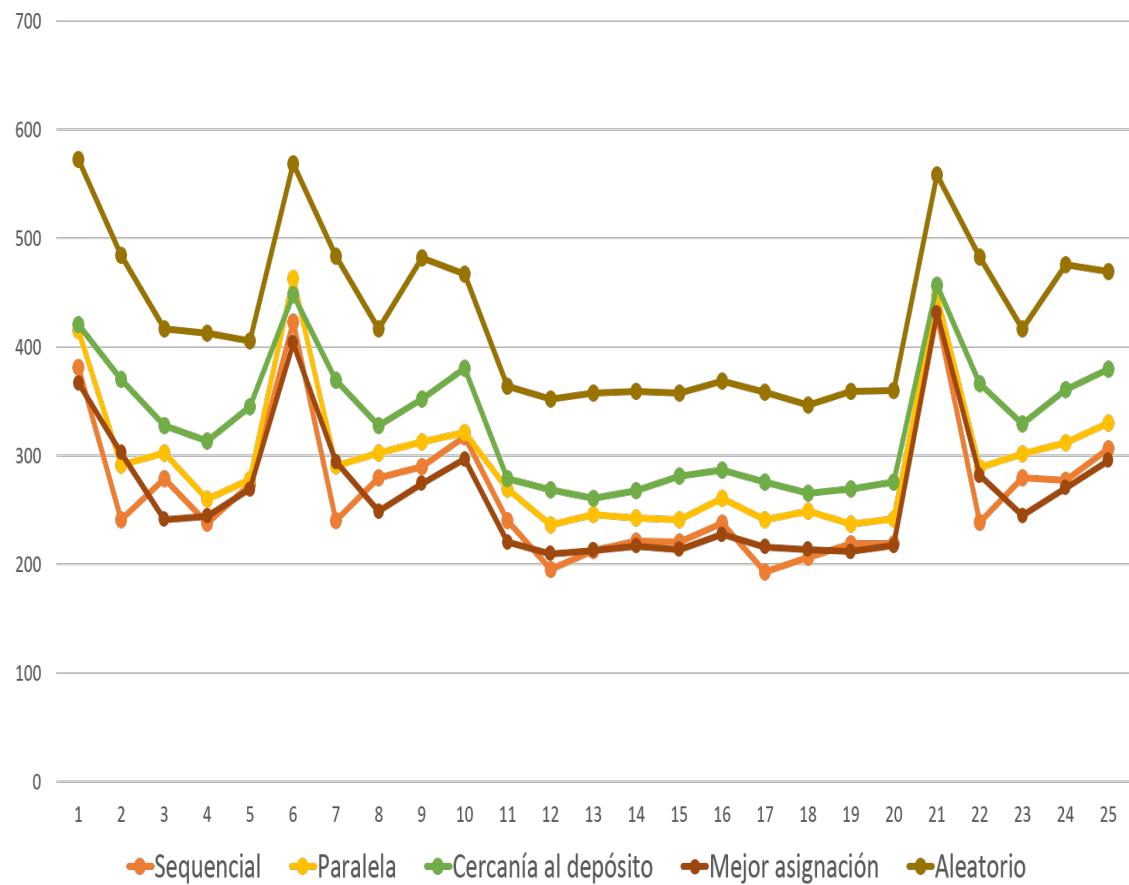


Figura 4.2: Comportamiento de los métodos de asignación en la variante MD.

Un análisis más profundo de ambos métodos de asignación permite identificar que los algoritmos Escalador de Colinas y Escalador de Colinas con Reinicio superan significativamente a los restantes algoritmos. Asimismo, las heurísticas de construcción utilizadas superan al método de construcción aleatorio con un nivel de significancia de 0.05. Ambas conclusiones se pueden contrastar en los gráficos de las Figuras 4.3 - 4.4.

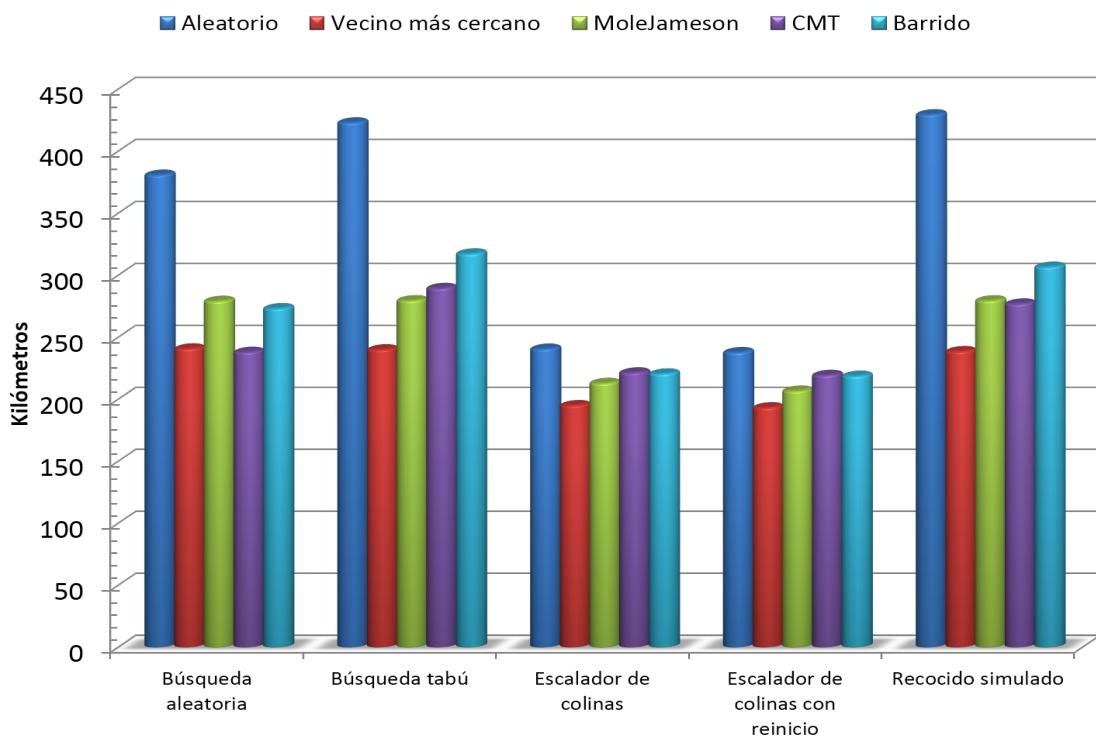


Figura 4.3: Comportamiento del método de asignación Trayectoria Secuencial en la variante MD.

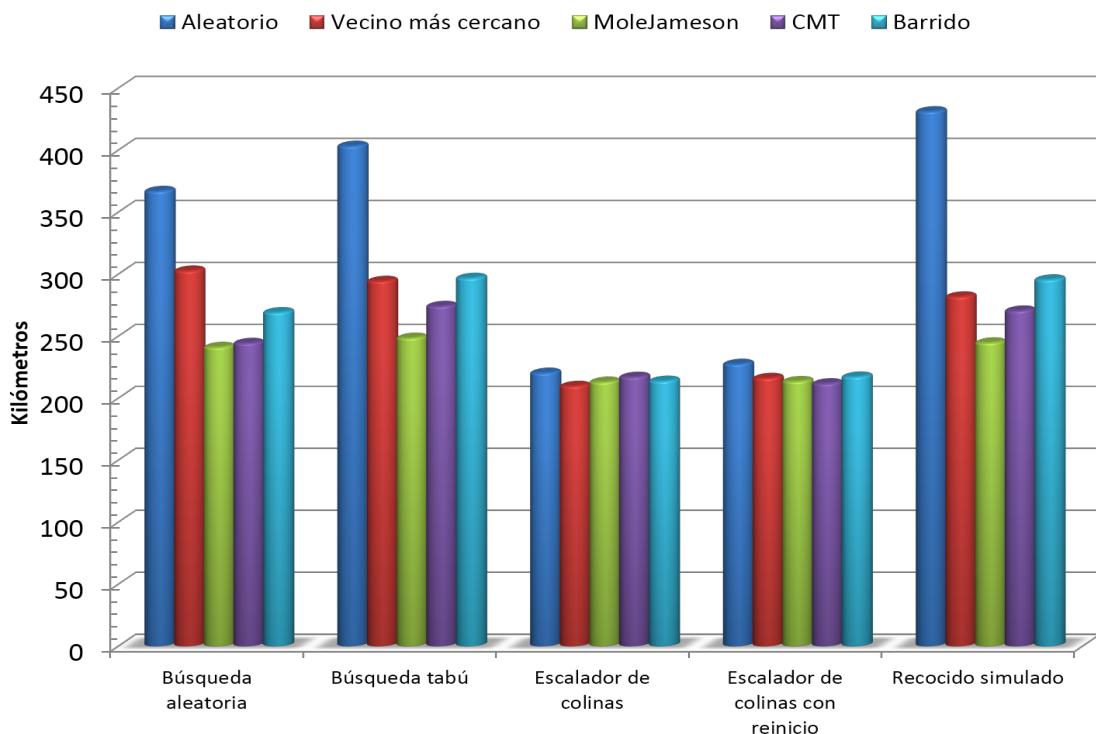


Figura 4.4: Comportamiento del método de asignación Mejor Asignación en la variante MD.

Además, vale mencionar que la mejor solución (193.25 km) se obtuvo con el algoritmo Escalador de Colinas con Reinicio utilizando la heurística de construcción del Vecino más cercano y el método de asignación Trayectoria Secuencial.

4.2.5. Análisis de los resultados para la variante desde la entidad a los depósitos

En el gráfico de la Figura 4.5 se muestra los resultados obtenidos después de haber ejecutado el modelo desde la entidad hacia los depósitos (variante FH) haciendo uso de todas las combinaciones posibles entre los algoritmos metaheurísticos y las heurísticas de construcción incorporadas a la herramienta **TransO**. En todos los casos se realizan 5 ejecuciones y 1000 iteraciones. Los datos con los cuales se ejecutan las pruebas se encuentran descritos en la sección 4.2.2: "Descripción de las instancias".

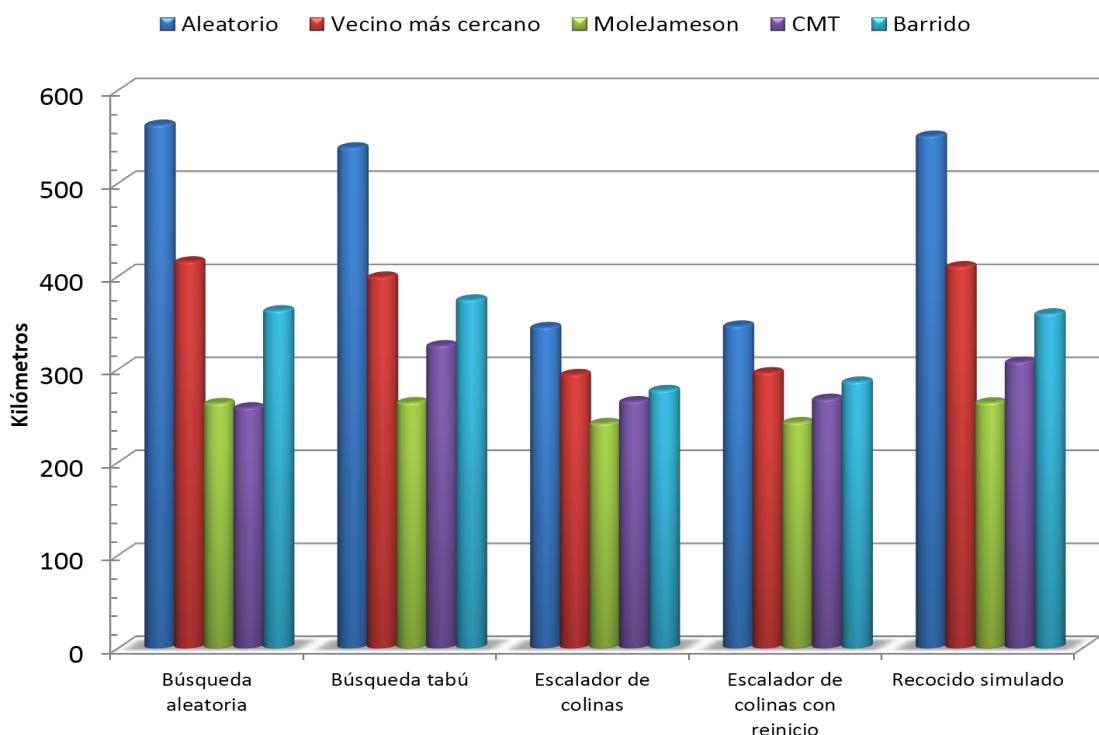


Figura 4.5: Comportamiento de los algoritmos en la variante FH.

Como se puede apreciar la mejor solución (241.746 km) es encontrada con el algoritmo Escalador de Colinas utilizando la heurística de construcción de Mole & Jameson. De manera general, la prueba de Friedman [100] indica que los algoritmos

Escalador de Colinas y Escalador de Colinas con Reinicio superan significativamente a los restantes algoritmos. Asimismo, las heurísticas de construcción: Mole & Jameson, CMT y Barrido superan a las restantes heurísticas de construcción con un nivel de significancia de 0.05.

4.3. Caso de estudios

En esta sección se analiza la aplicación de los modelos de optimización incorporados a la herramienta **TransO** sobre un caso de estudio con diferentes escenarios. Para realizar este caso de estudio se utiliza un fichero xml con los datos necesarios, obtenidos previamente con el personal de transporte y aseguramiento del CITI y para obtener la solución se utiliza la combinación de métodos que mejores resultados arroja en la sección 4.2.4: " Análisis de los resultados para la variante desde los depósitos a la entidad ".

El caso de estudio refleja la situación real del CITI a la hora de transportar a sus trabajadores para así garantizar la mayor seguridad posible en el traslado desde y hacia la misma y aumenta la puntualidad de la plantilla reduciendo el absentismo laboral. La principal función de este servicio de transportación es cumplir con la demanda de cada punto de recogida sin tener la necesidad de transitar por el mismo punto más de una vez. Además, de utilizar los recursos materiales necesarios evitando gastos indebidos.

4.3.1. Escenario A

En este escenario se simula el caso ideal donde los 101 trabajadores deben ser recogidos utilizando los 5 vehículos disponibles de la entidad.

Los resultados obtenidos en este escenario muestran un total de 5 recorridos con una distancia total de 203.07 km. A continuación, en la Tabla 4.3 se muestra para cada vehículo empleado, la descripción del recorrido, la cantidad de pasajeros a recoger y la cantidad de kilómetros a realizar. Además, para un mejor entendimiento de los recorridos en la Figura 4.6 se visualiza en un mapa los mismos.

Tabla 4.3: Descripción de los recorridos obtenidos en el escenario A.

No.	Vehículo	Recorrido	Cant. pasajeros	Kilómetros
1	B103815	Carlos - Ramón y S ^{ta} María - 2 y Vía Blanca - Central y Vía Blanca - Ciudamar y 1 - C. Cienfuego y Porvenir - CITI	14	25.03
2	B033948	Gilberto - 192 y 5 ^{ta} - 5 ^{ta} y 180 - 5 ^{ta} y 174 - 5 y 164 - 15 y 92 - 7 y 158 - 94A y Vía Blanca - Dolores y Delicias - Acosta y 10 Oct - Acosta y Cortina - CITI	17	32.14
3	B103814	Richard - 23 y 222 - 33 y 232 - 39 y 240 - 51 y 234 - 51 y 212 - 51 y 160 - 51 y 190 - 51 y 202 - 114 y 51 - 117 y 100 - 4 y Vento - Acosta y Goicuría - Bosco y San Mariano - Vento y Lacret - 41 y 30 - 41 y 26 - Piñeras y Ayestarán - Domínguez y Ayestarán - 20 Mayo y Ayestarán - J. María y Ave Puerto - Malecón y 27 - M y 23 - G y 19 - 21 y Paseo - 31 y 10 - 41 y 52 - 41 y 86 - 136 y 51 - CITI	42	52.37
4	M000838	Mario - Zusarte y Vía Blanca - Vento y 100 - Varona y Vento - CITI	8	19.51
5	M001565	Alberto - 246 y 251 - 289 y 319 - 405 y 184 - 379 y 184 - 136 y Bejucal - Cotorro y 1 ^{er} Anillo - Concepción y 10 Oct - Juan B. Zayas y Lacret - 5 ^{ta} y 190 - 5 ^{ta} y 280 - 5 ^{ta} y 170 - 330 y 251 - CITI	20	74.02

Como se puede observar en la solución presentada existen tantos recorridos como vehículos disponibles. Además, cada recorrido inicia en los depósitos de los conductores y termina en el CITI. Se puede observar también que se recogen al total de trabajadores que laboran en el CITI. En todos los recorridos se cumplen con la capacidad del vehículo asignado a cada ruta. Por último, se puede observar que el recorrido No. 5 es el de mayor costo aunque no es el vehículo con la mayor cantidad de trabajadores a recoger. Esto es debido a que en este recorrido se concentran aquellos trabajadores que se encuentran más dispersos geográficamente en la capital.



Figura 4.6: Solución obtenida en el escenario A en la variante MD.

4.3.2. Escenario B

En este escenario se simula el caso donde existen dos vehículos averiados técnicamente que le impiden realizar el servicio. Los vehículos con roturas son:

- **Vehículo 1:** B033948 con capacidad para 17 trabajadores.
- **Vehículo 2:** M001565 con capacidad para 29 trabajadores.

De igual forma, se deben transportar los 101 trabajadores de la entidad utilizando solamente los vehículos disponibles.

Los recorridos obtenidos en este escenario son 3 con una distancia total de 238.47 km. A continuación, en la Tabla 4.4 se muestra para cada vehículo empleado, la descripción del recorrido, la cantidad de pasajeros a recoger y la cantidad de kilómetros a realizar. El mapa de la Figura 4.7 visualiza dichos recorridos.

Tabla 4.4: Descripción de los recorridos obtenidos en el escenario B.

No.	Vehículo	Recorrido	Cant. pasajeros	Kilómetros
1	B103815	Carlos - Ciudamar y 1 - Central y Vía Blanca - 5 y 164 - 7 y 158 - 94A y Vía Blanca - Ramón y S ^{ta} María - 2 y Vía Blanca - Dolores y Delicias - Juan B. Zayas y Lacret - CITI	22	39.89
2	B103814	Richard - 5 ^{ta} y 190 - 5 ^{ta} y 280 - 5 ^{ta} y 170 - 114 y 51 - 51 y 190 - 51 y 234 - 39 y 240 - 33 y 232 - 23 y 222 - 330 y 251 - 51 y 212 - 51 y 202 - 51 y 160 - 136 y 51 - 41 y 86 - 41 y 52 - 31 y 10 - 41 y 26 - 41 y 30 - Piñeras y Ayestarán - Domínguez y Ayestarán - 20 Mayo y Ayestarán - 21 y Paseo - G y 19 - M y 23 - Malecón y 27 - Zusarte y Vía Blanca - 246 y 251 - 405 y 184 - 379 y 184 - 289 y 319 - 136 y Bejucal - Cotorro y 1 ^{er} Anillo - Concepción y 10 Oct - J. María y Ave Puerto - 15 y 92 - 192 y 5 ^{ta} - 5 ^{ta} y 174 - C. Cienfuego y Porvenir - Vento y Lacret - Bosco y San Mariano - Acosta y Goicuría - Acosta y Cortina - Acosta y 10 Oct - CITI	70	147.57
3	M000838	Mario - 117 y 100 - 5 ^{ta} y 180 - 4 y Vento - Vento y 100 - Varona y Vento - CITI	9	51.01

En este escenario se viola la restricción de capacidad de los vehículos permitiendo transportar más trabajadores que los admitidos. De esta manera, se brinda la opción de resolver de forma inmediata con los recursos disponibles en la entidad la situación que se presenta.

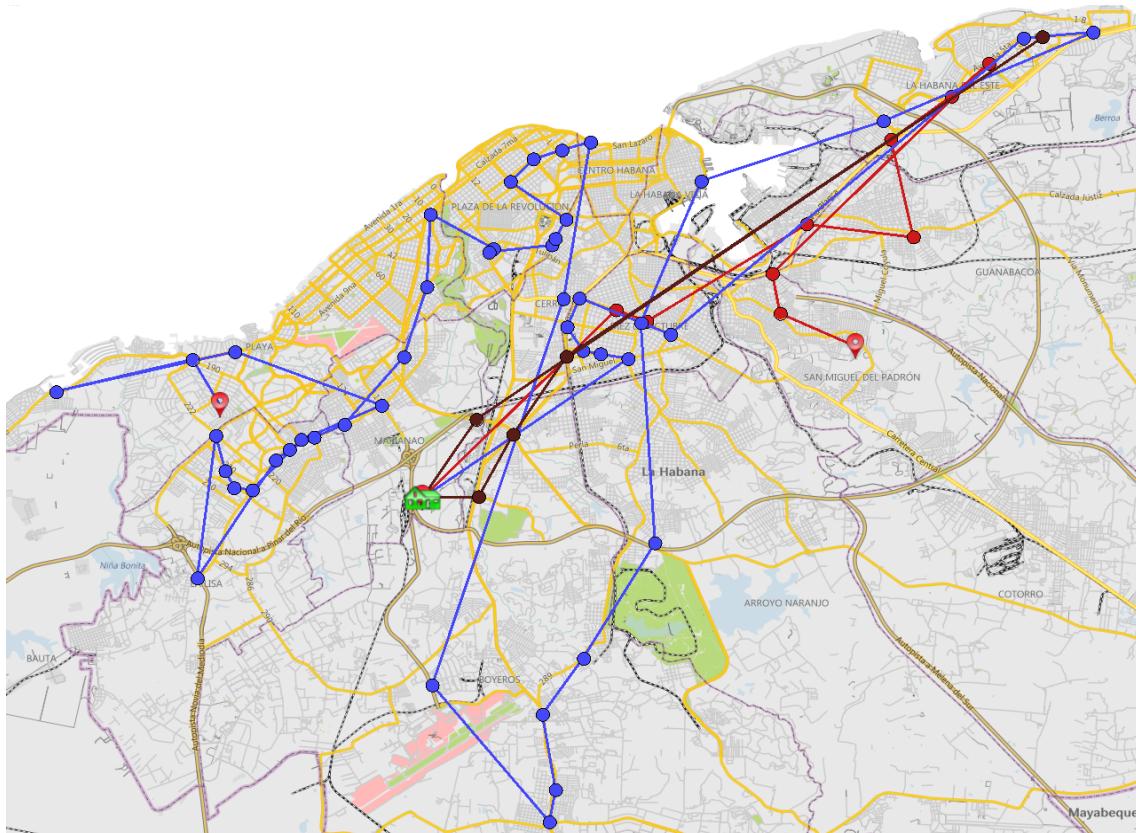


Figura 4.7: Solución obtenida en el escenario B en la variante MD.

4.3.3. Escenario C

En este escenario se simula la introducción de nuevos trabajadores al recorrido, ocasionando a su vez la aparición de un nuevo punto de recogida. En este escenario en particular se introduce el **Punto de Recogida**: "Hospital Naval" con dos nuevos trabajadores que residen en las direcciones:

- **Trabajador 1:** Avenida 9^{na} esq Calle Parqueo J, Camilo Cienfuegos, Habana del Este, La Habana.
- **Trabajador 2:** Calle 40 esq Calle 40 A, Camilo Cienfuegos, Habana del Este, La Habana.

Los resultados obtenidos en este escenario consisten en 5 recorridos con una distancia total de 200.47 km. A continuación, en la Tabla 4.5 se muestra para cada vehículo empleado, la descripción del recorrido, la cantidad de pasajeros a recoger y la cantidad de kilómetros a realizar. El mapa de la Figura 4.8 visualiza dichos recorridos.

Tabla 4.5: Descripción de los recorridos obtenidos en el escenario C.

No.	Vehículo	Recorrido	Cant. pasajeros	Kilómetros
1	B103815	Carlos - Ciudamar y 1 - 2 y Vía Blanca - Ramón y S ^{ta} María - Central y Vía Blanca - Dolores y Delicias - CITI	14	28.03
2	B033948	Gilberto - 5 ^{ta} y 174 - 192 y 5 ^{ta} - 5 ^{ta} y 180 - 5 y 164 - 7 y 158 - 94A y Vía Blanca - 15 y 92 - Hosp Naval - J. María y Ave Puerto - Malecón y 27 - CITI	17	37.43
3	B103814	Richard - 23 y 222 - 33 y 232 - 51 y 234 - 39 y 240 - 51 y 212 - 51 y 202 - 51 y 190 - 51 y 160 - 136 y 51 - 114 y 51 - 41 y 86 - 41 y 52 - Bosco y San Mariano - Acosta y Goicuría - Acosta y Cortina - Acosta y 10 Oct - Concepción y 10 Oct - Vento y Lacret - Domínguez y Ayestarán - Piñeras y Ayestarán - 41 y 30 - 41 y 26 - 20 Mayo y Ayestarán - M y 23 - G y 19 - 21 y Paseo - 31 y 10 - CITI	42	47.21
4	M000838	Mario - 117 y 100 - 4 y Vento - Vento y 100 - Varona y Vento - CITI	8	16.47
5	M001565	Alberto - Zusarte y Vía Blanca - Juan B. Zayas y Lacret - C. Cienfuego y Porvenir - Cotorro y 1 ^{er} Anillo - 136 y Bejucal - 379 y 184 - 405 y 184 - 289 y 319 - 246 y 251 - 5 ^{ta} y 170 - 5 ^{ta} y 190 - 5 ^{ta} y 280 - 330 y 251 - CITI	22	71.33

Como se puede observar en la solución presentada existen tantos recorridos como vehículos disponibles. Además, cada recorrido inicia en los depósitos de los conductores y termina en el CITI. Como se observa en el recorrido No. 2 se ha insertado un nuevo punto de recogida (Hosp Naval) en el cual se recogen dos nuevos trabajadores, esto no impide que se recojan a la totalidad de los trabajadores del centro a los que se les brinda el servicio. Todos estos recorridos cumplen con las capacidades disponibles en cada vehículo destinado a en cada ruta. Por último, se puede observar que el recorrido No. 5 de este escenario se comporta de forma similar a un recorrido del primer escenario.

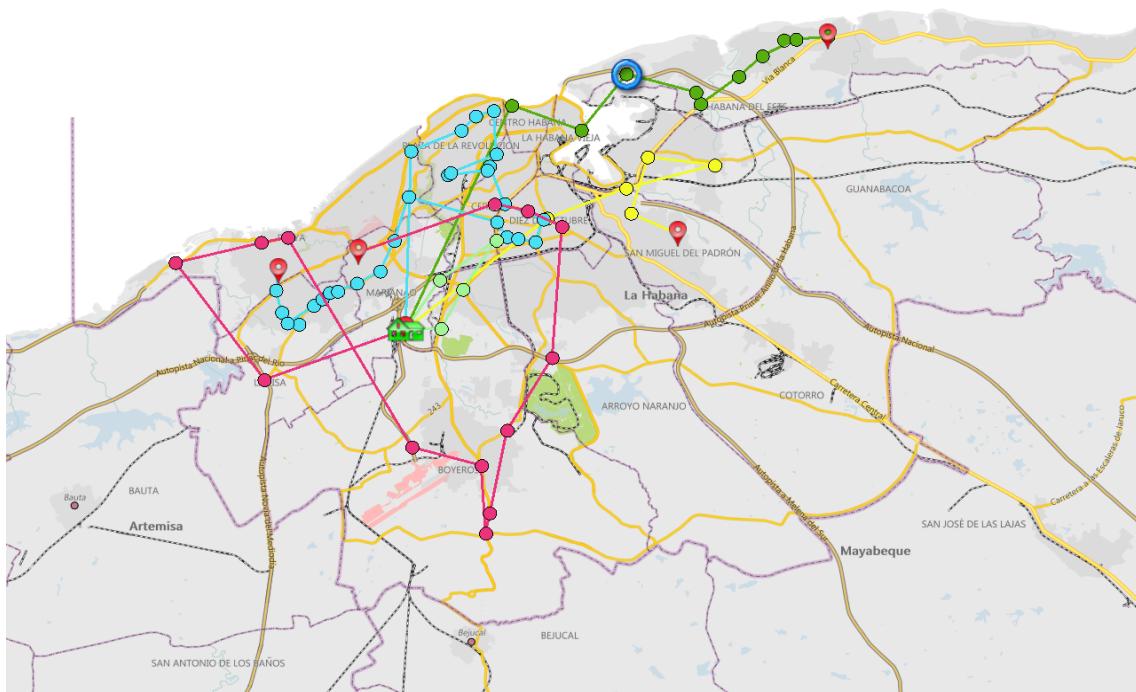


Figura 4.8: Solución obtenida en el escenario C en la variante MD.

4.4. Conclusiones parciales

En el presente capítulo se realiza una validación de la solución propuesta a partir de diferentes casos de pruebas que contemplan todas las combinaciones posibles para las dos variantes implementadas. Para la ejecución de las pruebas se seleccionan los datos extraídos del CITI y se describen los resultados obtenidos analizando

las mejores combinaciones posibles entre los algoritmos metaheurísticos, las heurísticas de construcción y los métodos de asignación. Por último, se hace un análisis de los resultados obtenidos en la herramienta mediante diferentes escenarios de un mismo caso de estudio.

En este capítulo se identificaron y ejecutaron aquellos aspectos que podían ser utilizados para la validación de la herramienta por tanto, se concluye que:

- ✓ Se identifican todas las combinaciones posibles para validar el funcionamiento de la herramienta para las variantes desde los depósitos hacia la entidad y desde la entidad hacia los depósitos.
- ✓ Se demuestra mediante el uso de pruebas estadísticas no paramétricas que en la variante desde los depósitos a la entidad las combinaciones dadas por los algoritmos Escalador de Colinas y Escalador de Colinas con Reinicio con las heurísticas de construcción Mole & Jameson, CMT, Vecino más cercano y Barrido en los métodos de asignación: Mejor asignación y Trayectoria Secuencial superan a las restantes con un nivel de significancia de 0.05.
- ✓ Se demuestra mediante el uso de pruebas estadísticas no paramétricas que en la variante desde la entidad hacia los depósitos las combinaciones dadas por los algoritmos Escalador de Colinas y Escalador de Colinas con Reinicio con las heurísticas de construcción Mole & Jameson, CMT y Barrido superan a las restantes con un nivel de significancia de 0.05.
- ✓ Se puede apreciar que para la aplicación del problema en un contexto real, se puede dar una respuesta inmediata a eventualidades que se presentan como la rotura de vehículos incapacitándolos para cumplir con los recorridos o la aparición de nuevos puntos de recogida debido a la inserción de nuevos trabajadores a la plantilla de la entidad.

Conclusiones

Después de dar cumplimiento a los objetivos trazados en este trabajo y a partir de los resultados alcanzados se puede concluir que los VRP son problemas de optimización que pertenecen a la clase de problemas NP-duro y una alternativa para su resolución es mediante algoritmos heurísticos y metaheurísticos. Existen diferentes variantes VRP, pero muchas de estas por si solas no pueden resolver problemas de la vida real. Sin embargo, la fusión de estas variantes permite modelar situaciones que se presentan en la realidad como el Problema del Transporte Obrero. En la actualidad existen herramientas que permiten resolver ejemplos prácticas de aplicaciones reales como es el Problema del Transporte Obrero. No obstante, la mayoría de estas herramientas son propietarias y algunas no ejecutan algoritmos de optimización para la planificación de rutas. Un aspecto importante es que es común el uso de mapas para visualizar los recorridos en estas herramientas.

Con el propósito de modelar, diseñar e implementar el Problema del Transporte Obrero se obtiene una herramienta denominada **TransO**. Esta herramienta sigue un flujo secuencial donde el usuario transita desde la carga, edición y selección de los datos del problema, la asignación y configuración de los elementos involucrados en el proceso y la ejecución de los algoritmos. Además, hace uso de mapas para visualizar tanto la información geográfica de entrada como los recorridos que se obtienen. Otra facilidad que brinda, consiste en la salva de distintas configuraciones, en diferentes formatos, en cada fase por donde el usuario transita para usos posteriores. Para garantizar la extensibilidad, reusabilidad en el diseño de la aplicación se utilizaron diferentes patrones. Asimismo, se incorporan distintos subsistemas que no son propios de **TransO**, pero son de vital importancia para su funcionamiento. Un análisis de esfuerzo y costo de implementación de este sistema mediante el método de estimación basado en casos de uso, permite afirmar que el desarrollo del sistema

es factible, con un esfuerzo de aproximadamente **22.8 meses** y un costo total que asciende a **5953.066 pesos**.

Después de haber identificado y ejecutado todas las combinaciones posibles para cada una de las variantes y utilizando pruebas estadísticas no paramétricas. Se puede afirmar que para las dos variantes implementadas en la herramienta las combinaciones dadas por los algoritmos Escalador de Colinas y Escalador de Colinas con Reinicio utilizando cualquier heurística de construcción como son Mole & Jameson, CMT, Vecino más cercano y Barrido brinda mejores resultados que al utilizar una heurística de construcción Aleatoria. Además, en la variante desde los depósitos hacia la entidad los métodos de asignación que mejor resultado arrojaron fueron: Mejor asignación y Trayectoria Secuencial. Por último, se puede apreciar que para la aplicación del problema en un contexto real, se puede dar una respuesta inmediata a eventualidades que se presentan como la rotura de vehículos incapacitándolos para cumplir con los recorridos o la aparición de nuevos puntos de recogida debido a la inserción de nuevos trabajadores a la plantilla de la entidad.

Recomendaciones

En el desarrollo de este trabajo de investigación se identificaron algunos aspectos a ser considerados en futuras iteraciones de este proceso de investigación. A continuación se enumeran las principales recomendaciones que se derivan de este trabajo.

1. Modelar el problema haciendo uso de otras representaciones de la solución que permitan el uso de algoritmos poblacionales.
2. Incorporar a las fases de solución de la herramienta la asignación de trabajadores a los puntos de recogidas haciendo uso de algoritmos de Minería de Datos.
3. Incorporar a la herramienta la entrada de datos desde otras fuentes de datos como puede ser un servicio web.
4. Incorporar nuevas conexiones con otros gestores de bases de datos.
5. Incorporar a CaSVRP los modelos implementados para el Problema del Transporte Obrero.
6. Incorporar otros algoritmos de asignación que muestren otras combinaciones para brindar al usuario un amplio espacio de soluciones.
7. Diseñar un componente que encapsule el comportamiento de los métodos de asignación utilizados en la variante de múltiples depósitos para su reutilización en otros problemas que lo requieran.
8. Incorporar a los modelos aspectos relacionados con el factor tiempo como horarios de recogida y duración de los recorridos.
9. Incorporar nuevas funcionalidades relacionadas con el trabajo con mapas.

10. Consumir otros servicios web como son el servicio de rutas y el servicio de geocodificación inversa que brinda la infraestructura de datos espaciales que se encuentra ubicada en el servidor del CITI.
11. Implementar las funcionalidades que no fueron aprobadas en esta iteración de la herramienta.
12. Aplicar en otras entidades la herramienta TransO para validar su aplicabilidad.

Referencias

- [1] R. Diaz, “Nueva versión de la biblioteca de clases biciam para solucionar problemas multiobjetivo,” Junio 2014.
- [2] O. G. Consortium. (2015, Octubre) Ogc. 13/10/2015. Open Geospatial Consortium. [Online]. Available: <http://www.opengeospatial.org/standards>
- [3] Y. Choong, I. Rosmanira, O. Khairuddin, and Z. Mourad, “Vehicle routing problem: Models and solutions,” *Journal of Quality Measurement and Analysis*, vol. 4, no. 1, pp. 205–218, 2008.
- [4] P. Toth and D. Vigo, *The vehicle routing problem*, P. Toth and D. Vigo, Eds. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2002, vol. 9.
- [5] G. Dantzig and J. Ramser, “The truck dispatching problem,” *Management Science*, vol. 6, no. 1, pp. 80–91, 1959.
- [6] J. Daza, J. Montoya, and F. Narducci, “Resolución del problema de enruteamiento de vehículos con limitaciones de capacidad utilizando un procedimiento metaheurístico de dos fases,” *Revista EIA*, vol. 12, no. 1, pp. 23–38, 2009.
- [7] G. Clarke and W. Wright, “Scheduling of vehicles from a central depot to a number of delivery points,” *Operations Research*, vol. 12, no. 4, pp. 568–581, 1964.
- [8] A. Olivera, “Heurísticas para problemas de ruteo,” Master’s thesis, Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, Agosto 2004.
- [9] I.-M. Chao, “A tabu search method for the truck and trailer routing problem.” *Computers & Operations Research*, vol. 29, no. 1, pp. 33–51, Enero 2002.

- [10] S. Scheuerer, "A tabu search heuristic for the truck and trailer routing problem," *Computers & Operations Research*, vol. 33, no. 4, pp. 894–909, 2006.
- [11] S. W. Lin, V. Yu, and S. Chou, "A note on the truck and trailer routing problem," *Expert Systems with Applications*, vol. 37, no. 1, pp. 899–903, Julio 2010.
- [12] E. Delgado, "El problema de la recolección de desechos hospitalarios en la ciudad de guayaquil, modelación y resolución por medio de una heurística basada en la búsqueda tabú," Master's thesis, Departamento de Matemática, Instituto de Ciencias Matemáticas, Guayaquil, Ecuador, 2007.
- [13] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Operational Research*, vol. 59, no. 3, pp. 345–358, 1992.
- [14] J. Cordeau, G. Desaulniers, J. Desrosiers, M. Solomon, and F. Soumis, "Vrp with time windows," in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds. Society for Industrial and Applied Mathematics, 2001, vol. 9, pp. 157–193.
- [15] M. Fisher and R. Jaikumar, "A generalized assignment heuristic for te vehicle routing problem," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.
- [16] R. Baldacci, P. Toth, and D. Vigo, "Exact algorithms for routing problems under vehicle capacity constraints," *Annals of Operations Research*, vol. 175, no. 1, pp. 213–245, 2010.
- [17] C. Hjorring, "The vehicle routing problem and local search," Master's thesis, University of Auckland, Auckland, Nueva Zelanda, 1995.
- [18] I. Gallego, A. Gómez, D. Arguelles, J. Puente, and N. García, "Desarrollo de un método híbrido para la resolución del mdvrp," *Revista de la Escuela Jacobea de Posgrado*, vol. 1, no. 5, pp. 45–64, 2013.
- [19] B. Golden, A. Assad, L. Levy, and F. Gheysens, "The fleet size and mix vehicle routing problem," *Computers & Operations Research*, vol. 11, no. 1, pp. 49–66, 1984.
- [20] H. ku Sugimoto. (2014, 3) Openvrp. Georepublic Osaka. 3-3-138, Sugimoto, Sumiyoshi-ku, Osaka, JAPAN. [Online]. Available: <http://openvrp.com>

- [21] Q. M. Alvina G.H. Kek, Ruey Long Cheu, "Distance constrained capacitated vehicle routing problems with flexible assignment of start and end depots," *Mathematical and Computer Modelling*, vol. 47, no. 1, pp. 140–152, Enero 2008.
- [22] J. Villegas, C. Prins, C. Prodhon, A. Medaglia, and N. Velasco, "A grasp with evolutionary path relinking for the truck and trailer routing problem," *Computers & Operation Research*, vol. 38, no. 9, pp. 1319–1334, 2011.
- [23] E.-G. Talbi, *Metaheuristics: from design to implementation*, ser. Wiley Series on Parallel and Distributed Computing. Hoboken, New Jersey: Wiley-Blackwell an imprint of John Wiley & Sons Ltd, 2009, vol. 74.
- [24] L. Algarra, "Un método cooperativo basado en hheurística simples para el dvrp," Master's thesis, Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, España, 2011.
- [25] D. Paredes and J. Fajardo, "Biblioteca de clases para la unificación de algoritmos metaheurísticos basados en un punto," Trabajo de Diploma, Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echeverría, La Habana, Cuba, 2008.
- [26] W. Fernández, "Nueva versión de la biblioteca de clases biciam," Facultad de Ingeniería Informática, Instituto Superior Politécnico José Antonio Echeverría (CUJAE), La Habana, Cuba, Tech. Rep., 2012.
- [27] S. S. Informáticos. (2016, Marzo) Novatrans software de gestión de flota. 2016. Solbyte Servicios Informáticos. Av. Juan López Peñalver, nº 21, C.P 29590, Campanillas (Málaga). [Online]. Available: www.novatrans.es
- [28] Nunsys. (2007, Agosto) Solución gestión eficaz de rutas escolares. 2007. Nunsys. Calle Gustave Eiffel, 3 Parque Tecnológico 46980 Paterna (Valencia). [Online]. Available: <http://www.traceus.es/>
- [29] F. G. F. E. A. B. Joana Seguí, M. Ruiz, "Sistema de información geográfica para el diseño, gestión, análisis y planificación de rutas de transporte escolar en las baleares (sigtebal)," *Revista Internacional de Ciencia Tecnología de la Información Geográfica*, no. 3, pp. 58–76, Abril 2003.

- [30] O. C. S.A.S. (2016, Febrero) Ontrack corporate. 2016. OnTrack Colombia S.A.S. Av. Ilaló No.148 y Geovani Farina,edificio Mariana de Jesús piso 5 Quito, Ecuador. [Online]. Available: <http://ontrack.global/corporate/>
- [31] D. Soto, Y. Pinzón, and W. Soto , “Una metaheurística híbrida aplicada a un problema de planificación de rutas,” Universidad Nacional de Colombia, Tech. Rep., 2008.
- [32] M. Dror, G. Laporte, and P. Trudeau, “Vehicle routing with split deliveries,” *Discrete Applied Mathematics*, vol. 50, no. 3, pp. 239–254, 1994.
- [33] G. Laporte, M. Gendreau, J. Potvin, and F. Semet, “Classical and modern heuristics for the vehicle routing problem,” *International Transaction in Operational Research*, vol. 7, no. 4–5, pp. 285–300, 2000.
- [34] J. López and S. Nieto, “Heurística para la generación de un conjunto de referencia de soluciones que resuelvan el problema de ruteo de vehículos con múltiples depósitos mdvrp,” in *Tenth LACCEI Latin American and Caribbean Conference (LACCEI 2012), Megaprojects: Building Infrastructure by fostering engineering collaboration, efficient and effective integration and innovative planning*, Panamá, julio 2012.
- [35] Lin, Shih-Wei, Yu, V. F., Chou, and Shuo-Yan, “Solving the truck and trailer routing problem based on a simulated annealing heuristic,” *Computers & Operations Research*, vol. 36, no. 5, pp. 1683–1692, Julio 2009.
- [36] G. L. Nobert, “A branch and bound algorithm for the capacitated vehicle routing problem,” *OR Spektrum*, vol. 5, no. 2, pp. 77–85, Abril 1983.
- [37] T. S. Laporte G., Nobert Y., “Solving a family of multi-depot vehicle routing and location-routing problems.” *Transportation Science*, vol. 22, no. 3, pp. 161–172, Abril 1988.
- [38] F. M. T. P. Carpaneto G., Dell’amico M., “A branch and bound algorithm for the multiple depot vehicle scheduling problem,” *Networks*, vol. 19, no. 5, pp. 531–548, Abril 1989.

- [39] S. N. I. H. F. J. N. H.-P. Jairo R. Montoya-Torres, Julián López Franco, "A literature review on the vehicle routing problem with multiple depots," *Computers & Industrial Engineering*, vol. 79, no. 11, pp. 115–129, Abril 2014.
- [40] C. K. C.D Tarantilis, "A meta-heuristic algorithm for the efficient distribution of perishable foods," *Journal of Food Engineering*, vol. 50, no. 1, pp. 1–9, Octubre 2001.
- [41] C. Prins, "Two memetic algorithms for heterogeneous fleet vehicle routing problems," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 6, pp. 916–928, Septiembre 2009.
- [42] J. Brandão, "A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem," *European Journal of Operational Research*, vol. 195, no. 3, pp. 716–728, Junio 2009.
- [43] C. T. K. Emmanouil E. Zachariadis, "An open vehicle routing problem metaheuristic for examining wide solution neighborhoods," *Computers & Operations Research*, vol. 37, no. 4, pp. 712–723, Abril 2010.
- [44] G. I. P. P. Repoussis, C. D. Tarantilis, "The open vehicle routing problem with time windows," *Journal of the Operational Research Society*, vol. 53, no. 3, pp. 355–367, Marzo 2007.
- [45] . P. S. Sariklis, D., "A heuristic method for the open vehicle routing problem," *Journal of the Operational Research Society*, vol. 51, no. 5, pp. 564–573, Abril 2000.
- [46] J. Brandão, "A tabu search algorithm for the open vehicle routing problem," *European Journal of Operational Research*, vol. 157, no. 3, pp. 552–564, Abril 2004.
- [47] R. W. E. A.N. Letchford, J. Lysgaard, "A branch-and-cut algorithm for the capacitated open vehicle routing problem," *Operational Research Society*, vol. 58, no. 12, pp. 1642–1651, Abril 2007.
- [48] O. de Antonio Suárez, "Una aproximación a la heuristica y metaheuristicas," *Ingeuan*, vol. 1, no. 1, pp. 44–51, enero 2011.

- [49] F. Narducci, "Programación de talleres intermitentes flexibles, por medio de la heurística del margen de tolerancia." Tesis de maestría, Universidad del Norte, Barranquilla, 2005.
- [50] R. Martí, "Procedimientos metaheurísticos en optimización combinatoria," *Mathématiques*, vol. 1, no. 1, pp. 3 – 62, 2003.
- [51] F. Hillier and G. Lieberman, *Introducción a la investigación de operaciones*, novena ed., McGRAW-Hill, Ed. México, D.F.: The McGraw-Hill Companies, 2010.
- [52] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, p. 17, 1986.
- [53] J. Brito, "Optimización de rutas de distribución con información y restricciones difusas," Ph.D. dissertation, Universidad de La Laguna, La Laguna, España, Marzo 2012.
- [54] L. V. S. Quintero and C. A. C. Coello, "Una introducción a la computación evolutiva y alguna de sus aplicaciones en economía y finanzas," *Revista de Métodos Cuantitativos para la Economía y la Empresa*, vol. 1, no. 2, pp. 3–26, 2006.
- [55] A. L. López, I. Eléctrica, C. A. C. Coello, and A. Carsteanu, "Un estudio de las estrategias evolutivas para problemas multiobjetivo," Tesis de Maestria, Departamento de Ingeniería Eléctrica, Sección de Computación, Mexico, D.F, 2003.
- [56] C. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.
- [57] C. Groer, B. Golden, and E. Wasil, "A library of local search heuristics for the vehicle routing problem," *Springer and Mathematical Programming Society*, vol. 2, no. 2, pp. 79–101, 2010.
- [58] L. Kallel, B. Naudts, and M. Schoenauer, "On functions with a given fitness-distance relation," in *Evolutionary Computation*, vol. 3. IEEE, 1999.

- [59] J. F. Calderín, “Biblioteca de clases para integrar algoritmos metaheurísticos basados en un punto,” *Serie Científica*, vol. 2, no. 9, 2009.
- [60] e. o. Alejandro Rosete, “Una solución flexible y eficiente para el trazado de grafos basada en el escalador de colinas estocástico,” Instituto Superior Politécnico José Antonio Echeverría, Tech. Rep., 2000.
- [61] R. Díaz, “Nueva versión de la biblioteca de clases biciam para solucionar problemas multiobjetivo,” Tesis de Diploma, Instituto Superior Politécnico José Antonio Echeverría, junio 2014.
- [62] M. Stutzle, Thomas Dorigo, *ACO Algorithms for the Traveling Salesman Problem*. John Wiley & Sons, 1999, ch. 1.
- [63] D. X. L. Zhihuan, L. Yinhong, “Non-dominated sorting genetic algorithm-ii for robust multiobjective optimal reactive power dispatch,” *IET Generation, vol. 4, no. 9, pp. 1000–1008, Septiembre 2010. [Online]. Available: <http://ieeexplore.ieee.org/document/5551065/>*
- [64] L. del Carmen Machado Lores, “Operadores de mutación basados en heurísticas de construcción para problemas de planificación de rutas de vehículos,” Master’s thesis, Universidad Tecnológica José Antonio Echeverría, Enero 2017.
- [65] A. Infante, “Algoritmos de trayectoria multiobjetivo aplicados al problema de asignación de recursos humanos a equipos de proyecto de software,” Master’s thesis, Instituto Superior Politécnico José Antonio Echeverría, 2012.
- [66] E. Alba and C. Cotta, “Congreso español de algoritmos evolutivos y bioinspirados,” in *Optimización en entornos geográficamente distribuidos. Proyecto MALLBA*, 2002.
- [67] R.Cordone and R.Calvo, *Heuristic for the Vehicle Routing Problem with Time Windows*. Journal of Heuristics, 2001, vol. 7.
- [68] M. Birattari, “Tuning metaheuristics: A machine learning perspective,” *Studies in Computational Intelligence. Berlin/Heidelberg, Germany: Springer*, vol. 197, 2009.

- [69] METSlib. (2014, Febrero) Metslib, metaheuristic framework. [Online]. Available: <http://code.100allora.it/metslib>
- [70] MOMH. (2014, Febrero) Momh multiple-objective metaheuristics. [Online]. Available: <http://home.gna.org/momh/index.html>
- [71] J. Dréo, J. P. Aumasson, and W.Tfaili. (2006, Febrero) Open metaheuristics. [Online]. Available: http://ometah.berlios.de/index.php/Main_Page
- [72] A. Alvarez and Y. González, “Biblioteca de clases para la integración de algoritmos metaheurísticos,” Tesis de Diploma, Instituto Superior Politécnico José Antonio Echeverría, Facultad de Ingeniería Informática, 2009.
- [73] J. Fajardo, “Algoritmo multigenerador de soluciones para la competencia y colaboración de generadores metaheurísticos,” Master’s thesis, Instituto Superior Politécnico José Antonio Echeverría, Facultad de Ingeniería Informática, Julio 2009.
- [74] K. Escalera, “Algoritmos metaheurísticos con estrategias de paralelización aplicadas al problema de asignación de personal a equipos de proyectos de software,” Ph.D. dissertation, Instituto Superior Politécnico José Antonio Echeverría, 2012.
- [75] F. R. J. T. Martin Lukasiewycz, Michael Glaß, “Opt4j - a modular framework for meta-heuristic optimization,” in *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)*, Dublin, Ireland, 2011, pp. 1723–1730.
- [76] F. L. B. D. E. A. Juan J. Durillo, Antonio J. Nebro, “jmetal: a java framework for developing multi-objective optimization metaheuristics,” Departamento de Lenguajes y Ciencias de la Computación E.T.S. Ingeniería Informática, Campus de Teatinos, 29071 Málaga (SPAIN), TECH-REPORT: ITI-2006-10 1, Diciembre 2006. [Online]. Available: <http://neo.lcc.uma.es/staff/paco/pdfs/TECHREP%20ITI-2006-10.pdf>
- [77] P. G. C. Castro, “Stable solving of cvrps using hyperheuristics,” in *GECCO 09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. Montréal, Québec, Canada: ACM, July 2009, p. 255 a 262.

- [78] G.Kendall, P.Cowling, and L. Han, “An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem,” University of Nottingham, Nottingham, Tech. Rep. 1, 2011.
- [79] H. Salazar, “Optimización multiobjetivo aplicado a un problema de ruta corta estocástico,” Ph.D. dissertation, Universidad de las Américas Puebla, 2004.
- [80] J. Puchinger and U. Pferschy, “The core concept for the multidimensional knapsack problem,” in *Evolutionary Computation in Combinatorial Optimization*. Springer, 2006, pp. 195–208.
- [81] D. A. P. Jenny Fajardo Calderín, Antonio D. Masegosa, “An algorithm portfolio for the dynamic maximal covering location problem,” *Memetic Computing*, vol. 8, no. 1, pp. 1–11, Agosto 2016.
- [82] Esri. (2015, 20 Diciembre) Arcgis network analyst. Esri Company. 380 New York Street, Redlands, CA 92373-8100. [Online]. Available: <http://www.esri.es/es/productos/arcgis/>
- [83] A. Rodriguez, “Capa de servicio para soluciones a problemas de ruteo de vehiculos,” Instituto Superior Politécnico Jose Antonio Echeverria, Marianao , La Habana, Cuba, Tech. Rep., Diciembre 2015.
- [84] C. Corporation. (2015, 15 de Diciembre) Transcad transportation planning software. 2015. Caliper Corporation. 1172 Beacon Street, Suite 300 Newton MA 02461-9926, USA. [Online]. Available: <http://www.caliper.com/tcovu.htm>
- [85] Opti-Time. (2014, Enero) Toursolver características técnicas. 2014. [Online]. Available: <http://www.opti-time.com/es/programa-rutas-TourSolver-caracteristicas>
- [86] J. R. I. Jacobson, G. Booch, *El Proceso Unificado de Desarrollo de Software*, 1st ed., S. A. PEARSON EDUCACION, Ed. Addison-Wesley, Julio 2000, vol. 7, no. 1. [Online]. Available: <https://es.scribd.com/doc/50327385/El-Proceso-Unificado-de-Desarrollo-de-Software-Jacobson-Booch-Rumbaugh>

- [87] F. N. D. Piraquive, "Gestión de procesos de negocio bpm (business process management), tics y crecimiento empresarial," *Univ. Empresa, Bogotá (Colombia)*, vol. 7, no. 15, pp. 151–176, Octubre 2008.
- [88] J. L. de la Vara González, "Captura de requisitos de sistemas de información a partir de procesos de negocio y metas," Master's thesis, Universidad Politécnica de Valencia, Junio 2008.
- [89] J. A. P. S. Portillo, "La especificación de requisitos con casos de uso: buenas y malas prácticas," Pontificia Universidad Católica del Perú, Tech. Rep. 1, 2003.
- [90] L. D. Subi, "Nueva versión de la biblioteca de heurísticas de construcción para problemas de planificación de rutas de vehículos," Ph.D. dissertation, Instituto Superior Politécnico José Antonio Echeverría, Junio 2016.
- [91] O. Project. (2016, Febrero) Geotools documentation. 01/2/2016. OSGeo. [Online]. Available: <http://docs.geotools.org/>
- [92] B. Eckel, "Thinking in patterns problem-solving techniques using java.-president, mindview," Inc, Tech. Rep., 2003.
- [93] J. Coplien, *Software Patterns*, 2nd ed., ser. ISBN 1-884842-50-X, T. H. Group, Ed. 71 West 23rd Street, Third Floor New York, New York 10010: SIGS Books and Multimedia, 2000, vol. 1, no. 1.
- [94] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.
- [95] J. Martínez, *Guía de construcción de software en Java con patrones de diseño*. Universidad de Oviedo, Escuela Universitaria de Ingeniería Técnica Informática, 2000.
- [96] G. Booch, *Object Oriented Analysis and Design with Applications.*, 3rd ed., ser. 020189551X, U. A. Wesley, Ed. Addison-Wesley Professional, Abril 2007, vol. 1, no. 1. [Online]. Available: <https://www.amazon.com/Object-Oriented-Analysis-Design-Applications-3rd/dp/020189551X>

- [97] M. C. B. Linda M. Laird, *Software Measurement and Estimation: A practical approach*, primera ed., ser. 2005028945, C. Society, Ed. United States of America: Wiley-Interscience, Enero 2006, vol. Vol. 2, no. 0-471-67622-5. [Online]. Available: https://books.google.com.cu/books?hl=es&lr=&id=3g8wtcpFHZcC&oi=fnd&pg=PR7&dq=Software+Measurement+and+Estimation:+A+practical+approach:+Wiley-Interscience&ots=RdXbcmvp2e&sig=pFnXEeADwdSp35JuO89iQ0-F3zQ&redir_esc=y#v=onepage&q&f=false
- [98] R. S. Pressman, *Ingeniería de software- Un enfoque práctico*, 7th ed., ser. 0-07-709677-0, F. S. A., Ed. Edificio Valrealty, I.a planta, Basauri, 17, 28023 Aravaca (Madrid): The McGraw-Hill Companies, Abril 2010, vol. 7, no. 7.
- [99] G. Myers, *The Art of the Software Testing*, 3rd ed., ser. 2011017548, W. Association, Ed. Canada: John Wiley & Sons, Abril 2011, vol. 1, no. 1.
- [100] M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance,” *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 674–701, 1937.
- [101] C. D. J.A. Pacheco, A. Aragón, “Diseño de algoritmos para el problema del transporte escolar. aplicación en la provincia de burgos,” *Questiío*, vol. 24, no. 1, pp. 52–82, Enero 2000.
- [102] I. Environmental Systems Research Institute. (2016, Agosto) Arcmap. 2016. Environmental Systems Research Institute, Inc. [Online]. Available: <https://desktop.arcgis.com/es/arcmap/latest/extensions/network-analyst/route.htm>
- [103] S. Luke, *Essentials of Metaheuristics (Second Edition)*, second edition ed., L. Com, Ed. 71 Second Street, Suite 300, San Francisco, California, 94105, USA: Lulu Com, Junio 2013, ISBN:978-1-300-54962-8. [Online]. Available: <https://sv2.proxyanonomo.com/browse.php?u=fk1Xd4Iuj9mf7VAjfbQFzKP13HfcctRYWNw%2FskNsSEq7K2%2FVKLFQ%2Bh8BJ7evfYXRuYuxmNxz6w%3D%3D&b=1>

Apéndice A

Comparación de herramientas empleadas para resolver el problema del Transporte Obrero.

El apéndice contiene una comparación entre diferentes herramientas que resuelven el Problema del Transporte Obrero. Para la comparación se consideran los siguientes criterios: lenguaje de programación en que fue desarrollada la herramienta, algoritmos utilizados para la optimización de los recorridos, uso de mapas para la visualización de la información, así como otras funcionalidades que le aporten un valor agregado.

Comparación de herramientas empleadas para resolver el problema del Transporte Obrero.

Herramientas	Lenguaje	Optimización de Rutas	Software Libre	Uso de Mapas	Otras Funcionalidades
Novatrans [27]	C++	Desconocido	X	✓	<ul style="list-style-type: none"> ■ Mantiene actualizado todos los datos sobre sus clientes. ■ Resumen total de facturaciones por clientes. ■ Mantiene un registro completo de los vehículos que utiliza tanto la última verificación técnica, como la cantidad de km que ha recorrido en su tiempo útil. ■ Mantiene un inventario completo de los productos transportados. ■ Registra todos los empleados y el estado de cada cual dependiendo de las horas de conducción y tiempo de descanso.
Traceus [28]	PHP	No ejecuta	X	✓	X
Sigtebal [29]	C++, PHP	Métodos de optimización específico [101], Métodos de cálculo de impedancias [102]	X	✓	X
Ontrack Corporate [30]	C++, PHP	Desconocido	X	✓	X

Apéndice B

Comparación de bibliotecas de clases que implementan métodos heurísticos.

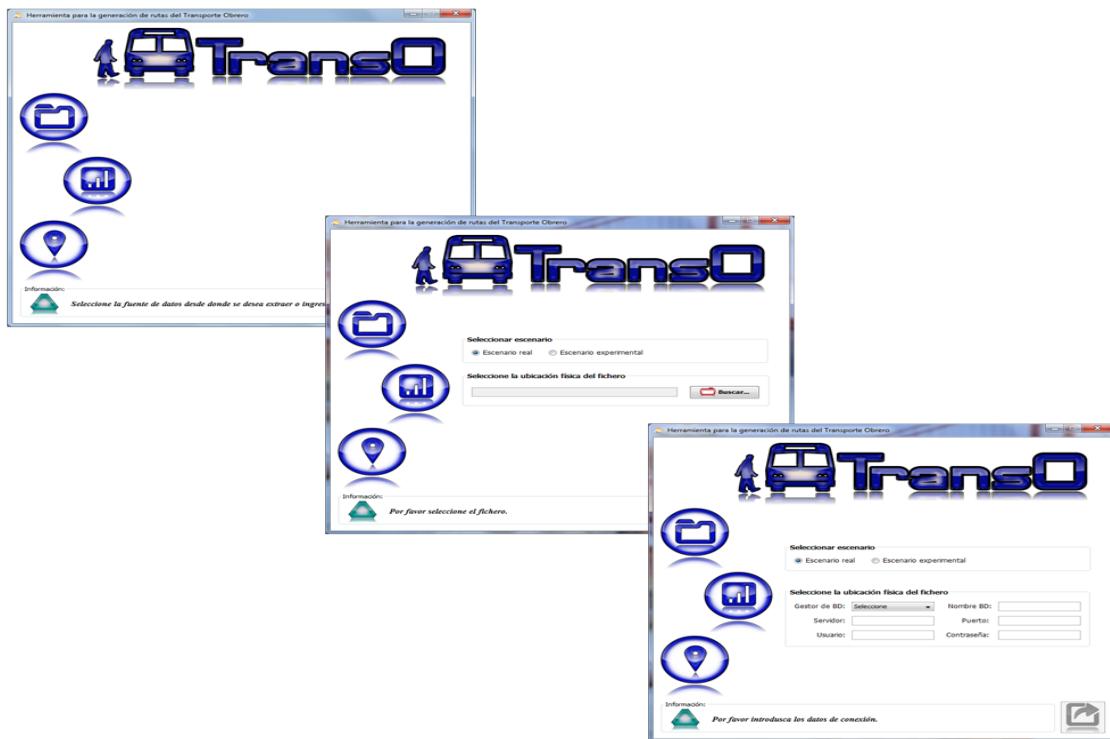
El apéndice contiene una comparación entre diferentes bibliotecas existentes en la literatura que implementan métodos heurísticos. Se presenta dicha comparación utilizando los siguientes criterios: leguaje de programación, algoritmos implementados, tipo de licencia y extensibilidad.

Comparación de bibliotecas de clases que implementan métodos heurísticos.

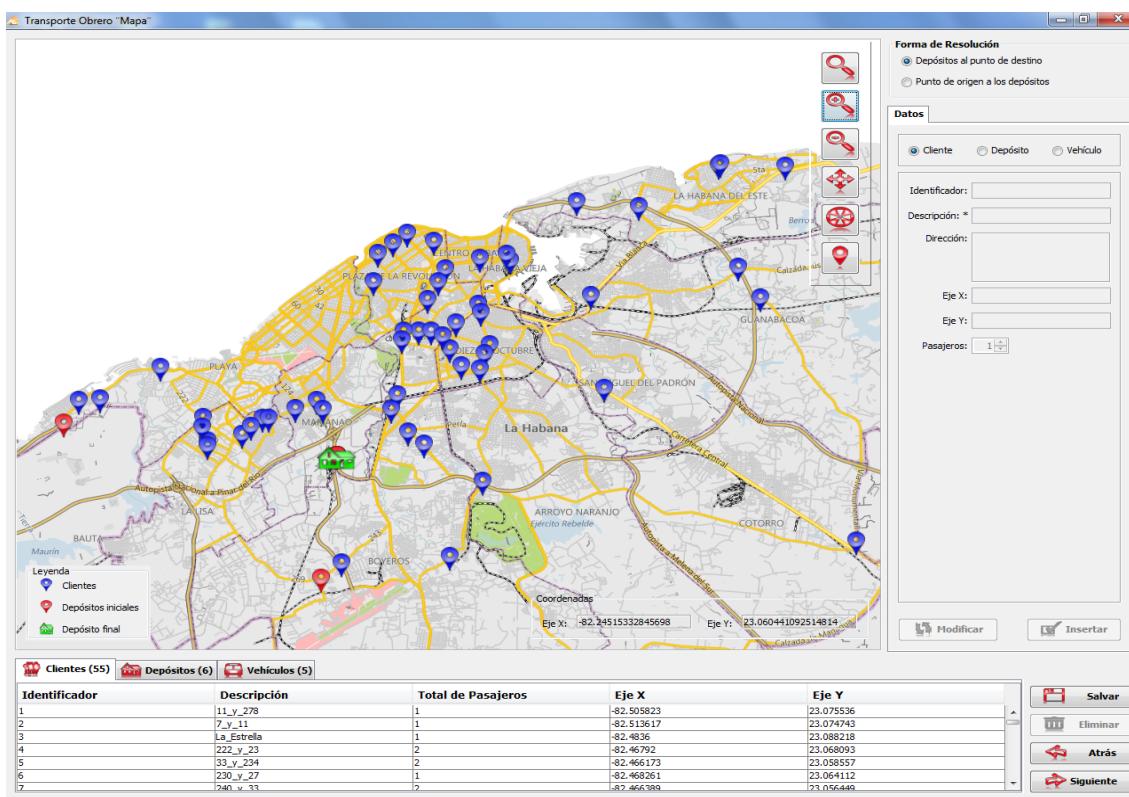
Bibliotecas	Lenguaje	Algoritmos	Licencia	Extensible
BiCIAM [61]	Java	Algoritmo Genético, Estrategia Evolutiva, Escalador de Colinas, Escalador de Colinas con Reinicio, Búsqueda Tabú, Recocido Simulado, NSGA-II, Escalador de Colinas Multi-objetivo, Búsqueda Tabú Multi-objetivo, Multigenerador, PSO, UMOSA, EDA, Recocido Simulado Multi-objetivo	Código abierto	✓
ECJ [103]	Java	Estrategias Evolutivas, Algoritmo Genético, Variantes de Programación Genética, Evolución diferencial, NSGA-II, SPEA2, PSO	Código abierto	✓
JMetal [76]	Java	NSGA-II, SPEA2, PAES, PESA-II, OMOPSO, MOEA/D, MOCell, AbYSS, GDE3, IBEA	GPL	✓
MALLBA [66]	C++	Algoritmo Genético, Recocido Simulado, Estrategias Evolutivas, Colonia de Hormigas, Híbrido de Recocido Simulado con Algoritmo Genético, Búsqueda Local Cooperativa, PSO	Código abierto	✓
METSlip [69]	C++	Búsqueda local aleatoria, Búsqueda con Vecindad Variable, Búsqueda Local Iterada, Recocido Simulado (con enfriamiento lineal, exponencial o personalizado), Búsqueda Tabú	GPL	✓
MOMHLib++ [70]	C++	Recocido Simulado de Pareto, Búsqueda Local Genética Multi-objetivo, Recocido Simulado Multi-objetivo, Algoritmo Memético de Pareto	GPL	✓
Open Metaheuristics [71]	C, C++	Algoritmo Genético, Búsqueda Tabú, Algoritmos Evolutivos, Recocido Simulado	GPL	✓
Opt4j [75]	Java	Algoritmos evolutivos (SPEA2, NSGA), Optimización de enjambre de partículas multi-objetivo, Recocido Simulado mono-objetivo con esquema de enfriamiento predefinido, Evolución diferencial multi-objetivo	LGPL	✓

Apéndice C

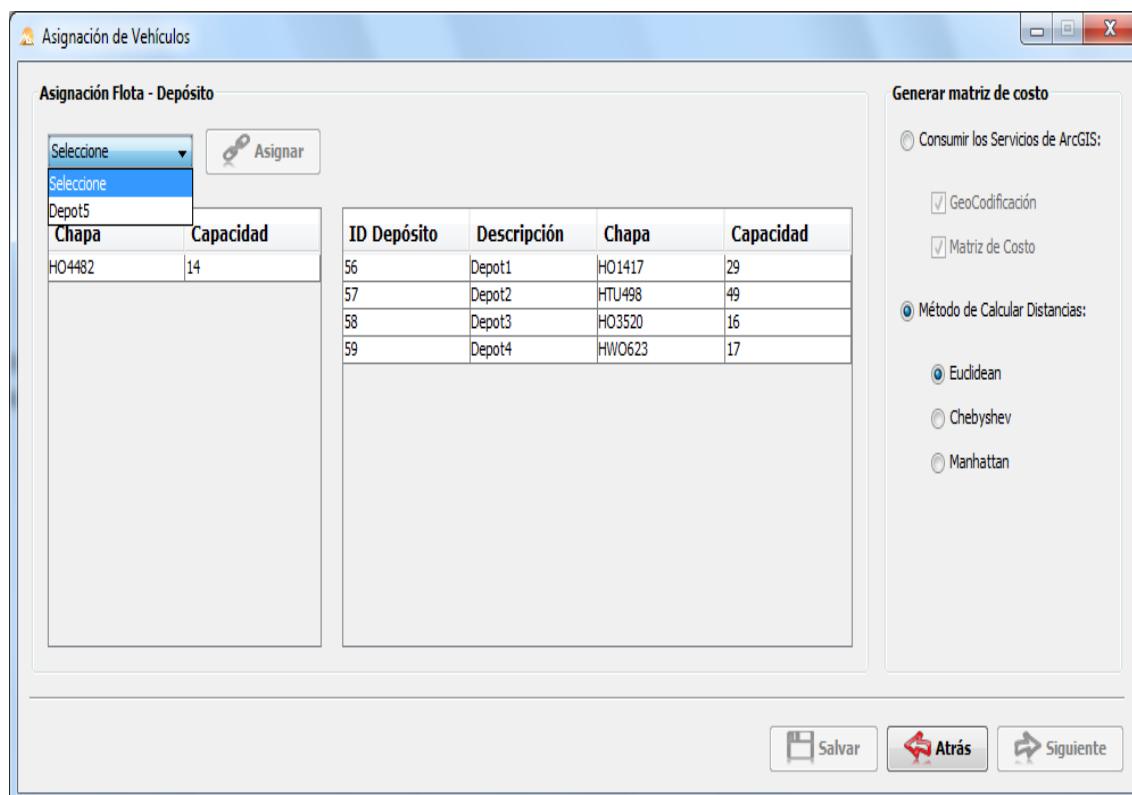
Interfaces gráficas de la herramienta TransO



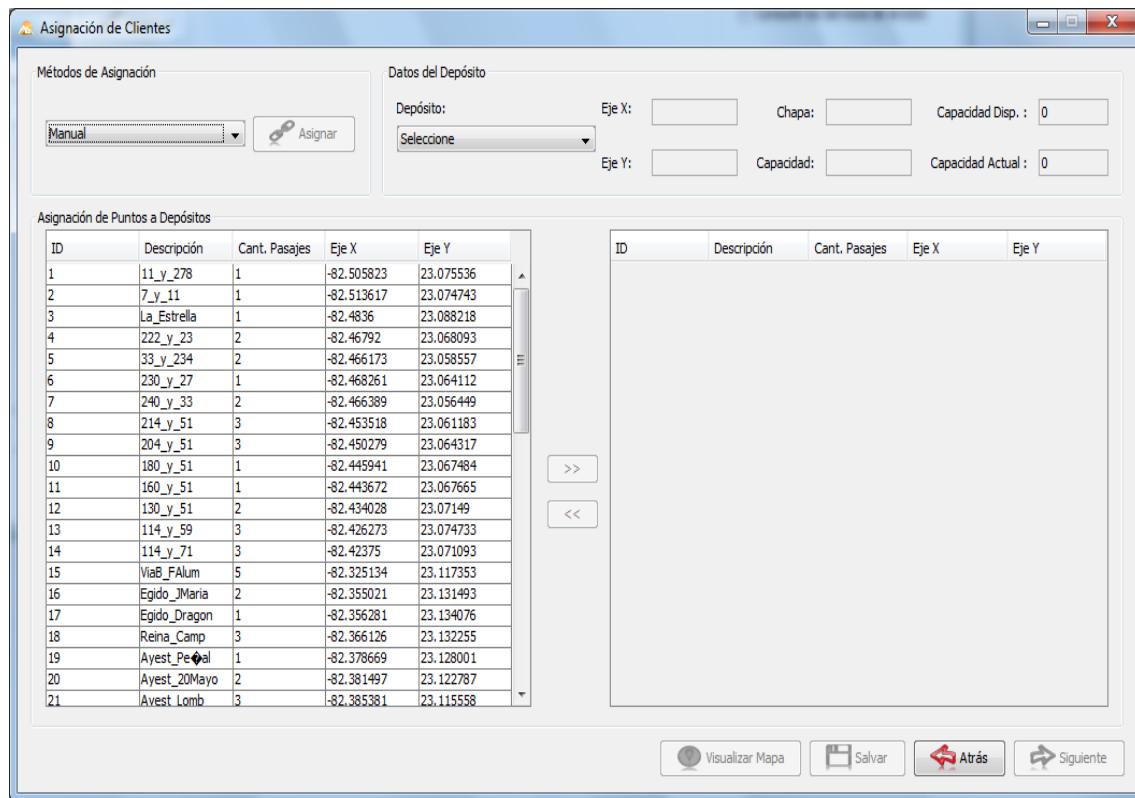
Interfaz gráfica para la carga de datos desde TransO.



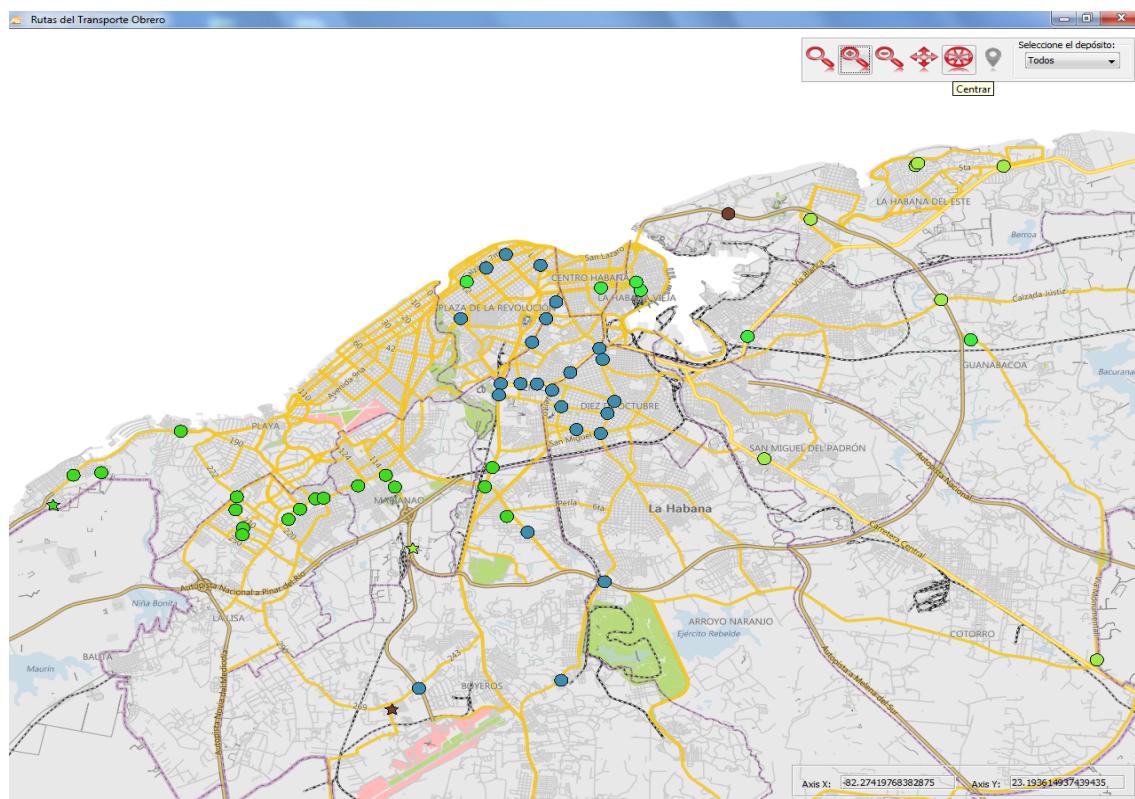
Interfaz gráfica para el trabajo con mapa desde TransO.



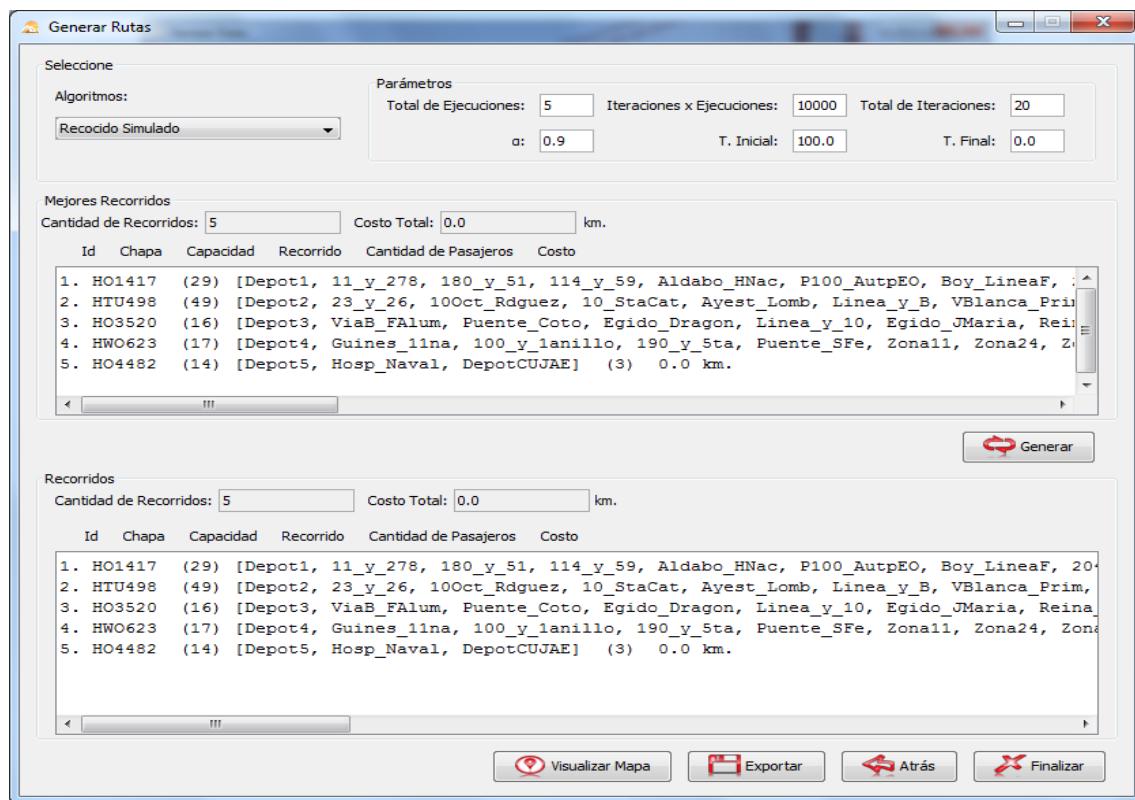
Interfaz gráfica para conformar la matriz de costos desde TransO.



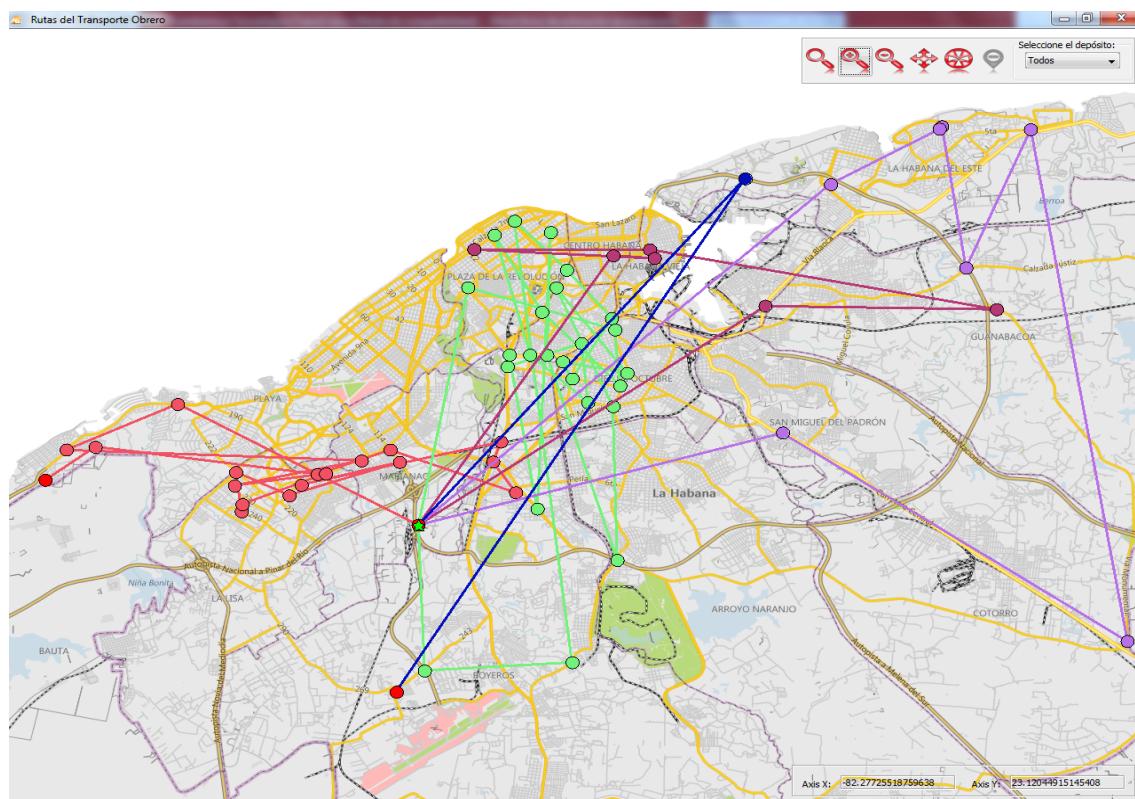
Interfaz gráfica para la asignación de puntos a depósitos desde TransO.



Interfaz gráfica para visualizar la asignación de puntos a depósitos desde TransO.



Interfaz gráfica para la generación de recorridos desde TransO.



Interfaz gráfica para visualizar los recorridos obtenidos en la resolución desde TransO.

Apéndice D

Diagramas de secuencia para la comunicación entre TransO y los subsistemas

El apéndice contiene los diagramas de secuencia de los subsistemas utilizados por la herramienta. Este anexo se divide en tres partes para mostrar los artefactos UML de la herramienta para la generación de rutas del transporte (**TransO**) y su comunicación con los subsistemas utilizados como son la Biblioteca de clases que integra algoritmos metaheurísticos (BiCIAM), Biblioteca de heurísticas de construcción para Problemas de Planificación de Rutas de Vehículos (BHCVRP) y Biblioteca para la ejecución de operadores de mutación en Problemas de Planificación de Rutas de Vehículos (LMOVVRP).



Diagrama de secuencia para la comunicación con BiCIAM.

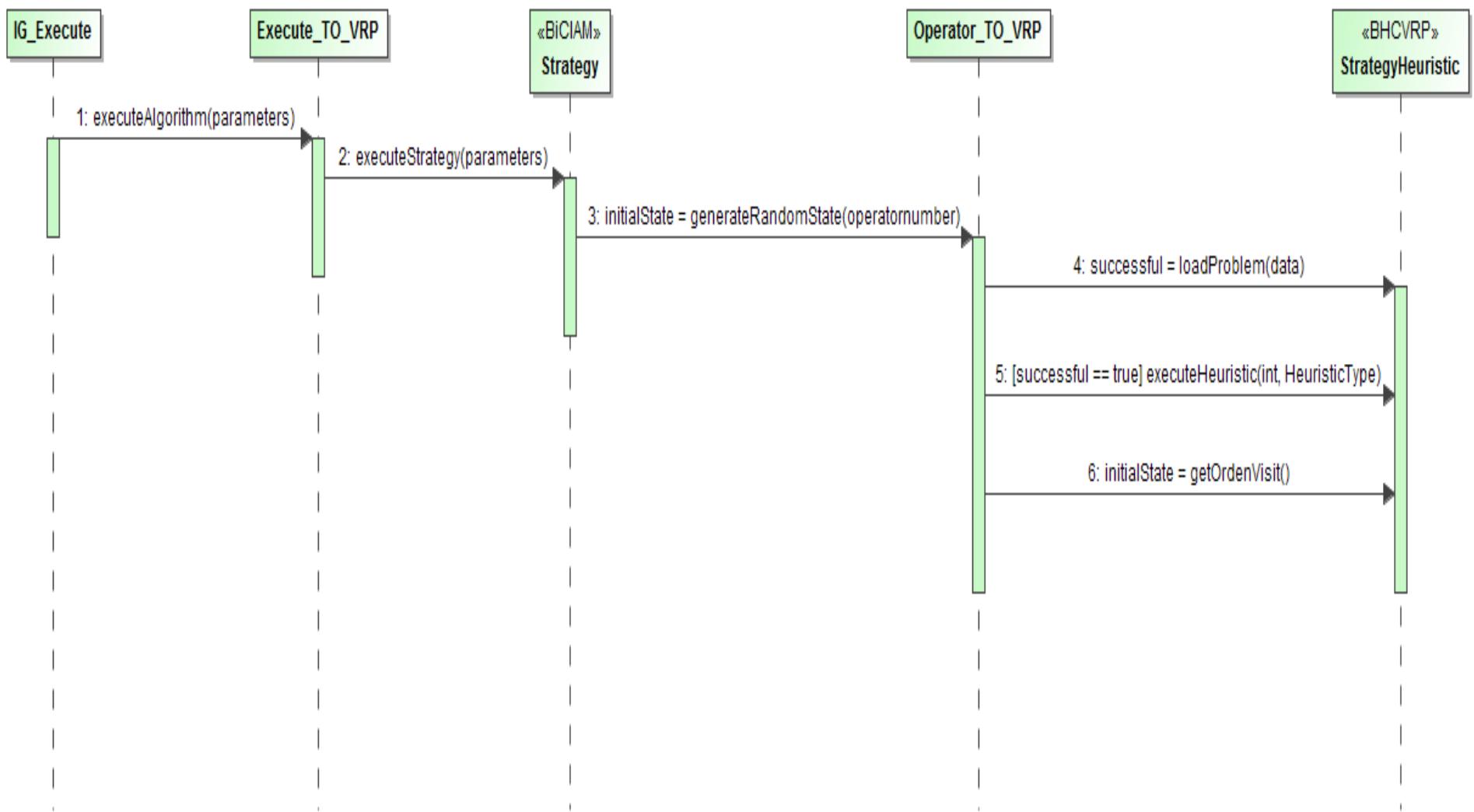


Diagrama de secuencia para la comunicación con BHCVRP.

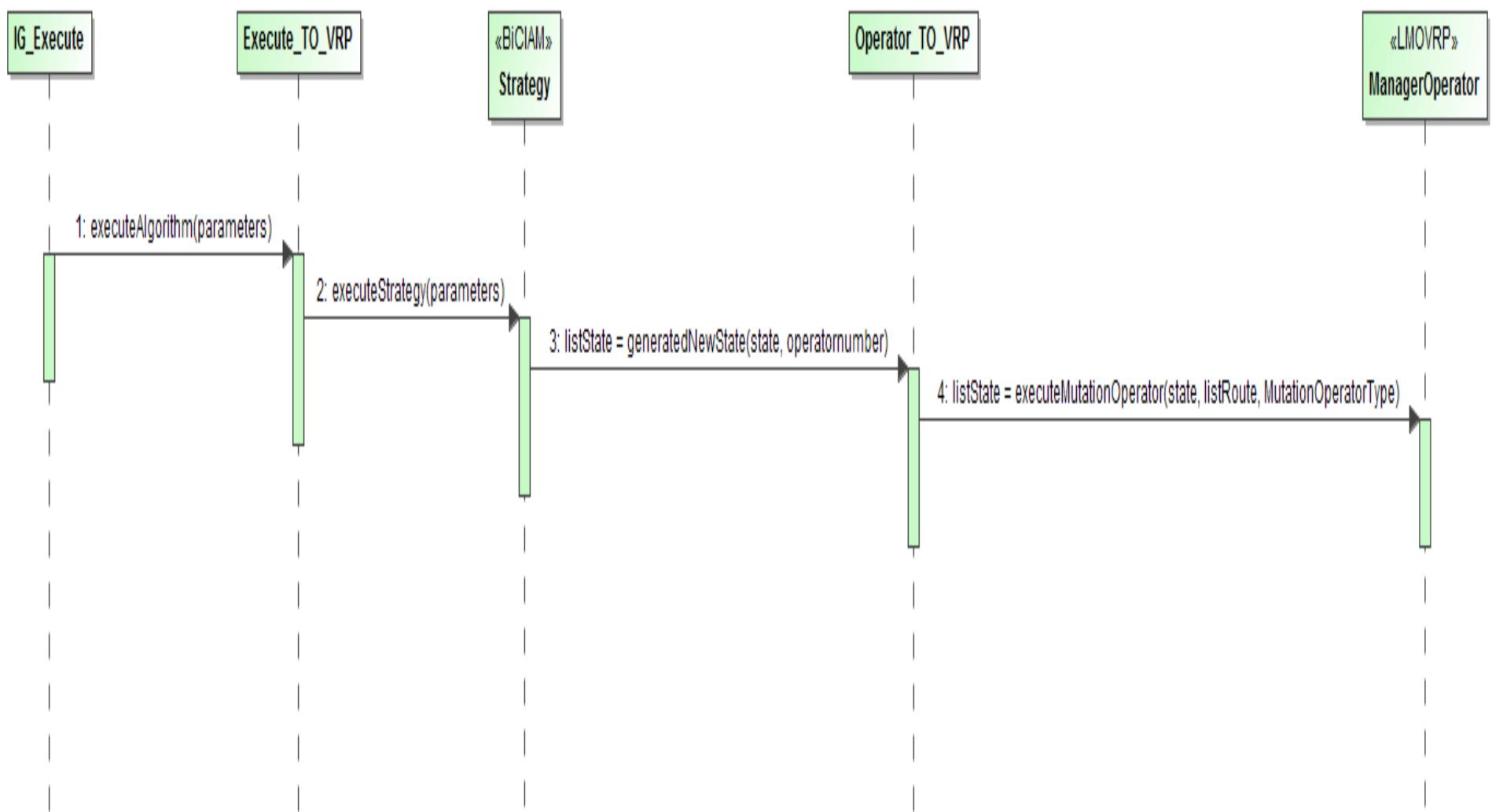


Diagrama de secuencia para la comunicación con LMOVVRP.

Apéndice E

Estudio de factibilidad económica de la propuesta de solución.

El apéndice presenta los pasos realizados para llevar a cabo el estudio de factibilidad del proyecto. El estudio de factibilidad realiza un análisis de esfuerzo y costo de implementación del sistema mediante el método de estimación basado en casos de uso. Los detalles acerca de la estimación se encuentran a continuación:

Calcular los puntos de caso de uso

Los puntos de caso de uso sin ajustar (PCU) se calculan al inicio de un proyecto de software, cuando apenas se conocen los casos de uso y sus actores asociados. Se puede proyectar una breve descripción de cada caso de uso, en el cual se describe de forma breve la funcionalidad que éste debe brindar. Estos puntos son calculados de la siguiente forma:

$$PCU = FPA + FPCU \quad (E.1)$$

Donde:

- **PCU:** puntos de Casos de Uso sin ajustar.
- **FPA:** factor de Peso de los Actores sin ajustar.
- **FPCU:** factor de Peso de los Casos de Uso sin ajustar.

Factor de peso de los actores (**FPA**): Este valor se calcula mediante un análisis de la cantidad de actores presentes en el sistema y la complejidad de cada uno de ellos. Los criterios se muestran en la siguiente tabla:

Criterios de complejidad de los actores.

Tipo de actor	Descripción	Factor de peso
Simple	Otro sistema que interactúa con el sistema a desarrollar mediante una interfaz de aplicación	1
Media	Otro sistema que interactúa con el sistema a desarrollar mediante un protocolo o una interfaz basada en texto	2
Compleja	Una persona que interactúa con el sistema a desarrollar mediante una interfaz gráfica	3

Existe un actor complejo (Usuario) los cuales representan personas que interactúan con el sistema a través de una interfaz gráfica. Además existe un actor medio (Servidor) que actúa con la aplicación a través de comandos y protocolos de comunicación.

$$FPA = CAC * FPCAC + CAM * FPCAM + CAS * FPCAS \quad (E.2)$$

Donde:

- **CAC:** cantidad de actores de complejidad compleja presentes en el sistema.
- **CAM:** cantidad de actores de complejidad media presentes en el sistema.
- **CAS:** cantidad de actores de complejidad simple presentes en el sistema.
- **FPCAC:** factor de peso del actor de complejidad compleja presente en el sistema.
- **FPCAM:** factor de peso del actor de complejidad media presente en el sistema.
- **FPCAS:** factor de peso del actor de complejidad simple presente en el sistema.

Después de haber realizado el cálculo de factor de peso de los actores sin ajustar se tiene como resultado que **FPA = 5**.

Factor de peso de los casos de uso (**FPCU**): Este valor se calcula mediante un análisis de la cantidad de Casos de Uso presentes en el sistema y la complejidad de cada uno de ellos. La complejidad de los Casos de Uso se establece teniendo

en cuenta la cantidad de transacciones efectuadas en el mismo. Los criterios se muestran en la siguiente tabla:

Criterios de complejidad de casos de uso.

Tipo de CU	Descripción	Factor de peso
Simple	El CU contiene de 1 a 3 transacciones	5
Media	El CU contiene de 4 a 7 transacciones	10
Compleja	El CU contiene 8 o más transacciones	15

A continuación se presentan para los 14 casos de uso del sistema la cantidad de transacciones que realiza cada caso de uso, la complejidad de los mismos y el peso que le corresponde.

Factor de peso de los casos de uso atendiendo a su complejidad.

Caso de uso	Transacciones	Complejidad	Peso
Cargar datos del problema	4	Media	10
Ingresar datos del problema	4	Media	10
Obtener recorridos	8	Compleja	15
Gestionar datos del problema	2	Simple	5
Salvar configuraciones	2	Simple	5
Asignar vehículos a depósitos	2	Simple	5
Asignar puntos a depósitos	4	Media	10
Exportar recorridos	2	Simple	5
Visualizar información	8	Compleja	15
Geocodificar dirección	4	Media	10
Calcular matriz de costo	6	Media	10
Calcular matriz de costo con las distancias clásicas	6	Media	10
Calcular matriz de costo con el servicio web	6	Media	10
Configurar parámetros de ejecución	2	Simple	5

$$FPCU = CCUS * FPS + CCUM * FPM + CCUC * FPC \quad (E.3)$$

Donde:

- **CCUS:** cantidad de casos de uso que presentan factor de peso de complejidad simple.
- **CCUM:** cantidad de casos de uso que presentan factor de peso de complejidad media.
- **CCUC:** cantidad de casos de uso que presentan factor de peso de complejidad compleja.
- **FPS:** factor de peso del caso de uso de complejidad simple.
- **FPM:** factor de peso del caso de uso de complejidad media.
- **FPC:** factor de peso del caso de uso de complejidad compleja.

Existen 5 casos de uso de complejidad simple, 7 de complejidad media y 2 de complejidad compleja. Obteniendo como resultado que **FPCU = 125**. Dependiendo de lo antes calculado se obtiene de la suma de **FPA** y **FPCU** que **PCU = 130**.

Calcular los puntos de caso de uso ajustados

Una vez que se tienen los puntos de casos de uso sin ajustar, se debe ajustar este valor mediante la siguiente ecuación:

$$PCUA = PCU * FCT * FA \quad (E.4)$$

Donde:

- **PCUA:** puntos de casos de uso ajustados
- **PCU:** puntos de casos de uso sin ajustar
- **FCT:** factor de complejidad técnica
- **FA:** factor de ambiente

Factor de complejidad técnica (**FCT**): Se estima mediante la cuantificación del peso de un grupo de factores que determinan la complejidad técnica del software donde a cada factor se le asigna un valor de 0 a 5 de acuerdo con la relevancia. La siguiente tabla muestra los factores tenidos en cuenta, el peso y el valor asignado a cada uno de estos:

Valores del factor de complejidad técnica.

Factor	Descripción	Peso	Valor	Comentario
T1	Sistema distribuido	2	0	El sistema es centralizado.
T2	Objetivos de rendimiento o tiempo de respuesta	1	4	Depende de los servicios y de la optimización de las rutas.
T3	Eficiencia del usuario final	1	2	Usuario con conocimientos básicos.
T4	Procesamiento interno complejo	1	5	Complejidad en la matriz de costo y optimizar una ruta.
T5	El código debe ser reutilizable	1	3	Fundamentalmente los aspectos relacionados con el proceso de optimización.
T6	Facilidad de instalación	0.5	0	Pocos requerimientos para instalarlo.
T7	Facilidad de uso	0.5	4	Bajo conocimiento técnico.
T8	Portabilidad	2	4	Cualquier S.O. de Windows.
T9	Facilidad de cambio	1	3	Arquitectura flexible.
T10	Concurrencia	1	1	Solo una persona puede trabajar en él.
T11	Incluye objetivos especiales de seguridad	1	0	No presenta.
T12	Provee acceso directo a terceras partes	1	5	Servicios, LMOVVRP y BHCVRP.
T13	Requiere facilidades especiales de entrenamiento a usuarios	1	0	No se requiere.

$$FCT = 0,6 + 0,01 * \sum (Peso_i * Valor_i) \quad (E.5)$$

Después de haber realizado el cálculo del factor de complejidad técnica se tiene como resultado que **FCT = 0.93**.

Factor de ambiente (**FA**): Para calcular el FA se consideran habilidades, entrenamientos y experiencias del grupo de desarrollo. Se estima de forma similar al FCT. A continuación se muestran los factores tenidos en cuenta, el peso y el valor asignado

a cada uno de estos:

Valores del factor de ambiente.

Factor	Descripción	Peso	Valor
E1	Familiaridad con el modelo de proyecto utilizado	1.5	4
E2	Experiencia en la aplicación	0.5	4
E3	Experiencia en orientación a objetos	1	4
E4	Capacidad del analista líder	0.5	3
E5	Motivación	1	5
E6	Estabilidad de los requerimientos	2	5
E7	Personal a tiempo compartido	-1	5
E8	Dificultad del lenguaje de programación	-1	4

$$FA = 1,4 - 0,03 * \sum (Peso_i * Valor_i) \quad (E.6)$$

De la ecuación antes calculada se obtiene como resultado que **FA = 0.815**. Teniendo en cuenta lo antes calculado se obtiene de la multiplicación de **PCU**, **FCT** y **FA** que **PCUA = 98.5335**.

Calcular el esfuerzo de desarrollo

Para calcular el esfuerzo (E) se utiliza el factor de conversión 20 horas-hombre/Punto de Casos de Uso ya que según Karner el total de los valores alcanzados anteriormente mencionados es menor que 2. En resumen, un punto de caso de uso toma 20 horas-hombre.

Fórmula para calcular el esfuerzo de desarrollo:

$$E = PCUA * FC \quad (E.7)$$

Donde:

- **FC:** factor de conversión estándar es de 20 horas / hombre (H/H).

De la ecuación antes calculada se obtiene como resultado que **E = 1970.67**.

Determinar la distribución del esfuerzo

Para una estimación más completa de la duración total del proyecto, hay que agregar a la estimación del esfuerzo obtenida por los Puntos de Casos de Uso, las estimaciones de esfuerzo de las demás actividades relacionadas con el desarrollo de software.

Según los creadores del método existe una distribución del esfuerzo total entre las principales actividades como se muestra a continuación.

Distribución del esfuerzo.

Tipo de actividad	Porciento	Esfuerzo (H/H)
Análisis	26	1138.609
Diseño	19	832.0607
Implementación	45	1970.67
Pruebas	10	437.9267
Otras actividades	0	0
Total	100	4379.267

Estimación de tiempo de desarrollo

El tiempo de desarrollo (TDes) de un proyecto se calcula dividiendo el esfuerzo (E) realizado entre la cantidad de hombres (CH) involucrados.

$$TDes = E/CH \quad (E.8)$$

Dado que el grupo está integrado por un solo desarrollador el valor de CH es 1.

De la ecuación antes calculada se obtiene como resultado que **TDes = 4379.267**.

Dado que una jornada de trabajo es de 8 horas y 24 días laborales al mes se obtiene un total de 192 horas de trabajo mensuales, dividiendo el TDES (total) entre este fondo de tiempo que es de 192 horas laborables.

$$Tiempo Real = TDes/192 \quad (E.9)$$

De la ecuación antes calculada se obtiene como resultado que el **Tiempo real = 22.8**.

Calcular el costo total

Una vez estimado el tiempo de desarrollo del proyecto y conociendo la cantidad de desarrolladores y el pago que recibe cada uno de estos se puede llevar a cabo una estimación del costo total (C) del proyecto referidos a los recursos humanos, el cual es calculado de la siguiente forma:

$$C = E * CHH \quad (\text{E.10})$$

Donde:

- **CHH:** costo de horas por hombre.

El costo de horas por hombre (**CHH**) es calculado de la siguiente manera:

$$CHH = K * THP \quad (\text{E.11})$$

Donde:

- **K:** coeficiente que tiene en cuenta los costos indirectos (1,5 y 2,0).
- **THP:** tarifa horaria promedio.

La tarifa horaria promedio (**THP**): Es el salario promedio de los trabajadores del proyecto dividido entre 160 horas. Es calculado de la siguiente manera:

$$THP = SP / 160 \quad (\text{E.12})$$

Donde:

- **SP:** salario promedio 145 pesos debido a que es un solo trabajador.

De la ecuación antes calculada se obtiene como resultado que el **THP = 0.90625**.

En resumen al estimar el esfuerzo y costo según el método análisis de puntos de caso de uso se ha llegado a la conclusión que el esfuerzo para desarrollar el proyecto es de aproximadamente **22.8 meses** y el costo total asciende a aproximadamente **5953.066 pesos**.