



Universidad de Jaén

Escuela Politécnica
Superior de Jaén

DESARROLLO DE UNA APLICACIÓN DE MÉTODOS METAHEURÍSTICOS CONSTRUCTIVOS

Autor: Diego Ismael Valdivia Alcalá

Grado: Ingeniería Informática

Directores: Cristóbal José Carmona del Jesús y Ángel Miguel García Vico
Departamento de los directores: Informática

Fecha: 04/06/2024

Licencia CC



C R E A



Universidad de Jaén

Escuela Politécnica Superior de Jaén

Departamento de Informática

Don Cristóbal José Carmona del Jesús y Don Ángel Miguel García Vico, tutores del Proyecto de Fin de Carrera titulado: Desarrollo de una Aplicación de Métodos Metaheurísticos Constructivos, que presenta Diego Ismael Valdivia Alcalá, autorizan su presentación para la defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2024

El alumno:

Los tutores:

TABLA DE CONTENIDOS

| | |
|--|-----------|
| CAPÍTULO 1. INTRODUCCIÓN | 1 |
| 1.1. Descripción del proyecto | 2 |
| 1.2. Objetivos | 3 |
| 1.3. Planificación de tiempos y costes..... | 4 |
| CAPÍTULO 2. ANTECEDENTES..... | 11 |
| 2.1. Introducción al TSP | 12 |
| 2.2. Desafíos del TSP | 12 |
| 2.3. Algoritmos tradicionales para resolver el TSP..... | 14 |
| 2.4. Limitaciones de los algoritmos tradicionales | 15 |
| 2.5. Estado del arte | 16 |
| CAPÍTULO 3. ANÁLISIS | 19 |
| 3.1. Requisitos funcionales | 20 |
| 3.2. Requisitos no funcionales | 20 |
| 3.3. Diagrama de casos de uso..... | 21 |
| 3.4. Gramática del caso de uso..... | 23 |
| 3.5. Diagrama de secuencia..... | 24 |
| CAPÍTULO 4. DISEÑO | 26 |
| 4.1. Arquitectura del sistema..... | 27 |
| 4.2. Vistas necesarias | 28 |
| 4.3. Mockups..... | 29 |
| 4.3.1. <i>Pantalla de inicio</i> | 29 |
| 4.3.2. <i>Pantalla de configuración</i> | 30 |
| 4.3.3. <i>Pantalla de ejecución y visualización</i> | 31 |
| 4.3.4. <i>Pantalla de ejecución completada</i> | 32 |
| 4.3.5. <i>Pantalla de visualización</i> | 33 |
| 4.4. Diagrama de clases..... | 34 |

CAPÍTULO 5. IMPLEMENTACIÓN 38

| | | |
|--------------|--|----|
| 5.1. | Lenguaje de programación utilizado..... | 39 |
| 5.2. | Back-end | 39 |
| 5.2.1. | <i>Modularidad</i> | 40 |
| 5.2.2. | <i>Exportación de la información</i> | 46 |
| 5.2.3. | <i>Importación de ficheros</i> | 48 |
| 5.2.3.1. | <i>Conjuntos de datos</i> | 48 |
| 5.2.3.2. | <i>Informes</i> | 49 |
| 5.2.3.3. | <i>Ficheros de parámetros</i> | 50 |
| 5.2.3.4. | <i>Fichero de configuración</i> | 51 |
| 5.2.4. | <i>Visualización de soluciones</i> | 56 |
| 5.2.5. | <i>Algoritmos</i> | 58 |
| 5.2.5.1. | <i>Constructivos</i> | 58 |
| 5.2.5.1.1. | <i>Sistema de Hormigas (SH)</i> | 58 |
| 5.2.5.1.1.1. | <i>Descripción</i> | 58 |
| 5.2.5.1.1.2. | <i>Pseudocódigo</i> | 63 |
| 5.2.5.1.2. | <i>Sistema de Colonias de Hormigas (SCH)</i> | 63 |
| 5.2.5.1.2.1. | <i>Descripción</i> | 63 |
| 5.2.5.1.2.2. | <i>Pseudocódigo</i> | 66 |
| 5.2.5.1.3. | <i>Sistema de Hormigas MAX-MIN (SHMM)</i> | 67 |
| 5.2.5.1.3.1. | <i>Descripción</i> | 67 |
| 5.2.5.1.3.2. | <i>Pseudocódigo</i> | 71 |
| 5.2.5.1.4. | <i>Sistema de Hormigas Mejor-Peor (SHMP)</i> | 71 |
| 5.2.5.1.4.1. | <i>Descripción</i> | 71 |
| 5.2.5.1.4.2. | <i>Pseudocódigo</i> | 74 |
| 5.2.5.2. | <i>Deterministas</i> | 75 |
| 5.2.5.2.1. | <i>Algoritmo Lin-Kernighan</i> | 75 |
| 5.2.5.2.1.1. | <i>Descripción</i> | 75 |
| 5.2.5.2.1.2. | <i>Pseudocódigo</i> | 79 |
| 5.3. | Front-end..... | 80 |
| 5.3.1. | <i>Vista de inicio</i> | 80 |
| 5.3.2. | <i>Vista de configuración</i> | 83 |
| 5.3.3. | <i>Vista de ejecución y visualización</i> | 87 |
| 5.3.4. | <i>Vista de ejecución completada</i> | 88 |
| 5.3.5. | <i>Vista de visualización</i> | 89 |

| | |
|---|------------|
| CAPÍTULO 6. EXPERIMENTACIÓN | 90 |
| 6.1. Sistema de Colonias de Hormigas (SCH) | 93 |
| 6.2. Sistema de Hormigas Max-Min (SHMM) | 100 |
| 6.3. Sistema de Hormigas del Mejor-Peor (SHMP)..... | 105 |
| 6.4. Comparación entre algoritmos constructivos | 106 |
| 6.5. Algoritmo de Lin-Kerninghan..... | 117 |
| 6.6. Comparación entre SCH y Lin-Kerninghan | 118 |
| CAPÍTULO 7. CONCLUSIONES | 130 |
| 7.1. Conclusión | 131 |
| 7.2. Conocimientos adquiridos | 132 |
| 7.3. Futuros trabajos | 133 |
| ANEXO 1. DETALLES DE LA EXPERIMENTACIÓN..... | 135 |
| ANEXO 2. MANUAL DE USUARIO..... | 153 |
| BIBLIOGRAFÍA | 163 |

LISTADO DE FIGURAS

| | |
|---|----|
| Figura 1. Diagrama Pert. Planificación de tiempos y costes..... | 5 |
| Figura 2. Diagrama de Gantt. Planificación de tiempos y costes..... | 6 |
| Figura 3. Diagrama de Casos de Uso. Análisis | 22 |
| Figura 4. Diagrama de secuencia. Análisis | 25 |
| Figura 5. Arquitectura del sistema. Diseño | 28 |
| Figura 6. Pantalla de inicio. Diseño | 29 |
| Figura 7. Pantalla de configuración. Diseño | 30 |
| Figura 8. Pantalla de ejecución y visualización. Diseño | 31 |
| Figura 9. Pantalla de ejecución completada. Diseño | 32 |
| Figura 10. Pantalla de visualización. Diseño | 33 |
| Figura 11. Diagrama UML. Diseño | 34 |
| Figura 12. Patrón Strategy. Implementación | 41 |
| Figura 13. Patrón Strategy. Implementación | 43 |
| Figura 14. Diagrama de flujo del algoritmo SH | 60 |
| Figura 15. Diagrama de flujo del algoritmo SCH | 64 |
| Figura 16. Diagrama de flujo del algoritmo SHMM | 68 |
| Figura 17. Diagrama de flujo del algoritmo SHMP | 72 |
| Figura 18. Movimiento r-opt Lin-Kerninghan. Implementación | 77 |
| Figura 19. Intercambio secuencial Lin-Kerninghan. Implementación | 77 |
| Figura 20. Intercambio secuencial no posible Lin-Kerninghan. Implementación | 78 |

| | |
|---|-----|
| Figura 21. Implementación vista inicio. Implementación | 80 |
| Figura 22. Resultado vista inicio. Implementación..... | 81 |
| Figura 23. Conjunto de datos erróneo. Implementación | 82 |
| Figura 24. Informe erróneo. Implementación..... | 82 |
| Figura 25. Implementación vista configuración. Implementación | 83 |
| Figura 26. Resultado vista configuración SCH. Implementación | 84 |
| Figura 27. Error parámetro obligatorio. Implementación | 84 |
| Figura 28. Error valor de parámetro incorrecto. Implementación..... | 85 |
| Figura 29. Resultado vista configuración SHMM. Implementación..... | 85 |
| Figura 30. Resultado vista configuración SHMP. Implementación | 86 |
| Figura 31. Resultado vista configuración Lin-Kerninghan. Implementación | 86 |
| Figura 32. Implementación vista ejecución y visualización. Implementación | 87 |
| Figura 33. Resultado vista ejecución y visualización. Implementación..... | 87 |
| Figura 34. Implementación vista ejecución completada. Implementación | 88 |
| Figura 35. Resultado vista ejecución completada. Implementación | 88 |
| Figura 36. Implementación vista visualización. Implementación | 89 |
| Figura 37. Resultado vista visualización. Implementación | 89 |
| Figura 38. Diagrama comparando algoritmos constructivos. Experimentos | 111 |
| Figura 39. Diagrama comparando algoritmos constructivos. Experimentos | 111 |
| Figura 40. Diagrama comparando algoritmos constructivos. Experimentos | 112 |
| Figura 41. Diagrama comparando algoritmos constructivos. Experimentos | 112 |

| | |
|---|-----|
| Figura 42. Diagrama comparando algoritmos constructivos. Experimentos | 113 |
| Figura 43. Diagrama comparando algoritmos constructivos. Experimentos | 113 |
| Figura 44. Diagrama comparando algoritmos constructivos. Experimentos | 114 |
| Figura 45. Diagrama comparando algoritmos constructivos. Experimentos | 115 |
| Figura 46. Diagrama comparando algoritmos constructivos. Experimentos | 115 |
| Figura 47. Diagrama comparando algoritmos constructivos. Experimentos | 116 |
| Figura 48. Diagrama comparando algoritmos constructivos. Experimentos | 116 |
| Figura 49. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 122 |
| Figura 50. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 122 |
| Figura 51. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 123 |
| Figura 52. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 123 |
| Figura 53. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 124 |
| Figura 54. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 124 |
| Figura 55. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 125 |
| Figura 56. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 125 |
| Figura 57. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 126 |
| Figura 58. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 127 |
| Figura 59. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 127 |
| Figura 60. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 128 |
| Figura 61. Diagrama comparando SCH y Lin-Kerninghan. Experimentos..... | 128 |
| Figura 62. Pantalla inicial. Manual de usuario | 155 |

| | |
|---|-----|
| Figura 63. Importación del conjunto de datos. Manual de usuario | 155 |
| Figura 64. Conjunto de datos erróneo. Manual de usuario..... | 156 |
| Figura 65. Selección de algoritmo. Manual de usuario..... | 156 |
| Figura 66. Configuración de parámetros. Manual de usuario | 157 |
| Figura 67. Error de parámetros. Manual de usuario | 157 |
| Figura 68. Pantalla de ejecución del algoritmo. Manual de usuario..... | 158 |
| Figura 69. Pantalla emergente de la solución. Manual de usuario | 158 |
| Figura 70. Pantalla de ejecución terminada. Manual de usuario | 159 |
| Figura 71. Pantalla emergente de la solución. Manual de usuario | 159 |
| Figura 72. Exportación de resultados. Manual de usuario..... | 160 |
| Figura 73. Fichero de resultados. Manual de usuario..... | 160 |
| Figura 74. Importación del fichero de resultados. Manual de usuario | 161 |
| Figura 75. Error en el fichero de resultados. Manual de usuario | 161 |
| Figura 76. Pantalla de visualización del fichero de resultados. Manual de usuario | 162 |
| Figura 77. Pantalla inicial. Manual de usuario | 162 |

LISTA DE TABLAS

| | |
|--|-----|
| Tabla 1. Planificación de tiempos. Planificación de tiempos y costes | 4 |
| Tabla 2. Planificación de costes. Planificación de tiempos y costes..... | 7 |
| Tabla 3. Gramática del caso de uso. Análisis..... | 23 |
| Tabla 4. Vistas necesarias. Diseño | 28 |
| Tabla 5. Ejemplo funcionamiento SH. Implementación | 61 |
| Tabla 6. Parámetros iniciales. Experimentos..... | 91 |
| Tabla 7. Resultados iniciales ejecución SCH. Experimentos..... | 93 |
| Tabla 8. Pruebas hormiga aportante SCH. Experimentos..... | 94 |
| Tabla 9. Resumen pruebas hormiga aportante SCH. Experimentos | 95 |
| Tabla 10. Pruebas tamaño población SCH. Experimentos..... | 96 |
| Tabla 11. Resumen pruebas tamaño población SCH. Experimentos | 97 |
| Tabla 12. Pruebas iteraciones y tiempo ejecución SCH. Experimentos | 98 |
| Tabla 13. Resumen pruebas iteraciones y tiempo ejecución SCH. Experimentos ... | 99 |
| Tabla 14. Resultados iniciales ejecución SHMM. Experimentos | 100 |
| Tabla 15. Pruebas con reinicialización SHMM. Experimentos..... | 101 |
| Tabla 16. Resumen pruebas reinicialización SHMM. Experimentos | 102 |
| Tabla 17. Pruebas con suavizado de feromona SHMM. Experimentos..... | 103 |
| Tabla 18. Resumen pruebas suavizado de feromona SHMM. Experimentos..... | 104 |
| Tabla 19. Resultados iniciales ejecución SHMP. Experimentos..... | 105 |
| Tabla 20. Comparación algoritmos constructivos. Experimentos | 106 |

| | |
|--|-----|
| Tabla 21. Comparación algoritmos constructivos. Experimentos | 107 |
| Tabla 22. Comparación algoritmos constructivos. Experimentos | 108 |
| Tabla 23. Comparación algoritmos constructivos. Experimentos | 109 |
| Tabla 24. Comparación algoritmos constructivos. Experimentos | 110 |
| Tabla 25. Resultados iniciales ejecución Lin-Kerninghan. Experimentos..... | 117 |
| Tabla 26. Comparación SCH y Lin-Kerninghan. Experimentos | 118 |
| Tabla 27. Comparación SCH y Lin-Kerninghan. Experimentos | 119 |
| Tabla 28. Comparación SCH y Lin-Kerninghan. Experimentos | 119 |
| Tabla 29. Comparación SCH y Lin-Kerninghan. Experimentos | 120 |
| Tabla 30. Comparación SCH y Lin-Kerninghan. Experimentos | 120 |
| Tabla 31. Resultados iniciales ejecución SCH. Anexo 1 | 136 |
| Tabla 32. Resultados aporte mejor global SCH. Anexo 1 | 137 |
| Tabla 33. Resultados aporte mejor iteración SCH. Anexo 1 | 138 |
| Tabla 34. Resultados aporte híbrido SCH. Anexo 1 | 139 |
| Tabla 35. Resultados población 5 hormigas SCH. Anexo 1 | 140 |
| Tabla 36. Resultados población 10 hormigas SCH. Anexo 1 | 141 |
| Tabla 37. Resultados población 20 hormigas SCH. Anexo 1 | 142 |
| Tabla 38. Resultados iteraciones y tiempo de ejecución SCH. Anexo 1 | 143 |
| Tabla 39. Resultados iniciales ejecución SHMM. Anexo 1 | 144 |
| Tabla 40. Resultados sin reinicialización SHMM. Anexo 1 | 145 |
| Tabla 41. Resultados con reinicialización SHMM. Anexo 1 | 146 |

| | |
|---|-----|
| Tabla 42. Resultados sin suavizado de feromona SHMM. Anexo 1 | 147 |
| Tabla 43. Resultados con suavizado de feromona SHMM. Anexo 1 | 148 |
| Tabla 44. Resultados iniciales ejecución SHMP. Anexo 1 | 149 |
| Tabla 45. Comparación algoritmos constructivos. Anexo 1 | 150 |
| Tabla 46. Resultados iniciales ejecución Lin-Kerninghan. Anexo 1..... | 151 |
| Tabla 47. Comparación SCH y Lin-Kerninghan. Anexo 1..... | 152 |

LISTA DE ECUACIONES

| | |
|--|----|
| Ecuación 1. Cálculo del gasto amortizado..... | 8 |
| Ecuación 2. Cálculo de la distancia entre dos ciudades..... | 59 |
| Ecuación 3. Cálculo del valor heurístico entre dos ciudades..... | 59 |
| Ecuación 4. Regla de transición | 61 |
| Ecuación 5. Actualización de feromona..... | 62 |
| Ecuación 6. Actualización local de feromona | 65 |
| Ecuación 7. Actualización global de feromona | 65 |
| Ecuación 8. Cálculo de la feromona máxima..... | 69 |
| Ecuación 9. Cálculo de la feromona mínima | 69 |
| Ecuación 10. Actualización de feromona..... | 70 |
| Ecuación 11. Suavizado de feromona | 70 |
| Ecuación 12. Actualización de feromona..... | 73 |
| Ecuación 13. Evaporación de feromona en la peor solución | 73 |
| Ecuación 14. Mutación de feromona | 73 |

LISTA DE CÓDIGOS

| | |
|--|----|
| Código 1. Aplicación del patrón Strategy..... | 42 |
| Código 2. Aplicación del patrón Strategy..... | 44 |
| Código 3. Clonación de instancias de algoritmos | 45 |
| Código 4. Aplicación del patrón Strategy para ejecución en línea de comandos | 45 |
| Código 5. Obtención de una copia de un algoritmo..... | 45 |
| Código 6. Exportación de información..... | 47 |
| Código 7. Lectura de informe | 50 |
| Código 8. Lectura de fichero de parámetros JSON | 51 |
| Código 9. Representación gráfica de una solución | 57 |
| Código 10. Pseudocódigo SH..... | 63 |
| Código 11. Pseudocódigo SCH | 66 |
| Código 12. Pseudocódigo SHMM..... | 71 |
| Código 13. Pseudocódigo SHMP | 74 |
| Código 14. Pseudocódigo Lin-Kerninghan | 79 |

CAPÍTULO 1

INTRODUCCIÓN

Las metaheurísticas, en la actualidad, desempeñan un papel crucial en la resolución de problemas complejos en diversas disciplinas, desde la optimización hasta la inteligencia artificial. Estas estrategias innovadoras ofrecen un enfoque flexible y general para guiar la búsqueda de soluciones de alta calidad en espacios de búsqueda extensos y multidimensionales.

En su esencia, las metaheurísticas representan conjuntos de reglas y principios diseñados para la iteración sistemática y la mejora progresiva de soluciones potenciales. A diferencia de los algoritmos clásicos, estas técnicas no dependen de información específica del problema, lo que las hace adaptables a diferentes contextos y problemáticas. La importancia de las metaheurísticas se destaca en la creciente complejidad de los problemas contemporáneos. Desde la optimización de procesos industriales hasta desafíos en inteligencia artificial y aprendizaje automático, estas estrategias ofrecen soluciones robustas y flexibles. Su capacidad para abordar problemas del mundo real, adaptarse a diversas situaciones y proporcionar resultados de alta calidad las convierte en herramientas esenciales en la toma de decisiones estratégicas y la mejora de procesos en múltiples campos.

La relevancia destacada de las metaheurísticas radica en su capacidad para abordar problemas de alta complejidad y dimensionalidad. Su aplicación abarca campos como la logística, la planificación, el diseño y la toma de decisiones estratégicas en entornos dinámicos. Además, estas estrategias han demostrado ser instrumentos valiosos en la resolución de problemas NP-completos, donde la complejidad computacional tradicionalmente limita las posibilidades de encontrar soluciones óptimas en tiempos razonables.

En resumen, este trabajo consistirá en resolver un problema de la realidad mediante el uso de distintas metaheurísticas, elaborando una interfaz gráfica de usuario (GUI) usable la cual permita la visualización de las distintas soluciones encontradas al problema. La implementación de dicha interfaz resulta de crucial

importancia puesto que su ausencia puede complicar al profano en la materia la aplicación de estas técnicas junto a la visualización de sus resultados, lo que provoca una fuerte barrera de entrada al uso de estas técnicas tan interesantes.

En este capítulo se dará una visión general sobre el proyecto a realizar y los objetivos que se espera conseguir. Además, se incluirá la planificación temporal y la estimación de costes, con el objetivo de evitar retrasos y aumento de costes durante el desarrollo del proyecto. En el capítulo [2](#) se describirá el problema que vamos a dilucidar con el sistema a desarrollar, mencionando la dificultad de su resolución y la elevada complejidad en tiempo de los algoritmos clásicos empleados para ello. Por último, justificaremos la inclusión del uso de metaheurísticas para la resolución de este problema y se mencionarán algunas herramientas utilizadas en la actualidad. En los capítulos [3](#) y [4](#) quedan recogidos todos los aspectos relacionados con la ingeniería del software del proyecto. En el capítulo [5](#) se describe cómo se ha implementado el sistema. En el capítulo [6](#) se realizan las pruebas de la aplicación utilizando varios conjuntos de datos para cada algoritmo implementado para comprobar el funcionamiento y calidad de la aplicación. Finalmente, se hará una comparación de los algoritmos con los resultados obtenidos. En el capítulo [7](#) se indican las conclusiones extraídas de la elaboración de este proyecto y algunas ideas de mejora para el futuro. En última instancia, se han incluido dos anexos: el [Anexo 1](#) detalla los resultados de las experimentaciones llevadas a cabo, mientras que el [Anexo 2](#) proporciona instrucciones sobre cómo utilizar el sistema.

1.1. Descripción del proyecto

El propósito de este proyecto es construir un sistema que sea capaz de resolver un problema de tipo TSP (Travelling Salesman Problem o Problema del Viajante de Comercio) mediante el uso de metaheurísticas constructivas basadas en interacción social y algoritmos deterministas.

Un TSP consiste en, dada una lista de ciudades y las distancias entre cada par de ellas, calcular la ruta más corta posible que pase por cada ciudad solo una vez y regrese a la ciudad de origen. Se trata de un problema de una complejidad muy elevada, concretamente es un problema NP-Hard, por lo que obtener la solución óptima en poco tiempo se considera prácticamente imposible, de ahí a que se opte

por el uso de metaheurísticas, las cuales nos permiten encontrar soluciones óptimas o casi óptimas en tiempos razonables.

El sistema constará de una interfaz, la cual permitirá seleccionar uno de los algoritmos implementados para solucionar el problema. También dispondrá de una opción para seleccionar el conjunto de datos de prueba, así como un apartado para configurar el valor de los parámetros del algoritmo seleccionado. Una vez hecho esto, podría iniciarse la ejecución, la cual mostraría por pantalla la mejor ruta encontrada hasta el momento y su coste. Cuando finalice la ejecución, ya sea porque el usuario la ha detenido o porque el algoritmo ha concluido, se puede guardar la solución encontrada. Esta solución se podrá volver a visualizar posteriormente si se selecciona la opción correspondiente en la pantalla inicial.

Por otro lado, el sistema también se puede ejecutar en la consola de comandos, indicando en un fichero de configuración los algoritmos a ejecutar y sus parámetros. De este modo, comenzaría la ejecución, se mostrarían las soluciones encontradas y se almacenarían en ficheros.

1.2. Objetivos

Los objetivos propuestos a cumplir con el sistema a desarrollar son los siguientes:

1. Revisión del estado del arte de metaheurísticas constructivas.
2. Diseño e implementación de modelos metaheurísticos constructivos.
3. Desarrollo de una interfaz gráfica que integre los modelos desarrollados y facilite su uso.

Para poder conseguir todos los objetivos planteados en este proyecto, en primer lugar, se planteará la división del trabajo a realizar en una serie de tareas o actividades. A continuación, se realizará un estudio para determinar el tiempo de ejecución de cada actividad y determinar cuáles de ellas se pueden realizar de forma simultánea. Finalmente, se establecerán las dependencias entre ellas para asegurar una ejecución fluida del proyecto. Siguiendo esta metodología, se logrará un control y gestión efectivos del proyecto, evitando retrasos temporales y aumentos de costos.

1.3. Planificación de tiempos y costes

Dado que el proyecto va a tener una duración de 300 horas, comenzando el 11 de septiembre y considerando una jornada laboral de lunes a viernes con dos horas diarias de trabajo, se procede a realizar la planificación de tiempos conforme a las etapas indicadas en la [Tabla 1](#).

Tabla 1. Planificación de tiempos. Planificación de tiempos y costes. Elaboración propia.

| Actividad | Precedentes | Duración Estimada |
|--|-------------|-------------------|
| A. Planificación del proyecto | - | 10 horas |
| B. Estudio ingeniería del software | - | 20 horas |
| C. Estudio documentación Java Swing | - | 10 horas |
| D. Implementación vistas principales interfaz | A, B, C | 20 horas |
| E. Implementación de algoritmos y patrón Strategy | A, B | 80 horas |
| F. Integración de algoritmos e interfaz | D, E | 15 horas |
| G. Implementación de exportación de informes | E, F | 5 horas |
| H. Implementación importación de informes | E, F | 10 horas |
| I. Pruebas | G, H | 10 horas |
| J. Experimentos | I | 50 horas |
| K. Documentación | J | 70 horas |
| | Total | 300 horas |

En este momento, se procede a realizar la planificación temporal del proyecto mediante el diagrama Pert de la [Figura 1](#).

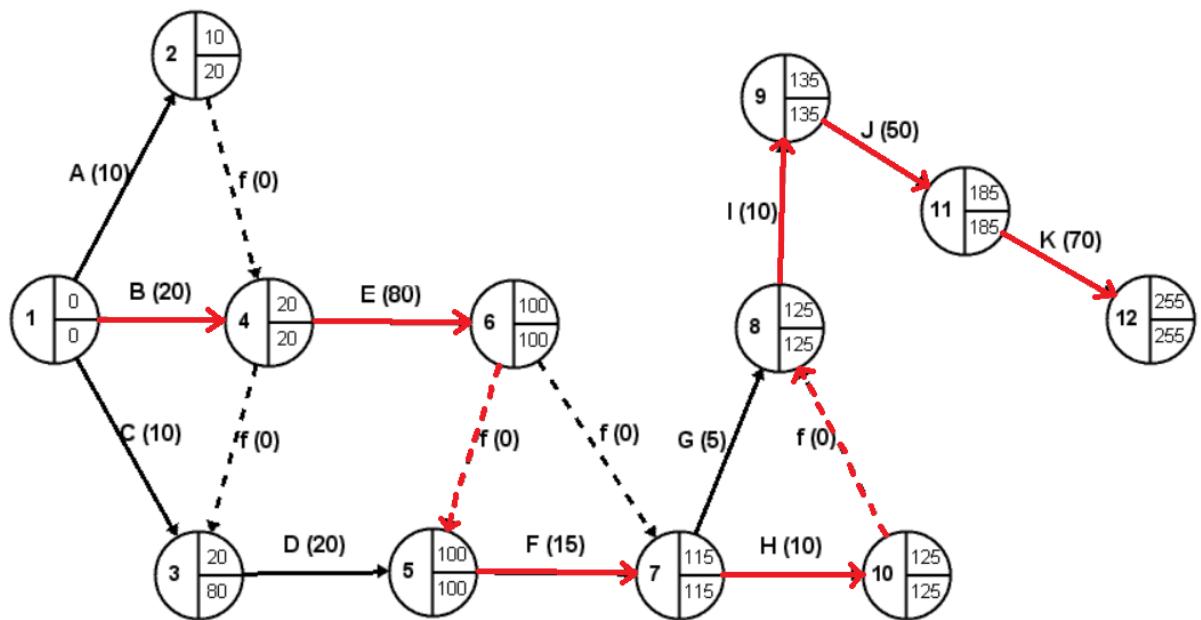


Figura 1. Diagrama Pert. Planificación de tiempos y costes. Elaboración propia.

Al realizar un paralelismo de actividades y seguir la planificación establecida, en lugar de emplear las 300 horas inicialmente previstas para el desarrollo del proyecto, podríamos completarlo en 255 horas. Esto implica que podríamos finalizar el proyecto 22,5 días antes de lo esperado, lo que equivale a 45 horas de trabajo, en caso de llevar a cabo las tareas de forma paralela.

Por otro lado, se observa que el camino crítico está compuesto por las actividades B, E, F, H, I, J y K. Cualquier retraso en alguna de estas actividades afectaría al proyecto, con el consiguiente aumento en los costes. Además, es importante destacar que algunas actividades permiten cierto margen de retraso. Por ejemplo, la actividad A tiene un margen de retraso de 10 horas sin afectar al proyecto en su totalidad. A continuación, en la [Figura 2](#) se presenta un diagrama de Gantt donde se mostrarán cada una de las actividades a desarrollar y su distribución a lo largo del tiempo.

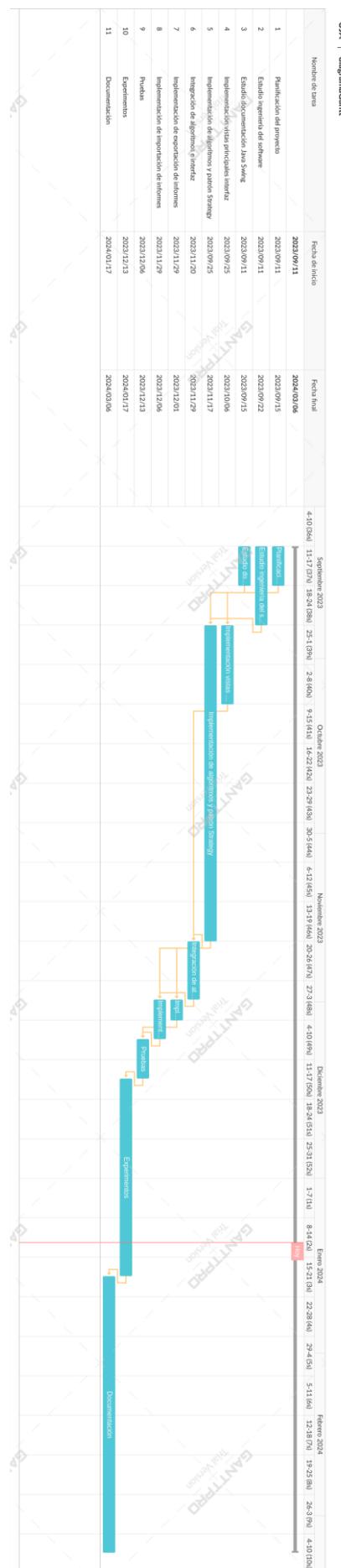


Figura 2. Diagrama de Gantt. Planificación de tiempos y costes. Elaboración propia.

Después de elaborar el cronograma y con la planificación de actividades realizada, se prevé que el proyecto concluya el 6 de marzo d 2024.

La siguiente cuestión a abordar es la estimación de costos del proyecto, centrándonos en los elementos especificados en la [Tabla 2](#).

Tabla 2. Planificación de costes. Planificación de tiempos y costes. Elaboración propia.

| Elemento | Precio (€) |
|--|------------|
| Recursos humanos | 5655 |
| Hardware | 126,90 |
| Material de oficina | 14,18 |
| Software | 2224 |
| Gastos generales | 1937,40 |
| Gestión de proyectos | 13806 |
| Soporte técnico y mantenimiento | 6000 |
| Impuestos | 21% IVA |
| Total | 36013,81 |

Seguidamente, se procede a detallar cada uno de estos elementos:

- **Recursos humanos.**

Considerando que se va a contratar a un único trabajador, en este caso yo mismo, y tomando en cuenta que el sueldo medio de un ingeniero en informática es de 13,85 €/hora («talent.com», 2024), los gastos en salarios ascienden a 5655 €.

El cálculo efectuado es el siguiente:

$$\text{Gastos en salarios} = 300 \text{ horas} * 13,85 \text{ €/hora} = 5655 \text{ €}$$

Cabe destacar que sobre mi figura recaerán todos perfiles involucrados en el desarrollo de un proyecto informático, tales como el director de proyecto, jefe de proyecto, programador, etc.

- **Hardware.**

Es necesario adquirir un ordenador portátil o de sobremesa para el desarrollo y prueba del sistema, el cual tendrá un coste de 1269 € («tiendaLenovo.es», 2023). Dado que este equipo se utilizará en proyectos futuros, se aplicará al cliente un cargo por el gasto amortizado correspondiente al uso del ordenador durante seis meses, calculado a partir de la [Ecuación 1.](#)

$$\text{Amortización Anual} = \frac{C - VR}{n}$$

Ecuación 1. Cálculo del gasto amortizado.

Donde:

- C: coste del equipo.
- VR: valor residual del equipo al final de su vida útil.
- n: vida útil estimada del ordenador en meses.

Suponiendo que el valor residual del equipo es de cero euros y que la vida útil normalmente se considera cinco años (60 meses), se cobrará al cliente:

$$\text{Amortización Anual} = \frac{1269 - 0}{60} = 253,80 \text{ €/año}$$

$$\text{Amortización en seis meses} = \frac{253,8}{2} = 126,90 \text{ €}$$

- **Material de oficina.**

Se necesitan utensilios de escritura para el diseño y la documentación:

- 10 bolígrafos bic: 3,80 € («solocantidad.es», 2023).
- Paquete de folios de formato A4: 9,99 € («amazon.es», 2023).
- Regla: 0,39 € («4pmarketing.es», 2023).

Por tanto, el coste total es de 14,18 €.

- **Software.**

- Entorno de desarrollo integrado (IDE) como Netbeans para el desarrollo del software. Coste: gratuito.

- Software para la realización de diagramas para el análisis y diseño, como VisualParadigm. Coste de la licencia Enterprise Edition: 1999 € («visual-paradigm.com», 2023).
- Software para la realización de mockups como MarvelApp. Coste: gratuito.
- Software para la estimación de tiempos, como EasyPERT. Coste: gratuito.
- Software de ofimática para el desarrollo de la documentación, como Word y Excel. Coste de licencia: 79 € y 149 € («microsoft.com», 2023), respectivamente.

En consecuencia, el coste total es de 2224 €.

- **Gastos generales.**

Suponiendo que la duración del proyecto es de seis meses según la planificación realizada, se estiman estos gastos generales:

- Alquiler de oficina: considerando que mi habitación es la oficina y que se paga por ella 200 €/mes, el coste del alquiler asciende a 1200 €.
- Electricidad: considerando un gasto medio de unos 100 €/mes, el coste en electricidad asciende a 600 €.
- Conexión a internet: 22,90 €/mes («finetwork.com», 2023). Por tanto, 137,40 €.

Por tanto, el coste total es de 1937,40 €.

- **Gestión de proyectos.**

Para garantizar que se cumplan los tiempos y los costes económicos establecidos, se contratará un equipo de gestión de proyectos formado por dos empleados con un sueldo de 23,10 €/hora cada uno («indeed.com», 2023). En este caso, los dos empleados serán mis tutores. En conclusión, contemplando que cada gestor realice 300 horas de trabajo, el desembolso total es de 13806 €.

- **Soporte técnico y mantenimiento.**

Una vez desarrollado el sistema, se ofrecerá al cliente un soporte técnico y un mantenimiento gratuito durante los siguientes tres meses a la entrega. Después de

este periodo, se aplicará una tarifa de 500 €/mes por el soporte del software, en caso de que el cliente lo desee.

- **Impuestos.**

Los costes fijos para desarrollar el proyecto ascienden a un total de 23763,48 euros. Considerando un IVA del 21%, el coste total asciende a 28753,81 €.

Los costes variables, incluyendo 1 año de mantenimiento (3 meses gratuitos), hacen que el coste del proyecto sea de 29763,48 €. Considerando un IVA del 21%, el gasto total del proyecto llega a 36013,81 €.

Cabe recalcar que este precio puede incrementarse por emplear recursos para solventar posibles imprevistos del proyecto. Algunos ejemplos de imprevistos podrían ser el retraso del proyecto, con el consecuente pago del alquiler de la oficina, ya que hemos considerado inicialmente que la oficina se alquila durante seis meses según la planificación realizada, o la necesidad de comprar equipamiento hardware nuevo debido al mal funcionamiento del que se está utilizando.

CAPÍTULO 2

ANTECEDENTES

En esta sección, se proporcionará una introducción al problema que se resolverá mediante el desarrollo del sistema. Luego, se abordará la complejidad de la resolución de dicho problema y se mencionarán algunos algoritmos clásicos que se han utilizado con este propósito. Posteriormente, se destacarán las limitaciones de cada uno de estos algoritmos y se justificará el uso de metaheurísticas para la resolución de este tipo de problemas. Por último, se cerrará el capítulo discutiendo algunas técnicas y herramientas utilizadas en la actualidad para abordar el problema en cuestión.

2.1. Introducción al TSP

El Problema del Viajante de Comercio (Travelling Salesman Problem o TSP, por sus siglas en inglés) es un problema clásico de optimización combinatoria que aborda la siguiente pregunta fundamental: "Dado un conjunto de ciudades y las distancias entre ellas, ¿cuál es la ruta más corta que visita cada ciudad exactamente una vez y regresa al punto de inicio?". Este problema es fundamental en campos como la logística, la planificación de rutas, la fabricación y la distribución, y tiene aplicaciones prácticas en la optimización de recursos y la reducción de costos (Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., 1985).

En el TSP, un comerciante debe encontrar la ruta más eficiente para minimizar la distancia total recorrida. La dificultad del problema radica en su naturaleza combinatoria: con n ciudades, hay $n!$ posibles rutas a considerar. Esto implica que, a medida que el número de ciudades aumenta, la cantidad de soluciones posibles crece de manera exponencial, lo que dificulta la búsqueda de la solución óptima mediante métodos exactos en un tiempo razonable.

En la actualidad, las metaheurísticas han demostrado ser herramientas poderosas para encontrar soluciones aproximadas al TSP en un tiempo razonable. Estas técnicas ofrecen una alternativa eficaz a los métodos tradicionales y han abierto nuevas posibilidades para abordar instancias más grandes y complejas del problema. La combinación de la complejidad inherente del TSP y la capacidad de las metaheurísticas para explorar eficientemente el espacio de soluciones hace que este sea un campo de investigación activo y prometedor.

2.2. Desafíos del TSP

El Problema del Viajante de Comercio (TSP) presenta diversos desafíos que han capturado la atención de la comunidad científica y la industria debido a su relevancia en la optimización combinatoria. Estos desafíos destacan la complejidad del problema y la necesidad de enfoques innovadores para abordarlo.

- **Complejidad computacional.**

El TSP es conocido por ser un problema NP-Hard, lo que significa que no se conocen algoritmos para resolverlo en tiempo polinómico en función del tamaño de la entrada. Esto implica que la búsqueda de la solución óptima se vuelve impráctica para instancias del problema con un número significativo de ciudades, ya que el tiempo de ejecución aumenta considerablemente. Con n ciudades, el número de posibles rutas a considerar es $n!$, lo que lleva a una explosión combinatoria de soluciones. La exploración exhaustiva de todas las posibilidades se vuelve inviable para instancias grandes del problema, ya que se requiere tiempo y recursos computacionales significativos (Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J., 2007).

- **Sensibilidad a la variabilidad de datos.**

La eficacia de las soluciones depende en gran medida de las distancias entre las ciudades. Pequeñas variaciones en los datos de entrada pueden conducir a soluciones sustancialmente diferentes. Esto aumenta la complejidad y la sensibilidad del problema, especialmente en contextos prácticos donde las distancias pueden estar sujetas a cambios (Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J., 2007).

- **Aplicaciones prácticas y relevancia industrial.**

Dada su relevancia en aplicaciones prácticas, el TSP se convierte en un problema crucial para la eficiencia operativa en la logística y la distribución. La necesidad de rutas óptimas se traduce en la búsqueda de enfoques que proporcionen soluciones de alta calidad en tiempo real. Métodos tradicionales, como los algoritmos exactos, pueden volverse ineficientes para instancias grandes, lo que destaca la necesidad de enfoques innovadores y adaptativos (Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J., 2007).

Estos desafíos subrayan la importancia de explorar enfoques alternativos, como las metaheurísticas, que están diseñadas para manejar problemas complejos y ofrecen soluciones aproximadas de alta calidad en un tiempo razonable. El reconocimiento de estos desafíos ha impulsado la investigación activa en el desarrollo de metaheurísticas para el TSP, con el objetivo de superar las limitaciones de los métodos tradicionales y proporcionar soluciones efectivas en la práctica.

2.3. Algoritmos tradicionales para resolver el TSP

El abordaje del Problema del Viajante de Comercio (TSP) ha involucrado a lo largo del tiempo varios enfoques tradicionales que buscan encontrar soluciones óptimas o aproximadas. Estos métodos han establecido una base sólida, pero enfrentan desafíos significativos, especialmente cuando se trata de instancias grandes y complejas del problema.

- **Algoritmos exactos.**

Los algoritmos exactos, como el algoritmo de fuerza bruta o backtracking, buscan encontrar la solución óptima al explorar exhaustivamente todas las posibles combinaciones de rutas. Aunque garantizan la solución óptima, su eficiencia disminuye drásticamente a medida que el tamaño del problema aumenta debido a la complejidad computacional inherente al TSP (Papadimitriou, C. H., & Steiglitz, K., 1982).

- **Heurísticas clásicas.**

Las heurísticas clásicas, como el algoritmo del vecino más cercano, buscan soluciones aproximadas de manera más eficiente al adoptar estrategias de toma de decisiones simplificadas. Sin embargo, estas heurísticas pueden quedar atrapadas en soluciones subóptimas y no ofrecen garantías de optimización global (Papadimitriou, C. H., & Steiglitz, K., 1982).

- **Programación dinámica.**

La programación dinámica es otra técnica utilizada para abordar el TSP. Sin embargo, su aplicación directa puede ser prohibitivamente costosa en términos de recursos computacionales, especialmente para instancias con un gran número de ciudades (Papadimitriou, C. H., & Steiglitz, K., 1982).

Aunque estos enfoques han sido fundamentales para la comprensión y la resolución temprana del TSP, sus limitaciones han motivado la búsqueda de métodos más eficientes y adaptables. Las metaheurísticas representan una alternativa valiosa al proporcionar soluciones de alta calidad en un tiempo razonable, incluso para instancias del TSP que desafían las capacidades de los métodos tradicionales. Este

cambio hacia enfoques más flexibles y escalables ha marcado una transición significativa en el campo de la optimización combinatoria y el TSP en particular.

2.4. Limitaciones de los algoritmos tradicionales

A pesar de la eficacia de los enfoques tradicionales para abordar el Problema del Viajante de Comercio (TSP) en casos de menor complejidad, estas metodologías enfrentan limitaciones sustanciales que han impulsado la necesidad de explorar alternativas más flexibles y eficientes. Además de los desafíos expuestos anteriormente, existen algunas de las limitaciones adicionales de gran relevancia como:

- **Falta de escalabilidad.**

La escalabilidad de los enfoques tradicionales es limitada, lo que significa que tienen dificultades para manejar problemas del mundo real con un gran número de ciudades. Esto los hace menos adecuados para aplicaciones prácticas que involucran rutas complejas y extensas (Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., 1985).

- **Falta de garantías de solución óptima global.**

Aunque los algoritmos exactos garantizan la solución óptima, las heurísticas tradicionales no pueden proporcionar garantías de optimización global. Estas técnicas pueden quedar atrapadas en soluciones subóptimas debido a sus estrategias de búsqueda limitadas (Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., 1985).

- **Dependencia de inicialización y parámetros.**

Algunos enfoques tradicionales pueden depender en gran medida de condiciones iniciales específicas o configuraciones de parámetros, lo que puede afectar la calidad de las soluciones obtenidas (Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., 1985).

- **Inadecuación para problemas dinámicos.**

Los enfoques tradicionales pueden resultar inadecuados para abordar el TSP en situaciones dinámicas donde las condiciones cambian con el tiempo, ya que suelen estar diseñados para problemas estáticos (Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B., 1985).

Estas limitaciones resaltan la necesidad de enfoques más adaptables y eficientes. Las metaheurísticas, al ofrecer estrategias de búsqueda más flexibles y la capacidad de encontrar soluciones aproximadas de alta calidad en un tiempo razonable, representan una alternativa valiosa para superar las restricciones inherentes a los métodos tradicionales en el contexto del TSP. Este cambio hacia enfoques más modernos y dinámicos ha impulsado la evolución del estado del arte en la resolución del TSP.

2.5. Estado del arte

El estado actual de la investigación en metaheurísticas aplicadas al Problema del Viajante de Comercio (TSP) refleja un panorama dinámico y en constante evolución. A continuación, se destacan algunas tendencias y desarrollos significativos:

- **Diversidad de metaheurísticas.**

La investigación ha demostrado una diversidad de metaheurísticas aplicadas al TSP, incluyendo algoritmos genéticos, búsqueda tabú, recocido simulado, optimización por enjambre de partículas (PSO), entre otros. Cada metaheurística tiene sus propias fortalezas y debilidades, y su eficacia puede variar según la naturaleza específica del problema y las características de las instancias del TSP (Blum, C., & Roli, A., 2003).

- **Enfoques híbridos.**

Los investigadores han explorado enfoques híbridos que combinan diferentes metaheurísticas o las integran con técnicas de aprendizaje de máquina. Estos enfoques buscan capitalizar las fortalezas de múltiples técnicas para mejorar la calidad de las soluciones y la eficiencia de la búsqueda (Blum, C., & Roli, A., 2003).

- **Adaptación dinámica.**

Se ha observado un interés creciente en el desarrollo de metaheurísticas que pueden adaptarse dinámicamente a cambios en el entorno o en las características del problema. La capacidad de ajuste automático de parámetros y estrategias de búsqueda mejora la robustez de estas técnicas en situaciones dinámicas (Blum, C., & Roli, A., 2003).

- **Optimización multiobjetivo.**

La optimización multiobjetivo, donde se consideran múltiples objetivos simultáneamente, ha sido una dirección activa de investigación. Las metaheurísticas se aplican para encontrar soluciones que representen compromisos óptimos entre objetivos competitivos, como la minimización de la distancia total y la maximización de la eficiencia en la entrega (Toth, P., & Vigo, D., 2013).

- **Benchmarking y comparación de metaheurísticas.**

Existen esfuerzos continuos para establecer conjuntos de datos de referencia y métricas de rendimiento estándar que permitan la comparación objetiva entre diferentes metaheurísticas. Esto facilita la evaluación y comprensión de la eficacia relativa de diversas técnicas en diferentes contextos (Toth, P., & Vigo, D., 2013).

- **Aplicaciones prácticas y problemas del mundo real.**

La aplicación exitosa de metaheurísticas a problemas del mundo real relacionados con el TSP ha sido un área de enfoque. La adaptación de estas técnicas a entornos logísticos, planificación de rutas de vehículos y otros problemas prácticos demuestra su utilidad en aplicaciones industriales y comerciales (Toth, P., & Vigo, D., 2013).

- **Desarrollo de frameworks y herramientas.**

Se han desarrollado frameworks y herramientas específicas para facilitar la implementación y experimentación con metaheurísticas en el contexto del TSP. Estas herramientas proporcionan entornos de desarrollo eficientes y permiten a los investigadores y profesionales explorar y comparar diferentes enfoques de manera más accesible (Toth, P., & Vigo, D., 2013).

En conjunto, estos avances ilustran la amplitud y la vitalidad del estado del arte en la aplicación de metaheurísticas al TSP. La intersección de la investigación teórica, la experimentación práctica y la aplicación en situaciones del mundo real ha llevado a un campo dinámico y en constante expansión que continúa influyendo en la forma en que abordamos y resolvemos problemas de optimización combinatoria como el TSP.

CAPÍTULO 3

ANÁLISIS

En esta sección, se llevará a cabo un análisis de los requisitos que el sistema debe cumplir, abarcando tanto los aspectos funcionales como los no funcionales. Posteriormente, se procederá a la creación del correspondiente diagrama de casos de uso, la especificación gramatical de cada caso de uso y la elaboración del diagrama de secuencia correspondiente

A continuación, procedemos a realizar el análisis de los requisitos que constituyen el sistema.

3.1. Requisitos funcionales

Los requisitos funcionales establecidos para el sistema incluyen:

- REQF01: El sistema debe ejecutarse tanto de forma interactiva como en la consola de comandos con un archivo de configuración.
- REQF02: El sistema debe permitir al usuario seleccionar el algoritmo a ejecutar de forma interactiva.
- REQF03: El sistema debe permitir al usuario modificar los parámetros de configuración del algoritmo seleccionado.
- REQF04: El sistema debe permitir al usuario visualizar la mejor solución obtenida hasta el momento en el proceso de búsqueda.
- REQF05: El sistema debe permitir al usuario detener la ejecución del algoritmo cuando lo deseé.
- REQF06: El sistema debe permitir al usuario guardar la mejor solución encontrada.
- REQF07: El sistema debe permitir al usuario cargar y visualizar una solución encontrada previamente.
- REQF08: El sistema debe permitir la ejecución concurrente de los distintos algoritmos cuando se ejecute desde la consola de comandos.

3.2. Requisitos no funcionales

Los requisitos no funcionales establecidos para el sistema son los siguientes:

- REQNF01: La interfaz debe ser intuitiva y fácil de usar, según los principios de diseño de usabilidad.
- REQNF02: El sistema debe ser modular y permitir la escalabilidad en el número de algoritmos.

- REQNF03: El tiempo de aprendizaje del sistema por usuario deberá ser menor a 20 minutos.
- REQNF04: El sistema debe contar con un manual de usuario.
- REQNF05: El sistema debe tener una disponibilidad del 99,99% de las veces en que un usuario intente acceder.
- REQNF06: El tiempo para iniciar o reiniciar el sistema no podrá ser mayor a 1 minuto.
- REQNF07: El sistema será desarrollado para Windows, MacOs y Linux.
- REQNF08: Se utilizará el lenguaje de programación Java.
- REQNF09: El nuevo sistema se acogerá a las reglas de las licencias generales públicas (GNU), es decir, será gratuito, de código abierto, permitiendo a cualquier persona modificar el software, sin patentes y sin garantías.
- REQNF10: La aplicación podrá gestionar instancias de TSP de diferentes tamaños sin degradación significativa del rendimiento.

3.3. Diagrama de casos de uso

Un diagrama de caso de uso es un diagrama que muestra los casos de uso representados en forma de elipses y a los actores en forma de personajes. También representa las diferentes relaciones entre los actores y los casos de uso. Estos diagramas describen bajo la forma de acciones y reacciones el comportamiento del sistema desde el punto de vista del usuario. Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno. Son descripciones de la funcionalidad del sistema independientes de la implementación y están basados en lenguaje natural.

Los actores son agrupaciones uniformes de personas, sistemas o máquinas que interactúan con el sistema. Son externos al sistema que vamos a desarrollar, por lo que nos permiten delimitar el sistema y definir su alcance. Un usuario puede acceder al sistema con distintos perfiles, según la forma en que intervenga en el sistema.

Para este sistema, el diagrama de casos de uso es el que se muestra en la [Figura 3.](#)

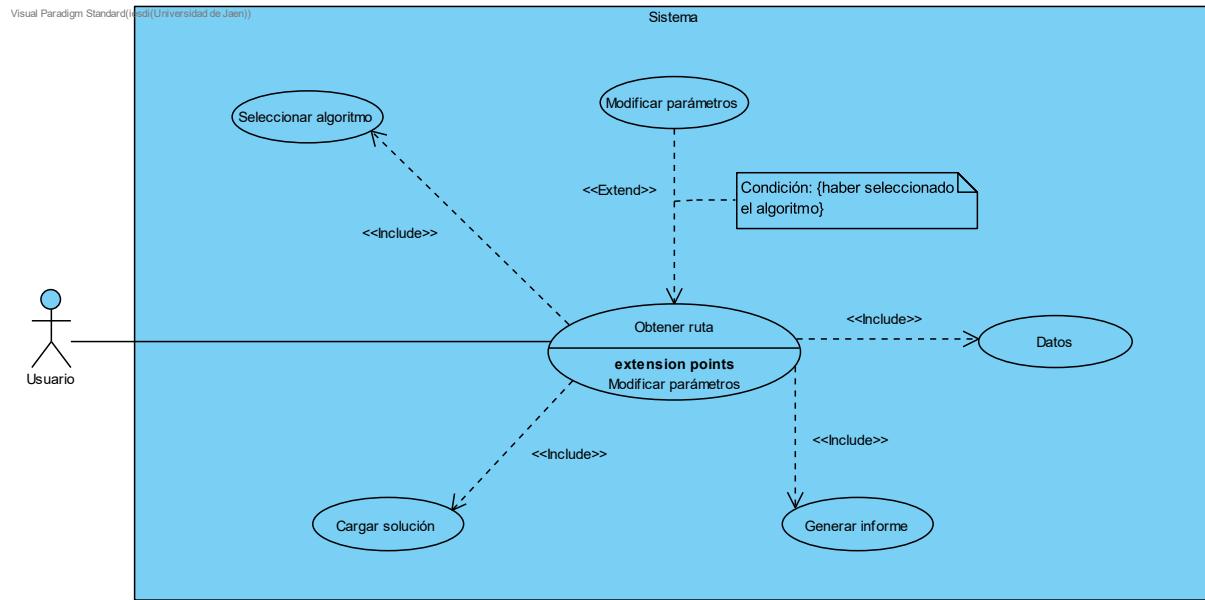


Figura 3. Diagrama de Casos de Uso. Análisis. Elaboración propia.

Como podemos verificar, se define una relación de comunicación entre el actor que interactúa con el sistema y el caso de uso “obtener ruta”. El actor es único y corresponde al usuario que ejecuta el sistema informático. Además, se representan los distintos requisitos que conforman el sistema en elipses, entre ellos: seleccionar el algoritmo, modificar sus parámetros, añadir el conjunto de datos de prueba, generar el informe de resultados y cargar una solución.

Los requisitos etiquetados como “include” son casos de uso utilizados para enriquecer al caso de uso general “obtener ruta”. Estos casos de uso representan funcionalidades que se aplicarán siempre que se utilice el sistema. Por otro lado, los requisitos etiquetados como “extend” son también casos de uso utilizados para enriquecer al caso de uso general “obtener ruta”, pero en este caso, se trata de funcionalidades opcionales. Por ejemplo, la funcionalidad de modificar parámetros no es obligatoria ya que estos tienen un valor por defecto, permitiendo al usuario no editarlos si no lo desea.

3.4. Gramática del caso de uso

Una gramática de un caso de uso pretende dar una descripción de los componentes, acciones y reacciones de los casos de uso. Representa los distintos escenarios que pueden producirse y consta de los siguientes componentes: nombre del caso de uso, actor primario, sistema al que pertenece el caso de uso, participantes (conjunto de actores), nivel del caso de uso: objetivo usuario o subfunción, condiciones previas, operaciones básicas del escenario principal y alternativas.

La gramática para el caso de uso definido en el apartado anterior se presenta en la [Tabla 3](#).

Tabla 3. Gramática del caso de uso. Análisis. Elaboración propia.

| | |
|---------------------------------|---|
| Caso de uso | Calcular ruta |
| Protagonista | Usuario |
| Sistema | Aplicación |
| Participantes | Usuario |
| Nivel | Objetivo usuario |
| Precondición | Haber seleccionado el algoritmo, configurado sus parámetros y proporcionado los datos |
| Operaciones Básicas | |
| 1 | Seleccionar el algoritmo a ejecutar |
| 2 | Configurar los parámetros de ejecución para el algoritmo seleccionado |
| 3 | Incorporar los datos |
| 4 | Generar el informe con los resultados |
| 5 | Cargar una solución |
| Operaciones Alternativas | |
| 1.A | ¿Se ha seleccionado un algoritmo? |
| 1.A.1 | Si sí, continuar |
| 1.A.2 | Si no, repetir paso 1 |
| 2.A | ¿Están correctamente configurados los parámetros del algoritmo? |
| 2.A.1 | Si sí, continuar |
| 2.A.2 | Si no, repetir paso 2 |
| 3.A | ¿Se han incorporado los datos necesarios para ejecutar el algoritmo? |
| 3.A.1 | Si sí, continuar |
| 3.A.2 | Si no, repetir paso 3 |
| 5.A | ¿Se quieren comprobar los resultados obtenidos? |
| 5.A.1 | Si sí, continuar |
| 5.A.2 | Si no, volver a paso 1 o terminar |

El protagonista del caso de uso general “calcular ruta” es el usuario que utiliza el sistema. Este caso de uso pertenece a la categoría de objetivo usuario, ya que constituye el principal objetivo del usuario y no se trata de una subfunción del sistema. La precondición para utilizar el sistema es que el usuario haya seleccionado el algoritmo, configurado sus parámetros y elegido el conjunto de datos de prueba.

En la sección de operaciones básicas, definimos las operaciones que debe realizar el usuario para utilizar el sistema: seleccionar el algoritmo a ejecutar, configurar sus parámetros, incorporar los datos de prueba, generar el informe de resultados y cargar las soluciones obtenidas.

En la sección de operaciones alternativas se definen las alternativas aparecen durante la realización del caso de uso.

3.5. Diagrama de secuencia

Un diagrama de secuencia muestra la interacción entre objetos organizados en una secuencia de tiempo. Puede dibujarse con distintos niveles de detalle y para satisfacer distintos objetivos en las diversas etapas del ciclo de vida del desarrollo del sistema. Normalmente se utiliza para representar la interacción entre objetos que se produce en un caso de uso o para una operación. Cuando se utiliza para modelar el comportamiento dinámico de un caso de uso puede considerarse como una especificación detallada del caso de uso.

La dimensión vertical representa el tiempo. Los objetos involucrados en la interacción están distribuidos horizontalmente en el diagrama. Cada objeto está representado por una línea de vida, que se representa por una línea vertical discontinua y con un símbolo de objeto en su parte superior. Un mensaje se representa por una flecha horizontal que va desde una línea de vida a otra.

Cuando se envía un mensaje a un objeto se está llamando a una operación de ese objeto. El nombre del mensaje generalmente coincide con el de la operación que se está llamando. El periodo de tiempo durante el que se está ejecutando una operación se conoce como una ocurrencia de ejecución o de activación. Dicho periodo de tiempo se representa mediante un bloque rectangular situado a lo largo de la línea de vida.

Para este sistema, el diagrama de secuencia es el que se vislumbra en la [Figura](#)

[4.](#)

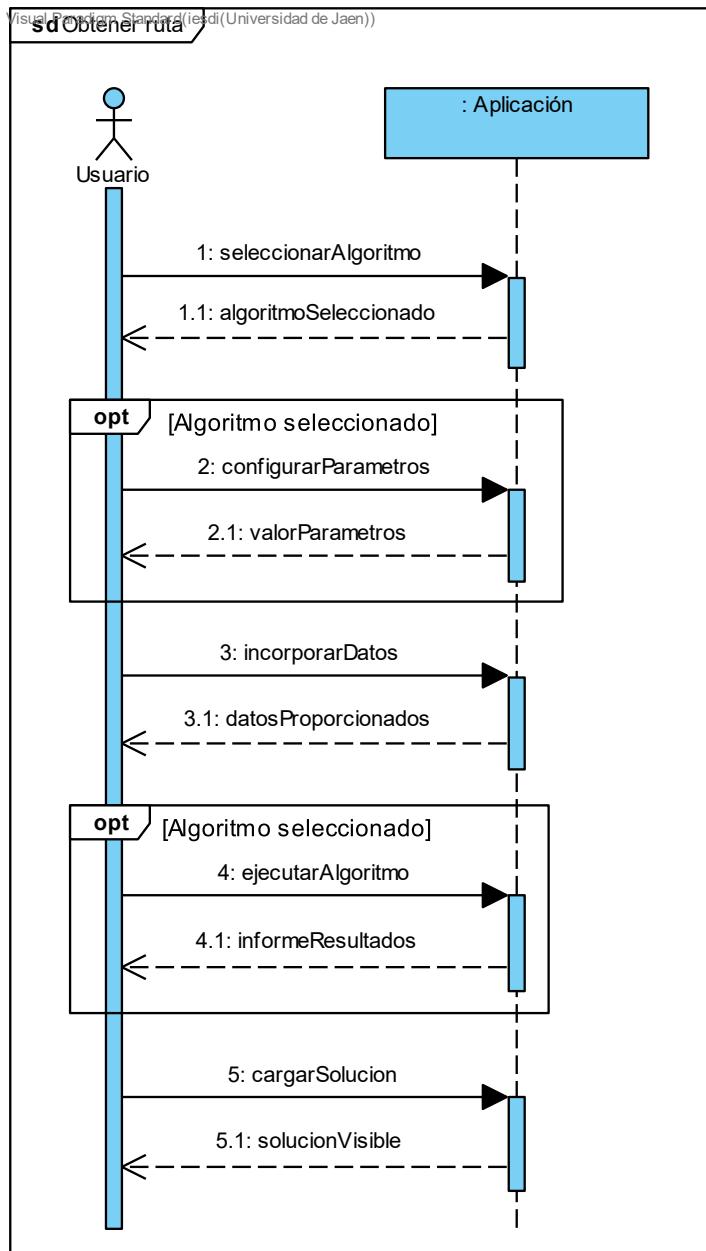


Figura 4. Diagrama de secuencia. Análisis. Elaboración propia.

Como se puede constatar, este diagrama ilustra cómo el usuario utiliza las distintas funcionalidades del sistema de forma ordenada en el tiempo. Cada flecha que sale del usuario al sistema representa el uso de una de sus funciones, y el sistema responde proporcionando el resultado de dicha llamada. Es notable mencionar que las partes del diagrama incluidas dentro de un marco “opt” indican que solo se llevarán a cabo si la condición especificada entre corchetes se evalúa como verdadera. A estos marcos comúnmente se les denomina fragmentos combinados.

CAPÍTULO 4

DISEÑO

En esta sección, se elaborará un esquema de la arquitectura del sistema para comprender su funcionamiento desde una perspectiva global. A continuación, se presentarán una serie de mockups que ilustrarán las diferentes vistas de la aplicación. Finalmente, se exhibirá un diagrama de clases que proporcionará una visión de la estructura genérica del código diseñado para la aplicación.

4.1. Arquitectura del sistema

En este apartado, se presenta la [Figura 5](#), que contiene un esquema que describe la arquitectura del sistema, detallando las entradas, salidas y operaciones realizadas internamente.

Para utilizar el sistema, en primer lugar, el usuario deberá elegir entre ejecutar un algoritmo o visualizar una solución obtenida previamente.

En el primer caso, el usuario deberá cargar un conjunto de datos que contenga las coordenadas de las ciudades. A partir de ahí, el sistema se encargará de leerlo y extraer los datos necesarios para ejecutar los algoritmos. Posteriormente, el usuario seleccionará un algoritmo de los disponibles, configurará sus parámetros y pulsará el botón correspondiente para iniciar la ejecución. En ese momento, el sistema tomará internamente los parámetros de configuración establecidos y el algoritmo seleccionado, pasándole los parámetros para que comience la ejecución. Se visualizarán las soluciones que se van encontrando durante este proceso, permitiendo al usuario detener la ejecución en cualquier momento. Una vez concluida la ejecución, se brinda al usuario la posibilidad de que el sistema genere un informe con los resultados obtenidos.

En el caso de que el usuario desee visualizar una solución obtenida previamente, solo tendrá que seleccionar dicha opción y elegir el archivo correspondiente. En ese momento, el sistema leerá todos los datos necesarios y mostrará en pantalla el mejor recorrido encontrado en esa ejecución, junto con el coste de la solución.

En resumen, el sistema se encarga internamente de todo el procesamiento de datos, los cálculos y su visualización, permitiendo al usuario una interacción simple y sin necesidad de tener demasiados conocimientos en la materia.

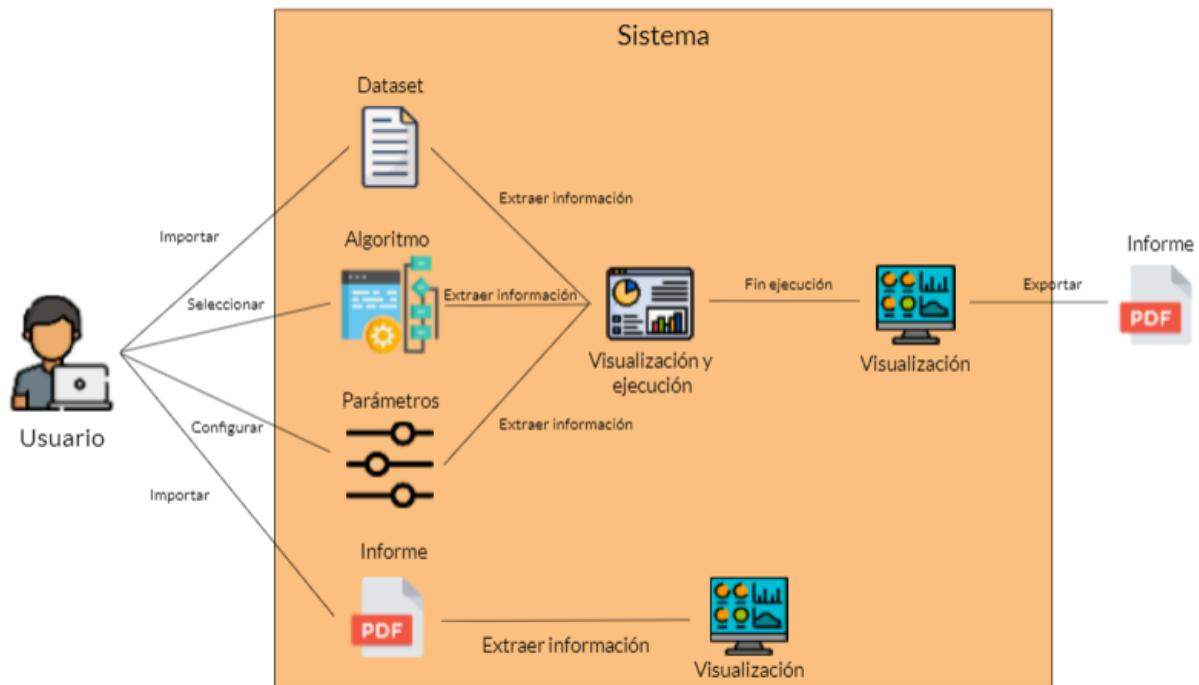


Figura 5. Arquitectura del sistema. Diseño. Elaboración propia.

4.2. Vistas necesarias

La aplicación de escritorio deberá contar con las vistas que se indican en la [Tabla 4](#) para cumplir con todos los requisitos y funciones especificados en el análisis.

Tabla 4. Vistas necesarias. Diseño. Elaboración propia.

| Número de vista | Vista | Finalidad |
|-----------------|---------------------------|--|
| 1 | Inicio | Presentar la aplicación, permitir seleccionar el conjunto de datos y el algoritmo a utilizar o cargar un informe de resultados |
| 2 | Configuración | Configurar el valor de los parámetros para el algoritmo seleccionado |
| 3 | Ejecución y visualización | Visualizar las soluciones que se van obteniendo durante la ejecución del algoritmo y permitir detener la ejecución |
| 4 | Ejecución completada | Visualizar la mejor solución encontrada y permitir la generación del informe de resultados |
| 5 | Visualización | Visualizar un informe de resultados |

4.3. Mockups

Un mockup es una representación muy avanzada del diseño gráfico y comunicativo de un proyecto. Se trata de una composición gráfica completa que introduce todos los elementos gráficos y visuales, sirviendo de un modelo a escala del producto para demostrar y probar un diseño.

Esta sección presenta los mockups elaborados para este sistema.

4.3.1. Pantalla de inicio

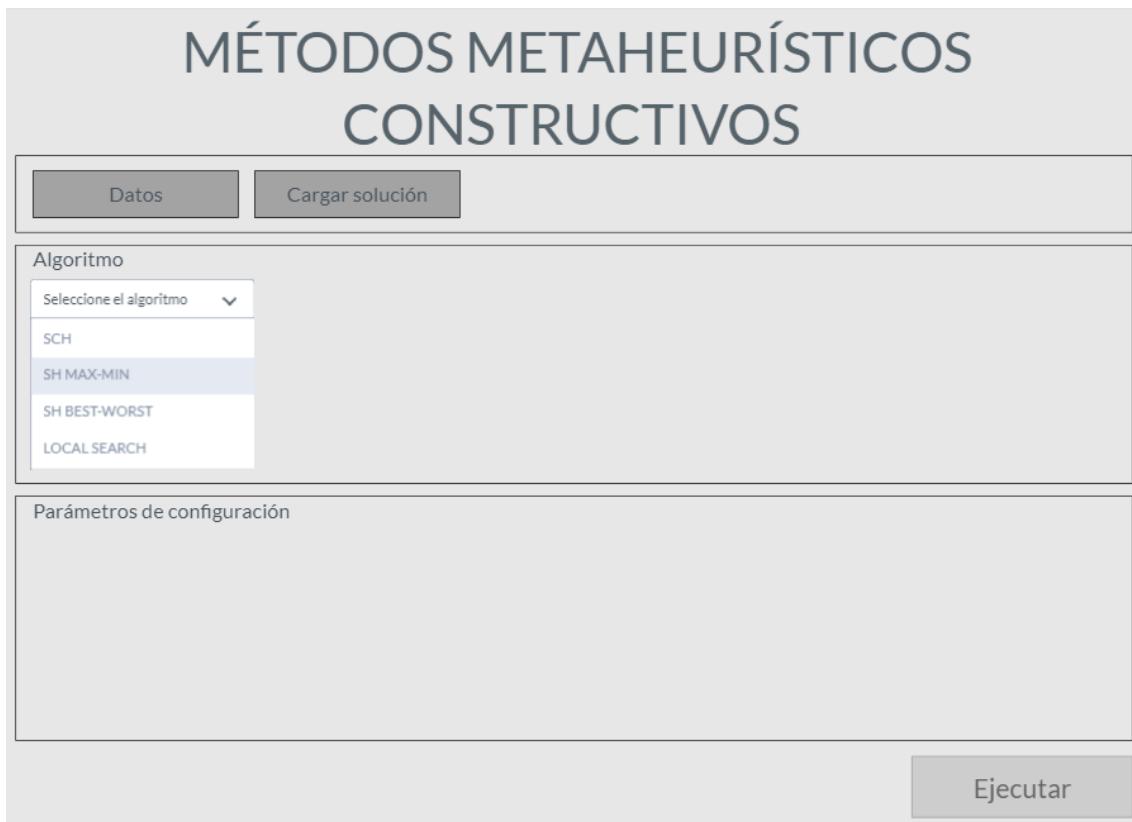


Figura 6. Pantalla de inicio. Diseño. Elaboración propia.

La [Figura 6](#) corresponde con la pantalla principal, donde podemos seleccionar el conjunto de datos de prueba y elegir un algoritmo para su ejecución. Asimismo, tenemos la opción de cargar una solución obtenida en una ejecución previa.

En este momento, el botón de “Ejecutar” no está activado, ya que el algoritmo y el conjunto de datos aún no han sido seleccionados.

4.3.2. Pantalla de configuración

The screenshot shows a user interface for configuring a constructive metaheuristic algorithm. At the top, there is a title section with the text "MÉTODOS METAHEURÍSTICOS CONSTRUCTIVOS". Below this, there are two buttons: "Datos" and "Cargar solución". A dropdown menu labeled "Algoritmo" is set to "SCH". The main area is titled "Parámetros de configuración" and contains several parameter settings:

| Tamaño de la población | 10 | Número de iteraciones | 80 | Actualización local | 0.1 |
|------------------------|------|-------------------------|-----|---------------------|-----|
| α | 1 | Tiempo de ejecución (s) | 90 | | |
| β | 2 | Feromonas inicial | 30 | | |
| q_0 | 0.95 | Actualización global | 0.1 | | |

At the bottom right of the configuration area is a "Ejecutar" button.

Figura 7. Pantalla de configuración. Diseño. Elaboración propia.

Después de seleccionar el algoritmo, accederíamos a la pantalla de la [Figura 7](#), muy similar a la anterior, pero con los parámetros que deberíamos configurar para el algoritmo elegido. Si también hemos seleccionado el conjunto de datos, el botón utilizado para ejecutar el algoritmo aparecería activo.

4.3.3. Pantalla de ejecución y visualización

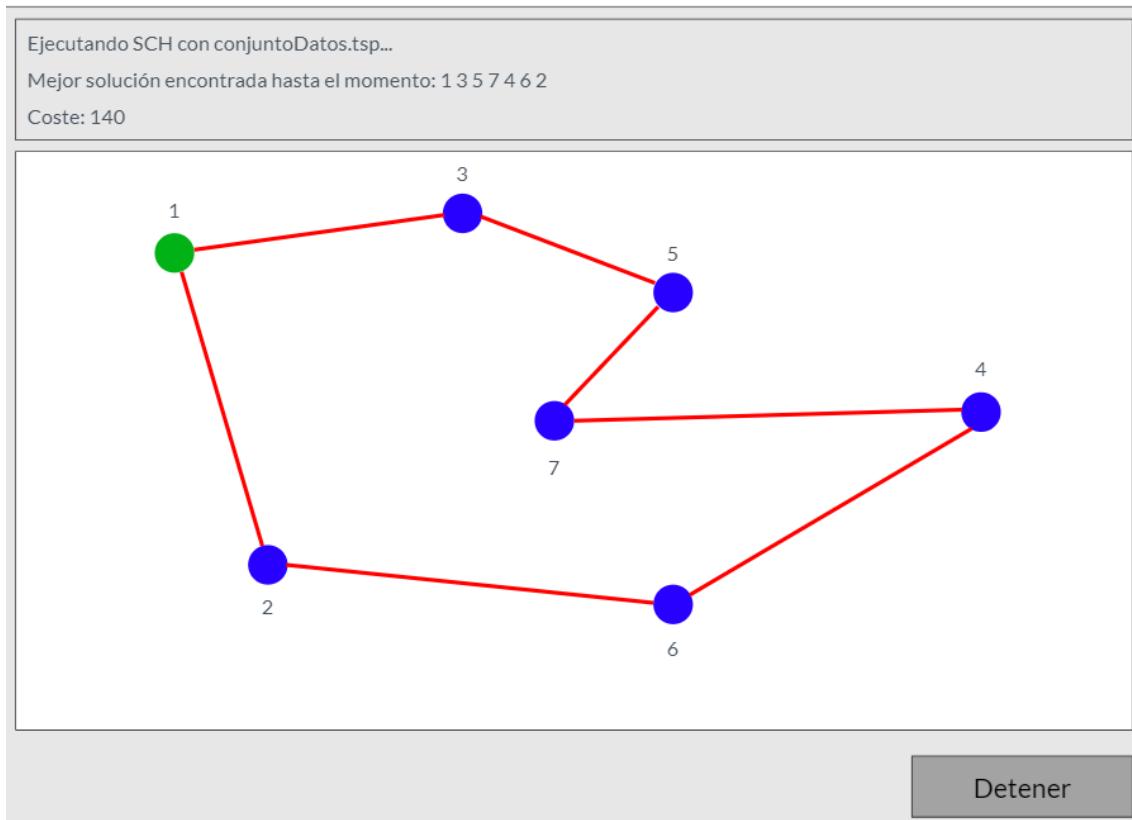


Figura 8. Pantalla de ejecución y visualización. Diseño. Elaboración propia.

Al iniciar la ejecución del algoritmo, se muestra la pantalla de la [Figura 8](#), donde podemos observar la mejor solución encontrada por el algoritmo seleccionado para el conjunto de datos elegido. Además, se ha añadido un botón para detener la ejecución si el usuario lo estima oportuno.

4.3.4. Pantalla de ejecución completada

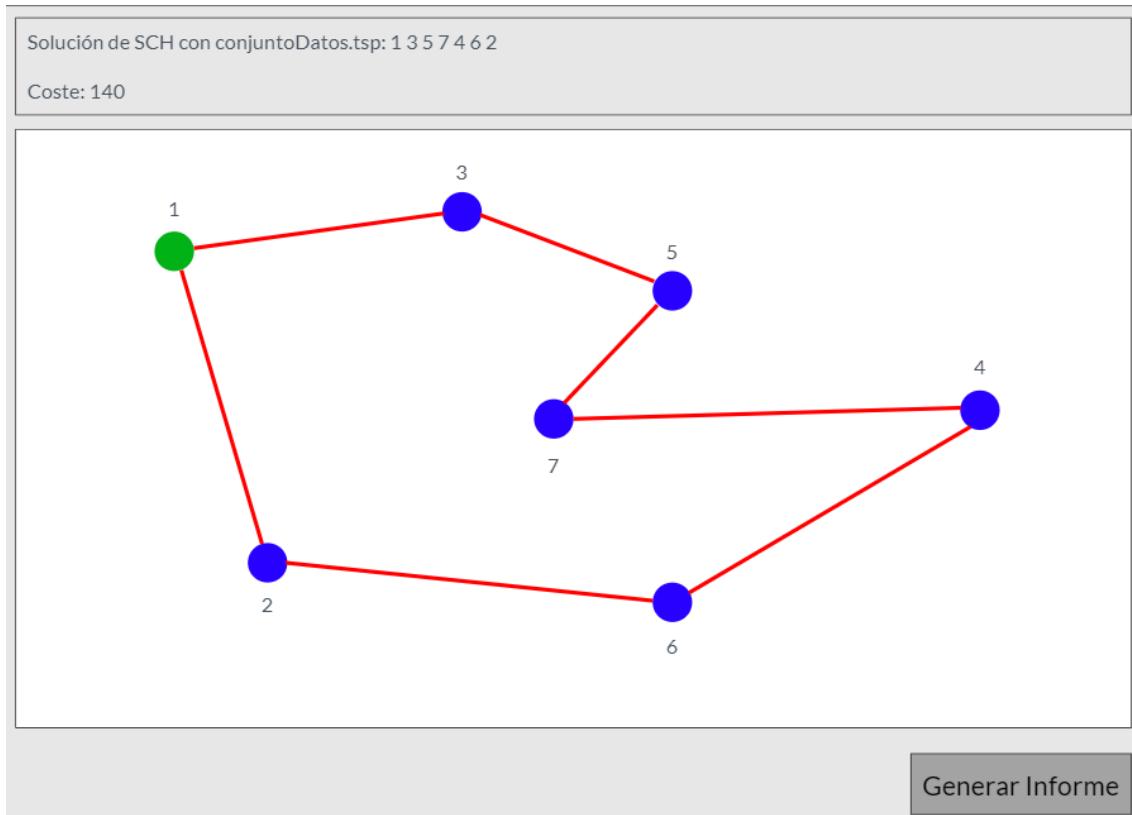


Figura 9. Pantalla de ejecución completada. Diseño. Elaboración propia.

La pantalla de la [Figura 9](#) se visualiza cuando el usuario ha detenido la ejecución o esta ha terminado por sí sola. En ella se presenta la mejor solución encontrada y su coste. Además, se ha incorporado un botón que genera un informe con esta, permitiendo al usuario visualizar la solución nuevamente en el futuro.

4.3.5. Pantalla de visualización

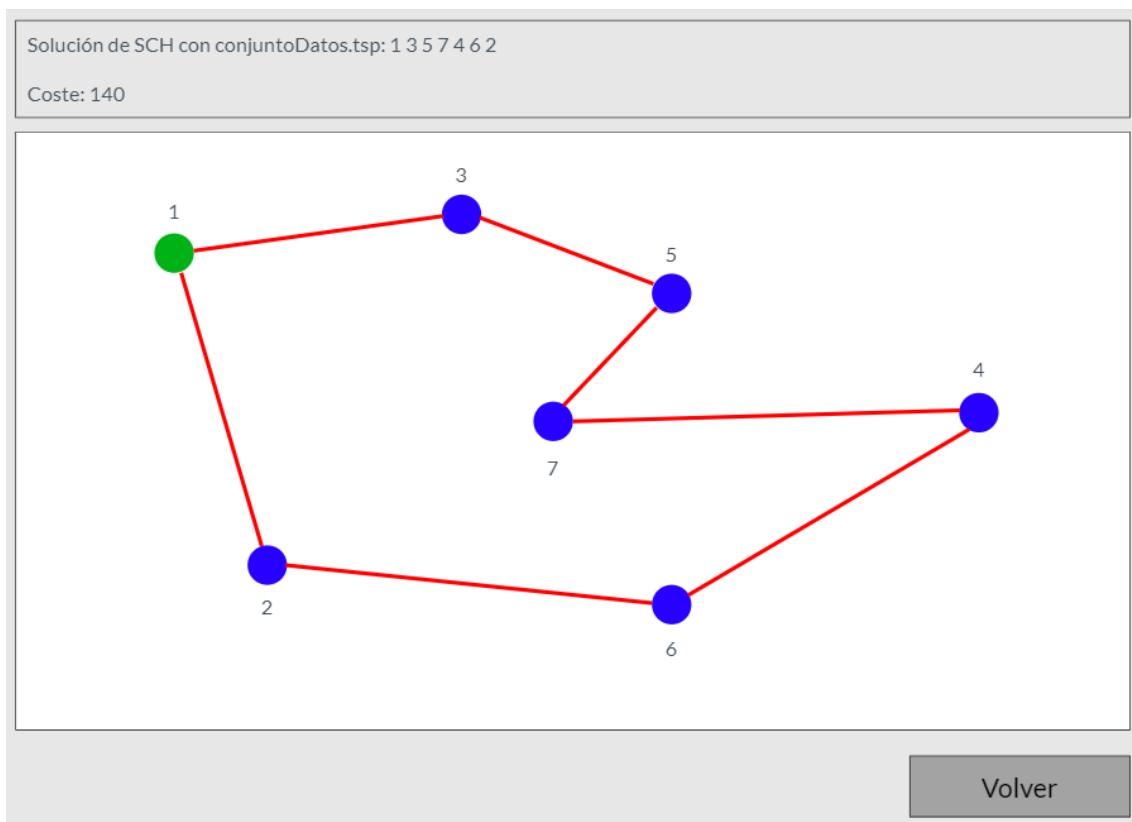


Figura 10. Pantalla de visualización. Diseño. Elaboración propia.

La pantalla de la [Figura 10](#) es la que se visualiza al seleccionar la opción de “Cargar solución” en la pantalla principal. Si hacemos clic en el botón “Volver”, regresaremos a la pantalla inicial del sistema.

4.4. Diagrama de clases

En esta sección se incluye el diseño del diagrama de clases para el sistema a desarrollar, el cual se refleja en la [Figura 11](#).

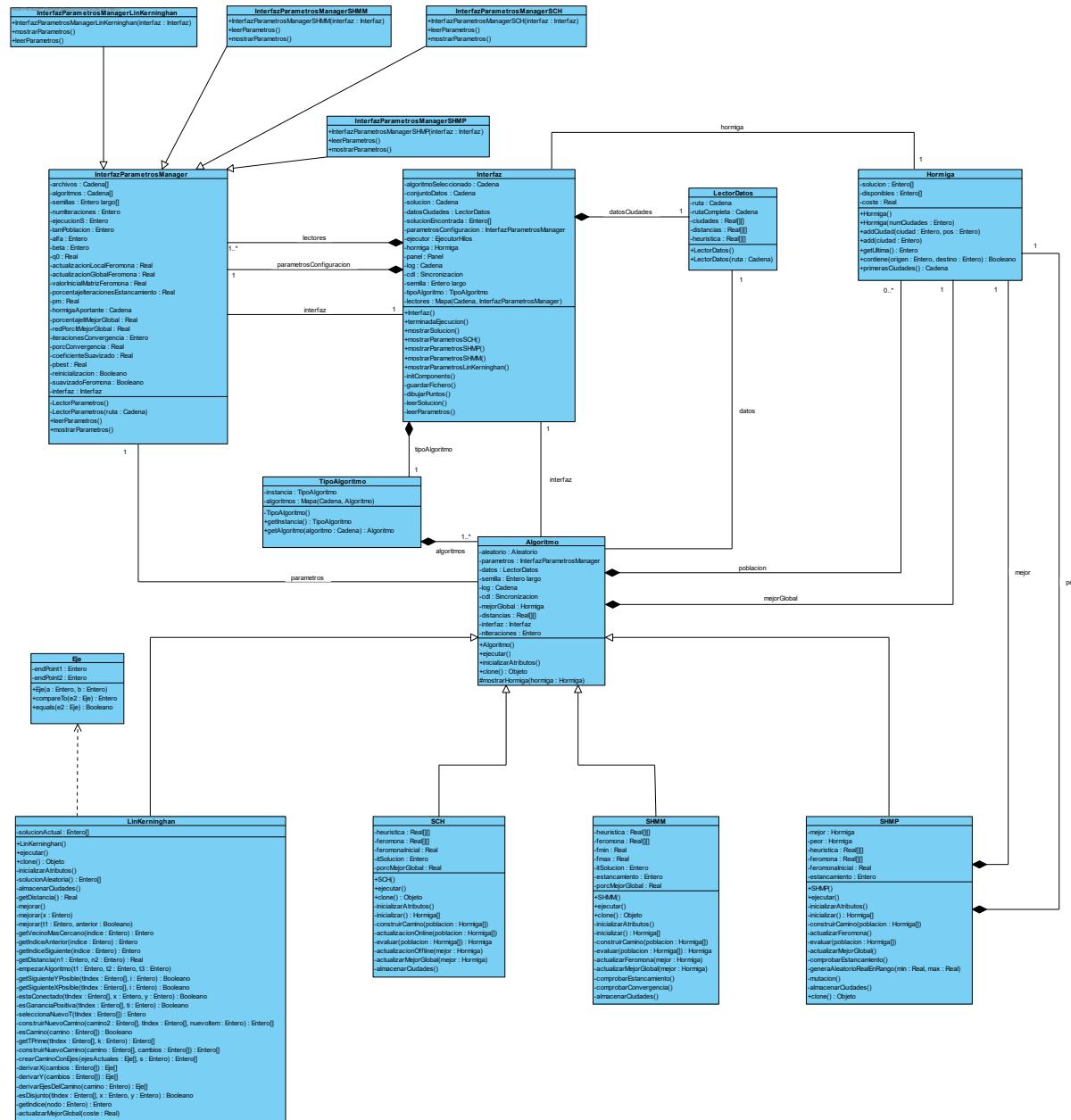


Figura 11. Diagrama UML. Diseño. Elaboración propia.

Como podemos evidenciar, el sistema estará compuesto por las siguientes clases:

- **Eje**: representa un eje o arista del grafo y es utilizada por la clase **LinKerninghan** para la resolución del problema.

- **Hormiga**: representará a una solución y su coste asociado.
- **InterfazParametrosManager**: superclase de:
 - InterfazParametrosManagerSCH.
 - InterfazParametrosManagerSHMM.
 - InterfazParametrosManagerSHMP.
 - InterfazParametrosManagerLinKerninghan.

utilizada para implementar el patrón Strategy, así como atributos y métodos comunes entre las clases. Se encargará de almacenar los parámetros de configuración del algoritmo que se va a ejecutar. Establecerá una relación de asociación con la clase `Interfaz` para actualizar la apariencia de esta con los parámetros correspondientes al algoritmo seleccionado.

- **InterfazParametrosManagerSCH**: se encarga de leer los parámetros de configuración del algoritmo de Sistema de Colonias de Hormigas y hacer uso de la interfaz para visualizarlos en pantalla. Es subclase de `InterfazParametrosManager` para implementar el patrón Strategy.
- **InterfazParametrosManagerSHMM**: asume la función de leer los parámetros de configuración del algoritmo de Sistema de Hormigas Max-Min y hace uso de la interfaz para visualizarlos en pantalla. Es subclase de `InterfazParametrosManager` para implementar el patrón Strategy.
- **InterfazParametrosManagerSHMP**: se dedica a leer los parámetros de configuración del algoritmo de Sistema de Hormigas del Mejor-Peor y hace uso de la interfaz para visualizarlos en pantalla. Es subclase de `InterfazParametrosManager` para implementar el patrón Strategy.
- **InterfazParametrosManagerLinKerninghan**: asume el control de leer los parámetros de configuración del algoritmo Lin-Kerninghan y hace uso de la interfaz para visualizarlos en pantalla. Es subclase de `InterfazParametrosManager` para implementar el patrón Strategy.

- **LectorDatos**: su función será la de cargar el conjunto de datos y construir la matriz de distancias y heurísticas.
- **LinKerninghan**: se ocupa de ejecutar el algoritmo de Lin-Kerninghan. Establece una relación de dependencia con la clase `Eje` ya que la utiliza de forma eventual para resolver el problema. Es subclase de `Algoritmo` para implementar el patrón Strategy.
- **SCH**: se responsabiliza de ejecutar el algoritmo de Sistema de Colonias de Hormigas. Es subclase de `Algoritmo` para implementar el patrón Strategy.
- **SHMM**: lleva a cabo la ejecución del algoritmo de Sistema de Hormigas Max-Min. Es subclase de `Algoritmo` para implementar el patrón Strategy.
- **SHMP**: se dedica a ejecutar el algoritmo de Sistema de Hormigas del Mejor-Peor. Es subclase de `Algoritmo` para implementar el patrón Strategy. Posee dos relaciones de composición con la clase `Hormiga` para almacenar la mejor y peor solución encontradas en cada iteración.
- **Algoritmo**: superclase de SCH, SHMM, SHMP y LinKerninghan utilizada para implementar el patrón Strategy y abstraer atributos y métodos comunes entre las clases. Establece una relación de asociación con la clase `Interfaz` ya que la utiliza para avisar de que ha terminado su ejecución o ir mandando las soluciones que va obteniendo para visualizarlas. También establece dos relaciones de composición con la clase `Hormiga`, una para almacenar y gestionar las distintas soluciones y otra para almacenar la mejor solución encontrada hasta el momento. Por otro lado, establece relaciones de asociación con las clases `InterfazParametrosManager` y `LectorDatos`, las cuales utiliza para obtener los parámetros y datos necesarios para la ejecución del algoritmo.
- **TipoAlgoritmo**: implementa el patrón Singleton. Su atributo más importante es de tipo mapa y se utiliza para devolver una instancia de los algoritmos a ejecutar en función del nombre del algoritmo que se indique. De esta manera, se define una relación de composición con la clase `Algoritmo`.

- **Interfaz**: se encarga de generar la interfaz de usuario y de la extracción de los datos necesarios para ejecutar los algoritmos. Dado que su función es iniciar la ejecución de cada uno de los algoritmos y, si esta desaparece, los algoritmos no se podrían ejecutar, se ha establecido una relación de composición entre esta clase y la clase `TipoAlgoritmo` para obtener una instancia del algoritmo seleccionado en la interfaz. También, se han establecido dos relaciones de composición con la clase `InterfazParametrosManager`, una para almacenar los distintos lectores de parámetros y otra para hacer referencia al que se está utilizando. Además, se ha establecido una relación de composición con la clase `LectorDatos` ya que, si desaparece la interfaz, desaparece el lector de datos que ha creado. Por otro lado, se ha añadido una relación de asociación entre esta clase y la clase `Hormiga` para tener una referencia a la solución que se está visualizando en pantalla.

Cabe destacar que, por facilitar la lectura y especificar únicamente los elementos importantes, en la clase `Interfaz` faltan por añadir los métodos correspondientes a la gestión de los eventos relativos a los distintos componentes con los que el usuario interactúa en la interfaz. En todas las clases faltan por añadir métodos getters y setters para obtener y establecer el valor de sus atributos, respectivamente.

CAPÍTULO 5

IMPLEMENTACIÓN

En este capítulo, se abordarán aspectos específicos relacionados con la implementación de la aplicación. En primer lugar, se indicará el lenguaje de programación utilizado y se mencionarán algunas de sus principales características. En segundo lugar, se indicará cómo se ha implementado el back-end de la aplicación, concretamente cómo se ha conseguido la modularidad del software y cómo se importan y exportan los distintos ficheros con los que trabaja el sistema, así como dar una descripción de cada uno de los algoritmos implementados. Por último, se explicará cómo se ha implementado el front-end o interfaz de usuario y se destacarán los puntos más importantes de la misma.

5.1. Lenguaje de programación utilizado

Para la implementación del proyecto se utilizará Java como lenguaje de programación. Java es un lenguaje de programación de propósito general que surgió en la década de 1990 gracias a Sun Microsystems, actualmente propiedad de Oracle Corporation. Destaca por su enfoque en la portabilidad, lo que significa que los programas escritos en Java pueden ejecutarse en diversas plataformas sin necesidad de modificaciones. Esta característica se logra mediante la compilación de código en un formato intermedio llamado bytecode, que puede ejecutarse en cualquier Máquina Virtual de Java (JVM).

El lenguaje está fundamentado en el paradigma de programación orientada a objetos, organizando el código en clases y objetos. Además, Java es reconocido por su soporte nativo para programación multihilo, permitiendo la ejecución concurrente de tareas, lo que resulta fundamental para aplicaciones que requieren procesamiento simultáneo.

La seguridad también es un aspecto esencial de Java. Se diseñó con características como la gestión automática de memoria y la capacidad de ejecutar código en un entorno controlado llamado "sandbox", proporcionando una capa adicional de seguridad.

Java es utilizado en una amplia gama de aplicaciones, desde el desarrollo web con Java EE hasta aplicaciones de escritorio con Java SE. Además, es el lenguaje principal para el desarrollo de aplicaciones Android. Su sintaxis, similar a la de otros lenguajes como C++ y C#, facilita a los desarrolladores aprender y trabajar en varios entornos.

En resumen, Java es un lenguaje versátil, seguro y portátil, ampliamente adoptado en la industria para una variedad de aplicaciones, desde sistemas empresariales hasta dispositivos móviles. Su lema "Write Once, Run Anywhere" refleja su capacidad para funcionar en distintas plataformas sin complicaciones.

5.2. Back-end

En esta sección se detallará la implementación del back-end de la aplicación, y se abordarán aspectos importantes que permitirán su correcto funcionamiento. Se

explicarán detalles tales como de qué manera la aplicación permite la modularidad en cuanto al número de algoritmos. Además, se describirá el proceso realizado para la importación de los conjuntos de datos e informes de resultados, así como la generación de estos últimos. También se explicará cómo se ha implementado la visualización de las soluciones encontradas. Finalmente, se hará una descripción del funcionamiento y estructura de los algoritmos implementados.

5.2.1. Modularidad

Para lograr que el software sea modular con respecto al número de algoritmos, esto es, que podamos añadir nuevos algoritmos sin necesidad de modificar demasiado el código original, se ha implementado el patrón Strategy en la aplicación.

Este patrón es la solución al siguiente problema:

Definir una **familia de algoritmos, encapsulados e intercambiables**. De esta forma cada algoritmo puede variar independientemente de los clientes que lo usan.

En el patrón Strategy, hay tres actores principales: una estrategia, varias estrategias concretas y un contexto.

- **Estrategia:** la estrategia es una interface que define la API común que van a tener las estrategias concretas. Define la familia de algoritmos que se quiere encapsular.
- **Estrategias concretas:** las estrategias concretas son clases que implementan la interface estrategia. Definen los distintos algoritmos encapsulados en la estrategia.
- **Contexto:** es la clase que usa la familia de algoritmos definida por la estrategia.

Este patrón se ha aplicado en la aplicación de manera que:

- **Estrategia:** la estrategia se corresponde con la clase `Algoritmo`. En ella se definen los atributos y métodos comunes a todos los algoritmos. El método más importante es el que se encarga de ejecutar el algoritmo, el cual implementa cada estrategia concreta.

- **Estrategias concretas:** las estrategias concretas corresponden a las clases que se encargan de ejecutar los distintos algoritmos. Estas son:
 - SCH.
 - SHMM.
 - SHMP.
 - LinKerninghan.
- **Contexto:** la clase que utiliza la familia de algoritmos definida por la estrategia es la clase `Interfaz` ya que se encargará de crear una instancia del algoritmo seleccionado en la interfaz de usuario y crear un hilo para que se ejecute.

Seguidamente, en la [Figura 12](#), se muestra el diagrama UML del patrón implementado.

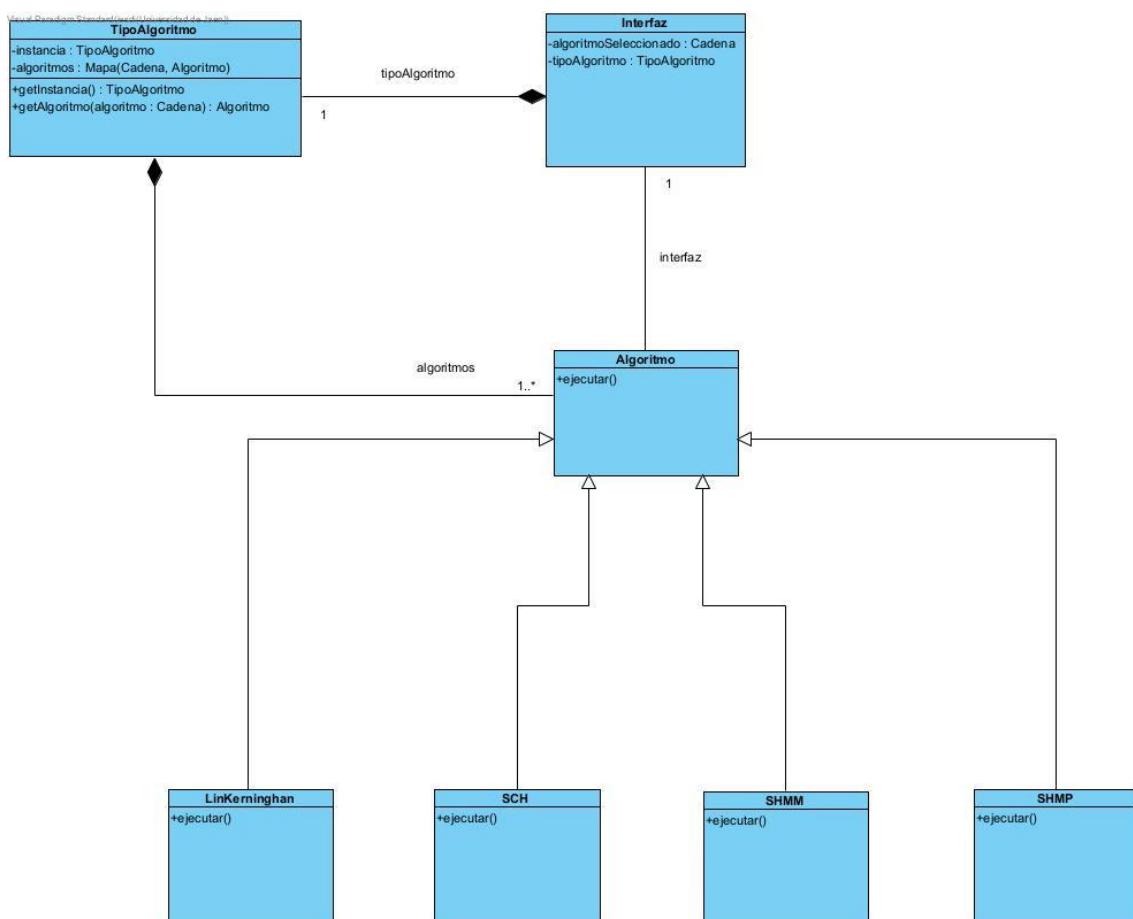


Figura 12. Patrón Strategy. Implementación. Elaboración propia.

Ahora, en el [Código 1](#), se muestra cómo se aplica dicho patrón en código Java en la clase Interfaz.

```
ExecutorService ejecutor =
Executors.newSingleThreadExecutor();

CountDownLatch cdl = new CountDownLatch(1);

Algoritmo algoritmo =
tipoAlgoritmo.getAlgoritmo(algoritmoSeleccionado);

algoritmo.setParametros(parametrosConfiguracion);

algoritmo.setDatos(datosCiudades);

algoritmo.setSemilla(semilla);

algoritmo.setInterfaz(this);

algoritmo.setCdl(cdl);

algoritmo.inicializarAtributos();

ejecutor.execute(algoritmo);
```

Código 1. Aplicación del patrón Strategy.

Como podemos comprobar en el [Código 1](#), se utiliza la clase que implementa el patrón Singleton `TipoAlgoritmo` para obtener una instancia del algoritmo a ejecutar a partir de la cadena de texto que lo identifica en la interfaz. Mediante polimorfismo, indicamos que es un objeto de la clase `Algoritmo`. A continuación, inicializamos sus atributos y lanzamos un hilo para ejecutar el algoritmo. A la hora de ejecutarse, se ejecutarán los métodos de la subclase ya que realmente es una instancia de esta.

De igual modo, para la lectura de los parámetros correspondientes al algoritmo seleccionado y actualizar la apariencia de la interfaz, se ha implementado de nuevo el patrón Strategy.

Este patrón se ha aplicado en la aplicación de manera que:

- **Estrategia:** la estrategia se corresponde con la clase `InterfazParametrosManager`. En ella se definen los atributos y métodos comunes a todos los lectores de parámetros. Los principales

métodos son los que se encargan de leer los parámetros y actualizar la apariencia de la interfaz, los cuales son implementados en cada estrategia concreta.

- **Estrategias concretas:** las estrategias concretas corresponden a las clases que se encargan de leer los parámetros de los distintos algoritmos. Estas son:
 - InterfazParametrosManagerSCH.
 - InterfazParametrosManagerSHMM.
 - InterfazParametrosManagerSHMP.
 - InterfazParametrosManagerLinKerninghan.
- **Contexto:** la clase que utiliza la familia de algoritmos definida por la estrategia es la clase `Interfaz` ya que se encargará de obtener una instancia del lector de parámetros correspondiente al algoritmo seleccionado en la interfaz de usuario para leer sus parámetros y actualizar su apariencia en consecuencia.

Acto seguido, se muestra en la [Figura 13](#) el diagrama UML del patrón implementado.

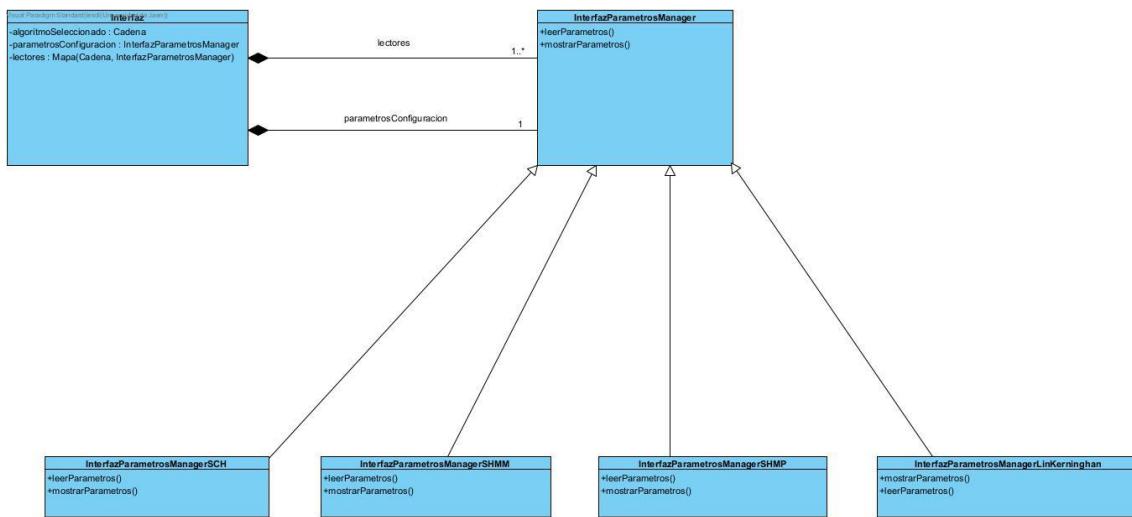


Figura 13. Patrón Strategy. Implementación. Elaboración propia.

Consecuentemente, se muestra en el [Código 2](#) cómo se aplica dicho patrón en Java en la clase `Interfaz`.

```
InterfazParametrosManager parametrosConfiguracion =  
lectores.get(algoritmoSeleccionado);  
  
parametrosConfiguracion.leerParametros();  
  
parametrosConfiguracion.mostrarParametros();
```

Código 2. Aplicación del patrón Strategy.

Como se puede evidenciar en el [Código 2](#), una vez ya se sabe qué algoritmo se ha seleccionado en la interfaz, obtenemos el lector de parámetros correspondiente a dicho algoritmo, leemos sus parámetros y actualizamos la apariencia de la interfaz.

La implementación de este patrón de diseño software nos permite la inclusión de nuevos algoritmos sin necesidad de modificar el código existente para ejecutarlos. Solo necesitamos agregar código en la clase `Interfaz` para poder seleccionar el nuevo algoritmo y configurar sus parámetros. Además, debemos añadir una nueva entrada al mapa que utiliza la clase `TipoAlgoritmo`, donde la clave sea la cadena con el nombre del nuevo algoritmo y el valor sea una instancia de este nuevo algoritmo. De igual manera, debemos agregar al mapa que utiliza la clase `Interfaz` una nueva entrada, donde la clave sea la cadena con el nombre del nuevo algoritmo y el valor sea una instancia del nuevo lector de parámetros.

Cabe mencionar que, para la ejecución de la aplicación por línea de comandos, el código desarrollado para la creación y ejecución de algoritmos se basa en la misma idea expuesta. Sin embargo, es importante tener en cuenta que un mismo algoritmo se puede ejecutar simultáneamente con distintos conjuntos de datos, por lo que necesitaríamos copias independientes de cada instancia y luego pasarles los parámetros correspondientes. Por lo tanto, se ha hecho que los algoritmos implementen la interfaz `Cloneable` para crear copias de instancias de algoritmos. En cada clase, se ha implementado el método `clone()` que se encarga de crear la copia independiente, es decir, si realizo modificaciones en el objeto original, no afectarán a la copia y viceversa.

Así pues, en el [Código 3](#) se muestra la implementación del método `clone()`.

```

@Override
public Object clone() {
    super.clone();
    return new Algoritmo();
}

```

Código 3. Clonación de instancias de algoritmos.

De igual manera, como estamos trabajando con polimorfismo, se ejecutará el método `clone()` de la clase a la que corresponda la instancia creada del algoritmo.

Tras esto, se muestra en el [Código 4](#) cómo se aplica esto cuando la ejecución de algoritmos se realiza desde la línea de comandos.

```

for (int i = 0; i < algoritmos.size(); i++) {
    for (int j = 0; j < semillas.size(); j++) {
        for (int k = 0; k < datos.size(); k++) {
            Algoritmo hilo =
                instancia.getAlgoritmo(algoritmos.get(i));
            hilo.setParametros(config);
            hilo.setDatos(datos.get(k));
            hilo.setSemilla(semillas.get(j));
            hilo.setCdl(cdl);
            hilo.inicializarAtributos();
            hilos.add(hilo);
            ejecutor.execute(hilo);
        }
    }
}

```

Código 4. Aplicación del patrón Strategy para ejecución en línea de comandos.

En el [Código 5](#) se presenta el método de la clase `TipoAlgoritmo` donde se utiliza la clonación de instancias.

```

public Algoritmo getNuevoAlgoritmo(String algoritmo) {
    Algoritmo alg = algoritmos.get(algoritmo);
    if (alg != null) {
        return (Algoritmo) alg.clone();
    }
    return alg;
}

```

Código 5. Obtención de una copia de un algoritmo.

5.2.2. Exportación de la información

La exportación de ficheros consistirá en almacenar la siguiente información en orden y forma:

- Algoritmo que ha ejecutado el usuario: se especificará el algoritmo ejecutado de la siguiente manera:

Algoritmo: SCH

- Conjunto de datos utilizado: se indicará el conjunto de datos utilizado para la ejecución del algoritmo con esta estructura:

Conjunto de datos: ch130.tsp

- Parámetros de configuración: se almacenarán los parámetros de configuración y los valores de los mismos que se han utilizado en la ejecución del algoritmo seleccionado:

Tamaño de la población: 10

Tiempo de ejecución: 600

Iteraciones: 1200

- Solución encontrada: se denotará en una línea “La mejor solución encontrada es:” y en la siguiente (o siguientes) se imprimirá la mejor solución encontrada durante la ejecución completa del algoritmo. Por ejemplo:

La mejor solución encontrada es:

[1, 2, 3, 4]

- Coste de la solución: se señalará, a continuación de la solución encontrada, el coste de la misma como se indica a continuación:

Coste: 1253

- Iteraciones realizadas: se designará el número de iteraciones realizadas por el algoritmo de este modo:

Número de iteraciones realizadas: 20

- Tiempo de ejecución: se expresará el tiempo en segundos de ejecución del algoritmo como sigue:

Tiempo de ejecución: 60 s

- Coordenadas de las ciudades: se guardará el número de la ciudad y las coordenadas de cada una para poder situarlas en la interfaz. El formato es el siguiente:

1 192.32 432.3

2 196.23 223.1

- Distancias entre ciudades: se depositará la distancia entre cada par de ciudades tal y como se indica a continuación:

Distancia de la ciudad 1 a la ciudad 3: 59

- Matriz de heurísticas: se almacenará el valor heurístico entre cada par de ciudades en los algoritmos SCH, SHMM y SHMP siguiendo esta pauta:

Heurística de la ciudad 1 a la ciudad 3: 20

Toda esta información la irá guardando el algoritmo correspondiente haciendo uso de la clase `StringBuilder` de Java. Una vez termine su ejecución, le pasará esta información a la interfaz y esta utilizará las clases `File`, `PdfWriter`, `PdfDocument` y `Document` de Java para generar el informe con el nombre y ubicación que desee el usuario en formato PDF como se especifica en el [Código 6](#).

```
File file = fileChooser.getSelectedFile();

PdfWriter writer = new PdfWriter(file.getAbsolutePath()
+ ".pdf");

PdfDocument pdfDocument = new PdfDocument(writer);

Document document = new Document(pdfDocument);

document.add(new Paragraph(log.toString()));

document.close();
```

Código 6. Exportación de información.

5.2.3. Importación de ficheros

En este apartado, se indicará el procedimiento realizado para la lectura de los conjuntos de datos, informes generados y fichero de parámetros, así como la estructura que tienen que tener para que su lectura se pueda realizar correctamente.

5.2.3.1. Conjuntos de datos

La estructura de los conjuntos de datos tendrá cuatro partes:

- Número de ciudades: se indicará el número de ciudades en una línea que ponga “DIMENSION: tamaño”, donde tamaño será un número entero. Por ejemplo:

DIMENSIÓN: 120

- Inicio de las coordenadas de las ciudades: se especificará en una línea independiente con la cadena “NODE_COORD_SECTION” que comienza la lista de ciudades y sus coordenadas.
- Coordenadas de las ciudades: se denotará en cada línea el número de la ciudad (siendo la primera el número 1), seguido de un espacio en blanco y las coordenadas en dos dimensiones de las ciudades separadas por un espacio en blanco de la siguiente manera:

1 1923.23 112.34

- Finalización del fichero: se designará la finalización del fichero después de las coordenadas de las ciudades en una línea independiente mediante la cadena “EOF”.

Por tanto, una vez conocemos la estructura que debe tener el fichero, pasaremos a realizar su lectura. Esta lectura se realizará por medio de las clases `FileReader` y `BufferedReader` de Java. Para ello, en primer lugar, comenzaremos leyendo líneas del fichero hasta que encontremos aquella que nos indica el tamaño del fichero. Una vez lleguemos a ella, la procesaremos y nos quedaremos solo con el valor numérico que es lo que realmente nos interesa para inicializar el tamaño de las matrices a utilizar.

A continuación, seguiremos leyendo líneas buscando la cadena “NODE_COORD_SECTION”, la cual indicará el inicio del listado de ciudades y sus coordenadas. A partir de este punto, continuaremos leyendo líneas, dividiendo cada una de ellas por el espacio en blanco (“ ”) y completando la matriz de coordenadas hasta que alcancemos a la cadena “EOF”, que señala el final del fichero.

Un ejemplo de este fichero puede ser:

```
NAME: ch130
TYPE: TSP
COMMENT: 130 city problem (Churritz)
DIMENSION: 130
EDGE_WEIGHT_TYPE: EUC_2D
NODE_COORD_SECTION
1 334.5909245845 161.7809319139
2 397.6446634067 262.8165330708
3 503.8741827107 172.8741151168
4 444.0479403502 384.6491809647
5 311.6137146746 2.0091699828
6 662.8551011379 549.2301263653
7 40.0979030612 187.2375430791
8 526.8941409181 215.7079092185
EOF
```

Según se ha explicado cómo se lee el fichero, las líneas que no sean relevantes para nosotros serán ignoradas.

En el caso de que el archivo no tenga el formato adecuado, se mostrará un mensaje de error en color rojo justo encima del botón que permite seleccionar el conjunto de datos. Si la ejecución se realiza por la línea de comandos, este error se indicará mediante un mensaje.

5.2.3.2. *Informes*

Ya estamos familiarizados con la estructura de los informes, la cual se ha explicado en el apartado [5.2.2](#). Por lo tanto, nos centraremos únicamente en el procedimiento de lectura de los mismos.

En primer lugar, emplearemos las clases `File`, `PDFDocument` y `PDFTextStripper` de Java para leer todo el contenido del archivo. Seguidamente, dividiremos el texto obtenido en líneas utilizando los saltos de línea presentes en el archivo, generando un array donde cada posición corresponde a una línea. En este

punto, procederemos a realizar un procesamiento línea por, extrayendo los datos necesarios del informe para mostrar el algoritmo utilizado, conjunto de datos, solución encontrada, coste y coordenadas de las ciudades.

Prosiguiendo, en el [Código 7](#) se muestra cómo se hace la lectura del fichero y su descomposición en líneas.

```
File pdfFile = new File(informe);

PDDocument document = PDDocument.load(pdfFile);

PDFTextStripper textStripper = new PDFTextStripper();

String contenidoPDF = textStripper.getText(document);

String[] lineas = contenidoPDF.split("\r?\n");
```

Código 7. Lectura de informe.

El procesamiento línea por línea es similar al explicado en apartado anterior, donde vamos buscando las cadenas de texto relevantes. En caso de que el archivo no tenga el formato adecuado, ya sea porque no pertenece a nuestra aplicación, se generará una excepción durante su procesamiento. En este escenario, se mostrará justo encima del botón de la interfaz para cargar el informe un mensaje de error en color rojo, informando de que el archivo no tiene un formato adecuado.

5.2.3.3. *Ficheros de parámetros*

Cada algoritmo implementado constará de un fichero de parámetros en formato JSON, donde se especificará, para cada parámetro, su valor por defecto. El formato de cada fichero será similar al siguiente:

```
{  
    "Parametro1": Valor1,  
    "Parametro2": Valor2,  
    .  
    .  
    .  
    "ParametroN": ValorN  
}
```

Para la lectura de estos ficheros, se utilizarán las clases `ObjectMapper`, `File` y `JsonNode` de Java tal, conforme se muestra en el [Código 8](#).

```
ObjectMapper objectMapper = new ObjectMapper();  
  
File archivoJSON = new File("parametros/Lin-  
Kernighan.json");  
  
JsonNode parametros = objectMapper.readTree(archivoJSON);  
  
numIteraciones = parametros.get("Iteraciones").asInt();
```

Código 8. Lectura de fichero de parámetros JSON.

La importancia del funcionamiento de este código reside en especificar correctamente la ruta del fichero de parámetros y el nombre del parámetro cuyo valor se quiere recuperar.

5.2.3.4. Fichero de configuración

El fichero de configuración contendrá la agrupación de todos los parámetros de los algoritmos implementados y solo se utilizará para la ejecución de la aplicación desde la línea de comandos. Cuando la ejecutamos desde la interfaz, podemos establecer el valor de los parámetros desde la misma.

Este fichero tendrá la misma estructura para todos sus parámetros. Cada parámetro se especificará mediante su nombre, seguido de un signo “=” y el valor o valores del parámetro separados por un espacio en blanco. A continuación, se detallan

cada uno de los parámetros posibles, cómo se establecen en el fichero de configuración (deben colocarse tal y como se indica a continuación) y los valores que pueden tomar:

- Archivos: aquí se indican los conjuntos de datos a utilizar. Se especifican de la siguiente manera:

Archivos=wi29.tsp dj38.tsp qa194.tsp

- Semillas: son números enteros utilizados para la generación de números aleatorios. Al utilizar la misma semilla siempre en un algoritmo y con el mismo conjunto de datos, siempre obtendremos el mismo resultado. Esto ayuda de cara a su reproducción futura. Se detallan de la siguiente forma:

Semillas=102934 2342 647 23

- Algoritmos: aquí se definen los algoritmos a ejecutar. Se presentan del siguiente modo:

Algoritmos=SCH SHMM SHMP Lin-Kerninghan

- Iteraciones: es un número entero positivo que indica el número máximo de iteraciones que realizará el algoritmo. Se indican con el siguiente formato:

Iteraciones=10000

- Segundos de ejecución: es un número entero positivo que indica el tiempo de ejecución total del/de los algoritmo/s en segundos. Se especifican de la siguiente manera:

Segundos de ejecución=600

- Tamaño de la población: es un número entero positivo que indica el número de hormigas que tendrá cada población para los algoritmos SCH, SHMM y SHMP. Se muestra de la siguiente manera:

Tamaño de la población=10

- Alfa: es un número entero positivo utilizado en la regla de transición en los algoritmos SCH, SHMM y SHMP. Se precisa de la siguiente manera:

Alfa=1

- Beta: es un número entero positivo utilizado en la regla de transición en los algoritmos SCH, SHMM y SHMP. Se define del siguiente modo:

Beta=2

- q_0 : es un número real comprendido entre 0 y 1 que se utiliza en la regla de transición del algoritmo SCH. Se expone de la siguiente manera:

$q_0=0.98$

- Actualización global de la feromona: es un número real comprendido entre 0 y 1 que se utiliza en la actualización de feromona en el algoritmo SCH. Se define como sigue:

Actualización global de la feromona=0.1

- Actualización local de la feromona: es un número real comprendido entre 0 y 1 que se utiliza en la actualización de feromona en los algoritmos SCH, SHMM y SHMP. Se especifica así:

Actualización local de la feromona=0.1

- Feromona inicial: es un valor real positivo que se utiliza para inicializar la matriz de feromona en los algoritmos SCH y SHMP. Se determina siguiendo este patrón:

Feromona inicial=100

- Reinicialización: es un valor booleano que se utiliza para indicar si el algoritmo se reinicia cuando se estanque. Se aplica a los algoritmos SHMM y SHMP. Se define de la forma que sigue:

Reinicialización=true

Reinicialización=false

- Porcentaje de iteraciones para considerar estancamiento: es un valor real entre 0 y 1 que se utiliza para indicar cuántas iteraciones sin mejora con respecto al total hacen que se considere un estancamiento del algoritmo. Se aplica en los algoritmos SHMM y SHMP. Se plasma según se establece a continuación:

Porcentaje de iteraciones para considerar estancamiento=0.05

- Probabilidad de mutación: es un valor real entre 0 y 1 que se emplea para modificar la feromona de un arco si se cumple la probabilidad de mutación en el algoritmo SHMP. Se presenta siguiendo este patrón:

Probabilidad de mutación=0.01

- Hormiga que aporta feromona: es una cadena de texto que se utiliza en los algoritmos SCH y SHMM para decidir qué hormiga es la que aporta feromona. Se expresa siguiendo este patrón:

Hormiga que aporta feromona=mejor global

Hormiga que aporta feromona=mejor iteración

Hormiga que aporta feromona=ambas

- Porcentaje de iteraciones inicial para usar mejor global: es un valor real entre 0 y 1 que indica cada cuántas iteraciones con respecto al total se utiliza la mejor hormiga encontrada hasta el momento para aportar feromona en el caso que aporten ambas hormigas. Se utiliza en los algoritmos SCH y SHMM. Se especifica de la siguiente manera:

Porcentaje de iteraciones inicial para usar mejor global=0.08

- Suavizado de feromona: es un valor booleano utilizado para indicar si se realiza suavizado de feromona en los arcos en el algoritmo SHMM. Se indica del siguiente modo:

Suavizado de feromona=true

Suavizado de feromona=false

- Porcentaje de reducción de iteraciones: es un valor real entre 0 y el porcentaje de iteraciones inicial para usar la mejor solución global, utilizado para que la mejor hormiga encontrada hasta el momento aporte feromona con más frecuencia en el caso de que aporten ambas hormigas. Se indica como sigue:

Porcentaje de reducción de iteraciones=0.005

- Número de iteraciones para comprobar la convergencia: es un valor entero positivo utilizado para comprobar periódicamente la convergencia del algoritmo utilizado en el algoritmo SHMM. Se precisa de este modo:

Número de iteraciones para comprobar la convergencia=100

- Factor de convergencia: es un valor real entre 0 y 1 que se utiliza para ver si se ha producido la convergencia del algoritmo SHMM. Se concreta de acuerdo con esto:

Factor de convergencia=0.05

- Coeficiente de suavizado de feromona: es un valor real entre 0 y 1 utilizado en el suavizado de feromona para el algoritmo SHMM. Se establece de la siguiente manera:

Coeficiente de suavizado de feromona=0.5

- Pbest: es un valor real entre 0 y 1 utilizado en el algoritmo SHMM para el cálculo de la feromona mínima. Se indica de la siguiente forma:

Pbest=0.05

Esta lectura será llevada a cabo por la clase `InterfazParametrosManager`, utilizando las clases `FileReader` y `BufferedReader` de Java. El proceso de lectura implica ir leyendo línea por línea del fichero y dividirla el espacio en blanco para obtener un array. Después de obtener el array, accederemos a la primera posición, que es donde se encuentra el parámetro en sí (Archivos, Semillas, Algoritmos...). En función de este parámetro, se realizará un procesamiento específico y se almacenarán los datos en las estructuras correspondientes. Este proceso se repetirá para cada línea del archivo.

Si se establece algún valor no permitido para algún parámetro, se generará una excepción indicando que el formato del fichero no es adecuado y se detendrá la ejecución.

5.2.4. Visualización de soluciones

Al iniciar la ejecución del algoritmo seleccionado, mostramos una ventana que presenta el algoritmo en ejecución, la mejor solución encontrada hasta el momento, su coste y su representación gráfica.

La representación se ha realizado mediante la adición de un `JPanel` de Java Swing a la ventana. Este `JPanel` se utiliza para crear componentes gráficos en interfaces de usuario. Para lograr esto, hemos definido una nueva clase que herede de `JPanel`, encargada de dibujar las soluciones.

En dicha clase, hemos implementado el método `paintComponent()`, el cual se invoca automáticamente cada vez que es necesario repaintar el panel. En este método se realiza la representación gráfica de las ciudades y la solución encontrada. Dentro de este método, se obtienen las dimensiones del panel (`panelWidth` y `panelHeight`) y los límites (`minX`, `maxX`, `minY`, `maxY`) de las coordenadas de las ciudades. Estos límites se utilizan para escalar las coordenadas de las ciudades al tamaño del panel.

Para dibujar las ciudades, se recorre el array bidimensional `ciudades` que contiene las coordenadas de las ciudades. Cada ciudad se representa como un círculo (óvalo) en el panel y se etiqueta con un número. Si se ha encontrado una solución (`solucionEncontrada`), la ciudad inicial de la solución se resalta en verde y el resto se dibujan en color azul. Además, se dibuja una línea roja para conectar las ciudades en el orden especificado por la solución.

Aún nos falta por hablar sobre cómo la interfaz sabe cuándo tiene que dibujar una nueva solución. Esta responsabilidad recae en el propio algoritmo. Cuando el algoritmo encuentra una solución mejor que la anterior, invoca a un método implementado en la interfaz para transmitirle esta nueva solución. Luego, la interfaz invoca al método `repaint()` del panel, lo que provoca la ejecución del método `paintComponent()`, logrando así la representación de la nueva solución.

En el [Código 9](#) se muestra cómo se ha implementado la representación gráfica.

```

private class Panel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        Graphics2D g2d = (Graphics2D) g;
        int panelWidth = getWidth();
        int panelHeight = getHeight();

        double[][] ciudades = datosCiudades.getCiudades();

        double maxX = Arrays.stream(ciudades).mapToDouble(coords ->
        coords[0]).max().orElse(0);
        double minX = Arrays.stream(ciudades).mapToDouble(coords ->
        coords[0]).min().orElse(Double.POSITIVE_INFINITY);
        double maxY = Arrays.stream(ciudades).mapToDouble(coords ->
        coords[1]).max().orElse(0);
        double minY = Arrays.stream(ciudades).mapToDouble(coords ->
        coords[1]).min().orElse(Double.POSITIVE_INFINITY);

        if (hormiga != null) {
            solucionEncontrada = hormiga.getSolucion();
        }

        g2d.setColor(Color.BLUE);

        for (int i = 0; i < ciudades.length; i++) {
            int x = (int) (((ciudades[i][0] - minX) / (maxX - minX)) *
            (panelWidth - 15 - 15) + 15);
            int y = (int) (((ciudades[i][1] - minY) / (maxY - minY)) *
            (panelHeight - 15 - 15) + 15);
            if (solucionEncontrada != null && i + 1 ==
            solucionEncontrada.get(0) + 1) {
                g2d.setColor(Color.GREEN);
            }
            g2d.fillOval(x, y, 10, 10);
            g2d.setColor(Color.BLACK);
            g2d.drawString(String.valueOf(i + 1), x, y - 5);
            g2d.setColor(Color.BLUE);
        }
        if (solucionEncontrada != null) {
            int offsetX = 4;
            int offsetY = 4;
            g2d.setColor(Color.RED);
            for (int i = 0; i < solucionEncontrada.size(); i++) {
                int x1 = (int)
                (((ciudades[solucionEncontrada.get(i)][0] + offsetX - minX) / (maxX - minX)) *
                (panelWidth - 15 - 15) + 15);
                int y1 = (int)
                (((ciudades[solucionEncontrada.get(i)][1] + offsetY - minY) / (maxY - minY)) *
                (panelHeight - 15 - 15) + 15);
                int x2 = (int) (((ciudades[solucionEncontrada.get((i +
                1) % solucionEncontrada.size())][0] + offsetX - minX) / (maxX - minX)) *
                (panelWidth - 15 - 15) + 15);
                int y2 = (int) (((ciudades[solucionEncontrada.get((i +
                1) % solucionEncontrada.size())][1] + offsetY - minY) / (maxY - minY)) *
                (panelHeight - 15 - 15) + 15);
                g2d.drawLine(x1, y1, x2, y2);
            }
        }
    }
}

```

Código 9. Representación gráfica de una solución.

Si se desea visualizar la solución plasmada en un informe, una vez que este ha sido leído y se ha extraído la información necesaria para la visualización, se añade una instancia de la nueva clase `Panel` mencionada anteriormente y se procede a dibujar la solución.

5.2.5. Algoritmos

En el sistema se implementarán dos tipos de algoritmos para resolver el TSP, permitiéndonos comparar su rendimiento: constructivos y deterministas.

5.2.5.1. Constructivos

Un algoritmo constructivo es un método de resolución de problemas de optimización que construye progresivamente una solución paso a paso. En cada iteración, el algoritmo toma decisiones locales para agregar componentes a la solución, basándose en algún criterio de elección.

Antes de entrar en la explicación de cada algoritmo implementado, procedemos a realizar una descripción del algoritmo Sistema de Hormigas (SH), en el cual se fundamentan el Sistema de Colonias de Hormigas (SCH), Sistema de Hormigas MAX-MIN (SHMM) y Sistema de Hormigas del Mejor-Peor (SHMP).

5.2.5.1.1. Sistema de Hormigas (SH)

5.2.5.1.1.1. Descripción

El algoritmo de sistema de hormigas, también conocido como "Ant Colony Optimization" (ACO), fue propuesto por Marco Dorigo en la década de 1990 y está inspirado en el comportamiento de las hormigas reales que buscan el camino más corto entre su colonia y una fuente de alimento. En el mundo real, las hormigas son ciegas y estas se mueven siguiendo los rastros de feromonas que van dejando las demás para así, gracias a la acción continuada de la colonia, poder encontrar cada vez un camino más corto desde el hormiguero a la comida. Nosotros nos basaremos en esto para encontrar el camino más corto que parte de una ciudad y pase por todas hasta volver a la inicial. Por tanto, este algoritmo pretende reproducir el comportamiento de las hormigas reales en problemas complejos de camino mínimo, donde cada agente artificial es un mecanismo probabilístico de construcción de soluciones.

Este algoritmo es comúnmente utilizado para resolver problemas de optimización combinatoria.

Para resolver el problema se hará uso de cuatro matrices:

- Matriz de ciudades: matriz con las coordenadas de cada ciudad obtenidas del fichero de datos.
- Matriz de distancias: matriz con las distancias entre cada par de ciudades. Se calcula a partir de la matriz anterior utilizando la [Ecuación 2](#) para calcular la distancia euclídea entre dos puntos n-dimensionales.

$$\text{distancia} = \sqrt{\sum_{i=1}^n (p_i - q_i)^2}$$

Ecuación 2. Cálculo de la distancia entre dos ciudades.

- Matriz heurística: matriz con el valor heurístico de cada uno de los caminos entre cada par de ciudades. Se calcula a partir de la matriz anterior según se indica en la [Ecuación 3](#).

$$\text{heurística}_{ij} = \frac{1}{\text{distancia}_{ij}} \quad \forall i, j \in \{0, n - 1\}$$

Ecuación 3. Cálculo del valor heurístico entre dos ciudades.

Se calcula de esta manera porque se supone que un arco tiene mejor heurística que otro cuando la distancia es menor.

- Matriz de feromona: matriz con el rastro de feromona para cada arco.

El funcionamiento del algoritmo se explica en el diagrama de actividades de la

[Figura 14](#).

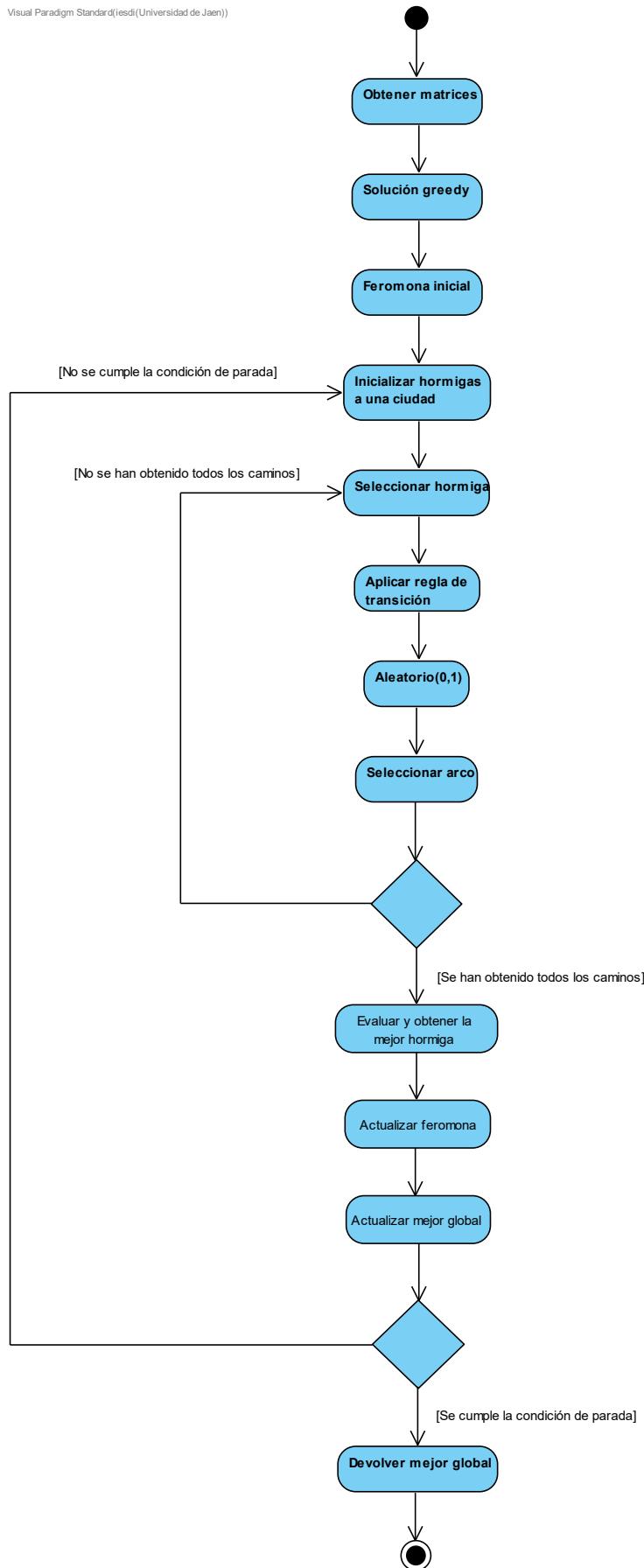


Figura 14. Diagrama de flujo del algoritmo SH. Elaboración propia.

Una vez que hemos calculado todos los datos necesarios en las matrices, buscamos una solución greedy, la cual evaluaremos para obtener el coste. Seguidamente, calculamos la feromona inicial (τ_0) como $1 / C(S) * m$, siendo m el número de hormigas.

Después, inicializamos todas las hormigas en la misma ciudad mediante un valor aleatorio y comenzamos a escoger para cada una la siguiente ciudad del recorrido. Para ello, en primer lugar, aplicamos lo que se conoce como regla de transición, esto es, calcular para cada hormiga la probabilidad con la que se moverá a cada ciudad a partir de la ciudad actual, haciendo uso de la [Ecuación 4](#).

$$p_k(r, s) = \begin{cases} \frac{[\tau_{rs}]^\alpha \cdot [\eta_{rs}]^\beta}{\sum_{u \in J_k(r)} [\tau_{ru}]^\alpha \cdot [\eta_{ru}]^\beta} & \text{si } s \in J_k(r) \\ 0 & \text{en otro caso} \end{cases}$$

Ecuación 4. Regla de transición.

Donde:

- $J_k(r)$: conjunto formado por las ciudades no visitadas.
- r : ciudad de origen.
- s : ciudad de destino.
- τ_{rs} : feromona del arco que va desde la ciudad r hasta la s .
- η_{ru} : valor heurístico del arco que va desde la ciudad r a la s .
- u : ciudad no visitada.
- τ_{rs} : feromona del arco que va desde la ciudad r hasta la u .
- η_{ru} : valor heurístico del arco que va desde la ciudad r a la u .
- α y β se utilizan para dar más importancia al valor de la feromona o a la heurística respectivamente.

Supongamos que partimos de un escenario en el que hay seis ciudades, una hormiga inicia en la primera ciudad y las probabilidades calculadas son las que se indican en la [Tabla 5](#).

Tabla 5. Ejemplo funcionamiento SH. Implementación. Elaboración propia.

| 1 | 2 | 3 | 4 | 5 | 6 | $U(0,1)$ |
|---------------|------------|----------------|---------------|---------------|------------|----------|
| Inicio | 0.322 | 0.144 | 0.144 | 0.161 | 0.227 | 0.00 |
| | [0, 0.322] | (0.322, 0.466] | (0.466, 0.61] | (0.61, 0.771] | (0.771, 1] | |

Como el número aleatorio cae en el rango de la segunda ciudad, la hormiga se movería ahí. Este proceso se repetirá para cada hormiga hasta obtener la ruta completa.

Cuando se ha construido el camino para cada hormiga, evaluamos cada una de las soluciones y nos quedamos con la de menor coste. Si esta solución tiene un costo menor que la mejor encontrada hasta el momento, la almacenamos.

Por último, procedemos a actualizar la matriz de feromonas, donde cada hormiga aportará feromona a los arcos que ha visitado en función del coste (C) de la solución que ha encontrado (S). Este aporte se calculará como $1 / C(S)$. Este proceso permitirá que, en la próxima iteración, las hormigas tiendan a visitar los arcos más prometedores, es decir, aquellos con mayor feromona. Además, se llevará a cabo la evaporación de feromona, disminuyendo su valor en los arcos menos visitados (menos prometedores). Este último paso se resume en la [Ecuación 5](#).

$$\tau_{rs}(t) = (1 - \rho) \cdot \tau_{rs}(t - 1) + \rho \cdot \sum_{k=1}^m \Delta\tau_{rs}^k$$

Ecuación 5. Actualización de feromona.

Este proceso se repetirá hasta que se cumpla la condición de parada del algoritmo, ya sea alcanzar el número de iteraciones o el tiempo de ejecución establecidos.

En conclusión, este algoritmo permite al principio una mayor exploración del espacio de búsqueda, dado que todos los arcos tienen la misma probabilidad de ser explorados. A medida que avanza, favorece la explotación de las mejores soluciones debido a los rastros de feromona más intensos dejados por la colonia.

5.2.5.1.1.2. Pseudocódigo

El esquema general del algoritmo es el que se presenta en el [Código 10.](#)

```

Para it iteraciones
    Para k=1 hasta m hormigas
        Inicializar a una ciudad
    Para i=2 hasta n nodos
        Para k=1 hasta m hormigas
            Construir solución
        Para k=1 hasta m hormigas
            Evaluar soluciones y seleccionar la mejor
        Para i=1 hasta n nodos
            Para j=1 hasta n nodos
                Actualizar feromonas
            Actualizar mejor global
    Fin iteraciones
    Devolver mejor global

```

Código 10. Pseudocódigo SH.

5.2.5.1.2. Sistema de Colonias de Hormigas (SCH)**5.2.5.1.2.1. Descripción**

El algoritmo de Sistema de Colonias de Hormigas extiende los conceptos del algoritmo de Sistema de Hormigas (SH) con tres aspectos básicos:

- Se modifica la regla de transición para buscar el equilibrio entre exploración y explotación.
- En la actualización de feromona solo aporta la mejor hormiga y se evapora en todos los arcos.
- Se añade la actualización de feromona online.

El funcionamiento de este algoritmo es el expresado en la [Figura 15.](#)

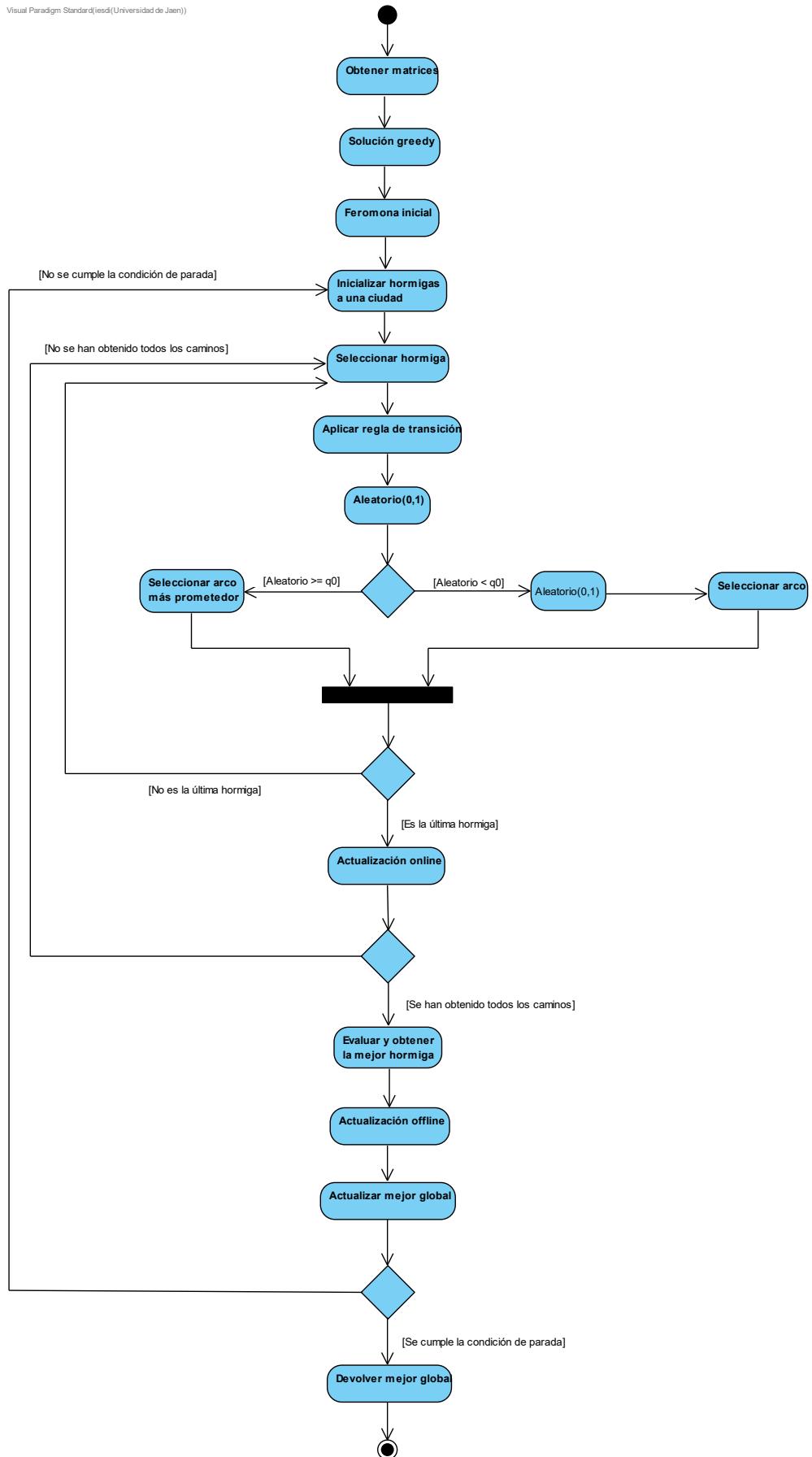


Figura 15. Diagrama de flujo del algoritmo SCh. Elaboración propia.

Como se puede atisbar, el funcionamiento del algoritmo es el mismo al del SH. La principal diferencia radica en la aplicación de la regla de transición y en la actualización de la feromona.

En cuanto a la regla de transición, se utiliza el mismo procedimiento que en el SCH, pero se introduce una modificación. Esta consiste en lanzar primero un aleatorio entre 0 y 1; si este es mayor o igual que el valor del parámetro q_0 , entonces se seleccionará el arco más prometedor, es decir, aquel que proporcione el mayor valor al aplicar la regla de transición. En caso de que esto no se cumpla, se aplicará la regla de transición como en el algoritmo de Sistema de Hormigas. Con esta modificación, se logra un mayor equilibrio entre exploración y explotación, ya que mediante una probabilidad se permite la selección de los arcos más prometedores (Gambardella, L. M., & Dorigo, M., 1997). La exploración se lleva a cabo al aplicar la regla de transición del SH, que permite seleccionar cualquier arco de los disponibles con una probabilidad asociada a cada uno. Los arcos más prometedores tendrán una mayor probabilidad de ser seleccionados, lo que facilita la explotación del espacio de soluciones.

Por otro lado, ahora se introduce la actualización de feromona local, también conocida como online, la cual consiste en evaporar en la matriz de feromona el arco que cada hormiga va seleccionando, favoreciendo así la exploración de arcos no visitados. La evaporación se hace siguiendo la [Ecuación 6](#).

$$\tau_{rs}(t) = (1 - \phi) \cdot \tau_{rs}(t - 1) + \phi \cdot \tau_0$$

donde ϕ es la fuerza de la evaporación

Ecuación 6. Actualización local de feromona.

Además, también se utiliza la [Ecuación 7](#) para realizar la actualización global, también conocida como offline, a partir del aporte de feromona de la mejor hormiga encontrada hasta el momento que se calcula como $1 / C(S)$:

$$\tau_{rs}(t) = (1 - \rho) \cdot \tau_{rs}(t - 1) + \rho \cdot \Delta\tau_{rs}^{mejor}$$

Ecuación 7. Actualización global de feromona.

Esta última actualización de feromona conducirá a que las hormigas tiendan a seleccionar los arcos de la mejor solución encontrada hasta el momento (enfoque de explotación). Cabe destacar que, para esta actualización, se pueden aplicar diversos

enfoques, como utilizar como hormiga aportante a la mejor de la iteración o un enfoque híbrido donde en algunas iteraciones aporte una u otra.

Dicho esto, se ha programado la posibilidad de configurar la hormiga aportante de feromona, es decir, se puede elegir en la interfaz si queremos que aporte feromona la mejor hormiga encontrada hasta el momento, la mejor de la iteración o ambas. En este último caso, la mejor hormiga de la iteración aportará feromona siempre, excepto cuando se cumpla un porcentaje del total de las iteraciones a ejecutar. En este caso, se utilizará la mejor hormiga encontrada hasta el momento. Este porcentaje disminuirá conforme avancen las iteraciones, guiando así a las hormigas hacia los arcos de la mejor solución encontrada durante toda la búsqueda. Todos estos parámetros son configurables.

5.2.5.1.2.2. Pseudocódigo

El esquema general del algoritmo es especificado en el [Código 11](#).

```
Para it iteraciones
    Para k=1 hasta m hormigas
        Inicializar a una ciudad
        Para i=2 hasta n nodos
            Para k=1 hasta m hormigas
                Construir solución
                Actualización online
            Para k=1 hasta m hormigas
                Evaluar soluciones y seleccionar la mejor
            Para i=1 hasta n nodos
                Para j=1 hasta n nodos
                    Actualización offline
                Actualizar mejor global
            Fin iteraciones
        Devolver mejor global
```

Código 11. Pseudocódigo SCH.

5.2.5.1.3. Sistema de Hormigas MAX-MIN (SHMM)**5.2.5.1.3.1. Descripción**

El Sistema de Hormigas Max-Min (Max-Min Ant System o SHMM) es una nueva extensión del SH con una mayor explotación de las mejores soluciones y un mecanismo adicional para evitar el estancamiento de la búsqueda.

Mantiene la regla de transición del SH y cambia:

- El mecanismo de actualización es más agresivo al evaporar todos los rastros y aportar solo en los de la mejor solución.
- Define unos límites mínimo y máximo para los rastros de feromonas.
- Reinicializa la búsqueda cuando se estanca.

Su funcionamiento se detalla en la [Figura 16](#).

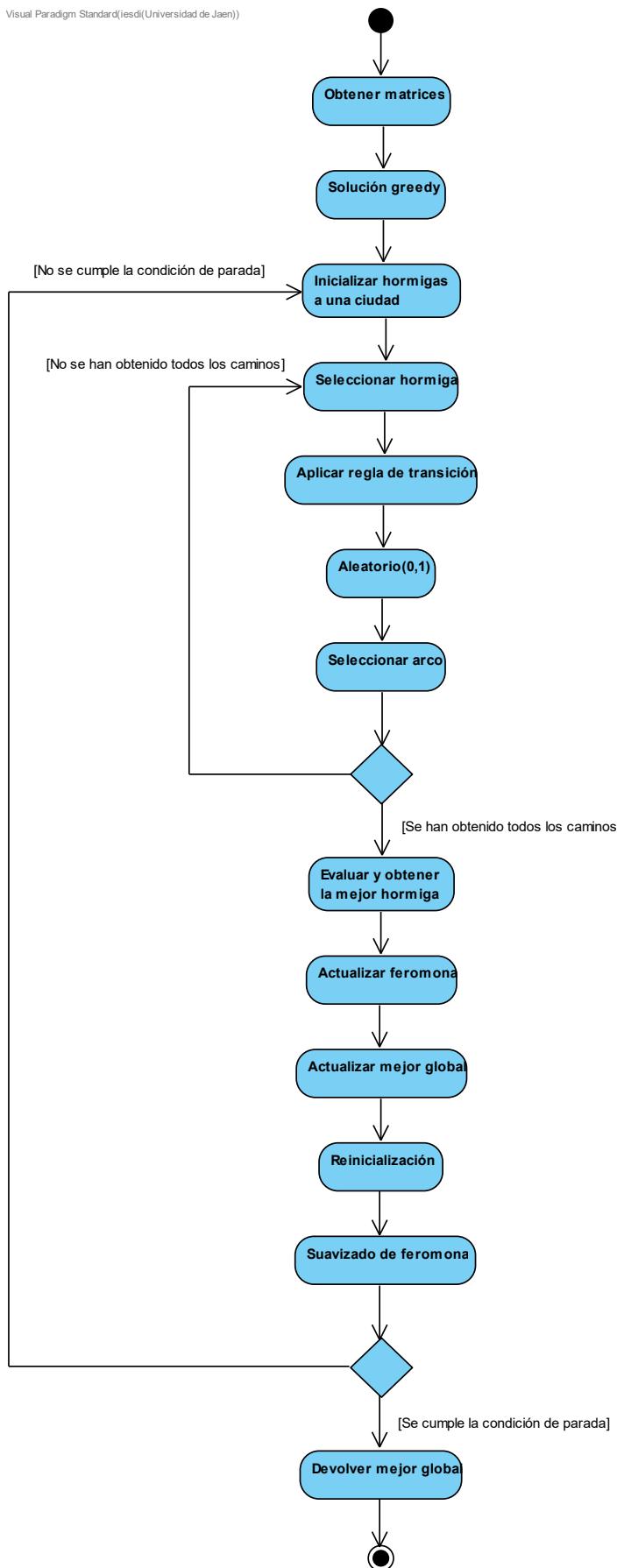


Figura 16. Diagrama de flujo del algoritmo SHMM. Elaboración propia.

Como podemos observar, el funcionamiento del algoritmo es similar al del SH. La principal diferencia radica en la actualización de la feromona, la definición de la feromona máxima y mínima, la aplicación de mecanismos de reinicialización de la búsqueda cuando esta se estanca y el suavizado de la feromona.

En primer lugar, se incluye un valor máximo y mínimo de feromona que puede tomar cada arco. En el momento en que se encuentre una solución que mejore a la mejor global, estableceremos los valores de feromona máxima y mínima.

La feromona máxima se calcula aplicando la [Ecuación 8](#).

$$\tau_{max} = \frac{1}{1 - \rho} \cdot \frac{1}{f(s^{opt})}$$

Ecuación 8. Cálculo de la feromona máxima.

Donde ρ es un parámetro configurable y $f(s^{opt})$ es el coste de la solución óptima.

La feromona mínima se calcula a partir de la [Ecuación 9](#).

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(avg - 1) \cdot \sqrt[n]{p_{best}}}$$

Ecuación 9. Cálculo de la feromona mínima.

Donde p_{best} es un parámetro configurable, n es el número de nodos y $avg = n/2$.

Una vez calculadas la feromona máxima y mínima, se aplicarán en la actualización de la feromona. Si el nuevo valor en la matriz de feromona es menor que la feromona mínima, el valor se establecerá en esta. Si el nuevo valor en la matriz de feromona es mayor que la feromona máxima, el valor se establecerá en esta. Así, al aplicar la regla de actualización, los arcos de las buenas soluciones mantienen valores altos, mientras que los de las soluciones menos prometedoras reducen el valor de sus rastros. Esto da lugar a una mayor exploración al comienzo de la ejecución del algoritmo.

En cuanto a la actualización de la feromona, se utiliza el aporte de la mejor solución y se calcula como $1 / C(S)$. Aquí es donde se restringirá el valor de la feromona en el rango $\tau_{min} \leq \tau_{rs}(t) \leq \tau_{max}$. La actualización se realiza por medio de la [Ecuación 10](#).

$$\tau_{rs}(t) = (1 - \rho) \cdot \tau_{rs}(t - 1) + \rho \cdot \Delta\tau_{rs}^{mejor}$$

Ecuación 10. Actualización de feromona.

Nuevamente, se puede considerar tanto la mejor solución global como la mejor solución de la iteración o un enfoque híbrido para el aporte. Esto es configurable.

Por otro lado, se añade un mecanismo de reinicialización que será opcional (se puede seleccionar en la interfaz). Este mecanismo se ejecutará cuando haya transcurrido un número específico de iteraciones sin encontrar una solución mejor que la mejor global. Este número de iteraciones será un porcentaje del total y será configurable. El mecanismo consistirá en establecer la feromona de cada arco a la feromona máxima, que es el valor inicial al principio del algoritmo.

Por último, se incorpora el mecanismo de suavizado de feromona que también será opcional. Este consistirá en comprobar cada cierto número iteraciones si el valor de feromona de todos los arcos de la mejor solución encontrada está muy próximo a la feromona máxima, es decir, se verificará si el algoritmo ha convergido. En ese caso, se incrementan los rastros de feromona de forma proporcional a su diferencia con la feromona máxima, como se indica en la [Ecuación 11](#).

$$\tau_{rs}(t) = \tau_{rs}(t) + \delta \cdot (\tau_{max}(t) - \tau_{rs}(t))$$

Ecuación 11. Suavizado de feromona.

Donde δ es un parámetro configurable entre 0 y 1.

De este modo, hay un mayor incremento de feromona en los arcos que tienen menos feromona, mientras que los que tienen más feromona prácticamente no se incrementan. Esto permitirá una mayor exploración del espacio de búsqueda (Stützle, T., & Hoos, H. H., 2000).

5.2.5.1.3.2. Pseudocódigo

El esquema general del algoritmo es el que se expone en el [Código 12.](#)

```

Para it iteraciones
    Para k=1 hasta m hormigas
        Inicializar a una ciudad
    Para i=2 hasta n nodos
        Para k=1 hasta m hormigas
            Construir solución
        Para k=1 hasta m hormigas
            Evaluar soluciones y seleccionar la mejor
        Calcular feromona máxima y mínima
        Para i=1 hasta n nodos
            Para j=1 hasta n nodos
                Actualizar feromona
        Reinicialización
        Suavizado de feromona
        Actualizar mejor global
    Fin iteraciones
    Devolver mejor global

```

Código 12. Pseudocódigo SHMM.

5.2.5.1.4. Sistema de Hormigas Mejor-Peor (SHMP)**5.2.5.1.4.1. Descripción**

El Sistema de Hormigas Mejor-Peor (Best-Worst Ant System o SHMP) es otra extensión del SH basada en la incorporación de componentes de Computación Evolutiva para mejorar el equilibrio exploración-explotación.

Mantiene la regla de transición del SH y cambia:

- El mecanismo de actualización es más explotativo al evaporar todos los rastros, reforzar positivamente sólo los de la mejor solución global y negativamente los de la peor solución actual.
- Aplica una mutación de los rastros de feromona para explorar.
- Reinicializa la búsqueda cuando se estanca.

El comportamiento de este algoritmo de plasma en la [Figura 17.](#)

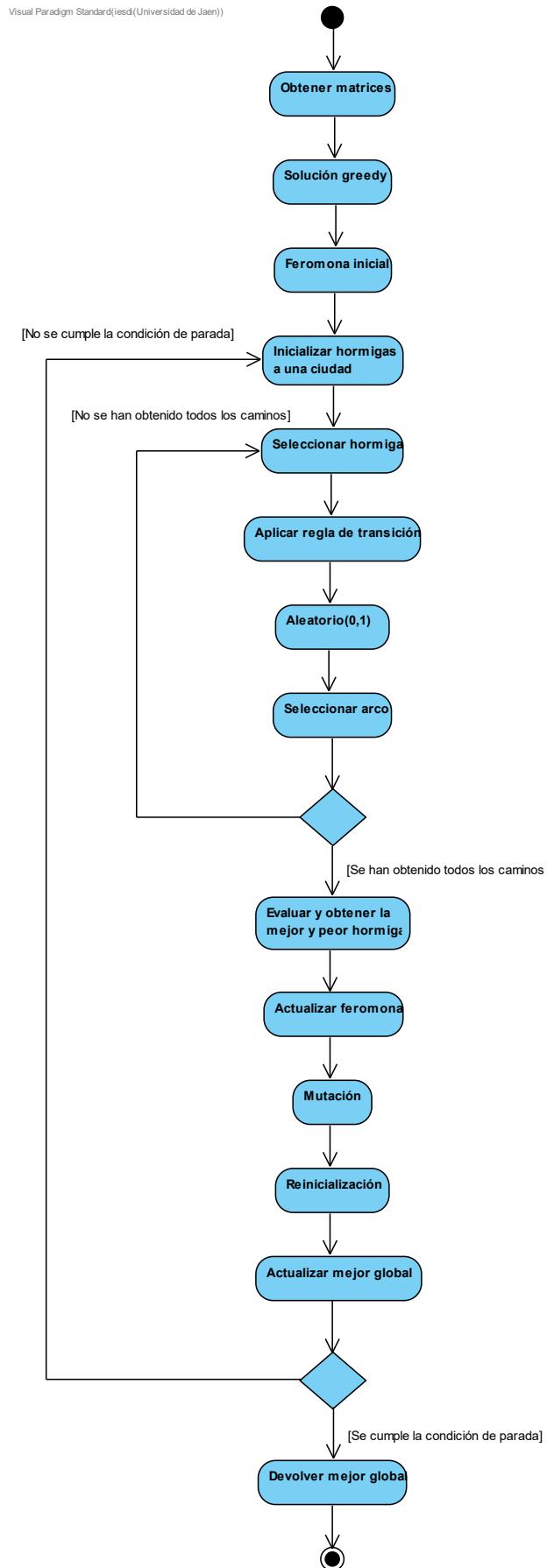


Figura 17. Diagrama de flujo del algoritmo SHMP. Elaboración propia.

Como se aprecia, el funcionamiento del algoritmo es similar al del SH. La principal diferencia radica en la actualización de la feromona, la mutación de los rastros de feromona y la aplicación de mecanismos de reinicialización de la búsqueda cuando esta se estanca.

En relación a la reinicialización, el SHMP considera la búsqueda estancada si durante un número consecutivo de iteraciones (un porcentaje del total) no se consigue mejorar la mejor solución global obtenida. Este porcentaje será configurable. En ese caso, se aplica la reinicialización, volviendo a poner todos los rastros de feromona a τ_0 .

Por otro lado, en cuanto a la actualización de feromona, una vez obtenidas la mejor solución global y la peor de la iteración, se evaporan todos los rastros de feromona y se aporta en los de la mejor solución global, tal y como se indica en la [Ecuación 12](#).

$$\tau_{rs}(t) = (1 - \rho) \cdot \tau_{rs}(t - 1) + \Delta\tau_{rs}^{mejor_global}$$

Ecuación 12. Actualización de feromona.

A continuación, se realiza una evaporación adicional de los rastros de feromona de la peor solución de la iteración actual que no estén contenidos en la mejor global, según la [Ecuación 13](#).

$$\tau_{rs}(t) = (1 - \rho) \cdot \tau_{rs}(t), \forall a_{rs} \in S_{peor_actual} \text{ y } a_{rs} \notin S_{mejor_global}$$

Ecuación 13. Evaporación de feromona en la peor solución.

El refuerzo negativo de S_{peor_actual} hace que la regla de actualización tenga un comportamiento de explotación (Cordón, O., Fernández de Viana, I., Herrera, F., & Moreno, Ll., 2000).

Por otro lado, para conseguir diversidad en el proceso de búsqueda, se mutan los valores de los rastros de feromona. La mutación se aplica en cada rastro de feromona con una probabilidad P_m , de la manera que se indica en la [Ecuación 14](#).

$$\tau_{rs}'(t) = \tau_{rs}(t) + N(0, \tau_{umbral}); \tau_{umbral} = \frac{c(S_{mejor_global})}{n}$$

Ecuación 14. Mutación de feromona.

Donde $N(0, \tau_{\text{umbral}})$ es un valor normal entre 0 y la feromona umbral y n es el número de nodos.

A cada rastro mutado se le suma un valor normal entre $[-\tau_{\text{umbral}}, \tau_{\text{umbral}}]$. τ_{umbral} corresponde a la media de los rastros de feromona de $S_{\text{mejor_global}}$.

Esta función de mutación se caracteriza por:

- La fuerza de la mutación aumenta con las iteraciones: primero, τ_{umbral} es cercano a τ_0 y la mutación es pequeña. Luego, según crecen los rastros de $S_{\text{mejor_global}}$ va siendo más grande.
- Al reinicializar, vuelve a su rango inicial.

El SHMP consigue un buen balance diversificación-intensificación combinando:

- La intensificación que introduce la regla de actualización de feromona con la mejor y la peor hormiga.
- La diversificación de la mutación de rastros de feromona y la reinicialización.

5.2.5.1.4.2. Pseudocódigo

En el [Código 13](#) se detalla el esquema genérico del funcionamiento de este algoritmo.

```

Para it iteraciones
    Para k=1 hasta m hormigas
        Inicializar a una ciudad
    Para i=2 hasta n nodos
        Para k=1 hasta m hormigas
            Construir solución
        Para k=1 hasta m hormigas
            Evaluar soluciones y seleccionar la mejor y la
            peor
        Para i=1 hasta n nodos
            Para j=1 hasta n nodos
                Actualizar feromona
                Mutación
            Reinicialización
            Actualizar mejor global
        Fin iteraciones
    Devolver mejor global

```

Código 13. Pseudocódigo SHMP.

5.2.5.2. Deterministas

Los algoritmos deterministas o clásicos son computacionalmente muy costosos para instancias grandes, lo que ha llevado al desarrollo de algoritmos aproximados más eficientes, como los sistemas de hormigas.

5.2.5.2.1. Algoritmo Lin-Kernighan

5.2.5.2.1.1. Descripción

El algoritmo 2-opt es un caso especial del algoritmo λ -opt, donde en cada paso λ enlaces del recorrido actual se reemplazan por λ enlaces de tal manera que se consigue un recorrido más corto. En otras palabras, en cada paso se obtiene un recorrido más corto eliminando λ enlaces y juntando los caminos resultantes de una forma nueva, posiblemente revirtiendo uno o más de ellos.

El algoritmo λ -opt se basa en el concepto λ -optimalidad: Se dice que un recorrido es λ -óptimo (o simplemente λ -opt) si es imposible obtener un recorrido más corto reemplazando cualquiera de sus enlaces por cualquier otro conjunto de λ enlaces (Chin-Chuan, L., & Kernighan, B. W., 1973).

A partir de esta definición es obvio que cualquier recorrido λ -óptimo también es λ' -óptimo para $1 \leq \lambda' \leq \lambda$. También es fácil ver que un recorrido que contiene n ciudades es óptimo si y sólo si es n -óptimo.

En general, cuanto mayor sea el valor de λ , más probable será que el recorrido final sea óptimo. Para λ bastante grande parece, al menos intuitivamente, que un recorrido λ -óptimo debería ser óptimo.

Desafortunadamente, aumenta rápidamente el número de operaciones para probar todos los λ -intercambios a medida que aumenta el número de ciudades. En una implementación ingenua, las pruebas de un λ -intercambio tiene una complejidad temporal de $O(n^{\lambda})$. Además, no hay límite superior no trivial del número de λ -intercambios. Como resultado, los valores $\lambda = 2$ y $\lambda = 3$ son los más utilizados.

Sin embargo, es un inconveniente que λ deba especificarse de antemano. Es difícil saber qué λ usar para lograr el mejor compromiso entre el tiempo de ejecución y calidad de la solución.

Lin y Kernighan eliminaron este inconveniente introduciendo un poderoso algoritmo de λ -opt variable. El algoritmo cambia el valor de λ durante su ejecución, decidir en cada iteración cuál debería ser el valor de λ . En cada paso de iteración el algoritmo examina, para valores ascendentes de λ , si un intercambio de λ enlaces puede resultar en un recorrido más corto. Dado que el intercambio de r enlaces está siendo considerado, se realiza una serie de pruebas para determinar si un intercambio de $r+1$ enlaces debe ser considerado. Esto continúa hasta que se cumplen algunas condiciones de parada.

En cada paso, el algoritmo considera un conjunto creciente de intercambios potenciales (comenzando con $r = 2$). Estos intercambios se eligen de tal manera que un recorrido factible pueda formarse en cualquier etapa del proceso. Si la exploración tiene éxito encontrando un nuevo recorrido más corto, entonces el recorrido real se reemplaza con el nuevo recorrido (Chin-Chuan, L., & Kernighan, B. W., 1973).

El algoritmo de Lin-Kernighan pertenece a la clase de los llamados algoritmos de optimización local. El algoritmo se especifica en términos de intercambios (o movimientos) que pueden convertir un recorrido en otro. Dado un recorrido factible, el algoritmo realiza repetidamente intercambios que reducen la duración del recorrido actual, hasta llegar a un recorrido en el que ningún intercambio produce una mejora. Este proceso puede repetirse muchas veces a partir de recorridos iniciales generados de manera aleatoria. El algoritmo se describe a continuación con más detalle.

Sea T el recorrido actual. En cada paso de iteración el algoritmo intenta encontrar dos conjuntos de enlaces, $X = \{x_1, \dots, x_r\}$ e $Y = \{y_1, \dots, y_r\}$, tales que, si los enlaces de X se eliminan de T y se reemplazan por los enlaces de Y , el resultado es un mejor recorrido. Este intercambio de enlaces se llama movimiento r -opt y se ejemplifica en la [Figura 18](#).

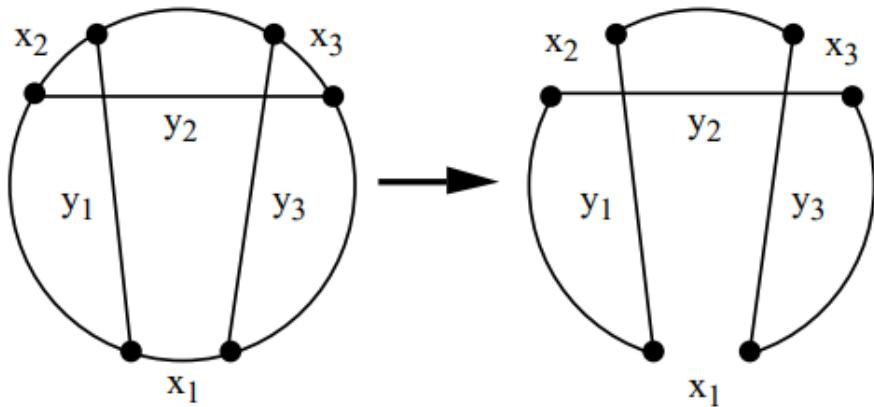


Figura 18. Movimiento r-opt Lin-Kernighan. Implementación (Chin-Chuan, L., & Kernighan, B. W., 1973).

Los dos conjuntos X e Y se construyen elemento por elemento. Inicialmente X e Y están vacíos. En el paso i un par de enlaces, x_i y y_i , son añadidos a X e Y, respectivamente («seas.gwu.edu», 2023).

Para lograr un algoritmo suficientemente eficiente, sólo los enlaces que cumplan con los siguientes criterios pueden entrar en X e Y.

1. Criterio de intercambio secuencial.

x_i y y_i deben compartir un nodo, al igual que y_i y x_{i+1} . Si t_1 denota uno de los dos puntos que forman x_1 , tenemos que: $x_i = (t_{2i-1}, t_{2i})$, $y_i = (t_{2i}, t_{2i+1})$ y $x_{i+1} = (t_{2i+1}, t_{2i+2})$ para $i \geq 1$. Este principio se ilustra en la [Figura 19](#).

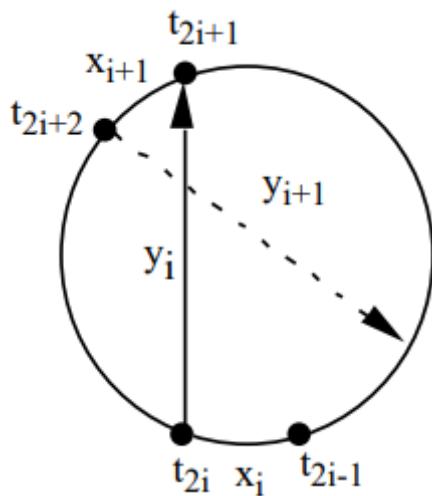


Figura 19. Intercambio secuencial Lin-Kernighan. Implementación (Chin-Chuan, L., & Kernighan, B. W., 1973).

Como se ve, la secuencia $(x_1, y_1, x_2, y_2, x_3, \dots, x_r)$ constituye una cadena de enlaces adyacentes.

Una condición necesaria (pero no suficiente) de que el intercambio de enlaces de X con enlaces de Y resulte en un recorrido es que la cadena está cerrada, es decir, $y_r = (t_{2r}, t_1)$. Este intercambio se llama secuencial.

Generalmente, una mejora de un recorrido se puede lograr mediante un intercambio secuencial mediante una numeración adecuada de los enlaces afectados. Sin embargo, este no es siempre es el caso. En la [Figura 20](#) se muestra un ejemplo en el que un intercambio secuencial no es posible.

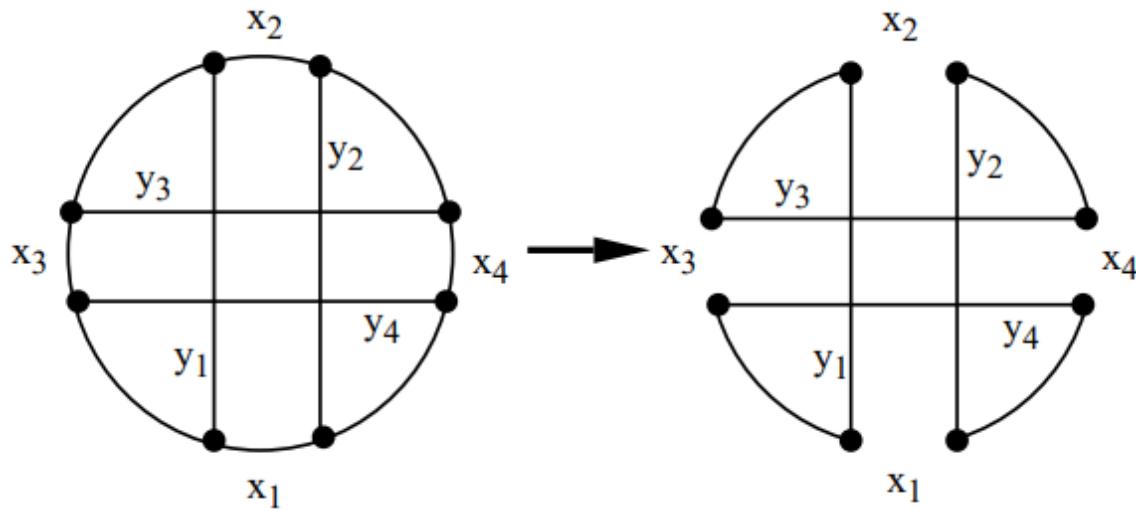


Figura 20. Intercambio secuencial no posible Lin-Kernighan. Implementación (Chin-Chuan, L., & Kernighan, B. W., 1973).

2. Criterio de factibilidad.

Se requiere que se elija un $x_i = (t_{2i-1}, t_{2i})$ de modo que, si t_{2i} se une a t_1 , la configuración resultante sea un recorrido. Este criterio se utiliza para $i \geq 3$ y garantiza que es posible cerrar un recorrido. Este criterio fue incluido en el algoritmo tanto para reducir el tiempo de ejecución como para simplificar la codificación.

3. Criterio de ganancia positiva.

Se requiere que y_i siempre se elija de modo que la ganancia, G_i , del conjunto propuesto de intercambios sea positiva. Supongamos que $g_i = c(x_i) - c(y_i)$ es la ganancia del intercambio de x_i con y_i . Entonces G_i es la suma $g_1 + g_2 + \dots + g_i$.

Este criterio de parada juega un papel importante en la eficiencia del algoritmo. El exigir que cada suma parcial, G_i , deba ser positiva parece ser muy restrictivo. Sin embargo, que esto no es así se deduce del siguiente simple hecho: si una secuencia

de números tiene una suma positiva, hay una permutación cíclica de estos números de modo que cada suma parcial sea positiva.

4. Criterio de disjuntividad.

Finalmente, se requiere que los conjuntos X e Y sean disjuntos. Esto simplifica la codificación, reduce el tiempo de funcionamiento y proporciona un criterio de parada eficaz.

Un cuello de botella del algoritmo es la búsqueda de enlaces para añadir a los conjuntos X e Y.

Por lo tanto, para aumentar la eficiencia, se debe tener especial cuidado limitando esta búsqueda. Sólo se deben considerar los intercambios que tengan una probabilidad razonable de dar lugar a una reducción de la duración del recorrido.

5.2.5.2.1.2. Pseudocódigo

El esquema general del algoritmo es el que se muestra en el [Código 14](#):

1. Generar solución inicial aleatoria
2. Establecer $i = 1$. Seleccionar t_1
3. Seleccionar $x_1 = (t_1, t_2) \in T$
4. Seleccionar $y_1 = (t_2, t_3) \notin T$ con $G_1 > 0$
Si no es posible, ir al paso 12
5. Establecer $i = i + 1$
6. Seleccionar $x_i = (t_{2i-1}, t_{2i}) \in T$ tal que
 - a) Si t_{2i} está conectado a t_1 , la configuración resultante es un camino, T' , y
 - b) $x_i \neq y_s$ para todo $s < i$
 Si T' es un camino mejor que T , establecer $T = T'$ e ir al paso 2
7. Seleccionar $y_i = (t_{2i}, t_{2i+1}) \notin T$ tal que
 - a) $G_i > 0$
 - b) $y_i \neq x_s$ para todo $s \leq i$, y
 - c) x_{i+1} existe
 Si y_i existe, ir al paso 5
8. Si existe una alternativa no probada para y_2 , establecer $i = 2$ e ir al paso 7
9. Si existe una alternativa no probada para x_2 , establecer $i = 2$ e ir al paso 6
10. Si existe una alternativa no probada para y_1 , establecer $i = 1$ e ir al paso 4
11. Si existe una alternativa no probada para x_1 , establecer $i = 1$ e ir al paso 3
12. Si existe una alternativa no probada para t_1 , ir al paso 2
13. Parar (o ir al paso 1)

Código 14. Pseudocódigo Lin-Kerninghan.

5.3. Front-end

Para la implementación de la interfaz de usuario se utilizará Java Swing. Java Swing es un conjunto de bibliotecas gráficas y herramientas proporcionadas por Java para la creación de interfaces gráficas de usuario (GUI). Forma parte del paquete javax.swing, y permite a los desarrolladores construir aplicaciones de escritorio interactivas y visualmente atractivas en entornos Java. Swing es una mejora de las bibliotecas gráficas anteriores de Java (como AWT) y proporciona componentes de interfaz de usuario más avanzados, mayor flexibilidad y capacidad de personalización. Los desarrolladores pueden construir ventanas, botones, menús, cuadros de diálogo y otros elementos de interfaz de usuario mediante clases y métodos de Swing. Además, Swing es independiente de la plataforma, lo que significa que las aplicaciones desarrolladas con Swing pueden ejecutarse en diferentes sistemas operativos sin modificaciones significativas.

Subsecuentemente, para cada una de las vistas de usuario definidas en el apartado [4.3](#) se indica cada uno de los componentes de Java Swing utilizados para su implementación.

5.3.1. Vista de inicio

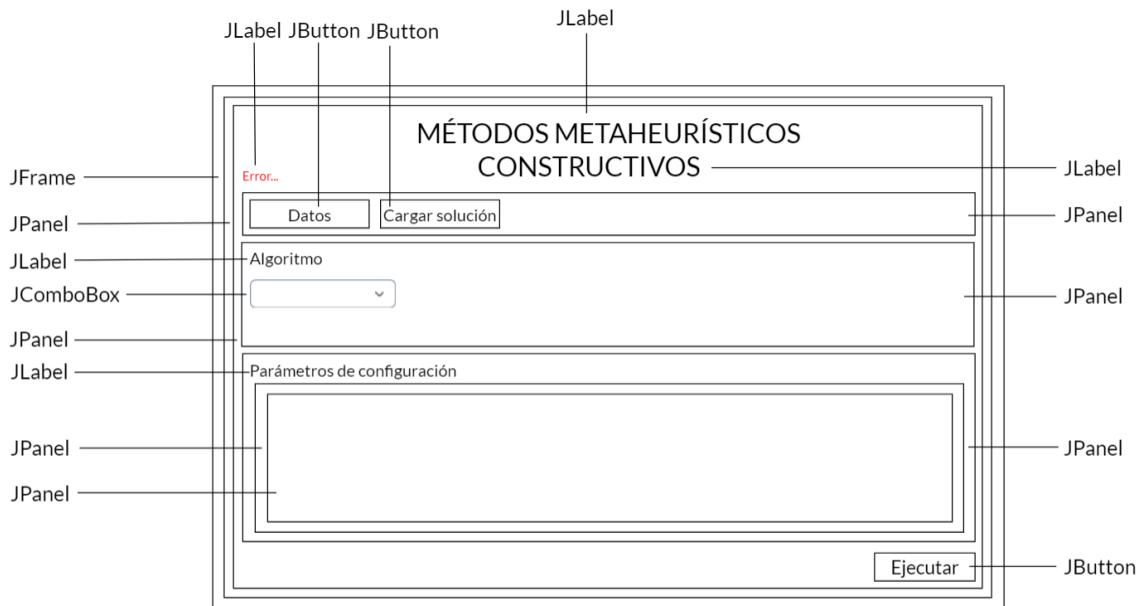


Figura 21. Implementación vista inicio. Implementación. Elaboración propia.

La base de la interfaz de usuario es un `JFrame` sobre el cual se colocarán los distintos componentes proporcionados por Java Swing. La idea que se ha seguido

para conseguir que la interfaz sea dinámica, es decir, que su apariencia cambie, es utilizar objetos de la clase `JPanel` estableciéndolos como `CardLayout`. Eso nos permitirá situar sobre estos otros paneles en modo `AbsoluteLayout` para que se acoplen al tamaño de este. Estos paneles se van a ir ocultando o mostrando en función de la interacción el usuario con la interfaz y nos permitirán el cambio de pantalla.

Por tanto, el primer elemento añadido al `JFrame` es un `JPanel` en modo `CardLayout`. Sobre este, se añade otro panel en modo `AbsoluteLayout` para representar la vista de la [Figura 21](#). Sobre este panel, se añadirán el resto de elementos necesarios para esta vista. Dado que en los parámetros de configuración van a cambiar en función del algoritmo seleccionado, esto se implementará mediante objetos `JPanel`, como se ha explicado previamente. En esta vista, no aparecen parámetros puesto que no se ha seleccionado todavía ningún algoritmo, por lo que se coloca un panel vacío.

Es relevante notar que el mensaje de error que aparece en rojo encima de los botones “Datos” y “Cargar solución” inicialmente estará oculto y solo se visualizará cuando se haya seleccionado un conjunto de datos o un informe erróneo.

El resultado de aplicar todo esto en código es el que se presenta en la [Figura 22](#).

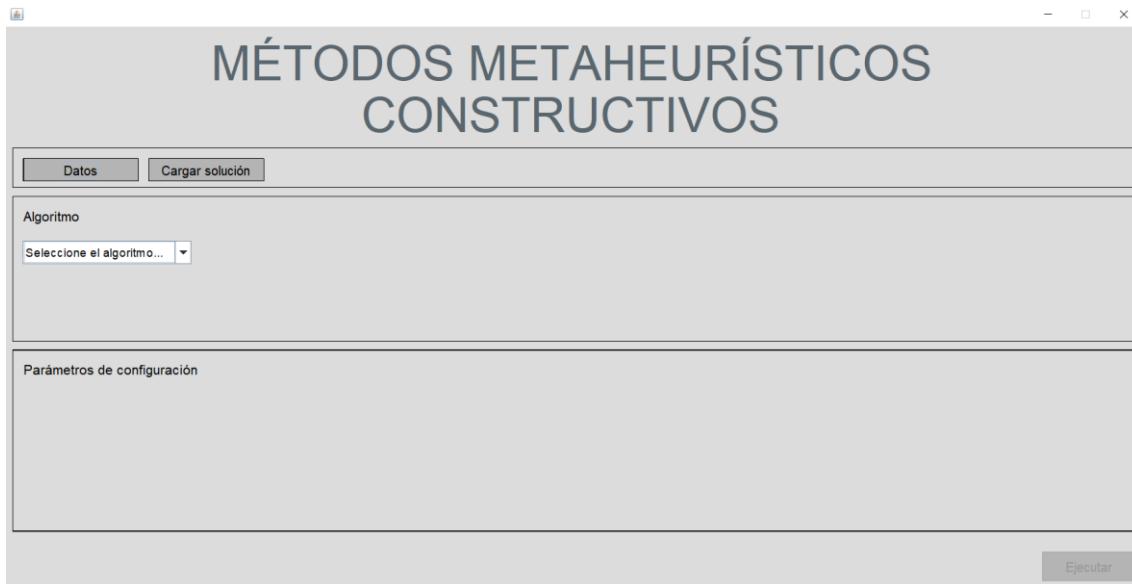


Figura 22. Resultado vista inicio. Implementación. Elaboración propia.

En la [Figura 23](#) se muestra cómo aparece el error al introducir un conjunto de datos incorrecto.

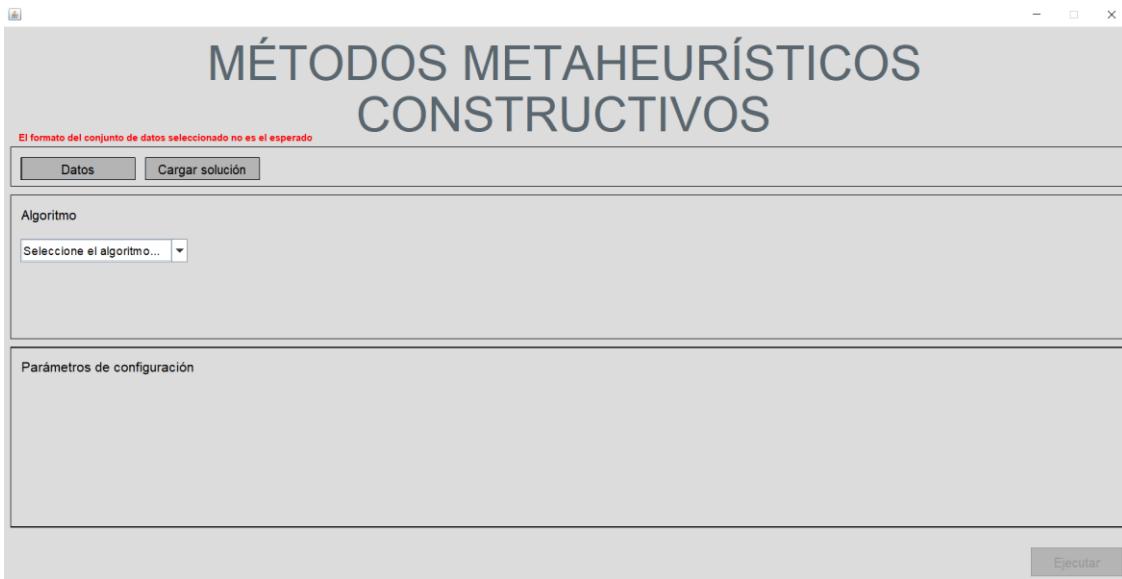


Figura 23. Conjunto de datos erróneo. Implementación. Elaboración propia.

Acto seguido, se muestra en la [Figura 24](#) cómo aparece el error al introducir un informe incorrecto.

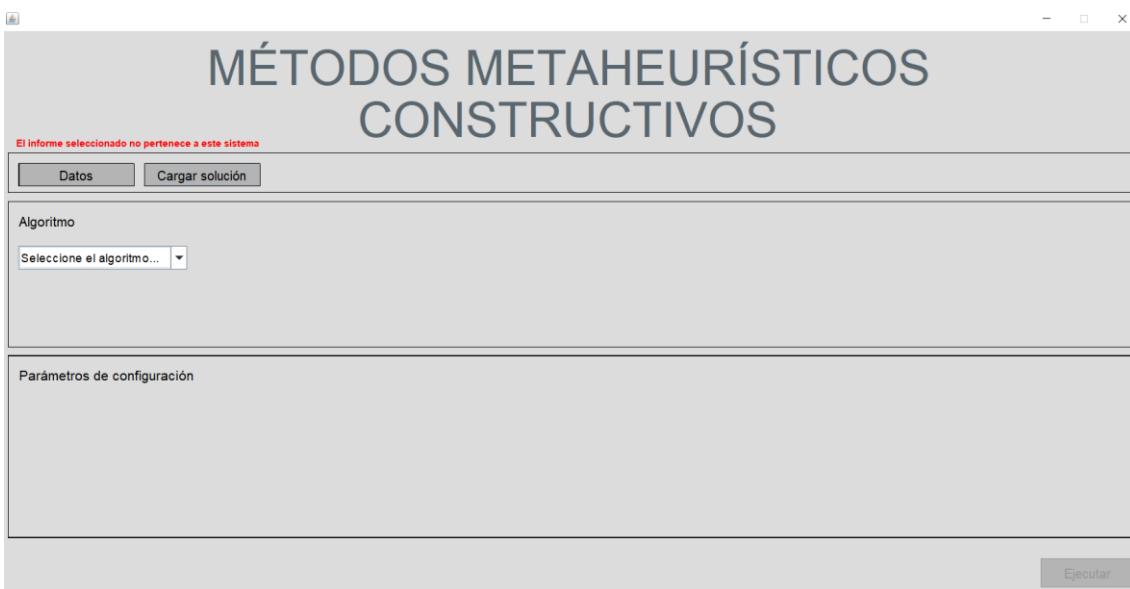


Figura 24. Informe erróneo. Implementación. Elaboración propia.

5.3.2. Vista de configuración

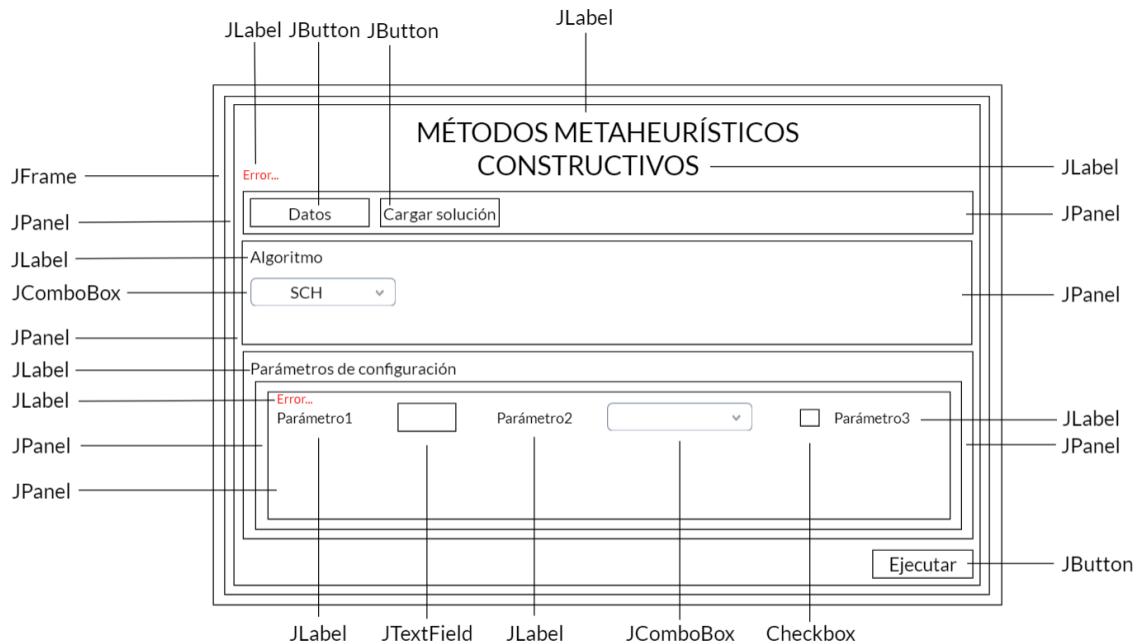


Figura 25. Implementación vista configuración. Implementación. Elaboración propia.

Dado que en la aplicación se han incorporado cuatro algoritmos, se realizará un análisis genérico de los componentes de Java Swing utilizados en la vista de configuración, la cual se corresponde con la [Figura 25](#).

Esta vista se basa en la vista anterior. La única diferencia que tiene es que los JPanel donde se sitúan los parámetros se van ocultando o mostrando en función del algoritmo seleccionado.

Para cada algoritmo, se muestran sus parámetros con un JLabel y un JTextField, JComboBox o Checkbox en función de si se requiere una entrada de texto, una selección de un desplegable o un click para indicar que se haga algo, respectivamente.

Encima de aquellos parámetros que requieran una entrada de texto del usuario por teclado, se les ha añadido encima un JLabel para indicar que el parámetro es obligatorio (en el caso de que lo sea y esté en blanco) o es erróneo, en el caso de que el usuario lo deje vacío o introduzca algún valor inapropiado. Este JLabel inicialmente estará oculto.

El resultado de implementar esto en código para el algoritmo SCH se presenta en la [Figura 26](#).

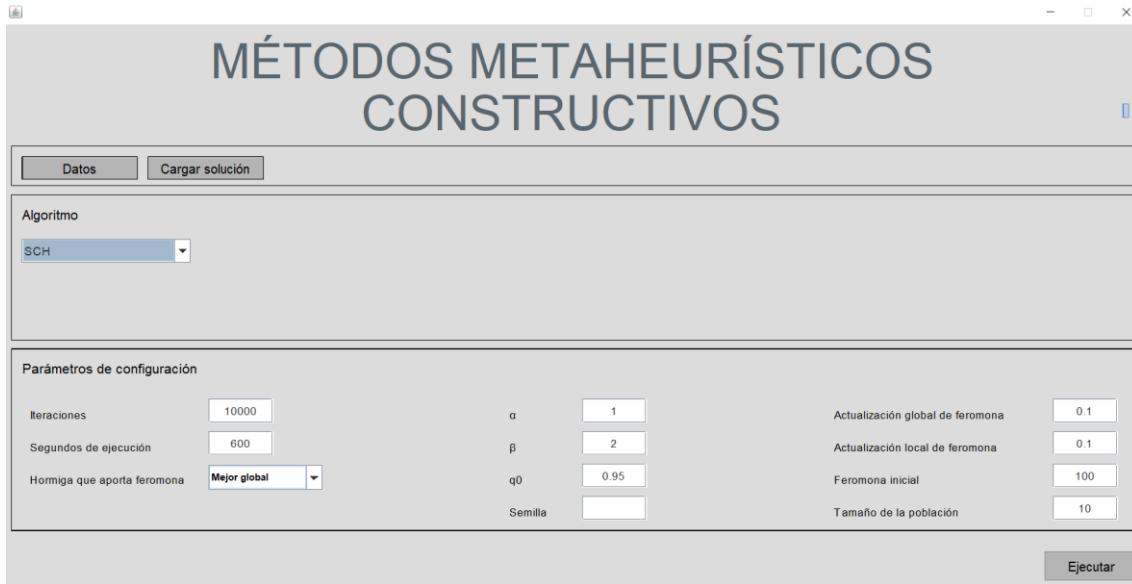


Figura 26. Resultado vista configuración SCH. Implementación. Elaboración propia.

A continuación de esto, se muestra en la [Figura 27](#) cómo se activa un mensaje de error al dejar en blanco un parámetro que es obligatorio para el algoritmo seleccionado.

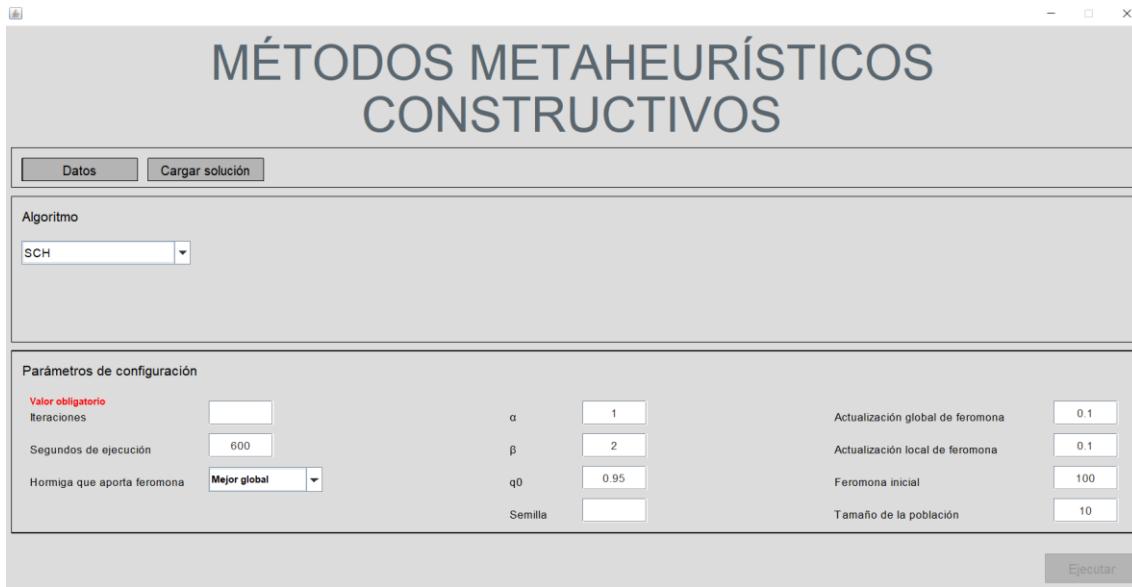


Figura 27. Error parámetro obligatorio. Implementación. Elaboración propia.

En la [Figura 28](#) se muestra cómo se activa un mensaje de error al introducir un valor incorrecto para un parámetro.

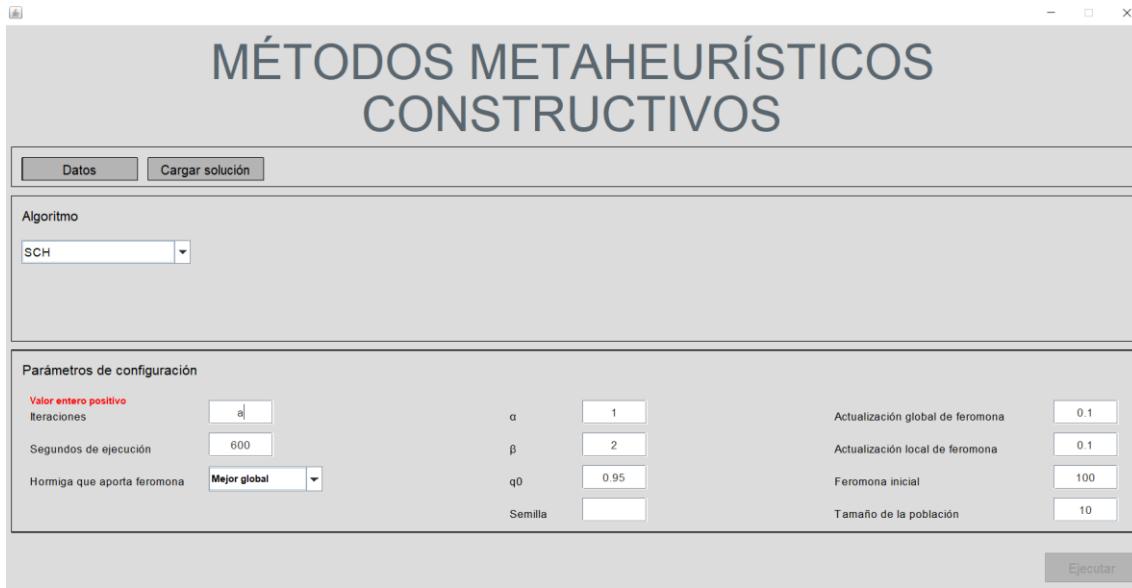


Figura 28. Error valor de parámetro incorrecto. Implementación. Elaboración propia.

Para concluir este subapartado, se presentan las vistas de configuración para los algoritmos SHMM, SHMP y Lin-Kerninghan en las figuras [29](#), [30](#) y [31](#), respectivamente.

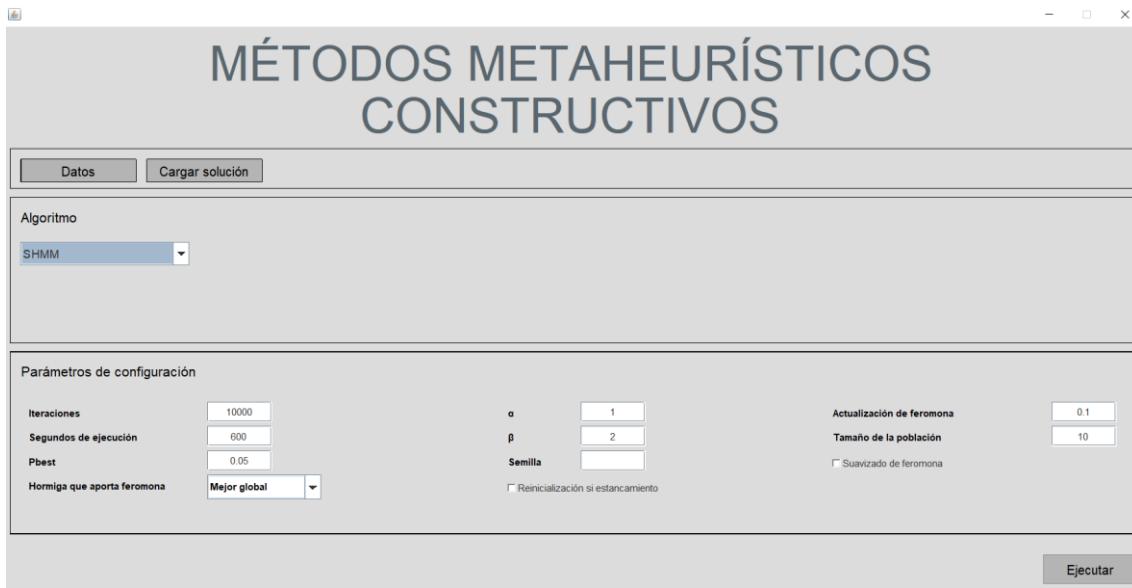


Figura 29. Resultado vista configuración SHMM. Implementación. Elaboración propia.

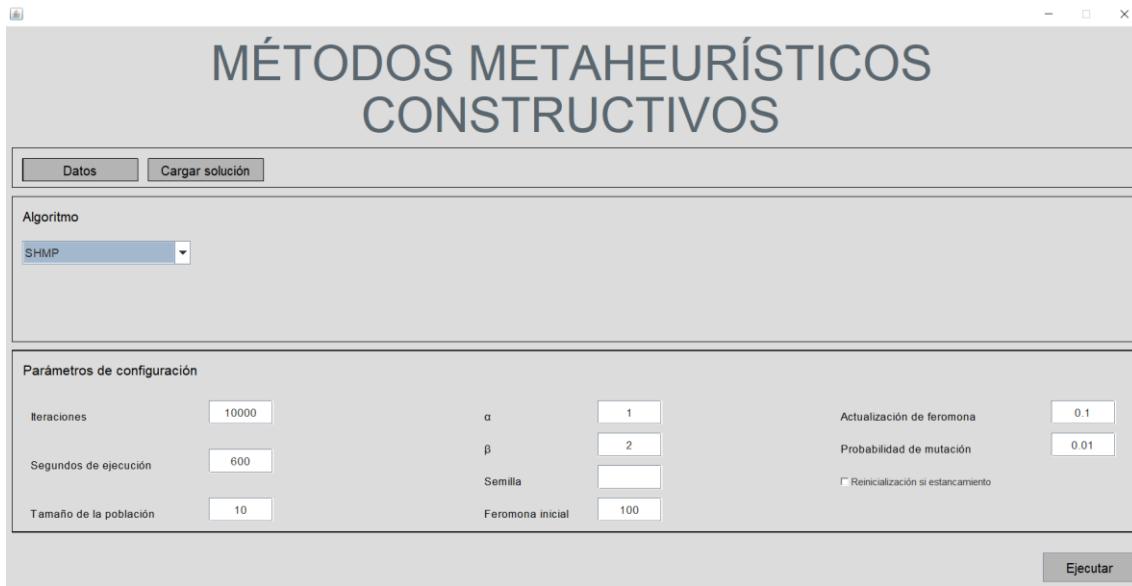


Figura 30. Resultado vista configuración SHMP. Implementación. Elaboración propia.

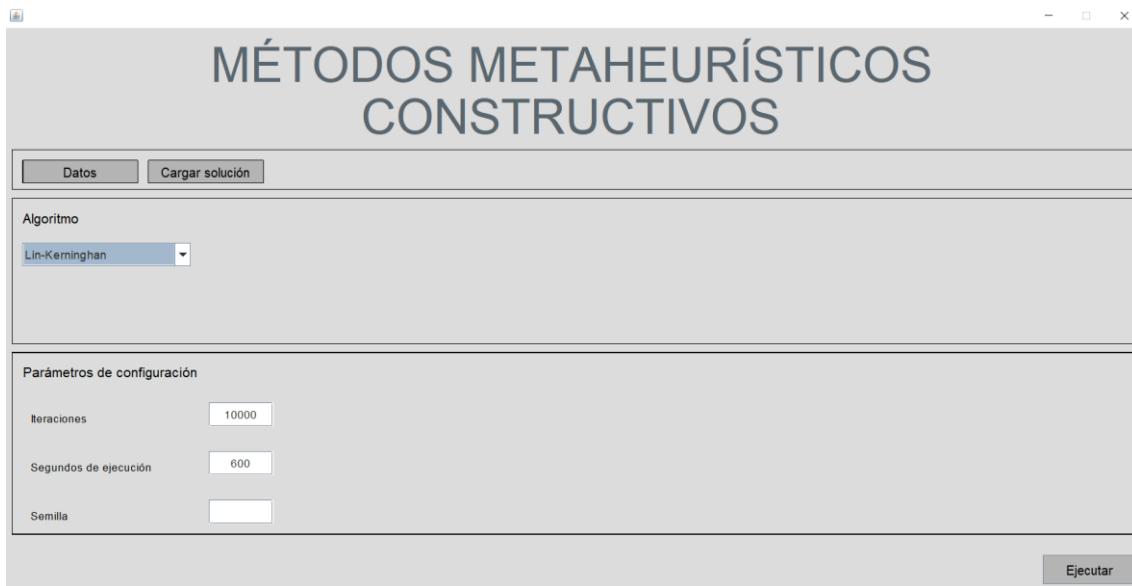


Figura 31. Resultado vista configuración Lin-Kerninghan. Implementación. Elaboración propia.

5.3.3. Vista de ejecución y visualización

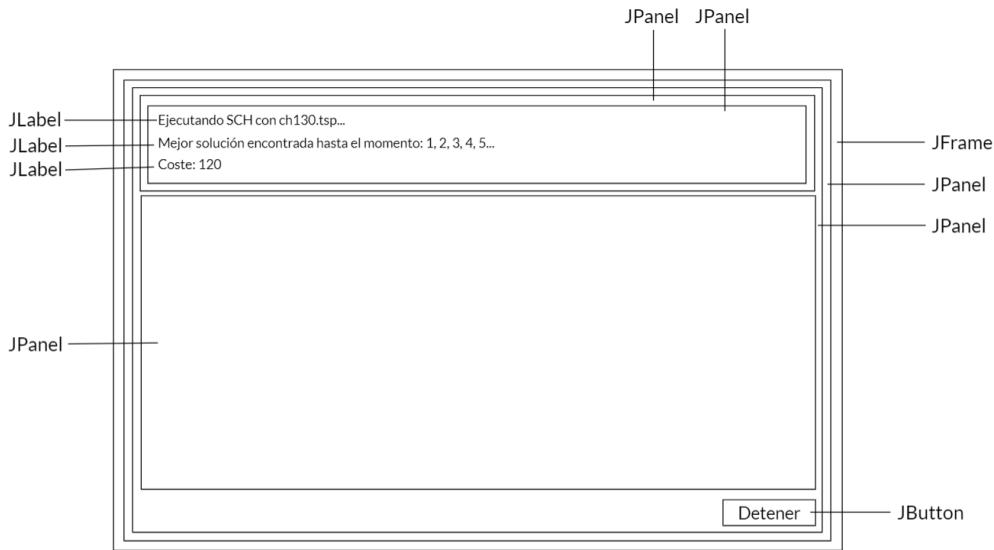


Figura 32. Implementación vista ejecución y visualización. Implementación. Elaboración propia.

La implementación de la vista de la [Figura 32](#) es muy similar a la implementación de las vistas ya presentadas. A pesar de ello, lo más interesante a recalcar es que, nuevamente, utilizaremos el mecanismo de ocultación de paneles en la parte donde se muestra el texto con el algoritmo que se está ejecutando y la solución que se va encontrando. Cuando la ejecución del algoritmo finalice, dicho panel se ocultará para visualizar otro con los resultados de la ejecución. Asimismo, el texto del botón que se encuentra en la parte inferior derecha cambiará tras el término del algoritmo.

El aspecto visual final de esta vista se ilustra en la [Figura 33](#).

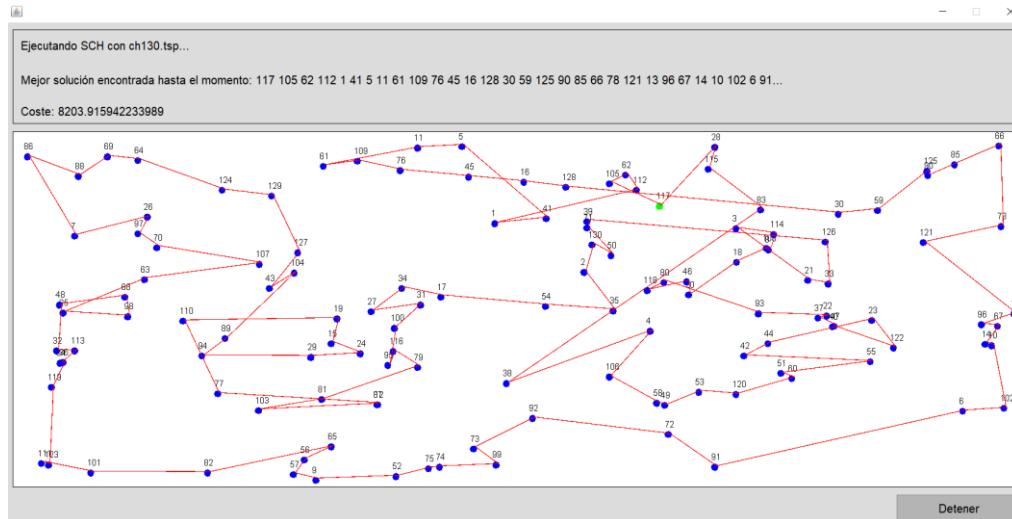


Figura 33. Resultado vista ejecución y visualización. Implementación. Elaboración propia.

5.3.4. Vista de ejecución completada

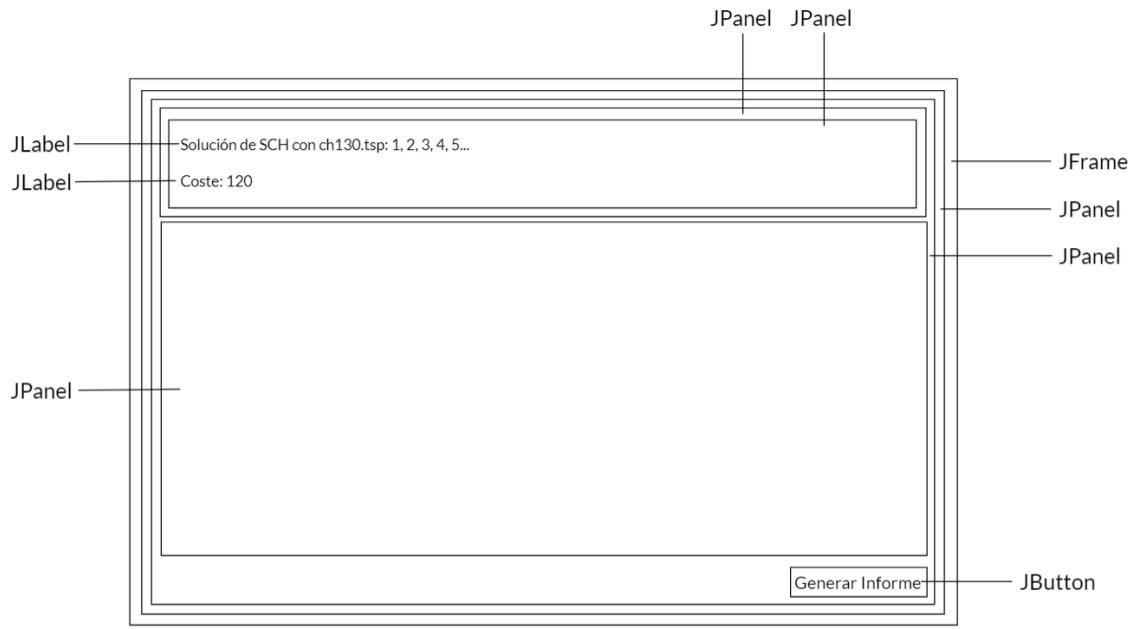


Figura 34. Implementación vista ejecución completada. Implementación. Elaboración propia.

Como ya se ha mencionado en el subapartado previo, cuando el algoritmo termina su ejecución, se oculta el panel de texto y muestra el nuevo, al mismo tiempo que cambia el texto del botón (junto con su funcionamiento). El diseño de esta vista corresponde a la [Figura 34](#). El resultado de implementar esta vista se clarifica en la [Figura 35](#).

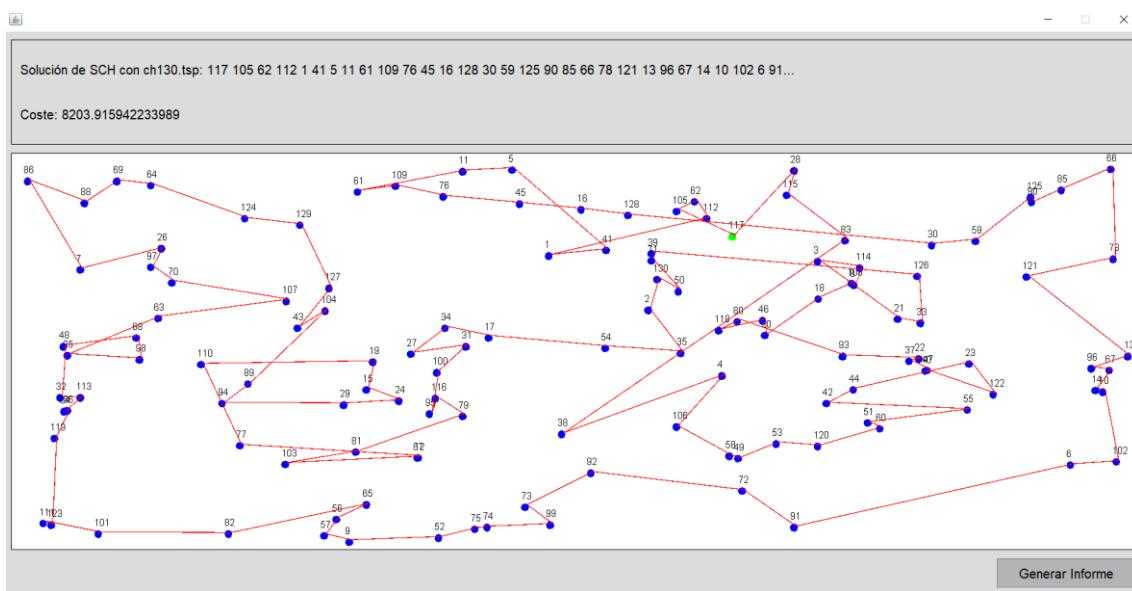


Figura 35. Resultado vista ejecución completada. Implementación. Elaboración propia.

5.3.5. Vista de visualización

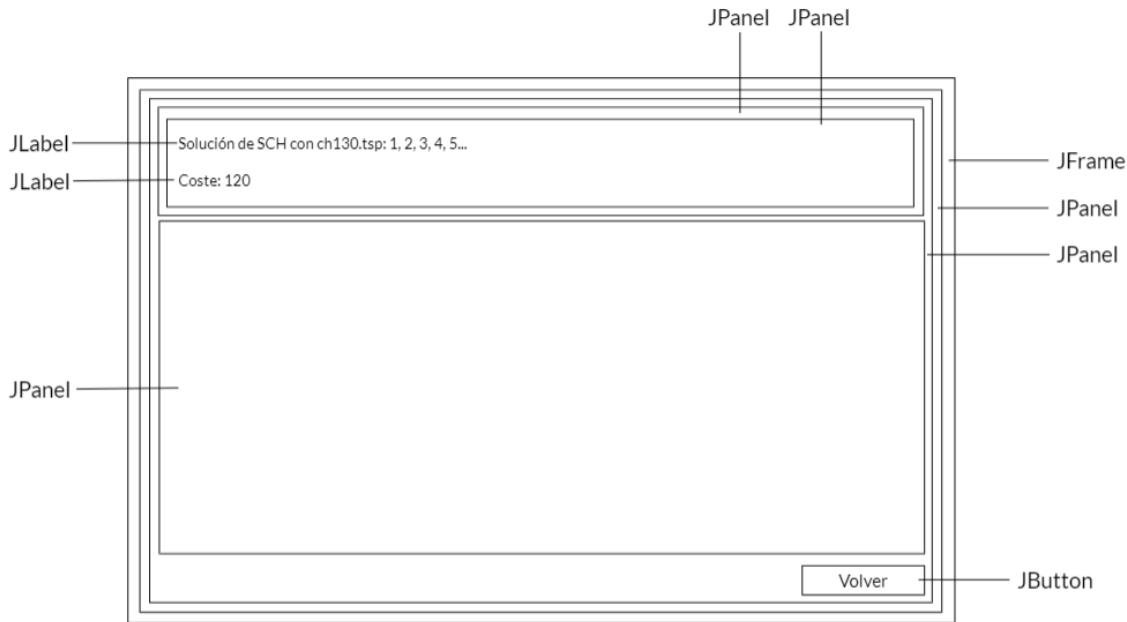


Figura 36. Implementación vista visualización. Implementación. Elaboración propia.

La vista de la [Figura 36](#) sigue los mismos principios que la vistas de ejecución del algoritmo y la vista que muestra los resultados una vez ejecutado el mismo. Como resultado de esta implementación, se obtiene la [Figura 37](#).

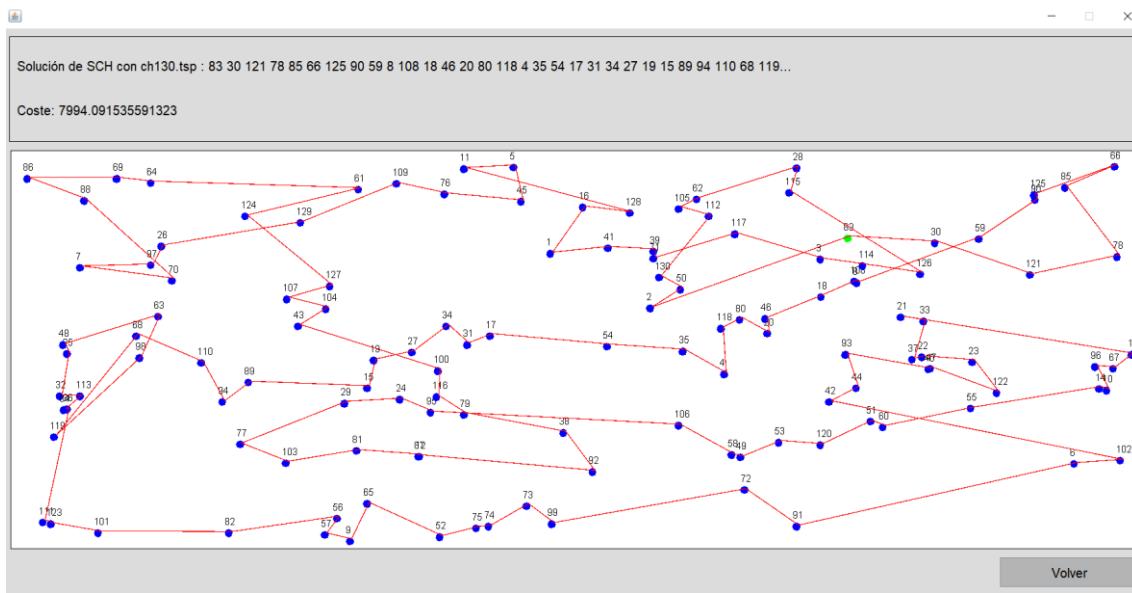


Figura 37. Resultado vista visualización. Implementación. Elaboración propia.

CAPÍTULO 6

EXPERIMENTACIÓN

En este capítulo, se realizará la prueba del software utilizando conjuntos de datos de diferentes tamaños con el objetivo de optimizar los valores de los parámetros de los algoritmos implementados y determinar cuál de ellos funciona mejor para el problema tratado. Los conjuntos se van a clasificar en función del número de ciudades según el siguiente criterio:

- Pequeño: < 500 .
- Mediano: ≥ 500 y ≤ 2000 .
- Grande: ≥ 2000 .

Para la realización de las pruebas, se han seleccionado tres conjuntos de datos de cada grupo. De tamaño pequeño, se ha seleccionado un conjunto de datos de Qatar con 194 ciudades, otro de Sahara Occidental con 29 ciudades y otro de Yibuti con 38 ciudades. De tamaño mediano, se ha seleccionado un conjunto de datos de Uruguay con 734 ciudades, otro de Zimbabue con 929 ciudades y otro de Omán con 1979 ciudades. De tamaño grande, se ha seleccionado un conjunto de datos de Egipto con 7146 ciudades, otro de Irlanda con 8246 ciudades y otro de Marruecos con 14185 ciudades. Todos estos conjuntos de datos se han descargado de la página web que se indica a continuación:

<https://www.math.uwaterloo.ca/tsp/world/countries.html>

Se van a realizar tres experimentos por algoritmo y dataset puesto que los algoritmos implementados son estocásticos, esto es, algoritmos que utilizan elementos de aleatoriedad o probabilidad en su ejecución.

Para cada algoritmo, comenzaremos observando los resultados que se obtienen con los parámetros por defecto y se estudiarán posibles modificaciones en los valores de estos para obtener mejores resultados. Conforme se vayan sacando conclusiones sobre el mejor valor para cada parámetro, iremos incorporando dicho

valor a las pruebas siguientes. Una vez optimizados los parámetros de cada algoritmo y obtenidos los mejores resultados posibles, realizaremos una comparación de los algoritmos implementados para determinar cual ofrece el mejor funcionamiento.

Es importante señalar que, para la optimización del valor de cada parámetro, es fundamental el uso de semillas, para que los valores aleatorios obtenidos durante la ejecución del algoritmo sean siempre los mismos en ejecuciones distintas de cada algoritmo.

Los parámetros y sus valores iniciales utilizados para cada algoritmo son los que se muestran en la [Tabla 6](#).

Tabla 6. Parámetros iniciales. Experimentos. Elaboración propia.

| Algoritmo/Parámetro | SCH | SHMM | SHMP | Lin-Kerninghan |
|---|-------|-------|-------|----------------|
| Iteraciones | 10000 | 10000 | 10000 | 10000 |
| Tiempo de ejecución (s) | 600 | 600 | 600 | 600 |
| Tamaño de la población | 10 | 10 | 10 | ✗ |
| α | 1 | 1 | 1 | ✗ |
| β | 2 | 2 | 2 | ✗ |
| q_0 | 0.95 | 0.95 | 0.95 | ✗ |
| Actualización global de feromona | 0.1 | 0.1 | 0.1 | ✗ |
| Actualización local de feromona | 0.1 | ✗ | ✗ | ✗ |
| Feromona inicial | 100 | 100 | 100 | ✗ |
| Porcentaje de reducción de iteraciones | 0.005 | 0.005 | ✗ | ✗ |
| Porcentaje de iteraciones inicial para usar mejor global | 0.08 | 0.08 | ✗ | ✗ |

| Algoritmo/Parámetro | SCH | SHMM | SHMP | Lin-Kerninghan |
|---|--------------|--------------|------|----------------|
| Hormiga que aporta feromona | Mejor global | Mejor global | ✗ | ✗ |
| Pbest | ✗ | 0.05 | ✗ | ✗ |
| Porcentaje de iteraciones para considerar estancamiento | ✗ | 0.05 | 0.05 | ✗ |
| Número de iteraciones para comprobar la convergencia | ✗ | 100 | ✗ | ✗ |
| Factor de convergencia | ✗ | 0.05 | ✗ | ✗ |
| Coeficiente de suavizado de feromona | ✗ | 0.5 | ✗ | ✗ |
| Reinicialización | ✗ | No | No | ✗ |
| Suavizado de feromona | ✗ | No | ✗ | ✗ |

El símbolo ✗ indica el algoritmo correspondiente no hace uso del parámetro indicado.

En los posteriores apartados, comenzará la realización de los distintos experimentos sobre los algoritmos mencionados haciendo uso de los parámetros especificados en la [Tabla 6](#). Todos los resultados que se extraigan de los experimentos se van a presentar en tablas, las cuales pueden encontrarse con mayor detalle en el [Anexo 1](#).

6.1. Sistema de Colonias de Hormigas (SCH)

Los resultados iniciales obtenidos para el algoritmo de Sistema de Colonias de Hormigas con los datasets mencionados anteriormente son los que se muestran en la [Tabla 7](#).

Tabla 7. Resultados iniciales ejecución SCH. Experimentos. Elaboración propia.

| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) |
|-------------|---------------|--------------------------|-------------------------------|
| wi29.tsp | 27600 | 27601,17 ± 0,00 | 19,67 ± 2,08 |
| dj38.tsp | 6656 | 6659,43 ± 0,00 | 28,00 ± 2,65 |
| qa194.tsp | 9352 | 12032,36 ± 37,18 | 317,00 ± 12,12 |
| uy734.tsp | 79114 | 129921,63 ± 1299,24 | 600,00 ± 0,00 |
| zi929.tsp | 95345 | 156929,61 ± 801,84 | 600,33 ± 0,58 |
| mu1979.tsp | 86891 | 452286,40 ± 19812,43 | 601,00 ± 1,00 |
| eg7146.tsp | 172387 | 1440459,32 ± 44898,15 | 613,67 ± 0,58 |
| ei8246.tsp | 206171 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 |
| mo14185.tsp | 427377 | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 |

En primer lugar, se va a analizar qué hormiga influye más en la calidad de las soluciones al aportar feromona durante la actualización global. Es decir, vamos a comparar las soluciones obtenidas considerando que la hormiga que aporta feromona en la actualización offline es la mejor encontrada hasta el momento, la mejor encontrada en la iteración o un enfoque híbrido. Los resultados obtenidos al realizar esta prueba son los que se indican en las tablas [8](#) y [9](#).

Tabla 8. Pruebas hormiga aportante SCH. Experimentos. Elaboración propia.

| Mejor global | | | Mejor iteración | | | Ambas | |
|--------------|---------------|------------------------------|-------------------------------|------------------------------|-------------------------------|------------------------------|-------------------------------|
| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| wi29.tsp | 27600 | 27601,17 ± 0,00 | 19,67 ± 2,08 | 27781,68 ± 126,76 | 4,67 ± 0,58 | 27601,17 ± 0,00 | 7,00 ± 0,00 |
| dj38.tsp | 6656 | 6659,43 ± 0,00 | 28,00 ± 2,65 | 6682,42 ± 32,18 | 6,67 ± 0,58 | 6659,43 ± 0,00 | 10,00 ± 0,00 |
| qa194.tsp | 9352 | 12032,36 ± 37,18 | 317,00 ± 12,12 | 12379,77 ± 122,47 | 134,67 ± 0,58 | 12092,85 ± 161,45 | 182,67 ± 0,58 |
| uy734.tsp | 79114 | 129921,63 ± 1299,24 | 600,00 ± 0,00 | 167138,61 ± 48852,80 | 604,33 ± 3,79 | 390701,36 ± 13196,63 | 604,00 ± 2,00 |
| zi929.tsp | 95345 | 156929,61 ± 801,84 | 600,33 ± 0,58 | 417808,64 ± 9738,73 | 613,33 ± 16,44 | 417808,64 ± 9738,73 | 606,00 ± 3,46 |
| mu1979.tsp | 86891 | 452286,40 ± 19812,43 | 601,00 ± 1,00 | 452286,40 ± 19812,43 | 623,33 ± 26,86 | 452286,40 ± 19812,43 | 642,33 ± 17,95 |
| eg7146.tsp | 172387 | 1440459,32 ± 44898,15 | 613,67 ± 0,58 | 1440459,32 ± 44898,15 | 706,33 ± 155,12 | 1440459,32 ± 44898,15 | 967,67 ± 212,68 |
| ei8246.tsp | 206171 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 | 2522195,43 ± 43151,61 | 1043,00 ± 429,99 | 2522195,43 ± 43151,61 | 959,33 ± 193,33 |
| mo14185.tsp | 427377 | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 | 5864485,55 ± 32839,03 | 1247,67 ± 386,93 | 5864485,55 ± 32839,03 | 1109,67 ± 449,10 |

Tabla 9. Resumen pruebas hormiga aportante SCH. Experimentos. Elaboración propia.

| Hormiga aportante | MEDIA | |
|---------------------|-------------------|------------|
| | Coste (km) | Tiempo (s) |
| Mejor global | 1179174,55 | 563,30 |
| Mejor iteración | 1212357,54 | 553,78 |
| Ambas | 1237143,35 | 565,41 |

Después de analizar los resultados las tablas [8](#) y [9](#), todo parece indicar que el algoritmo encuentra soluciones de mayor calidad cuando se utiliza la mejor hormiga encontrada hasta el momento en la actualización global. Normalmente, según afirman T. Stützle y H.H. Hoos, debería funcionar mejor el enfoque híbrido, el cual consiste en utilizar la mejor hormiga de la iteración para la actualización global y conforme avance la búsqueda, ir utilizando con más frecuencia la mejor hormiga encontrada hasta el momento. Esto favorece la exploración al principio y la explotación al final. Sin embargo, en este caso esta configuración no presenta el mejor funcionamiento debido a que el número de iteraciones y el tiempo de ejecución son pequeños, así como que el aporte de la mejor hormiga encontrada hasta el momento no influye demasiado. Si quisieramos que tuviera una gran relevancia deberíamos incrementar la frecuencia con la que aporta la mejor hormiga encontrada hasta el momento y aumentar considerablemente las iteraciones y tiempo de ejecución, algo que vamos a descartar por cuestiones prácticas.

Acto seguido, se va a analizar el tamaño de la población. Se probará una población de 5, 10, y 20 hormigas, y los resultados se presentarán en las tablas [10](#) y [11](#).

Tabla 10. Pruebas tamaño población SCH. Experimentos. Elaboración propia.

| Población de 5 hormigas | | | Población de 10 hormigas | | | Población de 20 hormigas | | |
|-------------------------|---------------|------------------------------|-------------------------------|------------------------------|-------------------------------|--------------------------|-------------------------------|------------------|
| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) |
| wi29.tsp | 27600 | 27650,35 ± 85,18 | 25,67 ± 22,50 | 27601,17 ± 0,00 | 19,67 ± 2,08 | 27601,17 ± 0,00 | 200,67 ± 109,50 | |
| dj38.tsp | 6656 | 6659,43 ± 0,00 | 44,33 ± 44,43 | 6659,43 ± 0,00 | 28,00 ± 2,65 | 6659,43 ± 0,00 | 272,67 ± 88,04 | |
| qa194.tsp | 9352 | 11540,91 ± 115,12 | 600,00 ± 0,00 | 12032,36 ± 37,18 | 317,00 ± 12,12 | 12738,31 ± 207,85 | 601,67 ± 2,89 | |
| uy734.tsp | 79114 | 128996,04 ± 2394,60 | 601,00 ± 1,00 | 129921,63 ± 1299,24 | 600,00 ± 0,00 | 397655,14 ± 2882,83 | 611,33 ± 6,66 | |
| zi929.tsp | 95345 | 155804,42 ± 1948,48 | 601,33 ± 2,31 | 156929,61 ± 801,84 | 600,33 ± 0,58 | 426547,61 ± 5050,65 | 615,33 ± 3,21 | |
| mu1979.tsp | 86891 | 463664,09 ± 4801,28 | 607,33 ± 5,86 | 452286,40 ± 19812,43 | 601,00 ± 1,00 | 472129,13 ± 9725,23 | 663,33 ± 34,24 | |
| eg7146.tsp | 172387 | 1450909,78 ± 16617,67 | 698,67 ± 52,79 | 1440459,32 ± 44898,15 | 613,67 ± 0,58 | 1478473,74 ± 19730,96 | 1400,67 ± 379,47 | |
| ei8246.tsp | 206171 | 2514470,50 ± 29009,49 | 709,33 ± 88,67 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 | 2553091,02 ± 30472,72 | 1724,67 ± 376,99 | |
| mo14185.tsp | 427377 | 5881658,31 ± 977,67 ± 303,55 | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 | 6023050,90 ± 57271,55 | 3013,67 ± 499,61 | | |

Tabla 11. Resumen pruebas tamaño población SCH. Experimentos. Elaboración propia.

| Tamaño de población | MEDIA | |
|---------------------------------|-------------------|------------|
| | Coste (km) | Tiempo (s) |
| Población de 5 hormigas | 1182372,65 | 540,59 |
| Población de 10 hormigas | 1179174,55 | 563,30 |
| Población de 20 hormigas | 1266438,49 | 1011,56 |

Tras observar los resultados expuestos en las tablas [10](#) y [11](#), todo parece indicar que cuanto mayor es el número de hormigas, más tiempo necesita el algoritmo para realizar una iteración, por lo que se completan menos iteraciones y se obtienen soluciones de peor calidad. Por lo tanto, no se va a continuar realizando pruebas con tamaños de población mayores. Por otro lado, a menor tamaño de la población, se realizan más iteraciones, pero con 5 hormigas no se obtienen las soluciones de mayor calidad, como si es el caso en un tamaño de población de 10 hormigas.

En cuanto al resto de parámetros (α , β , q_0 , actualización local, actualización global y feromona inicial), se mantienen en los valores por defecto, ya que se ha demostrado que son los que mejor funcionan en las investigaciones realizadas por Marco Dorigo.

Por último, a primera vista se observa que, en todas las pruebas, para los conjuntos de datos con pocas ciudades no se alcanza el tiempo total de ejecución. Esto se debe a que se realizan las iteraciones indicadas antes de agotar el tiempo especificado. Por otra parte, en los conjuntos de datos de tamaño medio y grande, ocurre lo contrario: se cumple el tiempo de ejecución especificado, pero no se alcanza el total de iteraciones debido a que, al haber más ciudades, el tiempo empleado para realizar una iteración es mucho mayor. Por tanto, se va a incrementar tanto el tiempo de ejecución como el número de iteraciones, sin excedernos para que el tiempo de ejecución no sea demasiado elevado. Estos resultados se exhibirán en las tablas [12](#) y [13](#).

Tabla 12. Pruebas iteraciones y tiempo ejecución SCH. Experimentos. Elaboración propia.

| 10000 iteraciones y 600 segundos de ejecución | | | 50000 iteraciones y 3600 segundos de ejecución | | |
|---|---------------|------------------------------|--|------------------------------|-------------------------------|
| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| wi29.tsp | 27600 | 27601,17 ± 0,00 | 19,67 ± 2,08 | 27601,17 ± 0,00 | 238,00 ± 91,16 |
| dj38.tsp | 6656 | 6659,43 ± 0,00 | 28,00 ± 2,65 | 6659,43 ± 0,00 | 445,00 ± 92,70 |
| qa194.tsp | 9352 | 12032,36 ± 37,18 | 317,00 ± 12,12 | 11924,34 ± 120,70 | 3600,33 ± 0,58 |
| uy734.tsp | 79114 | 129921,63 ± 1299,24 | 600,00 ± 0,00 | 126595,35 ± 855,29 | 3611,67 ± 11,50 |
| zi929.tsp | 95345 | 156929,61 ± 801,84 | 600,33 ± 0,58 | 153337,87 ± 1162,26 | 3616,00 ± 16,70 |
| mu1979.tsp | 86891 | 452286,40 ± 19812,43 | 601,00 ± 1,00 | 452286,40 ± 19812,43 | 3618,67 ± 18,50 |
| eg7146.tsp | 172387 | 1440459,32 ± 44898,15 | 613,67 ± 0,58 | 1440459,32 ± 44898,15 | 3912,00 ± 248,19 |
| ei8246.tsp | 206171 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 | 2522195,43 ± 43151,61 | 3924,67 ± 229,69 |
| m014185.tsp | 427377 | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 | 5864485,55 ± 32839,03 | 3979,67 ± 22,30 |

Tabla 13. Resumen pruebas iteraciones y tiempo ejecución SCH. Experimentos. Elaboración propia.

| Número de iteraciones y tiempo de ejecución | MEDIA | |
|---|-------------------|------------|
| | Coste (km) | Tiempo (s) |
| 10000 iteraciones y 600 s | 1179174,55 | 563,30 |
| 50000 iteraciones y 3600 s | 1178393,87 | 2994,00 |

A la luz de la observación manifestada en las tablas [12](#) y [13](#), tras incrementar el número de iteraciones y el tiempo de ejecución del algoritmo, se obtienen resultados ligeramente mejores, concretamente de unos 780 kilómetros menos de media. Sin embargo, el tiempo de ejecución es mucho mayor, haciendo que no merezca la pena el incremento en el tiempo de ejecución e iteraciones realizado. Esta escasa mejora se refleja principalmente en los conjuntos de datos de tamaño grande puesto que, aunque se realizan más iteraciones, siguen sin realizarse las iteraciones suficientes. Si se quisiera mejorar considerablemente la calidad de las soluciones habría que ejecutar el algoritmo mucho más tiempo para que la colonia de hormigas tenga la oportunidad de explorar más caminos nuevos, algo que para este trabajo sería impracticable.

En conclusión, tras las experimentaciones realizadas, se van a utilizar los valores obtenidos con los parámetros por defecto para comparar este algoritmo con los demás.

6.2. Sistema de Hormigas Max-Min (SHMM)

Los resultados iniciales obtenidos para el algoritmo de Sistema de Hormigas Max-Min con los datasets mencionados anteriormente son los que se muestran en la [Tabla 14](#).

Tabla 14. Resultados iniciales ejecución SHMM. Experimentos. Elaboración propia.

| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) |
|-------------|---------------|---------------------------|-------------------------------|
| wi29.tsp | 27600 | $29115,05 \pm 479,69$ | $35,33 \pm 3,21$ |
| dj38.tsp | 6656 | $6869,60 \pm 157,82$ | $50,67 \pm 5,77$ |
| qa194.tsp | 9352 | $11992,47 \pm 625,49$ | $503,33 \pm 10,69$ |
| uy734.tsp | 79114 | $116042,17 \pm 1442,85$ | $600,33 \pm 0,58$ |
| zi929.tsp | 95345 | $141275,98 \pm 3629,33$ | $600,33 \pm 0,58$ |
| mu1979.tsp | 86891 | $172895,29 \pm 30818,94$ | $601,67 \pm 1,15$ |
| eg7146.tsp | 172387 | $1456142,94 \pm 3271,35$ | $674,67 \pm 0,58$ |
| ei8246.tsp | 206171 | $2591233,15 \pm 31469,57$ | $893,00 \pm 0,00$ |
| mo14185.tsp | 427377 | $6122171,70 \pm 69410,71$ | $1331,67 \pm 28,02$ |

Dado que en apartado anterior se ha observado que la configuración por defecto es la que mejores soluciones obtiene en relación al tiempo, solo se va a comprobar si se obtienen resultados mejores aplicando la reinicialización de la búsqueda y el suavizado de feromona.

La experimentación aplicando el mecanismo de reinicialización se vislumbra en las tablas [15](#) y [16](#).

Tabla 15. Pruebas con reinicialización SHMM. Experimentos. Elaboración propia.

| | | Sin reinicialización | Con reinicialización | | |
|-------------|---------------|------------------------------|-------------------------------|------------------------------|-------------------------------|
| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| wi29.tsp | 27600 | 29115,05 ± 479,69 | 35,33 ± 3,21 | 27650,35 ± 85,18 | 18,67 ± 8,14 |
| dj38.tsp | 6656 | 6869,60 ± 157,82 | 50,67 ± 5,77 | 6659,43 ± 0,00 | 33,33 ± 13,58 |
| qa194.tsp | 9352 | 11992,47 ± 625,49 | 503,33 ± 10,69 | 11241,59 ± 458,44 | 600,00 ± 0,00 |
| uy734.tsp | 79114 | 116042,17 ± 1442,85 | 600,33 ± 0,58 | 125845,06 ± 15588,15 | 603,00 ± 1,73 |
| zi929.tsp | 95345 | 141275,98 ± 3629,33 | 600,33 ± 0,58 | 159133,87 ± 17672,75 | 609,00 ± 13,89 |
| mu1979.tsp | 86891 | 172895,29 ± 30818,94 | 601,67 ± 1,15 | 384403,48 ± 87974,05 | 638,33 ± 38,21 |
| eg7146.tsp | 172387 | 1456142,94 ± 3271,35 | 674,67 ± 0,58 | 1493501,37 ± 26430,14 | 1093,67 ± 272,48 |
| ei8246.tsp | 206171 | 2591233,15 ± 31469,57 | 893,00 ± 0,00 | 2593730,41 ± 21792,58 | 1093,33 ± 319,99 |
| mo14185.tsp | 427377 | 6122171,70 ± 69410,71 | 1331,67 ± 28,02 | 6086143,02 ± 64347,03 | 1117,33 ± 439,50 |

Tabla 16. Resumen pruebas reinicialización SHMM. Experimentos. Elaboración propia.

| Mecanismo | MEDIA | |
|-----------------------------|-------------------|------------|
| | Coste (km) | Tiempo (s) |
| Sin reinicialización | 1183082,04 | 587,89 |
| Con reinicialización | 1209812,06 | 645,19 |

Como cabría esperar, las tablas [15](#) y [16](#) muestran de nuevo que el número de iteraciones y el tiempo de ejecución influyen en la efectividad de este parámetro. En los conjuntos datos de tamaño pequeño, al realizar más iteraciones, se aplica la reinicialización más veces, lo que lleva a obtener soluciones de mayor calidad. Sin embargo, para el resto de conjuntos de datos, la reinicialización prácticamente no se aplica, por lo que se obtienen soluciones peores. En términos generales, aplicar reinicialización es la mejor opción para obtener soluciones de calidad cuando se realizan ejecuciones muy largas en tiempo y muchas iteraciones de los algoritmos. No obstante, en nuestro caso, donde buscamos soluciones en un tiempo corto, optaremos por no aplicar la reinicialización, ya que se obtienen soluciones mejores en término medio.

A continuación, se procede a analizar si conviene aplicar suavizado de feromona o no en función de los resultados que se mostrarán en las tablas [17](#) y [18](#).

Tabla 17. Pruebas con suavizado de feromona SHMM. Experimentos. Elaboración propia.

| Sin suavizado de feromona | | | Con suavizado de feromona | | |
|---------------------------|---------------|------------------------------|-------------------------------|------------------------------|-------------------------------|
| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| wi29.tsp | 27600 | 29115,05 ± 479,69 | 35,33 ± 3,21 | 28491,04 ± 376,88 | 45,33 ± 26,27 |
| dj38.tsp | 6656 | 6869,60 ± 157,82 | 50,67 ± 5,77 | 6727,17 ± 81,36 | 97,00 ± 37,00 |
| qa194.tsp | 9352 | 11992,47 ± 625,49 | 503,33 ± 10,69 | 11826,49 ± 691,85 | 600,00 ± 0,00 |
| uy734.tsp | 79114 | 116042,17 ± 1442,85 | 600,33 ± 0,58 | 116066,47 ± 1446,34 | 601,33 ± 1,15 |
| zi929.tsp | 95345 | 141275,98 ± 3629,33 | 600,33 ± 0,58 | 144854,13 ± 5154,19 | 602,00 ± 3,46 |
| mu1979.tsp | 86891 | 172895,29 ± 30818,94 | 601,67 ± 1,15 | 277183,30 ± 9883,44 | 624,67 ± 23,71 |
| eg7146.tsp | 172387 | 1456142,94 ± 3271,35 | 674,67 ± 0,58 | 1493501,37 ± 26430,14 | 830,67 ± 236,88 |
| ei8246.tsp | 206171 | 2591233,15 ± 31469,57 | 893,00 ± 0,00 | 2593730,41 ± 21792,58 | 1096,00 ± 277,51 |
| m014185.tsp | 427377 | 6122171,70 ± 69410,71 | 1331,67 ± 28,02 | 6122171,70 ± 69410,71 | 1334,33 ± 354,99 |

Tabla 18. Resumen pruebas suavizado de feromona SHMM. Experimentos. Elaboración propia.

| Mecanismo | MEDIA | |
|----------------------------------|-------------------|------------|
| | Coste (km) | Tiempo (s) |
| Sin suavizado de feromona | 1183082,04 | 587,89 |
| Con suavizado de feromona | 1199394,67 | 647,93 |

El suavizado de feromona se aplica cuando se detecta la convergencia del algoritmo, y esta se verifica cada n iteraciones, siendo n un porcentaje de iteraciones con respecto al total. Dado que en los conjuntos de datos de tamaño mediano y grande se realizan muy pocas iteraciones, el suavizado de feromona prácticamente no se aplica. Sin embargo, en los conjuntos de datos de tamaño pequeño, donde se alcanza el total de iteraciones indicado, sí se aplica el suavizado de feromona, lo que se traduce en soluciones de mayor calidad con respecto a la ejecución sin suavizado de feromona. Por tanto, los resultados revelados en las tablas [17](#) y [18](#) demuestran que es preferible utilizar el suavizado de feromona para obtener soluciones de mayor calidad en ejecuciones prolongadas. No obstante, en nuestro caso, donde se realizan pocas iteraciones y el tiempo de ejecución no es extenso, optaremos por no aplicar este elemento, ya que se obtienen mejores resultados en término medio.

En cuanto al parámetro $pbest$, vamos a dejar su valor por defecto, ya que T. Stützle y H.H. Hoos han demostrado en sus investigaciones que es el valor que funciona mejor.

En resumen, se considera que este algoritmo produce los mejores resultados con los valores predeterminados de sus parámetros.

6.3. Sistema de Hormigas del Mejor-Peor (SHMP)

Los resultados iniciales obtenidos para el algoritmo de Sistema de Hormigas del Mejor-Peor con los datasets mencionados anteriormente son los que se muestran en la [Tabla 19](#).

Tabla 19. Resultados iniciales ejecución SHMP. Experimentos. Elaboración propia.

| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) |
|-------------|---------------|---------------------------|-------------------------------|
| wi29.tsp | 27600 | $27601,17 \pm 0,00$ | $21,67 \pm 3,21$ |
| dj38.tsp | 6656 | $6659,43 \pm 0,00$ | $30,67 \pm 2,31$ |
| qa194.tsp | 9352 | $16852,46 \pm 325,65$ | $265,67 \pm 15,01$ |
| uy734.tsp | 79114 | $204620,70 \pm 7802,01$ | $600,00 \pm 0,00$ |
| zi929.tsp | 95345 | $233098,25 \pm 9171,69$ | $600,00 \pm 0,00$ |
| mu1979.tsp | 86891 | $462804,00 \pm 2846,78$ | $601,33 \pm 1,15$ |
| eg7146.tsp | 172387 | $1476520,22 \pm 325,65$ | $646,67 \pm 0,58$ |
| ei8246.tsp | 206171 | $2590304,68 \pm 20392,09$ | $892,67 \pm 2,08$ |
| mo14185.tsp | 427377 | $6122171,70 \pm 69410,71$ | $1026,00 \pm 220,76$ |

Dado que este algoritmo comparte parámetros con los dos algoritmos anteriores y hemos demostrado que los valores por defecto son los que obtienen mejores soluciones en tiempos de ejecución relativamente pequeños, optaremos por utilizar esos valores sin realizar pruebas adicionales.

6.4. Comparación entre algoritmos constructivos

En este apartado se va a realizar un estudio comparativo de los algoritmos constructivos implementados para resolver el problema del viajante de comercio (TSP). En las tablas [20](#), [21](#), [22](#), [23](#) y [24](#) podemos ver un resumen de los mejores resultados obtenidos con estos algoritmos.

Tabla 20. Comparación algoritmos constructivos. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|------------------------|-------------------------------|-----------------------|-------------------------------|
| | wi29.tsp | | wi29.tsp | |
| | Óptimo global | 27600 | Óptimo global | 6656 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | 27601,17 ± 0,00 | 19,67 ± 2,08 | 6659,43 ± 0,00 | 28,00 ± 2,65 |
| SHMM | 29115,05 ± 479,69 | 35,33 ± 3,21 | 6869,60 ± 157,82 | 50,67 ± 5,77 |
| SHMP | 27601,17 ± 0,00 | 21,67 ± 3,21 | 6659,43 ± 0,00 | 30,67 ± 2,31 |

Tabla 21. Comparación algoritmos constructivos. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|---|-------------------------------------|---|-------------------------------------|
| | qa194.tsp | | uy734.tsp | |
| | Óptimo global | 9352 | Óptimo global | 79114 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | $12032,36 \pm 37,18$ | $317,00 \pm 12,12$ | $129921,63 \pm 1299,24$ | $600,00 \pm 0,00$ |
| SHMM | $11992,47 \pm 625,49$ | $503,33 \pm 10,69$ | $116042,17 \pm 1442,85$ | $600,33 \pm 0,58$ |
| SHMP | $16852,46 \pm 325,65$ | $265,67 \pm 15,01$ | $204620,70 \pm 7802,01$ | $600,00 \pm 0,00$ |

Tabla 22. Comparación algoritmos constructivos. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|---|-------------------------------------|--|-------------------------------------|
| | zi929.tsp | | mu1979.tsp | |
| | Óptimo global | 95345 | Óptimo global | 86891 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | $156929,61 \pm 801,84$ | $600,33 \pm 0,58$ | $452286,40 \pm 19812,43$ | $601,00 \pm 1,00$ |
| SHMM | $141275,98 \pm 3629,33$ | $600,33 \pm 0,58$ | $172895,29 \pm 30818,94$ | $601,67 \pm 1,15$ |
| SHMP | $233098,25 \pm 9171,69$ | $600,00 \pm 0,00$ | $462804,00 \pm 2846,78$ | $601,33 \pm 1,15$ |

Tabla 23. Comparación algoritmos constructivos. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|----------------------------------|-------------------------------------|----------------------------------|-------------------------------------|
| | eg7146.tsp | | ei8246.tsp | |
| | Óptimo global | 172387 | Óptimo global | 206171 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | 1440459,32 ± 44898,15 | 613,67 ± 0,58 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 |
| SHMM | 1456142,94 ± 3271,35 | 674,67 ± 0,58 | 2591233,15 ± 31469,57 | 893,00 ± 0,00 |
| SHMP | 1476520,22 ± 11691,16 | 646,67 ± 0,58 | 2590304,68 ± 20392,09 | 892,67 ± 2,08 |

Tabla 24. Comparación algoritmos constructivos. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | MEDIA | | | |
|-----------|----------------------------------|-------------------------------------|-------------------|----------------------------|--|--|
| | mo14185.tsp | | | | | |
| | Óptimo global | 427377 | | | | |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste (km) | Tiempo de ejecución (s) | | |
| SCH | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 | 1179174,55 | 563,30 | | |
| SHMM | 6122171,70 ± 69410,71 | 1331,67 ± 28,02 | 1183082,04 | 587,89 | | |
| SHMP | 6122171,70 ± 69410,71 | 1026,00 ± 220,76 | 1237848,07 | 520,52 | | |

Como se evidencia en las tablas [20](#), [21](#), [22](#), [23](#) y [24](#), el algoritmo que mejor funciona en términos medios con los conjuntos de datos proporcionados es el Sistema de Colonias de Hormigas, ya que presenta un menor costo promedio en las soluciones encontradas. En cuanto a tiempos de ejecución, los tres algoritmos tienen tiempos de ejecución muy similares en media, por lo que podríamos decantarnos por cualquiera en base a este factor.

Además, cabe destacar que los tres algoritmos encuentran soluciones muy cercanas a la óptima. Sin embargo, a medida que aumenta el tamaño del conjunto de datos, el algoritmo logra realizar menos iteraciones en el tiempo indicado y, por tanto, la solución encontrada se aleja más del óptimo global. Por este motivo, en los conjuntos de datos de tamaño grande, observamos que el costo de las soluciones es muy elevado en comparación con el costo de la solución óptima. En consecuencia, esto podría resolverse incrementando el número de iteraciones y el tiempo de ejecución de los algoritmos a un gran número de horas, días, semanas o incluso meses. No obstante, al no disponer de tanto tiempo, se han realizado las pruebas con tiempos de ejecución de 10 minutos y 10000 iteraciones.

En conclusión, para la resolución de este problema el mejor algoritmo constructivo es el Sistema de Colonias de Hormigas.

Seguidamente se van a presentar una serie de gráficos para ilustrar las comparaciones realizadas de una forma más simple y visual.

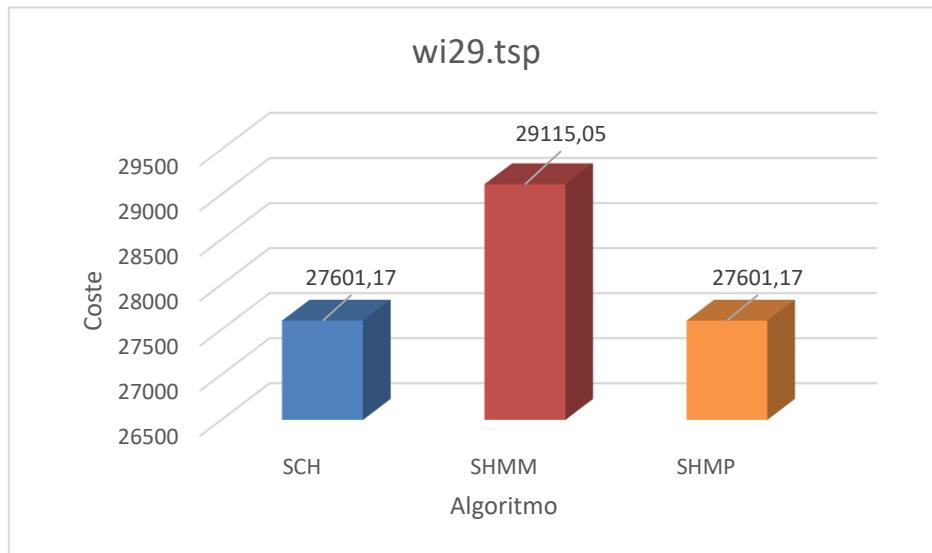


Figura 38. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 38](#) se observa que, para el conjunto de datos wi29.tsp, los algoritmos que obtienen las mejores soluciones son SCH y SHMP, con una diferencia de aproximadamente 15014 kilómetros respecto al SHMM.

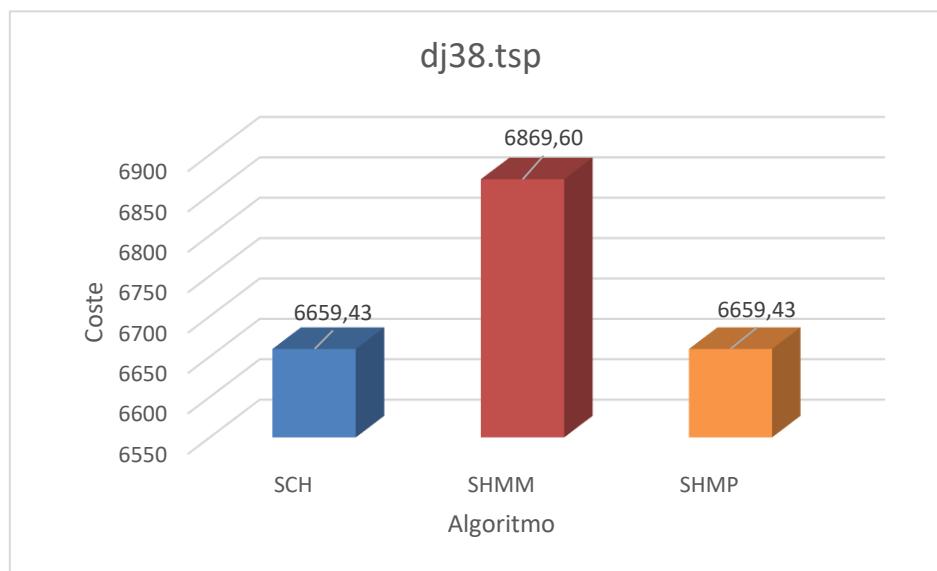


Figura 39. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 39](#) se nota que, para el conjunto de datos dj38.tsp, los algoritmos que obtienen las mejores soluciones son SCH y SHMP, con una diferencia de aproximadamente 210 kilómetros en relación con el SHMM.

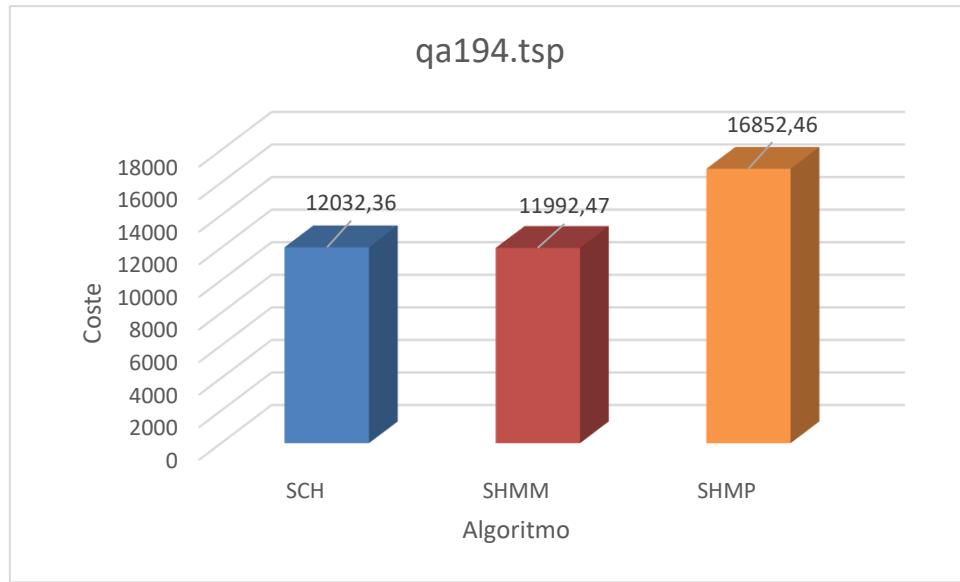


Figura 40. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 40](#) se percibe que, para el conjunto de datos qa194.tsp, el algoritmo que obtiene las mejores soluciones es el SHMM, con una diferencia de aproximadamente 40 kilómetros sobre el SCH y alrededor de 4860 kilómetros en cuanto al SHMP.

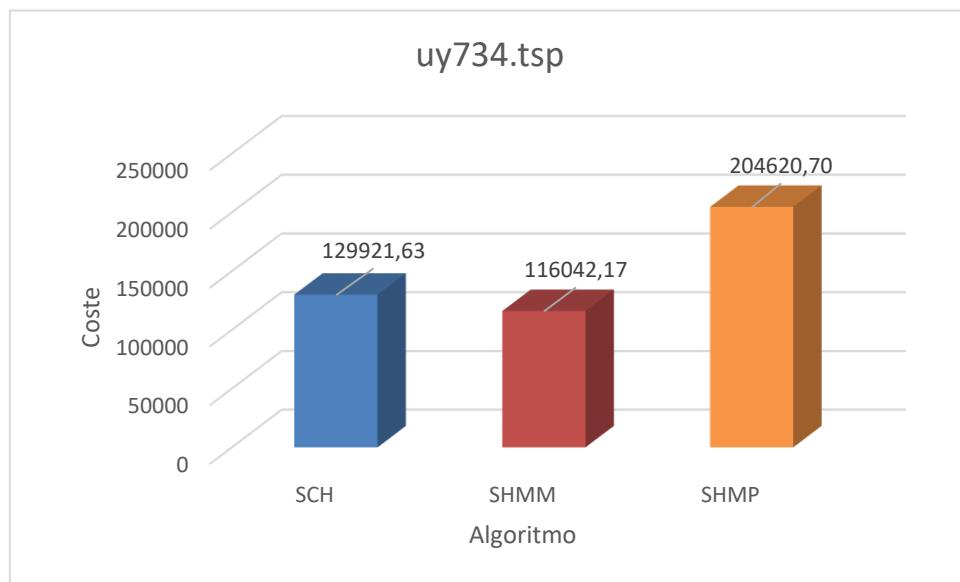


Figura 41. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 41](#) se advierte que, para el conjunto de datos uy734.tsp, el algoritmo que obtiene las mejores soluciones es SHMM, con una diferencia de aproximadamente 13879 kilómetros respecto al SCH y alrededor de 88578 kilómetros en referencia al SHMP.

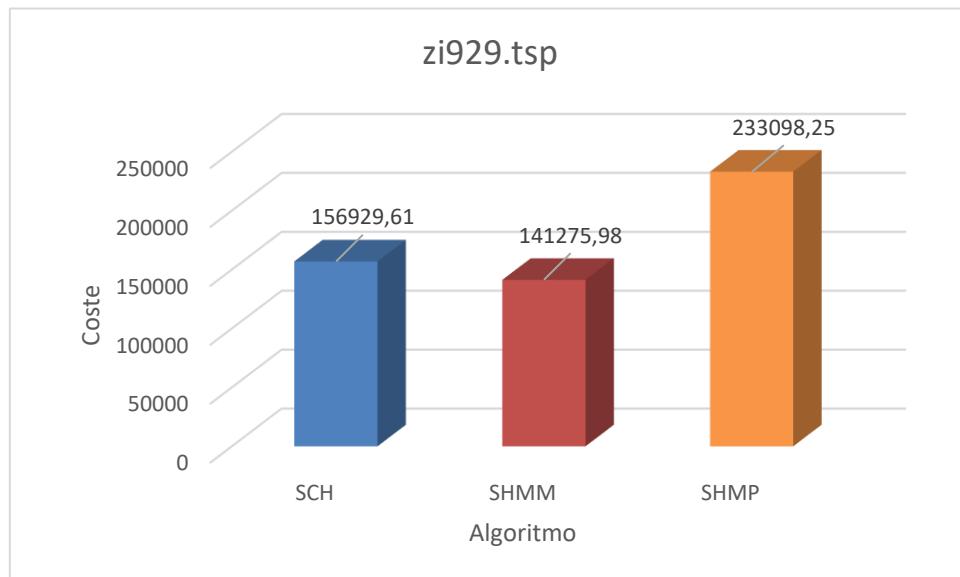


Figura 42. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 42](#) se aprecia que, para el conjunto de datos zi929.tsp, el algoritmo que obtiene las mejores soluciones es SHMM, con una diferencia de aproximadamente 15654 kilómetros respecto al SCH y alrededor de 91823 kilómetros en comparación con el SHMP.

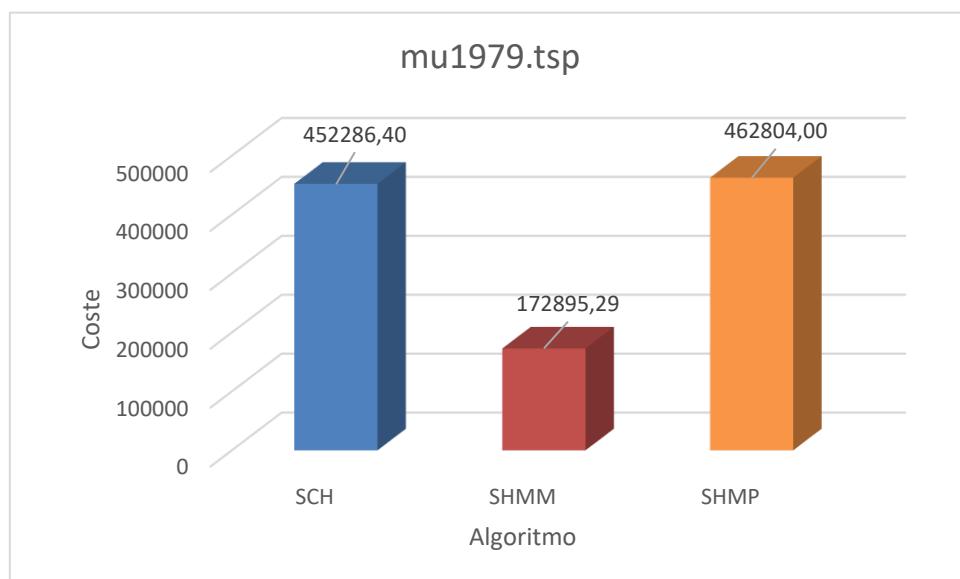


Figura 43. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 43](#) se registra que, para el conjunto de datos mu1979.tsp, el algoritmo que obtiene las mejores soluciones es SHMM, con una diferencia de aproximadamente 279391 kilómetros en relación con el SCH y alrededor de 289909 kilómetros sobre el SHMP.

Antes de continuar con la siguiente gráfica, cabe destacar que parece que el algoritmo SHMM funciona mejor que SCH y SHMP para conjuntos de datos de tamaño medio.

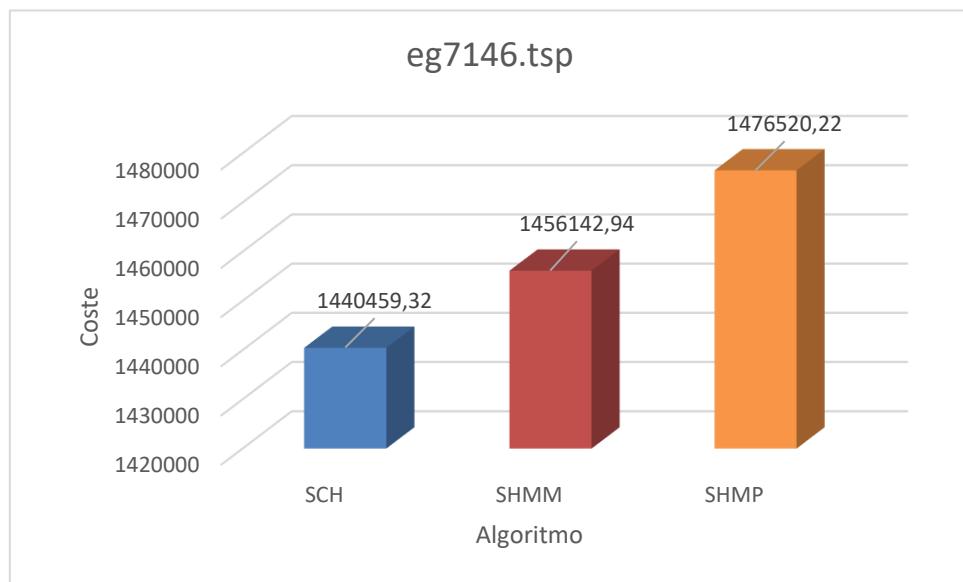


Figura 44. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 44](#) se ve que, para el conjunto de datos eg7146.tsp, el algoritmo que obtiene las mejores soluciones es SCH, con una diferencia de aproximadamente 15683 kilómetros respecto al SHMM y alrededor de 36061 kilómetros relativos al SHMP.

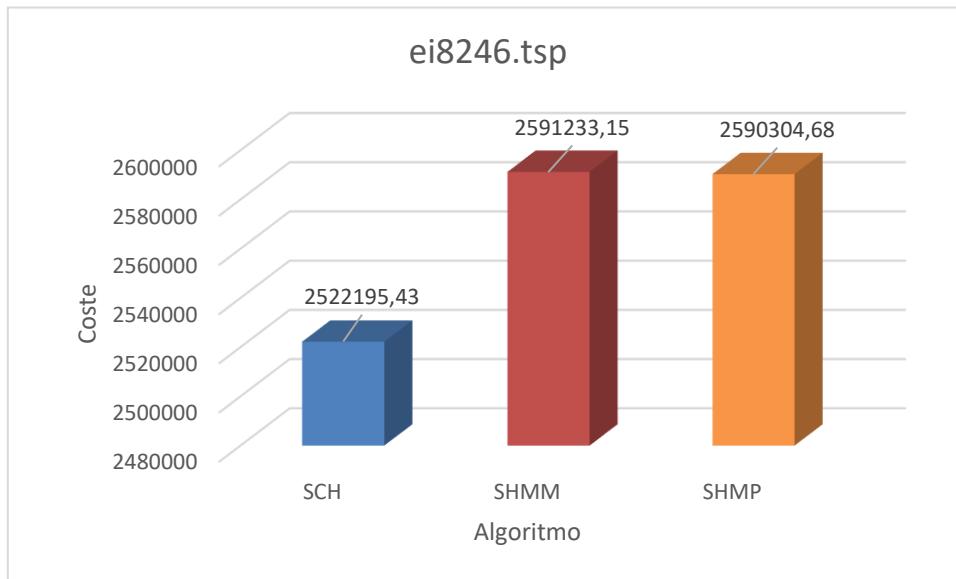


Figura 45. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 45](#) se constata que, para el conjunto de datos ei8246.tsp, el algoritmo que obtiene las mejores soluciones es SCH, con una diferencia de aproximadamente 69038 kilómetros en cuanto al SHMM y alrededor de 68109 kilómetros relativos al SHMP.

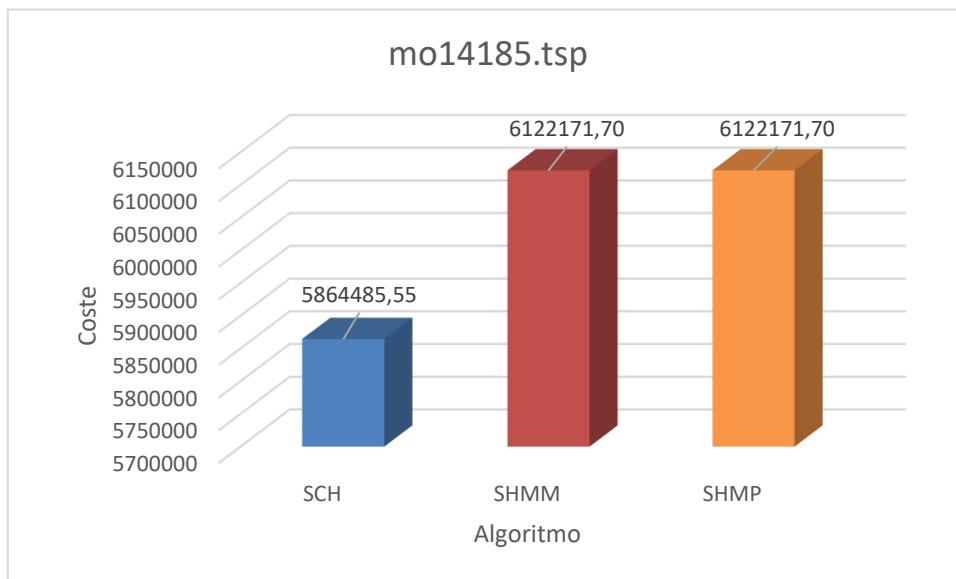


Figura 46. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 46](#) se evidencia que, para el conjunto de datos mo14185.tsp, el algoritmo que obtiene las mejores soluciones es SCH, con una diferencia de aproximadamente 257686 kilómetros en comparación tanto con SHMM como con SHMP.

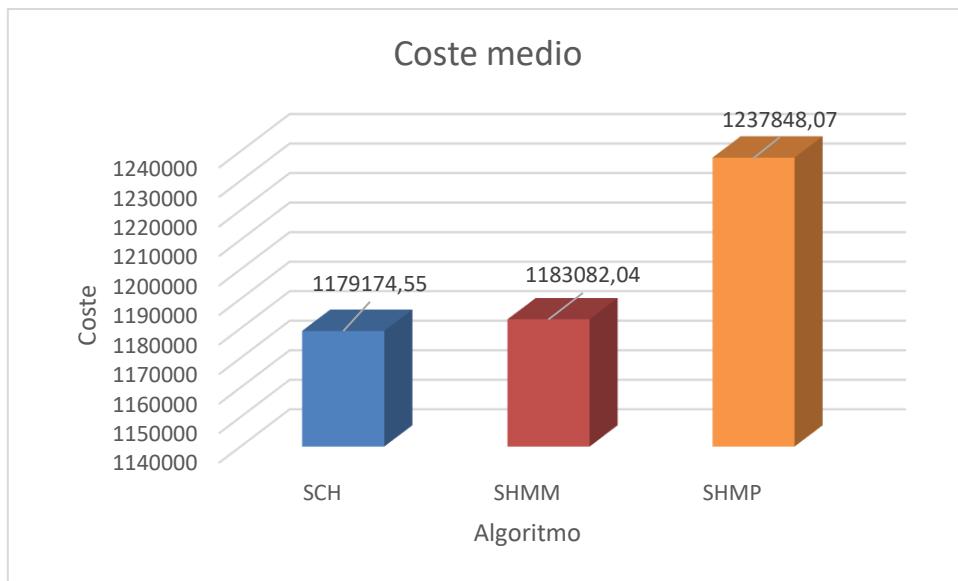


Figura 47. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 47](#) se confirma lo que se ha venido mencionando: SCH es el algoritmo constructivo que obtiene el mejor coste medio, con una diferencia de 3908 kilómetros respecto al SHMM y una diferencia de 58674 kilómetros en comparación con el SHMP.

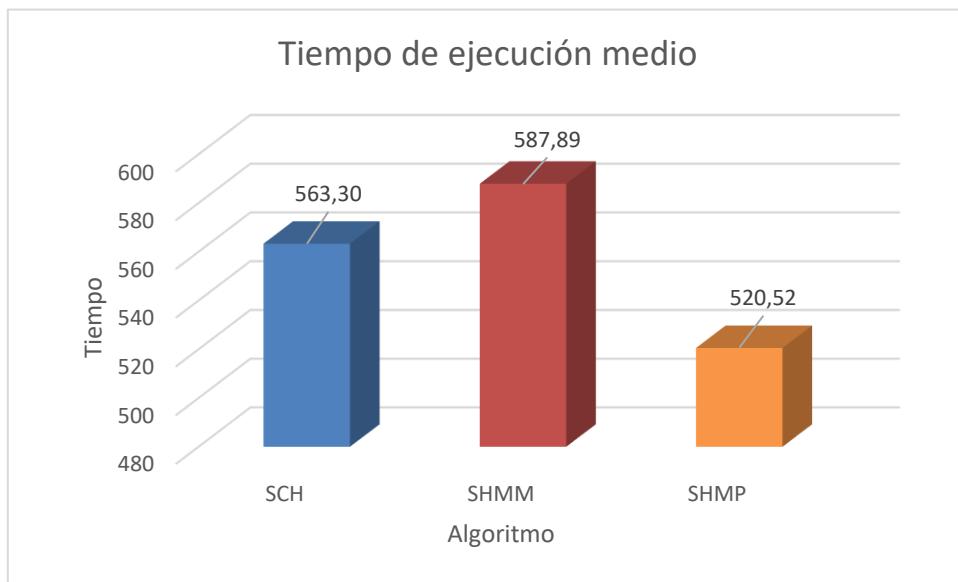


Figura 48. Diagrama comparando algoritmos constructivos. Experimentos. Elaboración propia.

En la [Figura 48](#) se muestra que el algoritmo que menos tarda en ejecutarse es SHMP, el cual, casualmente, es el que obtiene peores resultados, seguido por SCH y SHMM. Sin embargo, los tres algoritmos tienen tiempos de ejecución muy similares.

En definitiva, el mejor algoritmo constructivo es SCH, ya que obtiene soluciones de mayor calidad y presenta un tiempo de ejecución muy similar a los otros dos.

6.5. Algoritmo de Lin-Kerninghan

Los resultados iniciales obtenidos para el algoritmo de Lin-Kerninghan con los datasets mencionados al principio de este capítulo se presentan en la [Tabla 25](#).

Tabla 25. Resultados iniciales ejecución Lin-Kerninghan. Experimentos. Elaboración propia.

| Dataset | Óptimo global | Coste medio (km) | Tiempo de ejecución medio (s) |
|-------------|---------------|----------------------|-------------------------------|
| wi29.tsp | 27600 | 29343,51 ± 397,96 | 1,00 ± 0,00 |
| dj38.tsp | 6656 | 7139,41 ± 227,36 | 1,00 ± 0,00 |
| qa194.tsp | 9352 | 9691,12 ± 41,81 | 9,67 ± 2,31 |
| uy734.tsp | 79114 | 88016,74 ± 2946,95 | 442,33 ± 71,59 |
| zi929.tsp | 95345 | 105650,37 ± 2021,67 | 611,00 ± 1,00 |
| mu1979.tsp | 86891 | 262694,67 ± 29175,41 | 649,00 ± 1,00 |
| eg7146.tsp | 172387 | 454517,68 ± 9436,98 | 19231,00 ± 344,73 |
| ei8246.tsp | 206171 | 453941,04 ± 6709,09 | 28027,33 ± 98,29 |
| mo14185.tsp | 427377 | 1107241,04 ± 7745,48 | 84207,67 ± 468,49 |

Dado que este algoritmo solamente tiene dos parámetros, el número de iteraciones y el tiempo de ejecución, no se realizarán pruebas modificando los valores de estos parámetros para mantener igualdad con respecto a los otros algoritmos mencionados previamente.

6.6. Comparación entre SCH y Lin-Kernighan

En las tablas [26](#), [27](#), [28](#), [29](#) y [30](#), se realizará una comparación entre el mejor algoritmo constructivo y el algoritmo de Lin-Kernighan con el objetivo de determinar cuál de ellos es más efectivo en el problema tratado.

Tabla 26. Comparación SCH y Lin-Kernighan. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|------------------------|-------------------------------------|-----------------------|-------------------------------------|
| | wi29.tsp | | dj38.tsp | |
| | Óptimo global | 27600 | Óptimo global | 6656 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | 27601,17 ± 0,00 | 19,67 ± 2,08 | 6659,43 ± 0,00 | 28,00 ± 2,65 |
| LK | 29343,51 ± 397,96 | 1,00 ± 0,00 | 7139,41 ± 227,36 | 1,00 ± 0,00 |

Tabla 27. Comparación SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|---------------------------------------|-------------------------------------|--|-------------------------------------|
| | qa194.tsp | | uy734.tsp | |
| | Óptimo global | 9352 | Óptimo global | 79114 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | $12032,36 \pm 37,18$ | $317,00 \pm 12,12$ | $129921,63 \pm 1299,24$ | $600,00 \pm 0,00$ |
| LK | $9691,12 \pm 41,81$ | $9,67 \pm 2,31$ | $88016,74 \pm 2946,95$ | $442,33 \pm 71,59$ |

Tabla 28. Comparación SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|---|-------------------------------------|--|-------------------------------------|
| | zi929.tsp | | mu1979.tsp | |
| | Óptimo global | 95345 | Óptimo global | 86891 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | $156929,61 \pm 801,84$ | $600,33 \pm 0,58$ | $452286,40 \pm 19812,43$ | $601,00 \pm 1,00$ |
| LK | $105650,37 \pm 2021,67$ | $611,00 \pm 1,00$ | $262694,67 \pm 29175,41$ | $649,00 \pm 1,00$ |

Tabla 29. Comparación SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | | |
|-----------|----------------------------|-------------------------------|----------------------------|-------------------------------|
| | eg7146.tsp | | ei8246.tsp | |
| | Óptimo global | 172387 | Óptimo global | 206171 |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste medio (km) | Tiempo de ejecución medio (s) |
| SCH | 1440459,32 ± 44898,15 | 613,67 ± 0,58 | 2522195,43 ± 43151,61 | 900,67 ± 5,51 |
| LK | 454517,68 ± 9436,98 | 19231,00 ± 344,73 | 453941,04 ± 6709,09 | 28027,33 ± 98,29 |

Tabla 30. Comparación SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

| Algoritmo | Dataset | | MEDIA | |
|-----------|-----------------------------|-------------------------------|------------------|-------------------------|
| | mo14185.tsp | | | |
| | Óptimo global | 427377 | | |
| | Coste medio (km) | Tiempo de ejecución medio (s) | Coste (km) | Tiempo de ejecución (s) |
| SCH | 5864485,55 ± 32839,03 | 1389,33 ± 5,51 | 1179174,55 | 563,30 |
| LK | 1107241,04 ± 7745,48 | 84207,67 ± 468,49 | 279803,95 | 14797,78 |

Como podemos observar en las tablas [26](#), [27](#), [28](#), [29](#) y [30](#), parece que el algoritmo de Lin-Kernighan es el que mejores soluciones obtiene. Sin embargo, presenta un aspecto muy negativo: el tiempo de ejecución. Este algoritmo selecciona una solución inicial mediante un algoritmo greedy y realiza una búsqueda local a través del intercambio de nodos para mejorar gradualmente la solución inicial. Esto ocasiona que para instancias grandes del TSP tarde mucho tiempo en completar una iteración, por lo que resulta muy ineficiente, como demuestran los resultados obtenidos. A pesar de ello, para instancias pequeñas y medianas del TSP es muy eficiente y logra obtener soluciones bastante buenas, incluso mejores que el SCH.

Por tanto, la elección entre un algoritmo u otro dependerá del tamaño del TSP a resolver. Si se trata de problemas de tamaño pequeño o mediano, se podría optar por el algoritmo de Lin-Kernighan debido a que ofrece una mayor calidad en las soluciones y un tiempo de ejecución menor o igual al del SCH. Sin embargo, para problemas de tamaño grande, se recomienda el uso del SCH, aumentando el tiempo de ejecución y las iteraciones, para obtener soluciones mejores en menos tiempo en comparación con el algoritmo de Lin-Kernighan.

Antes de continuar, es importante destacar que el SCH tarda, en promedio, unos 23 minutos en realizar una iteración con el conjunto de datos mo14185.tsp, mientras que el algoritmo de Lin-Kernighan tarda aproximadamente 1403 minutos.

A continuación, se van a presentar estos resultados en gráficos de barras para que los algoritmos se puedan comparar de una forma más sencilla con un simple vistazo.

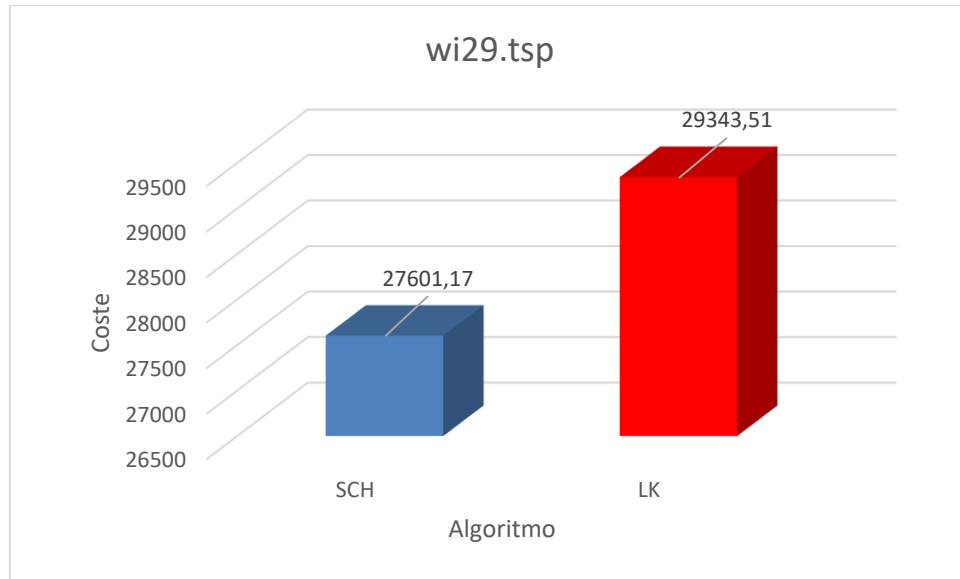


Figura 49. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Como podemos observar en la [Figura 49](#), el algoritmo SCH es el que obtiene las mejores soluciones para el conjunto de datos wi29.tsp, concretamente unos 1742 kilómetros menos en comparación con el algoritmo de Lin-Kerninghan.

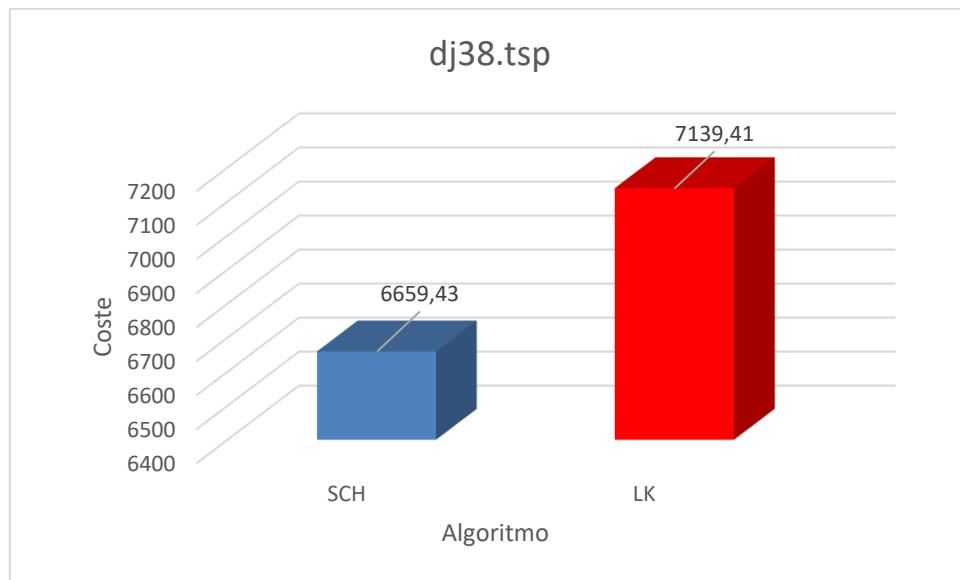


Figura 50. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se puede notar en la [Figura 50](#), el algoritmo SCH es el que obtiene las mejores soluciones para el conjunto de datos dj38.tsp, concretamente unos 480 kilómetros menos en relación con el algoritmo de Lin-Kerninghan.

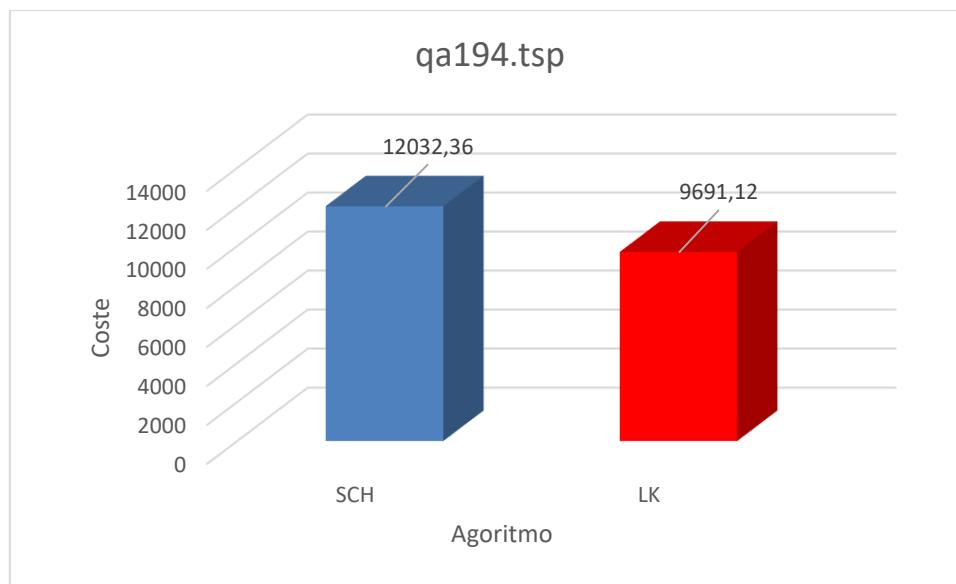


Figura 51. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Tal como se puede apreciar en la [Figura 51](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos qa194.tsp, concretamente unos 2341 kilómetros menos que el algoritmo SCH. Cabe destacar que parece que, conforme aumenta el tamaño del conjunto de datos, el rendimiento del algoritmo de Lin-Kerninghan mejora en comparación con el SCH.

En conclusión, para los conjuntos de tamaño pequeño, ambos algoritmos obtienen soluciones bastante similares en coste.

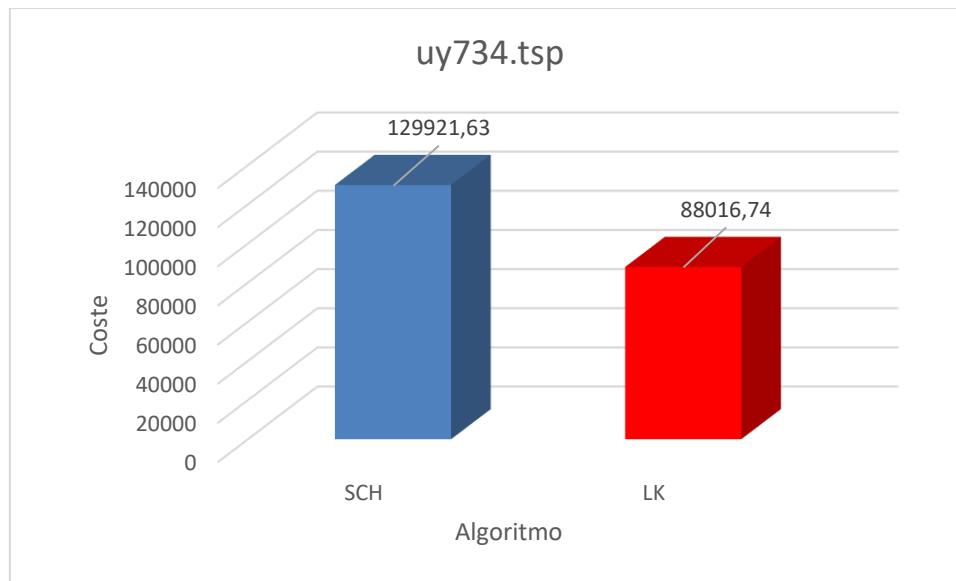


Figura 52. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

De acuerdo con lo que vemos en la [Figura 52](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos uy734.tsp, concretamente unos 41905 kilómetros menos respecto al algoritmo SCH.

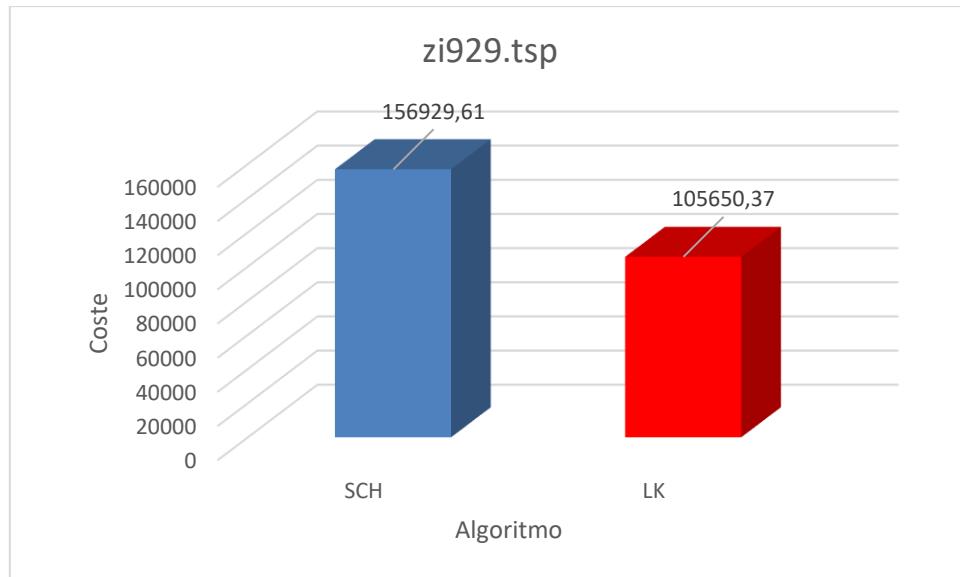


Figura 53. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se evidencia en la [Figura 53](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos zi929.tsp, concretamente unos 51279 kilómetros menos en relación al algoritmo SCh.

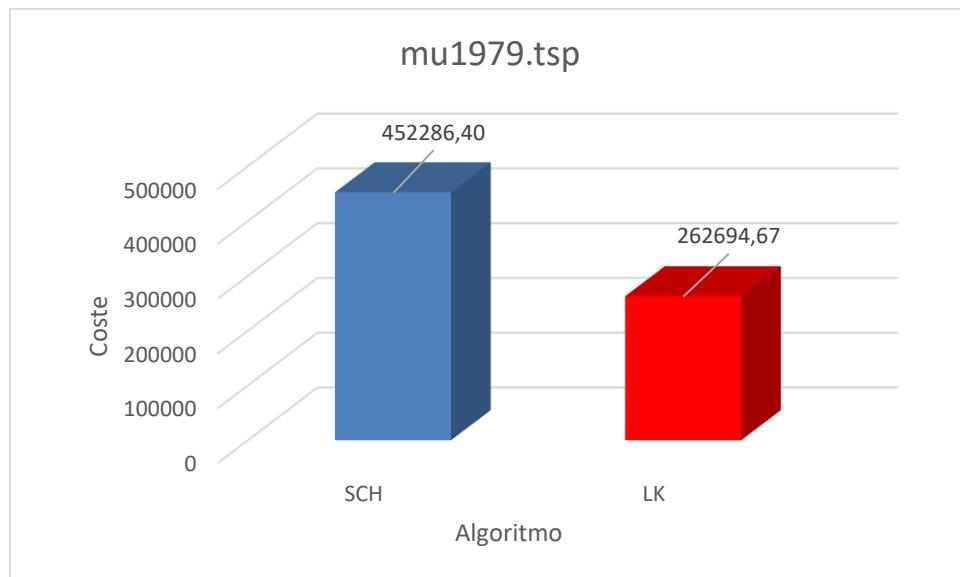


Figura 54. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se revela en la [Figura 54](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos mu1979.tsp, concretamente unos 189592 kilómetros menos en relación con el algoritmo SCH.

En conclusión, para conjuntos de datos de tamaño medio, el algoritmo de Lin-Kerninghan es el que mejores soluciones obtiene.

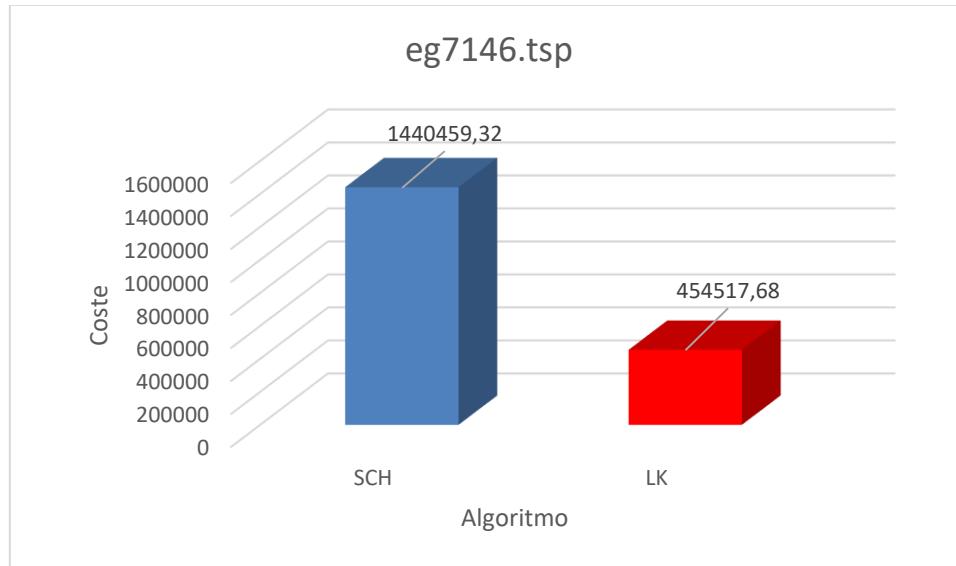


Figura 55. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se constata en la [Figura 55](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos eg7146.tsp, concretamente unos 985942 kilómetros menos con respecto al algoritmo SCH.

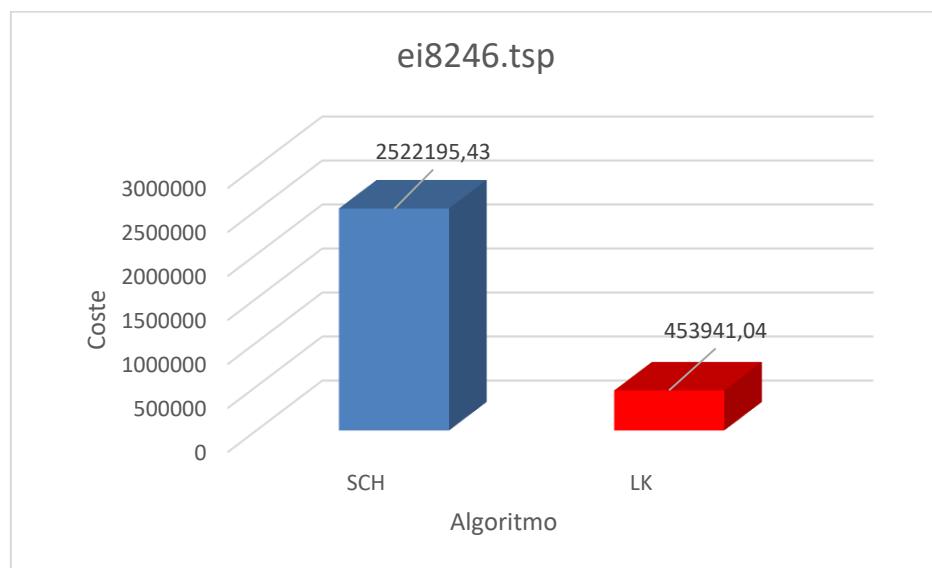


Figura 56. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se manifiesta en la [Figura 56](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones para el conjunto de datos ei8246.tsp, concretamente unos 2068254 kilómetros menos sobre el algoritmo SCH.

En conclusión, para conjuntos de datos de tamaño grande, el algoritmo de Lin-Kerninghan es el que mejores soluciones obtiene.

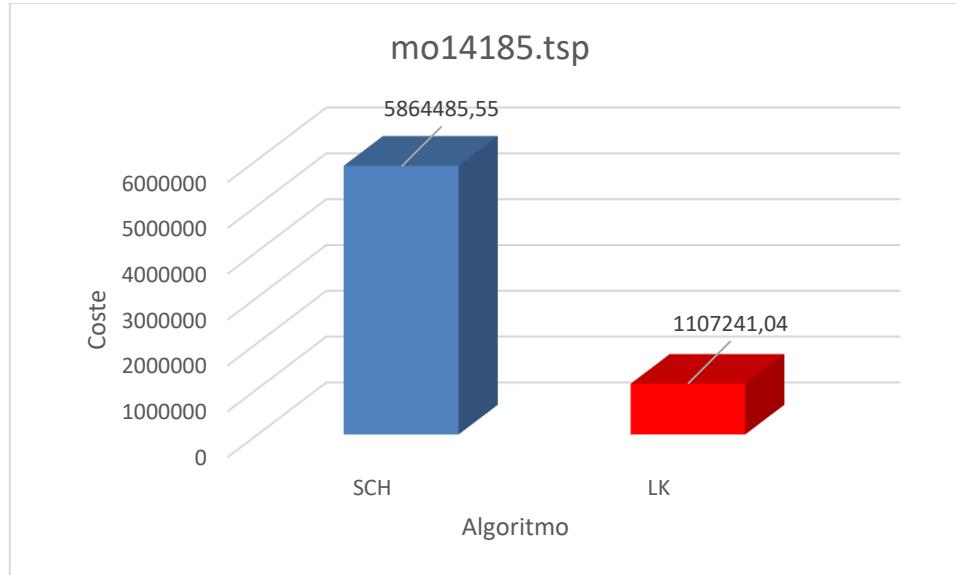


Figura 57. Diagrama comparando SCH y Lin-Kerninghan. Experimentos. Elaboración propia.

Como se muestra en la [Figura 57](#), el algoritmo de Lin-Kerninghan es el que obtiene las mejores soluciones por término medio para el conjunto de datos mo14185.tsp, concretamente unos 4757244 kilómetros menos en comparación con el algoritmo SCH. Esto parece indicar que deberíamos priorizarlo por delante del SCH, pero ahora vamos analizar los tiempos de ejecución para desmentir esto.

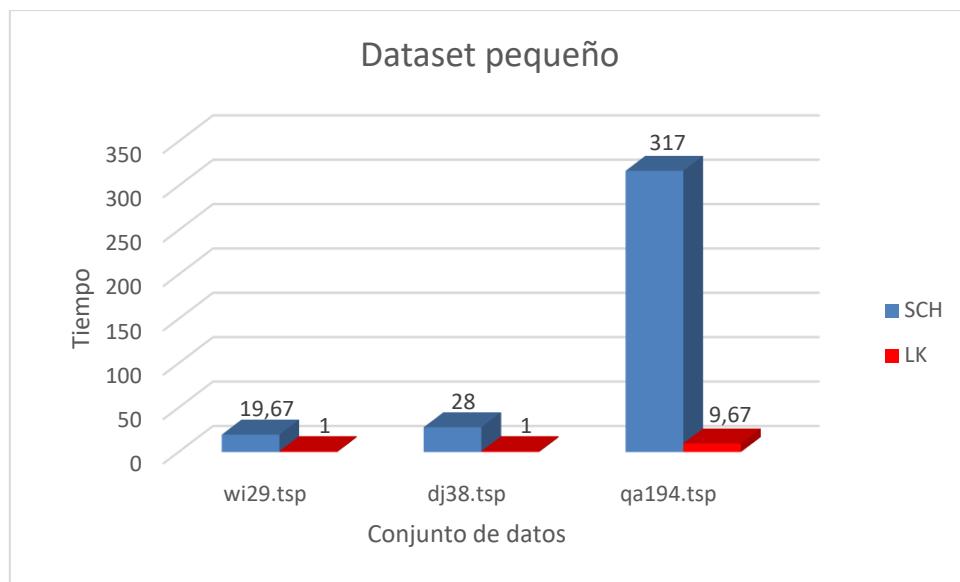


Figura 58. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

De acuerdo con lo observado en la [Figura 58](#), para datasets de tamaño pequeño, el algoritmo que menor tiempo de ejecución tiene es el de Lin-Kerninghan, ya que finaliza cuando en la iteración en curso no encuentra una solución mejor que la solución actual, realizando así pocas iteraciones. Por esa misma razón, el SCh tiene un tiempo de ejecución mayor pero aún así aceptable. Por tanto, podríamos decantarnos por usar cualquier algoritmo en datasets pequeños, ya que obtienen soluciones de una calidad similar y tiempos bajos de ejecución.

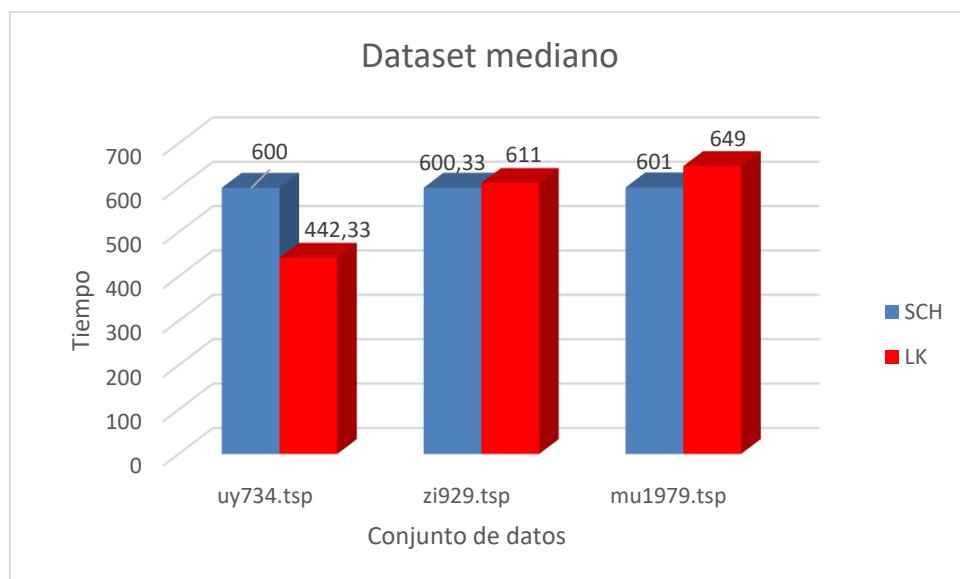


Figura 59. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

En la [Figura 59](#), podemos comprobar que, para datasets de tamaño medio, ambos algoritmos tienen tiempos de ejecución similares, lo que probablemente nos incline por el algoritmo de Lin-Kerninghan, que además obtiene mejores soluciones.

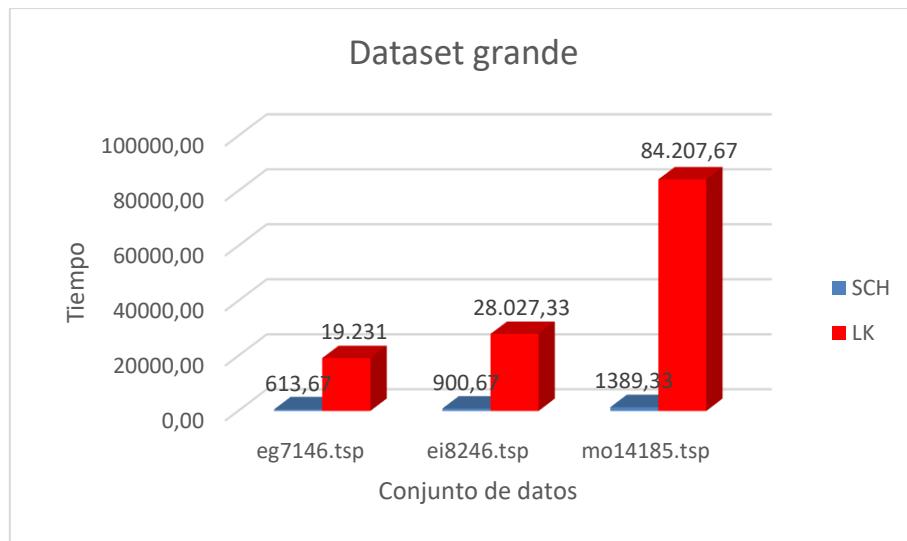


Figura 60. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

Como podemos comprobar en la [Figura 60](#), con datasets grandes el tiempo de ejecución se dispara en el algoritmo de Lin-Kerninghan debido a que su comportamiento hace que sea muy ineficiente para conjuntos de datos de este tamaño. Por tanto, aunque se obtengan mejores soluciones con el algoritmo de Lin-Kerninghan, yo optaría por el algoritmo de SCh para conjuntos de datos de tamaño grande porque presenta tiempos de ejecución mucho menores. Además, si ejecutamos este algoritmo por encima de los 10 minutos, realizará más iteraciones y obtendrá resultados mejores en menos tiempo que Lin-Kerninghan.

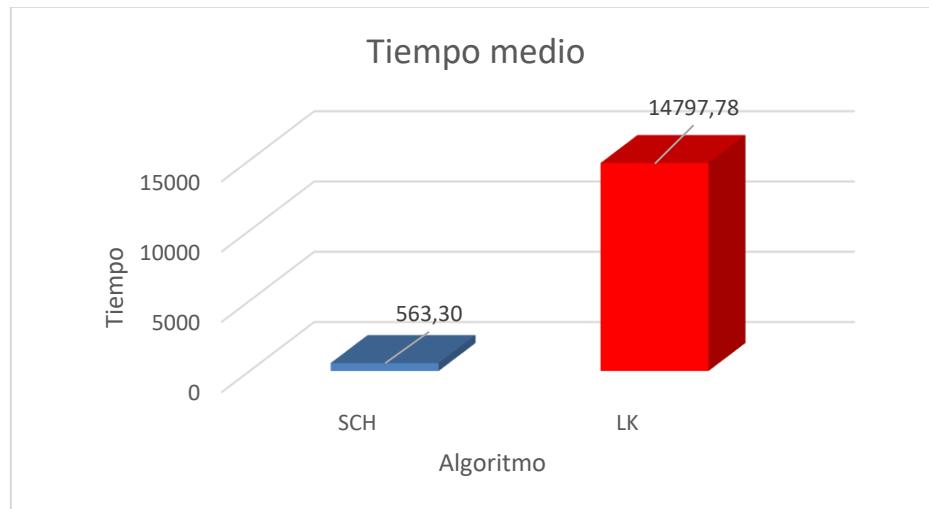


Figura 61. Diagrama comparando SCh y Lin-Kerninghan. Experimentos. Elaboración propia.

En la [Figura 61](#), se confirma nuevamente que, en términos medios, el algoritmo SCH es mucho más rápido que el algoritmo de Lin-Kernighan.

En conclusión, se recomienda el uso del algoritmo de Lin-Kernighan y SCH para datasets pequeños, ya que ofrecen un funcionamiento bastante similar. Para datasets medianos, se sugiere el algoritmo de Lin-Kernighan por proporcionar soluciones mejores en tiempos similares. Por último, para datasets grandes, se recomienda el SCH por ser más eficiente.

CAPÍTULO 7

CONCLUSIONES

Para concluir, se presentarán las conclusiones generales obtenidas a partir del desarrollo del proyecto y los conocimientos adquiridos durante el mismo. Además, se propondrán ideas para futuros trabajos basándose en la versión actual del sistema.

7.1. Conclusión

La aplicación de escritorio desarrollada en este proyecto es innovadora, destacando por su capacidad de representar gráficamente los resultados obtenidos a lo largo de su ejecución. Esta capacidad de visualización resulta especialmente beneficiosa para los usuarios que utilizan el sistema, permitiéndoles observar los resultados obtenidos por cada uno de los algoritmos implementados. A pesar de que la aplicación resalta por sus funcionalidades, existen otras opciones, como Concorde TSP Solver, que poseen funcionalidades muy similares.

Por otro lado, otra característica destacada del sistema es su capacidad para adaptarse a nuevos algoritmos. En otras palabras, cualquier persona que desee añadir un nuevo algoritmo solo tendrá que implementarlo y agregar el código correspondiente para que pueda seleccionarse en la interfaz y configurar sus parámetros. Esta característica hace que la aplicación sea escalable en cuanto al número de algoritmos y facilita la inclusión de nuevos métodos sin necesidad de modificar el código ya implementado.

Además de la visualización de resultados, la aplicación permite la importación de distintos conjuntos de datos, lo que posibilita la resolución de cualquier problema del tipo tratado. Esto hace que cualquier usuario pueda utilizar la aplicación para satisfacer sus necesidades, incluso las más personales, como la optimización del trayecto para llegar a su destino de vacaciones.

Adicionalmente, la aplicación tiene la capacidad de exportar los resultados obtenidos en archivos PDF, lo que facilita al usuario la visualización posterior, el análisis y la divulgación de los resultados.

Asimismo, el sistema facilita el paralelismo al permitir al usuario la ejecución simultánea de varias instancias del mismo o distintos algoritmos mediante la línea de comandos. Esto ahorra tiempos de espera al usuario y le permite despreocuparse mientras se están ejecutando los algoritmos.

De igual manera, el sistema es multiplataforma, ya que al tratarse de un archivo ejecutable de Java, cualquier sistema operativo que tenga instalada una Máquina Virtual de Java (JVM) podrá ejecutarlo. Esta característica es muy importante de cara a futuras ventas.

A su vez, el sistema ha permitido determinar cuál de los algoritmos implementados es el más eficiente para este problema, destacando específicamente el Sistema de Colonias de Hormigas y el algoritmo de Lin-Kernighan, siendo este último más adecuado para instancias pequeñas y medianas del TSP.

En última instancia, la solución propuesta ha logrado cumplir de manera satisfactoria con los objetivos y requisitos especificados al inicio de esta memoria. Se ha obtenido una interfaz usable e intuitiva, facilitando así el aprendizaje y la utilización por parte del usuario. Además, el software es escalable en cuanto al número de algoritmos gracias a la implementación del patrón de diseño de software Strategy. Esto asegura que la aplicación pueda evolucionar y adaptarse a las necesidades de los usuarios. Asimismo, se ha llevado a cabo un estudio exhaustivo e implementación de modelos metaheurísticos constructivos para resolver el problema cotidiano del TSP (Problema del Viajante de Comercio).

7.2. Conocimientos adquiridos

Durante la realización de este Trabajo de Fin de Grado, he tenido la oportunidad de ampliar significativamente mis conocimientos en diversas áreas relacionadas con la informática y la optimización combinatoria.

Uno de los aspectos fundamentales de este proyecto ha sido el desarrollo de la implementación en Java del Sistema de Colonias de Hormigas (SCH), Sistema de Hormigas Max-Min, Sistema de Hormigas del Mejor-Peor y la interfaz de usuario para su interacción. Esta experiencia me ha permitido profundizar mis conocimientos en programación orientada a objetos en Java y, más específicamente, en la creación de interfaces gráficas de usuario utilizando la biblioteca Java Swing. Aprendí a diseñar y estructurar componentes visuales, gestionar eventos de usuario y proporcionar una experiencia interactiva y eficiente.

La investigación y aplicación de metaheurísticas han sido un componente esencial de este proyecto. En particular, he explorado algoritmos constructivos, como el Sistema de Colonias de Hormigas (SCH), así como el Sistema de Hormigas Max-Min (SHMM) y el Sistema de Hormigas del Mejor-Peor (SHMP). Estos enfoques han ampliado mi comprensión sobre cómo las metaheurísticas pueden abordar problemas de optimización combinatoria, proporcionando soluciones eficientes y adaptativas.

La inclusión y estudio del algoritmo Lin-Kernighan (LK) han enriquecido mi comprensión de los algoritmos deterministas en el contexto de la optimización combinatoria. Este algoritmo, que se centra en la mejora de soluciones mediante movimientos de intercambio iterativos, ha sido una valiosa adición a mi conjunto de herramientas. La comparación entre enfoques deterministas y metaheurísticas ha enriquecido mi perspectiva sobre cómo abordar problemas de optimización con diferentes estrategias.

En conjunto, estos conocimientos adquiridos no solo han contribuido al éxito de este proyecto, sino que también han sentado las bases para mi crecimiento profesional futuro. La combinación de habilidades técnicas en Java, comprensión profunda de metaheurísticas y experiencia en la implementación de algoritmos deterministas ha creado una base sólida para enfrentar desafíos en la resolución de problemas de optimización y desarrollo de software.

Este proceso no solo ha fortalecido mis habilidades técnicas, sino que también ha cultivado mi capacidad para abordar problemas complejos y aplicar conocimientos teóricos en contextos prácticos. Este TFG ha sido una experiencia educativa integral que ha contribuido significativamente a mi desarrollo como profesional en el campo de la informática y la optimización combinatoria.

7.3. Futuros trabajos

Aunque la aplicación actualmente es funcional y cumple con los requisitos establecidos, siempre hay un margen para la mejora y la innovación. En esta sección, se proporcionará una lista de posibles mejoras y futuros trabajos que podrían implementarse en la aplicación para mejorar su funcionalidad y la experiencia del usuario:

- Esta aplicación está en constante crecimiento, ya que es ampliable para incluir nuevos algoritmos y visualizar los resultados. Actualmente, cuenta con algunos algoritmos constructivos para resolver el TSP, pero hay muchas variantes y nuevos algoritmos, como el del vecino más cercano (nearest neighbor), inserción más cercana (cheapest insertion), inserción del mejor (best insertion), etc. Además, al igual que el sistema tiene

implementado el algoritmo determinista de Lin-Kernighan, podrían incorporarse otros algoritmos, como el algoritmo de fuerza bruta.

- Dado que no se ha podido evaluar los algoritmos en ejecuciones prolongadas de varios días, sería beneficioso realizar un estudio de los mismos sobre instancias de problemas de tamaño considerable durante varios meses, aplicando mecanismos de reinicialización de la búsqueda y suavizado de feromonas. Esto ayudará a confirmar las conclusiones presentadas en esta memoria o a revisarlas.
- Con el objetivo de atraer a más usuarios, podría considerarse la modificación de la aplicación para que los algoritmos puedan aplicarse a otros tipos de problemas que son capaces de resolver, como el problema de la mochila, programación de tareas, secuenciación de ADN, entre otros.
- La aplicación actualmente opera con coordenadas 2D de las ciudades y calcula la distancia euclídea entre ellas. Sería posible realizar una modificación en el sistema para que funcione con coordenadas 3D y distancias reales entre las diversas ubicaciones.

ANEXO 1

DETALLES DE LA EXPERIMENTACIÓN

En este anexo, se presentarán los mismos resultados expuestos en el [capítulo 6](#), pero con un mayor nivel de detalle.

Tabla 31. Resultados iniciales ejecución SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 7914 | Óptimo global | 95345 | Óptimo global | 80891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|-------------|---------------|------------|---------------|--|---------------|--|---------------|---|---------------|--|---------------|--|---------------|--|---------------|--------------------------------|---------------|------------|
| | wi29.tsp | | dj38.tsp | <th>qa1934.tsp</th> <td><th>uy734.tsp</th><td><th>z1929.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td></td> | qa1934.tsp | <th>uy734.tsp</th> <td><th>z1929.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td> | uy734.tsp | <th>z1929.tsp</th> <td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td> | z1929.tsp | <th>mu1979.tsp</th> <td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td> | mu1979.tsp | <th>eg7146.tsp</th> <td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td> | eg7146.tsp | <th>ei8246.tsp</th> <td><th>mo14185.tsp</th><td></td></td> | ei8246.tsp | <th>mo14185.tsp</th> <td></td> | mo14185.tsp | |
| Ejecución 1 | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 2 | 27601,17 | 18 | 6659,43 | 27 | 12018,50 | 324 | 129498,44 | 600 | 156788,28 | 600 | 429470,43 | 601 | 1390977,79 | 614 | 2472382,69 | 906 | 5829242,57 | 1392 |
| Ejecución 3 | 27601,17 | 22 | 6659,43 | 26 | 12004,10 | 303 | 131379,70 | 600 | 156207,83 | 600 | 462243,14 | 600 | 1478598,90 | 613 | 2546062,94 | 895 | 5869988,79 | 1383 |
| Media | 27601,17 | 19,67 | 6659,43 | 28,00 | 12032,36 | 317,00 | 129921,63 | 600,00 | 156929,61 | 600,33 | 452286,40 | 601,00 | 1440459,32 | 613,67 | 2522195,43 | 900,67 | 5864485,55 | 1389,33 |
| Dsv. Típica | 0,00 | 2,08 | 0,00 | 2,65 | 37,18 | 12,12 | 1269,24 | 0,00 | 801,84 | 0,58 | 19812,43 | 1,00 | 44898,15 | 0,58 | 43151,61 | 5,51 | 32839,03 | 5,51 |

Tabla 32. Resultados aporte mejor global SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 w129.tsp | Óptimo global | 6656 d138.tsp | Óptimo global | 9352 qa194.tsp | Óptimo global | 79114 uj734.tsp | Óptimo global | 98345 z1929.tsp | Óptimo global | 86891 mu1979.tsp | Óptimo global | 172387 eg7146.tsp | Óptimo global | 206171 el8246.tsp | Óptimo global | 427377 mo14185.tsp |
|--------------|------------------|-------------------|------------------|------------------|------------------|-------------------|------------------|--------------------|------------------|--------------------|------------------|---------------------|------------------|----------------------|------------------|----------------------|------------------|-----------------------|
| | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601,17 | 19 | 6659,43 | 31 | 12074,48 | 324 | 129498,44 | 600 | 156788,28 | 600 | 429470,43 | 601 | 1390977,79 | 614 | 2472382,59 | 906 | 5829242,57 | 1392 |
| Ejecución 2 | 27601,17 | 18 | 6659,43 | 27 | 12018,50 | 324 | 128866,74 | 600 | 157792,72 | 601 | 465145,63 | 602 | 1451801,28 | 614 | 2548140,58 | 901 | 5894225,27 | 1393 |
| Ejecución 3 | 27601,17 | 22 | 6659,43 | 26 | 12004,10 | 303 | 131379,70 | 600 | 156207,83 | 600 | 462243,14 | 600 | 1478598,90 | 613 | 2546062,34 | 895 | 5869988,79 | 1383 |
| Media | 27601,17 | 19,67 | 6659,43 | 28,00 | 12032,36 | 317,00 | 129921,63 | 600,00 | 156929,61 | 600,33 | 452286,40 | 601,00 | 1440459,32 | 613,67 | 2522195,43 | 900,67 | 5864485,55 | 1389,33 |
| Desv. Típica | 0,00 | 2,08 | 0,00 | 2,65 | 37,18 | 12,12 | 1299,24 | 0,00 | 801,84 | 0,58 | 19812,43 | 1,00 | 44898,15 | 0,58 | 43151,61 | 5,51 | 32839,03 | 5,51 |

Tabla 33. Resultados aporte mejor iteración SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 w129.tsp | Óptimo global | 6656 dj38.tsp | Óptimo global | 9352 qa194.tsp | Óptimo global | 79114 uy734.tsp | Óptimo global | 95345 zg929.tsp | Óptimo global | 86891 mu1979.tsp | Óptimo global | 172387 eg7146.tsp | Óptimo global | 206171 ei8246.tsp | Óptimo global | 427377 mo14185.tsp |
|-------------|------------------|-------------------|------------------|------------------|------------------|-------------------|------------------|--------------------|------------------|--------------------|------------------|---------------------|------------------|----------------------|------------------|----------------------|------------------|-----------------------|
| | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27643,20 | 5 | 6662,35 | 7 | 12483,50 | 134 | 22357,85 | 606 | 407814,68 | 607 | 429470,43 | 612 | 1390977,79 | 628 | 2472382,59 | 678 | 5829242,57 | 1692 |
| Ejecución 2 | 27809,89 | 4 | 6719,54 | 7 | 12411,16 | 135 | 140571,27 | 600 | 418340,92 | 601 | 465145,63 | 604 | 1451801,28 | 606 | 2548140,38 | 1517 | 5894225,27 | 1066 |
| Ejecución 3 | 27891,97 | 5 | 6665,37 | 6 | 12244,66 | 135 | 137326,71 | 607 | 427270,31 | 632 | 462243,14 | 654 | 1478598,90 | 885 | 2546062,94 | 934 | 5869988,79 | 985 |
| Media | 27781,68 | 4,67 | 6682,42 | 6,67 | 12379,77 | 134,67 | 167138,61 | 604,33 | 417808,64 | 613,33 | 452286,40 | 623,33 | 1440459,32 | 706,33 | 2522195,43 | 1043,00 | 5864485,55 | 1247,67 |
| Dsv. Típica | 126,76 | 0,58 | 32,18 | 0,58 | 122,47 | 0,58 | 48852,80 | 3,79 | 9738,73 | 16,44 | 19812,43 | 26,86 | 44868,15 | 155,12 | 43151,61 | 429,99 | 32839,03 | 386,93 |

Tabla 34. Resultados aporte híbrido SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 7914 | Óptimo global | 95345 | Óptimo global | 86891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|--------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|
| | wi29.tsp | dj38.tsp | qa194.tsp | uy734.tsp | zg929.tsp | mu1979.tsp | eg7146.tsp | e8246.tsp | mo14185.tsp | | | | | | | | | |
| | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601.17 | 7 | 6659.43 | 10 | 12258.97 | 182 | 38019.51 | 606 | 407814.68 | 604 | 429470.43 | 622 | 1390977.79 | 1116 | 2472382.69 | 1105 | 5829242.57 | 1584 |
| Ejecución 2 | 27601.17 | 7 | 6659.43 | 10 | 12083.05 | 183 | 405453.70 | 604 | 418340.92 | 610 | 465145.63 | 649 | 1451801.28 | 1063 | 2548140.68 | 1033 | 5894225.27 | 1054 |
| Ejecución 3 | 27601.17 | 7 | 6659.43 | 10 | 11936.52 | 183 | 386630.87 | 602 | 427270.31 | 604 | 462243.14 | 656 | 1478598.90 | 724 | 2546062.94 | 740 | 5869988.79 | 691 |
| Media | 27601.17 | 7.00 | 6659.43 | 10.00 | 12092.85 | 182.67 | 390701.36 | 604.00 | 417808.64 | 606.00 | 452286.40 | 642.33 | 1440459.32 | 967.67 | 2522195.43 | 959.33 | 5864485.55 | 1109.67 |
| Desv. Típica | 0.00 | 0.00 | 0.00 | 0.00 | 161.45 | 0.58 | 13196.63 | 2.00 | 9738.73 | 3.46 | 19812.43 | 17.95 | 44898.15 | 212.68 | 43151.61 | 193.33 | 32839.03 | 449.10 |

Tabla 35. Resultados población 5 hormigas SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 86891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|--------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|
| | wi29.tsp | dij8.tsp | qat94.tsp | uy734.tsp | zij929.tsp | mu1979.tsp | eg7146.tsp | ej6246.tsp | mo14185.tsp | | | | | | | | | |
| Ejecución 1 | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601,17 | 8 | 6659,43 | 12 | 11486,55 | 600 | 126562,39 | 601 | 156432,24 | 604 | 463835,04 | 605 | 1432088,89 | 705 | 2484802,32 | 801 | 5827052,27 | 1262 |
| Ejecución 2 | 27748,71 | 18 | 6659,43 | 26 | 11673,15 | 600 | 131349,57 | 600 | 157361,59 | 600 | 458779,62 | 603 | 1463557,14 | 748 | 2542773,07 | 703 | 5880697,71 | 1013 |
| Ejecución 3 | 27601,17 | 51 | 6659,43 | 95 | 11463,05 | 600 | 129076,17 | 602 | 153619,41 | 600 | 468377,61 | 614 | 1457083,31 | 643 | 2515836,10 | 624 | 5937224,94 | 658 |
| Media | 27650,35 | 25,67 | 6659,43 | 44,33 | 11540,91 | 600,00 | 128996,04 | 601,00 | 155804,42 | 601,33 | 463664,09 | 607,33 | 1450909,78 | 698,67 | 2514470,50 | 709,33 | 5881658,31 | 977,67 |
| Desv. Típica | 85,18 | 22,50 | 0,00 | 44,43 | 115,12 | 0,00 | 2394,60 | 1,00 | 1948,48 | 2,31 | 4801,28 | 5,86 | 16617,67 | 52,79 | 28009,49 | 88,67 | 55092,62 | 303,55 |

Tabla 36. Resultados población 10 hormigas SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 w29.tsp | Óptimo global | 6656 dj38.tsp | Óptimo global | 9352 qat94.tsp | Óptimo global | 79114 uy734.tsp | Óptimo global | 95345 z1929.tsp | Óptimo global | 86891 mrt1979.tsp | Óptimo global | 172387 eg7146.tsp | Óptimo global | 206171 el8246.tsp | Óptimo global | 427377 mot14185.tsp |
|-------------|------------------|------------------|------------------|------------------|------------------|-------------------|------------------|--------------------|------------------|--------------------|------------------|----------------------|------------------|----------------------|------------------|----------------------|------------------|------------------------|
| | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601,17 | 19 | 6659,43 | 31 | 12074,48 | 324 | 129498,44 | 600 | 156788,28 | 600 | 429470,43 | 601 | 1390977,79 | 614 | 2472382,69 | 906 | 5829242,57 | 1392 |
| Ejecución 2 | 27601,17 | 18 | 6659,43 | 27 | 12018,50 | 324 | 128886,74 | 600 | 157792,72 | 601 | 465145,63 | 602 | 145181,28 | 614 | 2548140,68 | 901 | 5844225,27 | 1393 |
| Ejecución 3 | 27601,17 | 22 | 6659,43 | 26 | 12004,10 | 303 | 131379,70 | 600 | 156207,83 | 600 | 462243,14 | 600 | 147858,90 | 613 | 2546052,94 | 895 | 5869988,79 | 1383 |
| Media | 27601,17 | 19,67 | 6659,43 | 28,00 | 12032,36 | 317,00 | 129921,63 | 600,00 | 156929,61 | 600,33 | 452286,40 | 601,00 | 1440459,32 | 613,67 | 2522195,43 | 900,67 | 5864485,55 | 1389,33 |
| Dsv. Típica | 0,00 | 2,08 | 0,00 | 2,65 | 37,78 | 12,12 | 1289,24 | 0,00 | 801,84 | 0,58 | 19812,43 | 1,00 | 44898,15 | 0,58 | 43151,61 | 5,51 | 32839,03 | 5,51 |

Tabla 37. Resultados población 20 hormigas SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 wj29.tsp | Óptimo global | 6656 qj38.tsp | Óptimo global | 9352 qa194.tsp | Óptimo global | 79114 uj734.tsp | Óptimo global | 95345 zj929.tsp | Óptimo global | 86891 mu1979.tsp | Óptimo global | 172387 eg7146.tsp | Óptimo global | 206171 ei1246.tsp | Óptimo global | 427377 mo14185.tsp |
|-------------|------------------|-------------------|------------------|------------------|------------------|-------------------|------------------|--------------------|------------------|--------------------|------------------|---------------------|------------------|----------------------|------------------|----------------------|------------------|-----------------------|
| | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601,17 | 91 | 6659,43 | 215 | 12571,31 | 605 | 394909,33 | 604 | 424752,00 | 619 | 482803,99 | 668 | 1456847,59 | 1659 | 2522324,70 | 2051 | 6070169,12 | 3467 |
| Ejecución 2 | 27601,17 | 201 | 6659,43 | 229 | 12971,09 | 600 | 400657,80 | 617 | 422640,12 | 613 | 463772,49 | 695 | 1495495,13 | 1578 | 2553666,97 | 1811 | 5959305,54 | 3096 |
| Ejecución 3 | 27601,17 | 310 | 6659,43 | 374 | 12672,54 | 600 | 397398,27 | 613 | 432250,72 | 614 | 469810,89 | 627 | 1483078,48 | 965 | 2583261,39 | 1312 | 6039678,04 | 2478 |
| Media | 27601,17 | 200,67 | 6659,43 | 272,67 | 12738,31 | 601,67 | 397655,14 | 611,33 | 426547,61 | 615,33 | 472129,13 | 663,33 | 1478473,74 | 1400,67 | 2553091,02 | 1724,67 | 6023050,90 | 3013,67 |
| Dsv. Típica | 0,00 | 109,50 | 0,00 | 88,04 | 207,85 | 2,89 | 2882,83 | 6,66 | 5050,65 | 3,21 | 9725,23 | 34,24 | 1970,96 | 379,47 | 30472,72 | 376,99 | 57271,55 | 499,61 |

Tabla 38. Resultados iteraciones y tiempo de ejecución SCH. Anexo 1. Elaboración propia.

| SCH | Óptimo global | 27600 | Óptimo global | 6056 | Óptimo global | 9352 | Óptimo global | 7914 | Óptimo global | 95345 | Óptimo global | 86891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|--------------|---------------|---------|---------------|-----------|---------------|------------|---------------|------------|---------------|---------|---------------|---------|---------------|---------|---------------|---------|---------------|---------|
| | wi291isp | dj381sp | qa194 isp | uy734 isp | zg291isp | mu1979 isp | eg7146 isp | ei8246 isp | mo14185 isp | | | | | | | | | |
| Ejecución 1 | 27601,17 | 284 | 6659,43 | 394 | 12018,48 | 3600 | 126968,31 | 3623 | 152513,59 | 3634 | 429470,43 | 3619 | 1380977,79 | 4139 | 2472332,69 | 4178 | 5829242,57 | 4226 |
| Ejecución 2 | 27601,17 | 297 | 6659,43 | 389 | 11788,26 | 3601 | 127200,83 | 3612 | 152832,79 | 3613 | 465145,63 | 3637 | 1451801,28 | 3950 | 2548140,68 | 3866 | 5894225,27 | 3919 |
| Ejecución 3 | 27601,17 | 133 | 6659,43 | 552 | 11966,27 | 3600 | 125616,92 | 3601 | 154667,22 | 3601 | 462243,14 | 3600 | 1478598,90 | 3647 | 2546082,94 | 3730 | 5869988,79 | 3794 |
| Media | 27601,17 | 238,00 | 6659,43 | 445,00 | 11924,34 | 3600,33 | 126595,35 | 3611,67 | 153337,87 | 3616,00 | 452286,40 | 3618,67 | 1440459,32 | 3912,00 | 2522195,43 | 3924,67 | 5864485,55 | 3879,67 |
| Desv. Típica | 0,00 | 91,16 | 0,00 | 92,70 | 120,70 | 0,58 | 855,29 | 11,50 | 1162,26 | 16,70 | 19812,43 | 18,50 | 44898,15 | 248,19 | 43151,61 | 229,69 | 32839,03 | 222,30 |

Tabla 39. Resultados iniciales ejecución SHMM. Anexo 1. Elaboración propia.

| SHMM | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 88891 | Óptimo global | 172387 | Óptimo global | 20671 | Óptimo global | 42737 |
|--------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|
| | wi29.tsp | dj38.tsp | qa134.tsp | uy734.tsp | z1929.tsp | mu199.tsp | eg7146.tsp | ei8246.tsp | mo14185.tsp | | | | | | | | | |
| Ejecución 1 | Coste (km) | Tiempo (s) |
| | 29630,04 | 39 | 6742,73 | 54 | 12486,24 | 491 | 116155,97 | 601 | 142082,02 | 601 | 137311,39 | 601 | 1454722,34 | 675 | 2554895,22 | 893 | 6042064,93 | 1363 |
| Ejecución 2 | 29034,18 | 34 | 7046,33 | 54 | 11289,09 | 509 | 114545,78 | 600 | 144434,52 | 600 | 191070,21 | 603 | 1459884,43 | 675 | 2609402,12 | 893 | 6159983,43 | 1309 |
| Ejecución 3 | 28680,93 | 33 | 6819,74 | 44 | 12202,09 | 510 | 117424,75 | 600 | 13731,39 | 600 | 190304,27 | 601 | 1453822,05 | 674 | 2609402,12 | 893 | 6164466,75 | 1323 |
| Media | 29115,05 | 35,33 | 6869,60 | 50,67 | 11992,47 | 503,33 | 116042,17 | 600,33 | 141275,98 | 600,33 | 172895,29 | 601,67 | 1456142,94 | 674,67 | 2591233,15 | 893,00 | 6122171,70 | 1331,67 |
| Desv. Típica | 479,69 | 3,21 | 157,82 | 5,77 | 625,49 | 10,69 | 1442,85 | 0,58 | 3629,33 | 0,58 | 30818,94 | 1,15 | 3271,35 | 0,58 | 31469,57 | 0,00 | 69410,71 | 28,02 |

| SHMM | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 86891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|-------------|---------------|-------|---------------|--|---------------|--|---------------|---|---------------|--|---------------|--|---------------|--|---------------|--------------------------------|---------------|---------|
| | wi29.tsp | | dj38.tsp | <th>qa1934.tsp</th> <td><th>uy734.tsp</th><td><th>zg929.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td></td> | qa1934.tsp | <th>uy734.tsp</th> <td><th>zg929.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td> | uy734.tsp | <th>zg929.tsp</th> <td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td> | zg929.tsp | <th>mu1979.tsp</th> <td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td> | mu1979.tsp | <th>eg7146.tsp</th> <td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td> | eg7146.tsp | <th>ei8246.tsp</th> <td><th>mo14185.tsp</th><td></td></td> | ei8246.tsp | <th>mo14185.tsp</th> <td></td> | mo14185.tsp | |
| Ejecución 1 | 27601,17 | 13 | 6659,43 | 25 | 11239,68 | 600 | 117879,13 | 604 | 147270,82 | 602 | 298415,78 | 682 | 1511618,42 | 1375 | 2568844,19 | 1378 | 6042064,93 | 1558 |
| Ejecución 2 | 27748,71 | 15 | 6659,43 | 26 | 10784,11 | 600 | 115849,53 | 604 | 150685,65 | 625 | 380556,96 | 622 | 1505712,11 | 1075 | 2609402,12 | 1155 | 615983,43 | 1115 |
| Ejecución 3 | 27601,17 | 28 | 6659,43 | 49 | 11700,98 | 600 | 143806,52 | 601 | 179445,15 | 600 | 474237,70 | 611 | 1463173,58 | 831 | 2602944,92 | 747 | 6056380,69 | 679 |
| Media | 27650,35 | 18,67 | 6659,43 | 33,33 | 11241,59 | 600,00 | 125845,06 | 603,00 | 159133,87 | 609,00 | 384403,48 | 638,33 | 1493501,37 | 1093,67 | 2593730,41 | 1093,33 | 6086143,02 | 1117,33 |
| Dsv. Típica | 85,18 | 8,14 | 0,00 | 13,58 | 458,44 | 0,00 | 15588,15 | 1,73 | 17672,75 | 13,89 | 87974,05 | 38,21 | 26430,14 | 272,48 | 21792,58 | 319,99 | 64347,03 | 439,50 |

Tabla 40. Resultados sin reinicialización SHMM. Anexo 1. Elaboración propia.

| SHMM | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 86891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|-------------|---------------|-------|---------------|--|---------------|---|---------------|--|---------------|--|---------------|--|---------------|--|---------------|--------------------------------|---------------|---------|
| | wi29.tsp | | dj38.tsp | <th>qa194.tsp</th> <td><th>uy734.tsp</th><td><th>zg29.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td></td> | qa194.tsp | <th>uy734.tsp</th> <td><th>zg29.tsp</th><td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td></td> | uy734.tsp | <th>zg29.tsp</th> <td><th>mu1979.tsp</th><td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td></td> | zg29.tsp | <th>mu1979.tsp</th> <td><th>eg7146.tsp</th><td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td></td> | mu1979.tsp | <th>eg7146.tsp</th> <td><th>ei8246.tsp</th><td><th>mo14185.tsp</th><td></td></td></td> | eg7146.tsp | <th>ei8246.tsp</th> <td><th>mo14185.tsp</th><td></td></td> | ei8246.tsp | <th>mo14185.tsp</th> <td></td> | mo14185.tsp | |
| Ejecución 1 | 27601,17 | 13 | 6659,43 | 25 | 11239,68 | 600 | 117879,13 | 604 | 147270,82 | 602 | 298415,78 | 682 | 15116178,42 | 1375 | 2568844,19 | 1378 | 6042064,93 | 1558 |
| Ejecución 2 | 27748,71 | 15 | 6659,43 | 26 | 10784,11 | 600 | 115849,53 | 604 | 150685,65 | 625 | 380556,96 | 622 | 1505712,11 | 1075 | 2609402,12 | 1155 | 6159983,43 | 1115 |
| Ejecución 3 | 27601,17 | 28 | 6659,43 | 49 | 11700,98 | 600 | 143806,52 | 601 | 179445,15 | 600 | 474237,70 | 611 | 1463173,58 | 831 | 2602944,92 | 747 | 6056380,69 | 679 |
| Media | 27650,35 | 18,67 | 6659,43 | 33,33 | 11241,59 | 600,00 | 125845,06 | 603,00 | 159133,87 | 609,00 | 384403,48 | 638,33 | 1493501,37 | 1093,67 | 2593730,41 | 1093,33 | 6086143,02 | 1117,33 |
| Dsv. Típica | 85,18 | 8,14 | 0,00 | 13,58 | 458,44 | 0,00 | 15588,15 | 1,73 | 17672,75 | 13,89 | 87974,05 | 38,21 | 26430,14 | 272,48 | 21792,58 | 319,99 | 64347,03 | 439,50 |

Tabla 41. Resultados con reinicialización SHMM. Anexo 1. Elaboración propia.

Tabla 42. Resultados sin suavizado de feromona SHMM. Anexo 1. Elaboración propia.

| SHMM | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 88891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|--------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|
| | w29.tsp | | dj38.tsp | | qa194.tsp | | uy734.tsp | | zg929.tsp | | mu1979.tsp | | eg7146.tsp | | ei8246.tsp | | mo14185.tsp | |
| Ejecución 1 | Coste (km) | Tiempo (s) |
| Ejecución 1 | 29630,04 | 39 | 6742,73 | 54 | 12486,24 | 491 | 116155,97 | 601 | 142052,02 | 601 | 137311,39 | 601 | 1454722,34 | 675 | 2554895,22 | 893 | 6042064,93 | 1363 |
| Ejecución 2 | 29034,18 | 34 | 7046,33 | 54 | 11289,09 | 509 | 114545,78 | 600 | 144434,52 | 600 | 191070,21 | 603 | 1458884,43 | 675 | 2609402,12 | 893 | 6159983,43 | 1309 |
| Ejecución 3 | 28680,93 | 33 | 6819,74 | 44 | 12202,09 | 510 | 117424,75 | 600 | 137311,39 | 600 | 190304,27 | 601 | 1453822,05 | 674 | 2609402,12 | 893 | 6164466,75 | 1323 |
| Media | 29115,05 | 35,33 | 6869,60 | 50,67 | 11992,47 | 503,33 | 116042,17 | 600,33 | 141275,98 | 600,33 | 172895,29 | 601,67 | 1456142,94 | 674,67 | 2591233,15 | 893,00 | 6122171,70 | 1331,67 |
| Desv. Típica | 479,69 | 3,21 | 157,82 | 5,77 | 625,49 | 10,69 | 1442,85 | 0,58 | 3625,33 | 0,58 | 30818,94 | 1,15 | 3271,35 | 0,58 | 31469,57 | 0,00 | 69410,71 | 28,02 |

Tabla 43. Resultados con suavizado de feromona SHMM. Anexo 1. Elaboración propia.

| SHMM | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 88891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 |
|--------------|------------------|---------------|------------------|---|------------------|--|------------------|---|------------------|--|------------------|---------------|------------------|---------------|------------------|---------------|------------------|---------------|
| | w29.tsp | | dj38.tsp | <th>qa194.tsp</th> <td><th>uy734.tsp</th><td><th>zg929.tsp</th><td><th>mu1979.tsp</th><td></td><th>eg7146.tsp</th><td></td><th>ei8246.tsp</th><td></td><th>mo14185.tsp</th><td></td></td></td></td> | qa194.tsp | <th>uy734.tsp</th> <td><th>zg929.tsp</th><td><th>mu1979.tsp</th><td></td><th>eg7146.tsp</th><td></td><th>ei8246.tsp</th><td></td><th>mo14185.tsp</th><td></td></td></td> | uy734.tsp | <th>zg929.tsp</th> <td><th>mu1979.tsp</th><td></td><th>eg7146.tsp</th><td></td><th>ei8246.tsp</th><td></td><th>mo14185.tsp</th><td></td></td> | zg929.tsp | <th>mu1979.tsp</th> <td></td> <th>eg7146.tsp</th> <td></td> <th>ei8246.tsp</th> <td></td> <th>mo14185.tsp</th> <td></td> | mu1979.tsp | | eg7146.tsp | | ei8246.tsp | | mo14185.tsp | |
| Ejecución 1 | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 28909,07 | 36 | 6667,03 | 60 | 12363,39 | 600 | 116228,87 | 602 | 145293,08 | 606 | 282889,51 | 621 | 151618,42 | 685 | 256844,19 | 820 | 6042064,93 | 1709 |
| Ejecución 2 | 28177,24 | 25 | 6694,73 | 134 | 11045,73 | 600 | 114545,78 | 600 | 149774,80 | 600 | 282889,51 | 603 | 1505712,11 | 703 | 2609402,12 | 1375 | 6159983,43 | 1291 |
| Ejecución 3 | 28386,82 | 75 | 6819,74 | 97 | 12070,34 | 600 | 117424,75 | 602 | 139494,49 | 600 | 265770,89 | 650 | 1463173,58 | 1104 | 2602944,92 | 1093 | 6164466,75 | 1003 |
| Media | 28491,04 | 45,33 | 6727,17 | 97,00 | 11826,49 | 600,00 | 116066,47 | 601,33 | 144854,13 | 602,00 | 277183,30 | 624,67 | 1493501,37 | 830,67 | 2593730,41 | 1096,00 | 6122177,170 | 1334,33 |
| Desv. Típica | 376,88 | 26,27 | 81,36 | 37,00 | 691,85 | 0,00 | 1446,34 | 1,15 | 5154,19 | 3,46 | 9883,44 | 23,71 | 26430,14 | 236,88 | 21792,58 | 277,51 | 69410,71 | 354,99 |

Tabla 44. Resultados iniciales ejecución SHMP. Anexo 1. Elaboración propia.

| SHMP | Óptimo global | 27600 w129.tsp | Óptimo global | 6656 d138.tsp | Óptimo global | 9352 qat94.tsp | Óptimo global | 79114 uy734.tsp | Óptimo global | 95345 zg929.tsp | Óptimo global | 86891 mu1979.tsp | Óptimo global | 172387 eg7146.tsp | Óptimo global | 206171 ei8246.tsp | Óptimo global | 427377 mo14185.tsp |
|--------------|------------------|-------------------|------------------|------------------|------------------|-------------------|------------------|--------------------|------------------|--------------------|------------------|---------------------|------------------|----------------------|------------------|----------------------|------------------|-----------------------|
| | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) |
| Ejecución 1 | 27601,17 | 24 | 6659,43 | 28 | 17125,40 | 257 | 197783,9274 | 600 | 222703,7394 | 600 | 460424,73 | 602 | 1484949,576 | 647 | 2568826,865 | 895 | 6042064,925 | 1190 |
| Ejecución 2 | 27601,17 | 23 | 6659,43 | 32 | 16939,99 | 257 | 202958,2057 | 600 | 236538,9426 | 600 | 462029,33 | 602 | 1481437,491 | 647 | 2609402,118 | 892 | 6159983,431 | 1113 |
| Ejecución 3 | 27601,17 | 18 | 6659,43 | 32 | 16491,99 | 283 | 2131193952 | 600 | 240052,0547 | 600 | 465957,94 | 600 | 1463173,578 | 646 | 2592685,06 | 891 | 6164466,751 | 775 |
| Media | 27601,17 | 21,67 | 6659,43 | 30,67 | 16852,46 | 265,67 | 204620,70 | 600,00 | 233098,25 | 600,00 | 462804,00 | 601,33 | 1476520,22 | 646,67 | 2590304,68 | 892,67 | 6122171,70 | 1026,00 |
| Desv. Típica | 0,00 | 3,21 | 0,00 | 2,31 | 325,65 | 15,01 | 7802,01 | 0,00 | 9171,69 | 0,00 | 2846,78 | 1,15 | 11691,16 | 0,58 | 20392,09 | 2,08 | 69410,71 | 220,76 |

Tabla 45. Comparación algoritmos constructivos. Anexo 1. Elaboración propia.

| | | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 79114 | Óptimo global | 95345 | Óptimo global | 88891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 | MEDIA |
|--------|---------|---------------|---------|---------------|---------|---------------|--------|---------------|-------|---------------|---------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|-------|
| | | wi29.tspz | | dj38.tsp | | qa194.tsp | | uy734.tsp | | zr929.tsp | | mu1979.tsp | | eg7146.tsp | | eib246.tsp | | mo14185.tsp | | |
| GLOBAL | | | | | | | | | | | | | | | | | | | | |
| SCH | 27601,1 | 19,67 | 6659,43 | 28,00 | 12032,3 | 6 | 317,00 | 129921, | 63 | 600,00 | 156929, | 61 | 600,33 | 452286, | 40 | 601,00 | 144045 | 9,32 | 252219 | 5,43 |
| SHMM | 29115,0 | 35,33 | 6869,60 | 50,67 | 11992,4 | 7 | 503,33 | 116042, | 17 | 600,33 | 141275, | 98 | 600,33 | 172895, | 29 | 601,67 | 145614 | 674,67 | 259123 | 3,15 |
| SHMP | 27601,1 | 21,67 | 6659,43 | 30,67 | 16852,4 | 6 | 265,67 | 204620, | 70 | 600,00 | 233098, | 25 | 600,00 | 462804, | 00 | 601,33 | 147652 | 646,67 | 259030 | 4,68 |

| LK | Optimo global | 27800 | Optimo global | 6856 | Optimo global | 9352 | Optimo global | 79114 | Optimo global | 95345 | Optimo global | 86891 | Optimo global | 172387 | Optimo global | 206171 | Optimo global | 427377 |
|--------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|
| | wi291.tsp | dj38.tsp | qa194.tsp | uy734.tsp | zr929.tsp | mri979.tsp | eg7146.tsp | ei8246.tsp | mo14185.tsp | | | | | | | | | |
| | Coste (km) | Tiempo (s) |
| Ejecución 1 | 29096,9 | 1 | 7319,42 | 1 | 9712,74 | 7 | 85492,5 | 8073 | 521 | 107737,01 | 612 | 270810,87 | 649 | 465346,29 | 19424 | 461488,66 | 28036 | 1116040,85 |
| Ejecución 2 | 29802,6 | 1 | 6883,90 | 1 | 9717,70 | 11 | 87302,4 | 8376 | 425 | 105513,49 | 610 | 286952,64 | 650 | 450158,14 | 19436 | 448654,98 | 28121 | 1101457,31 |
| Ejecución 3 | 29130,9 | 1 | 7214,90 | 1 | 9642,93 | 11 | 91255,1 | 7036 | 381 | 103700,62 | 611 | 230320,49 | 648 | 448048,62 | 18833 | 451679,49 | 22925 | 1104224,96 |
| Media | 29343,5 | 1,00 | 7139,41 | 1,00 | 9691,12 | 9,67 | 88016,74 | 442,33 | 105650,37 | 611,00 | 262694,67 | 649,00 | 454517,68 | 19231,00 | 453941,04 | 28027,33 | 1107241,04 | 84207,67 |
| Desv. Tipica | 397,96 | 0,00 | 227,36 | 0,00 | 41,81 | 2,31 | 2946,95 | 71,59 | 2021,67 | 1,00 | 29175,41 | 1,00 | 9436,98 | 344,73 | 6709,09 | 98,29 | 7745,48 | 468,49 |

Tabla 46. Resultados iniciales ejecución Lin-Kernighan. Anexo 1. Elaboración propia.

Tabla 47. Comparación SCH y Lin-Kerninghan. Anexo 1. Elaboración propia.

| | | Óptimo global | 27600 | Óptimo global | 6656 | Óptimo global | 9352 | Óptimo global | 7914 | Óptimo global | 95345 | Óptimo global | 88891 | Óptimo global | 172387 | Óptimo global | 206171 | Óptimo global | 427377 | MEDIA | |
|--------|-----|---------------|------------|---------------|------------|---------------|---|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|---------------|------------|-----------|----------|
| | | wi29.tspz | | dj38.tsp | | qp194.tsp | <th>wy734.tsp</th> <td></td> <th>zb29.tsp</th> <td></td> <th>mu1979.tsp</th> <td></td> <th>eg7146.tsp</th> <td></td> <th>eb8246.tsp</th> <td></td> <th>mo14185.tsp</th> <td></td> <th></th> | wy734.tsp | | zb29.tsp | | mu1979.tsp | | eg7146.tsp | | eb8246.tsp | | mo14185.tsp | | | |
| GLOBAL | SCH | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | Coste (km) | Tiempo (s) | | |
| | SCH | 27601,1 | 19,67 | 6659,43 | 28,00 | 12032,3 | 317,00 | 129921,63 | 600,00 | 156929,61 | 601,00 | 452286,40 | 600,00 | 252219,54,3 | 900,67 | 586448,5,55 | 1389,33 | 117917,4,55 | 563,30 | | |
| | LK | 29343,5 | 1,00 | 7139,41 | 1,00 | 9691,12 | 9,67 | 88016,74 | 442,33 | 105650,37 | 611,00 | 262694,67 | 649,00 | 88016,74 | 422,33 | 453941,04 | 28027,33 | 110724,1,04 | 84207,67 | 279803,95 | 14797,78 |

ANEXO 2

MANUAL DE USUARIO

Antes de explicar las distintas maneras de utilizar la aplicación, es necesario obtener la carpeta del [repositorio](#) donde se encuentra.

Una vez obtenido el archivo, se puede iniciar la aplicación de dos maneras distintas, las cuales se describen a continuación:

1. Consola

Para iniciar la aplicación desde la consola, únicamente hay que ejecutar el siguiente comando:

```
java -jar -XmxAg [ruta_del_ejecutable]/TSP.jar [ruta_fichero_de_parametros]
```

Cabe destacar que en la parte -XmxAg del comando, es necesario reemplazar la A por un número entero positivo. Esta opción se utiliza para asignar más memoria al heap de la aplicación, especialmente útil para conjuntos de datos grandes. En mi caso, el mayor valor de memoria que puedo asignar al heap de la aplicación con mi ordenador es de 50 GB.

Por otro lado, también es importante asegurarse de que el formato del fichero de parámetros sea el correcto, tal y como se especificó en el apartado [5.2.3.4](#) de esta memoria. Dentro de este fichero, se deben especificar los conjuntos de datos a utilizar, siguiendo el formato TSPLIB explicado en la sección [5.2.3.1](#) de esta memoria.

Con el fin de facilitar la ejecución, se subirá al [repositorio](#) un ejemplo de archivo de parámetros y algunos conjuntos de datos asociados.

Por último, es crucial no modificar la estructura de la carpeta descargada del [repositorio](#) ni cambiar la ubicación del fichero ejecutable, ya que este depende de los archivos y directorios contenidos en dicha carpeta.

2. Ejecutable

Una vez hemos descargado la carpeta del [repositorio](#), solo tenemos que abrirla y hacer doble clic en el ejecutable TSP.jar. Antes de continuar, quiero recalcar nuevamente que no se debe modificar el contenido de la carpeta para que el ejecutable funcione correctamente.

Después de abrir la aplicación, aparecerá la pantalla de inicio que se ve en la [Figura 62](#). En esta pantalla, se nos permite seleccionar un conjunto de datos, un algoritmo o una solución obtenida previamente.

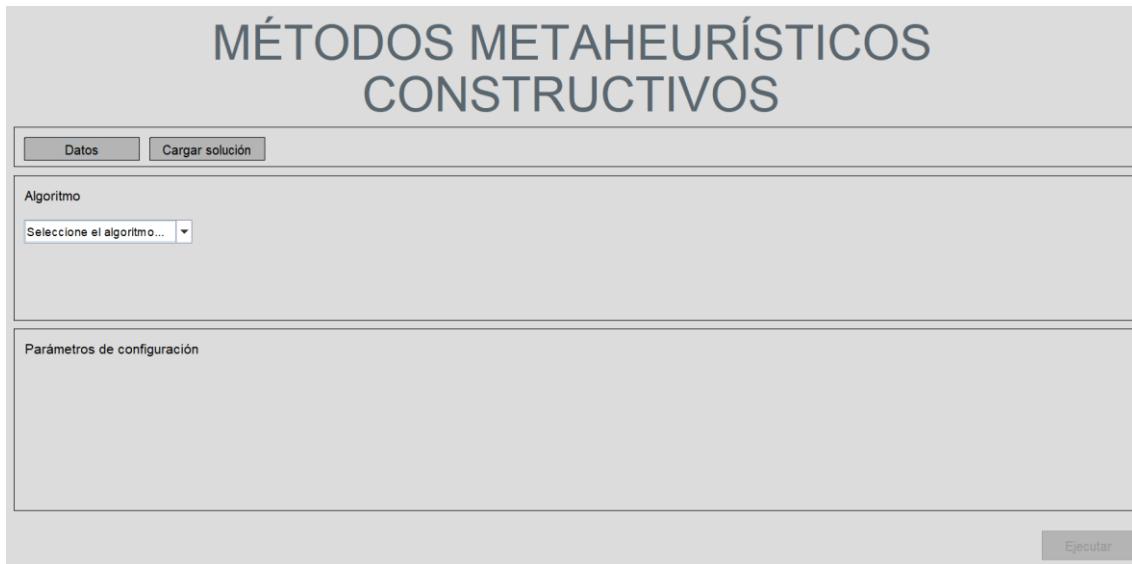


Figura 62. Pantalla inicial. Manual de usuario. Elaboración propia.

Comencemos seleccionando un conjunto de datos. Para ello, pulsaremos la opción correspondiente, lo que abrirá una ventana emergente de selección de archivos, como se muestra en la [Figura 63](#).

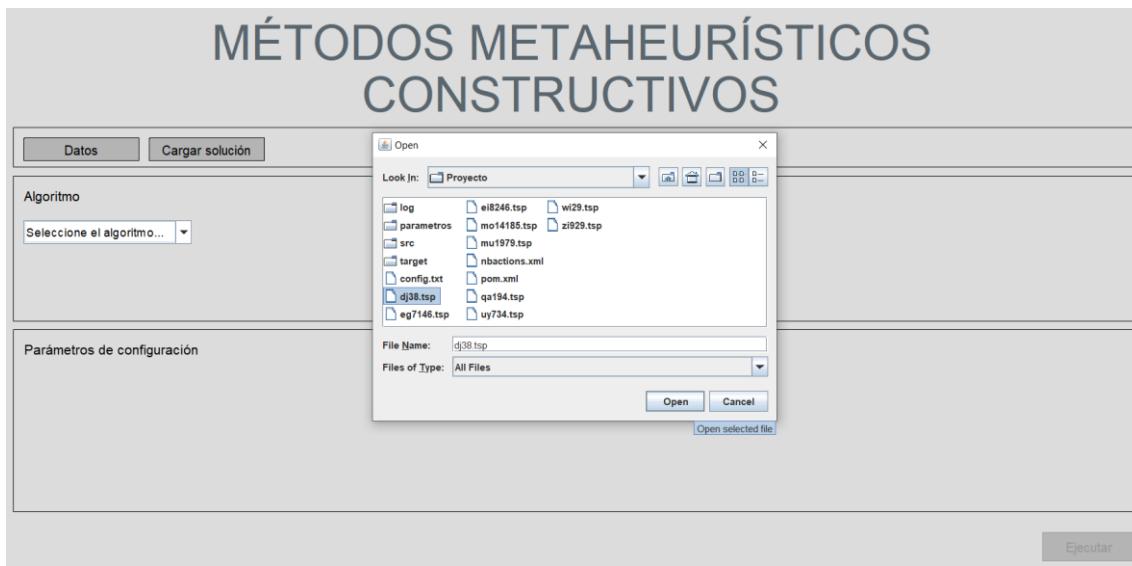


Figura 63. Importación del conjunto de datos. Manual de usuario. Elaboración propia.

Cabe destacar que, si el conjunto de datos no tiene el formato deseado, se le informará al usuario de que el archivo es incorrecto para que pueda seleccionar uno que sí sea válido, de acuerdo con lo representado en la [Figura 64](#).

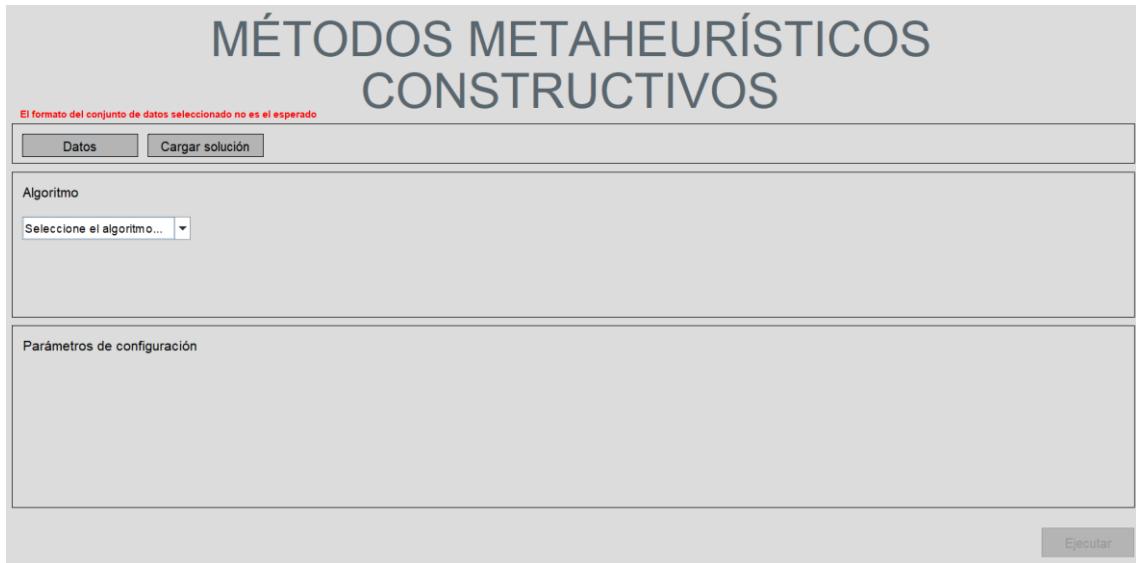


Figura 64. Conjunto de datos erróneo. Manual de usuario. Elaboración propia.

Tras haber seleccionado el conjunto de datos, procedemos a elegir el algoritmo a ejecutar, tal como se muestra en la [Figura 65](#). Quiero enfatizar que el algoritmo puede seleccionarse antes que el conjunto de datos.

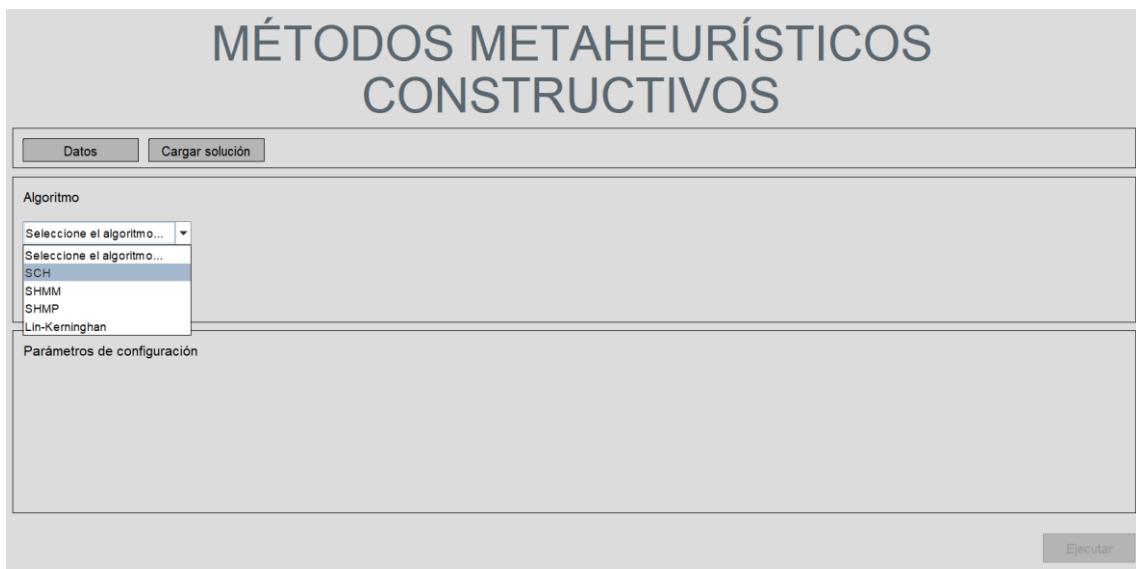


Figura 65. Selección de algoritmo. Manual de usuario. Elaboración propia.

Al seleccionar un algoritmo, aparecerán sus parámetros, los cuales podemos modificar o dejar por defecto. En el momento en que el conjunto de datos seleccionado sea correcto, se haya elegido un algoritmo y sus parámetros sean válidos, la aplicación nos permitirá ejecutarlo, como se ilustra en la [Figura 66](#).

The screenshot shows the application's main window titled 'MÉTODOS METAHEURÍSTICOS CONSTRUCTIVOS'. At the top, there are two buttons: 'Datos' and 'Cargar solución'. Below this is a section labeled 'Algoritmo' with a dropdown menu set to 'SCH'. The next section is 'Parámetros de configuración' containing the following parameters:

| | | | | | |
|-----------------------------|--------------|----------|------|-----------------------------------|-----|
| Iteraciones | 10000 | α | 1 | Actualización global de feromonas | 0.1 |
| Segundos de ejecución | 600 | β | 2 | Actualización local de feromonas | 0.1 |
| Hormiga que aporta feromona | Mejor global | q_0 | 0.95 | Feromona inicial | 100 |
| | | Semilla | | Tamaño de la población | 10 |

At the bottom right is a large 'Ejecutar' button.

Figura 66. Configuración de parámetros. Manual de usuario. Elaboración propia,

Según se detalla en la [Figura 67](#), en caso de que algún valor de un parámetro sea erróneo, se le avisará al usuario para que lo corrija y no se permitirá la ejecución del algoritmo.

The screenshot shows the same application interface as Figure 66. In the 'Parámetros de configuración' section, the 'Iteraciones' field contains the invalid value '-4'. A red error message 'Valor entero positivo' is displayed above the field. The other parameters are set to their default values: Iteraciones: -4, Segundos de ejecución: 600, Hormiga que aporta feromona: Mejor global, α : 1, β : 2, q_0 : 0.95, Semilla: (empty), Actualización global de feromonas: 0.1, Actualización local de feromonas: 0.1, Feromona inicial: 100, Tamaño de la población: 10. The 'Ejecutar' button is visible at the bottom right.

Figura 67. Error de parámetros. Manual de usuario. Elaboración propia.

Iniciada la ejecución del algoritmo seleccionado, podemos ver las soluciones que va encontrando, su coste y el conjunto de datos que está utilizando. Además, nos permite detener la ejecución del algoritmo en cualquier momento pulsando el botón correspondiente, como se muestra en la [Figura 68](#).

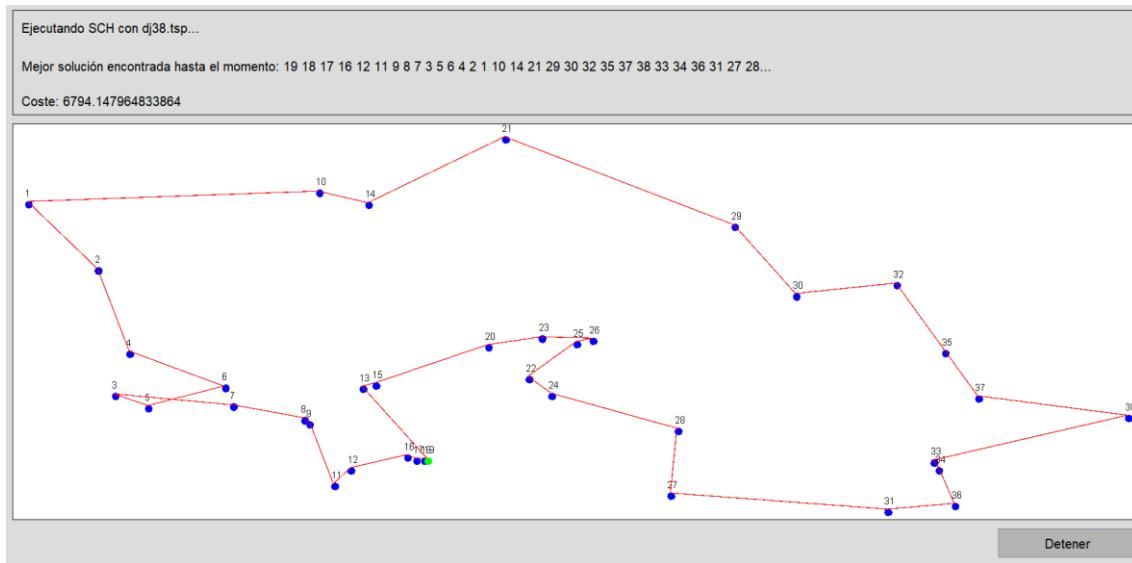


Figura 68. Pantalla de ejecución del algoritmo. Manual de usuario. Elaboración propia.

Durante la ejecución, debido a que en conjuntos de datos de tamaño grande no se puede presentar la solución completa en la interfaz, se muestra solo una parte de la misma. Si pulsamos en esta, se abrirá una ventana emergente donde se puede visualizar la solución completa, tal como se muestra en la [Figura 69](#).

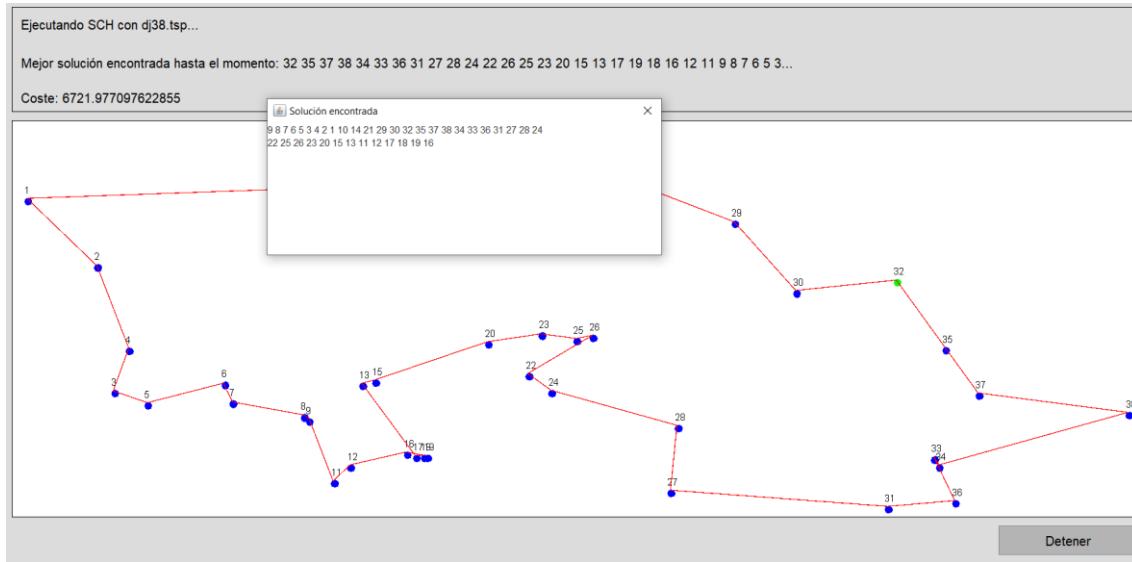


Figura 69. Pantalla emergente de la solución. Manual de usuario. Elaboración propia.

Una vez que finaliza o se detiene la ejecución del algoritmo, se visualiza la mejor solución encontrada y su coste. En este momento, se permite la exportación de los resultados a un archivo si se pulsa el botón correspondiente, como se aprecia en la [Figura 70](#).

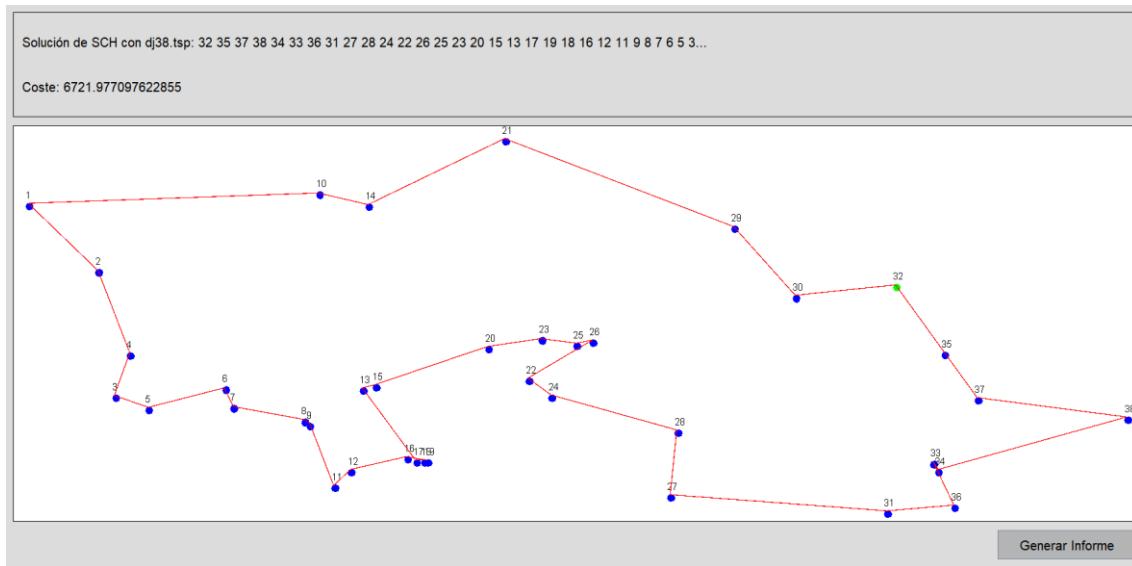


Figura 70. Pantalla de ejecución terminada. Manual de usuario. Elaboración propia.

De manera similar, en conjuntos de datos grandes no se puede mostrar la solución entera en la interfaz. Por lo tanto, se permite pulsar sobre ella para abrir una ventana emergente en la que se muestre al completo, como se detalla en la [Figura 71](#).

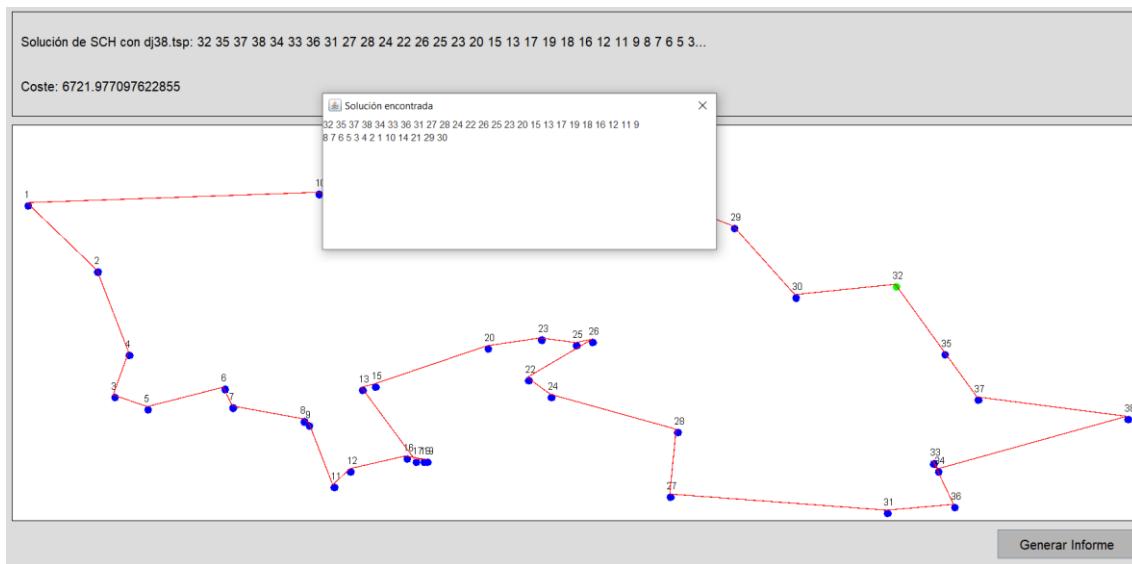


Figura 71. Pantalla emergente de la solución. Manual de usuario. Elaboración propia.

Cuando se pulsa la opción de generar informe, se abre una ventana emergente para guardar el archivo de resultados donde queramos y con el nombre que deseemos, tal como se ilustra en la [Figura 72](#). En el caso de que no queramos generar el informe, bastaría con cerrar la ventana y volveríamos a la pantalla principal.

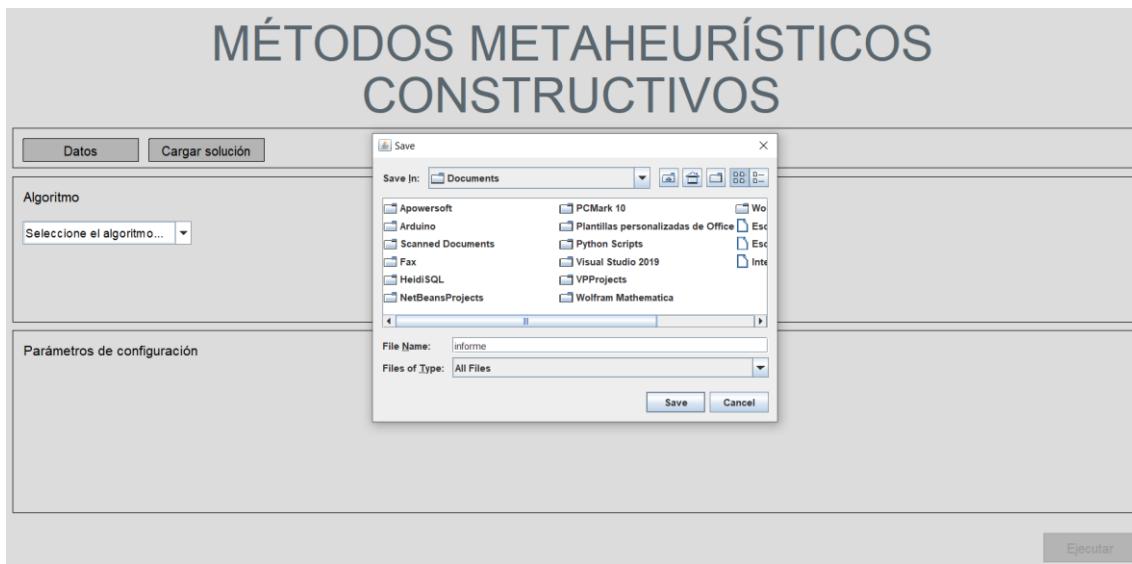


Figura 72. Exportación de resultados. Manual de usuario. Elaboración propia.

A continuación, en la [Figura 73](#), se muestra el contenido del informe de resultados generado.

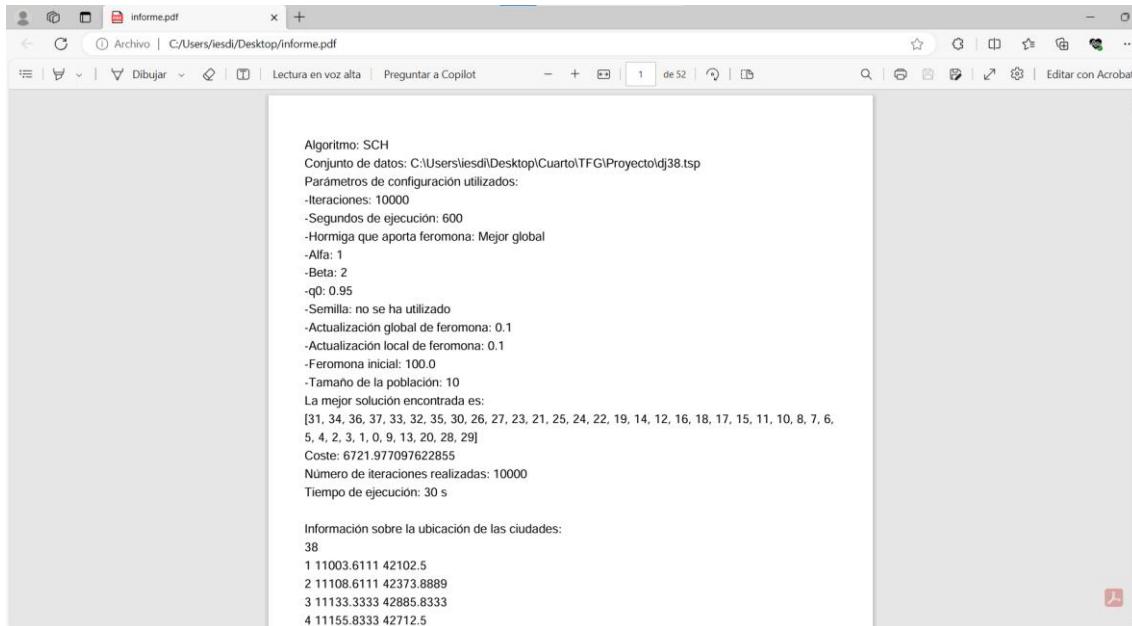


Figura 73. Fichero de resultados. Manual de usuario. Elaboración propia.

Una vez concluida la generación del informe de resultados, podemos visualizarlo nuevamente en la interfaz seleccionando la opción de cargar solución. En ese momento, se abrirá una ventana emergente para seleccionar el archivo con la solución, de acuerdo con la presentación de la [Figura 74](#).

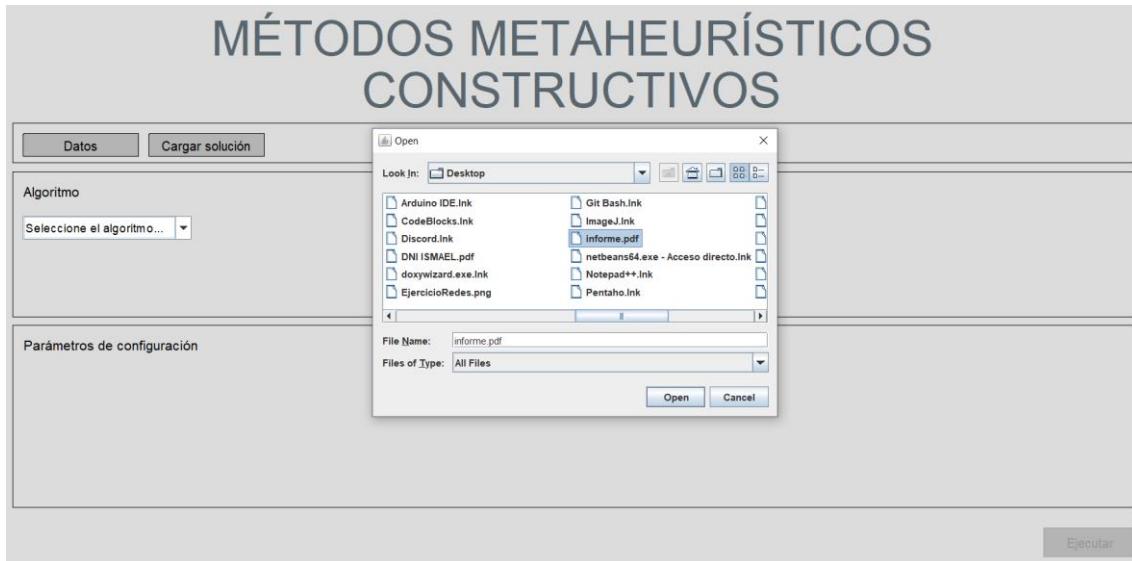


Figura 74. Importación del fichero de resultados. Manual de usuario. Elaboración propia.

En caso de que el fichero seleccionado no pertenezca a nuestro sistema, se le avisará al usuario para que seleccione un fichero válido, como se evidencia en la [Figura 75](#).

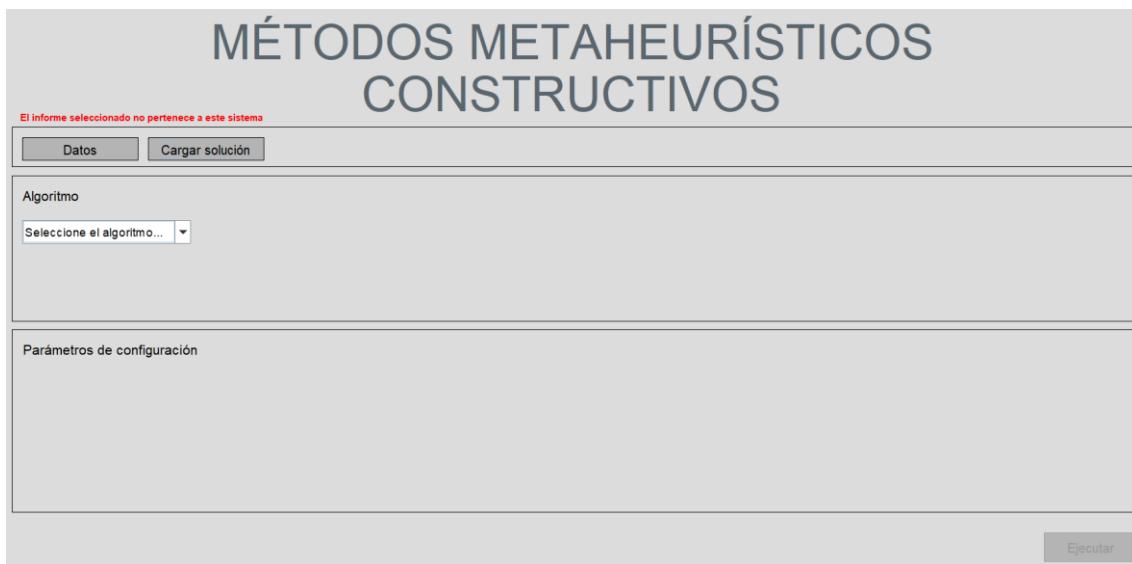


Figura 75. Error en el fichero de resultados. Manual de usuario. Elaboración propia.

Después de seleccionar un archivo correcto, se muestra el algoritmo que se ejecutó, el conjunto de datos utilizado, la solución y su coste. Además, permite al usuario regresar a la pantalla principal pulsando el botón correspondiente, como se presenta en la [Figura 76](#).

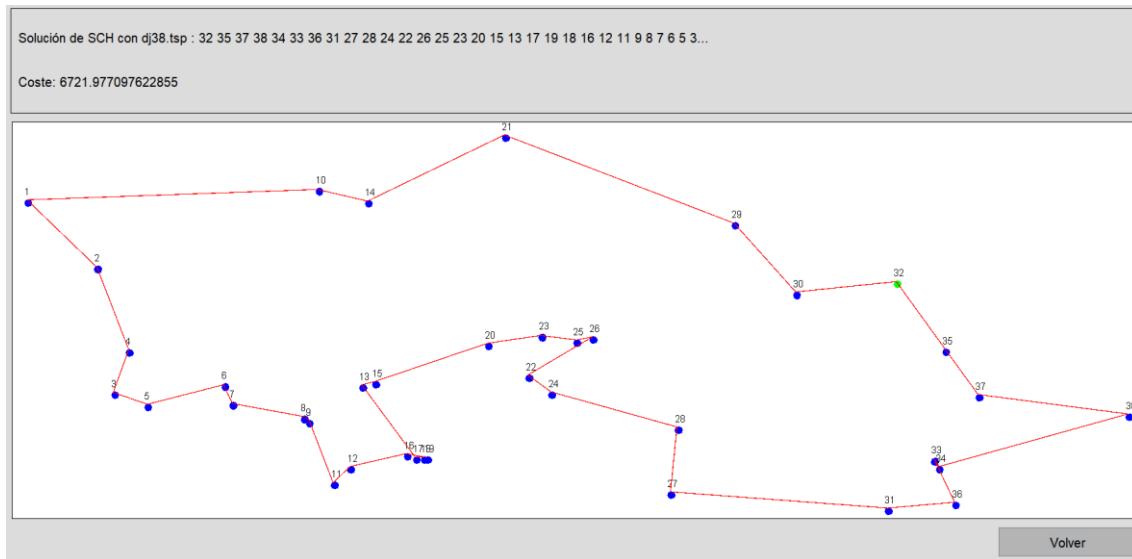


Figura 76. Pantalla de visualización del fichero de resultados. Manual de usuario. Elaboración propia.

Al pulsar el botón para volver, regresamos a la pantalla principal, como se muestra en la [Figura 77](#).

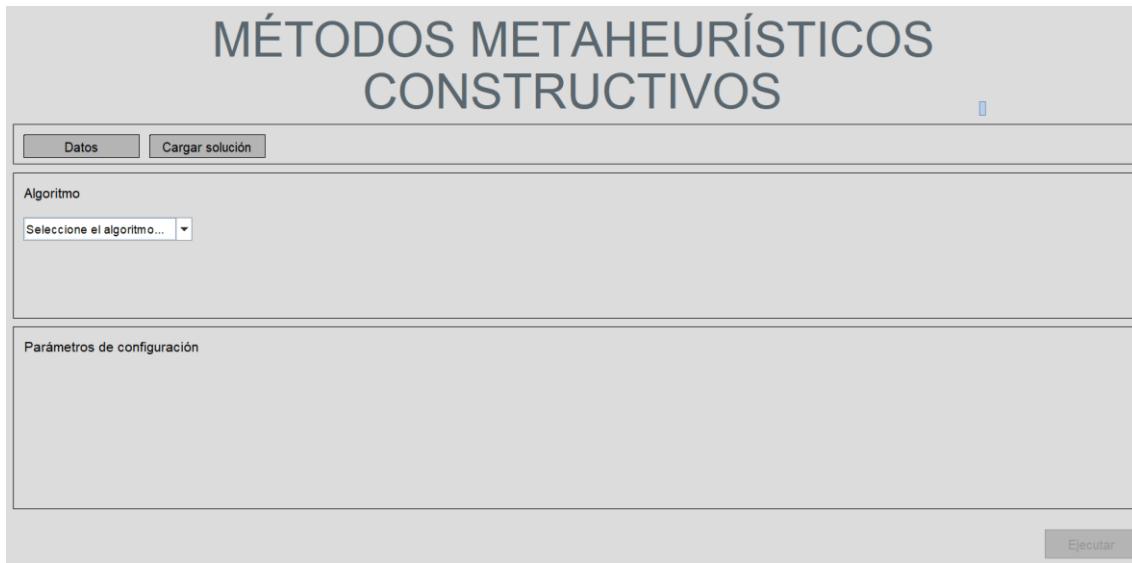


Figura 77. Pantalla inicial. Manual de usuario. Elaboración propia.

Finalmente, se proporciona un enlace al [repositorio](#) de GitHub que contiene el código de la aplicación. Además, se ha creado un [video tutorial](#) que demuestra y explica el uso del sistema.

BIBLIOGRAFÍA

- Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.
- Blum, C., & Roli, A. (2003). Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. *ACM Computing Surveys (CSUR)*, 35(3), 268-308.
- BOLIGRAFO BIC CRISTAL ORIGINAL PUNTA MEDIA. (2023, septiembre). *solocantidad.es*.

https://www.solocantidad.es/boligrafo-bic-cristal-original-punta-media?gclid=CjwKCAjw15eqBhBZEiwAbDomEvqnMX5Yj5hPwAf-hAFURaLVp-qOcFpcf97j_kOnGdkXPMw3fHBvoxoCQx0QAvD_BwE

Chin-Chuan, L., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*.

Cordón, O., Fernández de Viana, I., Herrera, F., & Moreno, L.I. (2000). A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System. *Actas de ANTS'2000*, 22-29.

¿Cuánto se gana como uno Gestor/a de proyectos en España? (2023, septiembre). *indeed.com*.

<https://es.indeed.com/career/gestor-de-proyectos/salaries>

Excel. (2023, septiembre). *microsoft.com*.

https://www.microsoft.com/es-es/microsoft-365/p/excel/cfq7ttc0hr4r?source=googleshopping&ef_id=_k_Cj0KCQjw-pyqBhDmARIsAKd9XIPIt_98NBs7gL92D9x32USxuYtfBy1NW1-mzWO24q9rlFUiLujNCzMcaAhf8EALw_wcB_k_&OCID=AIDcmm2iqf6ghp_SE_M_k_Cj0KCQjw-pyqBhDmARIsAKd9XIPIt_98NBs7gL92D9x32USxuYtfBy1NW1-mzWO24q9rlFUiLujNCzMcaAhf8EALw_wcB_k_&gad=1&gclid=Cj0KCQjw-pyqBhDmARIsAKd9XIPIt_98NBs7gL92D9x32USxuYtfBy1NW1-mzWO24q9rlFUiLujNCzMcaAhf8EALw_wcB&activetab=pivot:informaci%C3%B3ngeneral

Fibra 100 Mb. (2023, septiembre). *finetwork.com*.

<https://www.finetwork.com/tarifas-fibra/fibra-100mb>

Gambardella, L. M., & Dorigo, M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Trans. on Evolutionary Computation*, 1(1), 53-66.

Implementing Lin-Kernighan in Python. (2023, noviembre). *arthur.matheo.net*.

<https://arthur.matheo.net/implementing-lin-kernighan-in-python/>

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., & Shmoys, D. B. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Wiley.

Lenovo ThinkPad L15 Gen1 - 20U30017SP. (2023, septiembre). *tiendalenovo.es*

<https://www.tiendalenovo.es/lenovo-thinkpad-l15-gen1-20u30017sp>

MP – Folios Din A4 80gr, 500 Hojas, Papel Blanco Premium para Impresora Multifunción, para Uso de Oficina, Material Escolar, Paquete de impresión Multiusos (2023, septiembre). *amazon.es*.

https://www.amazon.es/MP-Fotografico-Impresora-Multifuncion-Papeleria/dp/B099GTM2QV/ref=asc_df_B099GTM2QV/?tag=googshopes-21&linkCode=df0&hvadid=542608507838&hvpos=&hvnetw=q&hvrand=18314936615769146798&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9049121&hvtargid=pla-1427651283998&psc=1

Papadimitriou, C. H., & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications.

Regla Eding. (2023, septiembre). *4pmarketing.es*.

https://www.4pmarketing.es/es/regla-eding---PP7049?gad=1&qclid=CjwKCAjw15eqBhBZEiwAbDomEvtbkJ8YpyyS-8VRwnP73llpqhg6HShc02OUZJE0CunVcDvPpXuFhoCvC4QAvD_BwE

Salario medio para Ingeniero informático en España 2024. (2024, enero). *talent.com*.

<https://es.talent.com/salary?job=ingeniero%2Binform%C3%A1tico>

Stützle, T., & Hoos, H. H. (2000). MAX-MIN Ant System. *Future Generation Computer Systems*, 16(8), 889-914.

The Traveling Salesman Problem (TSP). (2023, noviembre). *seas.gwu.edu*.

<https://www2.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html>

Toth, P., & Vigo, D. (2013). The Vehicle Routing Problem. SIAM Monographs on Discrete Mathematics and Applications.

Visual Paradigm Licensing Options. (2023, septiembre). *visual-paradigm.com*.

https://www.visual-paradigm.com/support/documents/vpuserguide/12/13/7572_licensing.html

Word Hogar y Estudiantes. (2023, septiembre). *microsoft.com*.

https://www.microsoft.com/es-es/microsoft-365/p/word-home-and-student/cfq7ttc0hlkl?source=googleshopping&ef_id=k_CjwKCAjw15eqBhBZ_EiwAbDomEvMTxaZ7KgEB3O3Y6_3dV6rPwu5oXBdrOtrpNuSz9IcSuE35bdVI_ixoC1Z8QAvD_BwE_k_&OCID=AIDcmm2iqf6ghp_SEM_k_CjwKCAjw15eqB_hBZEiwAbDomEvMTxaZ7KgEB3O3Y6_3dV6rPwu5oXBdrOtrpNuSz9IcSuE35_bdVlixoC1Z8QAvD_BwE_k_&qad=1&qclid=CjwKCAjw15eqBhBZ_EiwAbDomEvMTxaZ7KgEB3O3Y6_3dV6rPwu5oXBdrOtrpNuSz9IcSuE35bdVlixoC1Z8QA_vD_BwE&activetab=pivot:informaci%C3%B3nngeneraltab