Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado
Facultad de Ciencias Matemáticas
Unidad de Posgrado

Modificaciones del algoritmo sacrificio cortoplacista adaptativo 2OPT_R para solucionar el problema del agente viajero simétrico: comparación del nivel de eficacia

TESIS

Para optar el Grado Académico de Magíster en Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial

AUTOR

David Jonathan ASTOQUILLCA YARANGA

ASESOR

Dra. Esther BERGER VIDAL

Lima, Perú

2025



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Astoquillca, D. (2025). *Modificaciones del algoritmo sacrificio cortoplacista adaptativo 2OPT_R para solucionar el problema del agente viajero simétrico: comparación del nivel de eficacia*. [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	David Jonathan Astoquillca Yaranga
Tipo de documento de identidad	DNI
Número de documento de identidad	44430848
URL de ORCID	https://orcid.org/0009-0001-7027-7238
Datos de asesor	
Nombres y apellidos	Esther Berger Vidal
Tipo de documento de identidad	DNI
Número de documento de identidad	08766040
URL de ORCID	https://orcid.org/0000-0001-5282-6793
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	María del Pilar Álvarez Rivas
Tipo de documento	DNI
Número de documento de identidad	06445832
Miembro del jurado 1	
Nombres y apellidos	Flor Cagniy Cárdenas Mariño
Tipo de documento	DNI
Número de documento de identidad	45422233
Miembro del jurado 2	
Nombres y apellidos	Gabriel José Solari Carbajal
Tipo de documento	DNI
Número de documento de identidad	25471272
Miembro del jurado 3	
Nombres y apellidos	Luis Gonzalo Acosta Espejo
Tipo de documento	DNI

Número de documento de identidad	09437003
Datos de investigación	
Línea de investigación	A.3.3.1. Optimización Matemática
Grupo de investigación	Dinámica Ambiental - DINAMB
Agencia de financiamiento	Sin financiamiento
Ubicación geográfica de la investigación	Universidad Nacional Mayor de San Marcos País: Perú Departamento: Lima Provincia: Lima Distrito: Lima Av. Carlos Germán Amezaga #375 - Cercado de Lima Coordenadas geográficas Latitud: -12.0603753 Longitud: -77.0820784
Año o rango de años en que se realizó la investigación	2021 - 2023
URL de disciplinas OCDE	Matemáticas aplicadas https://purl.org/pe-repo/ocde/ford#1.01.02

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú, DECANA DE AMÉRICA

FACULTAD DE CIENCIAS MATEMÁTICAS
VICEDECANATO DE INVESTIGACIÓN Y POSGRADO
UNIDAD DE POSGRADO

ACTA DE SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGISTER

Siendo las .14:00 horas del día jueves dieciocho de diciembre de dos mil veinticinco, en el Auditorio de la Facultad de Ciencias Matemáticas de la Universidad Nacional Mayor de San Marcos, el Jurado Examinador de tesis, presidido por la Dra. María Del Pilar Álvarez Rivas e integrado por los siguientes miembros: Dra. Flor Cagniy Cárdenas Mariño (Jurado Informante), Mg. Gabriel José Solari Carbajal (Jurado Evaluador), Dr. Luis Gonzalo Acosta Espejo (Jurado Evaluador) y la Dra. Esther Berger Vidal como Miembro Asesora; se reunieron para la sustentación de la tesis titulada: **«Modificaciones del algoritmo sacrificio cortoplacista adaptativo 2OPT_R para solucionar el problema del agente viajero simétrico: comparación del nivel de eficacia»**, presentada por el Br. David Jonathan Astoquillca Yaranga, para optar el Grado Académico de Magíster en Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial.

Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, según tabla adjunta, habiendo obtenido el Bachiller **DAVID JONATHAN ASTOQUILLCA YARANGA**, el calificativo de ....18. MUY. BUENO.....

Habiendo sido aprobada la sustentación de la Tesis, el Jurado Examinador recomienda que el Consejo de Facultad apruebe el otorgamiento del **Grado Académico de Magister en Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial** al Bachiller **DAVID JONATHAN ASTOQUILLCA YARANGA**.

Siendo las .15:45. horas, se levantó la sesión, firmando para constancia la presente Acta:

Dra. María Del Pilar Álvarez Rivas
Presidente

Dra. Flor Cagniy Cárdenas Mariño
Miembro

Mg. Gabriel José Solari Carbajal
Miembro

Dr. Luis Gonzalo Acosta Espejo
Miembro

Dra. Esther Berger Vidal
Miembro Asesora

UNIVERSIDAD NACIONAL MAYOR DE
SAN MARCOS
Universidad del Perú, Decana de América

CERTIFICADO DE SIMILITUD

Yo, **Esther Berger Vidal** en mi condición de asesora acreditado con la Resolución Decanal N° 000075-2022-D-FCM/UNMSM de fecha 25 de enero de 2022, cambio de título Resolución Decanal N° 000035-2023-D-FCM/UNMSM, de fecha 23 de enero de 2023 y N° 000679-2024-D-FCM/UNMSM, de fecha 22 de abril de 2024, cuyo título es: "MODIFICACIONES DEL ALGORITMO SACRIFICIO CORTOPLACISTA ADAPTATIVO 2OPT_R PARA SOLUCIONAR EL PROBLEMA DEL AGENTE VIAJERO SIMÉTRICO: COMPARACIÓN DEL NIVEL DE EFICACIA", presentado por el Br. **DAVID JONATHAN ASTOQUILLCA YARANGA**, para optar el Grado Académico de Magíster en *Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial*.

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante la plataforma PIDS, el documento evaluado cuenta con el **7**% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su publicación en el repositorio institucional.

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado académico correspondiente.

Lima, 30 de octubre de 2024



Firma del Asesora

Nombres y Apellidos del asesor: Esther Berger Vidal

DNI: 08766040

FICHA CATALOGRÁFICA

ASTOQUILLCA YARANGA, DAVID JONATHAN

Modificaciones del algoritmo Sacrificio Cortoplacista Adaptativo 2Opt_r para solucionar el Problema del Agente Viajero Simétrico: Comparación del Nivel de Eficacia.
Lima, 2025.
xviii, 289p., 29.7 cm., (UNMSM, Magíster en Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial, 2025).
Tesis, Universidad Nacional Mayor de San Marcos
Unidad de Posgrado
Facultad de Ciencias Matemáticas
UNMSM / FdeCM

Dedicatoria

A mis padres, Julia Yaranga Risco y Justo Astoquillca Cárdenas, por la oportunidad que me dieron de seguir estudios universitarios en la Universidad Nacional Mayor de San Marcos, por darme un hogar y el cariño que siempre estará conmigo.

Agradecimientos

De igual manera a mis padres, por el sacrificio y la paciencia que tuvieron conmigo, a mis profesores de la Universidad Nacional Mayor de San Marcos, a mi hermano Yoel Astoquillca por su paciencia conmigo y en especial a mi asesora por brindarme su apoyo y guía durante el desarrollo de la presente tesis.

INDICE

FICHA CATALOGRÁFICA ..... **ii**
Dedicatoria ..... **iii**
Agradecimientos ..... **iv**
INDICE ..... **v**
Lista de Tablas ..... **x**
Lista de Figuras ..... **xiii**
RESUMEN ..... **xvii**
ABSTRACT ..... **xviii**
Introducción **1**
CAPÍTULO I PLANTEAMIENTO DEL PROBLEMA **3**

1.1. Situación Problemática 3
1.2. Formulación del Problema 5
1.2.1. Problemas General *5*
1.2.2. Problemas Específicos *6*
1.3. Justificación de la Investigación 6
1.3.1. Justificación Teórica *6*
1.3.2. Justificación Práctica *6*
1.4. Objetivos de la Investigación 7
1.4.1. Objetivo General *7*
1.4.2. Objetivos Específicos *7*
1.5. Hipótesis de la Investigación 8
1.5.1. Hipótesis General *8*
1.5.2. Hipótesis Específicas *8*
1.6. Limitaciones del estudio 8

CAPÍTULO II MARCO TEÓRICO **10**

2.1. Antecedentes de la Investigación 11

2.1.1. Antecedentes Nacionales 11

2.1.2. Antecedentes Internacionales 12

2.2. Bases Teóricas 16

2.2.1. Pensamiento Sistémico 16

2.2.2. El Juego de la Cerveza en el libro La Quinta Disciplina 19

2.2.3. Lecciones del Juego de la Cerveza en el libro La Quinta Disciplina 21

2.2.4. Problema del Agente Viajero 23

2.2.5. Definición de grafo 24

2.2.6. Conceptos básicos sobre grafos 25

2.2.7. Definición de Instancia 26

2.2.8. Representación del Problema del Agente Viajero 27

2.2.9. Tipos de Problemas del Agente Viajero 29

2.2.10. Métodos de solución del Problema del Agente Viajero 29

2.2.11. Tipos de búsqueda de los algoritmos metaheurísticos 30

2.2.12. Heurística del Vecino más Cercano (VMC). 32

2.2.13. Heurística de mejora de Intercambio Bidireccional (2Opt). 33

2.2.14. Ejemplo de Heurística 2Opt con política Best Improvement 37

2.2.15. Sacrificio Cortoplacista Adaptativo (SCA) 43

2.2.16. Ejemplo de Sacrificio Cortoplacista Adaptativo (SCA) 45

2.2.17. Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt) 47

2.2.18. Ejemplo de Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt) 49

2.2.19. Factor "r" en el Sacrificio Cortoplacista Adaptativo 2Opt 55

2.2.20. Eficacia Algorítmica 59

2.2.21. Eficiencia Algorítmica medida en base a su Complejidad 60

2.2.22. Tipos de Problemas 66

2.2.23. Definiciones previas para el algoritmo a proponer 67

2.3. Marco Conceptual....79

CAPÍTULO III METODOLOGÍA....82

3.1. Introducción....82

3.2. Tipo de investigación....83

3.3. Diseño de investigación....83

3.4. Nivel de investigación....84

3.5. Enfoque de investigación....84

3.6. Identificación de variables....84

3.6.1. Variable independiente....84

3.6.2. Variable dependiente....84

3.7. Unidad de análisis....84

3.8. Población del estudio....85

3.9. Tamaño de muestra....85

3.10. Selección de la muestra....85

3.11. Técnicas e instrumentos de recolección de datos....86

3.12. Procedimiento para la construcción de la matriz de pesos....86

3.13. Procedimiento para el cálculo del nivel de eficacia....88

3.14. Método....92

3.15. Descripción del proceso de investigación tecnológica....93

3.15.1. Observar....93

3.15.2. Determinar el problema del objeto de estudio....94

3.15.3. Documentar....95

3.15.4. Análisis de la estructura del SCA_2Opt....95

3.15.5. Análisis del Diseño del Algoritmo....97

3.15.6. Implementación de los algoritmos desarrollados....105

3.16. Algoritmos desarrollados en su versión Heavy....105

3.16.1. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Multi2Opt: SCA(1)_Multi2Opt 105

3.16.2. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Multi2Opt_r: SCA(1)_Multi2Opt_r 110

3.16.3. Sacrificio Cortoplacista Adaptativo de amplitud A, con criterio Multi2Opt: SCA(A)_Multi2Opt 118

3.17. Complejidad algorítmica de las versiones Heavy 126

3.18. Algoritmos desarrollados en su versión Soft 127

3.18.1. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Uni2Opt: SCA(1)_Uni2Opt 128

3.18.2. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Uni2Opt_r: SCA(1)_Uni2Opt_r 131

3.18.3. Sacrificio Cortoplacista Adaptativo de amplitud A, con criterio Uni2Opt: algoritmo SCA(A)_Uni2Opt 134

3.19. Complejidad algorítmica de las versiones Soft..... 139

CAPÍTULO IV RESULTADOS Y DISCUSIÓN..... 140

4.1. Presentación de Resultados 143

4.1.1. Selección de instancias de librería TSP para medir la Eficacia ... 144

4.1.2. Eficacia y Nivel de Eficacia de las versiones sin factor "r"..... 146

4.1.3. Eficacia y Nivel de Eficacia de las versiones con factor "r" 149

4.1.4. Eficacia y Nivel de Eficacia de versiones Heavy y versiones Soft vs algoritmos de Pérez (2011)..... 152

4.1.5. Variación del error relativo de versiones Heavy y Soft vs algoritmos de Pérez (2011) 158

4.1.6. Eficacia de versiones Heavy y Soft vs metaheurísticas seleccionadas 159

4.1.7. Nivel de eficacia de versiones Heavy y Soft vs metaheurísticas seleccionadas 161

4.1.8. Variación del error relativo según algoritmo metaheurístico 169

4.2. Validación de Hipótesis 170

4.2.1. Hipótesis General 170

4.2.2. Hipótesis Específica 1 172

4.2.3. Hipótesis Específica 2 174

4.2.4. Hipótesis Específica 3 177

4.3. Discusión de Resultados 181

CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES 185

5.1. CONCLUSIONES 185

5.2. RECOMENDACIONES 187

REFERENCIAS 188

ANEXOS 197

Lista de Tablas

Tabla 1 *Cantidad de investigaciones en Scopus 2014-2023* 5

Tabla 2 *Costo de ruta según el número de intercambio 2OptBI realizado según "r".*.....57

Tabla 3 *Descripción de órdenes de complejidad*..... 64

Tabla 4 *Determinación de rango de "r" para versiones con factor "r"*..... 117

Tabla 5 *Determinación de la "Amplitud" del Sacrificio Cortoplacista a utilizar* 123

Tabla 6 *Determinación del número de ejecuciones 2Opt del Multi2Opt*..... 125

Tabla 7 *Categorización del nivel de eficacia*..... 143

Tabla 8 *Solución y error relativo de algoritmos SCA_2Opt de Pérez (2011), SCA(1)_Uni2Opt y SCA(1)_Multi2Opt según instancia TSPLIB*. 147

Tabla 9 *Solución y error relativo de algoritmos SCA_2Opt_r de Pérez (2011), SCA(1)_Uni2Opt_r y SCA(1)_Multi2Opt_r según instancia*..... 150

Tabla 10 *Soluciones de las versiones Heavy, Soft y de Pérez (2011)*..... 153

Tabla 11 *Error relativo de las versiones Heavy, Soft y de Pérez (2011)*.... 154

Tabla 12 *Rango del error relativo de las versiones Heavy, Soft y de Pérez(2011)* 155

Tabla 13 *Nivel de eficacia promedio: versiones Heavy, Soft y de Pérez (2011)* 156

Tabla 14 *Ranking del nivel de eficacia: versiones Heavy, Soft y de Pérez(2011)* 157

Tabla 15 *Solución y error relativo de algoritmos metaheurísticos seleccionados vs versiones Soft y Heavy según instancia TSPLIB* 160

Tabla 16 *Mejor nivel de eficacia: versiones Heavy y metaheurísticas seleccionadas* 167

Tabla 17 *Mejor nivel de eficacia: versiones Soft y metaheurísticas seleccionadas* 168

Tabla 18 *Intervalo del error relativo por metaheurística (mínimo - máximo)* 178

Tabla 19 *Nivel de eficacia promedio de versiones Heavy, Soft y de los algoritmos metaheurísticos seleccionados*..... 179

Tabla 20 *Error relativo de versiones Heavy, Soft y metaheurística ACO*... 181

Tabla 21 *Nivel de eficacia entre las versiones Heavy, Soft y metaheurísticas ACO, VTSPO, ABCSS y DSMO* 184

Tabla 22 *Soluciones de versiones Heavy, Soft, Solver(Simplex), WinQSB(Branch and Bound), Vecino más Cercano, GRASP y Tabú Search en instancias pequeñas*..... 210

Tabla 23 *Escala de color mejor solución disponible: versiones Heavy y Soft; metaheurísticas Scatter, GRASP y Tabú Search para instancias medianas* 211

Tabla 24 *Escala de color de mejor solución disponible versiones Heavy, Soft vs metaheurísticas ACO, VTSPO, ABCSS y DSMO*..... 214

Tabla 25 *Detalle de las 65 instancias simétricas TSPLIB seleccionadas*... 243

Tabla 26 *Escala de color de mejor solución disponible según número de ejecuciones 2opt para el Multi2Opt* 245

Tabla 27 *Cantidad mínima de ejecuciones 2Opt para el Multi2Opt* 248

Tabla 28 *Determinación de constante "k" para cota de # de ejecuciones 2Opt en el Multi2Opt* 249

Tabla 29 *Escala de color de la mejor solución disponible: VMC,VMC_SCA, VMC_2Opt, versiones Soft y Heavy de 65 instancias TSPLIB* ... 253

Tabla 30 *Escala de color del Error Relativo: VMC,VMC_SCA,VMC_2Opt, versiones Soft y Heavy de 65 instancias simétricas TSPLIB* 255

Tabla 31 *Estadísticos para describir al error relativo según algoritmo Soft, Heavy, Vecino más Cercano (VMC), VMC_SCA, VMC_2Opt* 258

Tabla 32 *Tiempo de ejecución de algoritmos según instancia TSPLIB* 261

Tabla 33 *Categorización de algoritmos según tiempo y orden del grafo* 263

Tabla 34 *Puntos de apalancamiento según algoritmo SCA(1)_Uni2Opt y SCA(1)_Multi2Opt* 270

Tabla 35 *Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = n^k* 275

Tabla 36 *Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = $n^k \times log(n)$* 278

Tabla 37 *Vectores Prohibición de la mejor solución disponible para el SCA(5)_Multi2Opt según instancia simétrica TSPLIB* 283

Tabla 38 *Determinación de la amplitud del sacrificio cortoplacista adaptativo a utilizar* 287

Lista de Figuras

Figura 1 *Ejemplo de representación gráfica de un grafo* 28

Figura 2 *Ejemplo de representación matricial de un grafo*..... 28

Figura 3 *Búsqueda basada en la trayectoria y basada en la población*..... 31

Figura 4 *Grafo de aplicación de heurística de intercambio 2Opt* 33

Figura 5 *Ejemplo de actualización de indicador Dmax* 35

Figura 6 *Diagrama de flujo de la heurística 2OptBI para una ejecución* 37

Figura 7 *Matriz de costos para ejemplo de grafo con ocho ciudades*..... 38

Figura 8 *Evolución de solución según número de ejecuciones 2OptBI* 43

Figura 9 *Funcionamiento del Sacrificio Cortoplacista Adaptativo (SCA)* 46

Figura 10 *Diagrama de flujo del SCA_2Opt*..... 48

Figura 11 *Criterios de decisión en el SCA_2Opt* 49

Figura 12 *Ruta determinada por la heurística del "Vecino más Cercano"*.....49

Figura 13 *Ruta para el punto de apalancamiento en posición 1*..... 50

Figura 14 *Ruta para el punto de apalancamiento en posición 2*..... 51

Figura 15 *Ruta para el punto de apalancamiento en posición 3*..... 52

Figura 16 *Ruta para el punto de apalancamiento en posición 4*..... 53

Figura 17 *Ruta para el punto de apalancamiento en posición 5*..... 53

Figura 18 *Ruta para el punto de apalancamiento en posición 6*..... 54

Figura 19 *Ejemplo de factor "r" incorporado al SCA_2Opt*.....56

Figura 20 *Efecto del factor "r" en el SCA_2OptBI*.....58

Figura 21 *Grafica del concepto de "Amplitud"*.....67

Figura 22 *Arcos prohibidos en secuencia de amplitud 3 para posición 2*.... 68

Figura 23 *Sacrificio Cortoplacista Adaptativo de amplitud 1 [SCA(1)]* 69

Figura 24 *Sacrificio Cortoplacista Adaptativo de amplitud 3 [SCA(3)],* 69

Figura 25 *Funcionamiento del criterio Uni2Opt con el SCA(A)*..... 70

Figura 26 *Diagrama de flujo del criterio Uni2Opt con el SCA(A)* 71

Figura 27 *Funcionamiento del criterio Uni2Opt_r con el SCA(A)*..... 72

Figura 28 *Diagrama de flujo del Uni2Opt_r con el SCA(A)*..... 73

Figura 29 *Funcionamiento del criterio Multi2Opt con el SCA(A)*..... 74

Figura 30 *Diagrama de flujo de Multi2Opt dentro del SCA(A)_Multi2Opt* 75

Figura 31 *Funcionamiento del criterio Multi2Opt_r con el SCA(A)*..... 76

Figura 32 *Diagrama de flujo Multi2Opt_r dentro del SCA(A)_Multi2Opt_r*... 77

Figura 33 *Vector Prohibición para sacrificio cortoplacista de amplitud 3*..... 78

Figura 34 *Estructura de búsqueda del Vecino más Cercano con Sacrificio Cortoplacista Adaptativo (VMC_SCA)*..... 96

Figura 35 *Solución con política First Improvement y Best Imprevement* ... 100

Figura 36 *Solución por "mejora constante" vs "empeorar para mejorar"*.... 102

Figura 37 *Diagrama de bloques del SCA(1)_Multi2Opt*..... 107

Figura 38 *Funcionamiento del SCA(1)_Multi2Opt* 107

Figura 39 *Diagrama de flujo del SCA(1)_Multi2Opt*..... 108

Figura 40 *Comportamiento de la convergencia del SCA(1)_Multi2Opt*..... 109

Figura 41 *Diagrama de bloques del SCA(1)_Multi2Opt_r*..... 110

Figura 42 *Funcionamiento del SCA(1)_Multi2Opt_r* 111

Figura 43 *Diagrama de flujo del SCA(1)_Multi2Opt_r*..... 111

Figura 44 *Influencia del factor "r" en el SCA(1)_Multi2Opt_r* 113

Figura 45 *Determinación de rango de valores de "r"*..... 115

Figura 46 *Diagrama de bloques del SCA(A)_Multi2Opt*..... 119

Figura 47 *Funcionamiento del SCA(A)_Multi2Opt* 120

Figura 48 *Diagrama de flujo del SCA(A)_Multi2Opt*..... 122

Figura 49 *Diagrama de bloques del SCA(1)_Uni2Opt* 129

Figura 50 *Funcionamiento del SCA(1)_Uni2Opt*..... 129

Figura 51 *Diagrama de flujo del SCA(1)_Uni2Opt* 130

Figura 52 *Diagrama de bloques del SCA(1)_Uni2Opt_r* 132

Figura 53 *Funcionamiento del SCA(1)_Uni2Opt_r*..... 132

Figura 54 *Diagrama de flujo del SCA(1)_Uni2Opt_r* 133

Figura 55 *Diagrama de bloques del SCA(A)_Uni2Opt*..... 134

Figura 56 *Funcionamiento del SCA(A)_Uni2Opt* 135

Figura 57 *Diagrama de flujo del SCA(A)_Uni2Opt*..... 137

Figura 58 *Resumen de los algoritmos desarrollados*..... 140

Figura 59 *Síntesis del flujograma del SCA_2Opt(_r) de Pérez(2011)*..... 141

Figura 60 *Síntesis del flujograma de las versiones Heavy* 142

Figura 61 *Síntesis del flujograma de las versiones Soft* 142

Figura 62 *Gráfica de puntos del error relativo: versiones Heavy, Soft y de Pérez(2011)* 158

Figura 63 *Error relativo de versiones Heavy vs Algoritmos Genéticos* 164

Figura 64 *Error relativo de versiones Heavy vs System Colony Ant*..... 165

Figura 65 *Error relativo de versiones Heavy vs Programación Evolutiva*... 166

Figura 66 *Gráfica de puntos del error relativo: versión Heavy, Soft y metaheurísticas*..... 169

Figura 67 *Gráfica del error relativo de versiones Heavy y el SCA_2Opt* ... 172

Figura 68 *Gráfica del error relativo de versiones Soft y el SCA_2Opt*..... 173

Figura 69 *Gráfica del error relativo de versiones Heavy y el SCA_2Opt_r* 175

Figura 70 *Gráfica del error relativo de versiones Soft y el SCA_2Opt_r*.... 176

Figura 71 *Grupo de Algoritmos Genéticos tomados de Pérez (2011)* 201

Figura 72 *Grupo de Colonia de Hormigas tomados de Pérez (2011)*..... 202

Figura 73 *Grupo de Enfriamiento Simulado tomados de Pérez (2011)* 203

Figura 74 *Grupo de Otras Metaheurísticas tomadas de Pérez (2011)* 204

Figura 75 *Constante "k" para cota del número de ejecuciones 2Opt en el Multi2Opt*..... 251

Figura 76 *Gráfica de puntos del error relativo: VMC, VMC_SCA, VMC_2Opt, versiones Soft y Heavy* 257

Figura 77 *Agrupación del total de Vectores Prohibición para el SCA(5)*.... 265

Figura 78 *Comportamiento del Vector Prohibición para el SCA(3)*..... 267

Figura 79 *Funciones f(n) que acotan superiormente al VMC_2Opt*..... 281

Figura 80 *Nivel de eficacia por cada valor de amplitud(A) evaluado* 287

Figura 81 *Diagrama de flujo para el algoritmo VMC_2Opt_r* 289

RESUMEN

Modificaciones del algoritmo Sacrificio Cortoplacista Adaptativo 2Opt_r para solucionar el Problema del Agente Viajero Simétrico: Comparación del Nivel de Eficacia

DAVID JONATHAN ASTOQUILLCA YARANGA

2025

Asesora : Dra. Esther Berger Vidal

Grado obtenido: Magister en Investigación de Operaciones y Sistemas con mención en Optimización de Sistemas de Gerencia Empresarial

---

El presente trabajo se desarrolló, tomando en cuenta la escasa cantidad de investigaciones para el mejoramiento de una heurística considerada miope, como el algoritmo del Vecino más Cercano, observándose un mayor desarrollo de investigaciones enfocadas a las metaheurísticas. Razón por la cual, en esta investigación se buscó un diseño que logre mejorar el Nivel de Eficacia del algoritmo heurístico del Vecino más Cercano. Por ello, se tomó como base el enfoque del Sacrificio Cortoplacista Adaptativo, desarrollado por Pérez en 2011, cuya combinación con la heurística de mejoramiento 2Opt_r fue denominada por Pérez (2011) como SCA_2Opt_r. A partir de dicha heurística, se diseñaron las versiones Heavy y Soft en base a analogías con el análisis sistémico, logrando mostrar un Nivel de Eficacia de 100% para las versiones Heavy y entre el 31% al 85% según la versión Soft analizada, en comparación con el SCA_2Opt, SCA_2Opt_r. Posteriormente, se comparó con metaheurísticas de artículos internacionales consolidadas en la investigación de Pérez (2011), logrando alcanzar un Nivel de Eficacia competitivo, según la metaheurística a comparar.

Palabras clave: Problema del Agente Viajero Simétrico, Heurística del Vecino más Cercano, Sacrificio Cortoplacista Adaptativo, Heurística 2Opt_r

ABSTRACT

Modifications of the Adaptive Short-Term Sacrifice 2Opt_r Algorithm to Solve the Symmetric Traveling Salesman Problem: Comparison of the Level of Effectiveness

DAVID JONATHAN, ASTOQUILLCA YARANGA

2025

Guiding : Dra. Esther Berger Vidal

Obtained degree : Master's in operations research and systems with a specialization in Business Management Systems Optimization

---

The present work was developed considering the scarce amount of research for the improvement of a heuristic considered myopic such as the Nearest Neighbor algorithm, observing a greater development of research focused on metaheuristics. For this reason, in this research, a design was sought that would improve the Level of Effectiveness of the heuristic algorithm of the Nearest Neighbor. Therefore, the Adaptive Short-Sighted Sacrifice approach developed by Pérez in 2011 was taken as a basis, whose combination with the 2Opt improvement heuristic was called SCA_2Opt by Pérez (2011). From this heuristic, the Heavy and Soft versions were designed based on analogies with systemic analysis, achieving a level of effectiveness of 100% for Heavy versions and between 31% to 85% depending on the Soft version analyzed in comparison with SCA_2Opt, SCA_2Opt_r. Later, it was compared with consolidated international metaheuristics in Pérez's (2011) research, achieving a competitive level of effectiveness according to the metaheuristic being compared.

Keywords: Traveling Salesman Problem Symmetric, Nearest Neighbor Heuristic, Short-Term Adaptive Sacrifice, 2Opt Heuristic.

Introducción

Durante los últimos años, se ha visto un mayor desarrollo de los denominados algoritmos metaheurísticos. Esto se debe a que cuentan con mecanismos que evitan quedar atrapados en óptimos locales, a diferencia de los algoritmos heurísticos, que sí pueden presentar este problema. Sin embargo, las metaheurísticas presentan un mayor tiempo tanto de diseño como de experimentación; ese es el caso de los algoritmos genéticos, colonia de hormigas, redes neuronales, entre otros. Por otra parte, no se evidencia, en general, una cantidad elevada de investigaciones cuyo eje central sea la mejora de los algoritmos heurísticos para convertirlos en metaheurísticos y, en particular, de la heurística del Vecino más Cercano.

En la revisión de la literatura, se encontró a Pérez (2011), quien desarrolló un algoritmo basado en lo que denominó Sacrificio Cortoplacista, cuyo criterio se inspira en el pensamiento sistémico, campo muy poco explorado como fuente de inspiración para el desarrollo de heurísticas o metaheurísticas. El Sacrificio Cortoplacista consiste en, desde un nodo específico, sacrificar; es decir, evitar ir al nodo más cercano para elegir ir a la segunda mejor opción, con la esperanza de que dicho sacrificio permita, a mediano y largo plazo, encontrar mejores soluciones mientras el sistema se estabiliza. A tal solución, Pérez (2011) le aplicó la heurística de 2 Intercambio o 2Opt, y denominó a la combinación de tales criterios como SCA_2opt.

En esta investigación, se tomó como punto de partida el algoritmo denominado SCA_2Opt para, en base a analogías con el pensamiento sistémico, modificar su criterio de búsqueda, de tal manera que se explore con mayor amplitud el espacio de soluciones factibles y, por tanto, mejore la calidad de las soluciones obtenidas por nuestro algoritmo propuesto.

La tesis desarrollada en la presente investigación tiene la siguiente estructura:

Capítulo 1, se describió la situación problemática, la cual nos conduce al planteamiento del problema abordado, delimitando los objetivos de la investigación, alcance y justificación.

Capítulo 2, se desarrolló el Marco Teórico, mostrándose investigaciones tanto nacionales como internacionales que abordaron problemáticas similares o que contribuyeron con ideas para la construcción de nuestro algoritmo. También se presentaron conceptos previos, los cuales formaron parte de las bases teóricas, de términos relacionados con la investigación, así como la descripción de los algoritmos heurísticos usados y estructuras del pensamiento sistémico, las cuales se usaron en el Capítulo 3.

Capítulo 3, se explicó una metodología adaptada para una investigación tecnológica, que está orientada a la descripción del proceso de diseño del algoritmo híbrido elaborado en la investigación. Se definió el tipo y diseño de investigación, así como el enfoque usado para el análisis de las variables, objeto de estudio, muestra y población. También se definieron los instrumentos para analizar la muestra seleccionada.

Capítulo 4, se analizaron los resultados de los algoritmos desarrollados en la presente investigación, comparándolos con los resultados de nueve algoritmos: dos algoritmos propuestos en la investigación de Pérez (2011) y siete metaheurísticas de otras investigaciones, para comparar el nivel de eficacia algorítmica en la resolución de problemas para instancias simétricas de la librería TSPLIB.

Capítulo 5, con base en los resultados obtenidos en el capítulo 4, se verificó si se alcanzaron o no los objetivos de la investigación.

Posteriormente, se plantearon preguntas para futuras investigaciones, así como la presentación de las Conclusiones y Recomendaciones.

Finalmente, se presentó la relación de Referencias y los Anexos que complementan la información de la investigación.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1. Situación Problemática

En los últimos diez años, se ha evidenciado una mayor cantidad de investigaciones enfocadas a las Metaheurísticas, siendo, según la base de datos bibliográfica Scopus, los Algoritmos Genéticos el tema preferido por los investigadores (ver Tabla 1). En cambio, temas como las Heurísticas se han relegado de tal manera que, según Scopus, su principal uso ha sido para combinarlas con otras heurísticas y, así, mejorar las soluciones iniciales.

Entre las heurísticas menos usadas encontramos el algoritmo del Vecino más Cercano, sobre el que existen pocos esfuerzos para modificar su pensamiento, considerado *miope* al preocuparse solo en construir su ruta en base a lo mejor que encuentra en el corto plazo, y aún menores esfuerzos para realizar modificaciones inspiradas en el pensamiento sistémico, cuya disciplina busca identificar, describir, comprender y modificar la estructura de las interrelaciones de los sistemas organizacionales (Pérez, 2011). En tal contexto, Pérez (2011) utilizó dicho pensamiento para construir el Sacrificio Cortoplacista, el cual mejora las soluciones del algoritmo original, de tal manera que la solución brindada por el Sacrificio Cortoplacista es mejorada por la heurística 2Opt.

En esta investigación se desarrollaron modificaciones de los algoritmos propuestos por Pérez (2011), las cuales se sustentaron en el concepto de Sacrificio Cortoplacista y en la heurística de mejoramiento 2Opt, con el objetivo de mejorar el nivel de eficacia de nuestras versiones planteadas.

Se revisó la base de datos de referencias bibliográficas Scopus, fuente donde se encuentran recursos electrónicos para evaluar el desarrollo de los diferentes temas de investigación según el área de estudio y según el año de publicación.

La base de datos de referencias bibliográficas Scopus presenta artículos a nivel internacional, predominantemente en inglés, razón por la cual la búsqueda de palabras clave y filtros se realizó en inglés y, a partir de ello, se construyó la Tabla 1.

Se realizó una búsqueda sobre el pensamiento sistémico y heurísticas en el Repositorio Nacional de Acceso Libre a Información Científica para la Innovación (ALICIA) del Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica del Perú (CONCYTEC), al igual que en el Registro Nacional de trabajos de investigación (RENATI) de tesis, en los cuales no se encontraron artículos relacionados a algoritmos inspirados en el pensamiento sistémico. Por tanto, dicho campo puede ser utilizado como una fuente de ideas para nuevos algoritmos, de donde se obtuvieron analogías que permitieron mejorar el nivel de eficacia de las versiones desarrolladas por Pérez (2011).

Pérez (2011) define la *eficacia* como la calidad de la solución aportada por el algoritmo y la *eficiencia* como el tiempo de cómputo para obtener dicha solución.

Tabla 1

Cantidad de investigaciones en Scopus 2014-2023

Title, Keywords, Abs	Total	%	2014-2023	%
"Genetic Algorithms"	**211,980**	**68%**	**109,361**	**62%**
(Adicionando) └─► tsp*	1,414	**43%**	699	**39%**
"Heuristic Algorithm"	**56,678**	**18%**	**37,182**	**21%**
(Adicionando) └─► tsp*	631	**19%**	404	**22%**
"Local Search"	**25,564**	**8%**	**14,575**	**8%**
(Adicionando) └─► tsp*	646	**20%**	342	**19%**
"Metaheuristics"	**11,910**	**4%**	**8,856**	**5%**
(Adicionando) └─► tsp*	225	**7%**	167	**9%**
"Bee colony Algorithm"	**5,692**	**2%**	**4,758**	**3%**
(Adicionando) └─► tsp*	37	**1.1%**	27	**1.5%**
"Ant Algorithm"	**1,338**	**0.4%**	**409**	**0.2%**
(Adicionando) └─► tsp*	73	**2.2%**	17	**0.9%**
"2-opt" or "2Opt"	**752**	**0.2%**	**440**	**0.3%**
(Adicionando) └─► tsp*	242	**7.4%**	137	**7.6%**
"Nearest neighbor heuristic"	**110**	**0.04%**	**53**	**0.03%**
(Adicionando) └─► tsp*	20	**0.6%**	8	**0.4%**

Nota: Cantidad de documentos de Scopus filtrados por las áreas Computer Science, Engineering y Mathematics según palabras contenidas en Title, Keywords or Abstract y según palabras de búsqueda “tsp”.
Fuente: Scopus al 24/12/2023

1.2. Formulación del Problema

1.2.1. Problemas General

¿Qué diseño deben tener los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que con los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas?

1.2.2. Problemas Específicos

- ¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt?

- ¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del Sacrificio Cortoplacista Adaptativo 2Opt_r?

- ¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia de las metaheurísticas seleccionadas?

1.3. Justificación de la Investigación

1.3.1. Justificación Teórica

En búsqueda de generar nuevo conocimiento en el campo de las heurísticas y/o metaheurísticas inspiradas en el pensamiento sistémico, se modificó una heurística basada en el pensamiento del Vecino más Cercano: SCA_2Opt_r, por medio del cual se buscó ampliar las heurísticas y/o metaheurísticas disponibles para la resolución del Problema del Agente Viajero Simétrico, mejorando su nivel de eficacia con respecto a los algoritmos desarrollados por Pérez (2011), de tal manera que los nuevos algoritmos puedan ser el punto de partida para futuras investigaciones.

1.3.2. Justificación Práctica

Se buscó que, por medio de la aplicación de las nuevas versiones de los algoritmos propuestos, se disponga de algoritmos capaces de encontrar buenas soluciones a Problemas del Agente Viajero Simétrico. Sin embargo, la aplicación práctica de dicho algoritmo va más allá de su mera solución en

sí misma, basada en la mejor ruta del agente viajero. Esto debido a que, como indican Matai, Singh y Mittal (2010), el TSP puede ser usado para resolver problemas de ruteo de vehículos cuando no interviene el tiempo o la capacidad y el número de vehículos es fijo. Es decir, determinación de rutas.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General

- Determinar el diseño de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas.

1.4.2. Objetivos Específicos

- Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt.

- Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r.

- Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia de las metaheurísticas seleccionadas.

1.5. Hipótesis de la Investigación

1.5.1. Hipótesis General

- Mediante los algoritmos propuestos será posible resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que con los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas.

1.5.2. Hipótesis Específicas

- El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt.

- El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r.

- El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia de las metaheurísticas seleccionadas.

1.6. Limitaciones del estudio

- Se realizó un análisis comparativo de soluciones para instancias del problema del Agente Viajero Simétrico obtenidas de la librería virtual TSPLIB.

- Se resolvieron instancias del problema del agente viajero simétrico para grafos de hasta 1002 nodos, contenidos en la librería virtual TSPLIB.

- Se utilizaron los indicadores *Eficacia* y *Nivel de Eficacia* para evaluar la calidad de las soluciones obtenidas por los algoritmos seleccionados.

- El indicador *Eficacia* se usó para realizar la posterior medición del indicador *Nivel de Eficacia*.

- Se determinó la complejidad temporal de los algoritmos desarrollados, medidos por el método de la Big O, evaluando la cantidad de operaciones elementales según algoritmo. Es decir, con respecto al recurso tiempo.

CAPÍTULO II

MARCO TEÓRICO

Para mostrar cómo otros autores han abordado el presente tema de investigación, se procedió a buscar investigaciones, dentro de los últimos años, relacionadas con el Problema del Agente Viajero, algoritmos que solucionen dicho problema como la heurística del Vecino más Cercano, la heurística 2Opt, entre otras, con excepción de la tesis internacional del 2011 desarrollada por Pérez en Colombia, la cual fue necesariamente agregada por ser uno de los pocos antecedentes más recientes con respecto al Algoritmo del Vecino más cercano con Sacrificio Cortoplacista, algoritmo que es considerado clave dentro del presente trabajo.

Además, se han revisado investigaciones internacionales desde el año 2014 hasta 2023 (ver Tabla 1), encontrándose pocas enfocadas a heurísticas como el Vecino más Cercano o el algoritmo de mejoramiento heurística 2Opt, lo que resalta la escasez de estudios centrados en mejorar la eficacia de los algoritmos heurísticos tradicionales. Estas investigaciones no solo evidencian esta carencia, sino que también nos sirven como punto de partida para generar ideas y extraer analogías que fundamentan el desarrollo de la presente investigación.

2.1. Antecedentes de la Investigación

2.1.1. Antecedentes Nacionales

Tito, Silva, Alfaro y Fajardo (2019), en su investigación “Aplicación de Algoritmo de Colonia de Hormigas al Problema del Agente Viajero”, presentaron un modelo matemático basado en el Algoritmo de Colonia de Hormigas para resolver el Problema del Agente Viajero (TSP), para lo cual se usó una instancia de la librería virtual TSPLIB, Berlin52, cuya solución óptima reportada en TSPLIB es 7542, obteniendo una solución de 7680 de distancia con el Algoritmo de Colonia de Hormigas. Para tal solución, se consideraron 200 hormigas y se tomaron valores de parámetros tomados como recomendación en otra investigación. El algoritmo se implementó en Matlab. Como conclusión, se encontró que la mejor distancia encontrada fue de 7680 (Error Relativo de 1.83% con respecto al óptimo reportado en TSPLIB).

Quiñones, Villar y Gutiérrez (2017), en su investigación “Algoritmo del Agente Viajero para minimizar el recorrido de visitas programadas en una empresa de seguros”, realizaron una planificación de rutas para las visitas de una empresa aseguradora, modelando el Problema al Agente Viajero utilizando herramientas como el uso de software WinQSB (módulo Network Modeling), XLSTAT, Excel e IBM SPSS v20, donde se logró reducir en un 64% la distancia recorrida necesaria para el clúster 1, en un 24% para el clúster 2 y en un 63% para el clúster 3, con lo cual se redujo el tiempo de recorrido de los clústeres, logrando ampliar la cobertura y reducir los costos de operación. De la población estudiada, todos los afiliados y colaboradores de la oficina de seguros ubicada en Chimbote, se seleccionó una muestra de 23 afiliados, 11 para el clúster 1 y 12 para el clúster 2; cuyos domicilios fueron representados en coordenadas por medio de Google Maps y GPS para el posterior cálculo de la matriz de distancias, donde la creación de los clústeres ayudó a disminuir la dificultad de resolución del problema. Se logró un ahorro de 1289 minutos durante el recorrido de la red, proyectándose a una ampliación de la cobertura.

2.1.2. Antecedentes Internacionales

Eppen, Gould, Schmidt, Moore y Weatherford (2000), en su libro "Investigación de Operaciones en la Ciencia Administrativa. Capítulo 9: Toma de Decisiones con objetivos múltiples y heurística", mostraron el funcionamiento de una variante del algoritmo del Vecino más Cercano, como un intento de superar el denominado pensamiento miope del algoritmo del Vecino más Cercano. Fue la desarrollada por Gavett (1965, como se citó en Eppen et al., 2000), modificando la matriz de costos antes de aplicar el criterio del Vecino más Cercano, la cual consiste en esencialmente dos pasos. Paso 1: A cada valor de la matriz de costos se le resta el mínimo valor encontrado en su respectiva columna, construyendo una nueva matriz de costos. Y el Paso 2: A la matriz de costos transformada en el paso 1, se le aplica la heurística original, considerando los "0's" como caminos habilitados. De esta forma, busca mejorar. No muestra un ejemplo práctico para medir su eficacia, simplemente menciona cómo es el funcionamiento del algoritmo. Como conclusión, se indica que, si bien la heurística modificada no siempre proporciona los mejores resultados, su aplicación resulta sencilla de implementar y, en ocasiones, conduce a la obtención de buenos resultados teniendo en cuenta los recursos limitados.

Por su parte Rao y Jin (2010), en su paper conference "A New Hybrid Algorithm for Solving TSP", diseñaron un nuevo algoritmo híbrido para resolver el Problema del Agente Viajero (TSP), denominado NN&IN, el cual combinó las ventajas de la heurística del Vecino más Cercano (NN) con las del algoritmo denominado Inserción más Cercana (IN), con el objetivo de resolver el problema del agente viajero. Finalmente, se probó las soluciones determinadas por el algoritmo híbrido y sus versiones sin hibridación para resolver instancias de la librería virtual TSPLIB. Por medio de los resultados experimentales encontrados, se concluyó que la heurística híbrida construida resultó más eficiente que las heurísticas originales aplicadas por separado.

Pimentel (2011), en su artículo "Double-ended nearest and loneliest neighbour–a nearest neighbour heuristic variation for the travelling salesman

problem", introdujo el concepto de soledad de una ciudad, el cual se refiere a su grado de aislamiento en relación con otras ciudades. Esta medida se calcula a partir de la distancia promedio de la ciudad en cuestión a todas las demás ciudades en un área específica. Cuanto mayor sea la distancia promedio, mayor será la soledad percibida de la ciudad. Este concepto se puede combinar con las ideas del vecino más cercano para entender mejor la dinámica e interacción de los nodos del grafo, dando nuevas versiones del algoritmo del Vecino más Cercano. Los nuevos métodos, al mantener una complejidad temporal similar a sus homólogos anteriores, han demostrado ofrecer resultados superiores. Estos métodos dan prioridad a las ubicaciones más aisladas. Esta priorización se asemeja a una analogía de la distribución temprana de bienes y servicios a sitios distantes, reflejando una política social orientada hacia la equidad y la inclusión. Se presentaron tres versiones: Vecino Más Cercano de Doble Extremo (DENN), Vecino Más Cercano y Más Solitario (NMS) y Vecino Más Cercano y Más Solitario de Doble Extremo (DENLN), cuyas soluciones fueron comparadas por las obtenidas por el Vecino más Cercano en instancias de la librería TSPLIB. Se concluyó que, para las 11 instancias evaluadas, las versiones NMS y DENLN lograron determinar soluciones que superaron a su versión original del Vecino más Cercano en un 100% de los casos, mientras que la versión DENN solo superó al Vecino más Cercano en 5 de 8 instancias revisadas, es decir, un 63% de nivel de eficacia.

Kızılates y Nuriyeva (2013), en su artículo "A parametric hybrid method for the traveling salesman problem", presentaron un algoritmo denominado Método Híbrido Paramétrico (PHM), el cual se basa en las heurísticas del Vértice Más Lejano (FV) y el algoritmo heurístico Greedy para abordar el problema del vendedor viajero. Se aplicaron estos algoritmos y el Vecino Más Cercano en 22 instancias de la librería virtual TSPLIB. Los resultados experimentales revelaron que el PHM fue notablemente más eficiente que el algoritmo Greedy o el Vecino Más Cercano, logrando un nivel de eficacia del 100% (es decir, en las 22 instancias, la solución del PHM fue la mejor).

Pérez (2011), en su tesis de maestría “Heurística inspirada en el análisis sistémico del Vecino más Cercano, para solucionar instancias simétricas TSP, empleando una base comparativa multicriterio”, desarrolló el pensamiento del Sacrificio Cortoplacista, inspirado en el Pensamiento Sistémico, el cual modificó el algoritmo del Vecino más Cercano eligiendo, desde una ciudad, la segunda mejor alternativa para posteriormente seguir con el pensamiento original del algoritmo. Dicha solución mejoró con la aplicación de la heurística 2Opt, generando un algoritmo híbrido denominado SCA_2Opt. Para estudiar la eficacia de sus algoritmos desarrollados (SCA_2Opt y SCA_2Opt_r), se revisaron diversas instancias simétricas del Problema del Agente Viajero de la librería virtual TSPLIB, donde en una primera etapa se comparó sus soluciones con las óptimas reportadas en dicha librería y medir su error relativo (distancia relativa al óptimo) con respecto a la solución óptima reportada en TSPLIB; en una segunda etapa se comparó con metaheurísticas desarrolladas en investigaciones internacionales, las cuales agrupó de la siguiente manera: Algoritmos Genéticos, Colonia de Hormigas, Enfriamiento Simulado y Otras Metaheurísticas. Como conclusión, el autor indica que existe una tendencia de usar algoritmos híbridos y que existe un campo en el pensamiento sistémico por explorar como fuente de inspiración. Para las instancias revisadas, el SCA_2Opt presentó un error relativo al óptimo que varió entre un 2.7% a un 10%; mientras el SCA_2Opt_r entre un 0.9% a un 5.6%.

Mendoza (2017), en su tesis doctoral “Diseño de Algoritmos Heurísticos y Metaheurísticos eficientes para resolver el Problema del Agente Viajero”, mostró que las heurísticas no necesariamente presentan el mismo nivel de eficacia ante distintos tamaños de instancias, al comparar la heurística del Vecino más Cercano, Búsqueda Tabú, Scatter Search, GRASP, Algoritmos del WINQSB (Branch and Bound) y Solver de Excel (Simplex), tal que la Búsqueda Tabú resultó ser más eficiente para tamaños de instancias pequeños y el Scatter Search más eficiente para tamaños grandes de instancias. Las comparaciones se realizaron en dos grupos: en el primero se comparó el Solver (Simplex), WinQSB (Branch and Bound), Vecino más

Cercano, GRASP y la Búsqueda Tabú para doce instancias de tamaño medio (8 a 175 nodos) de TSPLIB; el segundo grupo comparó el Scatter Search, GRASP y Búsqueda Tabú para treinta instancias de tamaño medio alto (101 a 400 nodos), donde en veintidós de treinta instancias (73.3%), el Scatter Search resultó más eficaz. Se concluyó la Búsqueda Tabú resultó más eficaz para tamaños de instancias pequeños/medianos y Scatter Search para instancias grandes.

Akhand, Ayon, Shahriyar, Siddique y Adeli (2020), en su artículo “Discrete Spider Monkey Optimization for Traveling Salesman Problem”, implementaron una versión discreta del algoritmo Spider Monkey Optimization (SMO), basado en el comportamiento social de los monos araña, denominado Discrete Spider Monkey Optimization (DSMO) para resolver el TSP, la cual usa dos operadores implementados para su versión discreta: el primero basado en una “Secuencia de Intercambio” (Swap Sequence) y un “Operador de Intercambio” (Swap Operator), permiten la interacción entre los monos arañas (cada uno representa una solución) para ir a un óptimo global en base a la experiencia del mono araña líder local, líder mundial, o uno seleccionado al azar. El algoritmo imita el comportamiento social del mono araña en búsqueda de alimento, donde al inicio un solo grupo que está en búsqueda de comida se divide en subgrupos comandados por un sublíder (los cuales pueden seguir subdividiéndose en más subgrupos), lo cual se denomina “fisión”, para luego volver a reunirse en un solo grupo, lo cual se denomina “fusión”. Dicho proceso de fisión y fusión se repite la cantidad de veces necesaria hasta encontrar una fuente de comida abundante para el grupo y en cada reagrupación se comparte información para los miembros del grupo y así mejorar su proceso de búsqueda en cada iteración. Se usó instancias simétricas de TSPLIB para estudiar la eficacia de DSMO, comparada con otros algoritmos metaheurísticos mostrando la efectividad del DSMO para resolverlas. Por ello, se encontró que la eficacia del algoritmo DMSO desarrollado como la variante discreta del algoritmo SMO resultó ser mejor que la eficacia de los algoritmos ACO, VTPSO y ABCSS revisados.

Adicionalmente, otros autores como Alemayehu y Kim (2017), introducen variantes de la heurística del Vecino más Cercano con el objetivo de reducir el tiempo de transmisión de los datos desde un origen a otro en un sistema de red. Y, al seguir revisando la literatura, se encontraron pocos trabajos enfocados en la modificación del Criterio 2Opt o el Vecino más Cercano en la búsqueda de mejores soluciones.

2.2. Bases Teóricas

2.2.1. Pensamiento Sistémico

Desde el punto de vista matemático, hay casos en los que se acostumbra a analizar un problema aislándolo con el objetivo de entenderlo en un entorno controlado, para luego plantear hipótesis que nos ayuden a resolverlo. En otros casos, se agregan restricciones para disminuir el espacio de búsqueda, entendiendo los riesgos de perder elementos que podrían determinar mejores soluciones. Los anteriores enunciados son formas de obtener soluciones eficientes a ciertos problemas, pero si lo trasladamos a nivel de las organizaciones desde el punto de vista sistémico, como afirma Espinoza (2010), en los problemas organizacionales se debe ver el todo y no solo una parte, y cómo las partes interactúan, con el fin de entender el funcionamiento del todo y el efecto de los cambios a generar.

Senge (2010) indica la existencia de cinco disciplinas, resaltando la importancia de su desarrollo conjunto en las organizaciones actuales. Dichas organizaciones reciben el nombre de *organizaciones inteligentes,* que son las que aprenden y se adaptan mejor al entorno. Las disciplinas definidas son: Construcción de Visión Compartida, Aprendizaje en Equipo, Dominio Personal, Modelos Mentales y el Pensamiento Sistémico, donde esta última integra al resto de disciplinas con el objetivo de comprender el funcionamiento de una organización, viéndola como un sistema y sus interrelaciones internas para poder resolver los problemas emergentes durante su modificación.

Dentro de la estructura del pensamiento sistémico, existen estrategias que configuran una manera de pensar y actuar frente a distintas situaciones. Espinoza (2010) menciona las siguientes:

a. **Estructura Sistémica.** Según el análisis sistémico, el término Estructura Sistémica "tiene relación con la configuración de interrelaciones entre los sistemas, subsistemas y componentes claves" (Espinoza, 2010, p.21). Debemos notar que las interrelaciones existentes no implican conexión entre todas las componentes, sistemas y subsistemas pues existen también los subsistemas aislados, así como existen componentes que no se relacionan directamente con otras. Las estructuras sistémicas pueden ser invisibles hasta el momento que alguien encuentra un problema e intenta solucionarlo, por lo cual dichas estructuras no necesariamente se construyen a sabiendas, sino a partir de percepciones realizadas a lo largo del tiempo, de manera consciente o inconsciente.

   A nivel de un algoritmo, este concepto puede ser relacionado con el diseño del algoritmo y cómo realiza la búsqueda y construcción de soluciones eficientes. Según Alancay, Villagra y Villagra (2016a), las búsquedas basadas en poblaciones brindan mejores resultados que las búsquedas basadas en la trayectoria, por lo cual esta última estructura permite una mejor búsqueda por parte del algoritmo. Por ello, resulta evidente la importancia de la estructura de nuestro algoritmo, pero los efectos positivos no resultan tan evidentes hasta la ejecución del algoritmo, debido a que estos pueden estar basados en percepciones personales del investigador.

b. **Lenguaje del Pensamiento Sistémico.** No existen eventos ni interacciones aisladas, cada uno forma parte de un circuito de causalidad, denominado también circuito de retroalimentación, donde cada causa es también un efecto para otra parte del sistema. De esa manera, la influencia e interacción de una componente

sobre otra puede ser representada por medio de una flecha, lo cual también abre el pensamiento de que, al no existir flechas entre un par de componentes, estas no se relacionan directamente, pero pueden estar relacionadas por otras flechas que permitan dicha influencia. Además, se puede tener la presencia de ciclos repetitivos, los cuales pueden tener una influencia negativa o positiva en la situación deseada.

A nivel de los algoritmos, los ciclos repetitivos pueden ayudar a mejorar una solución al ejecutar consecutivamente determinadas acciones. Con esto, se busca un circuito de causalidad que produzca el mejor efecto posible sobre el sistema. Por ejemplo, la aplicación de un criterio de búsqueda múltiples veces.

c. **Ciclos Reforzadores.** Son representaciones de un pensamiento circular (Senge, 2010), es decir, que se repite hasta encontrar un equilibrio en el cual se detiene (un mecanismo compensador). Existen dos tipos de ciclos reforzadores:

   - Ciclo Virtuoso: En este ciclo un pequeño cambio puede producir un efecto que crece y se encamina a una dirección deseada

   - Ciclo Vicioso: Ciclo en el cual un pequeño cambio puede encaminarnos a una dirección no deseada.

   A nivel de los algoritmos, este concepto puede usarse para buscar un criterio el cual permita un efecto virtuoso en cada etapa de búsqueda de soluciones, es decir, ir hacia una dirección en la cual se mejore la solución del proceso anterior.

d. **Ciclos Compensadores.** Son mecanismos que sirven para limitar el efecto excesivo provocado por un ciclo virtuoso o un ciclo vicioso, hasta alcanzar una estabilidad (Senge, 2010). Muchos de ellos “surgen en situaciones que parecen ser auto correctivas y auto

reguladoras, al margen de la voluntad de los participantes" (Espinoza, 2010, p.22).

A nivel de los algoritmos, este concepto puede relacionarse con determinar un mecanismo que permita perturbar el ciclo reforzador, en búsqueda de reducir sus efectos en las soluciones obtenidas.

e. **Demoras.** Son procesos que retrasan la aparición de los efectos presentes tanto en los ciclos reforzadores como en los compensadores y que pueden cambiar la conducta del sistema. Para estos casos, Espinoza (2010) indica que "en los ciclos compensadores, las demoras pueden cambiar drásticamente la conducta del sistema" (p.23). En los algoritmos, donde el objetivo es determinar la mejor solución siguiendo los mejores pasos posibles, las demoras pueden verse como mecanismos que retrasan la aparición de un óptimo local, con el objetivo de evitar quedar atrapados en alguno de manera prematura y con la esperanza de conseguir encontrar un mejor óptimo local.

2.2.2. El juego de la cerveza en el libro "La Quinta Disciplina"

Consiste en una simulación desarrollada en los años 60 del Siglo XX, en la Escuela de Administración Sloan del MIT, la cual ilustra el actuar de los participantes en una simulación de gestión de la cadena de suministro de una marca de cerveza específica. A continuación, se presenta la historia descrita por Senge (2010), de cómo se desarrollan los hechos, contada desde el punto de vista de cada uno de los participantes:

a. **El minorista.** Es el dueño de una tienda o negocio pequeño, el cual provee de productos directamente a los usuarios finales. Para esta simulación, el producto angular es la Cerveza de los Enamorados. Una vez por semana viene el representante del mayorista, a quien por medio de un formulario se le realiza un pedido de cuántas cajas de Cerveza de los Enamorados desea

adquirir. Tal pedido es abastecido con una demora de cuatro semanas, donde se observó que la demanda de los consumidores es de 4 cajas por semana. Sucedió que la demanda de los usuarios finales se duplicó a 8 cajas de cerveza y, ante la demora en el abastecimiento por parte del proveedor y por temor a quedarse sin stock, el minorista empieza a incrementar la cantidad de cajas a pedir, hasta llegar a 24 cajas en la semana 8. Ante la desesperación de no recibir sus pedidos completos, sigue con el pensamiento de incrementar sus pedidos.

b. **El mayorista.** Es el intermediario entre la fábrica de cerveza de los Enamorados y los minoristas de una determinada región, en la cual solo se tiene la presencia de un único mayorista. El mayorista se comunica con la fábrica de cerveza mediante un formulario, el cual indica el número de gruesas a solicitar por semana (lo habitual es pedir 4 gruesas), donde una gruesa consiste en un cargamento en donde la camioneta está al tope con Cerveza de los Enamorados, con una demora de 4 semanas en recibir su pedido. Por su parte, el mayorista, ante el incremento de pedidos por parte del minorista, empezó a enviar su stock de reserva, el cual se acabó a las pocas semanas y, por tanto, empezó a enviar pedidos parciales a los minoristas, porque la fábrica también demoraba en abastecer sus pedidos y empezaron a recibir pedidos parciales del fabricante. Esto provocó pánico en el mayorista al no verse capaz de abastecer a sus minoristas, así que, poco a poco fue incrementando el número de gruesas pedidas a la fábrica, llegando hasta 60 gruesas en la semana 12. Lo cual, al final de la semana 15, le provocó que llegaran grandes cantidades de cerveza y se tuviera un exceso de stock, el cual ya no podía vender porque los minoristas empezaron a pedir cero cajas de cerveza.

c. **La fábrica de cerveza.** El encargado de la fábrica de cerveza siente gran satisfacción al notar que los pedidos por parte de los mayoristas se incrementan de 4 a 40 gruesas durante las primeras

semanas. Pero, a partir de la semana 7, empiezan a tener problemas para satisfacer la demanda, al tener una capacidad limitada de producción. En las siguientes semanas, se establecieron horas extras para aumentar la producción, lo cual hace que se incremente la cantidad en stock. Pero el encargado de la fábrica se empieza a sentir abrumado cuando, a partir de la semana 17, los pedidos disminuyeron y, en la semana 18, reciben pedidos de cero gruesas. Pero, en la fábrica se seguía trabajando con horas extras y bonos, pensando que la demanda seguía siendo alta, lo cual generó un gran exceso de stock y una disminución drástica de la producción, debido a que la demanda de los clientes finales siempre fue de cuatro cajas de cerveza. Sin embargo, ante el no abastecimiento oportuno, cada jugador del juego de la cerveza empezó a pedir más cajas de cerveza.

2.2.3. Lecciones de juego de la Cerveza en el libro "La Quinta Disciplina"

A partir del "Juego de la Cerveza" explicado por Senge (2010), se extraen las siguientes lecciones:

a. **La estructura influye sobre la conducta.** Se considera estructura a las interrelaciones entre variables claves del sistema que modelan el actuar de las personas, posibilitando cierto tipo de acontecimiento (Senge, 2010). Esto quiere decir que, dependiendo del tipo de interrelaciones definidas, puede llevar a acontecimientos distintos. Dentro de dichas estructuras se presentan los individuos, donde se dice que: "Cuando pertenecen al mismo sistema, las personas, a pesar de sus diferencias, suelen producir resultados similares" (Senge, 2010, p.57). Por ejemplo, si el administrador tomase el lugar del mayorista, el mayorista el lugar del minorista y el minorista el lugar del administrador, lo más probable es que se presentaría la misma conducta del sistema, la cual provocó el caos presentado al final del juego de la cerveza. Senge (2010) también indica que la

estructura del sistema frecuentemente es la responsable de generar sus propios momentos de crisis, los cuales no son necesariamente debido a errores individuales. Por ejemplo, en el juego de la cerveza, la estructura definió un período de cuatro semanas entre el pedido y el abastecimiento, lo cual provocó una crisis dentro del sistema.

b. **La estructura de los sistemas humanos es sutil.** La estructura de los sistemas humanos, entendiéndose como los múltiples subsistemas del cuerpo humano y sus interrelaciones (cardiovascular, neuronal, etc.), influye en el modo de tomar decisiones en base a percepciones que controlan la conducta (Senge, 2010). Es decir, los sistemas humanos corresponden a la forma de pensar, de actuar y a sus modelos mentales ante distintas situaciones. En el juego de la cerveza, se observó que la influencia de los jugadores no fue de la manera deseada, provocando la inestabilidad del sistema, donde cada jugador tomó decisiones basadas en sus propios modelos mentales, sin darse cuenta de cómo esto afectó a los demás jugadores.

c. **El punto de apalancamiento.** Definido por Senge (2010) como “un cambio que con un mínimo esfuerzo, llevaría a una mejora significativa y duradera” (p.85). Tal que dicho punto de apalancamiento, con frecuencia, se descubre con nuevas formas de pensar, debido a que, a menudo, no es obvio de determinar; pero, una vez determinado, puede producir cambios drásticos y positivos en la conducta del sistema.

   En el juego de la cerveza, los jugadores siguen una forma de pensar similar debido al no abastecimiento de sus pedidos en el tiempo oportuno, creando más inestabilidad en el sistema, el cual se pudiese eliminar o al menos reducir si se utilizaran nuevas formas de pensar y de afrontar el problema de no tener suficiente cerveza en el tiempo oportuno. Estas nuevas formas de pensar conducen a

nuevas formas de afrontar el problema y, por lo tanto, brindan la posibilidad de encontrar un punto de apalancamiento.

2.2.4. Problema del Agente Viajero

Problema sencillo de enunciar, pero que resulta ser uno de los más complejos de resolver, tiene aplicación en el ruteo de vehículos, buscando rutas eficientes en términos de costos o distancias (López, Salas y Murillo, 2014). Según Flood (1956), el Problema del Agente Viajero, enunciado en 1934 en un seminario en la universidad de Princeton por Hassler Whitney, busca encontrar una permutación de (n-1)! posibles resultados, de tal manera que se minimice el costo asociado a la permutación elegida.

Debemos diferenciar que, mientras el Problema del Agente Viajero (TSP, por sus siglas en inglés) se centra en determinar la ruta más corta que une o conecta un conjunto de ubicaciones, el VRP se centra en la optimización de rutas, donde se tiene un conjunto de vehículos, los cuales deben pasar por determinadas ubicaciones, en las cuales importa la capacidad limitada de los vehículos. Por tanto, a pesar de que el TSP puede ser utilizado como base para explicar el VRP, cada uno tiene sus propios métodos de resolución, según sus características distintivas.

El Problema del Agente Viajero (TSP) se define como:

> Dado un conjunto de n ciudades, de las cuales se conoce para cada par de ellas, la distancia que las separa, un agente viajero ha de partir de una ciudad de origen y debe visitar exactamente una vez cada ciudad del conjunto y retornar al punto de partida (Mendoza, 2017, p.5).

Medina, La Rota y Castro (2004) aportan una visión enfocada a la ruta, en la cual el TSP considera que se: “dispone de un solo vehículo que debe visitar a todos los clientes en una sola ruta y a costo mínimo. No suele haber un depósito (…), no hay demanda asociada a los clientes y tampoco hay restricciones temporales” (p.4).

Martí (2003): el Problema del Agente Viajero (TSP) fue uno de los primeros en estudiarse al desarrollarse la teoría de la complejidad algorítmica, donde Karp demostró en 1972 que dicho problema pertenece a los NP-Hard, problemas difíciles de resolver. Según la definición brindada por la Teoría de Grafos, el problema del agente viajero consiste en encontrar un circuito hamiltoniano que minimice el costo asociado a visitar todas las ciudades. Es importante señalar que, en un problema de enrutamiento en el que no se consideran las capacidades de los vehículos ni el tamaño de la flota para cubrir la demanda correspondiente, este escenario se asemeja al Problema del Agente Viajero. En dicho contexto, el problema puede ser representado por medio de un grafo sagital o una estructura matricial, lo cual permite visualizar y abordar las rutas y conexión entre cada par de nodos de manera más clara.

2.2.5. Definición de grafo

Menéndez (1998), plantea las siguientes perspectivas para definirlo:

- Gráficamente, es considerado como diagrama o dibujo.
- Algebraicamente, se considera como un par de conjuntos.
- Geométricamente, se trata como un conjunto de puntos en el espacio, los cuales se unen entre sí por medio de líneas.

Martín y Méndez (2004), definen un grafo de manera algebraica de la siguiente manera: Dado un grafo G, entonces G = (V, E), siendo V el conjunto de nodos del grafo; $E \subseteq V \times V$, donde $\{e / e \in E\}$, son las líneas formadas por la unión de dos nodos, denominadas aristas o arcos. Donde:

- Si e= (x, y) = (y, x) —► no existe orientación, “e” se denomina arista.
- Si e= (x, y) ≠ (y, x) —► sí existe orientación, “e” se denomina arco.

- Dado e = (x, y), “x” es denominado nodo inicial del arco (o arista) “e”, “y” es denominado nodo final del arco (o arista) “e”.

Por su parte, Menéndez (1998) indica que la utilidad de los grafos radica en que, por medio de ellos, se puede representar de manera visual una variedad de situaciones. Por ejemplo, un mapa de carreteras en el cual los nodos representarían las ciudades y las líneas que unen dichos nodos representarían a las carreteras o vías de comunicación entre dichas ciudades. De ello, se desprende que el Problema del Agente Viajero también puede ser representado por un grafo.

2.2.6. Conceptos básicos sobre grafos

A continuación, Melo (2011) expone algunos conceptos necesarios para el desarrollo de la presente investigación:

- Orden de un grafo: es el número de nodos del grafo.
- Nodo: estructura que posee algún tipo de información. Los nodos se unen por pares y el medio de unión es una línea no orientada o con orientación (Coto, 2003).
- Arista: línea que une dos nodos, la cual no tiene orientación.
- Arco: línea que une dos nodos, la cual tiene orientación.
- Adyacencia de aristas: aristas que tienen un nodo en común.
- Adyacencia de nodos: cuando tienen una arista que los une.
- Camino: sucesión de arcos adyacentes.
- Camino Hamiltoniano: sucesión de arcos adyacentes, que visitan todos los nodos del grafo una única vez.
- Circuito Hamiltoniano: camino hamiltoniano cuyo nodo final es igual al nodo inicial.

- Grafo con nodos ponderados: los nodos se distinguen de los otros por tener asignado un valor numérico.

- Grafo con aristas o arcos ponderados: las aristas o arcos se distinguen de otros por tener asignado un valor numérico o peso.

- Grafo Ponderado: aquel grafo que tiene sus nodos, aristas o arcos ponderados.

Por otro lado, Martí (2003), nos brinda la siguiente definición:

- Matriz de Costos: También denominada matriz de pesos, es aquella cuyos elementos "c_{ij}" representan el costo o peso de la arista (i, j), de tal modo que, al presentar un "Grafo Ponderado", se usa la siguiente notación: G = (V, E, C), donde: V es un conjunto de nodos, $E \subseteq V \times V$ es un conjunto de aristas y C = (c_{ij}) es una Matriz de costos.

2.2.7. Definición de instancia

Para definir una instancia, se usó la definición de dos autores, para luego buscar una definición que sintetice ambas definiciones. A continuación, se presentan las dos definiciones:

- Respecto a un grafo, una instancia es una particularización de "G=(V,E), donde V es el conjunto de nodos o vértices y E es el conjunto de aristas entre los vértices. Una cota $k \in Z$, con $k \leq |V|$." (Zuñiga, 2016, p.16), la cual sirve para diferenciar las características de un grafo con respecto a otro.

- "Una instancia de un problema es una especificación de sus parámetros" (Bonomo, 2016, diapositiva 92).

De ambas definiciones se concluyó que una *instancia* es la particularización de un problema, por medio de una configuración específica

del problema, expresada por medio de los parámetros propios al problema de grafos. Para instancias de TSPLIB, los parámetros corresponden al orden del grafo y a las siglas del nombre, que generalmente corresponde al nombre o iniciales del autor del problema, las cuales son usadas para diferenciar instancias del mismo tamaño. Ejemplo:

- Swiss42: Instancia la cual representa a un grafo de 42 nodos.
- Rd100: Instancia la cual representa a un grafo de 100 nodos.
- KroA100: Instancia la cual representa a un grafo de 100 nodos.
- Pcb442: Instancia la cual representa a un grafo de 442 nodos.
- Rat783: Instancia la cual representa a un grafo de 783 nodos.
- Pr1002: Instancia la cual representa a un grafo de 1002 nodos.

Por ejemplo, para Rd100 y KroA100, instancias de 100 nodos, se diferencian mediante iniciales que, generalmente, corresponden al autor del estudio. Las instancias de TSPLIB presentan datos en coordenadas o en matriz de distancias, entre otras representaciones e información específica, tal como el tipo de problema (TSP), la dimensión o el método de cálculo para las distancias.

2.2.8. Representación del Problema del Agente Viajero

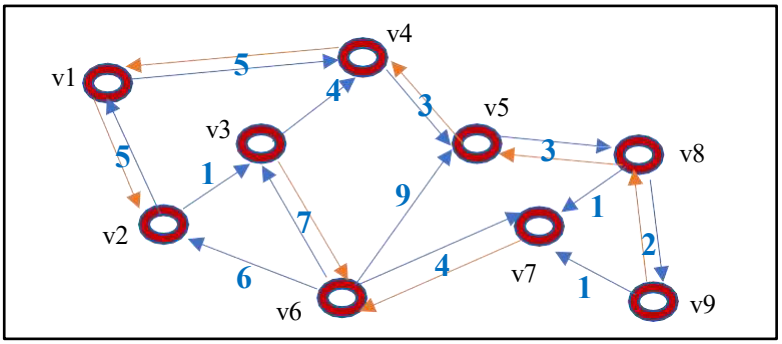
Dicho problema puede expresarse como un grafo, de tal manera que, para describir mejor las relaciones entre los nodos y aristas, se usan principalmente dos tipos de representación: Gráfica y Matricial. La representación matricial se usa para la lectura de los elementos de la matriz de costos por parte de los computadores y la representación gráfica para mostrar las relaciones de manera visual.

A continuación, se explica cada una de las dos representaciones indicadas:

a) ***Representación Gráfica.*** Su notación, según Mendoza (2017), es G = (V, E), cuyos nodos representan ubicaciones a visitar, los arcos representan el camino que une los nodos y las etiquetas de los arcos representa el costo de ir entre los nodos (ver Figura 1).

Figura 1

Ejemplo de representación gráfica de un grafo



Nota. Se representó un grafo donde el peso de ir del nodo "v_i" al "v_j" es igual al peso de ir del nodo "v_j" al nodo "v_i" para todo i ≠ j.

b) ***Representación Matricial - Matriz de costos.*** Se usa la matriz de costos para representar las relaciones existentes entre los nodos y aristas por medio de los costos asociados a estos. A partir de la Figura 1, se construyó la matriz de costos del grafo (ver Figura 2).

Figura 2

Ejemplo de representación matricial de un grafo

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
V_1	∞	5	∞	5	∞	∞	∞	∞	∞
V_2	5	∞	1	∞	∞	∞	∞	∞	∞
V_3	∞	∞	∞	4	∞	7	∞	∞	∞
V_4	5	∞	∞	∞	3	∞	∞	∞	∞
V_5	∞	∞	∞	3	∞	∞	∞	3	∞
V_6	∞	6	7	∞	9	∞	4	∞	∞
V_7	∞	∞	∞	∞	∞	4	∞	∞	∞
V_8	∞	∞	∞	∞	3	∞	1	∞	2
V_9	∞	∞	∞	∞	∞	∞	1	2	∞

En la Figura 2, a los nodos que no son adyacentes en la *matriz de costos* se les asignó un valor muy alto en comparación con los demás elementos de la matriz, representándolos con el símbolo infinito, con el objetivo de que, durante las operaciones realizadas, tales arcos no sean tomados en cuenta.

2.2.9. Tipos de Problemas del Agente Viajero.

Según la estructura de las componentes de la Matriz de Costos, podemos identificar principalmente dos tipos de variantes:

a. ***Agente Viajero Simétrico.*** Es aquel en el cual la distancia para ir del nodo "i" al "j" es igual a la distancia de ir del nodo "j" al nodo "i", $\forall$ i, j $\in$ I, donde I es el conjunto de índices (Chalco, 2008). Es decir, dado "a_{ij}" $\in$ M (matriz de costos), $\forall$ i, j $\in$ I = {1, 2, …, n} / "n" es el orden del grafo, si "M" representa a la matriz de un Agente Viajero Simétrico, entonces $a_{ij} = a_{ji}$, donde i ≠ j.

b. ***Agente Viajero Asimétrico.*** Es aquel en el cual la distancia para ir del nodo "i" al "j" es diferente a la distancia de ir del nodo "j" al nodo "i", $\forall$ i, j $\in$ I, donde I es el conjunto de índices (Chalco, 2008). Es decir, dado "a_{ij}" $\in$ M (matriz de costos), $\forall$ i, j $\in$ I = {1, 2, …, n} / "n" es el orden del grafo, si "M" representa a la matriz de un Agente Viajero Asimétrico, entonces $a_{ij} \neq a_{ji}$, donde i ≠ j.

2.2.10. Métodos de solución del Problema del Agente Viajero

Al ser considerado un problema NP-Hard, José Mendoza (2017) clasifica los métodos para solucionar el Problema de Agente Viajero en tres tipos de algoritmos: Exactos, Heurísticos y Metaheurísticos.

a. Algoritmos Exactos. Brindan una solución óptima al problema planteado; sin embargo, en la práctica son pocos los problemas que pueden resolverse encontrando el óptimo. En su mayoría, los

problemas de optimización son difíciles de resolver y si algún algoritmo puede encontrar una solución, el tiempo de ejecución puede ser tan elevado que resulta inaceptable (Martí, 2003).

b. ***Algoritmos Heurísticos.*** Son procedimientos eficientes que, de manera intuitiva, nos llevan a encontrar buenas soluciones, no necesariamente óptimas, en un tiempo aceptable, tomando en cuenta la estructura del problema (Nilson, 2003, como se citó en López, 2009).

c. ***Algoritmos Metaheurísticos.*** Como indican Alancay, Villagra y Villagra (2016b), el término metaheurística fue introducido por Glover en 1986 y proviene de dos términos griegos: “heuristic” y “meta”, que significan “encontrar” y “más allá, en un nivel superior” respectivamente. Al unir dichas palabras, se construye la definición de encontrar métodos para encontrar soluciones en un nivel superior. Blum y Rolo (2003, como se citó en Vélez y Montoya, 2007) explican que las metaheurísticas buscan evitar quedar atrapadas en óptimos locales, a diferencia de las heurísticas.

2.2.11. Tipos de búsqueda de los algoritmos metaheurísticos

Alancay, Villagra y Villagra (2016b) indican la existencia de distintas formas de clasificar a los algoritmos metaheurísticos. Por parte de la presente investigación, nos centramos en una clasificación basada en el número de soluciones que son procesadas (o utilizadas) durante la búsqueda del óptimo. Por ello, se distinguió dos tipos de búsquedas, las cuales están basadas en la Trayectoria o en la Población. A continuación, son explicadas:

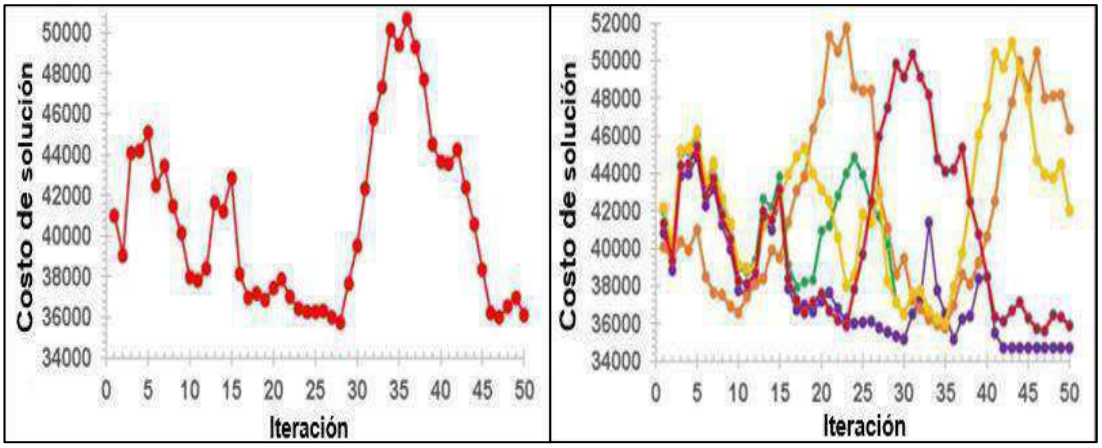
a. ***Basadas en la Trayectoria***. Se caracterizan por iniciar el proceso de búsqueda desde un punto, para luego ir mejorando la solución actual de manera iterativa, hasta cumplir un criterio de parada o un estancamiento del proceso de búsqueda al no encontrar mejores soluciones.

b. ***Basadas en la Población***. Caracterizadas por trabajar con un conjunto de soluciones iniciales, cada una de las cuales se mejora por medio de un proceso iterativo, mediante un criterio de búsqueda previamente definido y finalizando el proceso de búsqueda cuando se cumple un determinado criterio de parada.

A continuación, se presenta la Figura 3, la cual muestra la diferencia entre ambos tipos de búsqueda.

Figura 3

Búsqueda basada en la trayectoria y basada en la población



En la Figura 3 se observa lo siguiente:

- Que *la búsqueda basada en la trayectoria* parte de una solución inicial y, por medio de criterios iterativos, va explorando nuevas soluciones hasta converger a una solución (óptimo local).

- Que *la búsqueda basada en la población* tiene múltiples puntos de partida, cada uno de los cuales representa una primera solución, los cuales, por medio de un criterio iterativo, van explorando de manera independiente nuevas soluciones hasta encontrar una convergencia para cada una de las rutas iniciales.

2.2.12. Heurística del Vecino más Cercano (VMC).

La heurística del Vecino más Cercano (VMC, por sus siglas en español) sigue, por principio, el tomar la mejor decisión en el corto plazo, con la esperanza de que en el futuro traerá los mejores resultados, por lo cual se le considera una *heurística miope,* al solo preocuparse por el corto plazo (Pérez, 2011).

Esta heurística es considerada como un *método constructivo,* debido a que construye una solución por medio de la inserción de nodos en cada paso del algoritmo (Campo y Mendoza, 2018), de tal manera que la inserción de nodos se realiza eligiendo el nodo más próximo al último seleccionado. Dicho proceso se repite iterativamente hasta no tener nodos disponibles para seleccionar y, por tanto, se obtiene la ruta final construida.

Morales (2012), para la ejecución de la heurística del Vecino más Cercano (VMC), establece los siguientes pasos:

Paso 1: Seleccionar el nodo de partida o inicial.

Paso 2: Desde el nodo inicial, se determina el nodo más cercano en términos de costos y que aún no forma parte del camino construido.

Paso 3: El nodo final del paso anterior pasa a ser el nuevo nodo inicial, y se repetirá el paso 2 hasta no tener más nodos a seleccionar.

Paso 4: El camino final del paso 3 se añade al nodo inicial del paso 1, para así completar un circuito Hamiltoniano.

Como indican Eppen, Gould, Schmidt, Moore y Weatherford (2000), la heurística del Vecino más Cercano (VMC) no brinda una solución óptima, pero distintos estudios realizados han demostrado que, estadísticamente, el razonamiento para construir una solución secuencial no es inadecuado y es mejor que tener una solución construida sin algún criterio.

2.2.13. Heurística de mejora de Intercambio Bidireccional (2Opt).

Bautista y Pereira (2003) afirman que, ante la búsqueda de nuevos métodos para dar solución al Problema del Agente Viajero durante los años 50's y 60's, se llevó a cabo el desarrollo de los algoritmos heurísticos, dentro de los cuales encontramos la heurística 2Opt, planteada por Croes en 1958.

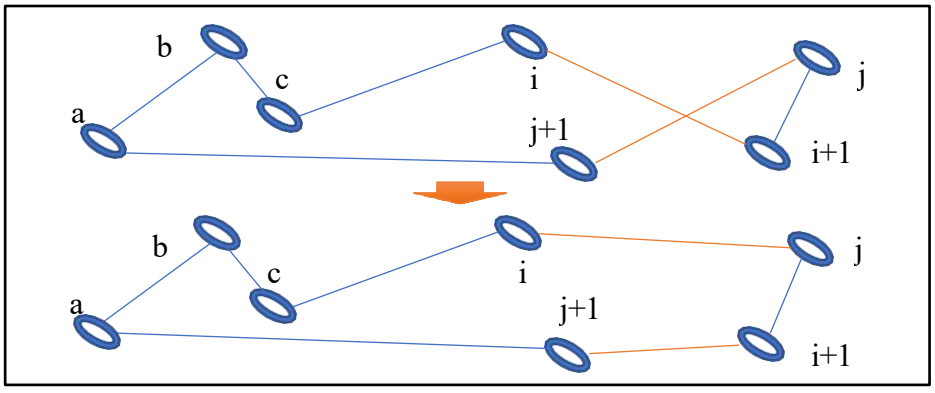
Esta heurística, también conocida como 2-Intercambio, 2-Óptimo, 2Opt o Bidireccional, es considerada como una heurística de búsqueda local, debido a que busca mejorar una solución inicial con el pensamiento de intercambio de subtour: si existen un par de aristas que se cruzan entre sí, la distancia recorrida entre ellas puede reducirse al intercambiar dos nodos de dichas aristas.

Por su parte, Hougardy, Zaiser y Zhong (2020) consideran a esta heurística como uno de los algoritmos más simples, pero capaces de encontrar buenas soluciones al problema del agente viajero en un tiempo razonable. Por motivo de su sencillez de implementación y la calidad de las soluciones que logra determinar, es usado en combinación con los algoritmos metaheurísticos, tales como los algoritmos genéticos, entre otros.

A continuación, en la Figura 4, se presenta gráficamente el funcionamiento de la heurística de Intercambio 2Opt.

Figura 4

Grafo de aplicación de heurística de Intercambio 2Opt



Pérez, Jaramillo, Parra y Moreno (2010) afirman que este procedimiento, que consiste en el intercambio de nodos, se repite hasta no encontrar un intercambio que mejore la solución actual. Debemos resaltar que distintos autores han presentado variantes para la heurística 2Opt y su criterio de intercambio de nodos.

Para la presente investigación, se optó por trabajar con el criterio de intercambio 2Opt utilizado por Pérez (2011) para resolver el Problema del Agente Viajero Simétrico, para lo cual se definieron las siguientes variables:

Costo Actual = Peso Arco (i, i+1) + Peso Arco (j, j+1)

Costo Nuevo = Peso Arco (i, j) + Peso Arco (i+1, j+1)

De tal manera que si:

Costo Nuevo < Costo Actual (Poveda, Gómez y León, 2008).

El criterio anterior es equivalentemente a:

Costo Nuevo - Costo Actual < 0 …. (Expresión 1)

Entonces, si la expresión 1 se cumple, se realiza el intercambio de nodos y se actualizan:

Ruta Actual ← Ruta asociada a Costo Nuevo

Costo Actual ← Costo Nuevo

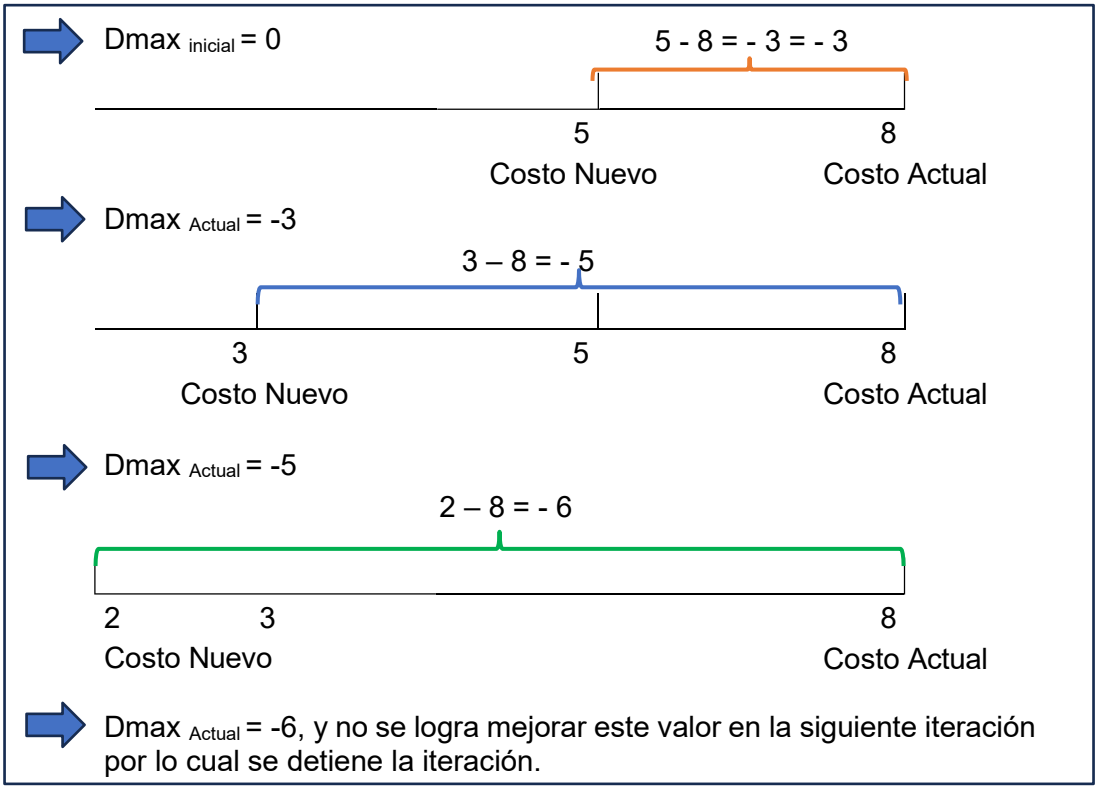
Es decir, nuestro indicador para realizar el intercambio de nodos en la heurística 2Opt consistió en el cumplimiento de la Expresión 1, la cual puede expresarse de manera más general, como lo indica Pérez (2011):

Costo Nuevo – Costo Actual < Dmax …. (Expresión 2)

Donde Dmax inicia con un valor cero, para posteriormente ir actualizándose con el valor más negativo encontrado en cada intercambio de nodos, tal como se muestra en la Figura 5.

Figura 5

Ejemplo de actualización de indicador Dmax



Nota. Adaptado de "Heurística de Intercambio 2Opt Best Improvement_r y Nivel de Eficacia de las soluciones del Problema del Agente Viajero Simétrico" (p.68), Astoquillca y Berger, 2023, *Revista Peruana de Computación y Sistemas*, 5(1).

Donde:

- Para un valor de Costo Nuevo = 5, Costo Actual = 8, Dmax $_{Inicial}$ = 0. Se tiene: 5-8 <0 → Se realiza el intercambio de nodos, y se actualiza el parámetro Dmax $_{Actual}$ = 5 - 8, cuyo valor absoluto representa la distancia relativa entre el *Costo Nuevo* y el *Costo Actual.*

- Para la siguiente iteración con Dmax = -3, se buscó un valor de *Costo Nuevo - Costo Actual* menor a -3 (nuestra nueva cota para el indicador de intercambio).

- Notar que si -3 < Costo Nuevo – Costo Actual < 0, el intercambio de nodos no mejora la distancia relativa medida por el indicador.

En la Figura 5 se observa que cada valor de Dmax que se actualiza está asociado a un determinado intercambio de nodos, por lo cual se tuvo tres posibles intercambios de nodos (para Dmax=-3, Dmax=-5, Dmax=-6). Ahora, cada una de estas posibles selecciones de intercambios pueden conducir a una dirección distinta de búsqueda de soluciones; por ello, existen distintas políticas para elegir un determinado intercambio 2Opt. A continuación, se presentan dos de ellas, las cuales fueron definidas por Da Costa, Rhuggenaath, Zhang y Akcay (2020):

- *First Improvement (FI*): Selecciona el primer intercambio que mejora el costo actual de la ruta construida, medido en base a la mejora del indicador Dmax. Para el caso de la Figura 5, se elige el intercambio 2Opt que genera la actualización de Dmax = -3.

- *Best Improvement (BI):* Selecciona el intercambio de nodos que produce la mejor mejora del costo actual de la ruta construida, medida con base en la mejora del indicador Dmax. Para el caso de la Figura 5, se elige el intercambio 2Opt que genera la actualización de Dmax = - 6 (más negativo).

De las dos políticas descritas, para la presente investigación a la versión del Intercambio 2Opt que utiliza:

- El First Improvement la denominamos 2OptFI, utilizado por Pérez (2011).

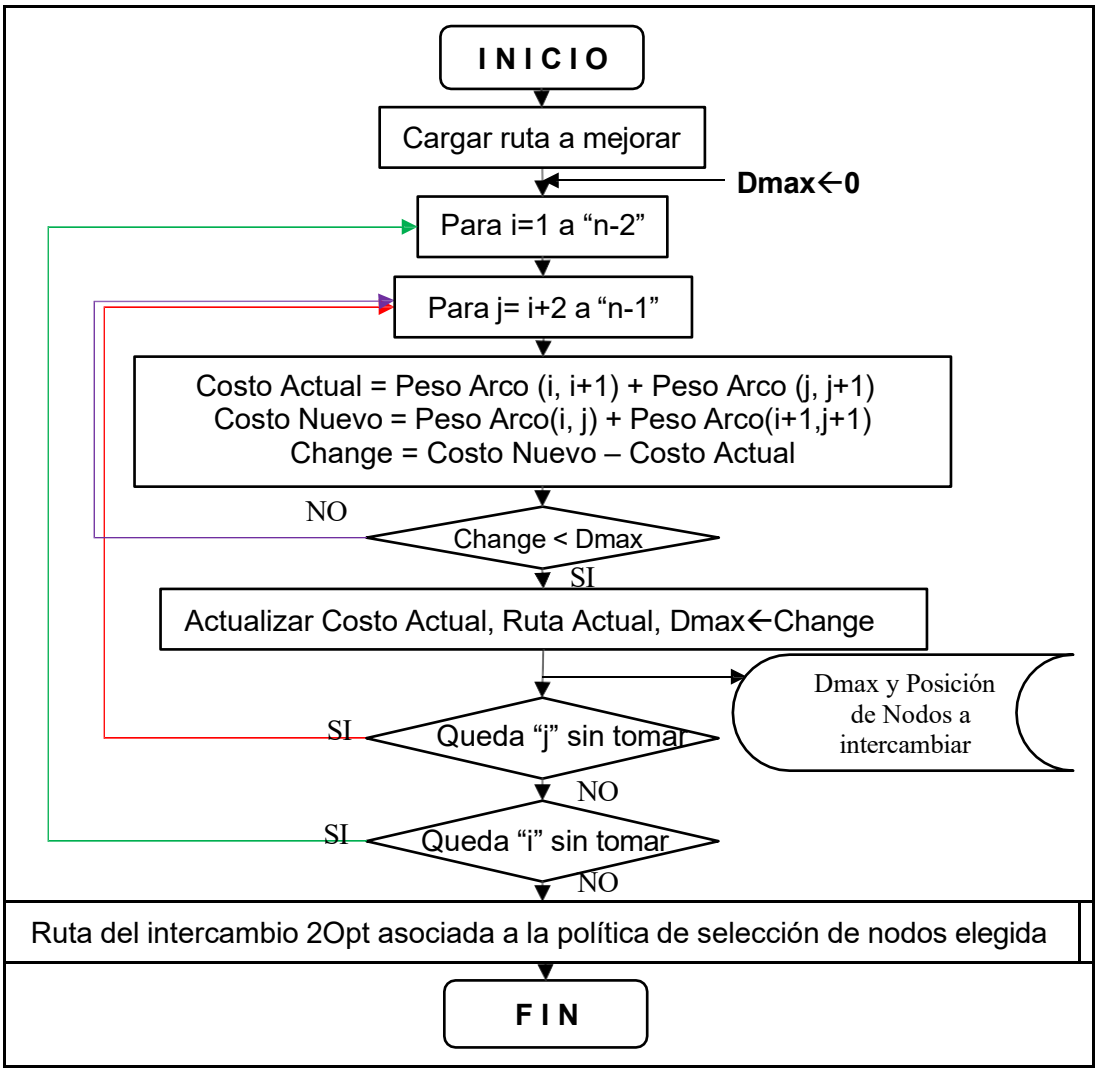
- El Best Improvement la denominamos 2OptBI, como una alternativa posible para mejorar los resultados.

En el inciso “c” del numeral “3.15.5 Reflexionar”, se determina que la Política Best Improvement proporciona mejores resultados que First Improvement para las instancias del TSP revisadas. Por ello, se aplicó dicha política en la presente investigación.

A continuación, se presenta el Diagrama de Flujo del 2OptBI en la Figura 6.

Figura 6

Diagrama de flujo de la heurística 2OptBI para una ejecución



2.2.14. Ejemplo de heurística 2Opt con política Best Improvement

Se presenta el siguiente e*jemplo de Problema del Agente Viajero*, planteándose la existencia de ocho ciudades a visitar, de tal manera que se pase por única vez por cada una de ellas, regresando a la ciudad inicial, de modo tal que la distancia recorrida en todo el viaje sea la menor posible. Además, se debe iniciar el recorrido en la "ciudad 1", y la "matriz de costos" asociada está dada en la Figura 7, cuyos elementos representan las distancias en kilómetros para ir entre dos ciudades.

Figura 7

Matriz de costos para ejemplo de grafo con ocho ciudades

Ciudad Inicial \ Ciudad Final	1	2	3	4	5	6	7	8
1	0	2	5	5	11	12	5	8
2	3	0	5	3	4	4	19	9
3	5	4	0	2	11	12	9	6
4	1	4	7	0	3	4	10	5
5	2	4	2	5	0	31	19	7
6	3	6	4	2	7	0	4	13
7	6	6	7	18	3	2	0	10
8	3	4	20	17	3	11	12	0

Se inició el proceso de mejoramiento a partir de una Ruta Inicial (ruta del Vecino más Cercano), con valor inicial Dmax = 0 en cada ejecución, que se actualiza en cada iteración, reiniciándolo en “0” para la siguiente ejecución:

Ruta Óptima ← Ruta Inicial (Vecino más Cercano)
Costo Óptimo ← Costo Ruta Inicial

Estos valores se actualizan en cada ejecución, según se cumpla el Criterio para realizar el Intercambio 2OptBI.

Ruta Óptima (VMC): 1→2→4→5→3→8→6→7→1, Costo Óptimo←37

- ***1º Ejecución Heurística 2OptBI (Resultado: Intercambiar 6 y 7)***

Ruta Actual (Óptima): 1→2→4→5→3→**8**→**6**→**7**→**1**, Costo Óptimo←37

Se inicia con el valor del indicador de intercambio: Dmax = 0.

Se presenta la Primera Iteración (intercambiar 2 y 8) donde: Costo Nuevo - Costo Actual < 0

*Criterio: [d(1,**8**) + d(**2**,6)] – [d(1,**2**)+d(**8**,6)] = [8+4] – [2+11] = -1 < 0*

Para la Segunda Iteración, ahora Dmax = -1. Se plantea intercambiar 6 y 7, de tal manera que:

*Criterio: [d(8,**7**)+d(**6**,1)] – [d(8,**6**)+d(**7**,1)] =[12+3] – [11+6] = -2 < -1*

Para la Tercera Iteración, ahora Dmax = -2, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 2, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo, es decir, intercambiar nodos 6 y 7.
Por tanto, la Ruta Óptima y Costo Total se actualizaron:

Ruta Óptima 1→2→4→5→3→**8**→**7**→**6**→**1**, **Costo Óptimo** ← 33

- ***2º Ejecución Heurística 2OptBI (Resultado: Intercambiar 4 y 6)***

Ruta Actual (Óptima): 1→**2**→**4**→5→3→8→7→**6**→**1**, Costo Óptimo←33

Se reinicia con el valor del indicador de intercambio: Dmax = 0.

Se presenta la Primera Iteración (intercambiar 4 y 6) donde: Costo Nuevo - Costo Actual < 0

*Criterio: [d(2,**6**)+d(**4**,1)] - [d(2,**4**)+d(**6**,1)] = [4+1] - [3+3] = -1 < 0*

Se actualiza en valor de Dmax = -1, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 1, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 4 y 6. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio:

Ruta Óptima 1→**2**→**6**→5→3→8→7→**4**→**1**, **Costo Total** ← 52

De manera análoga para el resto de las ejecuciones.

- ***3º Ejecución Heurística 2OptBI (Resultado: Intercambiar 5 y 7)***

Ruta Actual (Óptima): 1→2→**6**→**5**→3→8→**7**→**4**→1, Costo Óptimo ←52

*Dmax=0, Como: [d(1,**7**)+d(**2**,4)] - [d(1,**2**)+d(**7**,4)]=[5+3]-[2+18] = -12<0*
*Dmax=-12,Como:[d(6,**7**)+d(**5**,4)]-[d(6,**5**)+d(**7**,4)]=[4+5]-[7+18]=-16<-12*

Se actualiza el valor Dmax = -16, de modo tal que al realizar sucesivas iteraciones y al no encontrar intercambios que permitan que Dmax < -16, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 5 y 7. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→2→**6**→**7**→3→8→**5**→**4**→1**, Costo Óptimo**←32

- ***4º Ejecución Heurística 2OptBI (Resultado: Intercambiar 3 y 5)***

Ruta Actual (Óptima): 1→2→6→**7**→**3**→8→**5**→**4**→1**,** Costo Óptimo←32

Dmax=0. Como *[d(2,**5**)+d(**6**,4)] - [d(2,**6**)+d(**5**,4)]=[4+2] - [4+5]= -3 < 0*
Dmax=*-3, Como [d(7,**5**)+d(**3**,4)]-[d(7,**3**)+d(**5**,4)]=[3+2] - [7+5]= -7 < -3*

Se actualiza en valor de Dmax = -7, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 7, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 3 y 5. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→2→6→**7**→**5**→8→**3**→**4**→1**, Costo Óptimo**←43

- ***5º Ejecución Heurística 2OptBI (Resultado: Intercambiar 6 y 8)***

Ruta Actual (Óptima): 1→**2**→**6**→7→5→**8**→**3**→4→1**,** Costo Óptimo←43

Dmax=0. Como *[d(1,**8**)+d(**2**,3)] - [d(1,**2**)+d(**8**,3)]=[8+5]-[2+20]= -9<0*
Dmax=*-9. Como [d(2,**6**)+d(**8**,3)]-[d(2,**8**)+d(**6**,3)]=[9+4]-[4+20]= -11<-9*

Se actualiza en valor de Dmax = -11, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 11, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 6 y 8. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→**2**→**8**→7→5→**6**→**3**→4→1**, Costo Óptimo**←64

- ***6º Ejecución Heurística 2OptBI (Resultado: Intercambiar 5 y 7)***

  Ruta Actual (Óptima): 1→2→**8**→**7**→**5**→**6**→3→4→1**,** Costo Óptimo←64

  Dmax= 0. Como *[d(1,**5**)+d(**2**,6)]-[d(1,**2**)+d(**5**,6)]=[11+4]-[2+31]= -18<0*
  Dmax= *-18. Como [d(2,**5**)+d(**8**,6)] - [d(2,**8**)+d(**5**,6)] = -25 < -18*
  Dmax= *-25. Como [d(8,**5**)+d(**7**,6)]-[d(8,**7**)+d(**5**,6)]= -38 < -25*

  Se actualiza en valor de Dmax = -38, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 38, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 5 y 7. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→2→**8**→**5**→**7**→**6**→3→4→1**, Costo Óptimo**←42

- ***7º Ejecución Heurística 2OptBI (Resultado: Intercambiar 8 y 5)***

  Ruta Actual (Óptima): 1→**2**→**8**→**5**→**7**→6→3→4→1**,** Costo Óptimo←42

  Dmax= *0, Como [d(2,**5**)+d(**8**,7)]-[d(2,**8**)+d(**5**,7)]=[4+12]-[9+19]= -12<0*

  Se actualiza el valor de Dmax = -12, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < - 12, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 5 y 8. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→**2**→**5**→**8**→**7**→6→3→4→1**, Costo Óptimo**←34

- ***8º Ejecución Heurística 2OptBI (Resultado: Intercambiar 5 y 6)***

  Ruta Actual (Óptima): 1→**2**→**5**→8→7→**6**→**3**→4→1**,** Costo Óptimo←34

  Dmax=*0, Como [d(2,**6**)+d(**5**,3)]-[d(2,**5**)+d(**6**,3)]=[4+2]-[4+4] = -2 < 0*

  Se actualiza en valor de Dmax = -2, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que

Dmax < -2, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 5 y 6. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→**2**→**6**→8→7→**5**→**3**→4→1**, Costo Óptimo**←39

- ***9º Ejecución Heurística 2OptBI (Resultado: Intercambiar 8 y 7)***

Ruta Actual (Óptima): 1→2→**6**→**8**→**7**→**5**→3→4→1**,** Costo Óptimo←39

Dmax=*0, Como [d(2,**8**)+d(**6**,7)]-[d(2,**6**)+d(**8**,7)] = [9+4]-[4+12] = -3<0*
Dmax= *-9, Como [d(6,**7**)+d(**8**,5)]-[d(6,**8**)+d(**7**,5)]=[4+3]-[13+3] = -9<-3*

Se actualiza en valor de Dmax = -9, donde al realizar sucesivas iteraciones y al no encontrar intercambios los cuales permitan que Dmax < -9, nos quedamos con el intercambio cuyo indicador Dmax resultó ser más negativo: intercambiar nodos 7 y 8. Por tanto, la Ruta Óptima y Costo Total se actualizaron a la ruta asociada a dicho intercambio: **Ruta Óptima** 1→2→**6**→**8**→**7**→**5**→3→4→1**, Costo Óptimo**←28

- ***10º Ejecución Heurística 2OptBI***

Ruta Óptima 1→2→6→8→7→5→3→4→1**,** Costo Óptimo←28

Dmax = 0. Al realizar las iteraciones de los posibles intercambios entre los nodos, no se encontró un intercambio que tal que Dmax < 0.
Por tanto, a partir de la décima ejecución se encontró una convergencia de la variable Costo Óptimo a 28 y cuya ruta se mantiene tal como:

Ruta Óptima 1→2→6→8→**7**→5→3→4→1**, Costo Óptimo**←28

Por tanto, para este caso se consideró como *Criterio de Parada* el realizar 10 ejecuciones a la heurística 2OptBI, debido a que, a partir de dicha cantidad, no se encuentra mejoría. A continuación, el comportamiento de la solución encontrada en cada ejecución de la heurística 2OptBI, hasta un total de 50 ejecuciones, es representado en la Figura 8.

Figura 8

Evolución de solución según número de ejecuciones 2OptBI



Nota. Para el ejemplo del Problema del Agente Viajero para 8 ciudades en 2.2.14.

En la Figura 8 se puede apreciar que el “Costo de la Ruta Inicial”, obtenida a través del algoritmo del Vecino más Cercano, es de 37. A partir de esta ruta inicial se llevaron a cabo ejecuciones sucesivas de la heurística 2OptBI sobre la ruta mejorada por la anterior previa, continuando este proceso hasta que no se pudo obtener una mejora adicional en la solución. También se pudo notar que, aunque en algunas ejecuciones la solución empeoró, a medida que se continuó ejecutando la versión del algoritmo 2OptBI, la solución logró mejorar progresivamente hasta estabilizarse en una solución superior a las anteriores; lo cual valida lo indicado por Gutiérrez (2018), quien nos menciona que los movimientos que realiza el algoritmo 2Opt no necesariamente mejoran la solución, llegando a empeorarla entre un par de ejecuciones, pero al realizar más ejecuciones, la solución final sí mejora.

2.2.15. Sacrificio Cortoplacista Adaptativo (SCA)

Pérez (2011), en su búsqueda por mejorar la calidad de las soluciones construidas por el pensamiento del Vecino más Cercano, realizó un análisis tomando al pensamiento sistémico como fuente de inspiración para la modificación del criterio de búsqueda de dicho algoritmo, al cual Pérez (2011)

considera miope, por motivo de tener un criterio de búsqueda que solo se preocupa en tomar la mejor decisión en el corto plazo, sin preocuparse en cómo tales decisiones afectan a largo plazo en la construcción de la ruta final.

Basándose en principios definidos por Senge (2010), según los cuales pequeños cambios pueden producir grandes efectos en el largo plazo cuando son aplicados en ciertas ubicaciones, las cuales son denominadas "*Puntos de Apalancamiento".*

Pérez (2011) define el "*Punto de Apalancamiento"* como aquel nodo en el cual se aplica un criterio que modifica mínimamente el pensamiento del Vecino más Cercano, con el objetivo de mejorar su criterio de búsqueda para explorar nuevas zonas del espacio solución y, así, construir mejores soluciones.

Tal criterio, denominado "Sacrificio Cortoplacista Adaptativo (SCA)", consistió en que, desde el punto de apalancamiento, elegir ir al segundo mejor nodo en términos de costo; para posterior a dicha aplicación con el resto de los nodos "adaptarnos" al criterio de búsqueda del pensamiento del Vecino más Cercano. A tal criterio, Pérez (2011) lo denominó Sacrificio Cortoplacista de Nivel 2, siendo el Sacrificio Cortoplacista de nivel 1 el propio pensamiento del Vecino más Cercano. Así, al indicar un Nivel 3, se estaría eligiendo el tercer mejor nodo desde un punto de apalancamiento. Tal sacrificio se realiza con la esperanza de habilitar arcos que antes estaban bloqueados y encontrar mejores soluciones que antes no se exploraron con el pensamiento de la heurística original. Por lo anterior, Pérez (2011) afirma que, por medio del Sacrificio Cortoplacista Adaptativo, al ofrecer buenas soluciones en tiempos razonables, se constituye como evidencia para seguir utilizando al pensamiento sistémico como fuente de inspiración para desarrollar nuevas metaheurísticas.

2.2.16. Ejemplo de Sacrificio Cortoplacista Adaptativo (SCA)

Continuando con el ejemplo propuesto en el numeral 2.2.13, el cual muestra un Problema de Agente Viajero para ocho ciudades (ver Figura 7), se consiguió una ruta inicial, en base al criterio del Vecino más Cercano, la cual consistió en la siguiente: 1-►2-►4-►5-►3-►8-►6-►7-►1, cuyo costo total por recorrido fue de 37 km.

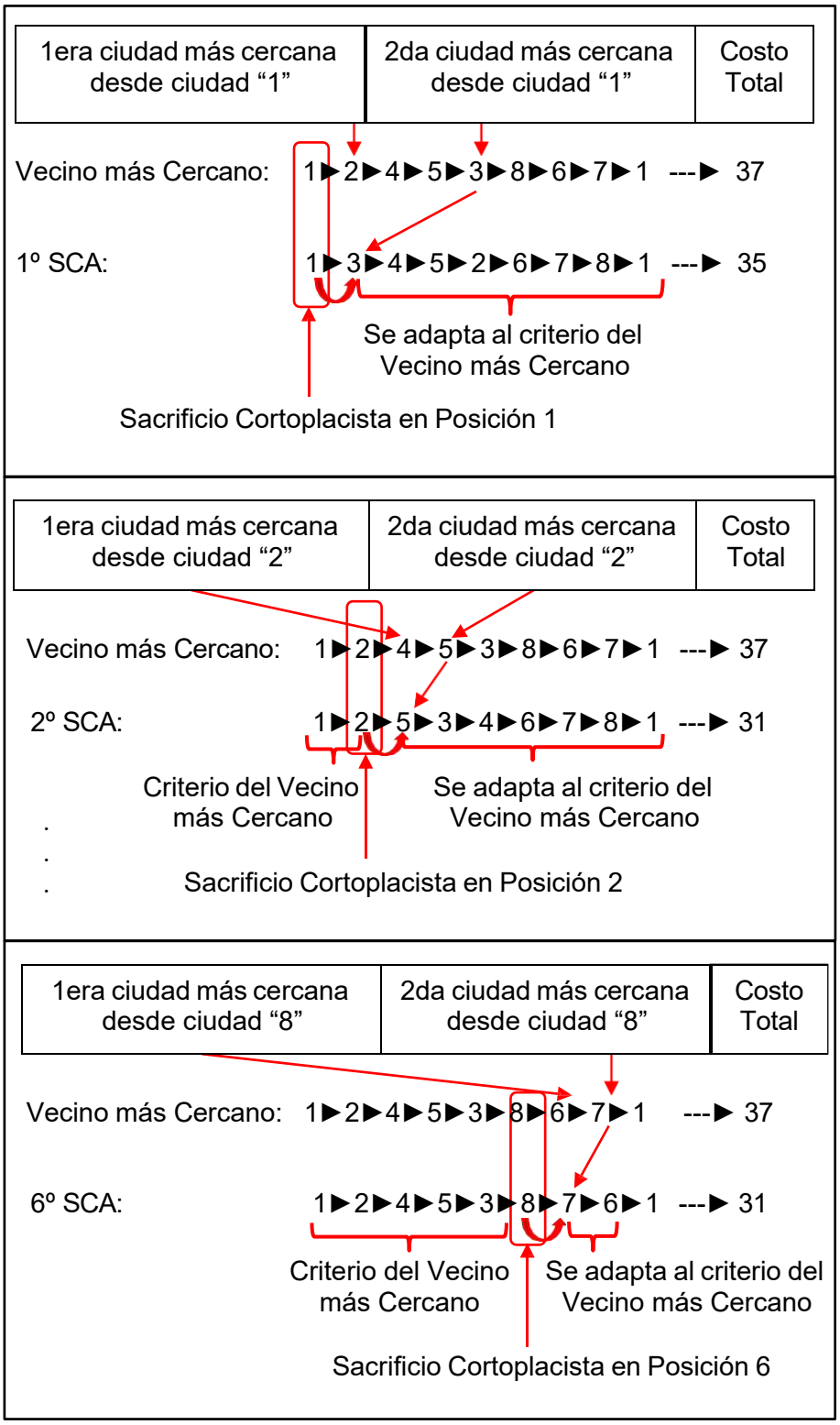
A partir de la Ruta Inicial (Vecino más Cercano), desde una ciudad determinada se elige ir a la segunda ciudad más cercana (Sacrificio Cortoplacista) y para el resto de las ciudades se sigue aplicando el pensamiento del Vecino más Cercano (ir a la ciudad más cercana). Con dicho Sacrificio Cortoplacista se consiguió la construcción de una ruta asociada a la posición de la ciudad donde se realizó el sacrificio. Posteriormente, se repitió el proceso de partir de la ruta inicial, escogiendo otra ciudad aún no tomada, desde la cual se eligió ir a la segunda ciudad más cercana (Sacrificio Cortoplacista) para volver a aplicar el criterio del Vecino más Cercano al resto de ciudades y así se construyó una ruta nueva asociada a la ciudad donde se realizó dicho sacrificio.

En general, para un grafo de orden “n”, solo se puede realizar un número máximo de “n-2” Sacrificios Cortoplacistas (una ruta por cada sacrificio realizado), tomándose en cuenta que desde el nodo “n-1” solo se podría seleccionar un único nodo a donde ir, por tanto, no se podría aplicar el Sacrificio Cortoplacista.

A continuación, un ejemplo del funcionamiento del Sacrificio Cortoplacista para 5 ciudades se representó en la Figura 9.

Figura 9

Funcionamiento del Sacrificio Cortoplacista Adaptativo (SCA)



Nota. Figura basada en el ejemplo de 5 ciudades de Pérez (2011). Repositorio Institucional de la Universidad Nacional de Colombia.

2.2.17. Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt)

Pérez (2011), en su búsqueda por mejorar la calidad de las soluciones construidas por su denominado Sacrificio Cortoplacista Adaptativo (SCA), combinó dicho criterio con la heurística 2Opt, con lo cual generó un algoritmo híbrido, al cual denominó como SCA_2Opt.

Los pasos para ejecutar el SCA_2Opt se describen a continuación:

a. **Paso 1.** Obtener ruta con el Vecino más Cercano

b. **Paso 2.** Aplicar Sacrificio Cortoplacista Adaptativo a nodo “t”, para “t” aún no evaluado y 2 < t < N-2. Donde “N” es el orden del grafo. Cuando ya no se disponga de más valores para “t” ir al paso 8.

c. **Paso 3.** Para el resto de los nodos, adaptar el criterio para la construcción de la ruta al pensamiento del Vecino más Cercano.

d. **Paso 4.** Guardar costo de ruta construida para “t” y actualizar la ruta y el mejor costo.

e. **Paso 5.** Se calcula valor de Change = Costo Nuevo – Costo Óptimo

f. **Paso 6.** Si “Change < Dmax”, se actualiza Costo Óptimo, ruta actual, Dmax ← Change y se mejora la ruta con la heurística 2Opt

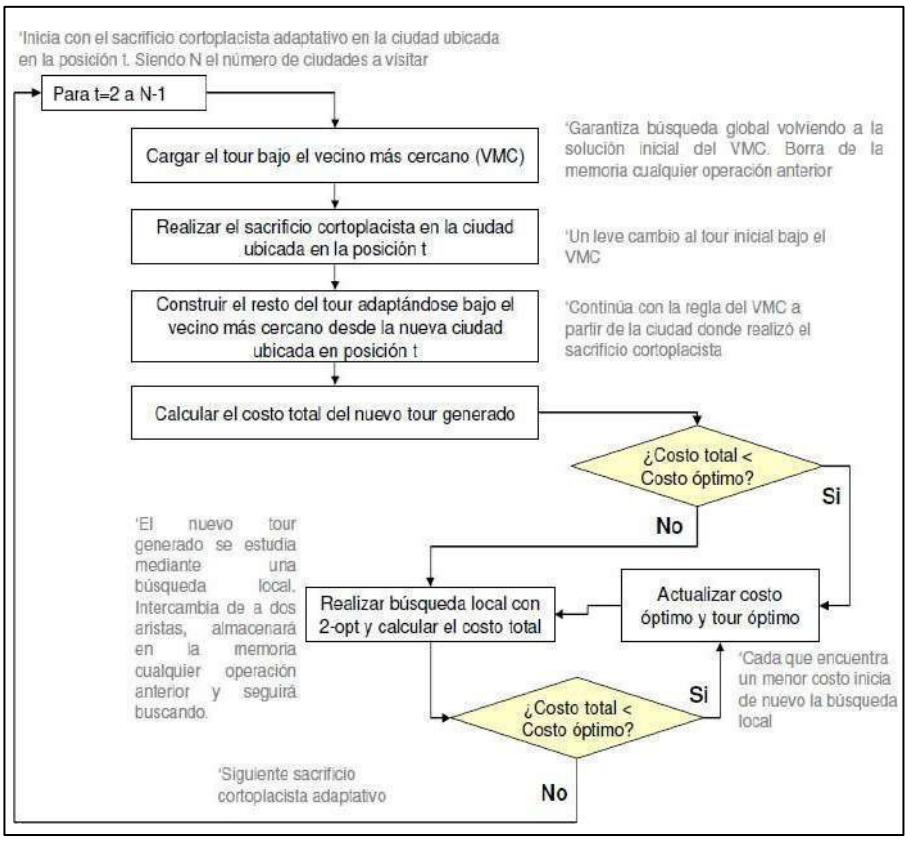
g. **Paso 7.** Evaluar si en la ruta mejorada se cumple que Costo Total < Costo Óptimo. Si se cumple aplicar otra vez heurística 2Opt e ir al paso 5, caso contrario ir al paso 2.

h. **Paso 8.** Elegir ruta asociada al último valor de Costo Óptimo.

Para mejor ilustración, se presenta el diagrama de flujo del SCA_2Opt (ver Figura 10), propuesto por Pérez (2011):

Figura 10

Diagrama de flujo del SCA_2Opt



Nota. Adaptado del diagrama de flujo del SCA_2Opt [Figura], Pérez (2011).

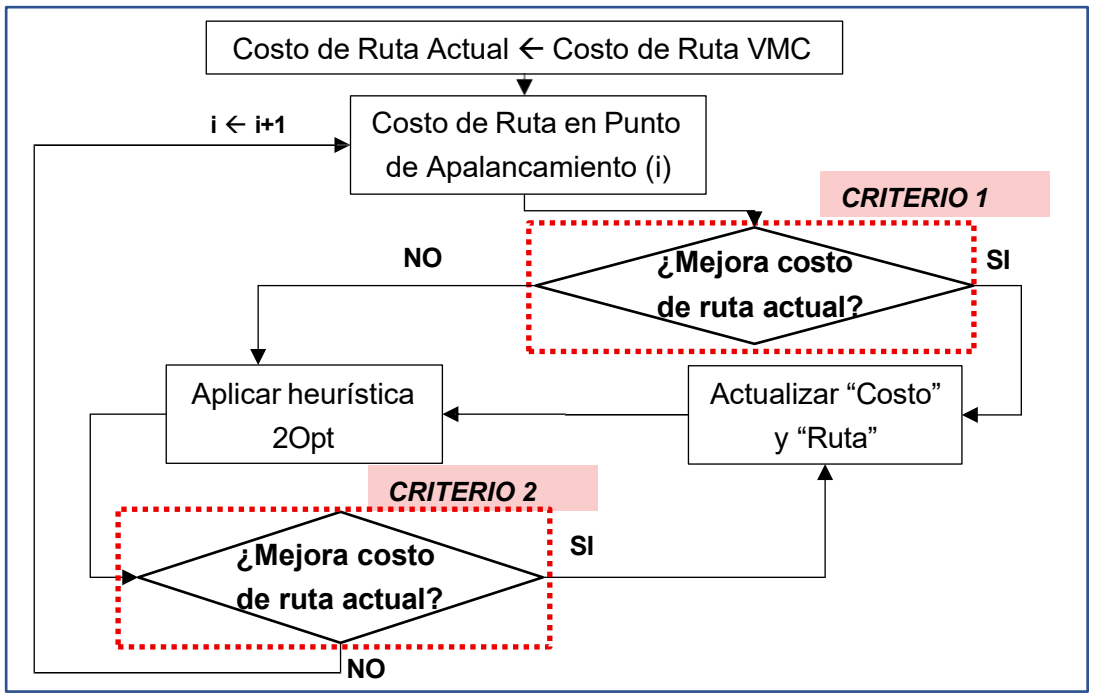
En la Figura 10, al enfocarnos en las dos condicionales, se reorganizó la estructura para resumirla en la Figura 11, tal que:

- Criterio 1. Evaluar el *Costo Óptimo* con el *Costo Actual* proveniente de una Ruta construida con el Apalancamiento.

- Criterio 2. Evaluar el *Costo Óptimo* con el *Costo Actual* proveniente de la ruta mejorada con la heurística 2Opt. De tal manera, que al no cumplirse con el Criterio 2, se procede a generar una nueva ruta proveniente de un nuevo apalancamiento, tal como se indica en la Figura 11.

Figura 11

Criterios de decisión en el SCA_2Opt



Nota. Adaptado de diagrama de flujo del SCA_2Opt [Figura], Pérez (2011).

2.2.18. Ejemplo de Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt)

A) Para el punto de apalancamiento posición 1

Paso 1. Se aplicó el pensamiento del Vecino más Cercano en el numeral 2.2.13, en la Matriz de Costos (ver Figura 7). Así, se construyó la ruta indicada en la Figura 12, donde el costo de dicha ruta pasa a ser el Costo Óptimo (temporalmente).

Figura 12

Ruta determinada por la heurística del "Vecino más Cercano"

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia
Ciudad(i)	1	2	4	5	3	8	6	7	0	Total VMC
Distancia (i, i+1)		2	3	3	2	6	11	4	6	37

Es decir, Ruta Óptima: 1→2→4→5→3→8→6→7→1, Costo Óptimo ← 37

Paso 2. Se aplicó el pensamiento del Sacrificio Cortoplacista, con punto de apalancamiento en la Posición 1: desde la ciudad 1, se eligió la segunda mejor alternativa, la cual consistió en ir a la ciudad 3. Es decir: Ruta Parcial 1→ 3.

Paso 3. Luego de aplicado el Sacrificio Cortoplacista (Posición 1), se continuó con la construcción de la ruta, donde se siguió el pensamiento del Vecino más Cercano para el resto de las ciudades (ver Figura 13).

Figura 13

Ruta para el punto de apalancamiento en posición 1

Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4							
Distancia(i,i+1)		5	2							7 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5						
Distancia(i,i+1)		5	2	3						10 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5	2					
Distancia(i,i+1)		5	2	3	4					14 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5	2	6				
Distancia(i,i+1)		5	2	3	4	4				18 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5	2	6	7			
Distancia(i,i+1)		5	2	3	4	4	4			22 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5	2	6	7	8		
Distancia(i,i+1)		5	2	3	4	4	4	10		32 (parcial)
Posición(i)	1	**2**	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:1)
Nodo(i)	1	3	4	5	2	6	7	8	1	
Distancia(i,i+1)		5	2	3	4	4	4	10	3	35 (Costo Total)

El costo de la ruta construida para el punto de “Apalancamiento = Posición 1” fue Costo Total ← 35.

1er Criterio: Costo Total (35) < Costo Óptimo (37), se actualizó:

Ruta Óptima 1→3→4→5→2→6→7→8→1, Costo Óptimo ← 35

Paso 4: A partir de la Ruta determinada en la Figura 13, se procedió a aplicar la heurística 2Opt y se intercambian los Nodos 3 y 7.

Ruta óptima 1→3→4→5→2→6→7→8→1, Costo óptimo← 35

Ruta Total 1→7→4→5→2→6→3→8→1, Costo Total ← 47

2do Criterio: Como Costo Total (47) > Costo Óptimo (35), no se volvió a aplicar la heurística 2Opt y se procedió a ir al Paso 5.

Paso 5. Para el punto de apalancamiento “t=2” (“t” aún no seleccionado), se aplicó el Paso 2, Paso 3 y Paso 4, se repitió el proceso hasta no tener más valores de “t” a seleccionar (1 ≤ t ≤ n-2, donde “n” es el orden del grafo).

B) Para el punto de apalancamiento posición 2

Se repitieron los pasos 1 al 5, de tal manera que se construyó la Figura 14.

Figura 14

Ruta para el punto de apalancamiento en posición 2

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (Punto de Apalancamiento:2) 31 (Costo Total)
Nodo(i)	1	2	5	3	4	6	7	8	1	
Distancia(i, i+1)		2	4	2	2	4	4	10	3	

1er Criterio: Como Costo Total (31) < Costo Óptimo (35), se actualizó:

Ruta Óptima 1→7→4→5→2→6→3→8→1, Costo Óptimo←31

2do Criterio: Se aplicó la heurística 2Opt, donde no se logró mejorar el *Costo Óptimo*, por lo cual se dejó de aplicar una nueva ejecución de la heurística 2Opt. Por tanto, la *Ruta Óptima* y el *Costo Óptimo* se mantuvieron y se procedió a evaluar el siguiente punto de apalancamiento.

Ruta Óptima 1→7→4→5→2→6→3→8→1, Costo Óptimo←31

C) Para el punto de apalancamiento en posición 3

Se repitieron los pasos 1 al 5, de tal manera que se construyó la Figura 15.

Figura 15

Ruta para el punto de apalancamiento en posición 3

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (punto de apalancamiento:3)
Nodo(i)	1	2	4	6	3	8	5	7	1	
D (i, i+1)		2	3	4	4	6	3	19	6	47 (Costo Total)

1er Criterio: Como Costo Total (47) > Costo Óptimo (31), se procedió a aplicar la heurística 2Opt para buscar mejorar la ruta.

Se intercambiaron los nodos 5 y 6, donde se actualizó:

Ruta Total 1→2→4→**5**→3→8→**6**→7→1, Costo Total ← 37

2do Criterio: Como en la Ruta mejorada se tuvo que el "Costo Total (37) > Costo Óptimo (31)", no se logró mejorar el *Costo Óptimo*, por lo cual se dejó de aplicar una nueva ejecución de la heurística 2Opt. Por tanto, la *Ruta Óptima* y el *Costo Óptimo* se mantuvieron y se procedió a evaluar el siguiente punto de apalancamiento.

Ruta Óptima 1→7→4→5→2→6→3→8→1, Costo Óptimo←31

D) Para el punto de apalancamiento en posición 4

Se repitieron los pasos 1 al 5, de tal manera que se construyó la Figura 16.

Figura 16

Ruta para el punto de apalancamiento en posición 4

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (punto
Nodo(i)	1	2	4	5	8	6	3	7	1	de apalancamiento:3)
D (i, i+1)		2	3	3	7	11	4	9	6	45 (Costo Total)

1er Criterio: Como "Costo Total (45) > Costo Óptimo (31)", se procedió a aplicar la heurística 2Opt para buscar mejorar la ruta. Se intercambiaron los nodos 3 y 8, donde se actualizó:

Ruta Total 1→2→4→5→8→6→3→7→1, Costo Total ← 53

2do Criterio: Como para la Ruta mejorada "Costo Total (53) > Costo Óptimo (31)", tal que no se logró mejorar el *Costo Óptimo*, por lo cual se dejó de aplicar una nueva ejecución de la heurística 2Opt. Por tanto, se procedió a evaluar el siguiente punto de apalancamiento.

Ruta Óptima 1→7→4→5→2→6→3→8→1, Costo Óptimo←31

E) Para el punto de apalancamiento en posición 5

Se repitieron los pasos 1 al 5, de tal manera que se construyó la Figura 17.

Figura 17

Ruta para el punto de apalancamiento en posición 5

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (punto
Nodo(i)	1	2	4	5	3	7	6	8	1	de apalancamiento:3)
D (i, i+1)		2	3	3	2	9	2	13	3	37 (Costo Total)

1er Criterio: Como “Costo Total (37) > Costo Óptimo (31)”, se procedió a aplicar la heurística 2Opt para buscar mejorar la ruta. Se intercambiaron los nodos 4 y 6, donde se actualizó:

Ruta Total 1→2→6→5→8→4→3→7→1, Costo Total ← 50

2do Criterio: Como para la Ruta mejorada “Costo Total (50) > Costo Óptimo (31)”, no se logró mejorar el *Costo Óptimo*. Por lo cual, se dejó de aplicar una nueva ejecución de la heurística 2Opt:

Ruta Óptima 1→7→6→5→2→4→3→8→1, Costo Óptimo←31

F) Para el punto de apalancamiento en posición 6

Se repitieron los pasos 1 al 5, de tal manera que se construyó la Figura 18.

Figura 18

Ruta para el punto de apalancamiento en posición 6

Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (Punto de Apalancamiento:6)
Nodo(i)	1	2	4	5	3	8	7	6	1	
D (i, i+1)		2	3	3	2	6	12	2	3	33 (Costo Total)

1er Criterio: Como “Costo Total (33) > Costo Óptimo (31)”, se procedió a aplicar la heurística 2Opt para buscar mejorar la ruta. Se intercambiaron los nodos 4 y 6, donde se actualizó:

Ruta Total 1→2→6→5→8→4→3→7→1, Costo Total ← 52

2do Criterio: Como para la ruta mejorada “Costo Total(52) > Costo Óptimo(31)”, de tal forma que no se logró mejorar el *Costo Óptimo*, por lo cual se dejó de aplicar una nueva ejecución de la heurística 2Opt. Y, al no tener más puntos de apalancamientos, el algoritmo finalizó con: Ruta Óptima 1→7→4→5→2→6→3→8→1, Costo Óptimo←31

2.2.19. Factor "r" en el Sacrificio Cortoplacista Adaptativo 2Opt

El parámetro Dmax, al iniciar el algoritmo, toma el valor de cero, pero en el SCA_2Opt, la aplicación de la heurística 2Opt fue incorporada a cada Sacrificio. Por lo cual, cuando se realiza un intercambio 2Opt, según el criterio de decisión, el valor de Dmax debe actualizarse a Dmax← Change cuando "Change < Dmax", o mantenerse en el último valor registrado de Dmax si "Dmax > Change" (Pérez, 2011).

La actualización del parámetro Dmax encuentra sustento debido a que, al encontrar un intercambio en el cual el resultado de "Costo Nuevo – Costo Actual*"* tome un valor negativo, por ejemplo -3, se realiza el intercambio 2Opt. Tal que, al aplicar otra vez la heurística 2Opt a una nueva ruta producto de un nuevo apalancamiento, se tiene que buscar intercambios que mejoren la diferencia actual entre el *Costo Nuevo y Costo Actual*. Es decir, se necesitan valores menores a "-3"; en caso contrario, se estaría realizando intercambios que no mejoran la solución actual. Para mayor detalle, ver Figura 5.

Se define el factor "r", como una perturbación que, según Pérez (2011), toma una expresión lineal igual a (1 + r). Tal que "r" pertenece al intervalo [0,1] y se adiciona al minuendo del criterio usado para realizar el intercambio 2Opt.

A continuación, se muestra el criterio del intercambio 2Opt_r:

> Change = (1 + r) × Costo Nuevo – Costo Actual, donde r ∈ [0,1].
>
> Sí Change < Dmax, se realiza el intercambio 2Opt y el valor de Dmax se actualiza según el proceso iterativo.

Para entender mejor lo indicado por Pérez (2011), nos centramos exclusivamente en la tabla contenida en la Figura 17 (la ruta construida con la aplicación del Sacrificio Cortoplacista en la Posición 5) y se aplica el criterio de "Intercambio 2Opt" para intercambiar nodos entre dos arcos, con la diferencia de la introducción del factor "(1 + r)" dentro de dicho criterio. Dicho procedimiento es presentado en la Figura 19.

Figura 19

Ejemplo de factor "r" incorporado al SCA_2Opt

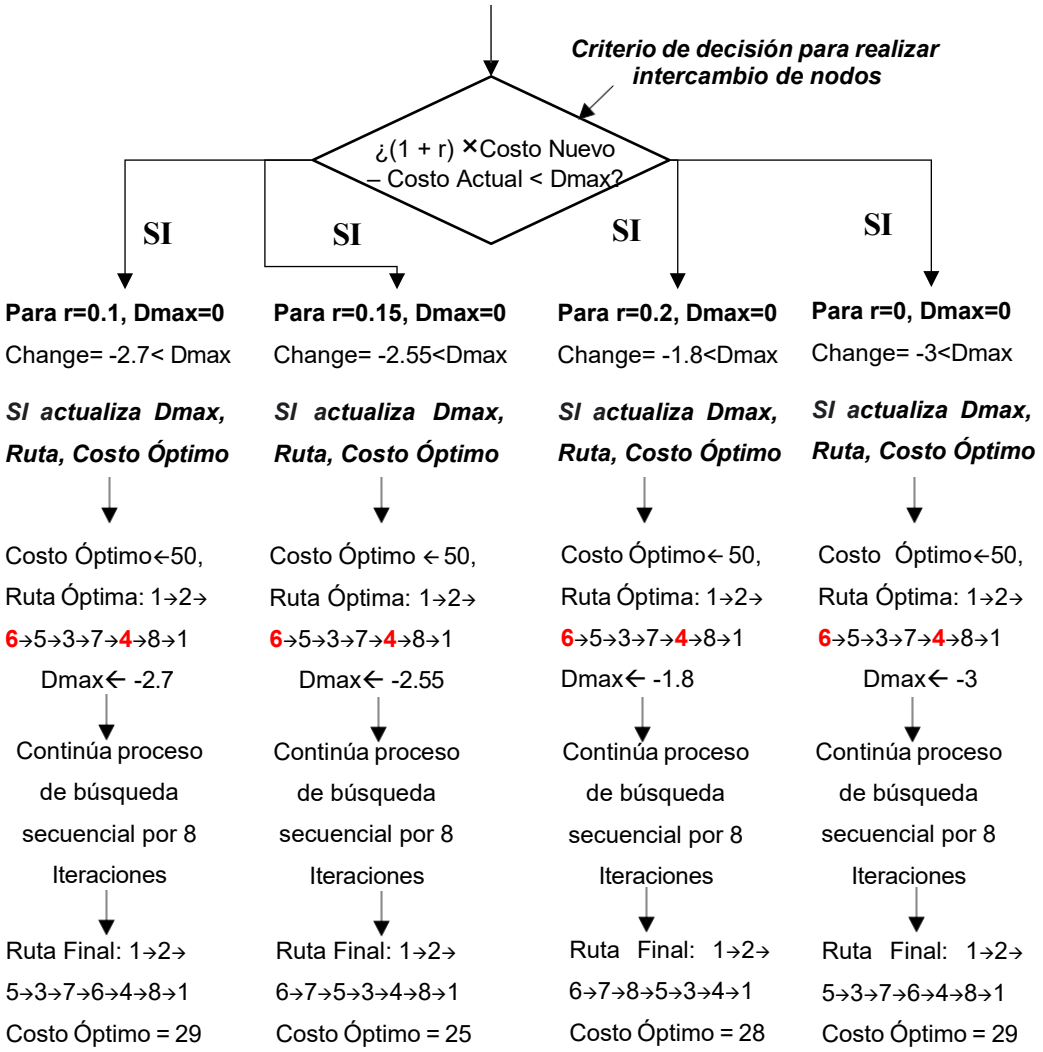
Posición(i)	1	2	3	4	5	6	7	8	9	Distancia VMC_SCA (punto
Nodo(i)	1	2	4	5	**3**	7	6	8	1	apalancamiento:5)
D (i, i+1)		2	3	3	2	9	2	13	3	37 (Costo Total)

Como "Criterio de Parada" no se usó el propuesto por Pérez (2011), sino una cantidad de ejecuciones igual a 8 para este ejemplo.

Intercambiar nodos **4** y **6**

- Costo Actual = Peso Arco (2, **4**) + Peso Arco (**6**, 8) = 2 + 4 = 6
- Costo Nuevo = Peso Arco (2, **6**) + Peso Arco (**4**, 8) = 2 + 1 = 3

Donde: Change = (1 + r) × Costo Nuevo - Costo Actual



De la Figura 19, tenemos que la diferencia del efecto del factor “r” para los casos descritos radica en “Dmax”, el cual es visto como un umbral que permite el intercambio o no entre nodos de los arcos evaluados en las próximas iteraciones, tal como se observa en la Tabla 2. Se debió determinar valores de “r” que produjeran efectos distintos entre sí para explorar el conjunto de soluciones factibles.

Tabla 2

Costo de ruta según el número de intercambio 2OptBI realizado según “r”.

	r = 0.1	**r = 0.15**	**r = 0**	**r = 0.2**
Vecino más Cercano (VMC)	37	37	37	37
Para 5to Apalancamiento	35	35	35	35
1er Intercambio 2Opt	50	50	50	50
2do Intercambio 2Opt	41	41	41	41
3er Intercambio 2Opt	35	32	35	45
4to Intercambio 2Opt	39	34	39	32
5to Intercambio 2Opt	40	25	40	43
6to Intercambio 2Opt	41	25	41	28
7mo Intercambio 2Opt	41	25	41	28
8vo Intercambio 2Opt	29	25	29	28

Nota 1. Los intercambios 2Opt son realizados introduciendo el factor “r”.
Nota. Para el ejemplo del Problema del Agente Viajero para 8 ciudades en 2.2.14.

En la Tabla 2, se puede observar que el valor r=0 y r= 0.1 producen el mismo efecto sobre la construcción de una ruta. Para el resto de los valores de “r”, el valor de las soluciones empieza a diferenciarse a partir del 3er Intercambio 2Opt, con lo cual cada valor de “r” realiza una búsqueda en un espacio distinto al otro, por lo cual se explora mejor el espacio de soluciones factibles.

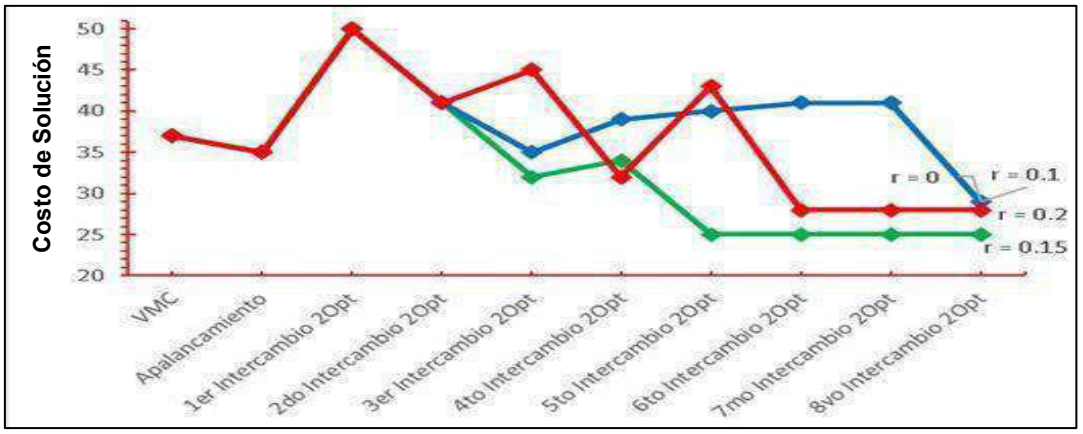
Para la elaboración de la Tabla 2, se observó que para r = 0.15 la mejor solución aparece en el 5to intercambio 2Opt, donde las soluciones de las

ejecuciones previas no son necesariamente mejores a la actual, lo cual muestra que no necesariamente se debe terminar la ejecución del algoritmo cuando no se encuentre mejora inmediata del costo de la ruta entre dos iteraciones sucesivas. Por lo cual, es preferible usar como "*Criterio de Parada"* una cantidad fija de ejecuciones.

A continuación, en la Figura 20 se muestra el comportamiento de las soluciones según el valor de "r" asignado de la Tabla 2, de tal manera que se visualiza una convergencia según aumentamos el número de ejecuciones.

Figura 20

Efecto del factor "r" en el SCA_2OptBI



Nota 1. En este caso se muestra el comportamiento para el 5to Apalancamiento
Nota 2. Para el ejemplo del Problema del Agente Viajero para 8 ciudades en 2.2.14.

En la Figura 20 se puede observar que la introducción del factor "r" dentro del criterio de intercambio 2OptBI permitió modificar ligeramente la dirección de búsqueda del algoritmo; donde, si bien al inicio pueden o no superar el umbral que habilita el intercambio de nodos, conforme se avanza en las ejecuciones, estas permiten pequeñas alteraciones las cuales construyen nuevas rutas en el caso del problema del agente viajero simétrico. Dichas nuevas soluciones, anteriormente no exploradas por medio del factor "r", posibilitan nuevas direcciones de búsqueda con la esperanza de encontrar la mejor solución disponible.

2.2.20. Eficacia Algorítmica

Antes de definir la *Eficacia Algorítmica*, se brinda la definición de algoritmo según Marcelo (2014), como el "método que describe la solución de un problema computacional mediante una serie de pasos precisos, definidos y finitos" (p.20). Tal que, normalmente, para expresarlo se usan herramientas como el diagrama de flujo o pseudocódigo.

Por su parte, Pérez (2011) utilizó el término *Eficacia Algorítmica* para referirse a la calidad de la solución determinada por el algoritmo. Desde esta perspectiva, mientras mejor sea la solución de un algoritmo en comparación con otros, mejor será su eficacia.

Por lo anterior, la *Eficacia* se definió como el grado de acercamiento de nuestra solución a la meta planteada.

Para la medición de la *Eficacia*, Pérez (2011) utilizó un indicador denominado "*Distancia Relativa",* el cual permite evaluar cuán cerca se encuentra la solución encontrada por nuestro algoritmo de la solución deseada (Óptima).

Dicho indicador es denominado por otros autores como "*Error Relativo",* el cual es expresado de la siguiente manera:

$$\underset{Relativo}{Error} = \frac{\underset{por\ el\ algoritmo\ evaluado}{Solución\ alcanzada} - Solución\ óptima\ TSPLIB}{Solución\ óptima\ TSPLIB}$$

Donde:

- La Solución Óptima TSPLIB es indicada en la librería TSPLIB.
- La solución alcanzada por el algoritmo evaluado es obtenida por la ejecución del algoritmo respectivo en cada instancia seleccionada.

Entre más pequeño sea el valor de este indicador, mayor será la "*Eficacia"* de la solución aportada por el algoritmo.

Tal indicador (Error Relativo) permite una medición de la distancia relativa respecto al óptimo.

Por lo cual, se adicionó el indicador "*Nivel de Eficacia"*, el cual se define como la proporción de instancias revisadas donde el algoritmo obtuvo la mejor eficacia y es cuantificado por medio de la siguiente expresión:

$$Nivel\ de\ Eficacia = \frac{Cantidad\ de\ instancias\ donde\ el\ algoritmo\ evaluado\ obtuvo\ la\ mejor\ Eficacia}{Cantidad\ total\ de\ instancias\ seleccionadas}$$

Tal que, el total de soluciones disponibles consistió en la mejor solución disponible de cada algoritmo revisado.

2.2.21. Eficiencia algorítmica medida en base a su complejidad

El término eficiencia hace referencia al uso de los recursos, donde entre menos recursos se use para alcanzar el objetivo, se dice que se es más eficiente. De tal manera que el término "*Eficiencia Algorítmica"*, según Guerequeta y Vallecillo (2019), nos permite medir el uso de los recursos computacionales que son necesarios para que un algoritmo pueda encontrar la solución a un problema.

Tal que, al usar el término tiempo, denotado por T(n), se hace referencia al número de instrucciones ejecutadas por el algoritmo; de tal manera que se tiene dos posibles estudios acerca del tiempo:

- *Medida Teórica o a priori*, la cual consiste en obtener una medida que acote (por encima o por debajo) el tiempo de ejecución del algoritmo. De tal manera, que podemos obtener una estimación del comportamiento del algoritmo independientemente del computador.

- *Medida Real o a posteriori,* la cual consiste en medir el tiempo de ejecución del algoritmo en un computador y para un tamaño de entrada específico.

A continuación, se presentan algunos conceptos necesarios para medir la eficiencia:

a. ***Principio de Invarianza.*** El cual nos indica que, si se tiene dos implementaciones (I_1 e I_2) con un tiempo $T_1(n)$ y $T_2(n)$, respectivamente, entonces el principio afirma que existe una constante real $c > 0$ y un número "n_0" / $\forall\ n \geq n_0$, se verifica que $T_1(n) \leq cT_2(n)$. Es decir, que el tiempo de dos implementaciones distintas para un mismo algoritmo varía en una constante multiplicativa (Guerequeta y Vallecillo, 2019).

b. ***Cota superior de complejidad – Notación O.*** Sea f: N→[0;∞). Se define un conjunto de funciones de orden O (Omicron) de f como (Guerequeta y Vallecillo, 2019, p.7):

$$O(f)= \{g: \mathbf{N}\rightarrow[0,\infty) / \exists c \in \mathbf{R}, c>0, \exists\ n_0 \in N \bullet g(n) \leq cf(n)\ \forall\ n \geq n_0\}$$

Diremos que una función $t:\mathbf{N}\rightarrow[0,\infty)$ es de orden O de f si $t \in O(f)$

La definición anterior nos permite entender intuitivamente que "t" tiene una cota superior por algún múltiplo de la función "f"/ $t \in O(f)$.

c. ***Propiedades de Notación O.*** A continuación, se indican algunas propiedades de la Notación O (también conocida como Big O, Gran O), mencionadas por Guerequeta y Vallecillo (2019), las cuales son las siguientes:

- Para cualquier función f se tiene que $f \in O(f)$.
- $O(f) = O(g) \Leftrightarrow f \in O(g)$ y $g \in O(f)$.
- Regla de la suma: Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1 + f_2 \in O(max(g,h))$
- Regla del producto: Si $f_1 \in O(g)$ y $f_2 \in O(h) \Rightarrow f_1 \cdot f_2 \in O(g \cdot h)$

d. Operaciones Elementales. ***"***Es la operación que realiza un computador en un tiempo acotado por una constante" (Tocto, 2012, p.16). Entre ellas tenemos:

- Operaciones aritméticas básicas
- Asignaciones a variables
- Llamadas a funciones, procedimientos y retorno desde ellos.
- Comparaciones lógicas (igualdad, desigualdad, etc.)
- Acceso a estructuras indexadas básica (vectores y matrices)

Cada una de las operaciones mencionada se contabiliza como 1 Operación Elemental (Guerequeta & Vallecillo, 2019).

e. **Medición del Tiempo.** El tiempo de ejecución de un algoritmo se expresa mediante una función que mide el número de operaciones elementales que realiza un algoritmo (Guerequeta & Vallecillo, 2019) para un tamaño de entrada dado, al cual se le denomina función de complejidad temporal, que permitirá el estudio de la complejidad del algoritmo (Tocto, 2012).

f. ***Reglas generales para el cálculo del número de Operaciones Elementales.*** A continuación, se presentan algunas reglas tomadas de Guerequeta y Vallecillo (2019):

- El tiempo de 1 OE, se considera por definición de orden 1.
- El tiempo de ejecución de una secuencia o bloque de código del programa se calcula sumando los tiempos de ejecución de cada instrucción. Y por el principio de invarianza O(1)+O(1)+…+O(1)= O(1), de manera análoga para O(n)+O(n)+…+O(n) = O(n)

- El tiempo de ejecución de una sentencia "**If** C **then** S1 **Else** S2 **End**" es: $T = T(C) + \max\{T(S_1),T(S_2)\}$

- El tiempo de ejecución de un bucle "**While** C **DO** S **End**" es: $T = T(C) + (\#\text{iteraciones}) \times [T(S)+T(C)]$

- El tiempo de ejecución de un bucle **FOR** i=1 to n **Do** S **END** puede calcularse a partir de su equivalente i=1; **While** i ≤ n **DO** S; Incrementar(i) **END**

- El tiempo de ejecución de una llamada a un procedimiento $F(P_1,P_2, \ldots, P_n)$ es 1 más el tiempo de evaluación de los parámetros más el tiempo de ejecución de F. Es decir, se tiene que $T= 1+T(P_1)+T(P_2)+\ldots+T(P_n)+T(F)$

g. ***Órdenes de Complejidad.*** Se resalta que Ayala de la Vega, Aguilar, García y Gómez (2019) indican que a un conjunto de funciones "f", las cuales comparten un mismo comportamiento asintótico (es decir, una misma cota), dicha cota se le denomina *orden de complejidad*.

Con respecto a las funciones que representan los órdenes de complejidad, estas pueden ser variadas; pero, por la experiencia y por su sencillez, pueden ser resumidas en las funciones que aparecen con mayor frecuencia y que son detalladas en la Tabla 3. De ahí que otras funciones aparezcan es poco habitual (Ayala de la Vega, Aguilar, García y Gómez, 2019). Es decir, las funciones indicadas en la Tabla 3 permiten una clasificación que abarca, en general, a los órdenes de complejidad. A continuación, se presenta un orden de complejidad según la mayor o menor cantidad de tiempo necesario para que un algoritmo solucione un problema, para un "$n>n_0$":

$O(1)\subset O(\log n)\subset O(n)\subset O(n\log n)\subset O(n^2)\subset O(n^3)\subset O(n^k)\subset O(2^n)\subset O(n!)$.
(Guerequeta, R., y Vallecillo, A., 2019, p.31).

Tabla 3

Descripción de órdenes de complejidad

Orden	***Nombre***	***Comentario***
O(1)	Constante	Algoritmo responde en un tiempo constante
O(logn)	Logarítmico	Tiempo crece de forma logarítmica
O(nlogn)	Linearítmico	Orden relativamente bueno. En este orden se encuentra el ordenamiento merge short
$O(n^c)$	Polinómico	Cuando c=1, se denomina orden lineal. Cuando c=2, se denomina cuadrático. Cuando c=3, se denomina orden cúbico. Intuitivamente para c=3 es el último orden aceptable, a partir de c>4, los algoritmos son difíciles de tratar.
$O(c^n)$, c>1	Exponencial	Algoritmos con crecimiento mucho mayor a los polinómicos. Problemas intratables
O(n!)	Factorial	Prueba todas las combinaciones posibles. Problema intratable
$O(n^n)$		Tan intratable como el O(n!).

Nota. Adaptado de Conjunto u órdenes de complejidad [Tabla], Ayala de la Vega et al. (2019, p.14)

De acuerdo con la Tabla 3, el orden de complejidad aceptable, en el caso polinomial, es de orden cúbico o uno cercano a él (Ayala de la Vega, Aguilar, García y Gómez, 2019). En el caso más extremo, un orden de un polinomio de grado 4 o cercano a él, pero delimitando un rango de tamaño de instancias en las cuales podamos encontrar una solución en un tiempo aceptable.

En la actualidad, los recursos computacionales que son estudiados para medir la complejidad de un algoritmo son los indicados por Oré (2010): complejidad temporal y complejidad espacial, los cuales son detallados a continuación:

- El tiempo (complejidad temporal): Consiste en el número de operaciones elementales necesarias para que el algoritmo encuentre una solución al problema estudiado. De tal manera, dicho recurso es denominado complejidad temporal (Maldonado, 2013) y cuya notación es expresada por “T(n)”, la cual indica la cantidad de operaciones elementales que un algoritmo realiza en función del tamaño de la entrada “n”.

- El espacio (complejidad espacial): Consiste en la cantidad de memoria RAM usada para la resolución de un problema por medio de la ejecución del algoritmo y es denominada complejidad espacial. Para este caso, generalmente se utiliza la notación “S(n)” para representar la cantidad de espacio/memoria virtual (RAM) que requiere una función de tamaño de entrada “n”.

De estos dos recursos, Oré (2010) menciona que un algoritmo que requiera un elevado tiempo para resolver un problema no será de gran utilidad, y que, si el algoritmo necesita una cantidad excesiva de memoria, difícilmente podrá utilizarse en otros computadores.

Es importante señalar que, para el desarrollo de la presente investigación, utilizamos un análisis de la complejidad temporal por medio de la notación de la Big O.

2.2.22. Tipos de Problemas

El proceso computacional, como indica Cortéz (2004), es conocido como un proceso algorítmico, donde la importancia de su estudio radica en que un computador no puede ejecutar un problema que no presenta una solución algorítmica. Por tanto, una limitación del proceso algorítmico constituye también una limitación para el proceso computacional.

Por otra parte, Tejero (2020) afirma que los problemas, en general, pueden dividirse en dos grupos: Los computables y los no computables; de tal modo que entendemos que un “Problema Computable” es aquel para el que existe, al menos, un algoritmo capaz de obtener soluciones; en caso contrario, es un “Problema No Computable”.

Por su parte, Maldonado (2013) denomina “*Decidibles”* a los problemas computables e *Indecidibles* a los no computables. Dentro de los problemas “Decidibles”, se pueden presentar casos en los cuales, si bien un algoritmo puede encontrar una solución al problema, se requiere una gran cantidad de recursos, lo que hace inviable tal algoritmo.

Al estudio de dichos recursos computacionales para resolver un problema se denomina complejidad. Por lo cual, resulta importante el estudio de la complejidad de un algoritmo en comparación con otros que puedan solucionar el mismo problema, con el fin de determinar su *Eficiencia*, la cual hace referencia al uso de los recursos computacionales necesarios para resolver el problema analizado. De tal manera que el principal recurso utilizado es el tiempo, expresado en cantidad de operaciones elementales necesarias por el algoritmo para solucionar el problema.

De acuerdo con esto, el Problema del Agente Viajero (TSP) es un problema *Decidible*, debido a que sí existen algoritmos para determinar la solución de una instancia dada.

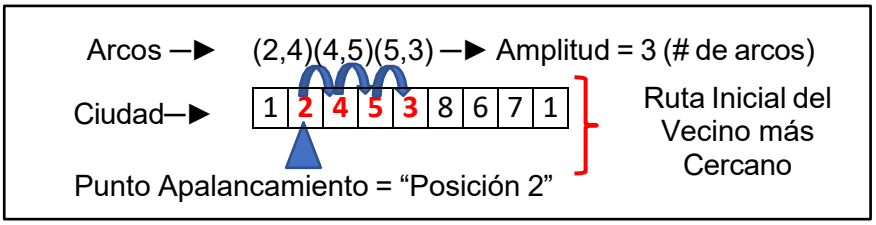
2.2.23. Definiciones previas para el algoritmo a proponer

A continuación, se presentan las siguientes definiciones:

a. ***Amplitud (A).*** La cantidad de arcos a partir de nuestro punto de apalancamiento sobre la ruta del Vecino más Cercano (ver Figura 21). Este concepto surge para evitar presionar, de manera excesiva, al proceso de búsqueda del algoritmo, recomendando valores pequeños (menores a 5) para el valor de "A".

Figura 21

Grafica del concepto de "Amplitud"



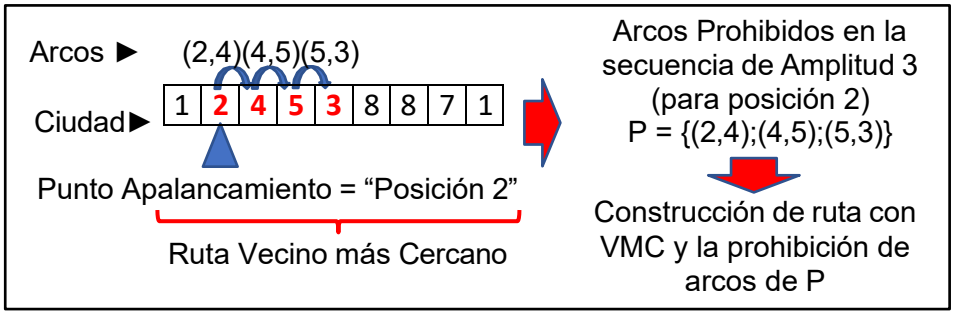
b. ***Rango de Puntos de Apalancamiento.*** Rango de nodos a los cuales se les aplica el Sacrificio Cortoplacista. Se determinó un rango entre el nodo "1" al "n-6", donde "n" representa el orden del grafo, basado en la regla del pensamiento sistémico que dice: "la causa y el efecto no están próximos en el tiempo y el espacio" (Senge, 2010, p.84). Por lo cual, se debió dar un tiempo adecuado para ver dichos efectos, es decir, una cantidad menor a los "n-2" primeros nodos (para mayor detalle, ver Anexo 23).

c. ***Prohibición de "A" arcos.*** Cantidad "A" de arcos contados a partir del punto de apalancamiento que no deben aparecer en la ruta final construida (ver Figura 22). Este concepto surgió de una de las conclusiones del juego de la cerveza, que dice: "Las personas pertenecientes a la misma estructura, tienden a producir resultados

cualitativamente similares" (Senge, 2010, p.53). Donde los arcos descritos en la ruta construida por el pensamiento del Vecino más Cercano son equivalentes a personas, los cuales debemos evitar que aparezcan en la ruta construida por nuestro algoritmo, para evitar presentar resultados similares a los encontrados por el pensamiento del Vecino más Cercano. Es decir, debemos usar una Prohibición de arcos provenientes del Vecino más Cercano.

Figura 22

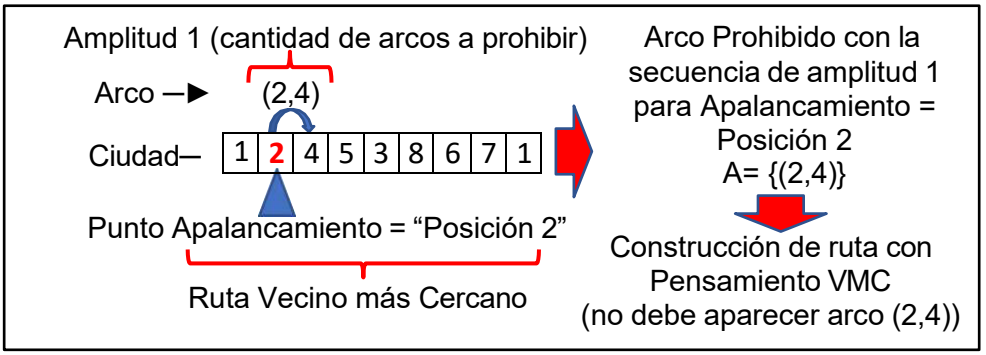
Arcos prohibidos en secuencia de amplitud 3 para posición 2



d. ***Sacrificio Cortoplacista Adaptativo de amplitud 1 (SCA(1))***. Consiste en prohibir un arco por cada punto de apalancamiento, el cual es construido a partir del punto de apalancamiento seleccionado, para luego seguir con la aplicación de la heurística del Vecino más Cercano (ver Figura 23) y se elige la mejor ruta construida. Dicho concepto es básicamente el Sacrificio Cortoplacista desarrollado por Pérez (2011) y Pérez y Jaramillo (2014), quienes basaron su algoritmo en los puntos de apalancamiento, los cuales son aquellas zonas donde al realizar una mínima acción pueden producir resultados positivos en la consecución de nuestros objetivos.

Figura 23

Sacrificio Cortoplacista Adaptativo de amplitud 1 [SCA(1)]

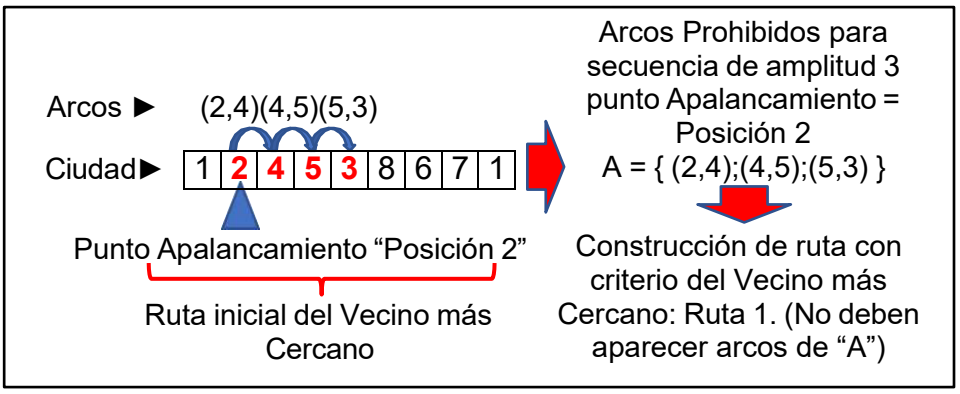


Nota. Para el punto apalancamiento en la posición 2

e. ***Sacrificio Cortoplacista Adaptativo de amplitud "A" (SCA(A), donde A≥2).*** Consiste en prohibir una cantidad de arcos igual al tamaño de la Amplitud por cada punto de apalancamiento, para proseguir con la aplicación de la heurística del Vecino más Cercano (ver Figura 24). Con esto, se generan tantas rutas como puntos de apalancamiento, donde la amplitud "A" debe tomar valores pequeños para no presionar demasiado al proceso de búsqueda del algoritmo (A ≤ 5); más adelante, fue analizado y validado en la Tabla 5.

Figura 24

Sacrificio Cortoplacista Adaptativo de amplitud 3 [SCA(3)],



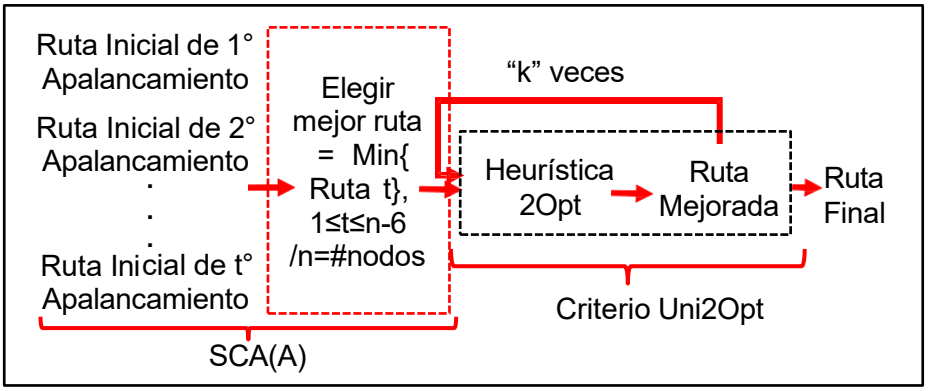
Nota. SCA(3) para el punto de apalancamiento en la posición 2

Observando la Figura 24, vemos que, al aplicar nuestro diseño de Sacrificio Cortoplacista con amplitud 3, se prohíben tres arcos adyacentes a partir de la posición 2 de la ruta construida por el Vecino más Cercano. Por medio de ello, se busca evitar la aparición de individuos de la estructura anterior.

f. ***Criterio Uni2Opt.*** Consiste en que, a partir de una ruta (la mejor de un conjunto de rutas), se aplica la heurística 2Opt una cantidad "k" de veces (ver Figura 25), donde el valor de "k" es determinado por ensayo y error, probando distintos valores. Dicho criterio surge para incrementar los efectos positivos de la aplicación de la heurística 2Opt; es decir, por medio de la aplicación de los ciclos repetitivos se busca incrementar los efectos positivos de la heurística.

Figura 25

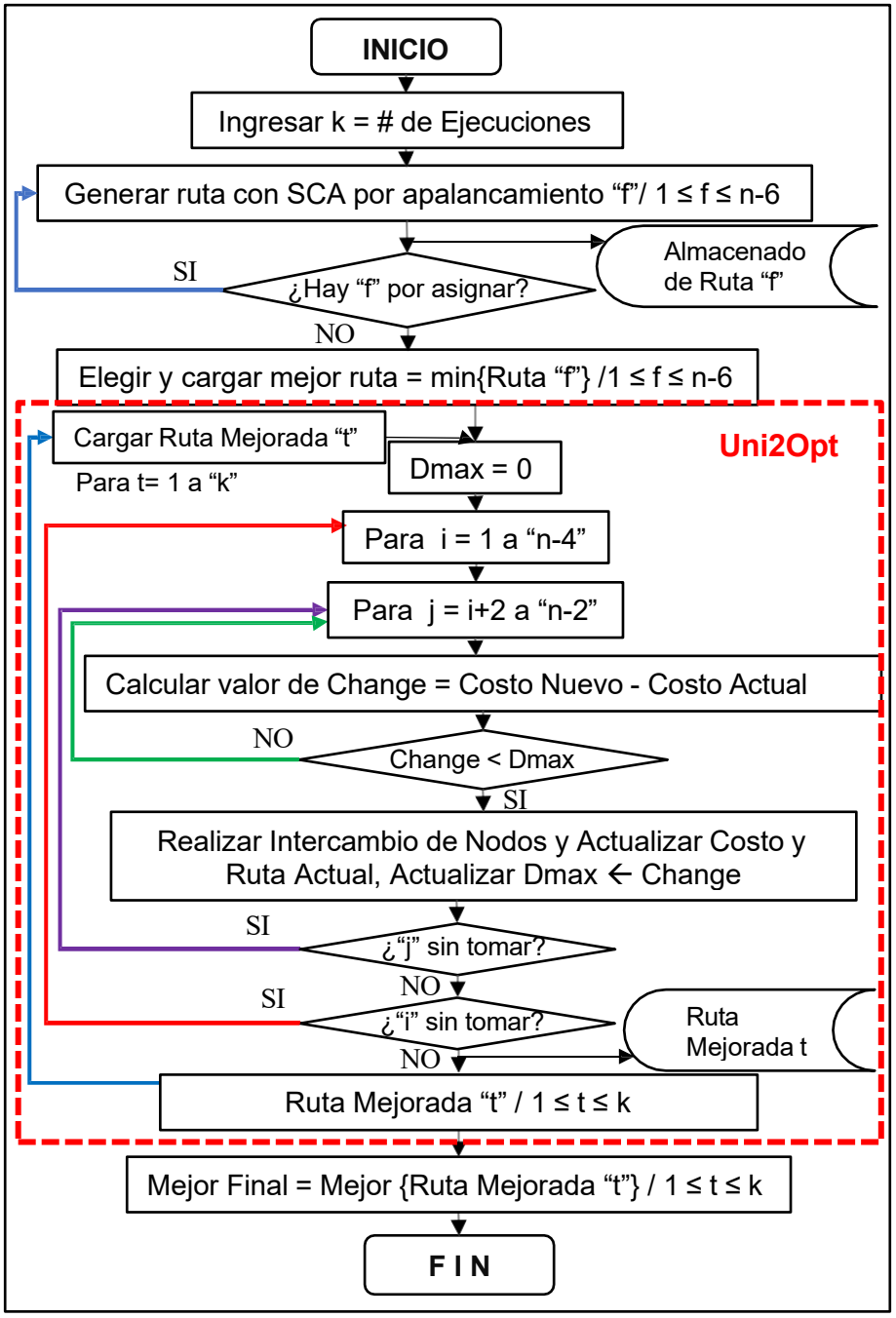
Funcionamiento del criterio Uni2Opt con el SCA(A)



A partir de la Figura 25, se aprecia que el término "Uni2Opt" se deriva de la aplicación repetida de la heurística Intercambio 2Opt sobre la ruta proveniente de la mejor ruta, resultado de SCA para cada punto de apalancamiento. De tal manera que, en cada ejecución, se actualiza la ruta a mejorar. Para describir este proceso de búsqueda que realiza el Criterio Uni2Opt, se presenta el Diagrama de Flujo en la Figura 26.

Figura 26

Diagrama de flujo del criterio Uni2Opt con el SCA(A)

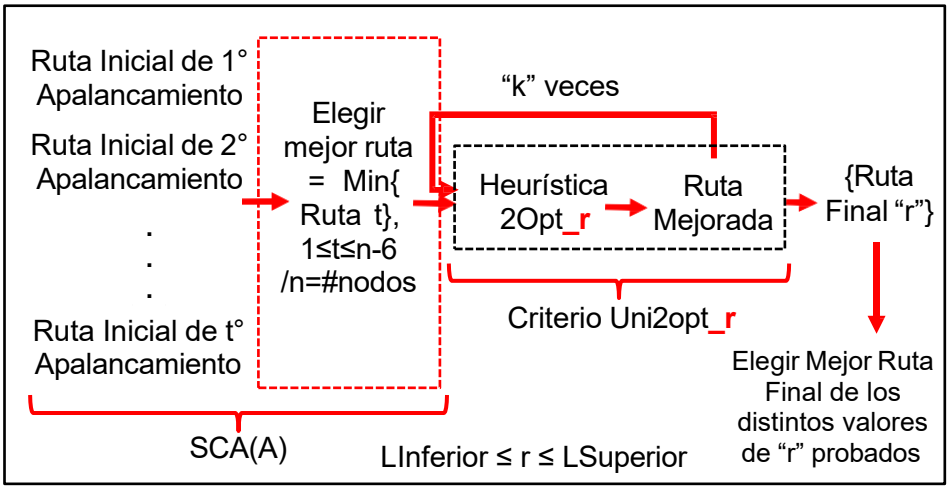


g. ***Criterio Uni2Opt_r***: Consiste en que, a partir de una ruta (la mejor de un conjunto de rutas), se aplica la heurística 2Opt_r una cantidad "k" de veces (ver Figura 27). El valor "k" se determina por ensayo y error, probando distintos valores durante la experimentación.

Es decir, se adiciona el factor “r” al Criterio Uni2Opt, con la adición de dicho término, se busca una mínima perturbación en el proceso de intercambiar nodos, con la esperanza de que, a largo plazo, tal perturbación produzca efectos positivos en el proceso de búsqueda si son aplicadas en las posiciones correctas (puntos de apalancamiento).

Figura 27

Funcionamiento del criterio Uni2Opt_r con el SCA(A)

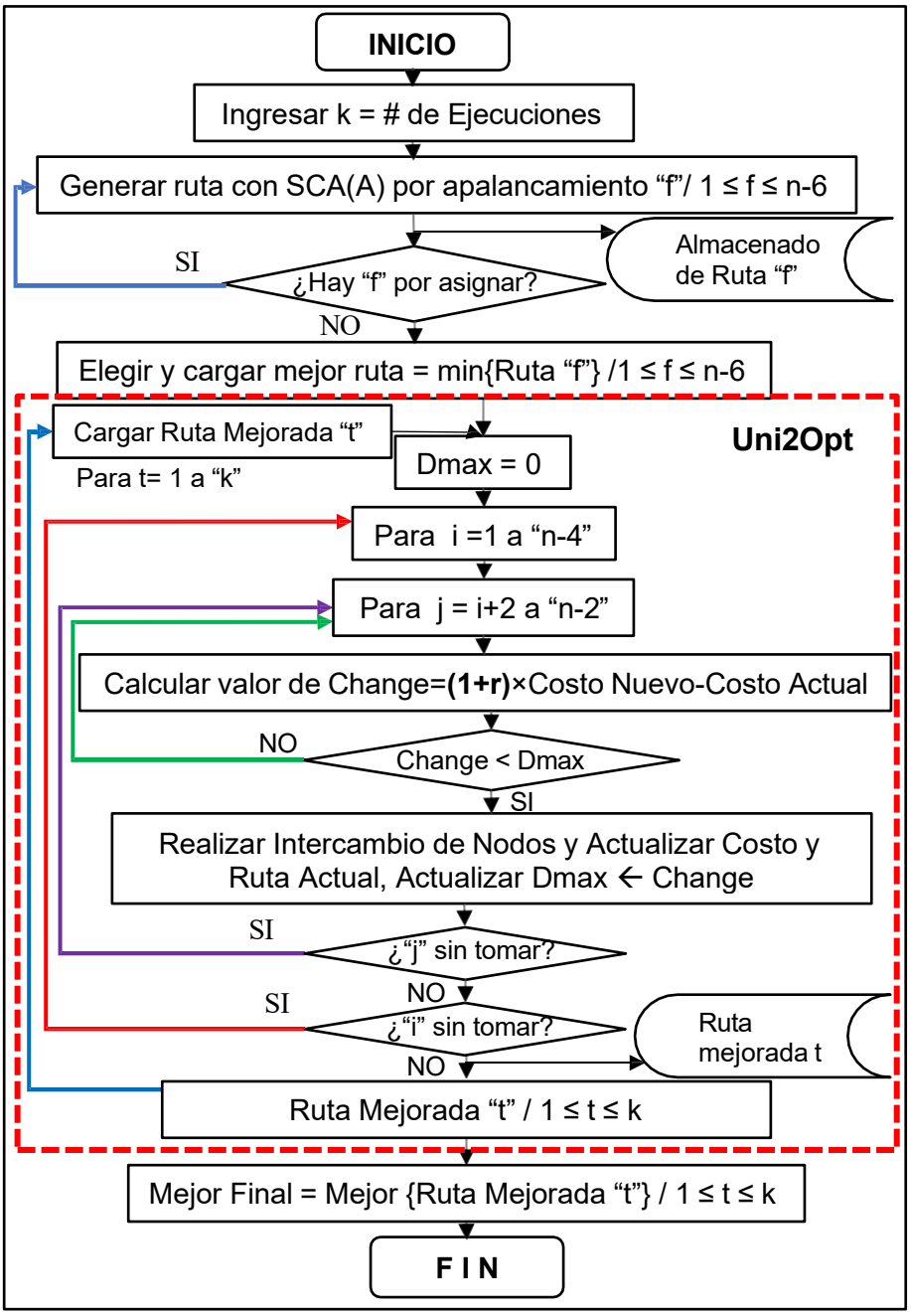


En la Figura 27, el criterio Uni2Opt_r es equivalente al Uni2Opt, con la adición del factor “r” al criterio de decisión para realizar el intercambio 2Opt de los nodos seleccionados, con lo cual se busca la exploración de nuevas zonas del espacio solución. Con respecto a “LInferior” y “LSuperior”, estos indican el rango que abarca el valor “r”, detallado en el Anexo 23, así como la cantidad de valores de “r” especificada en el Anexo 27.

Para describir su proceso de búsqueda, se presenta su respectivo Diagrama de Flujo en la Figura 28.

Figura 28

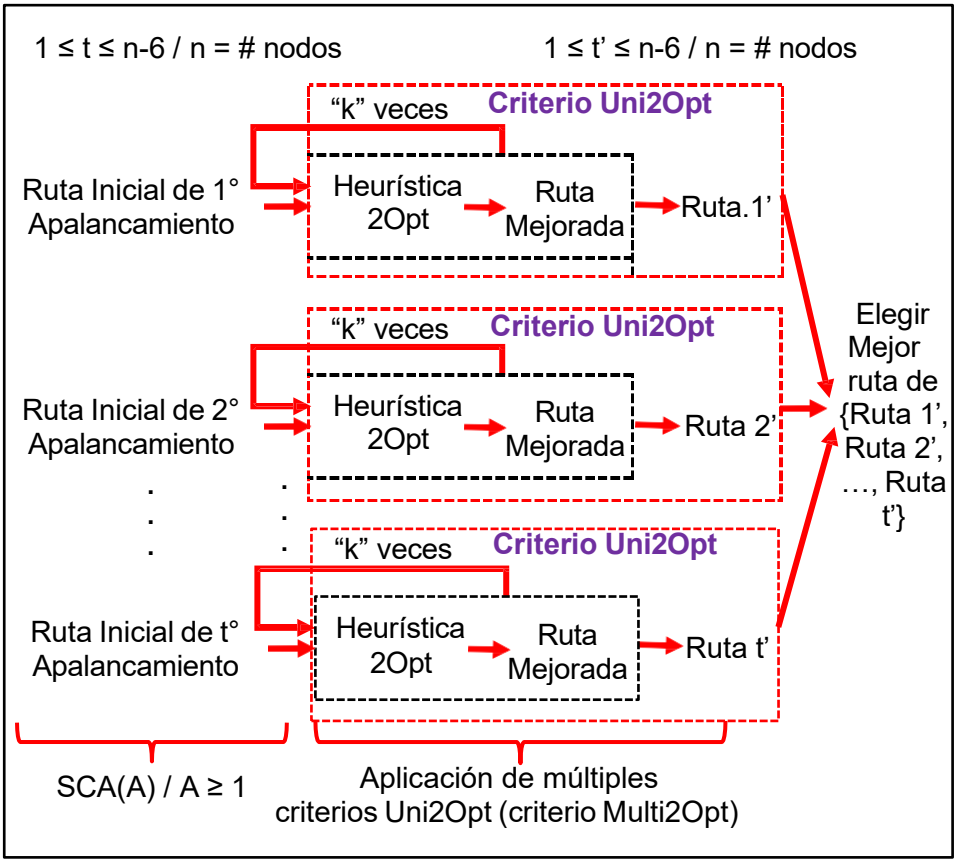
Diagrama de flujo del Uni2Opt_r con el SCA(A)



h. ***Criterio Multi2Opt.*** Consiste en que, a partir de distintas rutas iniciales, que varían según el punto de apalancamiento utilizado, se aplica la heurística 2Opt una cantidad "k" de veces por cada ruta a mejorar, es decir, el Criterio Uni2Opt (ver Figura 29). Dicho valor de "k" es determinado por ensayo y error.

Figura 29

Funcionamiento del criterio Multi2Opt con el SCA(A)

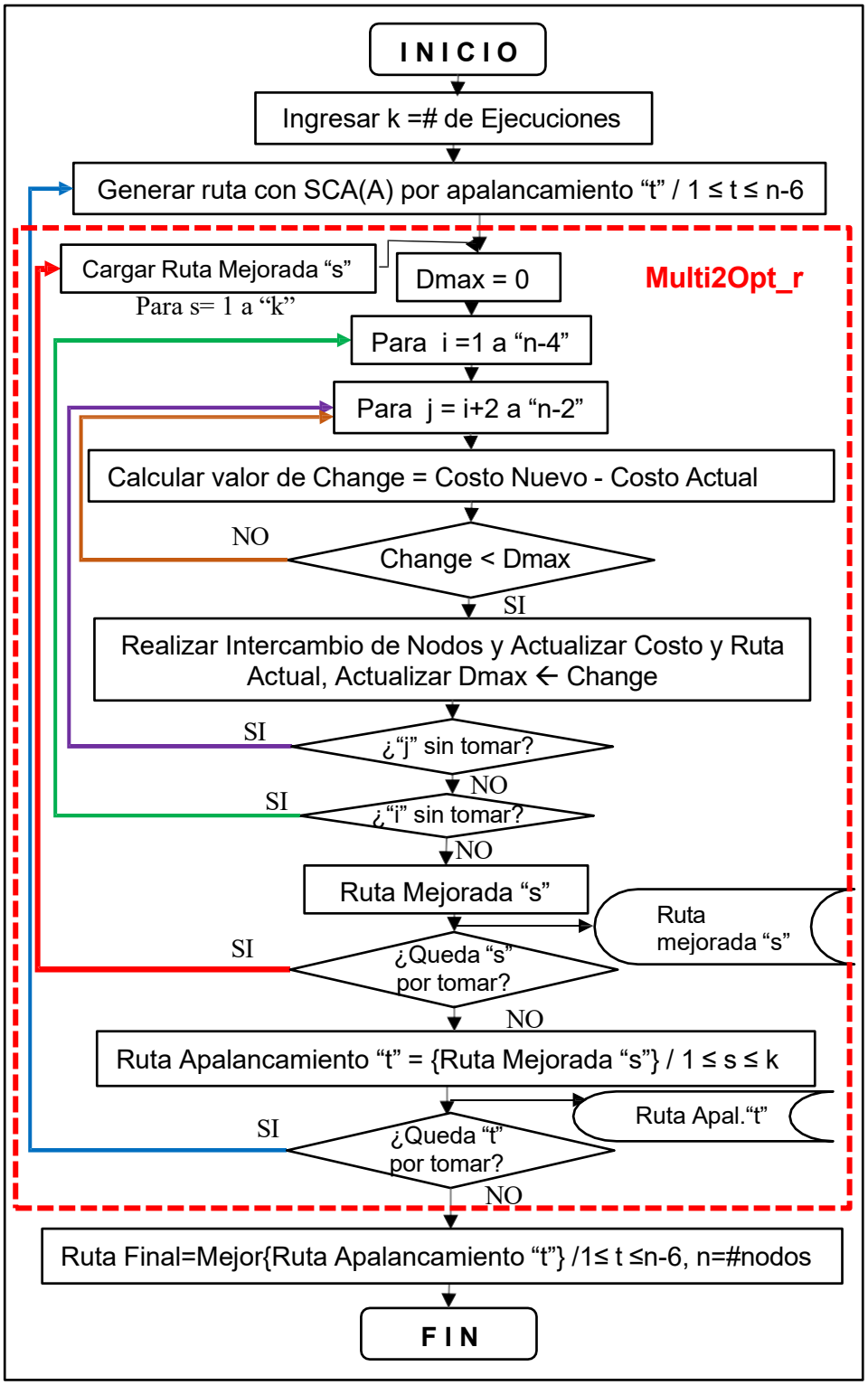


El término Multi2Opt, mostrado en la Figura 29, proviene de la aplicación de múltiples Criterios Uni2Opt a cada una de las rutas generadas, las cuales son resultado de la aplicación del Sacrificio Cortoplacista de amplitud "A" en cada punto de apalancamiento. Es decir, una estructura de búsqueda tipo poblacional.

En el caso mostrado en la Figura 29, se aplicaron "t" Sacrificios Cortoplacistas (uno por cada nodo seleccionado), generando en total "t" rutas iniciales, a cada una de las cuales se les aplicó la heurística 2Opt una cantidad "k" veces para luego elegir la mejor de las rutas mejoradas. Es decir: min {distancia (Ruta 1'), distancia (Ruta 2'), …, distancia (Ruta t')}. Para describir su proceso de búsqueda, se presenta el Diagrama de Flujo en la Figura 30.

Figura 30

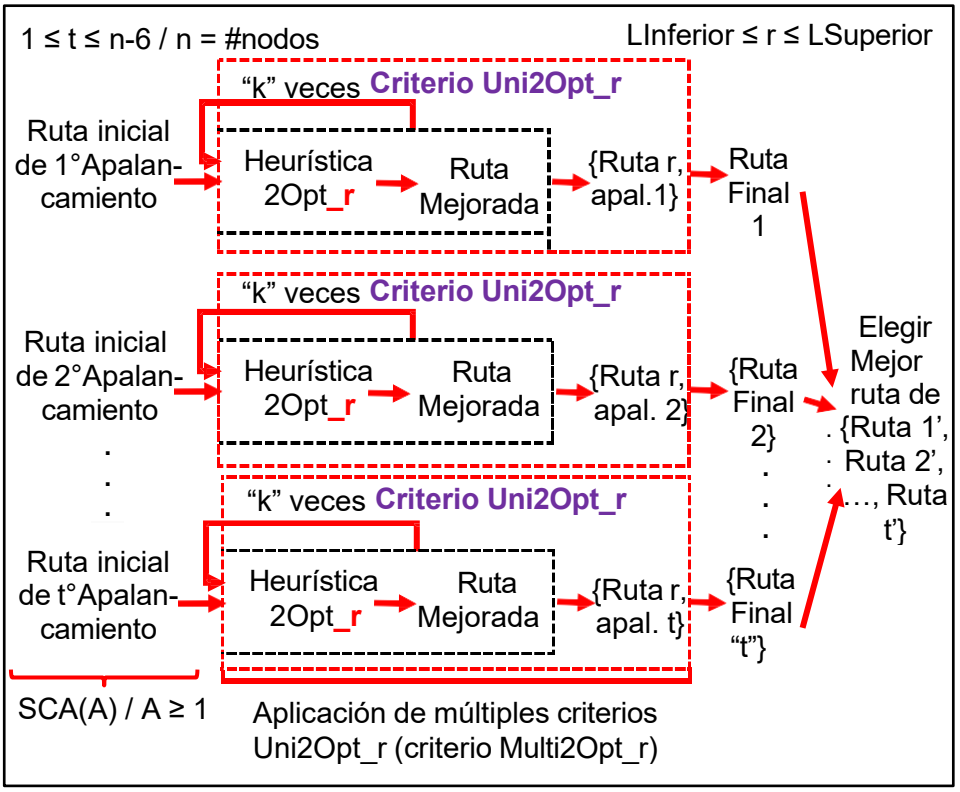
Diagrama de flujo de Multi2Opt dentro del SCA(A)_Multi2Opt



i. **Criterio Multi2Opt_r:** Consiste en la aplicación del criterio Uni2Opt_r una cantidad "k" de veces (ver Figura 31) por cada ruta construida por su respectivo punto de apalancamiento. Dicho valor "k" es determinado por ensayo y error, probando distintos valores. Es decir, se agrega el factor "r" al criterio de intercambio de arcos 2Opt; con la adición de este término, se busca una mínima perturbación en el proceso de intercambiar nodos, con la esperanza de que, a largo plazo, tal perturbación produzca efectos positivos en el proceso de búsqueda al ser aplicados en los puntos de apalancamiento. El término Multi2Opt_r proviene de la aplicación de múltiples criterios Uni2Opt_r a cada una de las rutas generadas por punto de apalancamiento. Sobre "LInferior" y "LSuperior", estos indican el rango que abarca el valor "r", detallado en el Anexo 23, así como la cantidad de valores de "r" especificada en el Anexo 27.

Figura 31

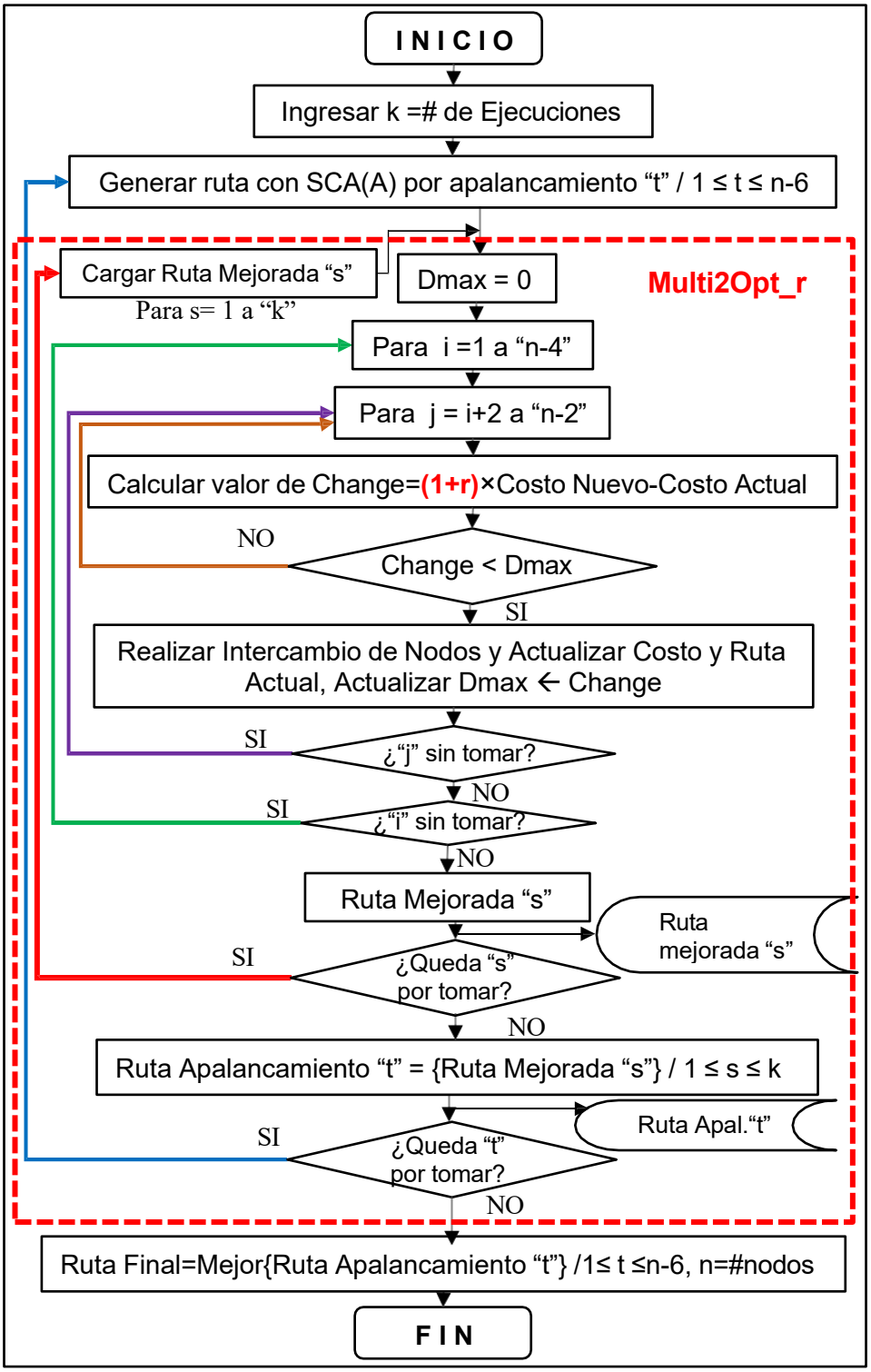
Funcionamiento del criterio Multi2Opt_r con el SCA(A)



Para describir el proceso del funcionamiento del **Criterio Multi2Opt_r,** se presenta el Diagrama de Flujo en la Figura 32.

Figura 32

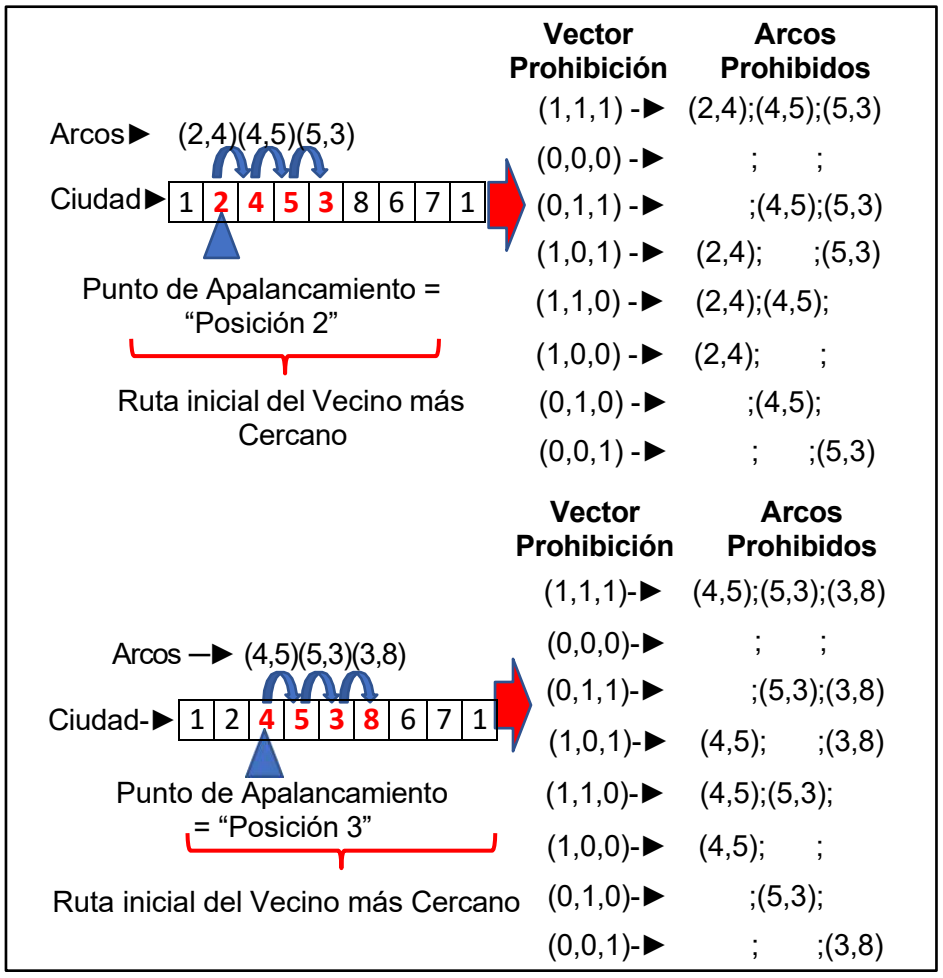
Diagrama de flujo Multi2Opt_r dentro del SCA(A)_Multi2Opt_r



j. **Vector Prohibición:** Vector de longitud igual a la amplitud, que indica la posición de “Arcos Prohibidos”, con componentes binarias. En este caso, el valor “1” indica prohibición y “0” no prohibición del arco en su respectiva posición. Dicho vector es agregado al funcionamiento del SCA(A)_Multi2Opt para ampliar su rango de búsqueda, donde la cantidad máxima de vectores prohibición será $2^{Amplitud}$ (ver Figura 33). La idea del “Vector Prohibición” es que la misma estructura del algoritmo se autorregule, buscando la mejor combinación de valores para el vector. De la Figura 33 podemos notar que los “Arcos Prohibidos”, indicados por dicho vector, varían según su punto de apalancamiento.

Figura 33

Vector Prohibición para sacrificio cortoplacista de amplitud 3



2.3. Marco Conceptual

A continuación, se brindan los conceptos relacionados al tema de investigación que se usaron durante su desarrollo.

- **Algoritmo híbrido:** Es la combinación del pensamiento y/o la lógica de dos o más algoritmos, con el objetivo de mejorar la calidad de las soluciones que podrían determinarse si se aplicaran por separado para resolver un mismo problema.
- **Cardinal de un conjunto:** Sea A un conjunto no vacío; entonces, el cardinal de A, expresado como |A|, consiste en el número de elementos del conjunto A.
- **Criterio de parada:** Condición que debe cumplir un algoritmo para concluir su ejecución.
- **Costo ruta solución:** Es el valor asociado al recorrido de la ruta determinada, el cual puede expresarse en términos de tiempo, unidades monetarias, distancias, etc., según el criterio de interés del investigador.
- **Diagrama de bloques:** Consiste en una representación de manera general de ideas y conceptos, expresada en bloques en forma secuencial.
- **Diagrama de flujo:** Diagrama que expresa los pasos a seguir en el computador, de manera más específica que en el diagrama de bloques.
- **Eficacia**: Mide qué tan cerca nos encontramos a la meta propuesta (grado de acercamiento al óptimo o meta planteada).
- **Eficiencia:** Mide el uso de los recursos necesarios para alcanzar la meta propuesta, donde a menor cantidad de recursos utilizados para

alcanzar la misma meta, mayor será la eficiencia. Para la presente investigación se utilizó el recurso tiempo computacional.

- **Ejecución:** Proceso por el cual se ejecuta una vez un algoritmo, de tal forma que se obtiene una solución final al problema por el cual el algoritmo fue ejecutado.

- **Indicadores de desempeño algorítmico:** Son instrumentos que sirven para proporcionar información cuantitativa sobre la calidad de la solución brindada por un algoritmo, para así poder compararlo con otros en la determinación de la solución para un mismo problema. Entre los principales, tenemos la Eficacia Algorítmica.

- **Instancia:** Es la particularización de un problema, significa referirse a un grafo concreto (y los parámetros adicionales que acompañan al problema para diferenciarlo de otros) que se entrega como entrada a un algoritmo.

- **Iteración:** Repetir una serie de operaciones para aproximarse progresivamente al resultado deseado (Merriam-Webster, s,f.).

- **Librería virtual TSPLIB:** Biblioteca virtual donde se encuentran almacenados distintos problemas para distintos tamaños de instancias del Problema del Agente Viajero, con sus respectivas soluciones óptimas (Reinelt, 1995). Publicado en 1991 por el científico de la computación Gerhard Reinelt, cuya ubicación en la nube tiene propiedad intelectual ©Copyright Universität Heidelberg. Adicionalmente, según TSPLIB FAQ (Preguntas y Respuestas) de (Reinelt, 1995), se indica que las soluciones reportadas en TSPLIB no son los mejores conocidos, sino las soluciones óptimas, las cuales ya han sido demostradas por medio de algoritmos como Branch and Bound y Branch and Cut y otros algoritmos. Y que, en caso de que alguien afirme haber encontrado una mejor solución, esto debe ser un error de método de cálculo de distancias o error por redondeos al usar decimales, cuando TSPLIB indica tomar valores enteros.

- **Mejor solución disponible:** Es la mejor solución encontrada durante las distintas experimentaciones realizadas sobre una misma instancia de TSPLIB, por los algoritmos revisados.

- **Nivel de Eficacia:** Se define como la proporción de la cantidad de instancias revisadas, donde el algoritmo obtuvo la mejor eficacia.

- **Orden de un grafo:** Dado un grafo G =(V,E), donde “V” representa al conjunto de nodos y “E” el conjunto de arcos entre los nodos. El orden de un grafo es definido como la cantidad de nodos que presenta el grafo, es decir, el cardinal del conjunto “V”, representado como |V|.

- **Problema del Agente Viajero Simétrico:** Problema donde el peso o costo del arco (v_i, v_j) es igual al peso o costo del arco (v_j, v_i) / i, j= 1, 2, …, n; donde “n” es el orden del grafo.

- **Punto de apalancamiento:** Zona en la cual, al realizar un pequeño esfuerzo, se pueden generar mejores resultados que aquellos que se obtendrían si se hubiera aplicado en otras zonas.

- **Ruta:** Un camino que conecta dos puntos determinados (Ministerio de Transporte y Comunicaciones del Perú [MTC], 2018).

- **Ruta solución:** Ruta que cumple los criterios establecidos, como, por ejemplo, visitar todos los nodos establecidos en la matriz de costos o no sobrepasar un tiempo establecido para su recorrido, entre otros.

- **Síntesis:** Explicación abreviada de un conjunto de ideas y/o conceptos.

- **Tecnología:** Según la Comisión Nacional de Investigación Científica y Tecnológica de Chile [CONICYT] (2008) define a la ciencia como un: “Conjunto de teorías y de técnicas que permiten el aprovechamiento práctico del conocimiento científico. En otra acepción, tecnología es el conjunto de los instrumentos y procedimientos industriales de un determinado sector o producto” (p.45).

CAPÍTULO III

METODOLOGÍA

3.1. Introducción

Se analizó la heurística del Vecino más Cercano con Sacrificio Cortoplacista 2Opt_r, heurística propuesta por Pérez (2011), con la finalidad de diseñar algoritmos que superen a su predecesora. Para lo cual, se siguió el proceso sugerido por Casaño (2016): Observar, Determinar el problema, Documentar, Reflexionar, Elaborar el proyecto de intervención, Valorar, Comunicar, Implementar, Seguimiento, Evaluación, con mínimas modificaciones aplicadas en la parte final, la cual se ajusta a nuestra realidad de Construcción. Como indicador de mejoría se usó:

- *La Eficacia*, definida por Pérez (2011) y detallada en el numeral "2.2.19. Eficacia Algorítmica", como la mejor solución encontrada por el algoritmo (costo del tour o ruta determinada). Para evaluar cuán eficaz fue nuestra solución, se usó el *Error Relativo,* el cual mide la distancia porcentual de cada solución al óptimo.

- El *Nivel de Eficacia*, detallado en el numeral "2.2.19. Eficacia Algorítmica", se definió como la proporción de la cantidad de instancias revisadas donde el algoritmo obtuvo la mejor eficacia al ser comparado con las soluciones de otros algoritmos y es cuantificado como el cociente entre la "Cantidad de instancias donde el algoritmo evaluado obtuvo la mejor eficacia" entre el "Cantidad total de instancias seleccionadas".

3.2. Tipo de investigación

Chávez (2001) indica que el tipo de investigación debe establecerse de acuerdo con el tipo de problema que se desea solucionar y con los objetivos que se desea alcanzar. Para lo cual, se debe evaluar la disponibilidad de los recursos necesarios para alcanzar tales objetivos.

El tipo de investigación usada es tecnológica, sobre la cual Piscoya (1987, citado por Esteban, 2018) indica que busca mejorar procedimientos, métodos y herramientas. En este caso, se buscó mejorar el nivel de eficacia de un algoritmo previamente diseñado, tal como la heurística del Vecino más Cercano con Sacrificio Cortoplacista 2Opt, diseñada por Pérez (2011).

Dentro de la investigación tecnológica, se pueden encontrar subdivisiones, donde el presente trabajo de investigación recae en la clasificación de investigación en tecnologías formales. Piscoya (1987, citado por Esteban, 2018) indica que: "(…) comprende los campos de la Programación de Computadoras, Análisis de Sistemas, la Investigación Operativa y la Cibernética. Es en este campo que se ha desarrollado la tecnología algorítmica" (p.4).

3.3. Diseño de investigación

Con respecto al diseño de la investigación, Kothari (2004, citado por Arbaiza, 2014) lo define como "la estructura conceptual esencial para conducir la investigación, extraer información necesaria y obtener evidencia; consiste en planificar una estrategia que permita conseguir los datos del estudio" (p.123). Es decir, el diseño nos indica cómo se realiza la investigación, por lo cual el diseño cuasi experimental resultó ser el más conveniente, el cual consiste en una investigación la cual altera parcialmente la realidad analizada.

3.4. Nivel de investigación

El nivel de investigación es explicativo, el cual no solo busca acercarse al problema, sino también conocerlo, para luego plantear las causas de este, relacionadas con lo que impide que los algoritmos propuestos por Pérez (2011) tengan un mayor nivel de eficacia.

3.5. Enfoque de investigación

Dado que el objetivo de la presente investigación consistió en la construcción de algoritmos que permitan un mejor nivel de eficacia respecto a las versiones originales desarrolladas por Pérez (2011), para medir el nivel de mejoría de la variable de estudio, se realizó un enfoque cuantitativo, por ser de interés la medición de dicha mejoría.

3.6. Identificación de variables

3.6.1. Variable independiente

Heurística híbrida del Vecino más Cercano con Sacrificio Cortoplacista 2Opt_r.

3.6.2. Variable dependiente

Solución del Problema del Agente Viajero Simétrico.

3.7. Unidad de análisis

La unidad de análisis consistió en un algoritmo que resuelve el Problema del Agente Viajero Simétrico.

3.8. Población del estudio

La población del estudio consistió en el total de algoritmos heurísticos y metaheurísticos conocidos, para resolver el Problema del Agente Viajero Simétrico.

3.9. Tamaño de muestra

Se tomó un total de nueve algoritmos: dos basados en el Sacrificio Cortoplacista desarrollado por Pérez (2011), cuatro metaheurísticas basadas en Algoritmos Genéticos (AG), dos metaheurísticas basadas en Colonia de Hormigas (ACS) y una metaheurística basada en Programación Evolutiva (PE). Dicha cantidad fue determinada según las soluciones de los algoritmos consolidados por Pérez (2011), que cumplieron una cantidad de filtros específicos (para obtener más información al respecto, ver numeral “4.1.1. Selección de instancias de librería TSP para medir Eficacia”).

3.10. Selección de la muestra

Se realizó una selección por conveniencia, ocurriendo que la medición de la *Eficacia* de dichos algoritmos, desarrollados por diversos investigadores a nivel mundial, se consolidó en la investigación de Pérez (2011); a partir de la cual se extrajeron los resultados a comparar. Por tanto, la muestra consistió en:

- Dos algoritmos heurísticos basados en el Sacrificio Cortoplacista Adaptativo, desarrollado por Pérez (2011): Vecino más Cercano con Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt) y Vecino más Cercano con Sacrificio Cortoplacista Adaptativo 2Opt_r (SCA_2Opt_r).

- Cuatro metaheurísticas basadas en Algoritmos Genéticos: Algoritmo Genético (AG), FRAG_GA, SWAP_GATSP, IGA. (para mayor detalle, ver Anexo 3).

- Dos metaheurísticas basadas en Colonia de Hormigas: Algoritmo de Colonia de Hormigas (ACS) y un Algoritmo de Colonia de Hormigas combinado con la heurística 3Opt (ACS_3Opt) (para mayor detalle, ver Anexo 3).

- Una metaheurística basada en Programación Evolutiva (para mayor detalle, ver Anexo 3).

3.11. Técnicas e instrumentos de recolección de datos

Para la recolección de datos de las instancias, se usó la técnica de revisión de data pública: instancias publicadas del Problema del Agente Viajero Simétrico, almacenadas en la librería virtual TSPLIB. Desde la perspectiva de que un instrumento puede entenderse como un mecanismo para recolectar y almacenar información, se utilizó el siguiente instrumento en la presente investigación:

- Matriz de costos, para cuya construcción se sigue el procedimiento indicado en el numeral 3.12.

3.12. Procedimiento para la construcción de la matriz de pesos

A continuación, se describen los pasos a seguir para la construcción de la “Matriz de Pesos” asociada a cada instancia seleccionada, según Anexos 4, 5, 6 y 7. Dichas matrices de pesos serán necesarias para la lectura, por parte de los algoritmos implementados en Java (Plataforma NetBeans), de las distancias asociadas entre cada par de nodos del grafo.

a. Ingresar a la librería virtual TSPLIB.

b. Ir a “TSP data” de la categoría “Symmetric Traveling Salesman Problem (TSP)”.

c. Del total de archivos disponibles en TSPLIB, se filtran exclusivamente Problemas del Agente Viajero Simétrico revisados por Pérez (2011) y/o archivos requeridos, según los filtros detallados en el numeral “4.1.1. Selección de Instancias de la Librería TSPLIB para medir la Eficacia”.

d. Descargar los archivos requeridos y luego pasar a su descompresión con WinRAR.

e. Como la extensión no es un archivo nativo de Windows, al momento de abrirlo se selecciona abrir con el “Bloc de Notas”.

f. La información del archivo abierto en “Bloc de Notas” se copia y pega en un archivo Excel (*.xlsx).

g. En la primera fila y columna de una hoja adicional del documento Excel (*.xlsx), se registra una cantidad de números que indique la cantidad de nodos del Problema del Agente Viajero Simétrico a estudiar, de tal manera que los índices de la columna 1 representan los nodos iniciales (i) y los índices de la fila 1 representen los nodos destino (j).

h. Los elementos de la matriz de pesos se calculan siguiendo las indicaciones de TSPLIB (Reinelt, 1995), según dos tipos de instancias analizadas: distancia euclidiana y pseudo distancia euclidiana, tal que el cálculo se realizó de la siguiente manera:

   Para la distancia euclidiana*,* se procede a calcular de la siguiente manera:

   nint (d_{ij}) / $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, donde

   nint es una función que redondea al entero más cercano

La pseudo distancia euclidiana, es usada para las instancias att48 y att532 de TSPLIB. Se procede a calcular de la siguiente manera:

$r_{ij} = \sqrt{\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{10}}$; $t_{ij} = \text{nint}\ (r_{ij})$

Si ($t_{ij} < r_{ij}$) entonces $d_{ij} = t_{ij} + 1$;

Caso contrario $d_{ij} = t_{ij}$

Donde:

- **nint** es una función que redondea al entero más cercano.
- $(x_i; y_i)$ coordenadas del nodo "i" en el plano cartesiano.
- $(x_j; y_j)$ coordenadas del nodo "j" en el plano cartesiano.

3.13. Procedimiento para el cálculo del nivel de eficacia

3.13.1. Para el objetivo específico 1

Determinar el *Nivel de Eficacia* de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el *Nivel de Eficacia* del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt.

a. Calcular la *Eficacia* medida de la siguiente manera:

$$Error\ Relativo = \frac{Solución\ alcanzada\ por\ el\ algoritmo\ evaluado - Solución\ óptima\ TSPLIB}{Solución\ óptima\ TSPLIB}$$

- El *algoritmo evaluado* varía entre las versiones Soft y Heavy en la presente investigación y el SCA_2Opt desarrollado por Pérez (2011).

- El *error relativo*, indicador decreciente, mide la distancia relativa al óptimo. Es decir, a menor valor del *error relativo,* mayor será su cercanía al óptimo y, por tanto, mayor su Eficacia alcanzada.

b. Contabilizar el número de instancias en las cuales cada algoritmo alcanzó la mejor eficacia disponible entre las soluciones comparadas, esto por medio del *Nivel de Eficacia*, el cual se define de la siguiente manera:

$$Nivel\ de\ Eficacia = \frac{Cantidad\ de\ instancias\ donde\ el\ algoritmo\ evaluado\ obtuvo\ la\ mejor\ Eficacia}{Cantidad\ total\ de\ instancias\ seleccionadas}$$

- El *algoritmo evaluado* varía entre las versiones Soft y Heavy en la presente investigación y el SCA_2Opt desarrollado por Pérez (2011).

- El *Nivel de Eficacia* es un indicador creciente. Es decir, a mayor valor del *Nivel de Eficacia,* mayor será la proporción de la cantidad de instancias donde su Eficacia alcanzada es la mejor disponible.

c. Se clasifica según el valor del *Nivel de Eficacia* según la Tabla 7.

3.13.2. Para el objetivo específico 2

Determinar el *Nivel de Eficacia* de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el *Nivel de Eficacia* del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r.

a. Calcular la *Eficacia* medida de la siguiente manera:

$$Error\ Relativo = \frac{Solución\ alcanzada\ por\ el\ algoritmo\ evaluado - Solución\ óptima\ TSPLIB}{Solución\ óptima\ TSPLIB}$$

- El *algoritmo evaluado* varía entre las versiones Soft y Heavy en la presente investigación y el SCA_2Opt_r de Pérez (2011).

- El *error relativo*, indicador decreciente, mide la distancia relativa al óptimo. Es decir, a menor valor del *Error Relativo,* mayor será su cercanía al óptimo y, por tanto, mayor su *Eficacia* alcanzada.

b. Contabilizar el número de instancias en las cuales cada algoritmo alcanzó la mejor eficacia disponible entre las soluciones comparadas. Esto por medio del *Nivel de Eficacia*, el cual es definido de la siguiente manera:

$$Nivel\ de\ Eficacia = \frac{Cantidad\ de\ instancias\ donde\ el\ algoritmo\ evaluado\ obtuvo\ la\ mejor\ Eficacia}{Cantidad\ total\ de\ instancias\ seleccionadas}$$

- Donde el algoritmo evaluado varía entre las versiones Soft y Heavy en la presente investigación y el SCA_2Opt_r desarrollado por Pérez (2011).

- Donde el *Nivel de Eficacia* es un indicador creciente. Es decir, a mayor valor del *Nivel de Eficacia*, mayor será la proporción de la cantidad de instancias donde su Eficacia alcanzada es la mejor disponible.

c. Se clasifica según el valor del *Nivel de Eficacia* según la Tabla 7.

Al concluir los objetivos 1 y 2, se procede a construir una tabla que compare el *Nivel de Eficacia* entre las versiones de Pérez (2011) con las versiones Heavy y Soft desarrolladas en la presente investigación.

3.13.3. Para el objetivo específico 3

Determinar el *Nivel de Eficacia* de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el *Nivel de Eficacia* de las metaheurísticas seleccionadas.

a. Calcular la *Eficacia* medida de la siguiente manera:

$$Error\ Relativo = \frac{Solución\ alcanzada\ por\ el\ algoritmo\ evaluado - Solución\ óptima\ TSPLIB}{Solución\ óptima\ TSPLIB}$$

- El *algoritmo evaluado* varía entre las versiones Soft y Heavy en la presente investigación y los algoritmos metaheurísticos seleccionados, los cuales son detallados en el Anexo 3 y cuyas instancias seleccionadas se encuentran detalladas en el Anexo 4, Anexo 5, Anexo 6 y Anexo 7.

- El *error relativo*, indicador decreciente, mide la distancia relativa al óptimo. Es decir, a menor valor del *error relativo,* mayor será su cercanía al óptimo y, por tanto, mayor su *Eficacia* alcanzada.

b. Contabilizar el número de instancias en las cuales cada algoritmo alcanzó la mejor eficacia disponible entre las soluciones comparadas. Esto por medio del *Nivel de Eficacia*, el cual es definido de la siguiente manera:

$$Nivel\ de\ Eficacia = \frac{Cantidad\ de\ instancias\ donde\ el\ Algoritmo\ evaluado\ obtuvo\ la\ mejor\ Eficacia}{Cantidad\ total\ de\ instancias\ seleccionadas}$$

c. Se clasifica según el valor del *Nivel de Eficacia* según la Tabla 7.

- El *algoritmo evaluado* varía entre las versiones Soft y Heavy en la presente investigación y los algoritmos metaheurísticos seleccionados, según el Anexo 4, Anexo 5, Anexo 6 y Anexo 7.

  - Algoritmos Genéticos (GA-Bélgica y Suiza, FRAG_GA, SWAP_GATSP y el IGA desarrollados en Libia),

  - Algoritmo de Colonia de Hormigas (Ant Colony System y Ant Colony System con Heurística de mejoramiento 3Opt, de Bélgica y Suiza).

  - Y Otras Metaheurísticas (un algoritmo basado en Programación Evolutiva, de Bélgica y Suiza).

- El *Nivel de Eficacia* es un indicador creciente. Es decir, a mayor valor del *Nivel de Eficacia*, mayor será la cantidad de instancias donde su Eficacia alcanzada fue la mejor disponible.

Al concluir el objetivo 3, se procede a construir una tabla que compare el *Nivel de Eficacia* entre las versiones Heavy y Soft desarrolladas en la presente investigación con los algoritmos metaheurísticos seleccionados.

3.14. Método

Según Rodríguez y Pérez (2017), los métodos de investigación consisten en las diferentes perspectivas que tiene el investigador de acercarse al objeto de estudio. Por su parte, Bello (2008, como se citó en Esteban, 2018) indica que la metodología de la investigación tecnológica es diferente a la de la investigación básica. La Investigación tecnológica usa diferentes métodos y técnicas, tal que: “parte de la observación-reflexión-diseño-praxis de la necesidad de análisis-síntesis del objeto de investigación (…)” (p.3).

Para la presente investigación se usó dos tipos de métodos:

- *Método Analítico-Sintético*, el cual posibilita la descomposición de un todo en sus partes para estudiarlas de manera individual, y luego unirlas, posibilitando encontrar relaciones entre esas partes.

- *Método de Analogías,* el cual, según Rodríguez y Pérez (2017), consiste en “inferir relaciones o consecuencias semejantes en fenómenos parecidos” (p.191). Es clasificado como un método de construcción de conocimiento que propone una consecuencia semejante a la consecuencia de otro fenómeno.

3.15. Descripción del proceso de investigación tecnológica

Para la presente investigación, al ser del tipo de tecnologías formales, se usó una modificación del proceso descrito por García y Trejo (2012, como se citó en Casaño, 2016) en su libro "*La perspectiva de la investigación tecnológica en educación"*: Observar, Determinar el problema, Documentar, Reflexionar, Elaborar el proyecto de intervención, Valorar, Comunicar, Implementar, Seguimiento y Evaluación. Este proceso fue tomado como el plan para realizar la modificación del algoritmo, con el objetivo de mejorar su capacidad de encontrar mejores soluciones y, por ende, mejorar el nivel de eficacia ante problemas de optimización combinatoria.

3.15.1. Observar

El pensamiento de la heurística del Vecino más Cercano con Sacrificio Cortoplacista 2Opt (SCA_2Opt) consiste en construir una ruta tomando la combinación de dos criterios: para un determinado punto de apalancamiento, aplicar un único Sacrificio Cortoplacista y, al resto de nodos, aplicar el pensamiento del Vecino más Cercano. De tal manera que dicha ruta pueda mejorar con la aplicación de la heurística 2Opt.

Del diagrama de flujo de esta heurística (ver Figura 10), se pudo observar que el algoritmo SCA_2Opt, desarrollado por Pérez (2011) presentó:

- Una estructura basada en la trayectoria en el proceso de búsqueda de soluciones, la cual por medio de procesos iterativos mejora una ruta que se va actualizando.

- Un criterio de parada, el cual consistió en finalizar el algoritmo cuando, al aplicar la heurística 2Opt sobre la ruta, producto del último apalancamiento, no se logrará mejorar la solución almacenada en iteraciones sucesivas (Ver Figura 11, Criterio 2).

- Una política para realizar la selección de arcos a intercambiar, basada en la selección First Improvement. Donde, si bien Pérez (2011) no indicó explícitamente su política de intercambios 2Opt, al revisar su código 2Opt (en sus Anexos), se identificó la aplicación de una variante de dicha política.

  Con base a la regla del pensamiento sistémico que afirma: “lo más rápido es lo más lento” (Senge, 2010, p.83), para nuestra investigación se prefirió usar la política Best Improvement. Es decir, retrasar la realización del intercambio 2Opt con la esperanza de encontrar mejores soluciones. Dicha decisión encuentra sustento en el inciso “c” del numeral “3.15.5. Reflexionar”, en el cual, mediante la experimentación, la política Best Improvement proporcionó mejores resultados que First Improvement para las instancias del TSP revisadas.

- Un criterio para realizar el Intercambio 2Opt, el cual, para Pérez (2011), fue una variante usada para el Problema del Agente Viajero Simétrico por Poveda, Gómez y León (2008). Donde al revisar otros autores, se notó que existen leves diferencias entre versiones, lo cual puede conllevar a soluciones diferentes.

3.15.2. Determinar el problema del objeto de estudio

Con respecto al SCA_2Opt, más que un problema, al ser producto de un análisis inspirado en analogías con el pensamiento sistémico, se evidenciaron posibilidades sin explorar, debido a la amplitud del campo de inspiración, a partir del cual se realizaron modificaciones inspiradas en analogías dentro del pensamiento sistémico y la estructura del algoritmo.

3.15.3. Documentar

Al revisar los algoritmos desarrollados por Pérez (2011): SCA_2Opt y SCA_2Opt_r, se evidenció la ausencia de otras investigaciones enfocadas en modificar el algoritmo del Vecino más Cercano tomando como inspiración el pensamiento sistémico. Según la Tabla 1, se evidenció, a nivel internacional, una mayor cantidad de investigaciones para el desarrollo de metaheurísticas y una mínima cantidad de investigaciones con el tema de la heurística del Vecino más Cercano. Por lo cual, en la sección de antecedentes se procedió a listar publicaciones de las cuales se pudo extraer ideas o analogías con el pensamiento sistémico, para poder ser fuente de inspiración para el desarrollo de la presente investigación.

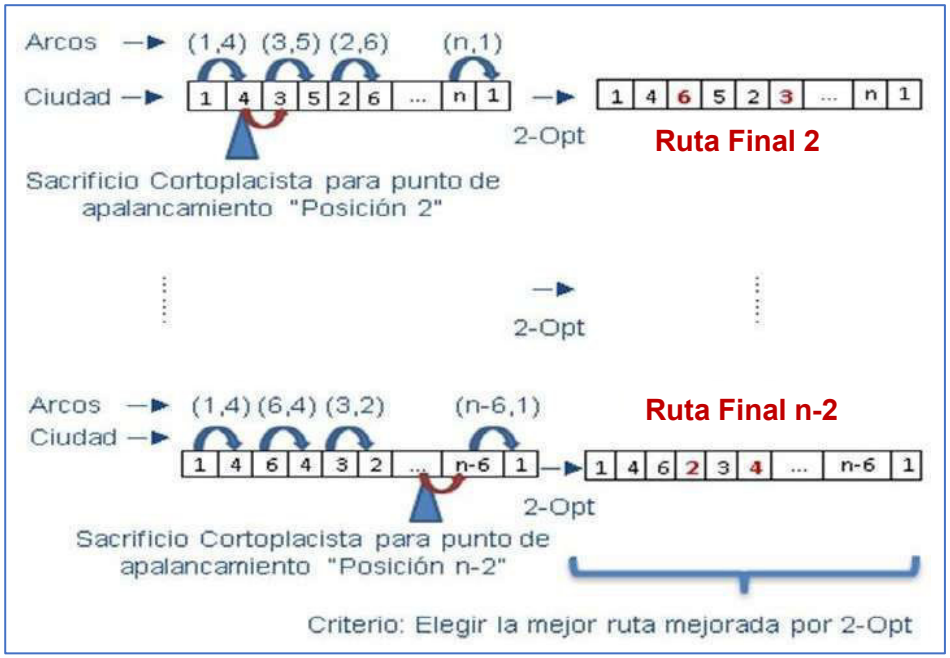
3.15.4. Análisis de la estructura del SCA_2Opt

Al enfocarnos en el pensamiento que indica que la estructura influye sobre la conducta, entendiendo como estructura no a las partes que componen el todo, sino, como indica Espinoza (2010), a la configuración de las interrelaciones existentes entre las componentes que conectan al sistema e interactúan entre sí. Se acota que, para nuestra versión del 2Opt, usamos la política Best Improvement, pero evitamos llamarla 2OptBI para no extender en demasía el nombre final del algoritmo. Por lo cual, la denominamos simplemente 2Opt. A continuación, podemos dividir el proceso de análisis para la construcción de los algoritmos propuestos en:

a. ***Etapa 1: Identificación de la estructura sistémica del pensamiento del Sacrificio Cortoplacista Adaptativo 2Opt.*** Para identificar mejor la estructura sistémica de la heurística SCA_2Opt, desarrollada por Pérez (2011), y las interrelaciones inherentes a su diseño, de tal manera que esa estructura nos lleve a una consecuencia traducida en una ruta, se analizó su funcionamiento, el cual se indica en la Figura 34.

Figura 34

Estructura de búsqueda del Vecino más Cercano con Sacrificio Cortoplacista Adaptativo (VMC_SCA)



En la Figura 34 se observa que, según la posición donde se realice el Sacrificio Cortoplacista, la aplicación de dicho pensamiento funciona como una “Prohibición de Arcos”, donde al seleccionar un nodo para aplicar dicho sacrificio, se prohíbe la elección del arco con menor costo, para así poder elegir el segundo mejor arco. El proceso de búsqueda original del Sacrificio Cortoplacista Adaptativo 2Opt de Pérez (2011) consistió en realizar una búsqueda tipo trayectoria, es decir, la ruta se va actualizando con base en un criterio de comparación. Por lo cual, para la presente investigación se optó por un proceso de búsqueda poblacional (mayor detalle ver 3.15.5. Reflexionar, inciso “a. Con respecto a la estructura del proceso de búsqueda de soluciones”), cuya estructura resulta más favorable que la basada en trayectoria, con el objetivo de explorar una mayor diversidad de soluciones factibles. Adicionalmente, se trabajó con algoritmos basados en la trayectoria.

b. ***Etapa 2: Identificación de elementos en la estructura sistémica del pensamiento del Sacrificio Cortoplacista Adaptativo 2Opt*.** Se pudieron identificar los siguientes elementos de la estructura sistémica, según la Figura 34, los cuales son descritos a continuación:

   i. Ubicaciones de las ciudades (nodos)

   ii. Arcos formados por la interacción entre ciudades

   iii. Posición de la aplicación del Sacrificio Cortoplacista

   iv. Aplicación del pensamiento del Vecino más Cercano

   v. Momento de aplicación de heurística 2Opt

   vi. Momento de aplicación del criterio de selección de la mejor solución.

   De los seis elementos identificados, según un principio de Senge (2010), pequeños cambios pueden producir efectos en el largo plazo cuando son aplicados en ciertas ubicaciones, las cuales denominamos puntos de apalancamiento. Así, se trató de determinar dónde producir esos cambios y que sean mínimos, para no presionar en demasía al proceso de búsqueda del algoritmo.

3.15.5. Análisis del diseño del algoritmo

Siendo el pensamiento sistémico un campo aún no explorado como fuente de inspiración para encontrar analogías destinadas a mejorar el pensamiento miope de la heurística del Vecino más Cercano, Pérez (2011) introduce el concepto de Sacrificio Cortoplacista Adaptativo. Este consiste en, desde un nodo en específico, sacrificar ir al nodo más cercano para tomar la decisión de ir al segundo nodo más cercano y para el resto de los nodos seguir aplicando el pensamiento original del Vecino más Cercano. Dicho sacrificio

que consiste en elegir la segunda mejor alternativa fue denominado por Pérez (2011) como Sacrificio Cortoplacista Adaptativo de nivel 2. Por lo cual, si se elige ir al “k-ésimo” mejor nodo, se define como Sacrificio Cortoplacista Adaptativo de nivel “k”, tal que k < n, donde n = orden del grafo.

Por nuestra parte, nos planteamos las siguientes interrogantes:

- *¿Por qué la estructura del algoritmo solo aplica un único Sacrificio Cortoplacista Adaptativo?*

Por ello, se planteó la aplicación de más de un Sacrificio Cortoplacista Adaptativo, a lo que denominamos MultiSacrificio Cortoplacista Adaptativo (MultiSCA), basados en el concepto de ciclos reforzadores, buscando producir cambios en un ciclo virtuoso.

- *¿Qué pasaría si modificamos la estructura del algoritmo para plantearnos múltiples sacrificios?*

Esto nos llevó a preguntarnos cuántos Sacrificios Cortoplacistas Adaptativos es conveniente realizar y cómo adicionar estos de tal manera que resulten de la inspiración del pensamiento sistémico, basados en la lección del juego de la cerveza, la cual indica que la “estructura influye sobre la conducta” (Senge, 2010, p.53), y “cuanto más se presiona, más presiona el sistema” (Senge, 2010, p.78). Porque según cuantos sacrificios se hagan y cuándo se realicen, podemos presionar en mayor o menor medida al sistema y llevar a realizar una búsqueda errática (acción compensadora) de soluciones al algoritmo o una búsqueda eficiente si evitamos presionar en demasía al sistema.

Ante lo indicado, nos enfocamos en la búsqueda de analogías del pensamiento sistémico, las cuales permitan ser fuente de inspiración para la estructura de nuestro MultiSacrificio Cortoplacista Adaptativo y, a su vez, que dicha estructura no presione significativamente al proceso de búsqueda de soluciones de nuestro algoritmo.

La contribución de Pérez (2011), con un nuevo algoritmo inspirado en el pensamiento sistémico, más específicamente en el concepto de puntos de apalancamiento, para realizar una pequeña pero significativa modificación a la heurística miope del Vecino más Cercano fue considerada como punto de partida para la construcción de un nuevo algoritmo, con la finalidad de encontrar mejores soluciones en problemas de optimización combinatoria. Al estudiar el proceso que realiza el SCA_2Opt y con base en las observaciones descritas en los incisos “a” y “b” del numeral “3.15.1. Observar”, se plantearon los siguientes mecanismos para mejorar la lógica de búsqueda del algoritmo:

a. ***Con respecto a la estructura del proceso de búsqueda de soluciones.*** Debemos notar que Pérez (2011) utilizó una estructura de búsqueda de soluciones basada en la trayectoria y para el desarrollo de nuestros algoritmos proponemos inicialmente una estructura de búsqueda basada en la población, sustentada mediante el numeral 2.2.1. Pensamiento sistémico, inciso “a” estructura sistémica, el cual nos indica la importancia de la estructura para la consecución de los objetivos. En este caso, al trabajar con un algoritmo, nos basamos en Alancay, Villagra y Villagra (2016a), quienes afirman que la búsqueda basada en la población tiende a encontrar mejores soluciones que las basadas en la trayectoria, debido a que exploran distintas áreas del espacio de solución de manera simultánea***.***

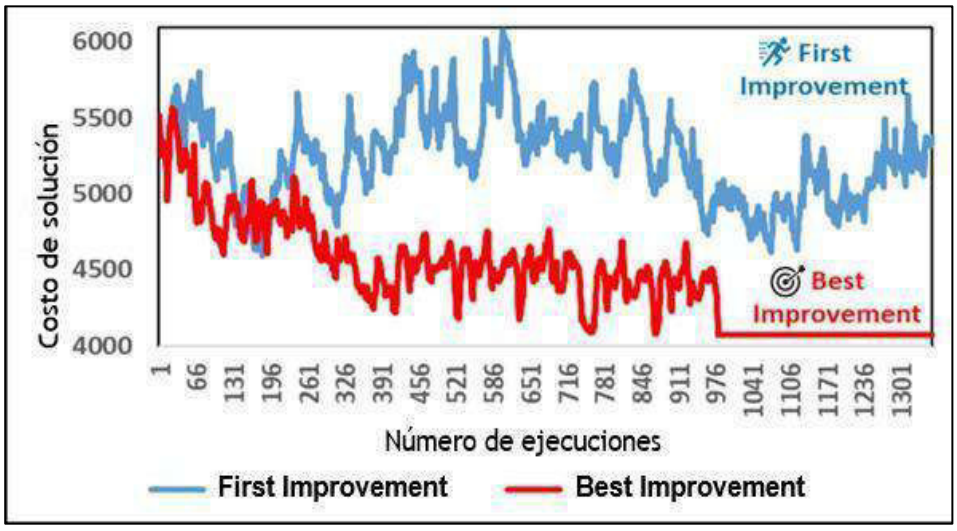
b. ***Con respecto al criterio de parada***. La ejecución del algoritmo original desarrollado por Pérez (2011) finaliza al aplicar la heurística 2Opt a la ruta construida por el último punto de apalancamiento. Con base en el pensamiento sistémico, donde las interacciones producen un efecto en el sistema, sea positivo o negativo, observamos el efecto positivo que logra la ejecución de nuestro algoritmo 2Opt para mejorar la ruta del Vecino más Cercano. Por lo cual, ahora buscamos incrementar dicho efecto positivo, incrementando la cantidad de ejecuciones de la heurística, donde la

ruta mejorada en una determinada ejecución del 2Opt dentro del Uni2Opt, será la ruta inicial para mejorar por la siguiente ejecución 2Opt, lo que se denomina ciclos repetitivos (para mayor detalle, ver numeral 2.2.1 Pensamiento sistémico, inciso “b” Lenguaje del pensamiento sistémico). Por tal motivo, para incrementar el efecto positivo de la aplicación de la heurística 2Opt en nuestro diseño, se propuso la ejecución de múltiples heurísticas 2Opt, donde dicha cantidad fue determinada por ensayo y error, según la cantidad mínima que brinde los mejores resultados (para mayor detalle, ver Anexo 19).

c. ***Con respecto a la política para seleccionar los nodos a intercambiar (intercambio 2Opt).*** Con base en el pensamiento de “lo más rápido es lo más lento”, se decidió optar por la política basada en Best Improvement, es decir, escoger el mejor de los mejores en cada ejecución. Por lo cual, nuestro intercambio 2Opt tuvo el nombre 2OptBI. A continuación, en la Figura 35, se muestra el comportamiento de la mejor solución según la política aplicada.

Figura 35

Solución con política First Improvement y Best Imprevement



Nota. Para instancia tsp225

d. Con respecto al diseño del criterio para realizar el intercambio 2OptBI. Al revisar el criterio definido por Pérez (2011), se pensó en modificarlo para buscar mejores soluciones, buscando una estructura que construya la mejor solución posible en el menor tiempo posible; pero lo anterior es contrario a una de las reglas del pensamiento sistémico, que afirma lo siguiente: “lo más rápido es lo más lento”. Razón por lo cual se procedió a analizar qué tan factible sería la modificación del criterio utilizado por Pérez (2011) en su intercambio 2Opt, aplicado a nuestro 2OptBI. Por esta razón, se definieron dos criterios para realizar o no el intercambio 2OptBI:

- Un criterio de mejoramiento constante de la solución.

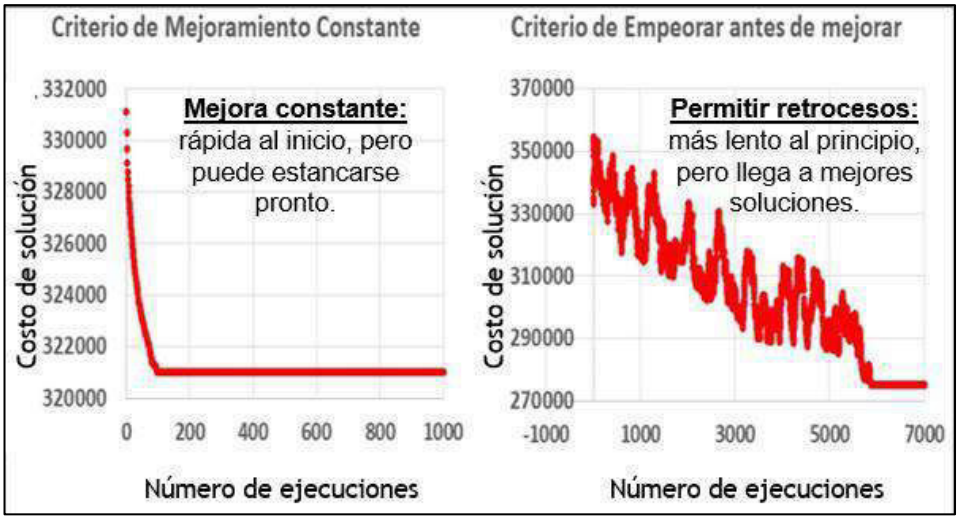
- Un criterio de empeorar antes de mejorar la solución.

Notar que los criterios descritos son distintos a las políticas First Improvement (FI) o una Best Improvement (BI), debido a que dichas políticas se centran en elegir en qué etapa se seleccionan los nodos a intercambiar (el intercambio de nodos ya está permitido), mientras que los criterios descritos se centran en un paso previo, el cual decide si se aplica o no el intercambio de nodos.

Para realizar la ejecución de las dos versiones del 2OptBI planteadas: criterio de mejoramiento constante y otra basada en empeorar antes de mejorar, se seleccionó la instancia pr1002 del Problema del Agente Viajero Simétrico de TSPLIB, a la cual se le realizó la ejecución respectiva y al evaluar cual de ambos criterios producía los menores costos se construyó la Figura 36.

Figura 36

Solución por "mejora constante" vs "empeorar para mejorar"



Nota. Para instancia pr1002

En la Figura 36 se observó que:

- Si bien el criterio de solo *mejoramiento constante* de la solución necesita menos ejecuciones para converger (menos tiempo), la solución no resultó ser la mejor en comparación con el criterio de *empeorar antes de mejorar.*

- El criterio de *empeorar antes de mejorar* encontró las mejores soluciones, empleando un mayor tiempo, lo cual va acorde con la regla del pensamiento sistémico de Senge (2010): "lo más rápido es lo más lento" (p.83). Es decir, para encontrar las mejores soluciones, lo más rápido no siempre es lo mejor.

Al probar con otras instancias, se encontraron resultados similares. Por lo cual, el diseño para permitir los intercambios 2OptBI, definido siguiendo la perspectiva dada por Pérez (2011) resultó ser eficiente por cumplir la regla que indica que "lo más rápido es lo más lento". Lo que puede ser interpretado en nuestro algoritmo como un proceso que retrasa la aparición del óptimo local.

e. ***Etapa 3: Analogías con partes del pensamiento sistémico.*** Al plantear la aplicación de múltiples sacrificios cortoplacistas, implícitamente se trabajó con el concepto de prohibición de arcos. Por lo cual, se buscaron analogías inspiradas en el pensamiento sistémico para tomarlas en la modificación de la estructura sistémica descrita en inciso “b” de 3.15.5. Debido a esto, surgen dos preguntas con respecto a la aplicación de múltiples sacrificios.

Pregunta 1: ¿En cuáles posiciones debemos realizar el resto de los sacrificios cortoplacistas y bajo qué criterio aplicarlos? Para ello, Senge (2010), en su libro “La Quinta Disciplina”, muestra el experimento del juego de la cerveza, en el cual se observó cómo la interrelación entre los jugadores creó un efecto compensador por parte del sistema, el cual entró en colapso cuando cada jugador tomó sus decisiones pensando en lo mejor para cada uno de ellos, sin pensar en cómo estas decisiones presionaban al sistema, provocando que los resultados de sus decisiones retornen y los afecten. Lo que Senge (2010) resumió en que cada acción produjo un efecto que retornó al inicio de la misma, modificándola. Dentro de las conclusiones desarrolladas de dicho juego, se indican las siguientes:

Analogía 1: *Las personas pertenecientes a una misma estructura sistémica, a pesar de sus diferencias, nos darán resultados similares.*

En la estructura sistémica del SCA_2Opt, identificada en el numeral 3.15.4, inciso b (parte ii. Arcos formados por la interacción entre ciudades), se identificó como personas a los arcos formados al unir dos ciudades adyacentes. Por lo cual, según la analogía 1, durante la construcción de una nueva ruta se debe evitar la aparición de arcos idénticos a los determinados por el pensamiento del Vecino más Cercano, para así evitar resultados similares.

Es decir, se debió definir una prohibición de arcos porque, según la analogía, arcos pertenecientes a una misma estructura (personas) van a producir rutas similares (resultados similares) y nosotros buscamos rutas mucho mejores a las iniciales. Por lo cual, necesitamos individuos diferentes, los cuales produzcan resultados diferentes.

Pregunta 2: ¿Cuál debe ser el alcance de nuestro MultiSacrificio Cortoplacista? Analizamos la analogía 2 para dar respuesta.

Analogía 2: *Cuanto más se presiona, más presiona el sistema.*

Es crucial evitar presionar en exceso al sistema, porque esto generaría lo que Senge (2010) denomina: “retroalimentación compensadora”, la cual consiste en el fenómeno de aplicar cambios en el sistema con el objetivo de mejorar los resultados finales; pero si son aplicados de manera agresiva, el sistema tiende a reaccionar con acciones (ciclos compensadores) que podrían generar resultados inferiores a los originales. Tal retroalimentación compensadora, también conocida como un regulador, es capaz de generar desorden y un resultado diferente al deseado. Por lo tanto, se planteó inicialmente una cantidad pequeña de Sacrificios Cortoplacistas a realizar; para nuestro algoritmo, se consideró que una cantidad entre 3 a 5 sería adecuada. Dichas cantidades fueron analizadas más adelante por medio de ensayo y error, y se eligió la que produjo los mejores resultados (para mayor detalle del análisis mencionado, ver numeral 3.16.3, inciso “a. Determinación del parámetro A-Amplitud”, Tabla 5). Como resultado de la comparación del algoritmo SCA_2Opt de Pérez (2011) con las analogías 1 y 2, pasamos a brindar algunos conceptos que se usaron para la construcción de los algoritmos a proponer, los cuales son detallados en el numeral 3.15.6., desarrollado a continuación.

3.15.6. Implementación de los algoritmos desarrollados

En esta etapa, con base en los conceptos explicados en el numeral 2.2.23, se procedió con la implementación de los algoritmos propuestos para su posterior experimentación. Durante la presente investigación, la implementación se desarrolló en el software libre NetBeans, mediante el uso del lenguaje de programación Java.

3.16. Algoritmos desarrollados en su versión Heavy

Con base en las versiones originales desarrolladas por Pérez (2011): SCA_2Opt y SCA_2Opt_r, se desarrollaron los algoritmos:

- SCA(1)_Multi2Opt

- SCA(1)_Multi2Opt_r

- SCA(5)_Multi2Opt.

3.16.1. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Multi2Opt: SCA(1)_Multi2Opt

Algoritmo metaheurístico, el cual solo prohíbe un arco por cada apalancamiento (ver Figura 23), para luego construir la ruta asociada a dicho apalancamiento, usando el pensamiento del Vecino más Cercano y luego ser mejorada por el criterio Multi2Opt (ver Figura 29).

En esencia, el criterio de búsqueda del SCA(1) es equivalente al Sacrificio Cortoplacista Adaptativo (SCA) desarrollado por Pérez (2011); lo que varió es el enfoque de búsqueda para nuestro algoritmo, el cual consistió en uno *del tipo poblacional,* al generar una ruta por cada punto de apalancamiento, la cual fue analizada de manera independiente.

Al añadir el criterio Multi2Opt a cada ruta generada por su respectivo apalancamiento, *búsqueda tipo poblacional*, se buscó explorar una mayor cantidad de rutas del espacio de soluciones factibles y no limitarnos a una *búsqueda de tipo trayectoria*, para así revisar una mayor cantidad de posibles soluciones e ir mejorando cada una sin detenerse de manera prematura.

A continuación, se describen los pasos a seguir para la aplicación del SCA(1)_Multi2Opt:

Paso 1: Se define Número de Ejecuciones = Número de nodos × k, donde k = 6 (mayor detalle en Anexo 19). Si se desea el valor de “k” puede variar a otro criterio según el investigador). Luego ir al Paso 2.

Paso 2: Obtener ruta con la heurística del Vecino más Cercano.

Paso 3: Definir el arco prohibido para el apalancamiento “t” / 1≤ t ≤n-6.

Paso 4: Asignar “0” como peso del arco prohibido, durante la construcción de la ruta para apalancamiento “t” / 1 ≤ t ≤ n-6.

Paso 5: Construir ruta con el criterio del Vecino más Cercano: Ruta “t” / 1 ≤ t ≤ n-6.

Paso 6: Mejorar ruta “t” con la aplicación del criterio Multi2Opt, ∀ t / 1 ≤ t ≤ n-6

Paso 7: Almacenar la ruta mejorada “t” con su distancia recorrida asociada, ∀ t / 1 ≤ t ≤ n-6.

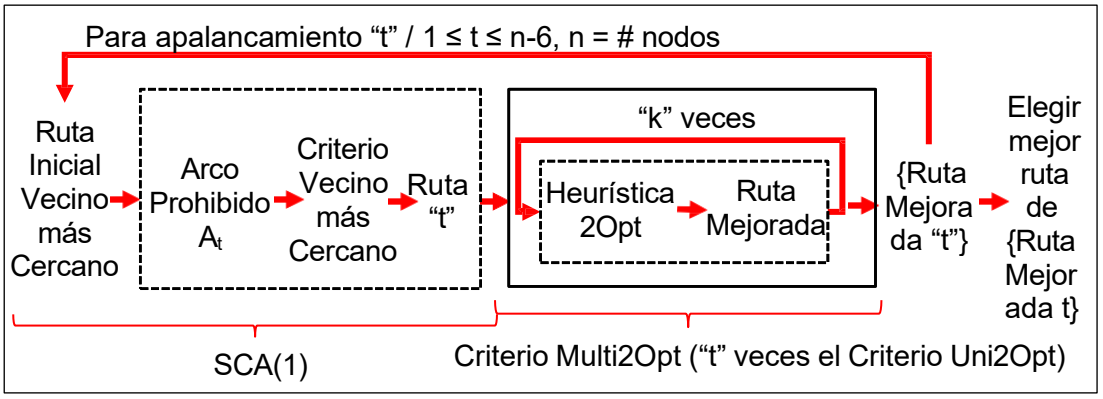
Paso 8: Ir al Paso 2 y elegir otro valor de “t” aún no tomado, en caso ya no tener más valores “t” a recorrer ir al paso 9.

Paso 9: Del conjunto de rutas mejoradas “t” guardadas elegir la que presente menor distancia recorrida.

Para entender mejor el funcionamiento del algoritmo SCA(1)_Multi2Opt se planteó su diagrama de bloques en la Figura 37.

Figura 37

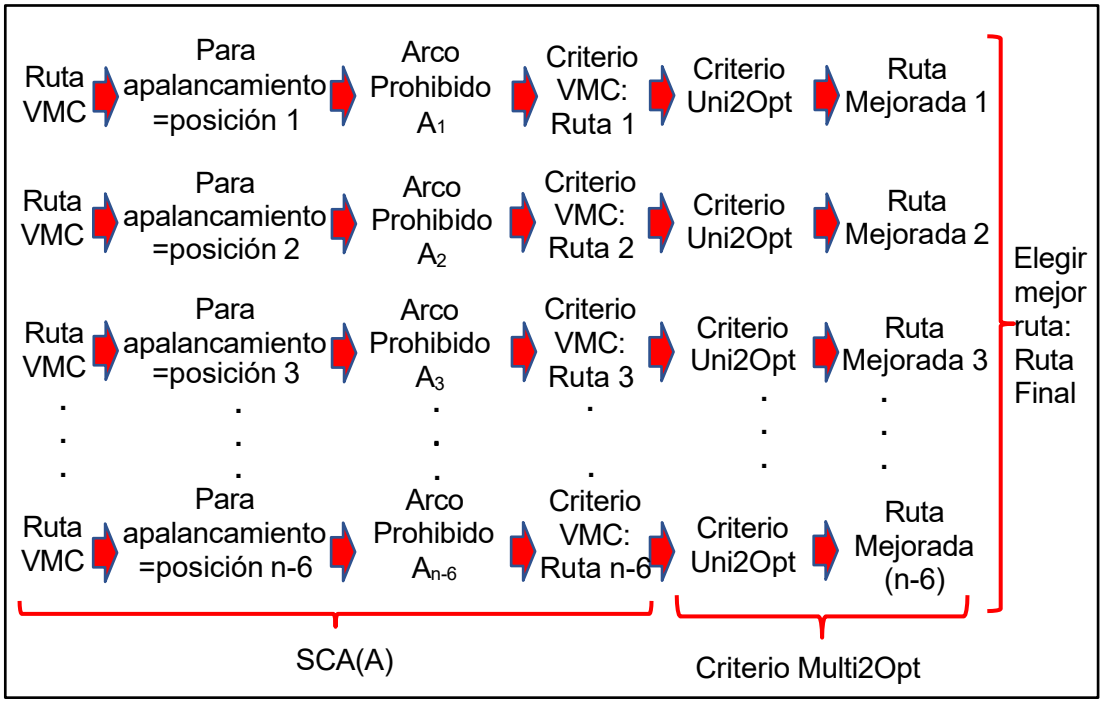
Diagrama de bloques del SCA(1)_Multi2Opt



Partiendo de la Figura 37, al desglosar el diagrama de bloques del SCA(1)_Multi2Opt, se muestra su funcionamiento en la Figura 38, para un mejor entendimiento del proceso de búsqueda tipo poblacional utilizado.

Figura 38

Funcionamiento del SCA(1)_Multi2Opt

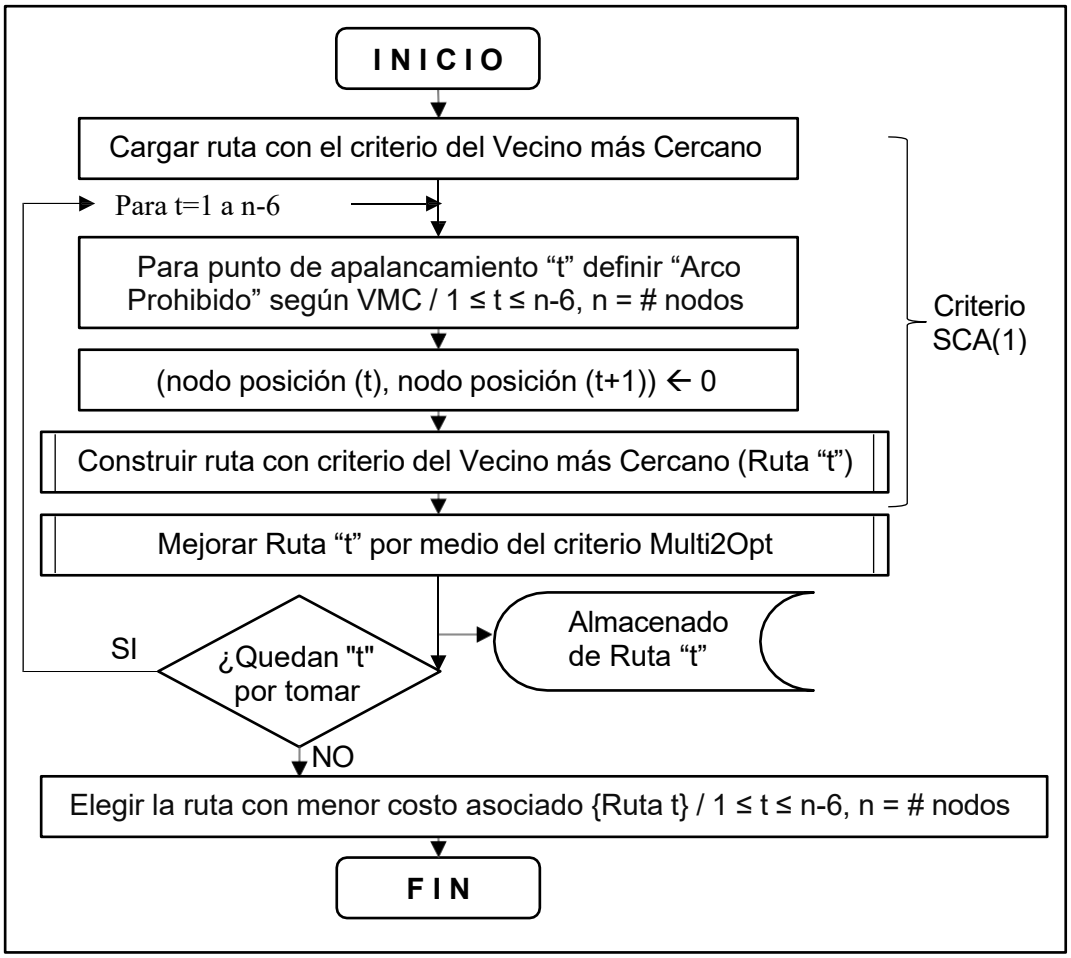


En la Figura 38, mientras el criterio original descarta rutas con base en su no mejoramiento para luego actualizarse, en nuestra propuesta, a cada ruta generada por su respectivo apalancamiento se le aplica el criterio del Vecino más Cercano, prohibiendo que el arco correspondiente aparezca en la ruta final, es decir, el SCA(1), para posteriormente ser mejorado por el Criterio Multi2Opt. Con tal modificación se busca generar una diversidad de soluciones, las cuales, de manera independiente, nos pueden conducir a mejores soluciones a largo plazo.

Para mayor detalle, se planteó el diagrama de flujo del algoritmo SCA(1)_Multi2Opt, propuesto en la Figura 39.

Figura 39

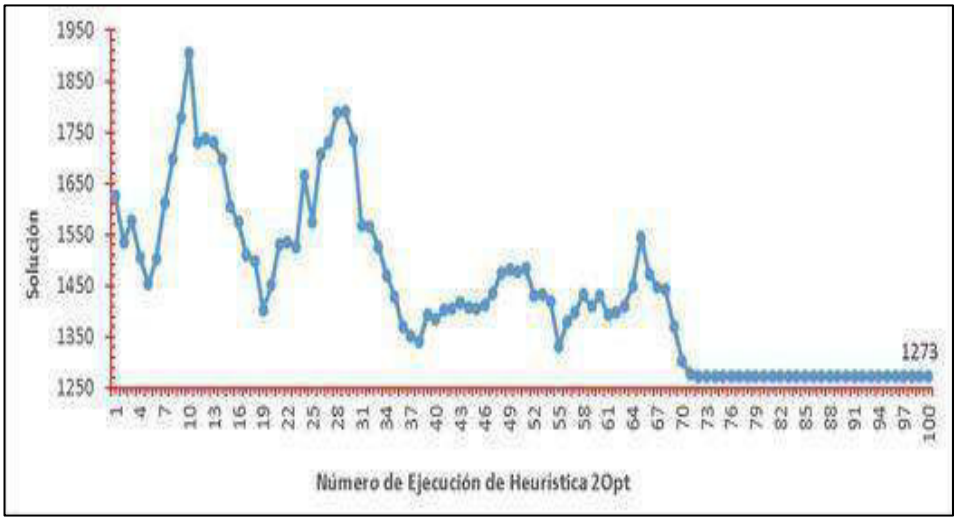
Diagrama de flujo del SCA(1)_Multi2Opt



a. ***Criterio de parada para el algoritmo SCA(1)_Multi2Opt.*** Como se identificó en el numeral 2.2.14, en el "Ejemplo de aplicación de la heurística 2Opt con política Best Improvement", la heurística 2OptBI no encontró resultados que mejoren en cada ejecución; estos pueden empeorar la solución en ejecuciones consecutivas o intercalar mejoría y empeoramiento, pero al seguir ejecutando la heurística 2OptBI, brinda una solución que, a largo plazo, será mejor que las anteriores, logrando estabilizarse, como se muestra en la Figura 40.

Figura 40

Comportamiento de la convergencia del SCA(1)_Multi2Opt



Nota. Para instancia Swiss42 en el primer punto de apalancamiento.

La Figura 40 muestra el comportamiento del mejor costo disponible para la instancia Swiss42 (de la librería virtual TSPLIB), para el primer punto de apalancamiento, donde al inicio empeora, para luego intercalar entre mejoría y empeoramiento, hasta alcanzar un valor para la variable "Costo óptimo" de 1273, el cual se estabiliza a partir de la ejecución 72.

Para determinar un valor de “k” adecuado, se ejecutó el algoritmo SCA(1)_Multi2Opt para 65 instancias seleccionadas del problema del agente viajero simétrico de TSPLIB, las cuales son indicadas en el Anexo 18. A partir del cual, se determinó una regla general, tal como se indica a continuación:

Número de ejecuciones = (Orden del Grafo) × 6

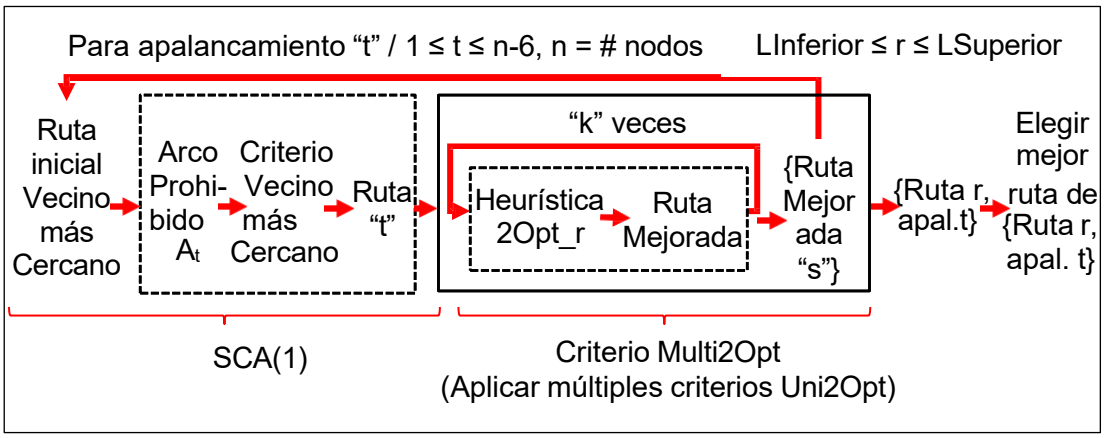
(para mayor detalle, ver las Tablas 26, 27 y 28 del Anexo 19)

3.16.2. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Multi2Opt_r: SCA(1)_Multi2Opt_r

Algoritmo que proviene del SCA(1)_Multi2Opt, al cual se le agregó el factor “r” al criterio de decisión para el intercambio de nodos de la heurística 2Opt. Sigue los mismos pasos en su versión sin el factor “r”, con excepción del paso 6, que consiste en mejorar la ruta con el 2Opt_r. Para entender mejor el algoritmo, se planteó su diagrama de bloques en la Figura 41, su funcionamiento en la Figura 42 y su diagrama de flujo en la Figura 43.

Figura 41

Diagrama de bloques del SCA(1)_Multi2Opt_r



“Ruta r, apal. s” = Mejor ruta para el valor de “r” en el apalancamiento “s” / 1≤s≤t

Figura 42

Funcionamiento del SCA(1)_Multi2Opt_r

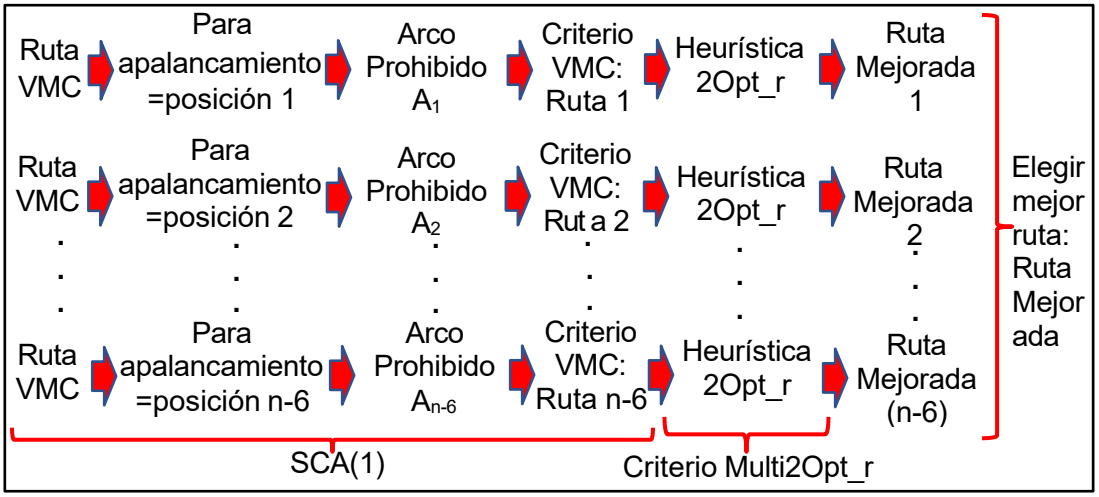
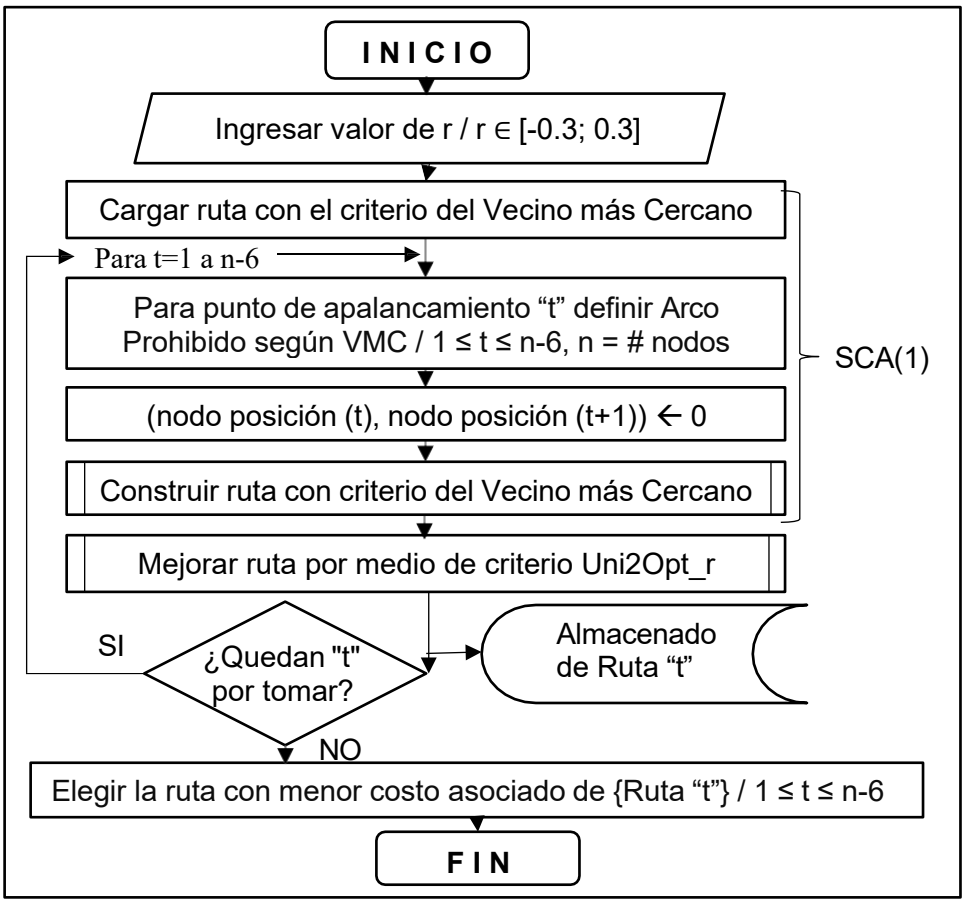


Figura 43

Diagrama de flujo del SCA(1)_Multi2Opt_r



a. Influencia del factor "r" en el costo óptimo determinado por el algoritmo SCA(1)_Multi2Opt_r.

Durante el diseño del SCA_2Opt original de Pérez (2011), surgió la idea del factor "r", el cual presenta la siguiente expresión lineal: "(1 + r)". Dicho factor "r" no estaba planificado dentro de los objetivos iniciales de Pérez (2011), por lo cual el mayor aporte con respecto al factor "r" consistió en sugerir un rango de valores permisibles entre 0 y 1. De tal manera que, al multiplicar "(1 + r)" con el mejor costo almacenado hasta dicha iteración (Costo Óptimo), se permita generar intercambios entre nodos que antes hubiesen estado prohibidos por no cumplir el criterio de decisión. Por nuestra parte, se amplió el rango de valores permitidos para "r" a valores negativos, los cuales buscan disminuir el valor del minuendo dentro del criterio para habilitar el intercambio 2Opt. A continuación, se muestra el criterio de decisión de aplicación del intercambio de nodos 2Opt, adicionando el factor "r":

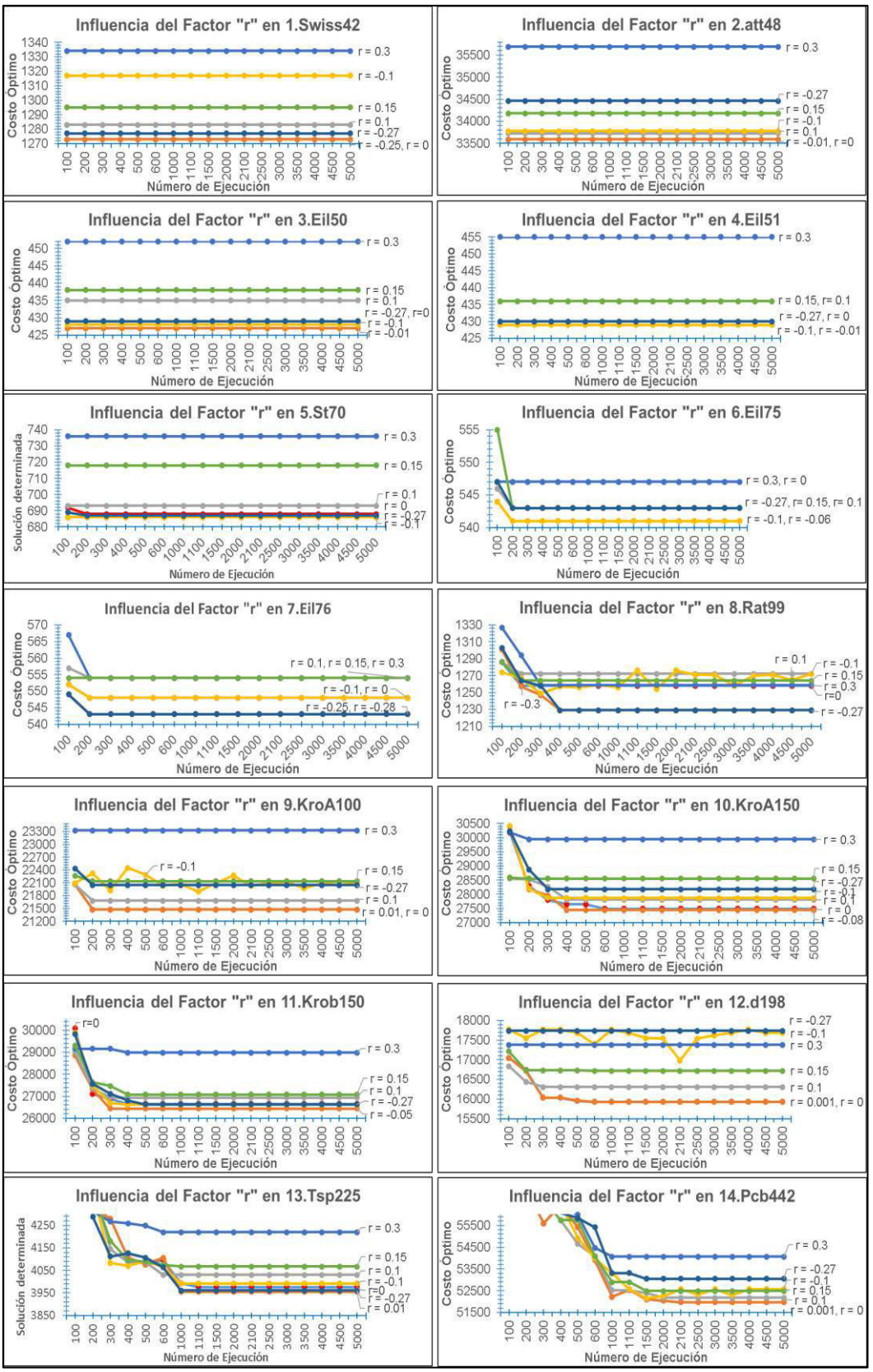
Change = (1 + r) × Costo Nuevo – Costo Actual.
Tal que, si Change < Dmax se realiza el intercambio de nodos asociado al Dmax más negativo.

De tal manera que, al permitir los valores negativos, se logró reducir el efecto de la variable "Costo Nuevo", con lo cual se buscó habilitar nuevos intercambios 2Opt que antes no hubiesen podido realizarse.

Por lo cual, como primer acercamiento para brindar una idea del efecto de adicionar "(1 + r)" dentro del criterio de intercambio 2Opt, se presenta la Figura 44. En esta figura se muestra la realización de distintas ejecuciones del algoritmo SCA(1)_Multi2Opt_r para distintas instancias del Problema del Agente Viajero Simétrico de la librería virtual TSPLIB.

Figura 44

Influencia del factor "r" en el SCA(1)_Multi2Opt_r



En la Figura 44 se aprecia que el factor “r”, dentro del SCA(1)_Multi2Opt, para las catorce instancias seleccionadas, provocó que los valores para la variable *Costo Óptimo* se desplacen por encima o por debajo de la curva original, en donde se distingue:

- Cuando r=0, se trata del algoritmo metaheurístico SCA(1)_Multi2Opt, es decir, sin influencia del factor “r”.
- Cuando r≠0, se trata del algoritmo metaheurístico SCA(1)_Multi2Opt_r, es decir, con influencia del factor “r”.

b. Determinación del rango de valores para el Factor “r”. Para la determinación del rango de valores del factor “r”, Pérez (2011) sugiere tantear los valores en base a ensayo y error, tal que $r \in [0, 1]$, de tal forma que se elija el valor de “r” que permita encontrar los mejores resultados. Tomando en cuenta que la variable “Change”, que se usa dentro del criterio para intercambiar los nodos durante la ejecución de la heurística 2Opt, es definida como:

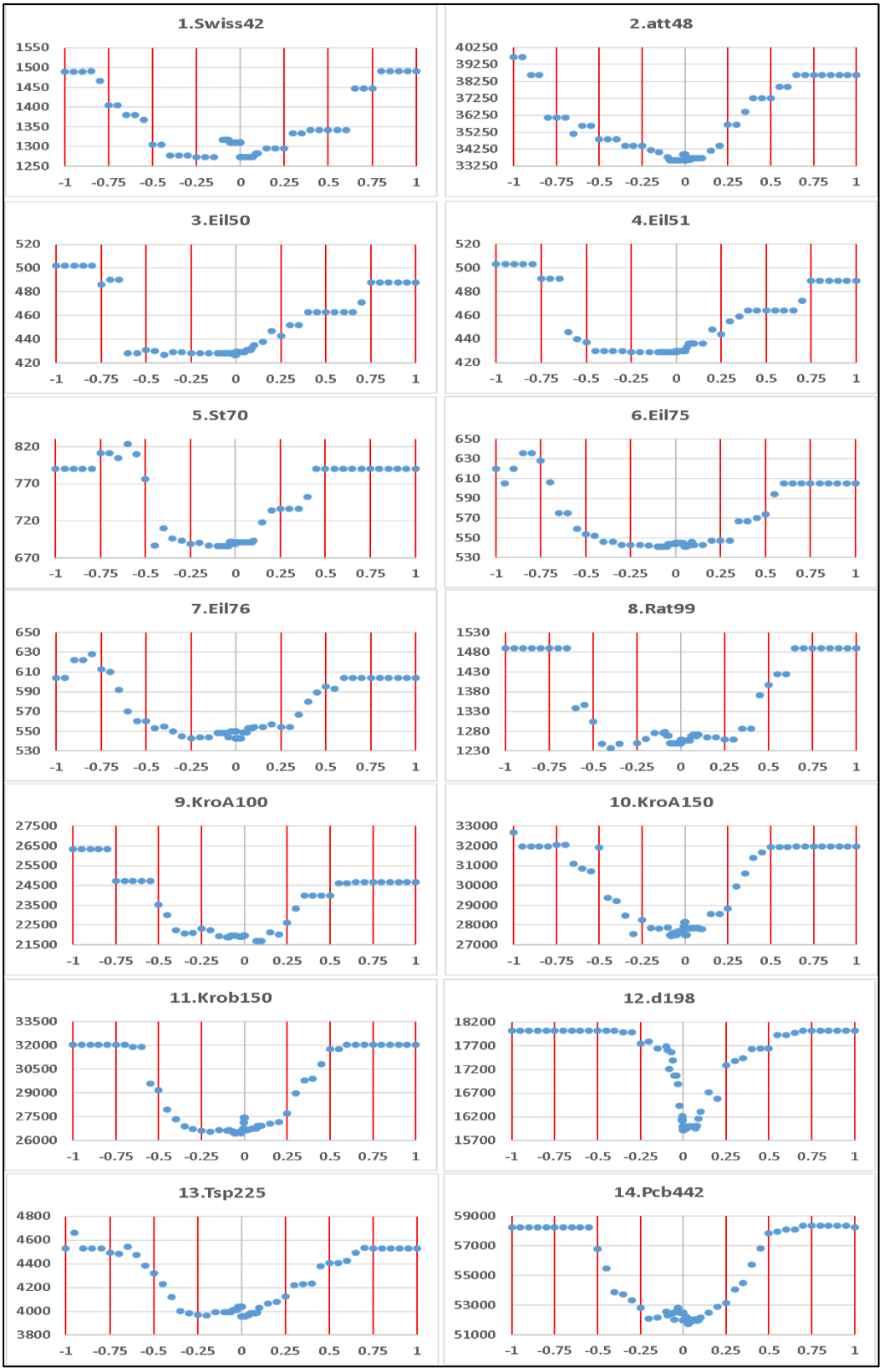
$$\text{Change} = (1 + r) \times \text{Costo Nuevo} – \text{Costo Actual}$$

Durante el desarrollo de la presente investigación, se exploró la posibilidad de tomar valores de “r” $\in [-1, 0]$, con el objetivo de disminuir el valor del minuendo “(1 + r) × Costo Nuevo” y, con esto, modificar levemente el valor de la variable “Change” para así realizar nuevos intercambios 2Opt. Para determinar si este razonamiento presentó una influencia positiva en la variable “Costo Óptimo", se analizó el comportamiento de 78 ejecuciones por cada una de las catorce instancias seleccionadas en la Figura 45, las cuales fueron tomadas de Pérez (2011), para distintos valores de “r” (39 con valores positivos y 39 con valores negativos).

Como primer paso, se realizó un análisis gráfico del valor de la variable “Costo Óptimo” según el “r” asignado para cada una de las catorce instancias seleccionadas (ver Figura 45).

Figura 45

Determinación de rango de valores de "r"



En la Figura 45, se puede observar que el comportamiento del mejor costo encontrado según el valor del factor “r” puede ajustarse a una parábola abierta hacia arriba, encontrando que los mejores costos se hallaron para valores cercanos a r = 0. Donde, a partir de un análisis gráfico, se pudo observar que, en general, los valores de r ∈ [-0.25, 0.25]. Por ello, se expuso a un segundo análisis para evaluar si dicho intervalo se debía ampliar o no.

Para valores mayores a 1 y menores a -1, se encontró que las soluciones no mejoraban, estabilizándose en un único valor.

Por ejemplo, los mejores costos se encontraron para valores de “r”:

- Para Swiss42, el menor costo se encontró para r ∈ [-0.25, 0.07]
- Para att48, el menor costo se encontró para r ∈ [-0.09, 0.006]
- Para Eil50, el menor costo se encontró para r ∈ [-0.01, -0.0009]
- Para Eil51, el menor costo se encontró para r ∈ [-0.01, -0.0009]
- Para St70, el menor costo se encontró para r ∈ [-0.1, -0.04]
- Para Eil75, el menor costo se encontró para r ∈ [-0.1, 0.06]
- Para Eil76, el menor costo se encontró para r ∈ [-0.25, 0.03]
- Para Rat99, el menor costo se encontró para r ∈ {–0.3}
- Para KroA100, el menor costo se encontró para r ∈ [0.0009, 0.02]
- Para KroA150, el menor costo se encontró para r ∈ {–0.08}
- Para Krob150, el menor costo se encontró para r ∈ [-0.06, -0.05]
- Para d198, el menor costo se encontró para r ∈ [0.0009, 0.003]
- Para Tsp225, el menor costo se encontró para r ∈ [0.0009, 0.02]
- Para Pcb442, el menor costo se encontró para r ∈ {0.03}

Como segundo paso, se realizó un análisis más exhaustivo, donde, de los 78 valores de “r” asignados para las pruebas, se seleccionó el valor de “r” que permitió encontrar el “*Mejor Costo”*, al cual lo denominamos “*Costo Óptimo”* y se definió un aproximado del rango de valores tanto positivos como negativos, para hallar el valor de la variable “*Costo Óptimo*” (ver Tabla 4).

Tabla 4

Determinación de rango de "r" para versiones con factor "r"

Instancia	Mejor costo	r (Negativos)	r (Positivos)	[-0.25;0.25]	[-0.3;0.3]
1.Swiss42	1273	[-0.25, -0.15]	[0.0009, 0.07]	X	X
2.att48	33587	[-0.09, -0.02]	[0.0009, 0.006]	X	X
3.Eil50	427	[-0.01, -0.0009]	∅	X	X
4.Eil51	429	[-0.01, -0.0009]	∅	X	X
5.St70	686	[-0.1, -0.04]	∅	X	X
6.Eil75	541	[-0.1, -0.05]	[0.05, 0.06]	X	X
7.Eil76	543	-0.25	[0.0009, 0.03]	X	X
8.Rat99	1229	-0.3	∅	X	X
9.KroA100	21474	∅	[0.0009, 0.02]	X	X
10.KroA150	27446	-0.08	∅	X	X
11.Krob150	26439	[-0.06, -0.05]	∅	X	X
12.d198	15926	∅	[0.0009, 0.003]	X	X
13.Tsp225	3955	∅	[0.0009, 0.02]	X	X
14.Pcb442	51730	∅	0.03	X	X

Nota 1. Las instancias fueron calculadas por distancia euclidiana y obtenidos de Pérez (2011)
Nota 2. Las soluciones fueron calculadas usando el algoritmo SCA(1)_Multi2Opt_r

De la Tabla 4:

- El valor de la variable *"Mejor Costo"* es el menor costo encontrado durante las 78 ejecuciones para cada instancia, donde se tanteó distintos valores de "r".

- El símbolo "∅" hace referencia a que, durante las 78 ejecuciones donde se tantearon valores para "r" (39 valores de r positivos y 39 negativos), no se encontró un "r" positivo (o negativo) que permitiera encontrar el menor valor para la variable *Mejor Costo*. Mientras el símbolo "X", indica que el menor costo se encuentra incluido en el rango de valores de "r" detallado en la columna.

- En trece de catorce instancias (92.9%), los mejores valores para la variable “Costo Óptimo” se hallaron para “r” ∈ [-0.25, 025] y solo una instancia (7.1%) para un valor de r = -0.3. Por tanto, con el objetivo de no omitir valores potencialmente buenos de “r”, se concluyó ampliar “r” / r ∈ [-0.3, 0.3].

c. ***Criterio de parada del algoritmo SCA(1)_Multi2Opt_r.*** Se utilizó el mismo valor para el criterio de parada que para su versión sin el factor “r”, es decir:

Número de ejecuciones = Orden del grafo × 6 (ver Anexo 19).

3.16.3. Sacrificio Cortoplacista Adaptativo de amplitud A, con criterio Multi2Opt: SCA(A)_Multi2Opt

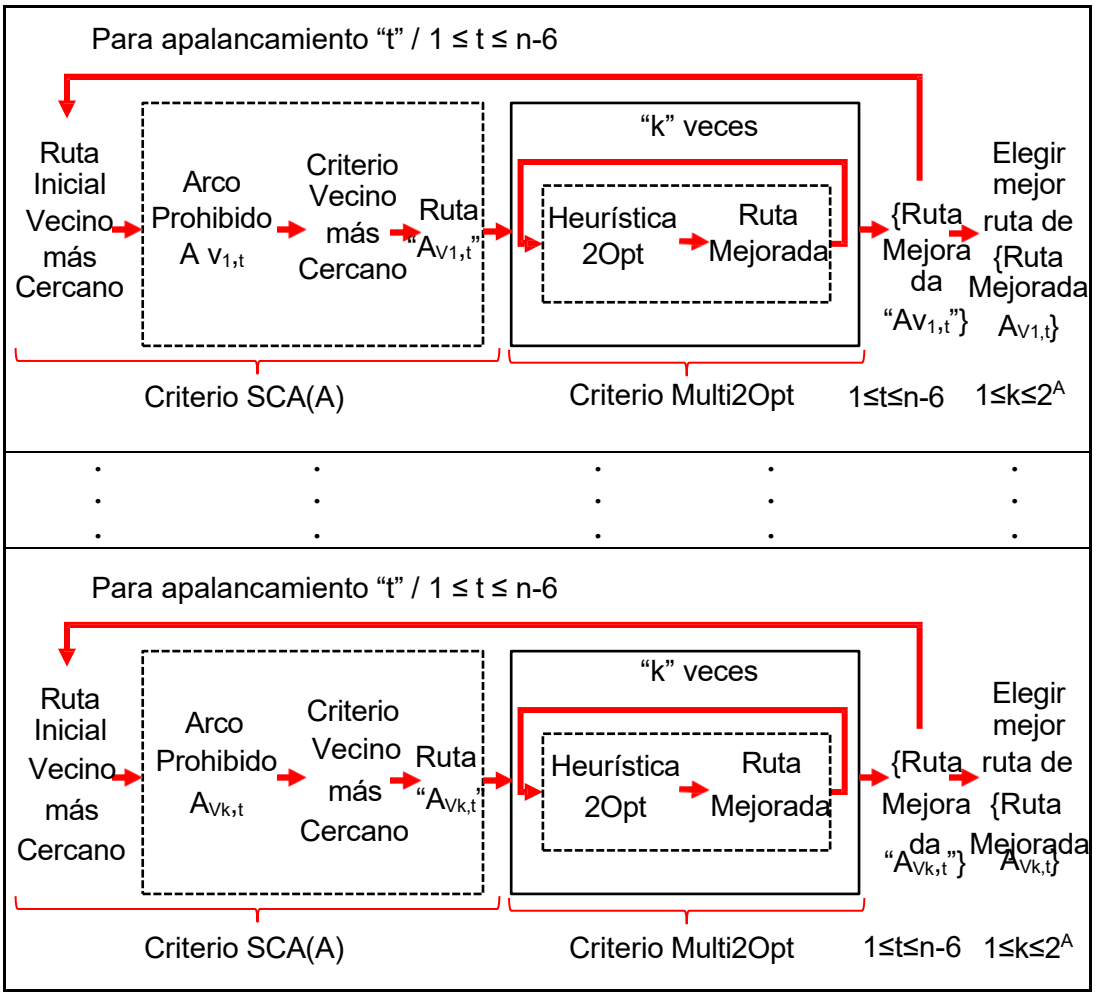
Durante el desarrollo del algoritmo SCA(1)_Multi2Opt, surgió la idea de prohibir más de un arco, por lo cual se desarrolló lo que se denominó SCA(A)_Multi2Opt, donde A > 1 representa la cantidad de arcos a prohibir, el cual se definió como un algoritmo metaheurístico con un proceso de búsqueda tipo poblacional, la cual es una estructura que explora una mayor cantidad de soluciones factibles, permitiendo así la posibilidad de encontrar una mejor solución con solo usar una búsqueda basada en la trayectoria.

Dicho proceso de búsqueda consistió en realizar un máximo de “2^A” búsquedas independientes (donde “A” es la amplitud), cada una guiada por un único vector prohibición (V_k), el cual generó tantas rutas como puntos de apalancamiento, las cuales fueron construidas por el criterio del Vecino más Cercano, para luego mejorarse con el criterio Multi2Opt. Finalmente, se elige la mejor ruta de todas las búsquedas independientes generadas.

Para mayor detalle, ver el diagrama de bloques en la Figura 46 y el funcionamiento del SCA(A)_Multi2Opt en la Figura 47.

Figura 46

Diagrama de bloques del SCA(A)_Multi2Opt

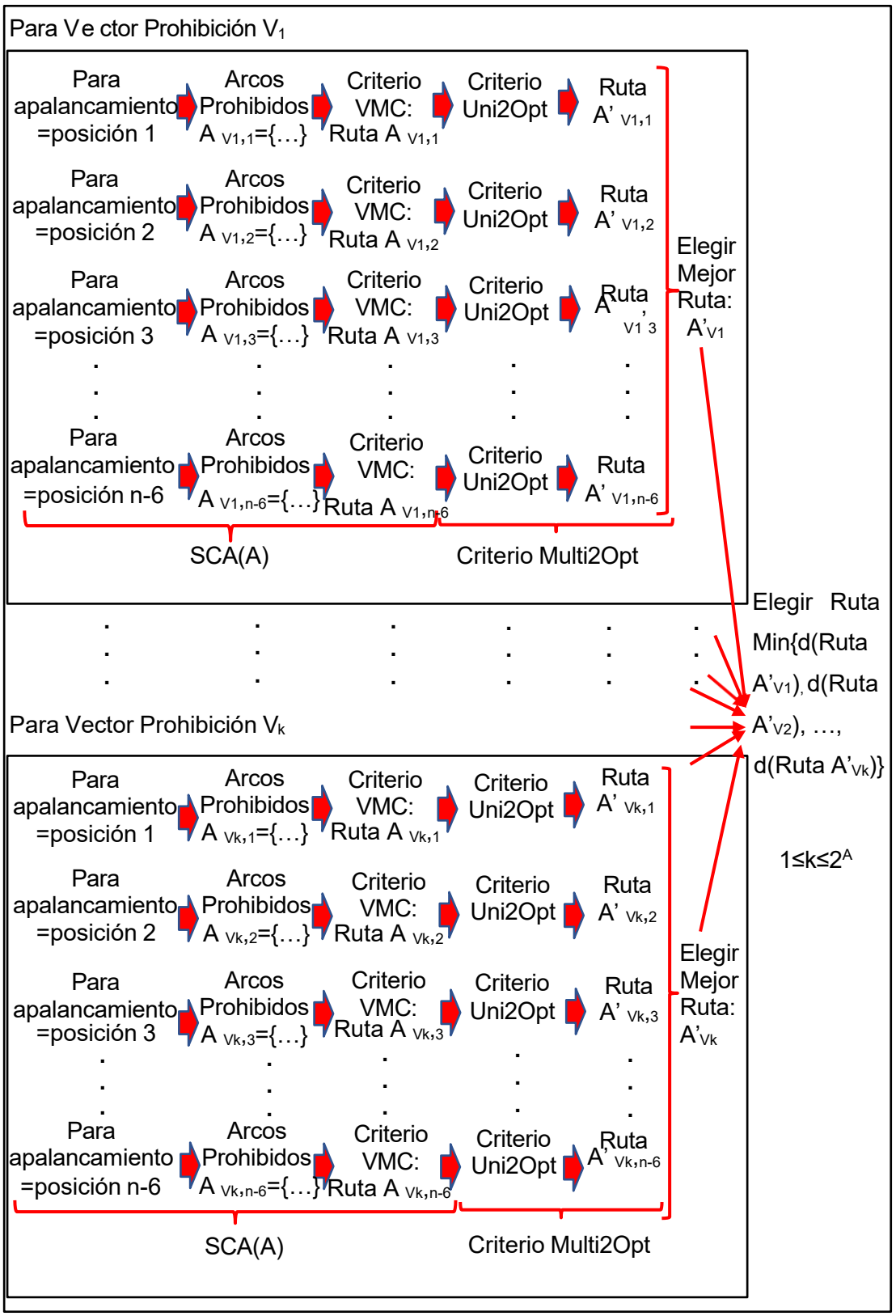


En la figura 46 se observa que, por cada ruta construida según el punto de apalancamiento respectivo y su correspondiente "Vector Prohibición", se mejora con la aplicación de la heurística de intercambio 2Opt. A dicho criterio de aplicar dicha heurística de mejoramiento a múltiples rutas se le denominó Multi2Opt (asociado a los algoritmos desarrollados en sus versiones Heavy).

A continuación, se presenta la Figura 47, la cual muestra el funcionamiento del SCA(A)_Multi2Opt.

Figura 47

Funcionamiento del SCA(A)_Multi2Opt



En la figura 47 se observa lo siguiente:

- La cantidad de vectores prohibición (V_1, V_2, ...,V_k) depende de la amplitud, quedando determinada por la siguiente fórmula:

  $k = 2^A$, donde A = Amplitud… (Ecuación 2)

- $A_{Vk,j}$: Arcos prohibidos según vector prohibición "V_k" y punto de apalancamiento "j".

  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo

- Ruta $A_{Vk,j}$: Ruta construida por la aplicación del criterio del Vecino más Cercano, con arcos prohibidos según Vector Prohibición "V_i" y punto de apalancamiento "j".

  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo.

- Ruta $A'_{Vk,j}$: Ruta mejorada al realizar la aplicación de heurística 2Opt sobre "Ruta $A_{V\,k,j}$".

  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo.

- Ruta "A_{Vk}": Ruta final resultado de la aplicación del vector prohibición "V_k" al SCA(A)_Multi2Opt.

- Ruta A_{Vk} = Ruta asociada al **Min {**d(Ruta $A'_{Vk,1}$) ; d(Ruta $A'_{Vk,2}$) ; d(Ruta $A'_{Vk,3}$) ; … ; d(Ruta $A'_{Vk,n-6}$) **}** / d(Ruta $A'_{Vk,j}$) = distancia recorrida por Ruta $A'_{Vk,j}$.

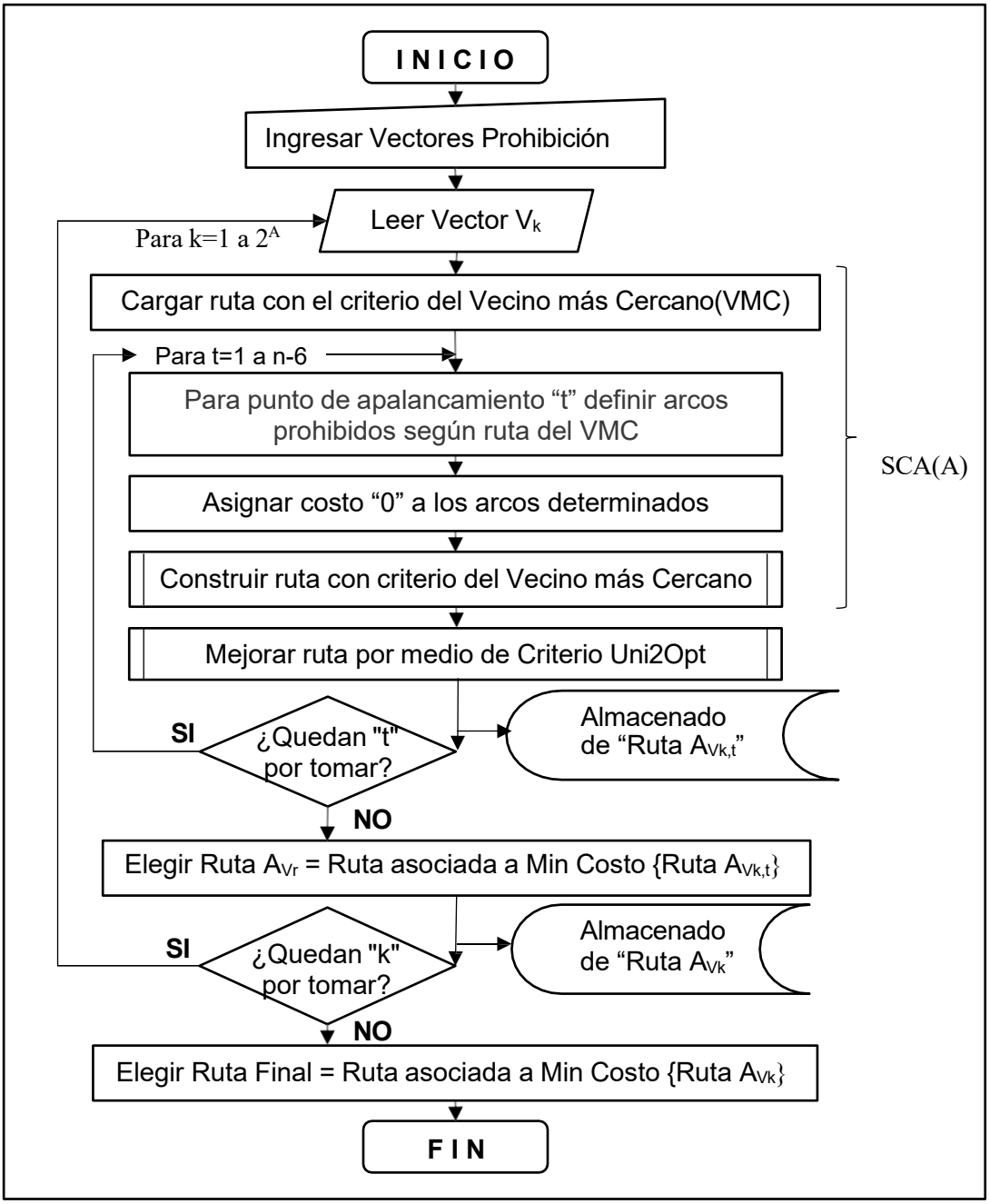
  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo.

- Ruta V_f = Ruta asociada al **Min {**d(Ruta A_{V1}); d(Ruta A_{V2}); … ; d(Ruta A_{Vk})**}** / d(Ruta A_{Vk}) = distancia recorrida por Ruta A_{Vk}.

A continuación, se presenta el funcionamiento del SCA(A)_Multi2Opt, mediante el diagrama de flujo de la Figura 48.

Figura 48

Diagrama de flujo del SCA(A)_Multi2Opt



a. Determinación del parámetro A (Amplitud). La cantidad de arcos prohibidos para dicha ejecución fue determinada por medio de ensayo y error, observándose que las mejores soluciones se encontraron para el valor de Amplitud = 5 (ver Tabla 5 y Anexo 26).

Tabla 5

Determinación de la "Amplitud" del Sacrificio Cortoplacista a utilizar

Instancia TSPLIB	Mejor Solución	Vectores Prohibición donde se encontró la mejor solución disponible
1. Swiss42	1273	Todos los Vectores Prohibición - {(0,0,0,0,0)}
2. Eil50	429	Todos los Vectores Prohibición - {(1,0,1,0,0) (0,1,0,1,0) (0,0,1,0,1) (1,0,1,0,1) (0,0,0,0,0) (1,0,1,0,0) (1,0,1,0,1)}
3. Eil51	430	Todos los Vectores Prohibición - {(1,0,1,0,0)} (0,1,0,1,0) (0,0,1,0,1) (0,0,0,0,0) (1,0,1,0,0) (1,0,1,0,1)}
4. St70	686	(1,0,0,1,0) (0,1,0,0,1) (1,1,0,0,1) (1,0,0,1,1) (1,0,0,1,0) (0,1,0,0,1) (1,0,0,0,1) (1,1,0,0,1) (1,0,0,1,1)
5. Eil75	542	**(1,0,1,0,1)**
6.Eil76	546	(1,0,0,1,0) (0,1,0,0,1) (0,1,0,1,1) (1,0,1,1,0) (1,0,1,1,1) (1,0,0,1,1)
7.Rat99	1238	**(1,0,0,0,1) (1,0,1,0,1)**
8.KroA100	21470	**(1,0,0,1,0) (1,0,0,1,1)**
9.KroA150	27243	(1,0,0,1,0) (0,1,0,0,1) (1,0,0,0,1) (1,0,0,1,1)
10.Krob150	26532	**(1,0,0,0,1) (1,1,0,0,1)**
11.d198	15902	(1,0,0,1,0)(0,1,0,0,1)(1,1,0,1,0)(0,1,1,0,1)(1,1,0,1,1) (1,0,0,1,1)
12.Tsp225	3976	(1,0,0,0,0) (1,1,0,0,0) (1,0,0,1,0) (0,1,0,0,1) (1,1,0,1,0) (0,1,1,0,1) (1,1,0,1,1) (1,0,0,1,1)
13.Pcb442	51811	(1,1,0,0,0) (1,1,1,0,0) (0,1,1,1,1) (1,1,1,1,0) (1,1,0,1,0); (0,1,1,0,1) (1,1,0,1,1); (1,1,1,0,1) (1,1,0,0,1) (1,1,1,1,1)

Nota. Los valores en rojo indican el único vector de prohibición con el que se logró encontrar la mejor solución disponible. Al probar con vectores de menor longitud, no se consiguió igualar dicha solución.

Para la construcción de la Tabla 5, se utilizaron 13 instancias del Problema del Agente Viajero Simétrico de TSPLIB, congruentes con el método de cálculo indicado por la librería virtual TSPLIB, utilizadas en la investigación de Pérez (2011).

De la Tabla 5 se observa que, para encontrar las soluciones de las instancias: 6.Eil75, 8.Rat99 y 11.Krob150, se tuvo que usar de manera obligatoria un vector prohibición de amplitud 5; en caso contrario, las soluciones brindadas hubiesen empeorado.

b. ***Criterio de parada para el algoritmo SCA(5)_Multi2Opt.*** Al igual que en la versión SCA(1)_Multi2Opt, se realizó un análisis para distintos valores de la variable "Número de Ejecuciones", para luego elegir aquel que nos brinde los mejores resultados para la variable "Costo Óptimo". Esto, por medio de experimentación en las instancias del Problema del Agente Viajero Simétrico de la librería virtual TSPLIB revisadas dentro de la investigación de Pérez (2011):

   Swiss42, att48, Eil50, Eil51, St70, Eil75, Eil76, Rat99, KroA100, KroA150, Krob150, d198, Tsp225, Pcb442.

   Se observó el comportamiento del criterio de Parada (número de ejecuciones) según lo indicado en la Tabla 6:

   - Para las 11 primeras instancias: 400 ejecuciones.
   - Para la instancia 12 – d198: 1000 ejecuciones.
   - Para la instancia 13 - Tsp225: 1100 ejecuciones.
   - Para la instancia 14 - Pcb442: 1500 ejecuciones.

   Al utilizar el criterio indicado para el SCA(1)_Multi2Opt:

   Número de ejecuciones = (Orden del grafo) × 6; con la fórmula indicada, se obtuvieron los mismos resultados que los indicados en la Tabla 6 para el número de ejecuciones.

   Por lo tanto, para esta versión del SCA(5)_Multi2Opt, también se utilizó el criterio definido según el Anexo 19, es decir:

   Número de ejecuciones = (Orden del grafo) × 6

Tabla 6

Determinación del número de ejecuciones 2Opt del Multi2Opt

Instancia TSPLIB	Número de Ejecuciones del Multi2Opt																
	100	200	300	400	500	600	1000	1100	1500	2000	2100	2500	3000	3500	4000	4500	5000
1. Swiss42	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273
2. att48	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587	33587
3. Eil50	429	429	429	429	429	429	429	429	429	429	429	429	429	429	429	429	429
4. Eil51	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430
5. St70	691	686	686	686	686	686	686	686	686	686	686	686	686	686	686	686	686
6. Eil75	546	542	542	542	542	542	542	542	542	542	542	542	542	542	542	542	542
7. Eil76	546	546	546	546	546	546	546	546	546	546	546	546	546	546	546	546	546
8. Rat99	1286	1249	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238	1238
9. KroA100	22054	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470	21470
10. KroA150	30186	28049	27722	27243	27243	27243	27243	27243	27243	27243	27243	27243	27243	27243	27243	27243	27243
11. Krob150	28144	27068	26893	26532	26532	26532	26532	26532	26532	26532	26532	26532	26532	26532	26532	26532	26532
12. d198	16894	16436	16031	16005	15929	15926	15902	15902	15902	15902	15902	15902	15902	15902	15902	15902	15902
13. Tsp225	4394	4296	4206	4088	4065	4066	3992	3976	3976	3976	3976	3976	3976	3976	3976	3976	3976
14. Pcb442	56772	55581	54919	53728	53563	53202	52159	51973	51811	51811	51811	51811	51811	51811	51811	51811	51811

Nota. La escala de color muestra el número de ejecuciones mínima para lograr la convergencia del algoritmo a un óptimo local.

Donde todas las instancias fueron calculadas con distancia euclidiana y obtenidos de Pérez (2011)

c. ***Determinación de los Vectores Prohibición para utilizar.*** Al presentar una amplitud cuyo valor es 5, se tuvo en total 2^5 vectores prohibición (ver Figura 77, Anexo 22), de los cuales, por medio del análisis se determinó un total de 16 vectores prohibición (ver Anexo 22), los cuales son detallados a continuación: (1 0 0 0 0) (1 1 0 0 0) (1 0 1 0 0) (1 1 1 0 0) (1 1 1 1 0) (1 0 0 1 0) (1 0 1 1 0) (1 1 0 1 0) (1,0,1,1,1) (1 1 0 1 1) (1 1 1 0 1) (1 0 0 0 1)(1 1 0 0 1) (1 0 0 1 1)(1 0 1 0 1)(1 1 1 1 1).

3.17. Complejidad algorítmica de las versiones Heavy

Según la definición indicada en el numeral "2.2.21 Eficiencia algorítmica medida en base a su complejidad", el término *Eficiencia* hace referencia al uso de recursos computacionales necesarios para resolver un problema algorítmico determinado. Para nuestro caso, el recurso analizado consistió en el *tiempo,* expresado en número de operaciones elementales necesario para que el algoritmo finalice brindándonos una solución.

Al realizar un análisis para determinar una medida teórica o a priori, se utilizó la notación Big O; de tal manera que, con base en un análisis del código del algoritmo implementado en NetBeans con lenguaje de programación Java (para mayor detalle, ver Anexo 10, Anexo 11, Anexo 12 y Anexo 13), se determinó una cota superior según análisis de complejidad. Resultado de dicho análisis de complejidad, para las tres versiones Heavy desarrolladas en la presente investigación, indicadas en el numeral "3.16. Algoritmos desarrollados en su versión Heavy", se tuvo que el:

- SCA(1)_Multi2Opt, presentó una complejidad de $O(n^4\log(n))$, para mayor detalle, ver Anexo 10 y Anexo 11.

- SCA(1)_Multi2Opt_r, presentó una complejidad $O(n^4\log(n))$, para mayor detalle, ver Anexo 10 y Anexo 11.

- SCA(5)_Multi2Opt, presentó una complejidad $O(n^4 log(n))$, para mayor detalle, ver Anexo 12 y Anexo 13.

Según la Tabla 3, la complejidad $O(n^4)$, o complejidades mayores, no son lo más eficiente posible. Ayala de Vega et al. (2019) indican que hasta una complejidad $O(n^3)$ es aceptable.

Al realizar distintas ejecuciones de las versiones Heavy para distintos tamaños de instancias de TSPLIB, se encontró que, para una cantidad mediana de nodos (hasta alrededor de 500 nodos), los algoritmos SCA(1)_Multi2Opt y el SCA(1)_Multi2Opt_r (notar que la versión con factor "r" resulta de aplicar tantas veces como sea necesario el algoritmo para distintos valores de r) presentan soluciones en un tiempo aceptable (pocos minutos). En cambio, para el SCA(5)_Multi2Opt la cantidad de nodos recomendada es de 250 nodos (con una determinación de una solución en un tiempo aún aceptable, en nuestro computador, menos de 30 minutos); una mayor cantidad de nodos para esta versión de múltiples sacrificios cortoplacistas implica un uso excesivo de los recursos computacionales del computador (mayor a una hora para nuestro caso).

3.18. Algoritmos desarrollados en su versión Soft

Con base en las tres versiones desarrolladas e indicadas en el numeral 3.16, las cuales hemos denominado versiones Heavy por presentar una complejidad de algoritmos de orden $O(n^4 log(n))$ (ver Anexos 10, 11, 12 y 13), en la búsqueda de reducir su complejidad se desarrollaron las versiones Soft, cuya complejidad fue de $O(n^3 log(n))$ (ver Anexos 14, 15, 16 y 17).

La principal diferencia entre las versiones Heavy y Soft consiste en la aplicación del criterio Multi2Opt (búsqueda basada en una población) y el criterio Uni2Opt (búsqueda basada en una trayectoria), respectivamente, con lo cual se logró la reducción del orden de complejidad.

3.18.1. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Uni2Opt: SCA(1)_Uni2Opt

Algoritmo metaheurístico, el cual solo prohíbe un arco por cada apalancamiento (ver Figura 23), para luego elegir la mejor ruta del conjunto de apalancamientos y proceder a mejorar solo a esa mejor ruta con el criterio Uni2Opt (ver Figura 25). En esencia, comparado con el SCA(1)_Multi2Opt, lo único que varía es el criterio Uni2Opt, el cual solo mejora una única solución (tipo de búsqueda en la trayectoria y no una basada en la población). A continuación, se describen los pasos a seguir para su aplicación:

Paso 1: Se define el Número de ejecuciones = (Número de nodos del grafo) × 6; para mayor detalle, ver Anexo 19, e ir al Paso 2. Si se desea, el usuario puede modificar este parámetro según su percepción.

Paso 2: Obtener la ruta construida con el pensamiento de la heurística del Vecino más Cercano.

Paso 3: Definir arco prohibido para apalancamiento “t” / $1 \leq t \leq n-6$.

Paso 4: Asignar “0” como peso del arco prohibido, durante la construcción de la ruta para apalancamiento “t” / $1 \leq t \leq n-6$.

Paso 5: Construir ruta con el criterio del Vecino más Cercano y almacenar costo asociado: Ruta “t” / $1 \leq t \leq n-6$.

Paso 6: Si aún quedan más valores de “t” sin tomar entonces ir al Paso 3 y elegir otro valor de “t” aún no tomado, en el caso de no tener más valores “t” sin tomar entonces ir al paso 7.

Paso 7: Elegir la mejor ruta generada a partir de los distintos puntos de apalancamiento = $\text{Min}\{\text{Ruta } t\}$, $\forall\, t / 1 \leq t \leq n-6$

Paso 8: Aplicar el criterio Uni2Opt para mejorar ruta del paso 7.

Paso 9: Cumplido el criterio de parada del Uni2Opt, se obtiene la ruta final con su respectivo costo asociado.

Para entender mejor el funcionamiento de este algoritmo, se planteó el diagrama de bloques del SCA(1)_Uni2Opt en la Figura 49; a partir de la cual, al desglosarlo, se describió su funcionamiento en la Figura 50 y su respectivo diagrama de flujo en la Figura 51.

Figura 49

Diagrama de bloques del SCA(1)_Uni2Opt

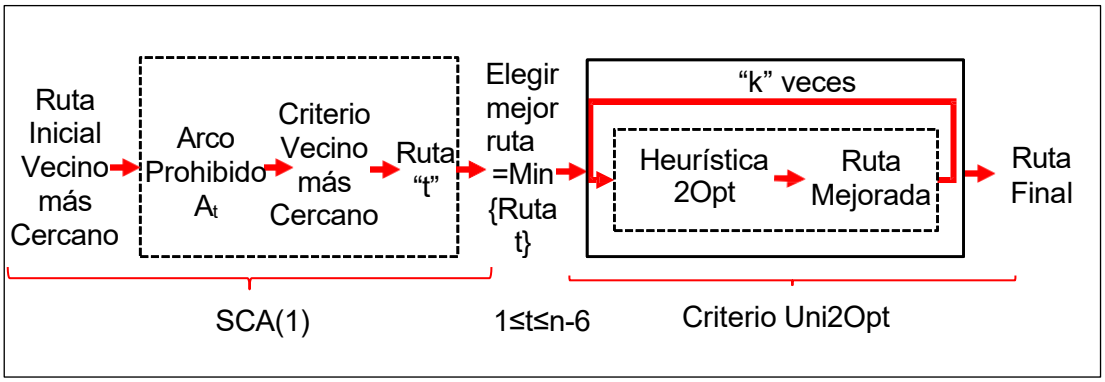
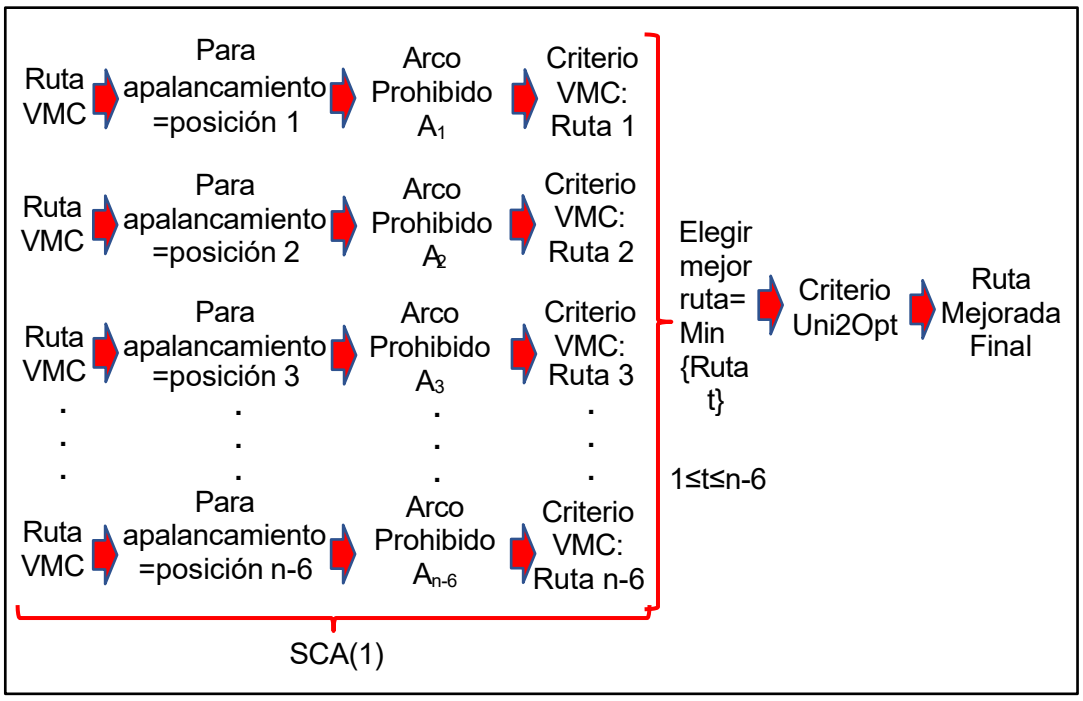


Figura 50

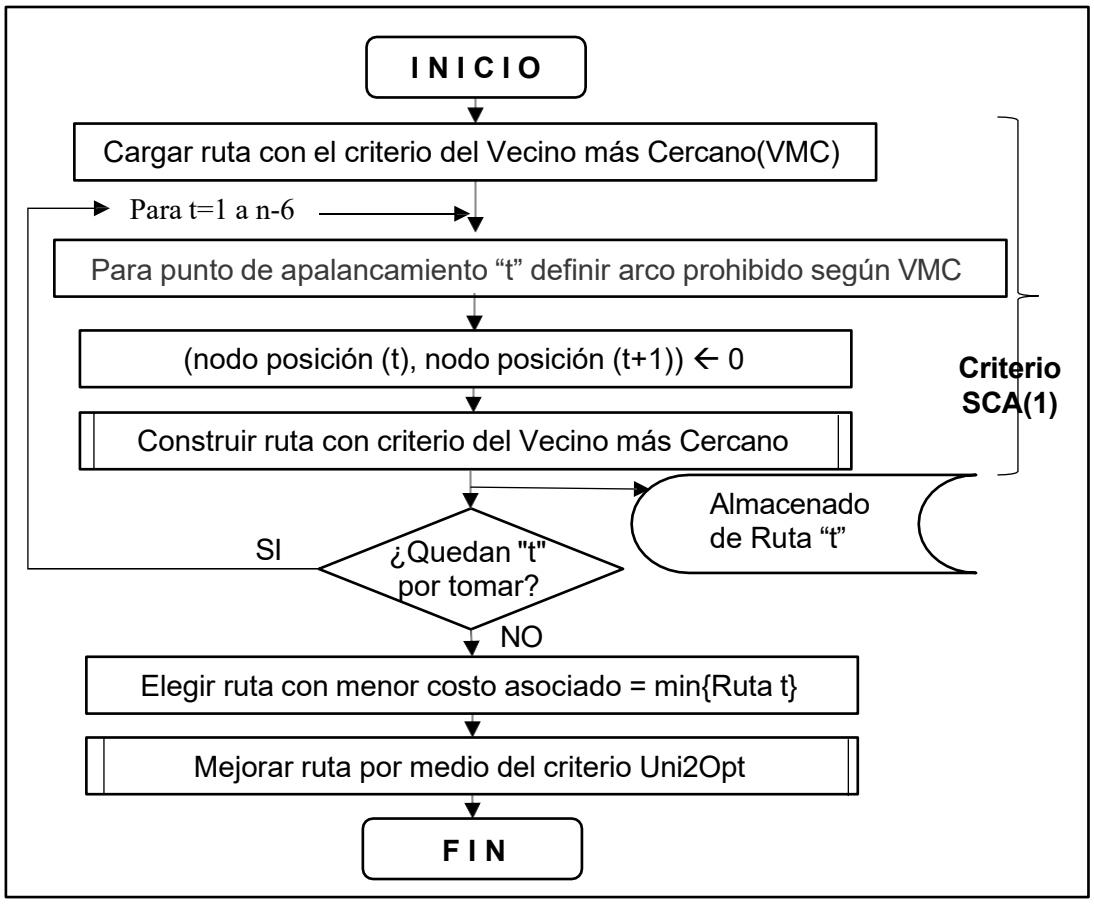
Funcionamiento del SCA(1)_Uni2Opt



En la Figura 50 se observa que, por medio del criterio Uni2Opt, se buscó mejorar a la mejor ruta determinada por los distintos apalancamientos. De esta forma, la búsqueda se transforma en una basada en la trayectoria y así reducimos su complejidad. Para mayor detalle, se muestra el diagrama de flujo del algoritmo SCA(1)_Uni2Opt en la Figura 51.

Figura 51

Diagrama de flujo del SCA(1)_Uni2Opt



a. Criterio de parada para el algoritmo SCA(1)_Uni2Opt.

Se decidió tomar el mismo criterio de parada que para la versión SCA(1)_Multi2Opt, debido a que el SCA(1)_Uni2Opt es un caso particular del SCA(1)_Multi2Opt.

3.18.2. Sacrificio Cortoplacista Adaptativo de amplitud 1, con criterio Uni2Opt_r: SCA(1)_Uni2Opt_r

Algoritmo metaheurístico, el cual solo prohíbe un arco por cada apalancamiento (ver Figura 23), para luego elegir la mejor ruta del conjunto de apalancamientos y proceder a mejorarla con el criterio Uni2Opt_r (ver Figura 27). En esencia, comparado con el SCA(1)_Multi2Opt, lo único que varía es el criterio Multi2Opt por el Uni2Opt_r, el cual solo mejora una única solución (tipo de búsqueda en trayectoria y no una poblacional). A continuación, se describen los pasos a seguir para la aplicación del SCA(1)_Uni2Opt_r:

Paso 1: Se define el Número de ejecuciones = (Número de nodos del grafo) × 6; e ir al Paso 2 (para mayor detalle, ver Anexo 19). Si desea el usuario puede modificar este valor según su percepción.

Paso 2: Obtener la ruta construida con el pensamiento de la heurística del Vecino más Cercano.

Paso 3: Definir el arco prohibido para apalancamiento “t” / $1 \leq t \leq n-6$.

Paso 4: Asignar “0” como peso del arco prohibido, durante la construcción de la ruta para apalancamiento “t” / $1 \leq t \leq n-6$.

Paso 5: Construir una ruta con el criterio del Vecino más Cercano y almacenar costo asociado: Ruta “t” / $1 \leq t \leq n-6$.

Paso 6: Ir al Paso 3 y elegir otro valor de “t” aún no tomado, en caso no tener más valores “t” a recorrer ir al paso 7.

Paso 7: Elegir la mejor ruta generada a partir de los distintos puntos de apalancamiento = $\text{Min}\{\text{Ruta t}\}, \forall\, t / 1 \leq t \leq n-6$

Paso 8: Aplicar el criterio Uni2Opt_r para mejorar ruta del paso 7.

Paso 9: Cumplido el criterio de parada del Uni2Opt_r, se obtiene la ruta final con su respectivo costo asociado.

Para entender mejor el funcionamiento de este algoritmo, se planteó el diagrama de bloques del SCA(1)_Uni2Opt_r en la Figura 52, la descripción del funcionamiento del SCA(1)_Uni2Opt_r en la Figura 53 y el respectivo diagrama de flujo en la Figura 54.

Figura 52

Diagrama de bloques del SCA(1)_Uni2Opt_r

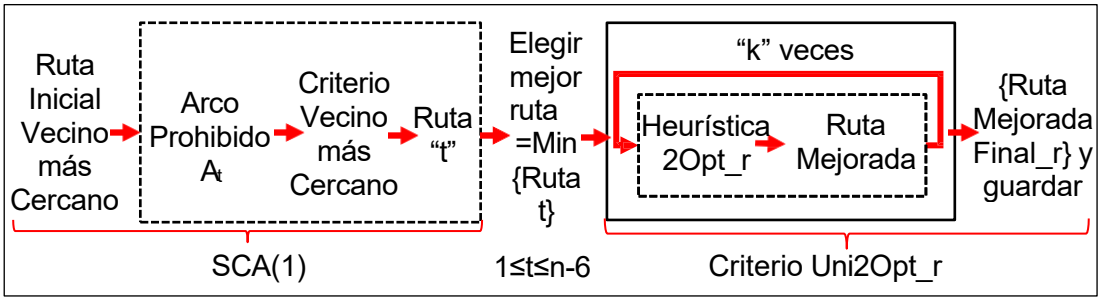
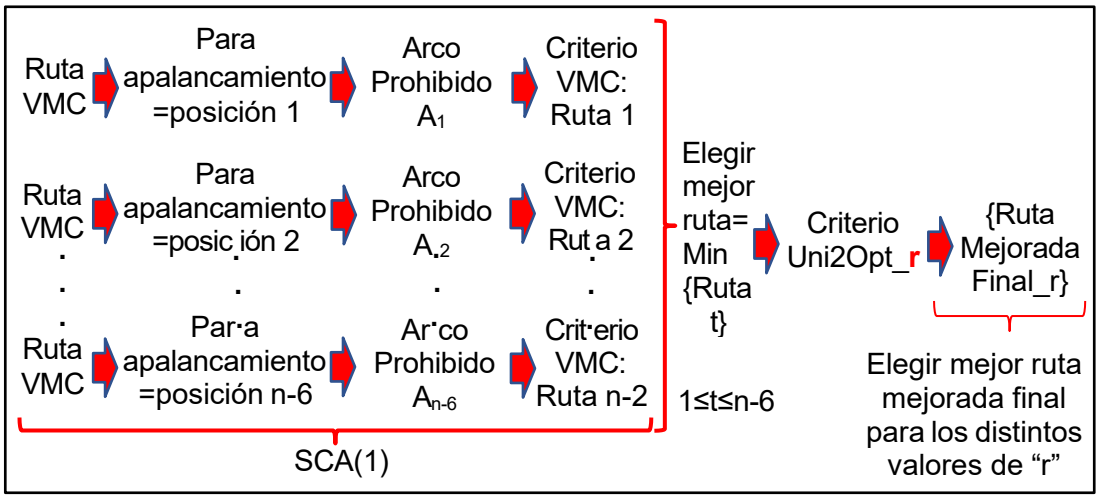


Figura 53

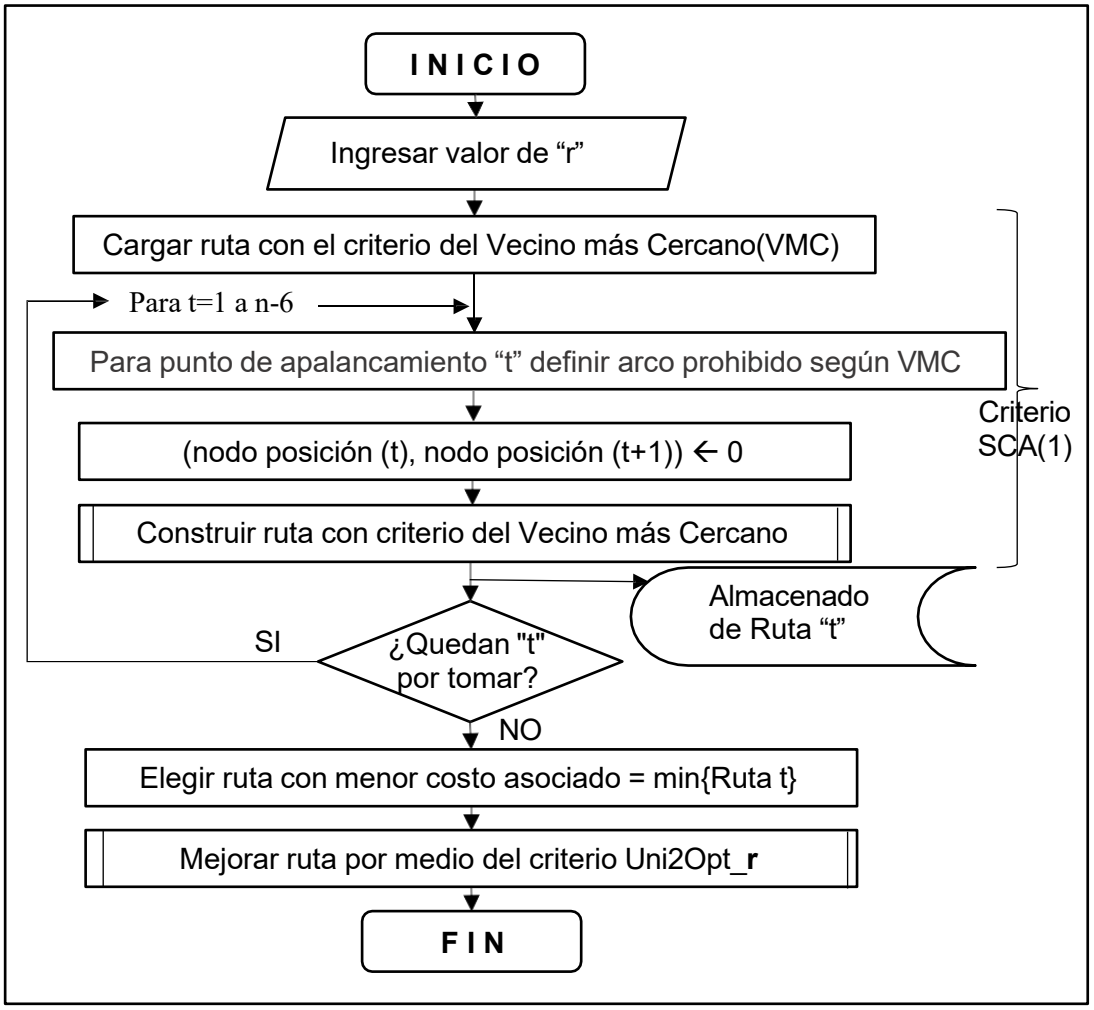
Funcionamiento del SCA(1)_Uni2Opt_r



De la Figura 53, el criterio Uni2Opt_r buscó mejorar solo la mejor ruta determinada por los distintos apalancamientos. De esta forma la búsqueda se transforma en una basada en la trayectoria y así reducimos su complejidad. Para mayor detalle, se muestra su diagrama de flujo en la Figura 54.

Figura 54

Diagrama de flujo del SCA(1)_Uni2Opt_r



a. ***Determinación de rango de valores para el factor “r”.*** Para el algoritmo SCA(1)_Uni2Opt_r, se decidió utilizar el mismo rango de valores del factor “r” definidos para el algoritmo SCA(1)_Multi2Opt_r, es decir, “r” ∈ [-0.3 a 0.3]. Debido a que el SCA(1)_Uni2Opt_r es un caso particular del SCA(1)_Multi2Opt_r.

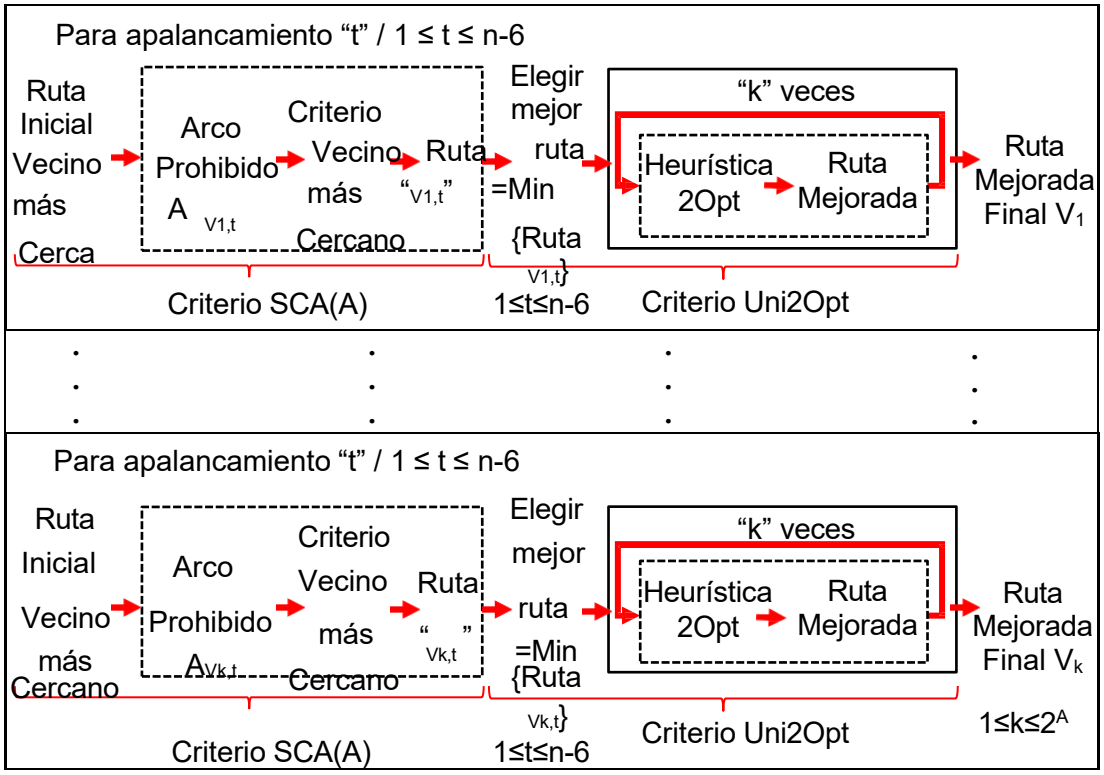
b. ***Criterio de parada para el algoritmo SCA(1)_Uni2Opt_r.*** Se decidió tomar el mismo criterio de parada que para su versión SCA(1)_Multi2Opt_r, debido a que el SCA(1)_Uni2Opt_r es un caso particular del SCA(1)_Multi2Opt_r.

3.18.3. Sacrificio Cortoplacista Adaptativo de amplitud A, con criterio Uni2Opt: algoritmo SCA(A)_Uni2Opt

Se define como un algoritmo metaheurístico con un proceso de búsqueda tipo poblacional que consiste en realizar un máximo de "k = 2^A" búsquedas independientes ("A" es la amplitud), cada una guiada por un único vector prohibición (V_k). Es equivalente al SCA(A)_Multi2Opt, con la diferencia de en vez de usar el criterio Multi2Opt se usó el criterio Uni2Opt. El diagrama de bloques es mostrado en la Figura 55.

Figura 55

Diagrama de bloques del SCA(A)_Uni2Opt

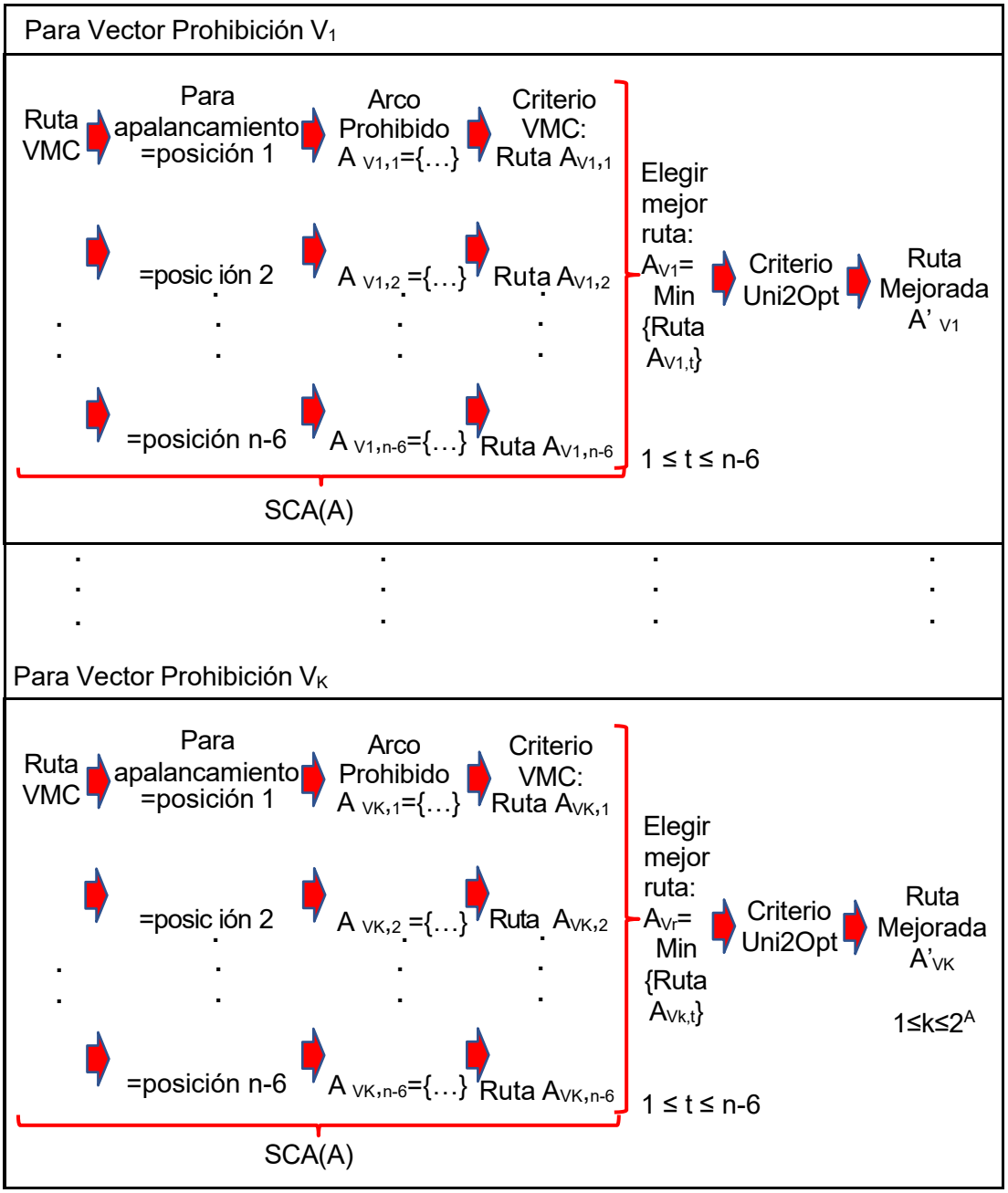


Nota. Se elige la mejor ruta de {Ruta Mejorada Final V_1, …, Ruta Mejorada Final V_k}

Partiendo de la Figura 55, al desglosar el diagrama de bloques del SCA(A)_Uni2Opt, se muestra la Figura 56 para un mejor entendimiento del proceso de búsqueda.

Figura 56

Funcionamiento del SCA(A)_Uni2Opt



Nota. Elegir ruta final asociada al Min{d(Ruta A'_{V1}), d(Ruta A'_{V2}), …, d(Ruta A'_{Vk})}

En la Figura 56 se observa lo siguiente:

- La cantidad de vectores prohibición "V_k" depende de la amplitud, quedando determinada por la siguiente fórmula:

  $k = 2^A$, donde A = Amplitud… (Ecuación 2)

- $A_{Vk,j}$: Arcos prohibidos según vector prohibición "V_k" y punto de apalancamiento "j".

  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, \quad 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo

- Ruta $A_{Vk,j}$: Ruta construida por la aplicación del criterio del Vecino más Cercano, con arcos prohibidos según vector prohibición "V_i" y punto de apalancamiento "j".

  Donde: $j, k \in Z^+ / 1 \leq j \leq n-6, \quad 1 \leq k \leq 2^{Amplitud}$, n: orden del grafo

- Ruta "V_r": Mejor ruta de las generadas por el vector prohibición $\{A_{Vr,j}\}$, para $1 \leq r \leq 2^{Amplitud}$, $1 \leq j \leq n-6$, donde, la existencia del vector prohibición permite que para esta sección se realice una búsqueda tipo poblacional, es decir se generan "k" rutas.

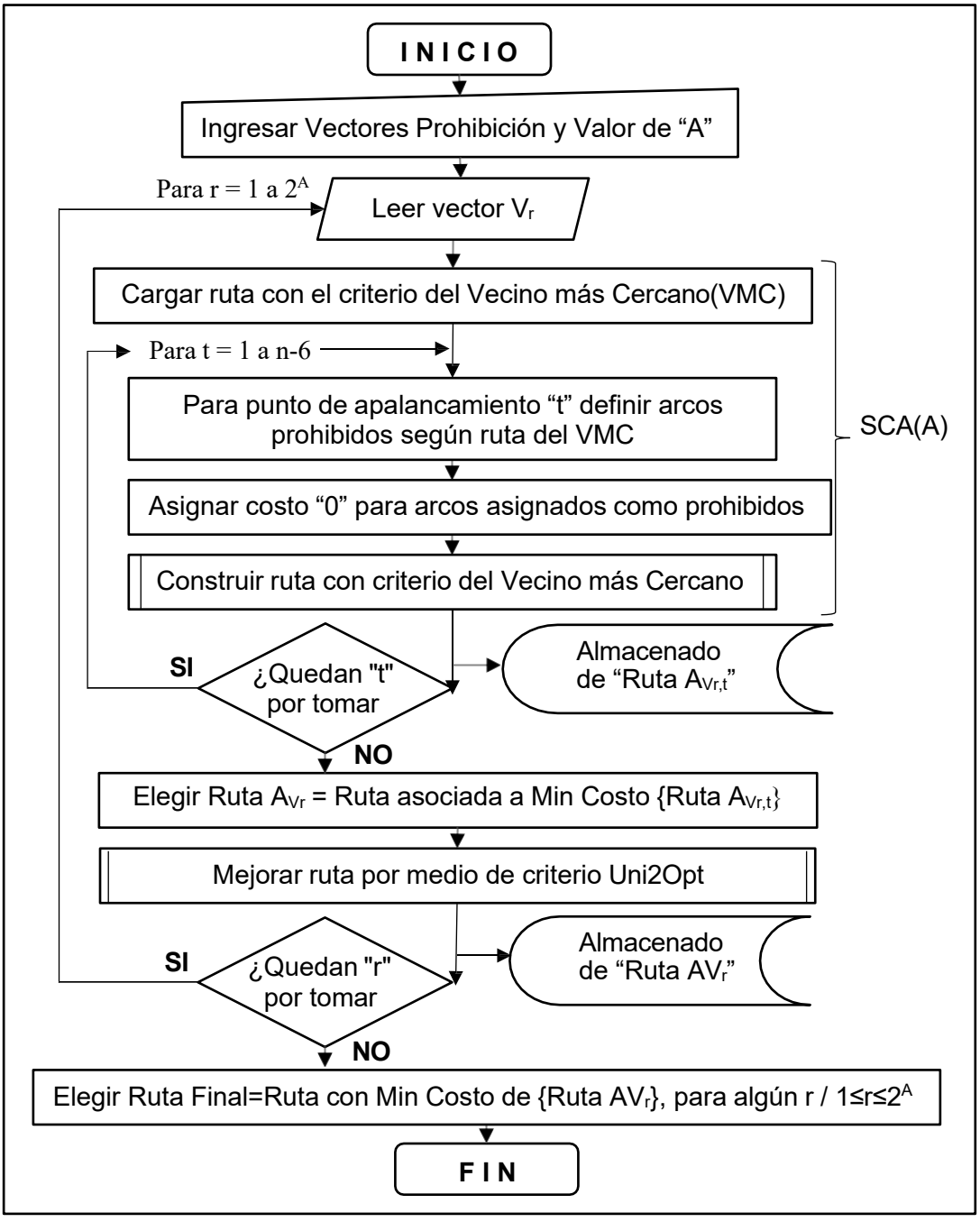
- Ruta A'_{Vr}: Ruta mejorada al realizar la aplicación de heurística 2Opt sobre "Ruta A_{Vr}", para algún $r \in Z^+ / 1 \leq r \leq k$ (en esta parte las "k" rutas generadas pasan a ser mejoradas) y $1 \leq k \leq 2^{Amplitud}$.

- Ruta V_f = Ruta asociada al **Min {**d(Ruta $A_{V'1}$); d(Ruta $A_{V'2}$); … ; d(Ruta A_{Vk})**}**

A continuación, para un mayor entendimiento del funcionamiento del SCA(A)_Multi2Opt, se presenta el diagrama de flujo en la Figura 57.

Figura 57

Diagrama de flujo del SCA(A)_Uni2Opt



a. ***Determinación del parámetro "A-Amplitud".*** Se utilizó el mismo valor de la "Amplitud" indicado para la versión Heavy SCA(A)_Multi2Opt, esto debido a que el SCA(5)_Uni2Opt es un caso particular del SCA(5)_Multi2Opt. Es decir, prohibir 5 arcos (A=5).

b. ***Criterio de parada para el algoritmo SCA(5)_Uni2Opt.*** Se utilizó el mismo criterio de parada usado en el algoritmo SCA(5)_Multi2Opt, esto debido a que el SCA(5)_Uni2Opt es un caso particular del SCA(5)_Multi2Opt. Es decir:

Número de Ejecuciones = Orden del grafo × 6

c. ***Determinación de los Vectores Prohibición para utilizar.*** Al igual que el caso de su versión Heavy, al presentar una amplitud de 5, se tuvo en total 32 vectores prohibición (ver Figura 77, Anexo 22). De los cuales, por medio del análisis, se eligieron un total de 16 vectores prohibición (ver Anexo 22). Los cuales se especifican a continuación:

- Vector Prohibición 1: (1 0 0 0 0)
- Vector Prohibición 2: (1 1 0 0 0)
- Vector Prohibición 3: (1 0 1 0 0)
- Vector Prohibición 4: (1 1 1 0 0)
- Vector Prohibición 5: (1 1 1 1 0)
- Vector Prohibición 6: (1 0 0 1 0)
- Vector Prohibición 7: (1 0 1 1 0)
- Vector Prohibición 8: (1 1 0 1 0)
- Vector Prohibición 9: (1,0,1,1,1)
- Vector Prohibición 10: (1 1 0 1 1)
- Vector Prohibición 11: (1 1 1 0 1)
- Vector Prohibición 12: (1 0 0 0 1)
- Vector Prohibición 13: (1 1 0 0 1)
- Vector Prohibición 14: (1 0 0 1 1)
- Vector Prohibición 15: (1 0 1 0 1)
- Vector Prohibición 16: (1 1 1 1 1)

3.19. Complejidad algorítmica de las versiones Soft

Según la definición indicada en el numeral “2.2.21 Eficiencia algorítmica medida en base a su complejidad”, el término *Eficiencia* hace referencia al uso de recursos computacionales necesarios para resolver un problema algorítmico determinado. Para nuestro caso, el recurso analizado consistió en el *tiempo*, expresado en número de operaciones elementales necesario para que el algoritmo finalice brindando una solución.

Se realizó un análisis para determinar una medida teórica o a priori de la complejidad algorítmica. Por ello, se utilizó la notación Big O, de tal manera que, con base en un análisis del código del algoritmo implementado en NetBeans con lenguaje de programación Java (para mayor detalle, ver Anexo 14, Anexo 15, Anexo 16 y Anexo 17), se determinó una cota superior según el análisis de complejidad. Como resultado del análisis de complejidad, para las tres versiones Soft desarrolladas en la presente investigación, indicadas en el numeral “3.17. Algoritmos desarrollados en su versión Soft”, se tuvo que el:

- SCA(1)_Uni2Opt, presentó una complejidad de $O(n^3\log(n))$, para mayor detalle ver Anexo 14 y Anexo 15.
- SCA(1)_Uni2Opt_r, presentó una complejidad $O(n^3\log(n))$, para mayor detalle ver Anexo 14 y Anexo 15.
- SCA(5)_Uni2Opt, presentó una complejidad $O(n^3\log(n))$, para mayor detalle ver Anexo 16 y Anexo 17.

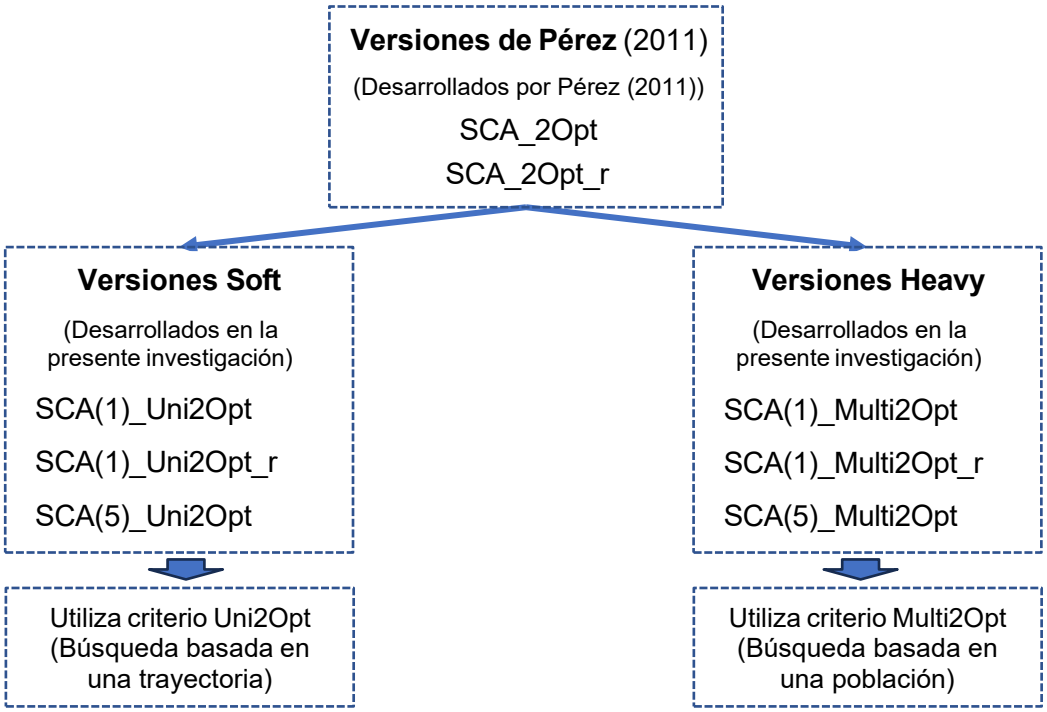
CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

Durante la etapa de construcción del algoritmo, se desarrollaron dos grupos de algoritmos, a los cuales hemos denominado Heavy y Soft. Dichas versiones surgieron con base en los algoritmos desarrollados por Pérez (2011): SCA_2Opt y SCA_2Opt_r, según lo indicado en la Figura 58.

Figura 58

Resumen de los algoritmos desarrollados



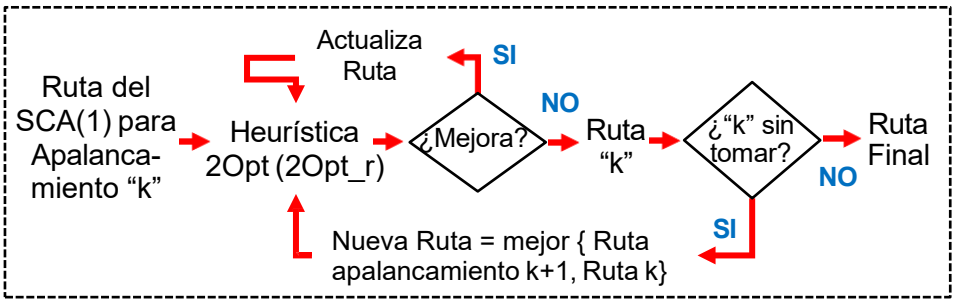
Blum y Rolo (2003, como se citó en Vélez y Montoya, 2007) explican que las metaheurísticas buscan evitar quedar atrapadas en óptimos locales, a diferencia de las heurísticas. Según dicha afirmación, tanto las versiones Heavy como Soft entrarían en la clasificación de metaheurísticas debido al factor de perturbación “r”, el cual busca prevenir dicho inconveniente.

A continuación, como se muestra en la Figura 58, se clasificaron para los algoritmos indicados de la siguiente manera:

- **Algoritmos de Pérez:** Versiones desarrolladas por Pérez (2011), basadas en la realización de un único Sacrificio Cortoplacista Adaptativo y en el mejoramiento de la solución por medio de la heurística 2Opt (o 2Opt_r), la cual se realiza a la ruta actualizada en cada iteración. Para describir mejor el funcionamiento de este grupo de algoritmos, se presenta la Figura 59.

Figura 59

Síntesis del flujograma del SCA_2Opt(_r) de Pérez(2011)

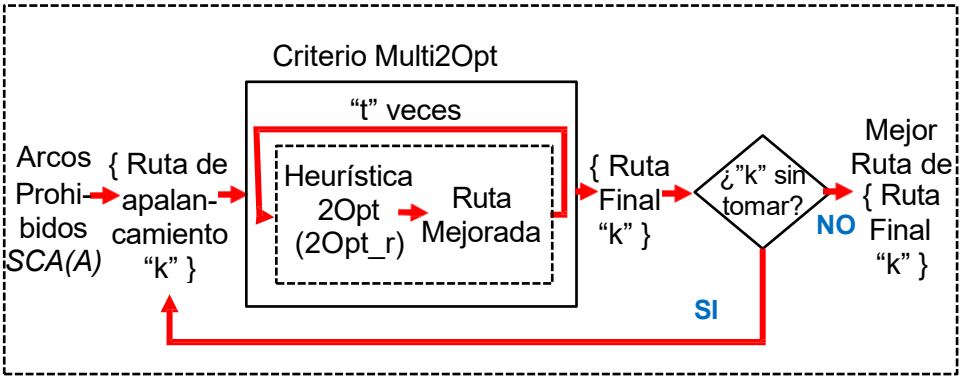


Nota. Los algoritmos de Pérez (2011) corresponden a su investigación en 2011

- **Versiones Heavy:** Versiones desarrolladas en la presente investigación, a partir de los algoritmos de Pérez (2011), los cuales aplican la heurística 2Opt (o 2Opt_r), con la política Best Improvement, por cada punto de apalancamiento seleccionado. Es decir, el criterio Multi2Opt presenta un proceso de búsqueda basado en la población. Para describir mejor el funcionamiento de este grupo de algoritmos, se presenta la Figura 60.

Figura 60

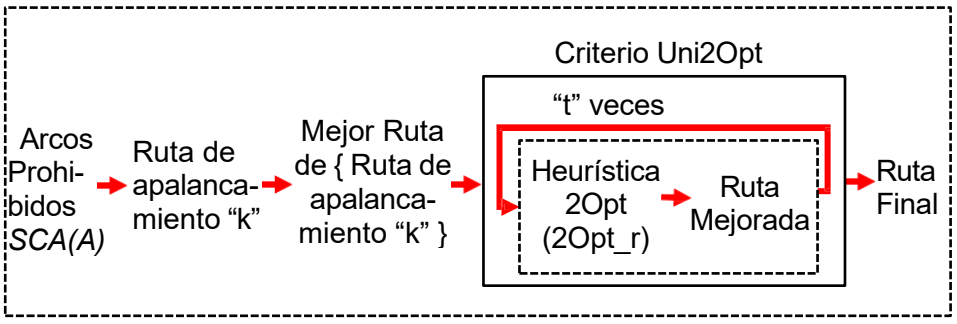
Síntesis del flujograma de las versiones Heavy



- **Versiones Soft:** Versiones desarrolladas en la presente investigación, a partir de los algoritmos de Pérez (2011), los cuales aplican la heurística 2Opt (o 2Opt_r), con la política Best Improvement, únicamente a la mejor solución asociada a un determinado punto de apalancamiento, por lo cual recibe el nombre de Uni2Opt. Lo que hace es convertir una búsqueda tipo poblacional a una basada en la trayectoria. Para describir mejor el funcionamiento de este grupo de algoritmos, se presenta la Figura 61.

Figura 61

Síntesis del flujograma de las versiones Soft



4.1. Presentación de Resultados

Antes de presentar los resultados, se hace hincapié en los indicadores utilizados en la presente investigación: *Eficacia* y *Nivel de Eficacia*.

Para determinar la ***Eficacia*** de las soluciones obtenidas por nuestros algoritmos, se usó el *Error Relativo* como primer indicador, el cual se definió de la siguiente manera:

$$Error\ Relativo = \frac{Solución\ alcanzada\ por\ el\ algoritmo\ evaluado - Solución\ óptima\ TSPLIB}{Solución\ óptima\ TSPLIB}$$

A partir del indicador ***Eficacia***, se calculó el ***Nivel de Eficacia***, el cual se define como la proporción de instancias donde la eficacia de cada algoritmo fue mejor que la del resto de algoritmos con los cuales se comparó:

$$Nivel\ de\ Eficacia = \frac{Cantidad\ de\ instancias\ donde\ el\ algoritmo\ evaluado\ obtuvo\ la\ mejor\ Eficacia}{Cantidad\ total\ de\ instancias\ seleccionadas}$$

A continuación, se muestra una categorización del *Nivel de Eficacia* en la Tabla 7.

Tabla 7

Categorización del nivel de eficacia

Rango Inferior	Rango Superior	Categoría
[0%	20%>	Muy Bajo
[20%	40%>	Bajo
[40%	60%>	Medio
[60%	80%>	Alto
[80%	100%]	Muy Alto

4.1.1. Selección de instancias de librería TSP para medir la Eficacia

Para comparar el *Error Relativo* de las soluciones obtenidas por los nuevos algoritmos desarrollados en la presente investigación, se realizó una selección de instancias simétricas de la librería virtual TSPLIB, divididas en dos grupos, los cuales se detallan a continuación:

a. ***Para comparar con los algoritmos SCA_2Opt y SCA_2Opt_r.*** Para comparar el *Error Relativo* de las soluciones obtenidas por las versiones Heavy y Soft y los algoritmos desarrollados por Pérez (2011): SCA_2Opt y el SCA_2Opt_r. Para lo cual, el criterio de selección consistió en elegir instancias del agente viajero simétrico recopiladas de la librería virtual TSPLIB, seleccionadas en la investigación de Pérez (2011) y que estén acordes al método de cálculo de distancia indicado en TSPLIB, para así poder realizar la medición del *Error Relativo* correcto. De las 14 instancias revisadas en la investigación de Pérez (2011), se encontraron 13 instancias acordes con el método de cálculo de distancia para la matriz de costos indicado por la librería virtual TSPLIB, las cuales son indicadas a continuación:

   Instancias: Swiss42, Eil50, Eil51, St70, Eil75, Eil76, Rat99, KroA100, KroA150, Krob150, d198, Tsp225 y Pcb442.

b. ***Para comparar con algoritmos metaheurísticos seleccionados.*** De las soluciones obtenidas por los algoritmos metaheurísticos recopilados en la investigación de Pérez (2011), donde, al presentar 29 algoritmos metaheurísticos (agrupados en cuatro bloques o grupos), se definieron las siguientes tres etapas para escoger aquellas metaheurísticas con las mejores soluciones obtenidas dentro de la investigación de Pérez (2011):

- ***Etapa 1***. Selección de aquellas metaheurísticas cuyas soluciones hayan logrado superar a la mejor solución entre el SCA_2Opt y el SCA_2Opt_r en todas las instancias comparadas y cuya publicación original (fuente primaria) se encuentre disponible en acceso público (Ver Anexo 3).

- ***Etapa 2***. De las metaheurísticas seleccionadas en la etapa 1, ir a la publicación original y extraer todas las instancias TSPLIB que correspondan al Problema del Agente Viajero Simétrico.

- ***Etapa 3***. Validación de que el método de cálculo de la matriz de costos de las instancias simétricas seleccionadas sea congruente con el método de cálculo indicado en la librería virtual TSPLIB.

Notar que la etapa 2 se realizó con el objetivo de recopilar instancias no tomadas en la investigación desarrollada por Pérez (2011) y así enriquecer la casuística analizada.

Al aplicar las etapas 1, 2 y 3, de los cuatro grupos de algoritmos metaheurísticos recopilados en la investigación de Pérez (2011), quedó un total de tres grupos de metaheurísticas para comparar con los algoritmos desarrollados en la presente investigación (ver Anexos 3 al 7), los cuales son:

- Grupo de Algoritmos Genéticos

- Grupo de Colonia de Hormigas

- Grupo de Otras metaheurísticas: Programación Evolutiva.

Por lo cual, las instancias seleccionadas del Problema del Agente Viajero Simétrico, ubicadas dentro de la librería virtual TSLIB según las metaheurísticas seleccionadas, resultado de la aplicación de los etapas 1, 2 y 3, son mostradas a continuación:

- **Para el grupo de Algoritmos Genéticos (para mayor detalle ver Anexo 4):**
  - ***Instancias Algoritmo Genético (AG):*** KroA100, Eil50, Eil75.
  - ***Instancias Fragment AG (FRAG_GA), Swapping GA for TSP (SWAP_GATSP) y el Improved GA(IGA):*** KroA100, d198, Pcb442, Rat783, Ts225.
- **Para el Grupo de Colonia de Hormigas (para mayor detalle ver Anexo 5):**
  - ***Instancias Algoritmo del System Colony Ant (ACS):*** Eil50, Eil75, KroA100, d198, Pcb442, att532, Rat783.
  - ***Instancias Algoritmo del System Colony Ant mejorado con 3Opt (ACS_3Opt):*** d198, lin318, Att532, Rat783.
- **Para el Grupo de Otras Metaheurísticas (para mayor detalle ver Anexo 7):**
  - ***Instancias Programación Evolutiva (PE o EP):*** Eil50, Eil75.

4.1.2. Eficacia y Nivel de Eficacia de las versiones sin factor "r"

En base a la Figura 58, se realizó una comparación del *Error Relativo* entre las soluciones obtenidas por los algoritmos sin el factor "r". Los resultados fueron consolidados en la Tabla 8.

- Versión Soft: SCA(1)_Uni2Opt,
- Versión Heavy: SCA(1)_Multi2Opt
- Versión Original de Pérez(2011): SCA_2Opt.

Tabla 8

Solución y error relativo de algoritmos SCA_2Opt de Pérez (2011), SCA(1)_Uni2Opt y SCA(1)_Multi2Opt según instancia TSPLIB

Instancia TSPLIB	Solución TSPLIB (*) (A)	Solución según instancia y algoritmo utilizado			Error relativo según instancia y algoritmo utilizado		
		SCA_ 2Opt (**) (B)	SCA(1)_ Uni2Opt (C)	SCA(1)_ Multi2Opt (D)	SCA_2Opt [(B-A)/A]	SCA(1)_Uni2Opt [(C-A)/A]	SCA(1)_Multi2Opt [(D-A)/A]
1.Swiss42	1273	1375	1273	1273	8.0%	0.0%	0.0%
2.Eil50	425	448.83	436	429	5.6%	2.6%	0.9%
3.Eil51	426	449.85	437	430	5.6%	2.6%	0.9%
4.St70	675	713.98	732	688	5.8%	8.4%	1.9%
5.Eil75	535	588.61	563	547	10.0%	5.2%	2.2%
6.Eil76	538	567.01	567	548	5.392%	5.390%	1.9%
7.Rat99	1211	1268.95	1301	1258	4.8%	7.4%	3.9%
8.KroA100	21282	22379.4	23314	21474	5.2%	9.5%	0.9%
9.KroA150	26524	29076.9	28139	27493	9.6%	6.1%	3.7%
10.Krob150	26130	28136.3	26651	26651	7.7%	2.0%	2.0%
11.d198	15780	16212	16260	15926	2.7%	3.0%	0.9%
12.Tsp225	3916	4071	4040	3976	4.0%	3.1%	1.5%
13.Pcb442	50778	55133.8	53223	51977	8.6%	4.8%	2.4%

Nota. (*) Solución óptima proporcionada por la librería virtual TSPLIB. (**) Solución proporcionada por el algoritmo original desarrollado por Pérez (2011). Se calculó el error relativo de los algoritmos con base en las 13 instancias seleccionadas de TSPLIB indicadas en el inciso “a” del numeral 4.1.1.

De la Tabla 8, para las trece instancias seleccionadas, el *Error Relativo* (Eficacia) de las soluciones según el algoritmo utilizado varió de la siguiente manera:

- El SCA(1)_Multi2Opt con un *Error Relativo* comprendido dentro del intervalo [0%; 3.9%], mientras que el SCA_2Opt de Pérez (2011) varió en [2.7% a 10%]. En este caso, sí se puede afirmar mejoría por parte de nuestro algoritmo en comparación al algoritmo de este autor.

- El SCA(1)_Uni2Opt con un *Error Relativo* comprendido dentro del intervalo [0%; 9.5%], mientras que el SCA_2Opt de Pérez (2011) varió en [2.7% a 10%]. En este caso, sí se puede afirmar mejoría por parte de nuestro algoritmo en comparación al algoritmo de Pérez.

Con respecto al *Nivel de Eficacia* para las trece instancias revisadas:

- Del SCA(1)_Multi2Opt comparado con el SCA_2Opt fue de 100% [(13/13) x 100%]. Es decir, las soluciones del SCA(1)_Multi2Opt, desarrollado en la presente investigación, resultaron mejores que las soluciones determinadas por el SCA_2Opt, desarrollado por Pérez (2011), en todas las 13 instancias seleccionadas.

- Del SCA(1)_Uni2Opt comparado con el SCA_2Opt fue de 77% [(10/13) x 100%]. Es decir, las soluciones del SCA(1)_Uni2Opt, desarrollado en la presente investigación, resultaron mejores que las soluciones determinadas por el SCA_2Opt, desarrollado por Pérez (2011), en 10 de las 13 instancias seleccionadas.

Adicionalmente, se comparó el *Nivel de Eficacia* de las dos versiones desarrolladas en la Tabla 8: SCA(1)_Multi2Opt y el SCA(1)_Uni2Opt, tal que el SCA(1)_Multi2Opt presentó un *Nivel de Eficacia* del 100% [(13/13) x 100%]. Es decir, las soluciones del SCA(1)_Multi2Opt resultaron mejores que las soluciones determinadas por el SCA(1)_Uni2Opt en el total de las 13 instancias seleccionadas de la librería virtual TSPLIB.

Conclusión. Para las trece instancias revisadas:

- El SCA(1)_Multi2Opt presentó una menor variación a comparación del SCA(1)_Uni2Opt y el SCA_2Opt, este último desarrollado por Pérez (2011). Por lo cual, el SCA(1)_Multi2Opt presentó mejor *Eficacia* al presentar errores relativos más próximos al cero.

- El *Nivel de Eficacia* del SCA(1)_Uni2Opt, desarrollado en la presente investigación, resultó mayor al SCA_2Opt, desarrollado por Pérez (2011) en un 77% de las 13 instancias revisadas.

- El *Nivel de Eficacia* del SCA(1)_Multi2Opt, desarrollado en la presente investigación, resultó ser mayor que el del SCA_2Opt en un 100% de las 13 instancias revisadas, este último algoritmo desarrollado por Pérez (2011).

4.1.3. Eficacia y Nivel de Eficacia de las versiones con factor "r"

Con base en la Figura 58, se realizó una comparación del *Error Relativo* entre las soluciones obtenidas por los algoritmos con el factor "r".

- Versión Soft: SCA(1)_Uni2Opt_r

- Versión Heavy: SCA(1)_Multi2Opt_r

- Versión Original de Pérez(2011): SCA_2Opt_r

Los resultados se consolidaron en la Tabla 9.

Tabla 9

Solución y error relativo de algoritmos SCA_2Opt_r de Pérez (2011), SCA(1)_Uni2Opt_r y SCA(1)_Multi2Opt_r según instancia

Instancia TSPLIB	Solución TSPLIB (*) (A)	r	Solución según instancia y algoritmo utilizado SCA_2Opt_r (B) (**)	r	SCA(1)_Uni 2Opt_r (C)	r	SCA(1)_Multi 2Opt_r (D)	Error Relativo según instancia y algoritmo utilizado SCA_2Opt_r (B-A)/A	SCA(1)_Uni 2Opt_r [(C-A)/A]	SCA(1)_Multi 2Opt_r [(C-A)/A]
1.Swiss42	1273	0.05	1285	0.01	1273	-0.25	1273	0.9%	0.0%	0.0%
2.Eil50	425	0.1	430.48	-0.1	429	-0.01	427	1.3%	0.9%	0.5%
3.Eil51	426	0.1	436.08	-0.1	430	-0.01	429	2.4%	0.9%	0.7%
4.St70	675	0.14	690.62	-0.1	728	-0.1	686	2.3%	7.9%	1.6%
5.Eil75	535	0.16	560.14	-0.35	553	-0.2352	535	4.7%	3.4%	0.0%
6.Eil76	538	0.25	559.61	-0.3	553	0.01	543	4.0%	2.8%	0.9%
7.Rat99	1211	0.18	1259.51	-0.4	1266	-0.3	1229	4.0%	4.5%	1.5%
8.KroA100	21282	0.16	22219.3	-0.1769	22786	0.01	21474	4.4%	7.1%	0.9%
9.KroA150	26524	0.19	27999.7	-0.09	28107	-0.08	27446	5.6%	6.0%	3.5%
10.Krob150	26130	0.18	26904.48	-0.07	26591	-0.05	26439	3.0%	1.8%	1.2%
11.d198	15780	0.01	16180.53	0.001	16009	0.015	15907	2.5%	1.5%	0.8%
12.Tsp225	3916	0.01	4054.77	-0.01	4040	0.01	3955	3.5%	3.1%	1.0%
13.Pcb442	50778	0.13	53287.29	-0.05	52871	0.03	51730	4.9%	4.1%	1.9%

Nota. (*) Solución óptima proporcionada por la librería Virtual TSPLIB. (**) Solución proporcionada por el algoritmo original desarrollado por Pérez (2011). Se calculó el error relativo de los algoritmos con base en las 13 instancias seleccionadas de TSPLIB indicadas en el inciso “a” del numeral 4.1.1.

A partir de la Tabla 9, se observa que, para las trece instancias seleccionadas, el *Error Relativo* (Eficacia) de las soluciones varió según el algoritmo utilizado:

- El SCA(1)_Multi2Opt_r con un *Error Relativo* contenido dentro del intervalo [0%; 3.5%], mientras que la versión SCA_2Opt_r de Pérez (2011) varió en [0.9% a 5.6%]. En este caso, sí se puede afirmar mejoría por parte de nuestro algoritmo en comparación al algoritmo de Pérez (2011).

- El SCA(1)_Uni2Opt_r con un *Error Relativo* contenido dentro del intervalo [0%; 7.9%], mientras que la versión SCA_2Opt_r de Pérez (2011) varió en [0.9% a 5.6%]. En este caso, no se puede afirmar mejoría por parte de nuestro algoritmo en comparación al algoritmo de Pérez (2011).

Con respecto al *Nivel de Eficacia* para las trece instancias revisadas:

- Del SCA(1)_Multi2Opt_r comparado con el SCA_2Opt_r fue de 100%[(13/13)x100%]. Es decir, las soluciones del SCA(1)_Multi2Opt_r, desarrollado en la presente investigación, resultaron mejores que las soluciones determinadas por el SCA_2Opt_r, desarrollado por Pérez (2011), en todas las 13 instancias seleccionadas.

- Del SCA(1)_Uni2Opt_r comparado con el SCA_2Opt_r fue de 69% [(9/13)x100%]. Es decir, las soluciones del SCA(1)_Uni2Opt, desarrollado en la presente investigación, resultaron mejores que las soluciones determinadas por el SCA_2Opt_r, desarrollado por Pérez (2011), en 9 de las 13 instancias seleccionadas.

Adicionalmente, se comparó el *Nivel de Eficacia* de las dos versiones desarrolladas en la Tabla 9: SCA(1)_Multi2Opt_r y el SCA(1)_Uni2Opt_r, tal que el SCA(1)_Multi2Opt_r presentó un *Nivel de Eficacia* del 100% [(13/13)x100%] en comparación con el SCA(1)_Uni2Opt en las 13 instancias seleccionadas.

Conclusión. Para las trece instancias revisadas:

- El SCA(1)_Multi2Opt_r presentó una variación menor a comparación con el SCA(1)_Uni2Opt_r y el SCA_2Opt_r. Por lo cual, el SCA(1)_Multi2Opt_r presentó mejor *Eficacia*.
- El *Nivel de Eficacia* del SCA(1)_Uni2Opt_r resultó ser mayor al del SCA_2Opt_r en un 69%.
- El *Nivel de Eficacia* del SCA(1)_Multi2Opt_r resultó ser el mejor comparado con el SCA(1)_Uni2Opt_ y el SCA_2Opt_r.

4.1.4. Eficacia y Nivel de Eficacia de versiones Heavy y versiones Soft vs algoritmos de Pérez (2011).

Se comparó el *Error Relativo* de las soluciones obtenidas por los algoritmos desarrollados en sus dos versiones, en comparación con el *Error Relativo* de las versiones originales desarrolladas por Pérez (2011):

Algoritmos, versiones Soft:

SCA(1)_Uni2Opt, SCA(1)_Uni2Opt_r y el SCA(5)_Uni2Opt

Algoritmos, versiones Heavy:

SCA(1)_Multi2Opt, SCA(1)_Multi2Opt_r y el SCA(5)_Multi2Opt

Algoritmos, versiones de Pérez (2011):

SCA_2Opt y SCA_2Opt_r

El objetivo de dichas comparaciones consistió en conocer cuál de las versiones propuestas en la presente investigación resultó ser más eficaz dentro de las instancias revisadas.

Por lo cual, los resultados se consolidaron en las Tablas 10 y 11.

Tabla 10

Soluciones de las versiones Heavy, Soft y de Pérez (2011)

		Algoritmos Pérez(2011)		Algoritmos versiones Soft			Algoritmos versiones Heavy		
Instancia TSPLIB	Óptimo(*) [A]	SCA_ 2Opt [B]	SCA_ 2Opt_r [C]	SCA(1)_ Uni2Opt [D]	SCA(1)_ Uni2Opt_r [E]	SCA(5)_ Uni2Opt [F]	SCA(1)_ Multi2Opt [G]	SCA(1)_ Multi2Opt_r [H]	SCA(5)_ Multi2Opt [I]
1.Swiss42	1273	1375	1285	1273	1273	1273	1273	1273	1273
2.Eil50	425	448.83	430.48	436	429	429	429	427	429
3.Eil51	426	449.85	436.08	437	430	430	430	429	430
4.St70	675	713.98	690.62	732	728	725	688	686	686
5.Eil75	535	588.61	560.14	563	553	563	547	535	542
6.Eil76	538	567.01	559.61	567	553	567	548	543	546
7.Rat99	1211	1268.95	1259.51	1301	1266	1265	1258	1229	1238
8.KroA100	21282	22379.4	22219.3	23314	22786	23278	21474	21474	21470
9.KroA150	26524	29076.9	27999.7	28139	28107	28075	27493	27446	27243
10.Krob150	26130	28136.3	26904.48	26651	26591	26651	26651	26439	26532
11.d198	15780	16212	16180.53	16260	16009	15902	15926	15907	15902
12.Tsp225	3916	4071	4054.77	4040	4040	4031	3976	3949	3976
13.Pcb442	50778	55133.8	53287.29	53223	52871	53167	51977	51730	51811

(*) Solución óptima proporcionada por la librería virtual TSPLIB, donde las 13 instancias TSPLIB fueron seleccionadas según inciso “a” del numeral 4.1.1.

Tabla 11

Error relativo de las versiones Heavy, Soft y de Pérez (2011)

		Algoritmos Pérez(2011)		Algoritmos versiones Soft			Algoritmos versiones Heavy		
Instancia TSPLIB	Óptimo (*)	SCA_ 2Opt [B-A]/[A]	SCA_ 2Opt_r [C-A]/[A	SCA(1)_ Uni2Opt [D-A]/[A]	SCA(1)_ Uni2Opt_r [E-A]/[A	SCA(5)_ Uni2Opt [F-A]/[A]	SCA(1)_ Multi2Opt [G-A]/[A]	SCA(1)_ Multi2Opt_r [H-A]/[A]	SCA(5)_ Multi2Opt [I-A]/[A]
1.Swiss42	1273	8.0%	0.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
2.Eil50	425	5.6%	1.3%	2.6%	0.9%	0.9%	0.9%	0.5%	0.9%
3.Eil51	426	5.6%	2.4%	2.6%	0.9%	0.9%	0.9%	0.7%	0.9%
4.St70	675	5.8%	2.3%	8.4%	7.9%	7.4%	1.9%	1.6%	1.6%
5.Eil75	535	10.0%	4.7%	5.2%	3.4%	5.2%	2.2%	0.0%	1.3%
6.Eil76	538	5.39%	4.0%	5.4%	2.8%	5.39%	1.9%	0.9%	1.5%
7.Rat99	1211	4.8%	4.0%	7.4%	4.54%	4.46%	3.9%	1.5%	2.2%
8.KroA100	21282	5.2%	4.4%	9.5%	7.1%	9.4%	0.9%	0.9%	0.88%
9.KroA150	26524	9.6%	5.6%	6.1%	6.0%	5.8%	3.7%	3.5%	2.7%
10.Krob150	26130	7.7%	3.0%	2.0%	1.8%	2.0%	2.0%	1.2%	1.5%
11.d198	15780	2.7%	2.5%	3.0%	1.5%	0.77%	0.9%	0.80%	0.77%
12.Tsp225	3916	4.0%	3.5%	3.1%	3.1%	2.9%	1.5%	0.8%	1.5%
13.Pcb442	50778	8.6%	4.9%	4.8%	4.1%	4.7%	2.4%	1.9%	2.0%

(*) Solución óptima proporcionada por la librería virtual TSPLIB, donde las 13 instancias TSPLIB fueron seleccionadas según inciso “a” del numeral 4.1.1.

En la Tabla 11, para las trece instancias seleccionadas, el *Error Relativo* de las soluciones según el algoritmo utilizado presentó una variación, según lo indicado en la Tabla 12, en la cual el rango del error relativo se obtuvo tomando el mínimo y máximo de los errores relativos de los algoritmos que figuran en la Tabla 11.

Tabla 12

Rango del error relativo de las versiones Heavy, Soft y de Pérez(2011)

Tipo de algoritmo	Descripción del algoritmo	Rango Error Relativo
Versiones Heavy	SCA(5)_Multi2Opt	[0%; 2.7%]
	SCA(1)_Multi2Opt_r	[0%; 3.5%]
	SCA(1)_Multi2Opt	[0%; 3.9%]
Versiones Soft	SCA(5)_Uni2Opt	[0%; 9.4%]
	SCA(1)_Uni2Opt_r	[0%; 7.9%]
	SCA(1)_Uni2Opt	[0%; 9.5%]
Versiones de Pérez (2011)	SCA_2Opt_r	[0.9%; 5.6%]
	SCA_2Opt	[2.7%; 10%]

Nota. Rango de Error Relativo para las 13 instancias revisadas de la librería virtual TSPLIB seleccionadas según inciso “a” del numeral 4.1.1.

En la Tabla 12 se visualiza rápidamente qué algoritmos presentan mayor eficacia al estar sus errores relativos más próximos al 0%. Con respecto a las trece instancias revisadas (Ver Tabla 13), se realizó la comparación entre los algoritmos desarrollados, los cuales se resumen a continuación:

- El Nivel de Eficacia del SCA(5)_Multi2Opt, comparado con el SCA_2Opt_r, fue de 100% [(13/13) x 100%].
- El Nivel de Eficacia del SCA(5)_Uni2Opt, comparado con el SCA_2Opt, fue de 85% [(11/13) x 100%].
- El Nivel de Eficacia del SCA(5)_Uni2Opt, comparado con el SCA_2Opt_r, fue de 54% [(7/13) x 100%].
- El Nivel de Eficacia del SCA(5)_Multi2Opt, comparado con el SCA(5)_Uni2Opt, fue de 100% [(13/13) x 100%].

Tabla 13

Nivel de eficacia promedio: versiones Heavy, Soft y de Pérez (2011)

		VERSIONES HEAVY			VERSIONES SOFT			PÉREZ		Nivel de Eficacia Promedio	Categoría Nivel de Eficacia
	Descripción del algoritmo	SCA(5)_ Multi2Opt	SCA(1)_ Multi2Opt_r	SCA(1)_ Multi2Opt	SCA(5)_ Uni2Opt	SCA(1)_ Uni2Opt_r	SCA(1)_ Uni2Opt	SCA_ 2Opt	SCA_ 2Opt_r		
HEAVY	SCA(5)_Multi2Opt	-	38%	100%	100%	100%	100%	100%	100%	91%	Muy Alto
	SCA(1)_Multi2Opt_r	77%	-	100%	92%	100%	100%	100%	100%	96%	Muy Alto
	SCA(1)_Multi2Opt	31%	15%	-	92%	92%	100%	100%	100%	76%	Alto
SOFT	SCA(5)_Uni2Opt	31%	15%	38%	-	62%	100%	85%	54%	55%	Medio
	SCA(1)_Uni2Opt_r	23%	8%	31%	62%	-	100%	85%	69%	54%	Medio
	SCA(1)_Uni2Opt	8%	8%	15%	31%	15%	-	69%	31%	25%	Bajo
PEREZ	SCA_2Opt	0%	0%	0%	23%	15%	38%	-	0%	11%	Muy Bajo
	SCA_2Opt_r	0%	0%	0%	46%	31%	69%	100%	-	35%	Bajo

Nota. Se calculó el Nivel de Eficacia en base a las 13 instancias seleccionadas de la librería virtual TSPLIB indicadas en el inciso “a” del numeral 4.1.1.

Los elementos de la Tabla 13 son leídos de izquierda a derecha (tomando el algoritmo indicado en la fila para ser comparado con el algoritmo tomado en la columna); por ejemplo:

- El Nivel de Eficacia del SCA(5)_Multi2Opt con respecto al SCA(1)_Multi2Opt_r fue de 38%.
- El nivel de Eficacia del SCA(1)_Multi2Opt_r con respecto al SCA(5)_Multi2Opt fue de 77%.

A partir de la Tabla 13, se adiciona el ranking del Nivel de Eficacia promedio cada algoritmo, el cual es consolidado en la Tabla 14.

Tabla 14

Ranking del nivel de eficacia: versiones Heavy, Soft y de Pérez(2011)

Versión	Descripción del algoritmo	Nivel Eficacia Promedio	Ranking
HEAVY	SCA(5)_Multi2Opt	91%	2do Puesto
	SCA(1)_Multi2Opt_r	96%	1er Puesto
	SCA(1)_Multi2Opt	76%	3er Puesto
SOFT	SCA(5)_Uni2Opt	55%	4to Puesto
	SCA(1)_Uni2Opt_r	54%	5to Puesto
	SCA(1)_Uni2Opt	25%	7mo Puesto
PÉREZ	SCA_2Opt	11%	8vo Puesto
	SCA_2Opt_r	35%	6to Puesto

Nota. Se calculó el Nivel de Eficacia en base a las 13 instancias seleccionadas de la librería virtual TSPLIB indicadas en el inciso “a” del numeral 4.1.1.

De la Tabla 14, se concluyó que el algoritmo con mayor “Nivel de Eficacia Promedio”, entre las versiones desarrolladas y originales, corresponde al SCA(1)_Multi2Opt_r.

Conclusión:

- El SCA(5)_Multi2Opt presentó una variación menor en comparación con las soluciones obtenidas por los demás algoritmos, siendo el SCA(1)_Multi2Opt_r el que presentó una segunda menor variación y por último lugar el SCA_2Opt.

- Se recomienda el uso del SCA(1)_Multi2Opt_r como la mejor versión de los algoritmos desarrollados.

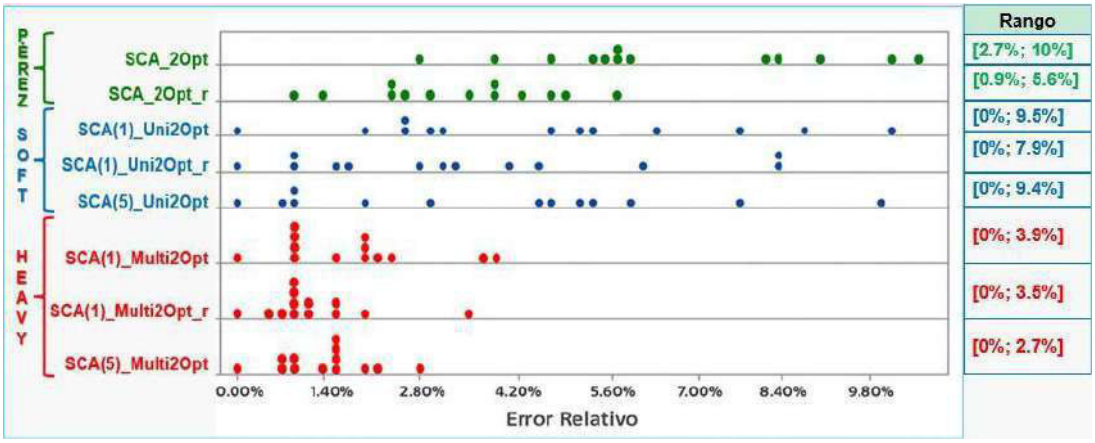
- Los algoritmos desarrollados en su versión Heavy presentaron un Nivel de Eficacia mayor que sus versiones Soft.

4.1.5. Variación del error relativo de versiones Heavy y Soft vs algoritmos de Pérez (2011)

Con la información mostrada en la Tabla 11, se construyó una gráfica de puntos en el software Minitab 20 (ver Figura 62), la cual nos ayuda a visualizar el rango en que están comprendidos los *Errores Relativos* distribuyéndolos a lo largo del eje horizontal, según el algoritmo utilizado para las instancias de la librería virtual TSPLIB que se analizaron: Swiss42, eil50, eil51, st70, eil75, eil76, rat99, KroA100, KroA150, Krob150, d198, Tsp225, Pcb442.

Figura 62

Gráfica de puntos del error relativo: versiones Heavy, Soft y de Pérez(2011)



Nota. Los puntos representan los valores del Error Relativo para las 13 instancias seleccionadas en el inciso “a” del numeral 4.1.1.

En la Figura 62, se puede observar que los *Errores Relativos* de las soluciones obtenidas por los tres algoritmos desarrollados, en sus versiones Heavy, en la presente investigación (SCA(1)_Multi2Opt, SCA(1)_Multi2Opt_r, SCA(5)_Multi2Opt), se acercan más al óptimo (caso cuando el error relativo es igual a 0%) que los *Errores Relativos* de las soluciones obtenidas por sus versiones originales (SCA_2Opt y SCA_2Opt_r). Esto nos sirve para, de manera visual, ver cuáles soluciones son más próximas al óptimo, donde cuanto más próximo se encuentre el error relativo al 0% mejor será su eficacia.

Por su parte, los algoritmos desarrollados en su versión Soft presentan un comportamiento similar a las versiones originales desarrolladas por Pérez (2011). Por lo cual, basándonos en la gráfica, no se puede decir cuál de las versiones resultó más eficaz: si las versiones Soft o sus versiones originales desarrolladas por Pérez (2011).

4.1.6. Eficacia de versiones Heavy y Soft vs metaheurísticas seleccionadas

Se comparó el *Error Relativo* de las soluciones obtenidas por los algoritmos desarrollados en la presente investigación, tanto en las versiones Heavy como Soft, con el *Error Relativo* de las soluciones obtenidas por las metaheurísticas seleccionadas, según Anexo 3, Anexo 4, Anexo 5, Anexo 6 y Anexo 7. Los algoritmos cuyas soluciones, para las mismas instancias, lograron superar a las obtenidas por los algoritmos originales desarrollados por Pérez (2011) se consolidaron en la Tabla 15.

Tabla 15

Solución y error relativo de algoritmos metaheurísticos seleccionados vs versiones Soft y Heavy según instancia TSPLIB

			Algoritmos Genéticos				Colonia de Hormigas		Otras Metaheurísticas	Versiones Soft			Versiones Heavy		
	PAIS ->		Bélgica y Suiza		Libia		Bélgica y Suiza		Bélgica y Suiza	Perú			Perú		
	Instancia TSPLIB	Óptimo TSPLIB	AG	FRAG_ GA	SWAP_ GATSP	IGA	ACS	ACS_ 3Opt	Programación Evolutiva (EP)	SCA(1) _Uni 2Opt	SCA(1) _Uni 2Opt_r	SCA(5) _Uni 2Opt	SCA(1) _ Multi 2Opt	SCA(1) _ Multi 2Opt_r	SCA(5) _ Multi 2Opt
DISTANCIA	1.Eil50	425	428				425		426	436	429	429	429	427	429
	2.Eil75	535	545				535		542	563	553	563	547	535	542
	3.KroA100	21282	21761	21282	21504	21282	21282			23314	22786	23278	21474	21474	21470
	4.D198	15780	-	15834	15992	15781	15888	15780		16260	16009	15902	15926	15907	15902
	5.Ts225	126643		126730	127012	--				130496	127794	130496	127512	126899	127161
	6.Pcb442	50778		51104	52620	51101	51268			53223	52871	53167	51977	51730	51811
	7.att532	27686					28147	27693		29081	28892	29081	28416	28393	28416
	8.Rat783	8806		9007	9732	8924	9015	8818		9210	9179	9184	9119	9105	9111
	9.lin318	42029						42029		44049	44049	44049	43450	43204	43046
ERROR RELATIVO	1.Eil50	425	0.7%				0.0%		0.2%	2.6%	0.9%	0.9%	0.9%	0.5%	0.9%
	2.Eil75	535	1.9%				0.0%		1.3%	5.2%	3.4%	5.2%	2.2%	0.0%	1.3%
	3.KroA100	21282	2.3%	0.0%	1.0%	0.0%	0.0%			9.5%	7.1%	9.4%	0.9%	0.9%	0.88%
	4.D198	15780	-	0.3%	1.3%	≈0.0%	0.7%	0.0%		3.0%	1.4%	0.77%	0.93%	0.81%	0.77%
	5.Ts225	126643		0.1%	0.3%					3.0%	0.9%	3.0%	0.7%	0.2%	0.4%
	6.Pcb442	50778		0.642%	3.6%	0.636%	1.0%			4.8%	4.1%	4.7%	2.4%	1.9%	2.0%
	7.att532	27686					1.7%	0.0%		5.0%	4.4%	5.0%	2.64%	2.55%	2.64%
	8.Rat783	8806		2.3%	10.5%	1.3%	2.4%	0.1%		4.6%	4.2%	4.3%	3.6%	3.4%	3.5%
	9.lin318	42029						0.0%		4.8%	4.8%	4.8%	3.4%	2.8%	2.4%

Nota. Las instancias de la librería virtual TSPLIB fueron seleccionadas en numeral 4.1.1. inciso b.

En la Tabla 15 se presentan las instancias seleccionadas según Anexo 3, Anexo 4, Anexo 5, Anexo 6 y Anexo 7. Se puede observar que las soluciones determinadas por los algoritmos desarrollados en la presente investigación, tanto en sus versiones Soft como Heavy, resultaron ser competitivas con las soluciones determinadas por la metaheurísticas seleccionadas, donde:

- Las soluciones proporcionadas por las versiones Soft se aproximan a las soluciones encontradas por las metaheurísticas para las nueve instancias revisadas.

- Con respecto a las versiones Heavy, encontramos una mejoría que supera, en todas sus instancias, a las versiones Soft y también supera, o, en el peor escenario, se aproxima a las soluciones encontradas por los algoritmos metaheurísticos seleccionados.

En la Tabla 15 se puede observar que:

- Para las versiones Soft, según las instancias revisadas, el Error Relativo se encuentra comprendido en el rango de 0.8% a 9.5% con respecto al óptimo reportado en librería virtual TSPLIB.

- Mientras que, para las versiones Heavy, según las instancias revisadas, el rango de variación del Error Relativo está comprendido entre 0% a 3.6% con respecto al óptimo reportado en TSPLIB.

4.1.7. Nivel de eficacia de versiones Heavy y Soft vs metaheurísticas seleccionadas

A continuación, a partir de la Tabla 15, se realizó un análisis para medir el Nivel de Eficacia de los tres algoritmos propuestos (al tomarse como un solo grupo), comparadas con las metaheurísticas ejecutadas en las instancias de la librería virtual TSPLIB seleccionadas para la presente investigación (para revisar instancias seleccionadas, ver Tabla 15).

Los resultados se muestran según el grupo de metaheurísticas descrito a continuación:

a. Del grupo de Algoritmos Genéticos

- Para el AG, el SCA(1)_Multi2Opt_r logró ser más eficaz para las tres instancias revisadas: Eil50, Eil75 y KroA100.
    - Es decir, el mejor Nivel de Eficacia alcanzado por las versiones Heavy fue de hasta 3/3 *(*Nivel de Eficacia muy alto*,* según clasificación indicada en la Tabla 7).
    - Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).

- Para el FRAG_GA, los tres algoritmos desarrollados en sus versiones Heavy no lograron ser más eficaces que el FRAG_GA en total de cinco instancias revisadas: KroA100, D198, Ts225, Pcb442, Rat783.
    - Es decir, el Nivel de Eficacia de las versiones Heavy fue de 0/5 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).
    - Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0/5 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).
- Para el SWAP_GATSP, el SCA(1)_Multi2Opt_r logró ser el mejor Nivel de Eficacia para las cinco instancias revisadas: KroA100, D198, Ts225, Pcb442, Rat783.

  - Es decir, el mejor Nivel de Eficacia de las versiones Heavy fue de hasta 5/5 (Nivel de Eficacia muy alto, según clasificación indicada en la Tabla 7).

  - Con respecto al mejor Nivel de Eficacia de las versiones Soft, se tuvo un valor de hasta 2/5 (Nivel de Eficacia medio, según clasificación indicada en la Tabla 7).

- Para el IGA, los tres algoritmos desarrollados en sus versiones Heavy no lograron ser más eficaces que el IGA, del total de cuatro instancias revisadas: KroA100, D198, Pcb442, Rat783.

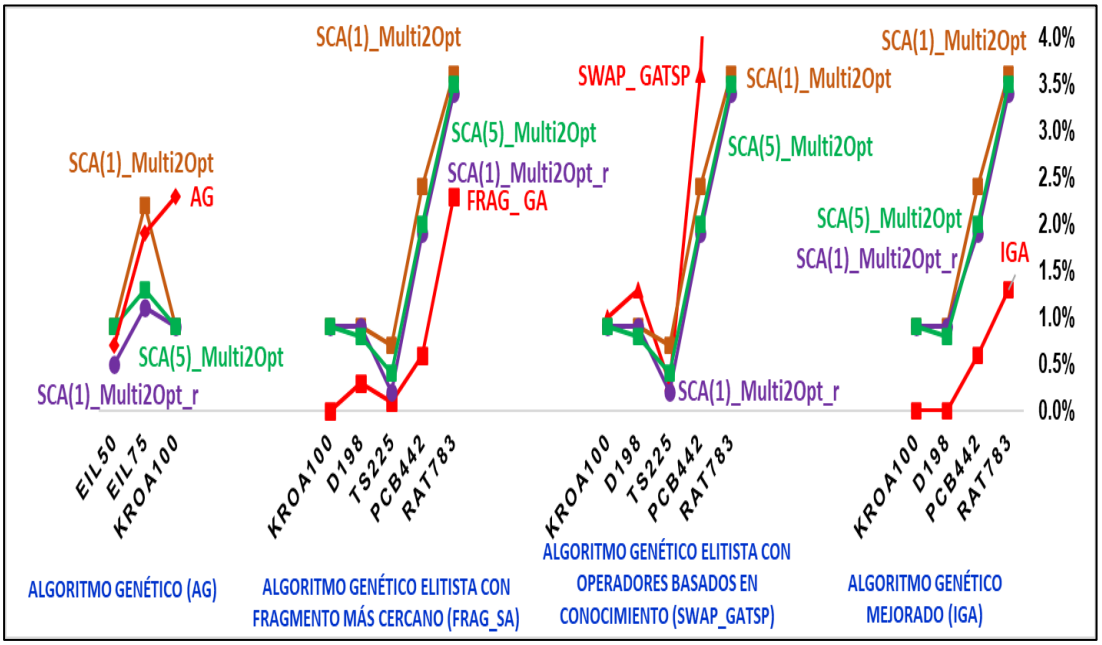
  - Es decir, el Nivel de Eficacia de las versiones Heavy fue de 0/4 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).

  - Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).

A partir de la Tabla 15, se construyó la Figura 63, la cual muestra el comportamiento del *Error Relativo* del grupo de Algoritmos Genéticos comparado con el *Error Relativo* de los algoritmos propuestos en sus versiones Heavy.

Figura 63

Error relativo de versiones Heavy vs Algoritmos Genéticos



Nota. En el algoritmo SWAP_GATSP, el valor de 10.5% para la instancia Rat783 se omitió para evitar distorsionar la gráfica.

b. Del grupo de algoritmos System Colony Ant (ACS)

- Para el ACS, solo el SCA(1)_Multi2Opt_r logró igualar al óptimo para la instancia Eil75, del total de siete instancias revisadas: Eil50, Eil75, KroA100, d198, Pcb442, att532, Rat783.

    - Es decir, el mejor Nivel de Eficacia de las versiones Heavy fue de 1/7 (Nivel de Eficacia muy bajo, según la clasificación indicada en la Tabla 7).

    - Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0/7 (Nivel de Eficacia muy bajo, según la clasificación indicada en la Tabla 7).

- Para el ACS_3Opt, ninguno de los tres algoritmos desarrollados en sus versiones Heavy logró mayor eficacia para las cuatro instancias TSPLIB revisadas: d198, att532, Rat783, lin318.

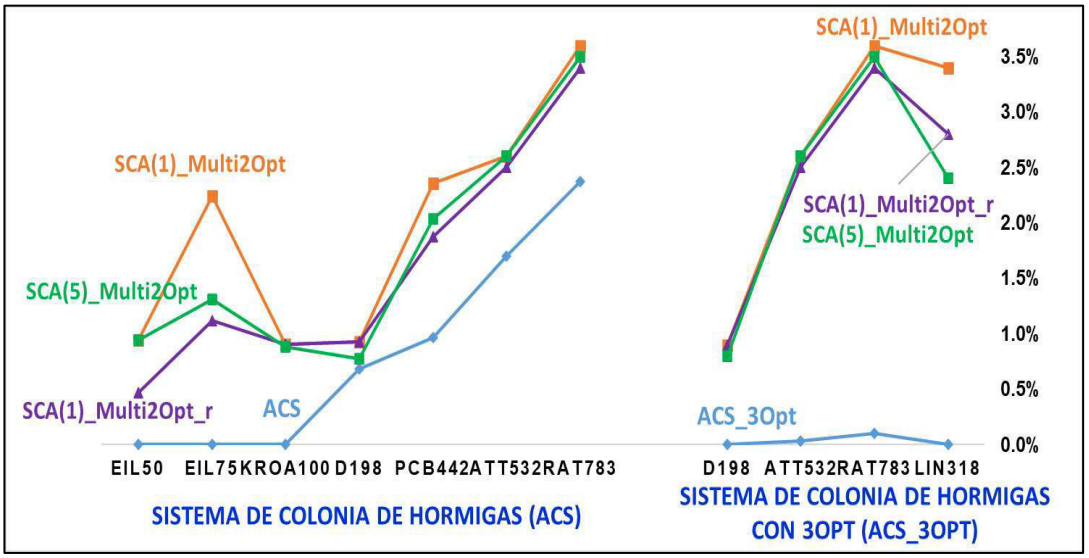
- Es decir, el Nivel de Eficacia de las versiones Heavy fue de 0 (Nivel de Eficacia muy bajo, según la clasificación indicada en la Tabla 7).

- Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0 (Nivel de Eficacia muy bajo, según la clasificación indicada en la Tabla 7).

A partir de la Tabla 15, se construyó la Figura 64, la cual muestra el comportamiento del Error Relativo del grupo de System Colony Ant, comparado con el Error Relativo de los algoritmos propuestos en sus versiones Heavy.

Figura 64

Error relativo de versiones Heavy vs System Colony Ant



c. Del Grupo de Otras Metaheurísticas

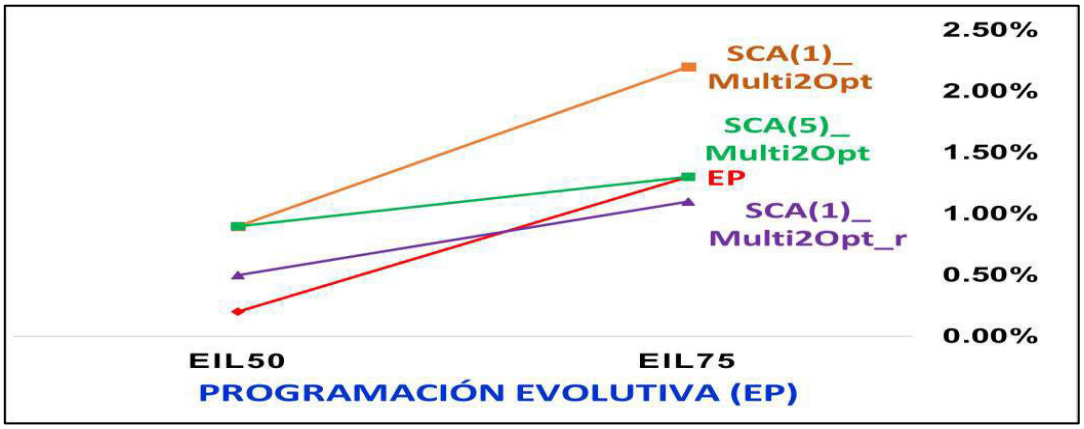
- Para el algoritmo de Programación Evolutiva (PE), solo el SCA(1)_Multi2Opt_r logró ser mayor eficacia para la instancia Eil75, del total de dos instancias revisadas: Eil50, Eil75.

- Es decir, el Nivel de Eficacia de las versiones Heavy fue de 1/2 (Nivel de Eficacia medio, según clasificación indicada en la Tabla 7).
- Con respecto al Nivel de Eficacia de las versiones Soft, se tuvo un valor de 0 (Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).

A partir de la Tabla 15, se construyó la Figura 65, la cual muestra el comportamiento del *Error Relativo* del grupo de Programación Evolutiva (PE), comparado con el *Error Relativo* de los algoritmos propuestos en sus versiones Heavy.

Figura 65

Error relativo de versiones Heavy vs Programación Evolutiva



A continuación, se muestra en la Tabla 16 el mejor nivel de Eficacia de las versiones Heavy al compararse con los tres grupos de metaheurísticas, con respecto a las instancias de la librería virtual TSPLIB seleccionadas del inciso “b” del numeral 4.1.1.

Tabla 16

Mejor nivel de eficacia: versiones Heavy y metaheurísticas seleccionadas

Descripción	Nivel de Eficacia	Categoría Eficacia
Algoritmos versiones Heavy vs AG	1	Muy Alto
Algoritmos versiones Heavy vs FRAG_GA	0	Muy Bajo
Algoritmos versiones Heavy vs SWAP_GATSP	1	Muy Alto
Algoritmos versiones Heavy vs IGA	0	Muy Bajo
Algoritmos versiones Heavy vs ACS	1/7	Muy Bajo
Algoritmos versiones Heavy vs ACS_3Opt	0	Muy Bajo
Algoritmos versiones Heavy vs PE	1/2	Medio

Nota. Las metaheurísticas se seleccionaron en el inciso “b” del numeral 4.1.1

A partir de la Tabla 16, se puede concluir que:

- El mejor Nivel de Eficacia de las versiones Heavy resultó ser de 100% al compararse con el Algoritmo Genético (AG) y con el SWAP_GATSP (el cual representa un Nivel de Eficacia muy alto, según clasificación indicada en la Tabla 7).

- El mejor Nivel de Eficacia de las versiones Heavy resultó ser de 0% al compararse con el FRAG_GA, IGA y el ACS_3Opt (el cual representa un Nivel de Eficacia muy bajo, según clasificación indicada en la Tabla 7).

- El mejor Nivel de Eficacia de las versiones Heavy resultó estar entre 14,2% a 50% para los algoritmos ACS y para Programación Evolutiva (PE) (el cual varió entre un Nivel de Eficacia muy bajo a un nivel medio, según clasificación indicada en la Tabla 7).

Por lo cual, el Nivel de Eficacia depende no solo del algoritmo metaheurístico utilizado, sino también de las peculiaridades intrínsecas de la instancia del Problema del Agente Viajero simétrico a resolver.

A continuación, se muestra en la Tabla 17 el *Mejor Nivel de Eficacia* de las versiones Soft al compararse con los tres grupos de metaheurísticas, con respecto a las instancias seleccionadas del inciso “b” del numeral 4.1.1.

Tabla 17

Mejor nivel de eficacia: versiones Soft y metaheurísticas seleccionadas

Descripción	Nivel de Eficacia	Categoría
Algoritmos versiones Soft vs AG	0	Muy Bajo
Algoritmos versiones Soft vs FRAG_GA	0	Muy Bajo
Algoritmos versiones Soft vs SWAP_GATSP	2/5	Medio
Algoritmos versiones Soft vs IGA	0	Muy Bajo
Algoritmos versiones Soft vs ACS	0	Muy Bajo
Algoritmos versiones Soft vs ACS_3Opt	0	Muy Bajo
Algoritmos versiones Soft vs PE	0	Muy Bajo

Nota. Las metaheurísticas se seleccionaron en el inciso “b” del numeral 4.1.1

De la Tabla 17, el Mejor nivel de eficacia de los algoritmos propuestos en sus versiones Soft, con respecto a las Metaheurísticas revisadas, resultó ser en general Muy Bajo o Medio, donde:

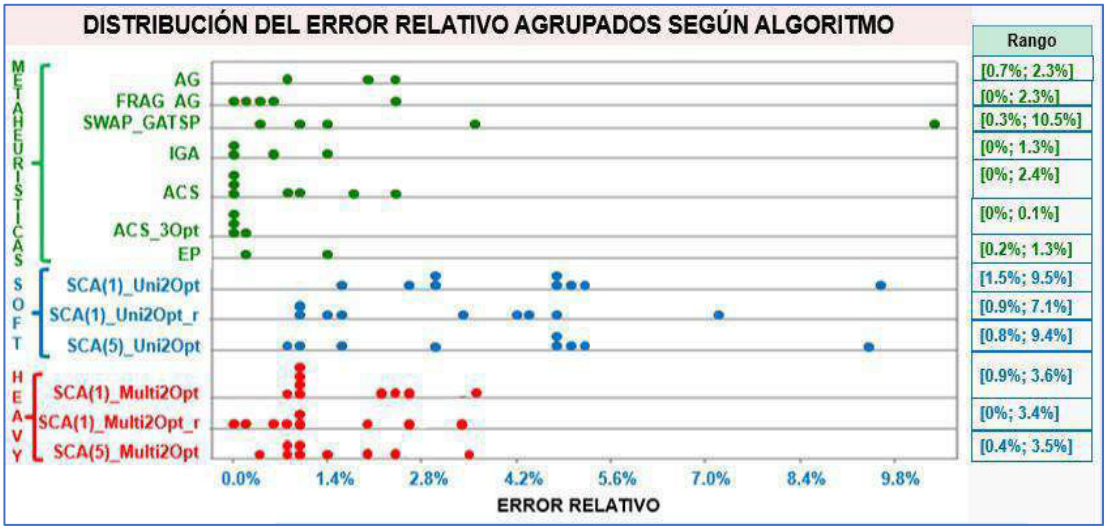
- El mejor nivel de eficacia de las versiones Soft resultó de 0% al compararse con el AG, FRAG_SA, IGA, ACS, ACS_3Opt y PE (Nivel de Eficacia muy bajo, según clasificación en Tabla 7).
- El mejor nivel de eficacia de las versiones Soft resultó de un 40% (2/5) al compararse con el SWAP_GATSP (Nivel de Eficacia medio, según clasificación indicada en Tabla 7).

4.1.8. Variación del error relativo según algoritmo metaheurístico

A partir de la Tabla 15, se construyó una gráfica de puntos, en el software Minitab 20, la cual ayuda a visualizar la dispersión de los errores relativos, distribuyéndolos a lo largo del eje horizontal (ver Figura 66). Las instancias de la librería virtual TSPLIB revisadas fueron las siguientes: Eil50, Eil51, KroA100, D198, Ts225, Pcb442, att532, Rat783, lin318.

Figura 66

Gráfica de puntos del error relativo: versión Heavy, Soft y metaheurísticas



Nota. Los puntos muestran el Error Relativo de las soluciones obtenidas para las metaheurísticas y las instancias seleccionadas en el inciso “b” del numeral 4.1.1.

En la Figura 66, se puede observar que el *Error Relativo* de las soluciones obtenidas por los tres algoritmos desarrollados en la presente investigación, en sus versiones Heavy, son muy cercanos al *Error Relativo* de las soluciones obtenidas por las metaheurísticas seleccionadas (AG, FRAG_GA, SWAP_GATSP, IGA, ACS, ACS_3Opt, EP). Lo cual muestra la Eficacia de las soluciones obtenidas por nuestros algoritmos.

Tener presente que Reinelt (s.f.) indica que, al ser calculadas las matrices de costos según las indicaciones correspondientes para su construcción indicadas en TSPLIB (distancia euclidiana, pseudo euclidiana,

distancia GEO, según corresponda), los valores indicados como solución por la librería virtual TSPLIB son los valores óptimos, los cuales fueron validados por algoritmos de *Branch and Bound y Branch and Cut*, entre otros algoritmos.

4.2. Validación de Hipótesis

A continuación, se procedió a exponer la hipótesis general y las hipótesis específicas, con su respectivo argumento, el cual indica el cumplimiento o no de estas hipótesis, según la presentación de resultados del numeral 4.1.

4.2.1. Hipótesis General

Mediante el algoritmo propuesto será posible resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que con los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas.

Explicación

Se indican las modificaciones realizadas a las versiones originales del SCA_2Opt y el SCA_2Opt_r:

- Criterio Multi2Opt, para convertir el proceso de búsqueda de tipo basado en la población para las versiones Heavy; y criterio Uni2Opt para las versiones Soft, para una búsqueda basada en la trayectoria.

- Establecer como criterio de parada una cantidad de ejecuciones del Multi2Opt, la cual es determinada por medio de ensayo y error, según el orden del grafo. La cual puede generalizarse a un criterio de parada igual a la cantidad de nodos multiplicada por el número 6, según el Anexo 19.

- El factor “r”, expresado linealmente como “(1 + r)”, el cual es indicado dentro de la investigación de Pérez (2011), puede incluir valores negativos para así reducir el efecto del “Costo Nuevo” en el criterio de decisión para intercambiar nodos de la heurística 2Opt, tal que se sugieren valores para $r \in [-0.3; 0.3]$. Dicho rango se determinó con base en ejecuciones para 14 instancias simétricas de TSPLIB del algoritmo SCA(1)_Multi2Opt_r; para mayor detalle, ver numeral 3.16.2, inciso “b. Determinación del rango de valores para el factor “r”, como conclusiones de la Tabla 4.

- *Vector Prohibición*, para poder ampliar nuestro rango de búsqueda al prohibir múltiples arcos, según indicaciones del vector prohibición, con la esperanza de encontrar nuevas regiones del espacio de soluciones factibles sin explorar.

De los cuatro puntos mencionados, se logró:

- Construir tres algoritmos en sus versiones denominadas Heavy:
  - SCA(1)_Multi2Opt (ver diagrama de flujo en Figura 39).
  - SCA(1)_Multi2Opt_r y (ver diagrama de flujo en Figura 43).
  - SCA(A)_Multi2Opt (ver diagrama de flujo en Figura 48).

- Construir tres algoritmos en sus versiones denominadas Soft:
  - SCA(1)_Uni2Opt (ver diagrama de flujo en Figura 51).
  - SCA(1)_Uni2Opt_r y (ver diagrama de flujo en Figura 54).
  - SCA(A)_Uni2Opt (ver diagrama de flujo en Figura 57).

De la Tabla 13 y Tabla 14, para las trece instancias TSPLIB comparadas con la investigación de Pérez (2011) se encontró que:

- El Nivel de Eficacia de las versiones Heavy alcanzó un valor de 100% en comparación con los algoritmos originales de Pérez (2011) (SCA_2Opt y SCA_2Opt_r). Es decir, un Nivel de Eficacia muy alto.

- El Nivel de Eficacia de las versiones Soft alcanzó un valor comprendido entre 31% a 85% en comparación con los algoritmos originales de Pérez (2011) (SCA_2Opt y SCA_2Opt_r). Es decir, un Nivel de Eficacia moderado, medio, alto y muy alto.

4.2.2. Hipótesis Específica 1

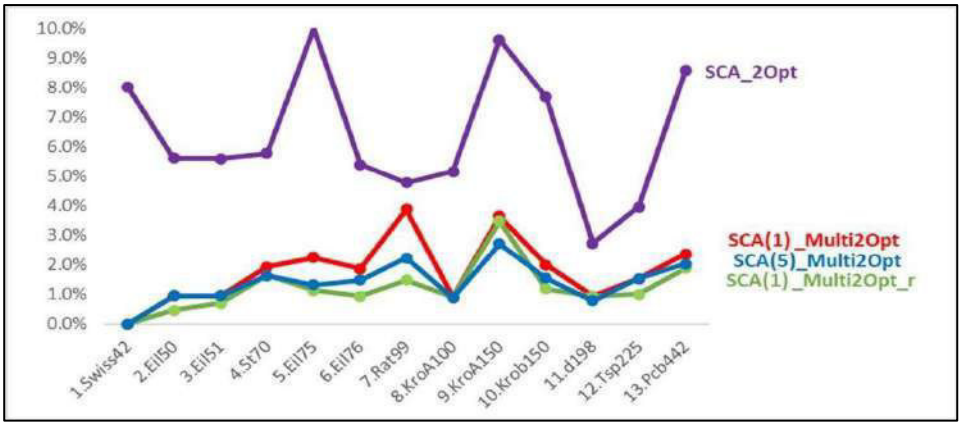
El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt.

a. Para las versiones Heavy

- **Para la Eficacia**: A partir de la Tabla 11 se construyó la Figura 67, la cual muestra el *Error Relativo* de las soluciones de los algoritmos desarrollados en sus versiones Heavy comparadas con el Error Relativo de las soluciones determinadas por el SCA_2Opt.

Figura 67

Gráfica del error relativo de versiones Heavy y el SCA_2Opt



Nota. Las 13 instancias fueron seleccionadas en 4.1.1., inciso “a” .

De la Figura 67, se puede observar que las soluciones obtenidas por las tres versiones de algoritmos desarrollados en sus versiones Heavy presentaron un menor *Error Relativo*, en todas las instancias revisadas, al compararse con el *Error Relativo* del algoritmo SCA_2Opt de Pérez (2011).

- **Para el Nivel de Eficacia:** De la Figura 67, se puede observar también que, para las trece instancias revisadas, en un 100% (Nivel de Eficacia) las soluciones obtenidas por las versiones Heavy presentaron mayor proximidad el óptimo al compararse con el SCA_2Opt (valores más cercanos al 0%).

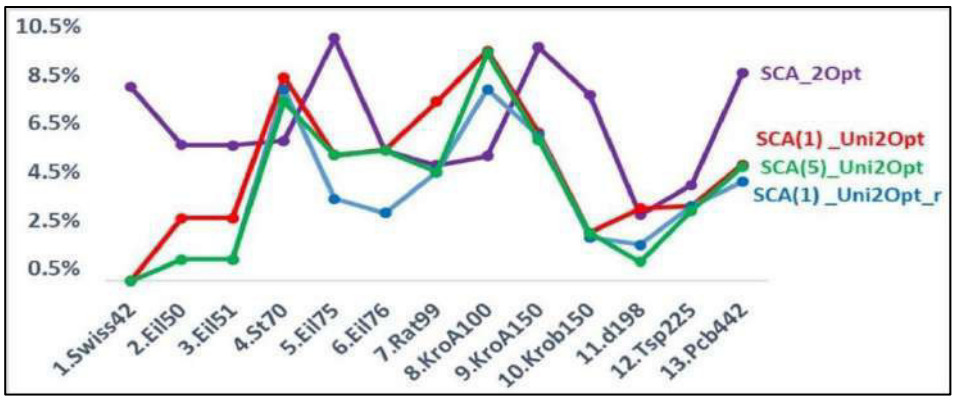
  Por tanto, para las versiones Heavy, se cumple la hipótesis específica 1.

b. Para las versiones Soft

- **Para la Eficacia**: A partir de la Tabla 11, se construyó la Figura 68, la cual muestra el *Error Relativo* de las soluciones de algoritmos desarrollados en sus versiones Soft, comparados con el Error Relativo de las soluciones del SCA_2Opt.

Figura 68

Gráfica del error relativo de versiones Soft y el SCA_2Opt



Nota. Las 13 instancias fueron seleccionadas en 4.1.1, el inciso “a”.

A partir de la Figura 68, se observa que las soluciones obtenidas por tres versiones Soft presentaron un *Error Relativo* variable, es decir, en algunas instancias resultó mejor o peor que el *Error Relativo* del SCA_2Opt. Por lo cual, se necesita del *Nivel de Eficacia* para cuantificar dicha comparación.

- **Para el Nivel de Eficacia:** De la Tabla 13, con respecto a las trece instancias revisadas (ver Tabla 11 para detalle de instancias) del problema del agente viajero simétrico de TSPLIB, se concluyó que el Nivel de Eficacia del algoritmo SCA(5)_Multi2Opt y el SCA(1)_Multi2Opt_r alcanzó un 85% al compararse con el SCA(1)_2Opt para ambos casos y del 69% al compararse con el SCA(1)_Multi2Opt. Es decir, las versiones Soft presentaron un Nivel de Eficacia entre el 69% al 85% al compararse con el SCA_2Opt.

Conclusión de la hipótesis específica 1:

- Las **versiones Heavy** para las trece instancias del problema del agente viajero simétrico de TSPLIB, presentaron un Nivel de Eficacia del 100% con respecto al Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt); por ello, **se cumple la hipótesis específica 1**.

- Las **versiones Soft** para las trece instancias del problema del agente viajero simétrico de TSPLIB presentaron un Nivel de Eficacia entre 69% a 85% con respecto al Sacrificio Cortoplacista Adaptativo 2Opt (SCA_2Opt). Por ello, **se cumple la hipótesis específica 1**.

4.2.3. Hipótesis Específica 2

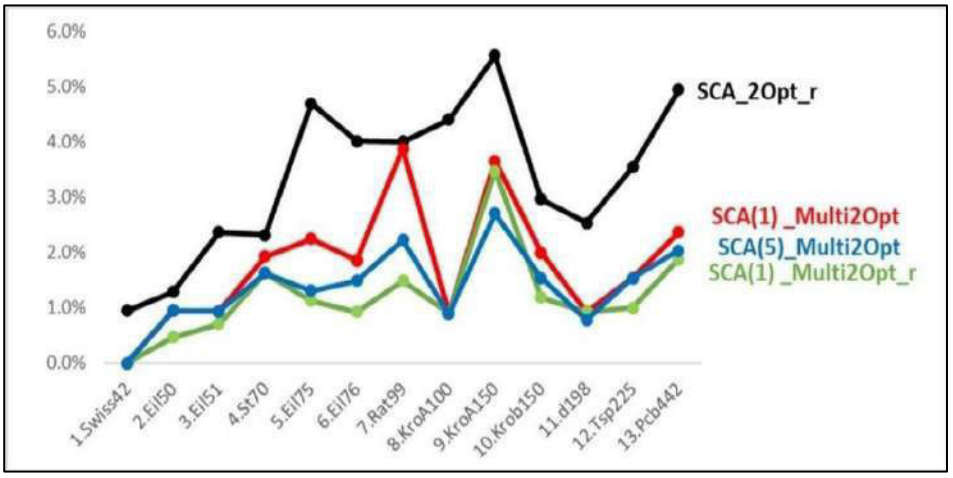
El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r.

a. Para las versiones Heavy

- **Para la Eficacia**: A partir de la Tabla 11, se construyó la Figura 69, la cual muestra el *Error Relativo* de las soluciones de algoritmos desarrollados en su versión Heavy, comparados con el Error Relativo de las soluciones del SCA_2Opt_r.

Figura 69

Gráfica del error relativo de versiones Heavy y el SCA_2Opt_r



Nota. Las 13 instancias fueron seleccionadas de 4.1.1., inciso “a”.

En la Figura 69, se observa que las soluciones obtenidas por las tres versiones Heavy presentaron un menor *Error Relativo*, en todas las instancias revisadas, al compararse con el *Error Relativo* del SCA_2Opt_r de Pérez (2011).

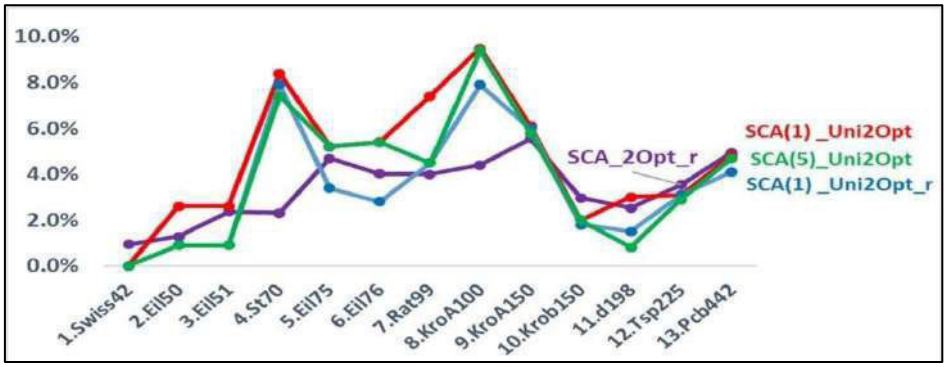
- **Para el Nivel de Eficacia:** En la Figura 69, se observa que, para las trece instancias revisadas, en un 100% las soluciones obtenidas por las versiones Heavy tuvieron mayor proximidad al óptimo que las soluciones del SCA_2Opt_r (valores más cercanos al 0%). Es decir, un Nivel de Eficacia de 100% por parte de las tres versiones Heavy comparándolas con el SCA_2Opt_r.

b. Para las versiones Soft

- **Para la Eficacia**: A partir de la Tabla 11, se construyó la Figura 70, la cual muestra el *Error Relativo* de las soluciones de algoritmos desarrollados en su versión Soft, comparadas con el *Error Relativo* de las soluciones del SCA_2Opt_r.

Figura 70

Gráfica del error relativo de versiones Soft y el SCA_2Opt_r



Nota. Las 13 instancias fueron seleccionadas del 4.1.1., inciso “a”.

En la Figura 70, se observa que las soluciones obtenidas por las tres versiones Soft presentaron un *Error Relativo* variable; es decir, en algunas instancias mejores y en otras peores que el *Error Relativo* obtenido del SCA_2Opt_r. Por lo cual, se necesita del Nivel de Eficacia para cuantificar dicha comparación.

- **Para el Nivel de Eficacia:** De la Tabla 13, con respecto a las trece instancias revisadas (ver Tabla 11 para detalle de instancias) del problema del agente viajero simétrico de TSPLIB, se concluyó que el Nivel de Eficacia de las versiones Soft: SCA(5)_Uni2Opt, SCA(1)_Uni2Opt_r y el SCA(1)_Uni2Opt, alcanzaron un Nivel de Eficacia del 54%, 69% y 31% respectivamente, al compararse con el algoritmo SCA(1)_2Opt_r. Es decir, se cumple parcialmente que las versiones Soft encuentran mejores soluciones que el SCA_2Opt_r.

Conclusión de Hipótesis Específica 2:

- Las **versiones Heavy**, para las trece instancias del Problema del Agente Viajero Simétrico de TSPLIB revisadas, alcanzaron un Nivel de Eficacia de 100% con respecto al Sacrificio Cortoplacista Adaptativo 2Opt_r (SCA_2Opt_r).

  Por lo cual**, se cumple** de manera completa **la hipótesis específica 2** para las instancias revisadas.

- Las **versiones Soft**, para las trece instancias del problema del agente viajero simétrico de TSPLIB revisadas, presentaron un Nivel de Eficacia entre un 31% a 69% con respecto al Sacrificio Cortoplacista Adaptativo 2Opt_r (SCA_2Opt_r). Como se tuvo un caso menor al 50%, esto quiere decir que el SCA(1)_Uni2Opt presentó un Nivel de Eficacia menor al SCA_2Opt_r.

  Por lo cual, **se cumple de manera parcial la hipótesis específica 2** para las instancias revisadas.

4.2.4. Hipótesis Específica 3

El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia de las metaheurísticas seleccionadas.

a. Para las versiones Heavy y Soft

- **Eficacia.** A partir de la Tabla 15, se obtuvieron los Errores Relativos de las soluciones mostradas según el grupo de algoritmos metaheurísticos, para las mismas instancias de la librería virtual TSPLIB con las cuales se comparó, las cuales se presentan en forma resumida en la Tabla 18.

Tabla 18

Intervalo del error relativo por metaheurística (mínimo - máximo)

Detalle de grupo de algoritmos	Error Relativo	
	Mínimo	**Máximo**
Algoritmos Genéticos (4 versiones)	0.0%	10.5%
Versiones Heavy (3 versiones)	0.0%	3.6%
Versiones Soft (3 versiones)	0.8%	9.5%
Grupo de Algoritmos	**De**	**A**
Colony System Ant (2 versiones)	0.0%	2.4%
Versiones Heavy (3 versiones)	0.0%	3.4%
Versiones Soft (3 versiones)	0.8%	9.5%
Grupo de Algoritmos	**De**	**A**
Programación Evolutiva (1 versión)	0.0%	1.3%
Versiones Heavy (3 versiones)	0.0%	2.2%
Versiones Soft (3 versiones)	0.9%	5.2%

Nota. Dicho rango se determinó con base en las instancias de TSPLIB y las metaheurísticas seleccionadas en el inciso “b” del numeral 4.1.1.

En la Tabla 18, se observa que los Errores Relativos de las soluciones de las versiones Heavy presentaron una menor variación con respecto al grupo de los algoritmos genéticos. Mientras que, para los grupos de Colony System Ant y Programación Evolutiva, los Errores Relativos de dichos grupos resultaron más cercanos al óptimo y menos dispersos que los Errores Relativos de las versiones Heavy y Soft.

- **Nivel de Eficacia.** De la Tabla 15, se construyó la Tabla 19, la cual consolida el nivel de eficacia y nivel de eficacia promedio encontrado para las versiones Heavy y Soft de los algoritmos desarrollados al compararlos con las metaheurísticas seleccionadas y los categoriza según lo indicado en la Tabla 7.

Tabla 19

Nivel de eficacia promedio de versiones Heavy, Soft y de los algoritmos metaheurísticos seleccionados

	Descripción del algoritmo	AG	FRAG_GA	SWAP_GA TSP	IGA	ACS	ACS_3Opt	EP	SCA(1)_Uni 2Opt	SCA(1)_Uni 2Opt_r	SCA(5)_Uni 2Opt	SCA(1)_Multi 2Opt	SCA(1)_Multi 2Opt_r	SCA(5)_Multi 2Opt	Nivel de Eficacia Promedio	Categoría Nivel de Eficacia
METAHEURISTICAS	**ACS_3Opt**	-	100%	100%	100%	100%	-	-	100%	100%	100%	100%	100%	100%	**100.0%**	Muy Alto
	IGA	100%	100%	100%	-	100%	0%	-	100%	100%	100%	100%	100%	100%	**90.9%**	Muy Alto
	FRAG_GA	100%	-	100%	25%	100%	0%	-	100%	100%	100%	100%	100%	100%	**84.1%**	Muy Alto
	Programación Evolutiva (EP)	100%	-	-	-	0%	-	-	100%	100%	100%	100%	50%	100%	**81.3%**	Muy Alto
	ACS	100%	25%	100%	25%	-	0%	100%	100%	100%	100%	100%	100%	100%	**79.2%**	Alto
	AG	-	0%	0%	0%	0%	-	-	100%	100%	100%	100%	0%	33%	**43.3%**	Medio
	SWAP_GATSP	100%	0%	-	0%	0%	0%	-	80%	80%	80%	20%	0%	20%	**34.5%**	Bajo
HEAVY	**SCA(1)_Multi2Opt_r**	100%	0%	100%	0%	0%	0%	50%	100%	100%	89%	100%	-	67%	**58.8%**	Alto
	SCA(5)_Multi2Opt	67%	0%	80%	0%	0%	0%	50%	100%	100%	100%	100%	33%	-	**52.5%**	Medio
	SCA(1)_Multi2Opt	33%	0%	80%	0%	0%	0%	0%	100%	100%	89%	-	11%	22%	**36.3%**	Bajo
SOFT	**SCA(1)_Uni2Opt_r**	0%	0%	20%	0%	0%	0%	0%	100%	-	89%	11%	0%	11%	**19.3%**	Muy Bajo
	SCA(5)_Uni2Opt	0%	0%	40%	0%	0%	0%	0%	100%	33%	-	22%	11%	22%	**19.1%**	Muy Bajo
	SCA(1)_Uni2Opt	0%	0%	20%	0%	0%	0%	0%	-	11%	44%	0%	0%	0%	**6.3%**	Muy Bajo

Nota. Se realizaron los cálculos en base a las instancias en común entre cada para de algoritmos a comparar, según la tabla 15.

A partir de la Tabla 19, los valores encontrados se interpretan de la siguiente manera:

- Al comparar dos algoritmos en específico, con las mismas instancias en común se concluyó que el nivel de eficacia:
    - Del SCA(1)_Uni2Opt_r con respecto al SCA(5)_Uni2Opt alcanzó un nivel de eficacia de 89%.
    - Por ejemplo, el ACS con respecto al IGA alcanzó un nivel de eficacia de 25%.
- Al comparar el nivel de eficacia promedio, se obtuvo:
    - El algoritmo ACS_3Opt alcanzó el mayor nivel de eficacia promedio con 100%, lo cual significa que en todas las instancias en las que se comparó con los otros algoritmos, el ACS_3Opt obtuvo los mejores resultados.

Conclusión de la Hipótesis Específica 3:

- El nivel de eficacia de las versiones Heavy varió según la metaheurística a comparar. En específico, 2 de las versiones Heavy, SCA(1)_Multi2Opt_r y SCA(5)_Multi2Opt, lograron presentar un nivel de eficacia promedio superior al de las metaheurísticas AG y SWAP_GATSP (ver Tabla 19), pero inferior al resto de metaheurísticas a comparar.
  Por lo cual, para las **versiones Heavy se cumple de manera parcial la hipótesis específica 3**.
- El nivel de eficacia de las versiones Soft, ninguna de sus 3 versiones logró superar el nivel de eficacia promedio al compararse con las metaheurísticas seleccionadas (ver Tabla 19).
  Por lo cual, para las **versiones Soft no se cumple la hipótesis específica 3**.

4.3. Discusión de Resultados

a. Con respecto a la investigación de Tito, Silva, Alfaro y Fajardo (2015), el algoritmo de Colonia de Hormigas (ACO) que implementaron para resolver la instancia Berlin52, de la librería virtual TSPLIB, presentó una solución cuyo Error Relativo alcanzó un valor de 1.8%. Los resultados del Error Relativo del resto de metaheurísticas se presentan en la Tabla 20.

Tabla 20

Error relativo de versiones Heavy, Soft y metaheurística ACO

Tipo de Algoritmo	Algoritmo Utilizado	Solución	Error Relativo
Solución TSPLIB	Óptimo	7542	0.00%
Metaheurística	ACO	7680	1.80%
HEAVY	SCA(1)_ Multi2Opt	7715	2.30%
	SCA(1)_ Multi2Opt_r	7715	2.30%
	SCA(5)_ Multi2Opt	7657	1.50%
SOFT	SCA(1)_ Uni2Opt	7793	3.33%
	SCA(1)_ Uni2Opt_r	7792	3.31%
	SCA(5)_ Uni2Opt	7793	3.33%

Nota. ACO de Tito, Silva, Alfaro y Fajardo (2015) para instancia Berlin52

De la Tabla 20, solo el algoritmo SCA(5)_Multi2Opt logró superar la eficacia del ACO para la instancia Berlin52 (1.5% < 1.8%). Por lo cual, el nivel de eficacia (para la única instancia revisada) del SCA(5)_Multi2Opt resultó de 1 o 100%.

b. Las versiones Heavy y Soft, con respecto a la investigación de Pérez (2011), SCA_2Opt y SCA_2Opt_r, el rango del *Error Relativo* para las mismas trece instancias seleccionadas (para ver el detalle de las

instancias, ver Tabla 11) en la presente investigación alcanzó un valor que varió según el algoritmo seleccionado.

Con respecto al SCA_2Opt de Pérez (2011):

- Para el SCA_2Opt, entre un 2.7% a 10%; para las versiones Heavy se tuvo un valor entre 0% a 3.9% y para las versiones Soft un valor entre 0% a 9.5%.
    - El nivel de eficacia de las tres versiones Heavy fue de 100% con respecto al SCA_2Opt (Ver Tabla 13). Es decir, del total de instancias revisadas, las tres versiones Heavy superaron a las soluciones del SCA_2Opt.
    - El nivel de eficacia de las versiones Soft varió entre un 69% a 85% con respecto al SCA_2Opt (Ver Tabla 13). Es decir, del total de instancias revisadas, las tres versiones Soft superaron al SCA_2Opt en más del 69%.

Con respecto al SCA_2Opt_r de Pérez (2011):

- Para el SCA_2Opt_r, entre un 0.9% a un 5.6%; para las versiones Heavy se tuvo un valor entre 0% a 3.9% y para las versiones Soft un valor entre 0% a 9.5%.
    - El nivel de eficacia de las tres versiones Heavy fue de 100% con respecto al SCA_2Opt_r (Ver Tabla 13). Es decir, del total de instancias revisadas las tres versiones Heavy superaron a las soluciones del SCA_2Opt_r.
    - El nivel de eficacia de las tres versiones Soft fue de 31%, 69% y 54% para los algoritmos SCA(1)_Uni2Opt, SCA(1)_Uni2Opt_r y SCA(5)_Uni2Opt respectivamente, en relación con el SCA_2Opt_r (Ver Tabla 13). Es decir, del total de instancias revisadas los algoritmos SCA(1)_Uni2Opt_r y SCA(5)_Uni2Opt lograron encontrar, al menos en la mitad de las instancias, mejores soluciones que las del SCA_2Opt_r.

c. Con respecto a la investigación de Mendoza (2017), dividió el análisis de las instancias según el tamaño del problema, tal que, para las instancias revisadas, en su investigación, de pequeño y mediano (hasta 175 nodos), al comparar las soluciones obtenidas por nuestros algoritmos con la heurística del Vecino más Cercano, Búsqueda Tabú, Scatter Search, GRASP, algoritmos del WINQSB (Branch and Bound) y Solver (Simplex) de Excel, se concluyó que nuestros algoritmos resultaron encontrar mejores soluciones (más próximas a las óptimas) para instancias de tamaño pequeño: de 17 hasta 48 nodos (Ver Tabla 22 del Anexo 8).

Antes de indicar la conclusión para las instancias de tamaño mediano y grande, debemos señalar que Gerhard (s.f) en su documentación acerca de Frequent Questions and Answers (FAQ) indica que las soluciones reportadas en TSPLIB son óptimas, demostrables por algoritmos de “Branch and Bound” y “Branch and Cut”. Por lo tanto, toda solución que indique superar a la óptima de TSPLIB, lo más probable es que no haya sido calculada correctamente. Por tal motivo, al analizar las instancias medianas y grandes (101 a 400 nodos), en las soluciones reportadas por Mendoza (2017), consolidadas en la Tabla 23 del Anexo 8, se observó que no podemos realizar una comparación correcta por tener una cantidad de 22 de 30 instancias que reporta Mendoza haber superado al Óptimo TSPLIB. Lo que sí podemos afirmar es que el *Error Relativo* de nuestras soluciones oscila entre el 0% al 3.71% para las 30 instancias seleccionadas.

d. Con respecto a la investigación de Akhand M., A. H.; Ayon Safial Islam; Shahriyar, S. A.; Siddique, N.; Adeli, H (2020), quienes realizaron una comparación del rendimiento de cuatro algoritmos metaheurísticos para un total de 45 instancias de TSPLIB, de las cuales, para comparar con nuestros tres algoritmos, seleccionamos aquellos cuyo método de cálculo de distancias haya consistido en la distancia euclidiana, quedando un total de 30 instancias (para revisar el detalle de las instancias, ver Tabla 24, Anexo 9). A partir de las cuales se calculó su

nivel de eficacia, el cual es mostrado en la Tabla 21, en la cual se adicionó un ranking según el nivel de eficacia alcanzado.

Tabla 21

Nivel de eficacia entre las versiones Heavy, Soft y metaheurísticas ACO, VTSPO, ABCSS y DSMO

Algoritmo	Nivel Eficacia	Ranking
ACO	0/30	7mo lugar
VTPSO	2/30	5to lugar
ABCSS	4/30	4to lugar
DSMO	9/30	2do lugar
SCA(1)_Multi2Opt	1/30	6to lugar
SCA(1)_Multi2Opt_r	12/30	1er lugar
SCA(5)_Multi2Opt	6/30	3er lugar
SCA(1)_Uni2Opt	0/30	7mo lugar
SCA(1)_Uni2Opt_r	0/30	7mo lugar
SCA(5)_Uni2Opt	1/30	6to lugar

Nota. ACO, VTSPO, ABCSS y DSMO tomados de Akhand, Ayon, Shahriyar, Siddique y Adeli (2020)

A partir de la Tabla 21, tenemos que de las soluciones obtenidas por el total de diez algoritmos comparados se puede apreciar que:

- El SCA(1)_Multi2Opt y el SCA(5)_Uni2Opt presentaron un Nivel de Eficacia bajo, quedando en 6to lugar.
- El SCA(1)_Multi2Opt_r presentó un Nivel de Eficacia alto, quedando en 1er lugar.
- El SCA(5)_Multi2Opt presentó un Nivel de Eficacia medio, quedando en 3er lugar.
- El SCA(1)_Uni2Opt y SCA(1)_Uni2Opt_r presentaron un nivel de eficacia muy bajo junto con la metaheurística ACO.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1. CONCLUSIONES

A continuación, se presentan las conclusiones de la presente investigación.

Con respecto a la versión SCA_2Opt de Pérez (2011):

- El Nivel de Eficacia de las tres versiones Heavy, al ser comparadas con el SCA_2Opt, logró alcanzar un valor de 100%, es decir, que en cada una de las trece instancias simétricas TSPLIB revisadas, las soluciones de nuestras tres versiones Heavy resultaron mejores a las brindadas por el SCA_2Opt.

- Con respecto a las versiones Soft, su Nivel de Eficacia varió entre 69% a 85% con respecto al SCA_2Opt (ver Tabla 13), las cuales presentaron un mayor Nivel de Eficacia comparados al SCA_2Opt.

Por lo cual, se cumplió que tanto para las versiones Heavy y Soft, su Nivel de Eficacia superó al del SCA_2Opt.

Con respecto a la versión SCA_2Opt_r de Pérez (2011):

- El Nivel de Eficacia de las tres versiones Heavy, al ser comparadas con el SCA_2Opt_r, lograron alcanzar un valor de 100%, es decir, que en cada una de las trece instancias TSPLIB revisadas, las

soluciones de nuestras tres versiones resultaron mejores a las obtenidas por el SCA_2Opt_r; mientras que las versiones Soft solo alcanzaron un Nivel de Eficacia entre el 31% al 69% según la versión Soft seleccionada (ver Tabla 13), logrando solo el SCA(5)_Uni2Opt y el SCA(1)_Uni2Opt_r superar el Nivel de Eficacia del SCA_2Opt_r de Pérez (2011).

Por lo cual, se cumplió que las versiones Heavy presentaron un Nivel de Eficacia superior al SCA_2Opt_r, mientras que para las versiones Soft, su Nivel de Eficacia solo superó en dos de tres algoritmos al SCA_2Opt_r.

Con respecto a las metaheurísticas seleccionadas en Pérez (2011):

- El Nivel de Eficacia de cada una de las tres versiones Heavy varió en un rango de 0% a 100%, según la metaheurística con la cual se comparó (ver Tabla 16). Mientras que, para las versiones Soft, los niveles de eficacia para las instancias comparadas alcanzaron un nivel Bajo, de 0 a 40% (ver Tabla 17), al compararlas con las metaheurísticas seleccionadas. Por lo cual, se puede concluir que las versiones Heavy presentan un mayor Nivel de Eficacia que las versiones Soft, el cual se aproxima mejor al Nivel de Eficacia de las metaheurísticas seleccionadas.

  Se hace hincapié en que los rangos del Nivel de Eficacia fueron calculados en base a las instancias con las cuales se realizó la comparación, por lo cual, estos pueden variar en otras investigaciones al reemplazar o incrementar la cantidad de instancias, o al utilizar variantes del mismo algoritmo metaheurístico.

5.2. RECOMENDACIONES

A continuación, se presentan las recomendaciones para la presente investigación en búsqueda de mejorar aún más el Nivel de Eficacia:

- Se propone explorar la implementación de la heurística de mejora 3Opt tanto para las versiones Heavy como Soft, aunque esto podría elevar el costo computacional. Por ello, se debe evaluar si los resultados justifican tal implementación.

- Se propone hacer variable el factor “r” según intervalos de iteraciones, en búsqueda de modificar la trayectoria de búsqueda de soluciones.

- Para no realizar un sobreesfuerzo computacional, se plantea que el criterio de parada consista en detener el algoritmo cuando la solución no mejora en una determinada cantidad de ejecuciones sucesivas. Al explorar este punto, se evidenció que, para la mayoría de las instancias seleccionadas (hasta alrededor de 700 nodos), con una cantidad de 30 ejecuciones sucesivas sin mejorar era suficiente para la convergencia de los algoritmos.

- Inicialmente se introdujo el factor “r”, expresado en forma lineal por “(1 + r)”, al criterio de intercambio 2Opt para generar una pequeña, pero significativa, perturbación en la habilitación de los Intercambios 2Opt, pero se pueden evaluar otras formas de realizar tal perturbación. Por ejemplo, utilizar expresiones exponenciales.

- Se recomienda estudiar la eficacia y el Nivel de Eficacia de la versión del algoritmo con el Vecino más Cercano y la heurística 2Opt_r; es decir: VMC_2Opt_r, el cual fue mencionado y descrito de forma general en el Anexo 27, con su diagrama de flujo en la Figura 82. Donde, el VMC_2Opt_r es resultado de la presente investigación y se agregó como anexo por no incluir el SCA.

REFERENCIAS

Akhand M., A. H., Ayon, S. I., Shahriyar, S. A., Siddique, N., & Adeli, H. (2020). *Discrete Spider Monkey Optimization for Traveling Salesman Problem.* https://pure.ulster.ac.uk/files/78147609/Accepted_Vers_NSiddique_DSMO_TSP_Revised_Manuscript_ASOC_D_19_00291_R2_190825.pdf

Alancay, N., Villagra, S. M., & Villagra, N. A. (2016a). Metaheurísticas de trayectoria y poblacional aplicadas a problemas de optimización combinatoria. *Informes Científicos Técnicos, 8*(1), 202-220. https://doi.org/10.22305/ict-unpa.v8i1.157

Alancay, N., Villagra, S., & Villagra, N. A. (2016b). Algoritmos metaheurísticos trayectoriales para optimizar problemas combinatorios. *Informe Científico Técnico-UNPA. 8*(3), 56-75. https://dialnet.unirioja.es/descarga/articulo/5761753.pdf

Alemayehu, T., & Kim, J. (2017). Efficient nearest neighbor heuristic TSP algorithms for reducing data acquisition latency of UAV relay WSN. *Wireless Personal Communications, 95*(3), 3271-3285. doi:10.1007/s11277-017-3994-9

Arbaiza Fermini, L. (2014). *Como elaborar una Tesis de Grado.* Lima: Universidad ESAN: ESAN Ediciones.

Astoquillca Yaranga, D. J., & Berger Vidal, E. (2023). Heurística de Intercambio 2Opt Best Improvement_r y Nivel de Eficacia de las soluciones del Problema del Agente Viajero Simétrico. *Revista Peruana de Computación y Sistemas*. https://doi.org/10.15381/rpcs.v5i1.25806

Ayala de la Vega, J., Aguilar Juárez, I., García Lamont, F., & Gómez Ayala, H. (2019). Introducción al Análisis de Algoritmos. http://ri.uaemex.mx/bitstream/handle/20.500.11799/105198/Libro%20Complejidad.pdf?sequence=1&isAllowed=y

Ayala de la Vega, J., Aguilar Juárez, I., García Lamont, F., & Gómez Ayala, H. (2019). Introducción al Análisis de Algoritmos [Tabla]. http://ri.uaemex.mx/bitstream/handle/20.500.11799/105198/Libro%20Complejidad.pdf?sequence=1&isAllowed=y

Bautista, J., & Pereira, J. (2003). *Adaptación de la Heurística LIN-KERNINGHAM para la resolución de problemas de diseño de itinerarios.* http://www.nissanchair.com/pubs/seio_lleida_2.pdf

Bonomo, F. (2016). *Clases de grafos y problemas de optimización combinatoria [Diapositiva de Power Point].* https://www.matematica.uns.edu.ar/uma2016/material/curso%20UMA%202016_fbonomo_handout.pdf

Campo Zuñiga, B., & Mendoza Mendoza, A. (2018). Propuesta de un modelo de ruteo de vehículos abierto en una institución prestadora de servicios de salud. *Entramado, 14*(2), 288-298. http://www.scielo.org.co/pdf/entra/v14n2/1900-3803-entra-14-02-288.pdf

Casaño, C. D. (2016). Metodología de la investigación tecnológica en ingeniería. *Ingenium, 1*(1), 43-46. http://journals.continental.edu.pe/index.php/ingenium/article/view/392/394

Chalco Chipayo, H. H. (2018). *Sistema de optimización de rutas para el levantamiento catastral de infraestructura en telecomunicaciones.* https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/14248/Chalco_chh.pdf?sequence=1&isAllowed=y

Chávez , R. (2001). *Marco metodológico.* http://virtual.urbe.edu/tesispub/0059916/cap03.pdf

Comisión Nacional de Investigación Científica y Tecnológica de Chile. (2008). *Conceptos básicos de Ciencia Tecnología e Innovación.* https://www.conicyt.cl/wp-content/uploads/2012/09/Conceptos-B%C3%A1sicos-de-Ciencia-Tecnolog%C3%ADa-e-Innovaci%C3%B3n-2008.pdf

Cortéz, A. (2004). Teoría de la Complejidad Computacional y teoría de la Computabilidad. *Revista de Investigación de Sistemas e Informática. 1*(1), 102-105. https://sisbib.unmsm.edu.pe/bibvirtualdata/publicaciones/risi/N1_2004/a14.pdf

Coto, E. (2003). Algoritmos Básicos de Grafos. *Lecturas en Ciencias de computación. ISSN 1316, 6239*.

Da Costa, P., Rhuggenaath, J., Zhang, Y., & Akcay, A. (2020). Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In *Asian conference on machine learning*. 465-480. https://proceedings.mlr.press/v129/costa20a/costa20a.pdf

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation, 1*(1), 53-66. https://doi.org/10.1109/4235.585892

Eppen, G., Gould, F., Schmidt, C., Moore, J., & Weatherford, L. (2000). Toma de decisiones con objetivos múltiples y heurística. En *Investigación de Operaciones en la Ciencia Administrativa* (pp.400-401). *México: Pearson Educación*. https://jrvargas.files.wordpress.com/2009/01/investigacic3b3n-de-operaciones-en-la-ciencia-administrativa-5ta-edicic3b3n.pdf

Espinoza Veintimilla, A. (2010). *Estudio sobre el aprendizaje organizacional y la administración del conocimiento en las empresas basado en el pensamiento sistémico.* https://dspace.ucuenca.edu.ec/bitstream/123456789/2527/1/tm4383.pdf

Esteban Nieto, N. (2018). Tipos de Investigación. https://core.ac.uk/download/pdf/250080756.pdf

Flood, M. M. (1956). The Traveling-Salesman Problem. *Operations Research, 4(1)*, 61-75. https://pubsonline.informs.org/doi/abs/10.1287/opre.4.1.61#d46761e1

Guerequeta, R., & Vallecillo, A. (2019). *Técnicas de diseño de algoritmos.* http://190.57.147.202:90/jspui/bitstream/123456789/451/1/tecnicas_de_diseno_de_algoritmos.pdf

Gutiérrez Espinosa, I. (2018). *Uso de algoritmos heurísticos para el diseño de rutas de recolección de residuos sólidos urbanos* [Tesis de Maestría, Universidad Politécnica de Tulancingo]. http://www.upt.edu.mx/Contenido/Investigacion/Contenido/TESIS/MOP/2018/MOP_T_2018_01_IGE.pdf

Hougardy, S., Zaiser, F., & Zhong, X. (2020). The approximation ratio of the 2-Opt Heuristic for the metric Traveling Salesman Problem. *Operations Research Letters, 48*(4), 401-404.

Kizilates, G., & Nuriyeva, F. (2013). A parametric hybrid method for the traveling salesman problem. *Mathematical and Computational Applications, 18*(3), 459-466. doi:10.3390/mca18030459

López Guevara, R. (2009). Minimización del ancho de banda de matrices simétricas dispersas aplicando algoritmos genéticos.

López, E., Salas, Ó., & Murillo, Á. (2014). El Problema del Agente Viajero: Un Algoritmo determinístico usando Búsqueda Tabú. *Revista de Matemática Teoría y Aplicaciones, 21*(1), 127-144. https://www.scielo.sa.cr/pdf/rmta/v21n1/a08v21n1.pdf

Maldonado, C. E. (2013). Un problema fundamental en la investigación: Los problemas P vs. NP. *Revista LOGOS CIENCIA & TECNOLOGÍA. 4*(2), 10-20. https://www.redalyc.org/pdf/5177/517751544002.pdf

Marcelo Villalobos, R. W. (2014). *Fundamentos de programación Java. Más de 100 algoritmos codificados.(2da Edición).* Editorial Macro.

Martí, R. (2003). Procedimientos Metaheurísticos en Optimización Combinatoria. *Matemátiques. 1*(1), 1-60. https://www.uv.es/rmarti/paper/docs/heur1.pdf

Martí, R. (2003). Procedimientos Metaheurísticos en Optimización Combinatoria. *[Figura].* https://www.uv.es/rmarti/paper/docs/heur1.pdf

Martín Novo, E., & Méndez Alonso, A. (2004). Aplicaciones de la teoría de grafos a algunos juegos de estrategia. *Suma*, 31-35. https://redined.educacion.gob.es/xmlui/bitstream/handle/11162/13874/031-035.pdf?sequence=1&isAllowed=y

Matai, R., Singh, S. P., & Mittal, M. L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. *Traveling salesman problem, theory and applications, 1*(1), 1-25. https://books.google.es/books?hl=es&lr=&id=gKWdDwAAQBAJ&oi=fnd&pg=PA1&dq=Applications+of+the+TSP&ots=aa8tZ21iJ1&sig=kMLzJDfL1W9jRDQOOGOVSKY9zWQ#v=onepage&q=Applications%20of%20the%20TSP&f=false

Medina, L. B., La Rota, E. C., & Castro, J. A. (2004). Una revisión al estado del arte del problema de ruteo de vehículos: Evolución histórica y métodos de solución. *Ingeniería, 16*(2), 35-55. https://www.colibri.udelar.edu.uy/jspui/bitstream/20.500.12008/3508/1/TR0408.pdf

Melo Ospina, J. A. (2011). *Una métrica sobre grafos ponderados como medida de similitud molecular* [Tesis de Maestría, Universidad Nacional de Colombia]. https://repositorio.unal.edu.co/bitstream/handle/unal/10867/jennyadrianameloospina.2011.pdf?sequence=1&isAllowed=y

Mendoza Casanova, J. J. (2017). *Resumen Traveling Salesman Problem. Diseño de Algoritmos Heurísticos y Metaheurísticos eficientes para resolver el Problema del Agente Viajero* [Tesis doctoral, Universidad Nacional Autónoma de Nicaragua] https://repositorio.unan.edu.ni/8827/1/02-Resumen-Tesis%20Doctoral.pdf

Menéndez Velázquez, A. (1998). Una breve introdución a la teoría de grafos. *Suma*, 11-26.

https://redined.educacion.gob.es/xmlui/bitstream/handle/11162/13526/011-026.pdf?sequence=1

Merriam-Webster. (s. f.). *Iteration*. En Merriam-Webster.com Dictionary. https://www.merriam-webster.com/dictionary/iteration

Ministerio de Transporte y Comunicaciones del Perú. (12 de Enero de 2018). *Glosario de Términos de Uso frecuente en proyectos de Infraestructura Vial.* http://transparencia.mtc.gob.pe/idm_docs/normas_legales/1_0_4032.pdf

Morales Silva, O. M. (2012). *Modelo de Ruteo de Vehículos* [Trabajo de Grado, Universidad EAN]. https://repository.ean.edu.co/bitstream/handle/10882/1525/MoralesOscar2012.pdf;jsessionid=5198E302B9BC4E87C877C601AA716DB1?sequence=7

Oré Lujan, J. C. (2010). *Optimización de Sistemas de atención al cliente mediante el uso de algoritmos genéticos* [Tesis de Maestría, Universidad Nacional Mayor de San Marcos]

Pérez Rave, J. I. (2011). *Heurística inspirada en el análisis sistémico del "Vecino más cercano", para solucionar instancias simétricas TSP, empleando una base comparativa multicriterio* [Tesis de Maestria, Universidad Nacional de Colombia]. https://core.ac.uk/download/pdf/11055302.pdf

Pérez Rave, J. I. (2011). *Heurística inspirada en el análisis sistémico del "Vecino más cercano", para solucionar instancias simétricas TSP, empleando una base comparativa multicriterio [Figura].* https://repositorio.unal.edu.co/bitstream/handle/unal/8756/71225056.2011.pdf?sequence=1&isAllowed=y

Pérez Rave, J. I., Jaramillo Álvarez, G. P., Parra Mesa, C. M., & Moreno Velásquez, L. F. (2010). TETRAHEURÍSTICA SISTÉMICA (THS)

PARA EL TSP. *Ingeniare. 18*(2), 187-202. https://scielo.conicyt.cl/pdf/ingeniare/v18n2/art06.pdf

Pérez Rave, J., & Jaramillo Álvarez, P. (2014). Sacrificio cortoplacista adaptativo 2opt (SCA_2opt): Una heurística inspirada en el pensamiento sistémico. *Production. 24*(1), 26-40. https://doi.org/10.1590/S0103-65132013005000033

Pimentel, F. G. (2011). Double-ended nearest and loneliest neighbour–a nearest neighbour heuristic variation for the travelling salesman problem. *Revista de Ciências da Computação, 6*, 17-30. https://www.researchgate.net/profile/Fernando-Pimentel-4/publication/258241208_Double-ended_nearest_and_loneliest_neighbour-a_nearest_neighbour_heuristic_variation_for_the_travelling_salesman_problem/links/004635278d16584dcc000000/Double-ended-nearest-and

Poveda Ch., R., Gómez P., J., & León G., E. (2008). GRISLAS: Un algoritmo genético paralelo que combina los modelos de grillas e islas para encontrar soluciones óptimas cercanas al problema del agente viajero. *Revista Avances en Sistemas e Informática, 5*(3), 13-19.

Quiñones Reyes, A. J., Villar Tiravantti, L. M., & Gutiérrez Ascón, J. E. (2017). Algoritmo del agente viajero para minimizar el recorrido de visitas programadas en una empresa de seguros. *INGENIERÍA: Ciencia, Tecnología e Innovación. 4*(2). https://revistas.uss.edu.pe/index.php/ING/article/view/728/634

Rao, W., & Jin, C. (2010). A new hybrid algorithm for solving TSP. Paper presented at the ICLEM 2010:. *Logistics for Sustained Economic Development - Infrastructure, Information, Integration - Proceedings of the 2010 International Conference of Logistics Engineering and Management*, 387, 3327-3334. doi:10.1061/41139(387)464.

Ray, S. S., Bandyopadhyay, S., & Pal, S. K. (2004). An Improved Genetic Algorithm to Solve the Traveling Salesman. *In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. 2*, 497-500. https://paperzz.com/doc/7075250/an-improved-genetic-algorithm-to-solve-the-traveling-sale...

Reinelt, G. (1995). TSPLIB 95. *Interdisziplinäres Zentrum für Wissenschaftliches Rechnen (IWR), Heidelberg*, 338, 1-16. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp95.pdf

Reinelt, G. (s.f.). *TSPLIB: Questions and Answers.* http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/TSPFAQ.html

Rodríguez Jiménez, A., & Pérez Jacinto, A. O. (2017). Métodos científicos de indagación y de construcción del conocimiento. *Revista EAN, (82)*, 179-200.http://www.scielo.org.co/pdf/ean/n82/0120-8160-ean-82-00179.pdf

Sallabi, O. M., & El-Haddad, Y. (2009). An Improved Genetic Algorithm to Solve the Traveling Salesman Problem. *World Academy of Science, Engineering and Technology*, 471-474. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.4146&rep=rep1&type=pdf

Senge, P. (2010). La Quinta Disciplina: El arte y la práctica de la organización abierta al aprendizaje. https://clea.edu.mx/biblioteca/files/original/94ce0359b5ac5efaccca3a941d41d396.pdf

Tejero, E. L. (2020). Algoritmos. El totalitarismo determinista que se avecina. ¿La pérdida final de libertad?. *Revista de Pensamiento Estratégico y Seguridad CISDE, 5*(1), 85-101. http://uajournals.com/ojs/index.php/cisdejournal/article/view/620/406#

Tito Chura, H. E., Silva Delgado, C. A., Alfaro Gonzales, E. E., & Fajardo Espinoza, E. (2019). Aplicación de Algoritmo de Colonia de Hormigas al problema del agente viajero. *Revista Ciencia & Desarrollo (20)*, 98-102. https://revistas.unjbg.edu.pe/index.php/cyd/article/view/521/516

Tocto Inga, P. M. (2012). *Comparación de los Algoritmos PRIM y KRUSKAL.* [Tesis de Licenciatura, Universidad Nacional de Ingeniería]. https://web.archive.org/web/20180503200539id_/http://cybertesis.uni.edu.pe/bitstream/uni/3416/1/tocto_ip.pdf

Vélez, M. C., & Montoya, J. A. (2007). Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones. *Revista EIA, (8)*, 99-115. http://www.scielo.org.co/pdf/eia/n8/n8a09.pdf

Zuñiga Vargas, L. (2016). *Un algoritmo heurístico para el coloreo de grafos.* [tesis de licenciatura, Benemérita Universidad Autónoma de Puebla] https://repositorioinstitucional.buap.mx/handle/20.500.12371/12969

ANEXOS

Anexo 1. Matriz de Consistencia

TITULO: Modificaciones del algoritmo Sacrificio Cortoplacista Adaptativo 2Opt_r para solucionar el Problema del Agente Viajero Simétrico: Comparación del Nivel de Eficacia

PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	VARIABLES	METODOLOGÍA
¿Qué diseño deben tener los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que con los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas?	Determinar el diseño de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas	Mediante los algoritmos propuestos será posible resolver el Problema del Agente Viajero Simétrico con mayor Nivel de Eficacia que con los algoritmos del Sacrificio Cortoplacista Adaptativo 2Opt, Sacrificio Cortoplacista Adaptativo 2Opt_r y metaheurísticas seleccionadas	**Variable Independiente** Heurística Híbrida del Vecino más Cercano con Sacrificio Cortoplacista 2Opt_r. **Variable Dependiente** Solución del Problema del Agente Viajero Simétrico.	**TIPO DE INVESTIGACIÓN** Tecnológica. **Enfoque:** Cuantitativo. **DISEÑO DE INVESTIGACIÓN** No experimental. **MÉTODO DE INVESTIGACIÓN** Método Análisis-Síntesis y Método de Analogías, con enfoque sistémico.
PROBLEMAS ESPECÍFICOS	**OBJETIVOS ESPECÍFICOS**	**HIPÓTESIS ESPECÍFICAS**	**INDICADORES**	**POBLACIÓN, MUESTRA E INSTRUMENTOS**
¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt?	Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt	El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt	**Eficacia** de las soluciones obtenidas por los algoritmos desarrollados vs Eficacia de los algoritmos metaheurísticos seleccionados con respecto al óptimo reportado en TSPLIB. **Nivel de Eficacia** de soluciones obtenidas por los algoritmos desarrollados en comparación con los algoritmos metaheurísticos seleccionados para las mismas instancias evaluadas.	**Unidad de Análisis:** La unidad de análisis consistió en un algoritmo que resuelve el Problema del Agente Viajero Simétrico. **Población de estudio:** El total de algoritmos heurísticos y metaheurísticos conocidos para resolver el Problema del Agente Viajero Simétrico.

PROBLEMAS ESPECÍFICOS	OBJETIVOS ESPECÍFICOS	HIPÓTESIS ESPECÍFICAS	INDICADORES	POBLACIÓN, MUESTRA E INSTRUMENTOS
¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del Sacrificio Cortoplacista Adaptativo 2Opt_r?	Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r.	El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia del algoritmo del Sacrificio Cortoplacista Adaptativo 2Opt_r	**Eficacia** Error Relativo (Eficacia) = **[**Solución Alcanzada por el Algoritmo evaluado - Solución Óptima TSPLIB**]** / **[**Solución Óptima TSPLIB**]** **Nivel de Eficacia** **[** Cantidad de instancias donde el algoritmo seleccionado obtuvo la mejor eficacia **]** / **[** Cantidad total de soluciones disponibles del total de algoritmos seleccionados **]** **Medición de Nivel de Eficacia** Se procederá a medir su Nivel de Eficacia en comparación con la del Pérez (2011) y las metaheurísticas seleccionadas para instancias seleccionadas del problema del agente viajero simétrico ubicadas en librería virtual TSPLIB.	**Tamaño de muestra:** Se tomó un total de nueve algoritmos. **Selección de muestra:** No aleatoria, por conveniencia: Dos heurísticos basados en el Sacrificio Cortoplacista desarrollado por Pérez (2011), cuatro metaheurísticas basadas en Algoritmos Genéticos, dos metaheurísticas basadas en Colonia de Hormigas (ACS), 1 metaheurística basada en Programación Evolutiva. **Técnicas de Recolección de Datos** Revisión de data pública publicada en TSPLIB **Instrumentos** Matriz de Costos
¿Cómo será el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia de las metaheurísticas seleccionadas?	Determinar el Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico, comparado con el Nivel de Eficacia de las metaheurísticas seleccionadas	El Nivel de Eficacia de los algoritmos propuestos para resolver el Problema del Agente Viajero Simétrico será mayor que el Nivel de Eficacia de las metaheurísticas seleccionadas		**Construcción de Matriz de Costos** 1. Acceder al link de la librería virtual TSPLIB y descargar los archivos requeridos según filtros definidos. 2. Cálculo de elementos de la Matriz de Costos de cada instancia descargada acorde con el método de cálculo indicado en TSPLIB.

Anexo 2. Operacionalización de Variables

Variable	Tipo de Variable	Definición Conceptual	Definición Operacional	Dimensiones	Indicadores	Definición de Indicadores	Unidades
Heurística Híbrida del Vecino más Cercano con Sacrificio Cortoplacista 2Opt_r	Independiente	Pérez (2011) introduce el término Sacrificio Cortoplacista como una variante del Vecino más Cercano, en la cual desde una ciudad se decide ir a la segunda más cercana con la esperanza de habilitar arcos que antes estaban prohibidos y construir una mejor ruta. El término 2Opt hace referencia al intercambio de arcos 2Opt	**Observar**. Se presenta el algoritmo, así como su criterio para construcción de soluciones. **Determinar el Problema.** Establecer una desventaja del algoritmo **Documentar.** Listar variantes del Vecino más Cercano. **Reflexionar.** Genera una solución a la "Determinación del Problema". **Construcción de los algoritmos.** Descripción de proceso para realizar la modificación del algoritmo. **Implementación de los algoritmos desarrollados.** Determinación del entorno de programación en el cual se ejecuta el algoritmo. **Descripción de los Algoritmos desarrollados** Descripción de proceso de búsqueda, criterio de parada **Comparación y Evaluación de Eficacia.** Medir eficacia de los algoritmos propuestos y determinar si la modificación logra mejorar la eficacia del algoritmo original.	Eficacia Algorítmica	**Error Relativo** Permite evaluar cuan cerca se encuentra la solución encontrada por nuestro algoritmo a la solución óptima.	**Eficacia Algorítmica (Error Relativo)** **= [**Solución Alcanzada por el Algoritmo evaluado - Solución óptima TSPLIB**] / [**Solución Óptima TSPLIB**]**	%

Variable	Tipo de Variable	Definición Conceptual	Definición Operacional	Dimensiones	Indicadores	Definición de Indicadores	Unidades
Solución del Problema del Agente Viajero Simétrico	Dependiente	Dado un conjunto de “n” ciudades, de las cuales se conoce para cada par de ellas, la distancia que las separa, un agente viajero ha de partir de una ciudad de origen y debe visitar exactamente una vez cada ciudad del conjunto, y retornar al punto de partida. (Mendoza, 2017, p.5)	Búsqueda de investigaciones con algoritmos heurísticos y metaheurísticos Filtro de investigaciones donde se utilizó la librería TSPLIB para validar la eficacia de los algoritmos implementados. Selección de investigaciones que cumplan filtros definidos para la investigación. Medición de la Eficacia de los algoritmos para las instancias seleccionadas. Medición del Nivel de Eficacia de los algoritmos propuestos vs los algoritmos desarrollados en las investigaciones seleccionadas con respecto a las instancias seleccionadas del problema de agente viajero simétrico obtenidas de TSPLIB	Nivel de Eficacia Algorítmica	Proporción de la cantidad de instancias revisadas donde el algoritmo obtuvo la mejor eficacia	**Nivel de Eficacia = [** Cantidad de instancias donde el algoritmo evaluado obtuvo la mejor eficacia **] / [** Total de instancias seleccionadas**]**	%

Anexo 3. Selección de metaheurísticas para comparar

a. Para las metaheurísticas basadas en Algoritmos Genéticos. Se presentan los resultados consolidados en la investigación de Pérez (2011) en la Figura 71.

Figura 71

Grupo de Algoritmos Genéticos tomados de Pérez (2011)

Procesadores				Pent 4, 3,06 GHz								Intel Core 2 Duo 2,33 GHz						
Nº	País			Colombia		Bélgica y Suiza [18]	Brasil [21]	Taiwan [22]			Brasil [23]			Libia [24]			
	Instancia TSP	Valor r	Variab	SCA_2opt_r	SCA_2op	GA	AGIV	SGA	MDDCGA	DDCGA	AG serial	AG paralelo	AG grupo	FRAG_GA	SWAP GATSP	OX_SIM	IGA
1	Swiss42	0,05	CM	1.285	1.375		1.343,6										
			MC	1.285	1.375		ND										
			TM	(0,078)	(0,063)		(ND)										
			NT	[72]	[61]		[ND]										
2	Eil76	0,25	CM	559,61	567,01						591	590	589				
			MC	559,61	567,01						ND	ND	ND				
			TM	(1,40)	(0,352)						(1.390)	(790)	(750)				
			CPC	4,28	1,08						3238,7	1840,7	1747,5				
			NT	[964]	[118]						[ND]	[ND]	[ND]				
3	KroA100	0,16	CM	22.219,3	22.379,4	ND		27.117,9	26.564,6	25.739,6				21.350	21.900	22.670	21.294,5
			MC	22.219,3	22.379,4	21.761		25.347,5	25.093,1	23.924,1				21.282	21.504	22.400	21.282
			TM	(1,07)	(0,875)	(ND)		(ND)	(ND)	(ND)				(ND)	(ND)	(ND)	(ND)
			NT	[206]	[140]	[103.000]		[ND]	[ND]	[ND]				[800]	[5.000	[25.000	[454]
4	KroA150	0,19	CM	27.999,7	29.076,9			33.495,7	33.130,3	32.737,0							
			MC	27.999,7	29.076,9			31.194,2	30.338,7	30.700,1							
			TM	(9,78)	(2,89)			(ND)	(ND)	(ND)							
			NT	[1.023]	[204]			[ND]	[ND]	[ND]							
5	d198	0,01	CM	16.180,5	16.212	15.780								15.964	16.132	18.200	15.799,3
			MC	16.180,5	16.212	ND								15.834	15.992	16.720	15.781
			TM	(6,49)	(6,35)	(253)								(ND)	(ND)	(ND)	(ND)
			NT	[246]	[241]	[ND]								[3.000]	[7.000	[25.000	[2.514]
6	Pcb442	0,13	CM	53.287,3	55.133,8			136.426,3	112.253,2	111.235,2				51.930	53.820	59.740	51.208
			MC	53.287,3	55.133,8			125.779	104.121	103.740				51.104	52.620	53.402	51.101
			TM	(86,87)	(76,79)			(ND)	(ND)	(ND)				(ND)	(ND)	(ND)	(ND)
			NT	[659]	[543]			[ND]	[ND]	[ND]				[8.000]	[15.000	[40.000	[9.998]

ND: No detectado. Celdas: CM: costo medio, MC: mejor costo, TM (): tiempo medio, CPC: gigaciclos de PC (velocidad de procesador x TM), NT []: nro de tours.

Según la Figura 71, de las doce variantes de Algoritmos Genéticos, solo cuatro lograron encontrar soluciones que superan a las del SCA_2Opt y SCA_2Opt_r en todas sus instancias comparadas, las cuales se detallan a continuación:

- Un Algoritmo Genético (AG) cuyos resultados fueron consolidados por Dorigo y Gambardella (1997), pero cuyo desarrollo original fue presentado por Whitley, Starkweather y Fuquay (1989).

- Algoritmos Genéticos FRAG_GA, SWAP_GATSP y el IGA (Algoritmo Genético Mejorado), desarrollados en la investigación de Sallabi y El-Haddad (2009) en Libia.

b. ***Para las metaheurísticas basadas en Colonia de Hormigas.*** Se presentan los resultados consolidados en la investigación de Pérez (2011) en la Figura 72.

Figura 72

Grupo de Colonia de Hormigas tomados de Pérez (2011)

Nº	Procesadores			Pent 4, 3.06 GHz		0.175 GHz DEC		Pent 4, 2.4 GHz		Pent 4, 2.8 GHz		
	País:			Colombia		Bélgica y Suiza [18]		China [19]		USA [20]		
	Instancia TSP	Valor r	Variables	SCA_2opt_r	SCA_2opt	ACS	ACS_3opt	MNACO	MMAS	ACO MP	ACO SLR	ACO MLR
1	Eil50	0,10	CM	430,48	448,83	ND				513,61	471,98	462,51
			MC	430,48	448,83	425				513,61	460,9	453,64
			TM	(0,114)	(0,098)	(ND)				(10)	(7)	(36)
			CPC	0,34884	0,29988	(ND)				28	19,6	100,8
			NT	[92]	[62]	[1.830]				[400]	[400]	[400]
2	Eil51	0,1	CM	436,08	449,85			429,04	430,06			
			MC	436,08	449,85			428,87	428,98			
			TM	(0,219)	(0,109)			(10,98)	(40)			
			CPC	0,67	0,33			26,35	96			
			NT	[186]	[63]			[ND]	[ND]			
3	Eil75	0,16	CM	560,14	588,61	ND				606,75	597,96	572,53
			MC	560,14	588,61	535				606,75	585,69	566,01
			TM	(0,805)	(0,320)	(ND)				(17)	(13)	(75)
			CPC	2,46	0,98	(ND)				47,60	36,40	210
			NT	[451]	[94]	[3.480]				[400]	[400]	[400]
4	KroA100	0,16	CM	22.219,34	22.379,4	ND		21.288,13	21.342,97	25.483	24.251,0	23.898,04
			MC	22.219,34	22.379,40	21.282		21.285,40	21.285,40	25.483	24229,9	23.531,40
			TM	(1,07)	(0,87)	(ND)		(26,88)	(104,8)	(26)	(22)	(130)
			CPC	3,27	2,66	(ND)		64,51	251,52	72,80	61,60	364
			NT	[206]	[140]	[4.820]		[ND]	[ND]	[400]	[400]	[400]
5	d198	0,01	CM	16.180,53	16.212	16.054	15.781,7			18.381	18.375,28	16.918,76
			MC	16.180,53	16.212	15.888	15.780			18.381	17.982,90	16.752,80
			TM	(6,49)	(6,35)	(ND)	(238)			(ND)	(ND)	(ND)
			CPC	19,86	19,43	(ND)	41,65			(ND)	(ND)	(ND)
			NT	[246]	[241]	[585.000]	[ND]			[ND]	[ND]	[ND]
6	Pcb442	0,13	CM	53.287,29	55.133,8	51.690				71.742	69.572,73	81.885,64
			MC	53.287,29	55.133,80	51.268				71.742	68.519,80	61.365,30
			TM	(86,87)	(76,79)	(ND)				(ND)	(ND)	(ND)
			CPC	265,82	234,98	(ND)				(ND)	(ND)	(ND)
			NT	[659]	[543]	[595.000]				[ND]	[ND]	[ND]

ND: No detectado. Celdas: CM: costo medio, MC: mejor costo, TM (): tiempo medio, CPC: gigaciclos de PC (velocidad de procesador x TM), NT []: nro de tours.

Fuente: Pérez (2011)

Del total de siete variantes del Algoritmo Colonia de Hormigas, solo cuatro lograron encontrar soluciones que superan a las del SCA_2Opt y SCA_2Opt_r en todas sus instancias comparadas. De las cuatro, solo dos proceden de una publicación de acceso libre y se detallan a continuación:

- Un Algoritmo de System Colony Ant (ACS).
- Un Algoritmo de System Colony Ant cuya solución fue mejorada con la heurística 3Opt (ACS_3Opt).

En ambos casos, dichos algoritmos presentados dentro de la investigación de Dorigo y Gambardella (1997).

c. ***Para las metaheurísticas basadas en Enfriamiento Simulado***. Se presentan los resultados consolidados en la investigación de Pérez (2011) en la Figura 73.

Figura 73

Grupo de Enfriamiento Simulado tomados de Pérez (2011)

Nº	País:			Colombia		Bélgica y Suiza [18]	Repúbica de Korea [25]			China [26]
	Instancia TSP	Valor r	Variab	SCA_2opt_r	SCA_2opt	SA	CSA	TFSA	TRFSA	PSO-SA
1	Att48	0,01	CM	**34.021,18**	34.933					**34.512**
			MC	**34.021,18**	34.933					**33.966**
			TM	(0,078)	(0,094)					(ND)
			NT	[65]	[66]					[ND]
2	Eil50	0,10	CM	430,48	448,83	ND				
			MC	**430,48**	448,83	**443**				
			TM	(0,114)	(0,098)	(ND)				
			NT	[92]	[62]	[68.512]				
3	Eil51	0,1	CM	**436,08**	**449,85**		480,09	477,85	480,16	
			MC	**436,08**	449,85		**449,66**	453,48	**449,66**	
			TM	(0,219)	(0,109)		(ND)	(ND)	(ND)	
			NT	[186]	[63]		[136.000]	[105.000]	[95.588]	
4	Eil75	0,16	CM	560,14	588,61	ND				
			MC	**560,14**	588,61	**580**				
			TM	(0,805)	(0,320)	(ND)				
			NT	[451]	[94]	[173.250]				
5	Eil76	0,25	CM	**559,61**	**567,01**		649,05	639,86	647,22	
			MC	**559,61**	**567,01**		613,7	603,68	618,03	
			TM	(1,40)	(0,352)		(ND)	(ND)	(ND)	
			NT	**[964]**	**[118]**		[204,000]	[159,000]	[147.550]	

ND: No detectado. Celdas: CM: costo medio, MC: mejor costo, TM (): tiempo medio, NT []: nro de tours.

Fuente: Pérez (2011)

Del total de cinco variantes del algoritmo de Enfriamiento Simulado consolidadas por Pérez (2011), solo una logró encontrar soluciones que superan a las soluciones del SCA_2Opt y SCA_2Opt_r en todas las instancias comparadas. Pero el método de cálculo de las distancias resultó incongruente al indicado por TSPLIB. Por tanto, no se tiene alguna instancia para comparar con el Grupo de Metaheurísticas de Enfriamiento Simulado

d. ***Para Otras Metaheurísticas***. Se presentan los resultados consolidados en la investigación de Pérez (2011) en la Figura 74.

Figura 74

Grupo de Otras Metaheurísticas tomadas de Pérez (2011)

Procesador				Pent 4, 3,06 GHz			ND		ND	IBM 0, 533 GHz
Nº	País:			Colombia		Bélgica y Suiza [18]	China [27]	China [28]	China [29]	Serbia [30]
	Instancia TSP	Valor r	Variab	SCA_2opt_r	SCA_2opt	PE	PSO-TS-CO - 2opt	REA	SFLA	BCO
1	Burma14	0,02	CM	**30,8**	**31,21**			32,77	**30,8**	
			MC	**30,8***	**31,21***			31,88*	**30,8***	
			TM	**(0,00)**	**(0,00)**			(ND)	(1,84)	
			NT	[24]	[22]			[ND]	[ND]	
2	Att48	0,01	CM	**34.021,18**	**34.933**			35.413		
			MC	**34.021,18**	**34.933**			34.707		
			TM	(0,078)	(0,094)			(ND)		
			NT	[65]	[66]			[ND]		
3	Eil50	0,10	CM	430,48	448,83	ND				
			MC	**430,48**	448,83*	**426**				
			TM	(0,114)	(0,098)	(ND)				
			NT	**[92]**	**[62]**	[100.000]				
4	Eil51	0,1	CM	**436,08**	449,85		440,9	455	**436,76**	ND
			MC	436,08	449,85		ND	451	**428,87**	**428,87**
			TM	**(0,219)**	**(0,109)**		(30)	(ND)	(17,42)	(29)
			CPC	**0,67**	**0,33**		ND	(ND)	9,28	15,46
			NT	[186]	[63]		[ND]	[ND]	[ND]	[ND]
5	Eil75	0,16	CM	560,14	588,61	ND				
			MC	**560,14**	588,61	**542**				
			TM	(0,805)	(0,320)	(ND)				
			NT	**[451]**	**[94]**	[325.000]				

ND: No detectado. Celdas: CM: costo medio, MC: mejor costo, TM (): tiempo medio, CPC: gigaciclos de PC (velocidad de procesador x TM), NT []: nro de tours.

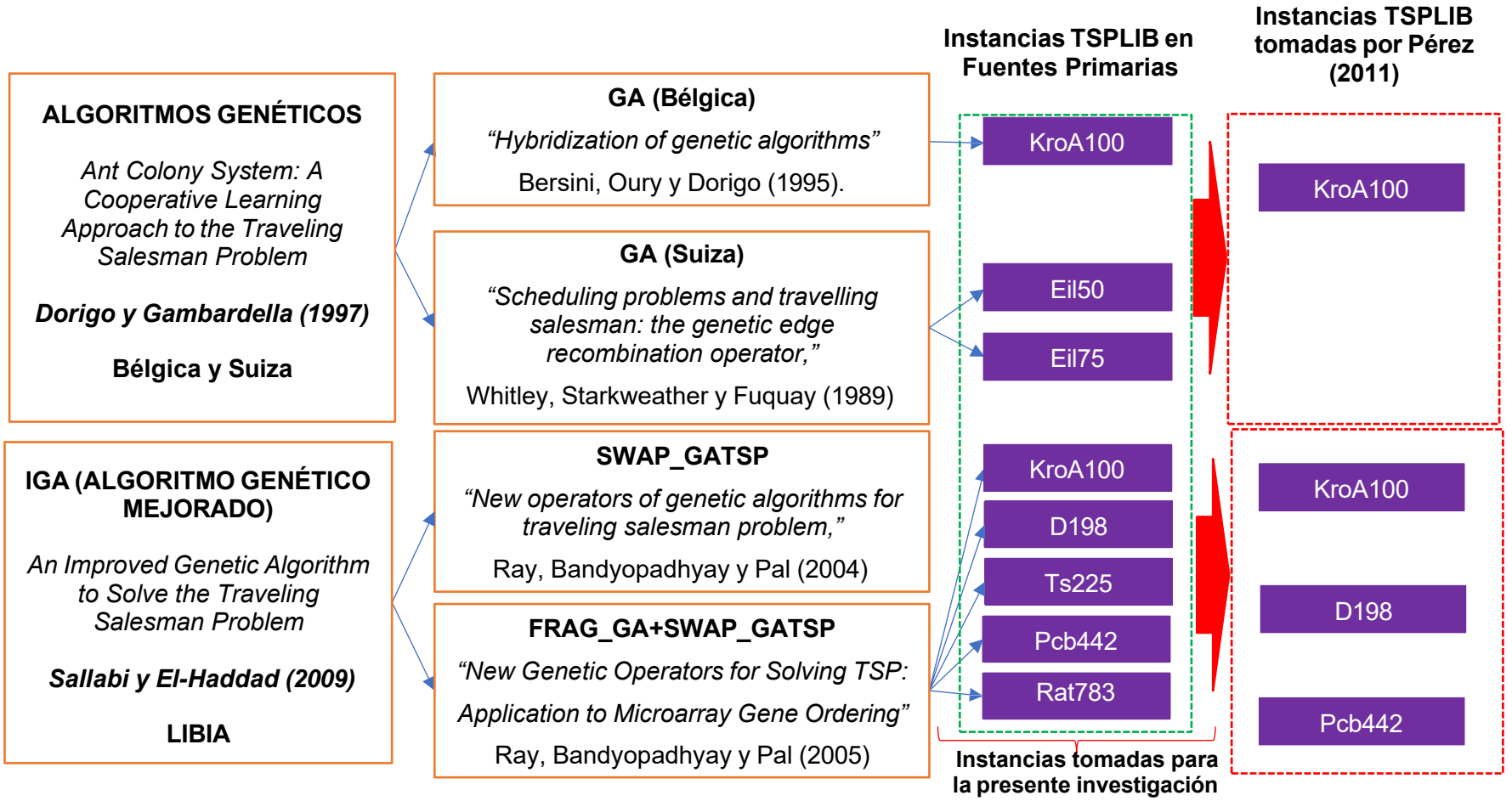
Fuente: Pérez (2011)

Del total de cinco algoritmos, solo dos lograron encontrar soluciones que superaron a las soluciones del SCA_2Opt y SCA_2Opt_r en todas las instancias comparadas:

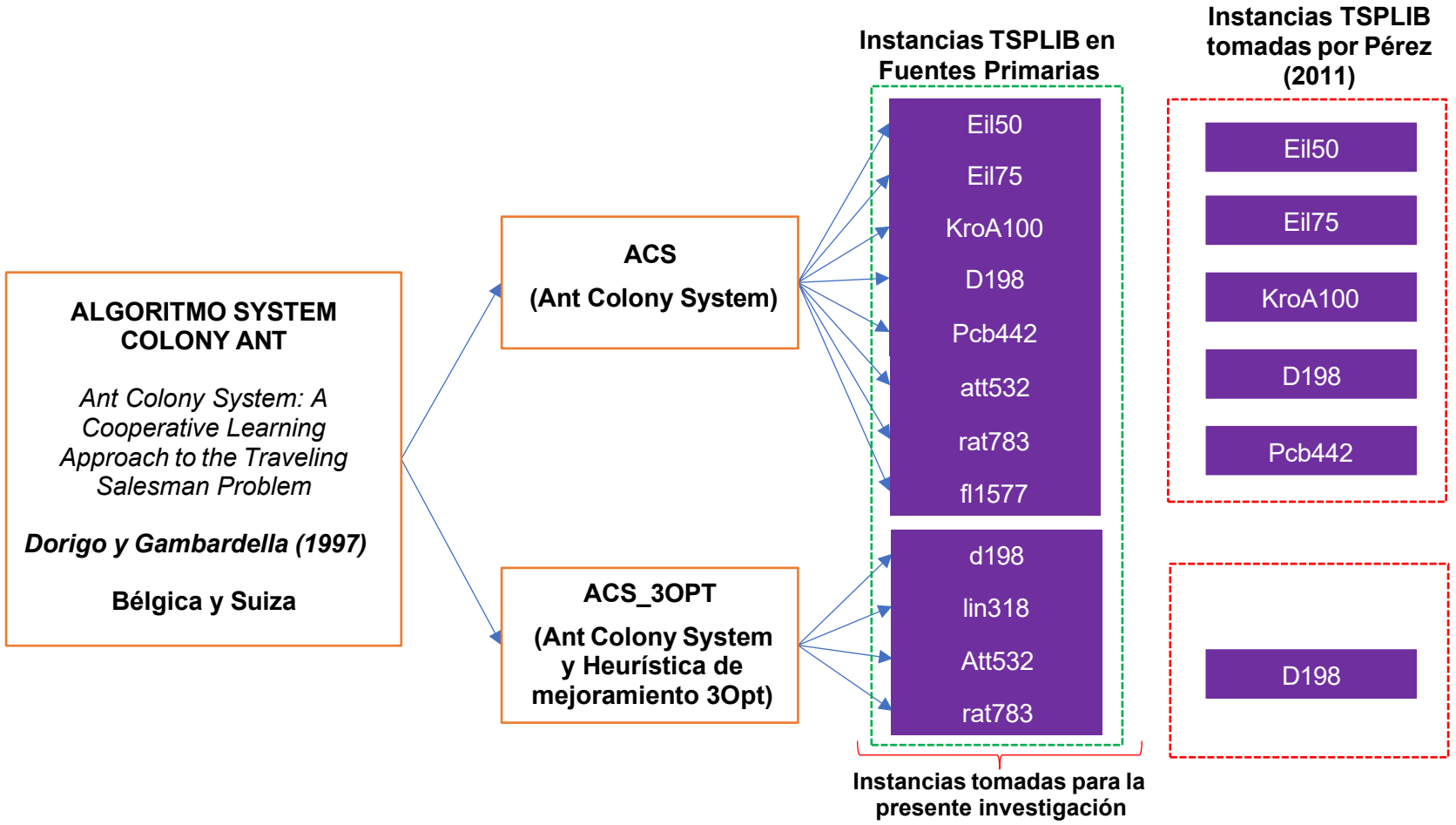
- Un algoritmo basado en Programación Evolutiva, encontrado en la investigación de Dorigo y Gambardella (1997), en Bélgica y Suiza, en el cual se utilizó el *Error Relativo* de las soluciones obtenidas para medir la calidad de la solución.

- Un algoritmo basado en Optimización de Colonia de Abejas, encontrado dentro de la investigación de Teodorovic (2008), en Serbia, en el cual se utilizó el *Error Relativo* de las soluciones obtenidas para medir la calidad de la solución.

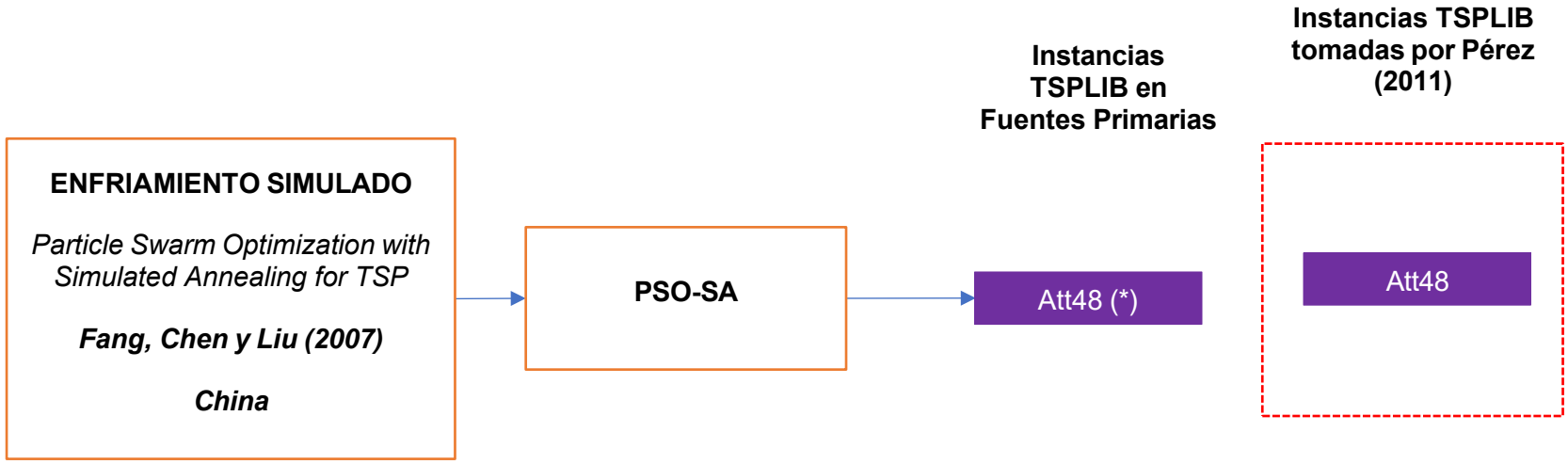
Anexo 4. Determinación de instancias para comparar según la metaheurística Algoritmos Genéticos



Anexo 5. Determinación de instancias para comparar según la metaheurística Colonia de Hormigas

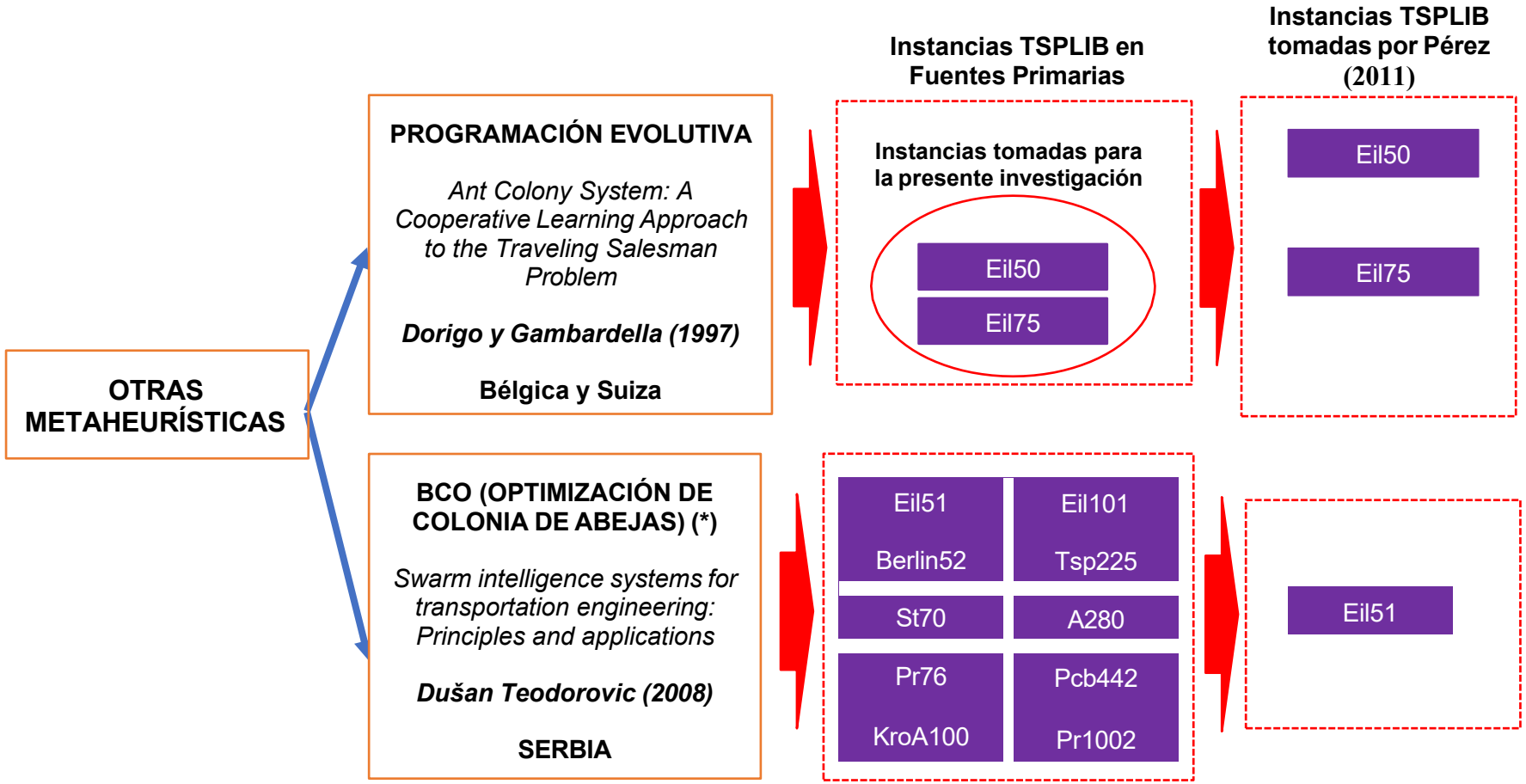


Anexo 6. Determinación de instancias a comparar según la metaheurística Enfriamiento Simulado



Nota. (*) Según indicaciones de TSPLIB, se debió usar una distancia pseudo euclidiana (filtro 2 para la selección de casos indicados en la parte "b" del numeral 4.1.1) y Pérez (2011) realizó el cálculo con una distancia euclidiana, siendo incongruente a una comparación bajo las mismas condiciones. Por esa razón, no se considera alguna metaheurística dentro del Grupo de Enfriamiento Simulado.

Anexo 7. Determinación de instancias a comparar según Otras Metaheurísticas



Nota. (*) Algoritmo donde el autor trabajó con decimales, siendo que TSPLIB indica trabajar con enteros. Por lo cual, se omitió el BCO del análisis comparativo al presentar una medición incorrecta y no tendríamos una base para compararlos con las soluciones determinadas por nuestras versiones.

Anexo 8. Resultados de algoritmos propuestos vs algoritmos en la investigación de José Mendoza (2017)

Tabla 22

Soluciones de versiones Heavy, Soft, Solver(Simplex), WinQSB(Branch and Bound), Vecino más Cercano, GRASP y Tabú Search en instancias pequeñas

Instancia	Solución Óptima TSPLIB	MÉTODOS EN MENDOZA (2017)					VERSIONES HEAVY			VERSIONES SOFT		
		Solver (Simplex)	WINQSB (Branch and Bound)	Vecino más Cercano	GRASP	Tabú Search	SCA(1)_ Multi2Opt	SCA(1)_ Multi2Opt_r	SCA(5)_ Multi2Opt	SCA(1)_ Uni2Opt	SCA(1)_ Uni2Opt_r	SCA(5)_ Uni2Opt
1.brazil58	25395	47918	29380	27384	28607	27243	25400	25400	25400	25704	25699	25704
2.dantzig42	699	699	794	864	802	853	699	700	699	704	700	704
3.Fri26	937	953	1046	965	937	937	953	953	937	973	957	937
4.gr17	2085	2090	2157	2178	2085	2085	2085	2085	2085	2085	2085	2085
5.gr21	2707	2803	3119	3003	2707	3031	2707	2707	2707	2707	2707	2707
6.gr24	1272	1388	1372	1553	1355	1490	1294	1272	1272	1314	1300	1314
7.gr48	5046	6557	5845	5840	5968	5800	5097	5094	5075	5101	5101	5101
8.gr120	6942	29261	7822	8438	9836	8280	7069	7081	7051	7377	7217	7241
9.hk48	11461	16829	13015	12137	13837	12002	11461	11461	11461	12163	11895	11735
10.si175	21407	26361	ND	22000	33292	22043	21441	21440	21440	21549	21487	21522
11.swiss42	1273	1864	1481	1437	1427	1466	1273	1273	1273	1273	1273	1273

Nota 1. El color amarillo indica que el valor óptimo es de TSPLIB.
Nota 2. Las instancias de TSPLIB fueron tomadas de la investigación de Mendoza (2017) para instancias pequeñas (hasta alrededor de 175 nodos).

Tabla 23

Escala de color mejor solución disponible: versiones Heavy y Soft; metaheurísticas Scatter, GRASP y Tabú Search para instancias medianas

			METAHEURÍSTICAS			ALGORITMOS VERSIONES SOFT			ALGORITMOS VERSIONES HEAVY		
Instancia TSPLIB	**TIPO**	**Óptimo TSPLIB**	**SCATTER**	**GRASP**	**TABU**	**SCA(1)_ Uni2Opt**	**SCA(1)_ Uni2Opt_r**	**SCA(5)_ Uni2Opt**	**SCA(1)_ Multi2Opt**	**SCA(1)_ Multi2Opt_r**	**SCA(5)_ Multi2Opt**
1.bier127	EUC2D	118282	118256	118613	124831	120724	120724	120724	118928	118928	118928
2.d198	EUC2D	15780	15793	15814	16175	16342	16024	15902	15926	15926	15902
3.eil101	EUC2D	629	611	613	657	645	639	638	638	635	632
4.fl417	EUC2D	11861	11798	11779	12227	12375	12369	12375	12092	12071	12092
5.gil262	EUC2D	2378	2350	2344	2452	2523	2502	2518	2440	2440	2433
6.kroA100	EUC2D	21282	21247	21247	22172	23314	22958	23314	21474	21474	21470
7.kroA150	EUC2D	26524	26485	26707	28364	28139	28107	28139	27493	27446	27243
8.kroA200	EUC2D	29368	29487	29521	31400	29955	29746	29868	29591	29534	29565
9.kroB100	EUC2D	22141	22086	22086	23236	22454	22291	22349	22349	22163	22211
10.kroB150	EUC2D	26130	26063	26078	28900	26651	26591	26651	26651	26439	26532
11.kroB200	EUC2D	29437	29530	29665	30857	32008	31457	31067	30651	30127	30651
12.kroC100	EUC2D	20749	20703	20703	23128	21162	21162	21162	21162	21162	20965
13.kroD100	EUC2D	21294	21249	21265	22336	22420	21945	22330	21861	21640	21853
14.kroE100	EUC2D	22068	22053	22029	22831	22872	22872	22620	22355	22226	22355
15.lin105	EUC2D	14379	14354	14354	15062	14810	14810	14712	14484	14438	14484

Nota 1. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios en cercanía al óptimo.

Nota 2. Las instancias simétricas de TSPLIB fueron tomadas de la investigación de Mendoza (2017) para instancias medianas (entre 100 a 270 nodos).

Tabla 23

Escala de color mejor solución disponible: versiones Heavy y Soft; metaheurísticas Scatter, GRASP y Tabú Search para instancias medianas

			METAHEURÍSTICAS			ALGORITMOS VERSIONES SOFT			ALGORITMOS VERSIONES HEAVY		
Instancia TSPLIB	TIPO	Óptimo TSPLIB	SCATTER	GRASP	TABU	SCA(1)_ Uni2Opt	SCA(1)_ Uni2Opt_r	SCA(5)_ Uni2Opt	SCA(1)_ Multi2Opt	SCA(1)_ Multi2Opt_r	SCA(5)_ Multi2Opt
16.lin318	EUC2D	42029	43170	42982	44154	44049	44049	44049	43450	43204	43046
17.pr107	EUC2D	44303	44280	44280	45729	44469	44446	44326	44303	44303	44303
18.pr124	EUC2D	59030	59011	59011	59882	59416	59246	59413	59246	59246	59246
19.pr136	EUC2D	96772	96967	98103	105209	106816	104439	104165	100384	100358	100384
20.pr144	EUC2D	58537	58521	58521	59106	59814	59814	59702	59143	59143	58669
21.pr152	EUC2D	73682	73622	73622	75196	74967	74967	74967	74606	74448	74606
22.pr226	EUC2D	80369	80431	80506	81038	82602	82440	81285	81285	81285	81243
23.pr264	EUC2D	49135	49685	49428	53894	52003	51950	52003	50985	50165	50985
24.pr299	EUC2D	48191	48580	48892	52306	50908	50908	50400	49217	48987	49006
25.rat195	EUC2D	2323	2329	2291	2432	2387	2363	2387	2354	2347	2354
26.rd100	EUC2D	7910	7858	7858	8358	8452	8431	8452	8018	7965	8018
27.rd400	EUC2D	15281	15883	15555	16320	15872	15851	15872	15631	15641	15641
28.ts225	EUC2D	126643	126809	126850	127580	130496	127794	130496	127512	126899	127161
29.tsp225	EUC2D	3919	3901	3849	4017	4040	4040	4031	3976	3955	3976
30.u159	EUC2D	42080	42045	42045	44355	45526	44705	42798	42547	42375	42511

Nota 1. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios en cercanía al óptimo.

Nota 2. Las instancias simétricas de TSPLIB fueron tomadas de la investigación de Mendoza (2017) para instancias medianas (entre 100 a 270 nodos).

De la Tabla 22, el SCA(5)_Multi2Opt (versión Heavy) logró encontrar el óptimo una mayor cantidad de veces para todas las instancias. En cambio, para las versiones Soft se encontró el óptimo casi en la misma cantidad de veces que las metaheurísticas con las cuales se comparó.

De la Tabla 23, las instancias de color turquesa indican que la solución óptima reportada en TSPLIB fue superada por la solución determinada por algunos de los algoritmos SCATTER o GRASP. Al revisar en la investigación original desarrollada por Mendoza (2017), no se encontraron los anexos respectivos indicando el detalle de la matriz de costos. Por lo cual, podemos intuir que la matriz de costos no fue calculada correctamente, debido a que las soluciones reportadas en la librería virtual TSPLIB son las óptimas y, por tanto, no pueden superarse, según lo indicado en TSPLIB FAQ, Preguntas y Respuestas (Reinelt, 1995).

Anexo 9. Resultados de versiones Heavy y Soft vs algoritmos metaheurísticos en Akhand, Ayon, Shahriyar, Siddique y Adeli (2020)

Tabla 24

Escala de color de mejor solución disponible versiones Heavy, Soft vs metaheurísticas ACO, VTSPO, ABCSS y DSMO

			METAHEURÍSTICAS				VERSIONES HEAVY			VERSIONES SOFT		
Instancia TSPLIB	TIPO	ÓPTIMO	ACO	VTPSO	ABCSS	DSMO	SCA(1)_ Multi2Opt	SCA(1)_ Multi2Opt_r	SCA(5)_ Multi2Opt	SCA(1)_ Uni2Opt	SCA(1)_ Uni2Opt_r	SCA(5)_ Uni2Opt
1.eil51	EUC 2D	426	499.92	429.51	428.98	428.86	430	429	430	437	430	430
2.berlin52	EUC 2D	7542	7870.45	7544.37	7544.37	7544.37	7715	7715	7657	7793	7792	7793
3.st70	EUC 2D	675	734.19	682.57	682.57	677.11	688	686	686	732	728	725
4.eil76	EUC 2D	538	581.42	559.25	550.24	558.68	548	543	546	567	553	567
5.pr76	EUC 2D	108159	127025.9	109586.1	108879.7	108159.4	109254	109055	109049	111086	109067	111069
6.rat99	EUC 2D	1211	1366.3	1256.25	1242.32	1225.56	1258	1229	1238	1301	1266	1265
7.kroa100	EUC 2D	21282	24504.9	21307.44	21299	21298.21	21474	21474	21470	23314	22958	23278
8.kroB100	EUC 2D	22141	24664.13	22475.67	22229.71	22308	22349	22163	22211	22454	22291	22349
9.rd100	EUC 2D	7910	9120.92	8094.75	7944.32	8041.3	8018	7965	8018	8452	8431	8452
10.eil101	EUC 2D	629	715.35	653.16	646.05	648.66	638	635	632	645	639	647
11.lin105	EUC 2D	14379	15364.58	14581.58	14406.12	14383	14484	14438	14484	14810	14810	14712
12.pr107	EUC 2D	44303	46317.71	44436.25	44525.68	44385.86	44303	44303	44303	44469	44446	44326
13.pr124	EUC 2D	59030	65145.25	61076.73	59030.74	60285.21	59246	59246	59246	59416	59246	59416
14.pr136	EUC 2D	96772	110872.2	99247.01	97853.91	97538.68	100384	100358	100384	106816	104439	104165

Nota 1. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios en cercanía al óptimo.
Nota 2. Instancias simétricas TSPLIB seleccionadas de la investigación de Akhand, Ayon, Shahriyar, Siddique y Adeli (2020).

Tabla 24

Escala de color de mejor solución disponible versiones Heavy, Soft vs metaheurísticas ACO, VTSPO, ABCSS y DSMO

			METAHEURÍSTICAS				VERSIONES HEAVY			VERSIONES SOFT		
Instancia TSPLIB	**TIPO**	**ÓPTIMO**	**ACO**	**VTPSO**	**ABCSS**	**DSMO**	**SCA(1)_ Multi2Opt**	**SCA(1)_ Multi2Opt_r**	**SCA(5)_ Multi2Opt**	**SCA(1)_ Uni2Opt**	**SCA(1)_ Uni2Opt_r**	**SCA(5)_ Uni2Opt**
15.kroA150	EUC 2D	26524	30546.06	27232.1	26981.98	27591.44	27493	27446	27243	28139	28107	28075
16.kroB150	EUC 2D	26130	29124.59	26579.73	26760.79	26601.94	26651	26439	26532	26651	26591	26651
17.pr152	EUC 2D	73682	79153.02	74414.17	74337.62	74243.91	74606	74453	74606	74967	74967	74967
18.u159	EUC 2D	42080	47514.43	43579.82	42862.51	42598.3	42547	42375	42511	45526	44705	42798
19.rat195	EUC 2D	2323	2534.83	2452.92	2469.31	2372.89	2354	2347	2354	2387	2363	2387
20.d198	EUC 2D	15780	17301.47	16066.44	16270.22	15978.13	15926	15926	15902	16260	16024	15902
21.kroA200	EUC 2D	29368	34547.69	30602.81	30701.86	30481.35	29591	29534	29565	29955	29746	29868
22.kroB200	EUC 2D	29437	34207.79	30767.52	31508.85	30716.5	30651	30127	30651	32008	31457	31067
23.tsp225	EUC 2D	3919	4396.39	4095.01	4140.24	4013.68	3976	3955	3976	4040	4040	4031
24.pr226	EUC 2D	80369	90501.46	81050.23	82266	83587.98	81285	81285	81243	82602	82440	81285
25.gil262	EUC 2D	2378	2730.52	2547.16	2713.75	2543.15	2440	2440	2433	2523	2502	2518
26.pr299	EUC 2D	48191	56700.65	50571.83	64464.76	50579.82	49217	48987	49006	50908	50908	50400
27.lin318	EUC 2D	42029	47442.95	44724.38	55744.52	44118.66	43450	43204	43046	44049	44049	44049
28.fl417	EUC 2D	11861	13296.85	12376.53	22253.99	12218.98	12092	12071	12092	12375	12369	12375
29.pr439	EUC 2D	107217	127228	112088	206233.14	112105.2	108569	108568	108569	112213	111894	112213
30.d493	EUC 2D	35002	39254	37132.09	68556.68	36844.63	35913	35771	35759	36793	36713	36699

Nota 1. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios en cercanía al óptimo.
Nota 2. Instancias simétricas TSPLIB seleccionadas de la investigación de Akhand, Ayon, Shahriyar, Siddique y Adeli (2020).

Anexo 10. Código del SCA(1)_Multi2Opt y SCA(1)_Multi2Opt_r

Para facilitar el análisis de la complejidad algorítmica, se dividió el código en bloques.

```
package vmc_sca_multi2opt;
import java.lang.management.ManagementFactory; import java.io.File;
import jxl.Workbook; import java.lang.management.MemoryMXBean;
import java.io.IOException; import java.lang.management.MemoryUsage;
import java.util.Iterator; import java.io.FileInputStream;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;
import jxl.read.biff.BiffException; import java.text.DecimalFormat;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import jxl.Cell; import jxl.Sheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class VMC_SCA_Multi2OPT {
public static void main(String[] args) throws IOException,
            BiffException {

long TInicio, TFin; Double tiempo; long contador=0;
double r=0.000;  // FACTOR "r"
TInicio=System.currentTimeMillis(); int iteraciones=2000;
int dataSize= 1024*1024;
String archivo="C:\\Users\\Jonathan\\TSPLIB\\si535.xlsx";
FileInputStream fileInStream = new FileInputStream(archivo);
XSSFWorkbook workBook = new XSSFWorkbook(fileInStream);
XSSFSheet selSheet = workBook.getSheetAt(0);
int filas=selSheet.getPhysicalNumberOfRows();
int Linferior=1, Lsuperior=filas-4;
int matrizpesos[][]=new int[filas][filas];
int matrizclone[][]=new int[filas][filas];
int matrizpesos2[][]=new int[filas][filas];
int matrizanswer[][]=new int[filas][filas];
int indFila = -1;   int indCol = -1;
Iterator rowIterator = selSheet.iterator();

while (rowIterator.hasNext()) {
      indFila += 1;     indCol =-1;
      Row row = (Row) rowIterator.next();
      Iterator cellIterator = row.cellIterator();
   while (cellIterator.hasNext()) {
     indCol +=1; org.apache.poi.ss.usermodel.Cell cell =
     org.apache.poi.ss.usermodel.Cell) cellIterator.next();
     if(cell.getCellType()==CellType.STRING) {
     matrizpesos[indFila][indCol]= Math.toIntExact
     (Long.parseLong(cell.getStringCellValue()));    }
     else if(cell.getCellType()==CellType.NUMERIC) {
      matrizpesos[indFila][indCol]=Math.toIntExact((long)
      cell.getNumericCellValue());
      matrizclone[indFila][indCol]=Math.toIntExact((long)
      cell.getNumericCellValue());
      matrizpesos2[indFila][indCol]=Math.toIntExact((long)
      cell.getNumericCellValue());
    }
  }
}
```

BLOQUE I — **O(1)**

BLOQUE II — **$O(n^2)$**

```
//Inicio de Obtengo ruta del VMC
int pos[]=new int[filas]; int pos2[]=new int[filas];
int posnew[]=new int[filas]; int valor=1;
int Dsuma[]=new int [filas]; int Dsuma2[]=new int [filas];
pos[0]=valor; pos2[0]=pos[0];
for (int i = 1; i< filas; i++) { matrizpesos2[i][pos2[0]]=0;}
for (int i = 1; i < filas; i++){ int minimo2 = 1000000000;
    for (int j = 1; j < filas; j++)
    {   if (minimo2>matrizpesos2[pos2[i-1]][j] &
            matrizpesos2[pos2[i-1]][j]!=0)
        { minimo2 = matrizpesos2[pos2[i-1]][j];  pos2[i]=j; }
    }
    for (int k=1;k<filas;k++){ matrizpesos2[k][pos2[i]]=0;  }
  pos2[filas-1]=pos2[0];
}
System.out.println("Ruta del VMC original: ");
System.out.println("r="+r+" , iteraciones = "+iteraciones);
int sumax=0;
for(int i=0;i<filas;i++){System.out.print(pos2[i]+"-");
    matrizanswer[0][i]=pos2[i];
}
for(int i=0;i<filas-1;i++){
    sumax=sumax+matrizclone[pos2[i]][pos2[i+1]];
}
Dsuma[0]=sumax; Dsuma2[0]= sumax; System.out.println("");
System.out.println("Distancia del VMC es: "+sumax);
//Fin de obtengo ruta del VMC
```

BLOQUE III

$O(n^2)$

```
// INICIO DE SACRIFICIO CORTOPLACISTA
for (int d = Linferior; d < Lsuperior; d++) {
  int indFila2=-1;  int indCol2=-1;
  Iterator rowIterator2=selSheet.iterator();
  while (rowIterator2.hasNext()) {
    indFila2 += 1; indCol2 =-1;
    Row row2=(Row) rowIterator2.next();
    Iterator cellIterator2=row2.cellIterator();
    while (cellIterator2.hasNext()) {
       indCol2 +=1;
       org.apache.poi.ss.usermodel.Cell
            cell=(org.apache.poi.ss.usermodel.Cell)
            cellIterator2.next();
       if(cell.getCellType()==CellType.STRING){
       matrizpesos[indFila2][indCol2]= Math.toIntExact
       (Long.parseLong(cell.getStringCellValue()));}
       else if(cell.getCellType()==CellType.NUMERIC) {
        matrizpesos[indFila2][indCol2]=Math.toIntExact(
       (long)cell.getNumericCellValue());}
      }
   }
```

BLOQUE IV

$O(n^2)$

$O(n^4)$

. . .

```
int temp1=matrizpesos[pos2[d-1]][pos2[d]];
matrizpesos[pos2[d-1]][pos2[d]]=0;
  for (int i=1; i<filas; i++){
      matrizpesos[i][pos[0]]=0;
  }
  //Inicio Obtengo mínimo valor según fila de la matriz
  for (int i=1; i<filas; i++){ int minimo=1000000000;
     for (int j = 1; j < filas; j++){
         if (minimo>matrizpesos[pos[i-1]][j] &
                  matrizpesos[pos[i-1]][j]!=0)
         { minimo=matrizpesos[pos[i-1]][j]; pos[i]=j; }
       }
     for (int k=1; k<filas; k++){
         matrizpesos[k][pos[i]]=0;
     }
   pos[filas-1]=pos[0];
  }  //Finalizo mínimo de cada fila de la matriz
            int suma=0;
  for(int i = 0;i<filas-1;i++){
      suma = suma+matrizclone[pos[i]][pos[i+1]];
  }
  Dsuma[d]=suma;

  //APERTURO CODIGO PARA 2-OPT
  int cambio=1;
  for(int cont=0;cont<filas*6;cont++){
     double min_change= 0, change;
     int b=0, c=0, j, min_i=0,min_j=0;
     int costoactual, costonuevo;
     for(int i=0;i<=filas-4;i++){
        for(j=i+2;j<=filas-2;j++){
            costonuevo=matrizclone[pos[i]][pos[j]]+
                       matrizclone[pos[i+1]][pos[j+1]];
            costoactual=matrizclone[pos[i]][pos[i+1]]+
                       matrizclone[pos[j]][pos[j+1]];
            change = (1+r)*costonuevo-costoactual;
            if (change<min_change){
                min_change=change; b=pos[j];c=pos[i+1];
               min_i=i+1; min_j=j;
            }
        }
     }
      pos[min_i]=b; pos[min_j]=c;
     int suma2=0; pos[0]=valor;
     for(int i=0;i<filas;i++){
     matrizanswer[d][i]=pos[i];
     }
     for(int i=0;i<filas-1;i++){
      suma2=suma2+matrizclone[pos[i]][pos[i+1]];
     }
     suma2[d]=suma2;
     if (cont==iteraciones-1){
     System.out.println("La distancia asociada al "+d+"_
     apalancamiento después del "+_
     (cont+1)+" intercambio 2OPT es: " +suma2);
     }
  }
  // CIERRO CODIGO PARA 2-OPT
}
// CIERRO SACRIFICIO CORTOPLACISTA
```

BLOQUE V

$O(n^2)$

BLOQUE VI

$O(n^3 log(n))$

$O(n^4 log(n))$

```
  int posi=0, min= 999999999;
for(int i = Linferior; i < Lsuperior; i++){
   if (min > Dsuma2[i]){
      min = Dsuma2[i];
      posi=i;
   }
}
System.out.println("RESULTADOS");
System.out.println("La menor distancia para el_
Multi_2opt aplicados a los distintos puntos de_
apalancamiento: "+Dsuma2[posi]);
System.out.println("cuyo punto de apalancamiento_
fue: "+posi);
```

BLOQUE VII

O(n)

```
for (int i = 0; i < filas; i++){
    if (i<filas-1){
       posnew[i]=matrizanswer[posi][i];
       System.out.print(matrizanswer[posi][i]+"-");
    }
    else if (i==filas-1){
       posnew[filas-1]=valor;
       System.out.println(valor);
    }
}
```

BLOQUE VIII

O(n)

```
Runtime runtime= Runtime.getRuntime();
System.out.println("Memoria total: "+ _
      runtime.totalMemory()/dataSize + "MB");
System.out.println("Memoria libre: "+ _
      runtime.freeMemory()/dataSize + "MB");
System.out.println("Memoria usada:"+(double)
((runtime.totalMemory()-runtime.freeMemory())*1000
/dataSize)/1000 + "MB");

TFin = System.currentTimeMillis();
tiempo = (double)(TFin - TInicio);
System.out.println("La cantidad de segundos es: "_
+tiempo/1000 +" segundos");
```

BLOQUE IX

O(1)

```
   }  //Fin de public static void main(String[] args) throws
            IOException, BiffException
} // Fin de public class VMC_SCA_Multi2OPT
```

Anexo 11. Análisis de la complejidad a priori del SCA(1)_Multi2Opt y SCA(1)_Multi2Opt_r

Se hace hincapié en que el algoritmo SCA(1)_Multi2Opt es un caso particular cuando r = 0 del SCA(1)_Multi2Opt_r, por lo cual ambas versiones presentan la misma complejidad según notación Big O.

A continuación, para la determinación de la complejidad algorítmica tanto del SCA(1)_Multi2Opt como del SCA(1)_Multi2Opt_r, según el método de la Big O, se particionó su código, implementado en NetBeans mediante lenguaje de programación Java, en bloques descritos en el Anexo 10 de tal manera que se pueda determinar la complejidad de los mencionados bloques para luego calcular la complejidad total del algoritmo.

Bloque I: Se tiene instrucciones principalmente de asignación y declaración de variables.

Complejidad = O(1)

Bloque II: Se tienen instrucciones para la lectura de los elementos de la matriz de costos, por medio de dos While.

Complejidad = $O(n)O(n) = O(n^2)$

Bloque III: Código para obtener ruta del vecino más cercano.

$$
\begin{aligned}
Complejidad &= O(1) + O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(n) + O(1) \\
&= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(1) \\
&= O(n^2) + O(1) \\
&= O(n^2)
\end{aligned}
$$

Bloque IV: Lectura de elementos de la matriz de costos para el SCA(1).

Complejidad = $O(n) \times O(n) = O(n^2)$

Bloque V: Prohíbe arco según nodo de apalancamiento + Pensamiento del Vecino más cercano = Sacrificio Cortoplacista.

$$
\begin{aligned}
&\textit{Complejidad} = O(1) + O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(1) \\
&= O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(1) \\
&= O(n^2) + O(n)O(n) + O(1) + O(n) + O(1) \\
&= O(n^2) + O(1) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(1) \\
&= O(n^2) + O(1) \\
&= O(n^2)
\end{aligned}
$$

Bloque VI: Ejecución de la heurística 2Opt para mejorar cada ruta construida en el Bloque V (cuya complejidad se determinó en el Anexo 24).

$\textit{Complejidad} = O(n^3 log(n))$

Bloque VII: Se obtiene la posición y distancia del apalancamiento que produce el menor costo del total de rutas mejoradas en el Bloque VI y se imprime en la pantalla de la consola.

$$
\begin{aligned}
&\textit{Complejidad} = O(1) + O(n) + O(1) \\
&= O(n) + O(1) \\
&= O(n)
\end{aligned}
$$

Bloque VIII: Imprime la secuencia de nodos que se debe seguir para la ruta construida por la posición de apalancamiento determinada en el Bloque VII.

$\textit{Complejidad} = O(n)$

Bloque IX: Imprime en la consola la cantidad de memoria RAM consumida y tiempo en segundos necesarios para la ejecución del algoritmo.

$\textit{Complejidad} = O(1)$

De tal manera que se tiene el siguiente código:

```
public class VMC_SCA_Multi2OPT {
 public static void main(String[] args) throws IOException, BiffException {
 BLOQUE I  - Complejidad = O(1)
 BLOQUE II - Complejidad = O(n^2)
 BLOQUE III - Complejidad = O(n^2)

  // Inicio de sacrificio cortoplacista

 for (int d=Linferior;d<Lsuperior;d++) {

 BLOQUE IV - Complejidad = O(n^2)
 BLOQUE V - Complejidad = O(n^2)
 BLOQUE VI - Complejidad = O(n^3log(n)

 }    // Cierro sacrificio cortoplacista

 BLOQUE VII - Complejidad = O(n)
 BLOQUE VIII- Complejidad = O(n)
 BLOQUE IX - Complejidad = O(1)
  }
}
```

=O(1)+O(n²)+O(n²)
= O(n²)

+

O(n)×[O(n²)+O(n²)+
O(n³log(n))]
= O(n)×[O(n³log(n)]
= O(n⁴log(n))

+

=O(n)+O(n)+O(1)
= O(n) + O(1)
= O(n)

O(n⁴log(n)

Donde, como resultado final, se obtiene:

Complejidad del algoritmo SCA(1)_Multi2Opt = $O(n^2) + O(n^4log(n)) + O(n)$

Complejidad del SCA(1)_Multi2Opt_r y SCA(1)_Multi2Opt = $O(n^4log(n))$

Anexo 12. Código del SCA(5)_Multi2Opt_r

Para facilitar el análisis de la complejidad algorítmica, se dividió el código en bloques.

```
package vmc_multisca_multi2opt;

import java.lang.management.ManagementFactory;
import java.lang.management.MemoryMXBean; import jxl.Sheet;
import jxl.read.biff.BiffException; import java.io.IOException;
import java.lang.management.MemoryUsage; import jxl.Workbook;
import java.io.FileInputStream;  import java.util.Iterator;
import org.apache.poi.xssf.usermodel.XSSFWorkbook; import jxl.Cell;
import org.apache.poi.ss.usermodel.CellType; import java.io.File;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet;

public class VMC_MultiSCA_Multi2OPT {

public static void main(String[] args) throws IOException,
BiffException {
 int dataSize= 1024*1024;
 Obtener_Ruta(1,0,0,0,0);  Obtener_Ruta(1,1,0,0,0);
 Obtener_Ruta(1,0,1,0,0);  Obtener_Ruta(1,1,1,0,0);
 Obtener_Ruta(0,1,1,1,1);  Obtener_Ruta(1,0,0,1,0);
 Obtener_Ruta(1,0,1,1,0);  Obtener_Ruta(1,1,0,1,0);
 Obtener_Ruta(1,0,1,1,1);  Obtener_Ruta(1,1,0,1,1);
 Obtener_Ruta(1,1,1,0,1);  Obtener_Ruta(1,0,0,0,1);
 Obtener_Ruta(1,1,0,0,1);  Obtener_Ruta(1,0,0,1,1);
 Obtener_Ruta(1,0,1,0,1);  Obtener_Ruta(1,1,1,1,1);

 Runtime runtime= Runtime.getRuntime();
   System.out.println("Memoria total: "+
            runtime.totalMemory()/dataSize + "MB");
   System.out.println("Memoria libre: "+
            runtime.freeMemory()/dataSize + "MB");
 System.out.println("Memoria usada: "+(double)
          ((runtime.totalMemory()-runtime.freeMemory())
          *1000/dataSize)/1000 + "MB");
}

private static String Obtener_Ruta(Integer bn1, Integer bn2, Integer
bn3, Integer bn4, Integer bn5) throws IOException, BiffException {

String sRuta=""; int dataSize= 1024*1024;
double r=0; //r=0 hace no tener "factor r"
String archivo="C:\\Users\\Jonathan\\TSPLIB\\st70.xlsx";
FileInputStream fileInStream = new FileInputStream(archivo);
XSSFWorkbook workBook = new XSSFWorkbook(fileInStream);
XSSFSheet selSheet = workBook.getSheetAt(0);
int filas=selSheet.getPhysicalNumberOfRows();
int Linferior=1, Lsuperior=filas-6;
int iteraciones=filas*6;
int indFila=-1; int indCol=-1;
int matrizpesos[][]=new int[filas][filas];
int matrizclone[][]=new int[filas][filas];
int matrizpesos2[][]=new int[filas][filas];
int matrizanswer[][]=new int[filas][filas];
```

BLOQUE X

BLOQUE I

O(1)

```
Iterator rowIterator = selSheet.iterator();
while (rowIterator.hasNext()) {
    indFila+=1; indCol=-1;
    Row row=(Row) rowIterator.next();
    Iterator cellIterator = row.cellIterator();
    while (cellIterator.hasNext()) {
    indCol +=1; org.apache.poi.ss.usermodel.Cell cell =
    (org.apache.poi.ss.usermodel.Cell) cellIterator.next();
       if(cell.getCellType()==CellType.STRING)
        matrizpesos[indFila][indCol] = Math.toIntExact_
        (Long.parseLong(cell.getStringCellValue()));
       else if(cell.getCellType()==CellType.NUMERIC)
        matrizpesos[indFila][indCol] = Math.toIntExact_
        ((long)cell.getNumericCellValue());
        matrizclone[indFila][indCol] = Math.toIntExact_
        ((long)cell.getNumericCellValue());
        matrizpesos2[indFila][indCol] = Math.toIntExact_
        ((long)cell.getNumericCellValue());
    }
}
int pos[]= new int [filas];
int pos2[]= new int [filas];
int posnew[]= new int [filas];
int Dsuma[]= new int [filas];
int Dsuma2[]= new int [filas];
pos[0]=1; pos2[0]=pos[0];
System.out.println("iteraciones="+iteraciones+" y r="+ r);
```

BLOQUE II

$O(n^2)$

```
//Inicio de Algoritmo del VMC
for (int i=1; i<filas; i++){
    matrizpesos2[i][pos2[0]]=0;
}
for (int i=1; i<filas; i++){
    int minimo2 = 1000000000;
    for (int j=1; j<filas; j++){
        if(minimo2>matrizpesos2[pos2[i-1]][j] &
            matrizpesos2[pos2[i-1]][j]!=0){
             minimo2 = matrizpesos2[pos2[i-1]][j];
             pos2[i]=j;
        }
    }
    for (int k=1; k<filas; k++){
        matrizpesos2[k][pos2[i]]=0;
    }
  pos2[filas-1]=pos2[0];
}
for(int i = 0;i<filas;i++){
    matrizanswer[0][i]=pos2[i];
}
int sumax=0;
for(int i = 0;i<filas-1;i++){
sumax = sumax+matrizclone[pos2[i]][pos2[i+1]];
}
sumax= sumax+ matrizclone[pos2[filas-2]][pos2[0]];
Dsuma[0]=sumax; Dsuma2[0]= sumax;
//Fin de Algoritmo del VMC
```

BLOQUE III

$O(n^2)$

```
//INICIO DE SACRIFICIO CORTOPLACISTA
for(int d=Linferior; d<Lsuperior; d++){ int indFila2=-1;
int indCol2=-1;Iterator rowIterator2=selSheet.iterator();
  while (rowIterator2.hasNext()) {
   indFila2 += 1; indCol2 =-1;
   Row row2 = (Row) rowIterator2.next();
   Iterator cellIterator2 = row2.cellIterator();
    while (cellIterator2.hasNext()) { indCol2 +=1;
     org.apache.poi.ss.usermodel.Cell cell=(org.apache_
     .poi.ss.usermodel.Cell) cellIterator2.next();
       if(cell.getCellType()==CellType.STRING)
         matrizpesos[indFila2][indCol2]=Math.toIntExact
         Long.parseLong(cell.getStringCellValue()));
       else if(cell.getCellType()==CellType.NUMERIC)
         matrizpesos[indFila2][indCol2]=Math.toIntExact
         ((long)cell.getNumericCellValue());
    }
  }

 if(bn1==1){matrizpesos[pos2[d-1]][pos2[d]]=0; }
 if(bn2==1){matrizpesos[pos2[d]][pos2[d+1]]=0; }
 if(bn3==1){matrizpesos[pos2[d+1]][pos2[d+2]]=0;}
 if(bn4==1){matrizpesos[pos2[d+2]][pos2[d+3]]=0;}
 if(bn5==1){matrizpesos[pos2[d+3]][pos2[d+4]]=0;}

 for(int i=1;i<filas;i++){matrizpesos[i][pos[0]]=0;}
 for(int i=1;i<filas;i++){int minimo=1000000000;
   for(int j = 1; j < filas; j++){
      if(minimo>matrizpesos[pos[i-1]][j] &
      matrizpesos[pos[i-1]][j]!=0){
      minimo=matrizpesos[pos[i-1]][j]; pos[i]=j;}
   }
   for(int k=1;k<filas;k++){matrizpesos[k][pos[i]]=0;}
     pos[filas-1]=pos[0];
 } int suma=0;
 for(int i=0;i<filas;i++){matrizanswer[d][i]=pos[i];}
 for(inti=0;i<filas-1;i++){
 suma=suma+matrizclone[pos[i]][pos[i+1]];}Dsuma[d]=suma;

 //APERTURO CODIGO PARA 2-OPT
  int cambio=1;
  for (int cont = 0; cont < iteraciones; cont++){
     double min_change= 0, change; int min_i=0,min_j=0;
     int costoactual, costonuevo, b=0, c=0, j;
   for(int i=0;i<=filas-4;i++){
      for (j=i+2;j<=filas-2;j++){
          costonuevo=matrizclone[pos[i]][pos[j]]+
            matrizclone[pos[i+1]][pos[j+1]];
          costoactual=matrizclone[pos[i]][pos[i+1]]+
            matrizclone[pos[j]][pos[j+1]];
          change = (1+r)*costonuevo-costoactual;
        if (change < min_change){ min_i=i+1; min_j=j;
           min_change=change; b=pos[j];c=pos[i+1];  }
      }
   } pos[min_i]=b; pos[min_j]=c; int suma2=0;  pos[0]=1;
   for(int i=0;i<filas;i++){matrizanswer[d][i]=pos[i];}
   for(int i=0;i<filas-1;i++){suma2=suma2+
      matrizclone[pos[i]][pos[i+1]]; }  Dsuma2[d]=suma2;
 }   //CIERRO CODIGO PARA 2-OPT
} // FIN DE SACRIFICIO CORTOPLACISTA
```

BLOQUE IV

BLOQUE V

BLOQUE VI

BLOQUE VII

$O(n^2)$

$O(1)$

$O(n^4 log(n))$

$O(n^2)$

$O(n^3 log(n))$

```
int posi=0, min= 999999999;
for (int i = Linferior; i < Lsuperior; i++)
{
    if (min>Dsuma2[i]){min=Dsuma2[i];posi=i;}
}
System.out.println("La menor distancia MultiSCA aplicada_
a los puntos de apalancamiento del vector "+"_
("+bn1+","+bn2+","+bn3+","+bn4+","+bn5+");  "+_
Dsuma2[posi]+";  "+posi);
for (int i = 0; i < filas; i++){
    if (i<filas-1){ posnew[i]=matrizanswer[posi][i];  }
    else if (i==filas-1){ posnew[filas-1]=1;  }
}

for (int i = 0; i < filas; i++){
    System.out.print(posnew[i]+"-");
}
 System.out.println();
 Runtime runtime= Runtime.getRuntime();
 return sRuta;
  } // Fin de private static String Obtener_Ruta
} // Fin de public class VMC_MultiSCA_Multi2OPT
```

BLOQUE VIII — **O(n)**

BLOQUE IX — **O(1)**

Anexo 13. Análisis de la complejidad a priori del SCA(5)_Multi2Opt

Se hace hincapié en que el algoritmo SCA(5)_Multi2Opt es una ampliación (o extensión) del SCA(1)_Multi2Opt, donde se realizan a lo más cinco sacrificios cortoplacistas en la construcción de una ruta, la cual es guiada por el denominado vector prohibición. Por lo cual, se presenta una complejidad de $m \times O(n^4 \log(n))$, donde "m" representa la cantidad de vectores prohibición definidos y, por tanto, según notación Big O, se tendría una complejidad $O(n^4 \log(n))$.

A continuación, para la determinación de la complejidad algorítmica del SCA(5)_Multi2Opt, según el método de la Big O, se particionó su código, implementado en NetBeans mediante lenguaje de programación Java, en bloques descritos en el Anexo 12, de tal manera que se pueda determinar la complejidad de los mencionados bloques para luego calcular la complejidad total del algoritmo.

Bloque I: Se tiene instrucciones principalmente de asignación y declaración de variables.

Complejidad = $O(1)$

Bloque II: Se tienen principalmente instrucciones para lectura de los elementos de la matriz de costos, por medio de dos While.

Complejidad = $O(1) + O(n)O(n) + O(1) = O(n^2)$

Bloque III: Código para obtener ruta del vecino más cercano.

$$\textit{Complejidad} = O(n^2) + O(n)O(n) + O(n)O(n) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(n^2) + O(n^2) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(n) + O(1)$$
$$= O(n^2) + O(1) = O(n^2)$$

Bloque IV: Lectura de elementos de la matriz de costos para el SCA(5).

Complejidad = $O(n) \times O(n) = O(n^2)$

Bloque V: Condicional para la activación de los distintos vectores prohibición.

Complejidad = O(1)

Bloque VI: Construye rutas en base al Sacrificio Cortoplacista, donde los vectores prohibición que guían la búsqueda se definieron en el Bloque V.

$$\begin{aligned}
&\textit{Complejidad} = O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(n) + O(1) \\
&= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(1) \\
&= O(n^2) + O(1) \\
&= O(n^2)
\end{aligned}$$

Bloque VII: Ejecución de la heurística 2Opt para mejorar cada ruta construida en el Bloque VI (cuya complejidad se determinó en el Anexo 24).

$$\textit{Complejidad} = O(n^3 \log(n))$$

Bloque VIII: Se obtiene la posición y distancia del apalancamiento que produce el menor costo del total de rutas mejoradas en el Bloque VII y se imprime en la pantalla de la consola.

$$\begin{aligned}
&\textit{Complejidad} = O(1) + O(n) + O(1) + O(n) \\
&= O(n) + O(1) + O(n) \\
&= O(n) + O(n) \\
&= O(n)
\end{aligned}$$

Bloque IX: Imprime la ruta final según el vector prohibición utilizado.

$$\textit{Complejidad} = O(1)$$

De tal manera que se tiene el siguiente código:

```
public class VMC_SCA_Multi2OPT {
 public static void main(String[] args) throws IOException, BiffException {
  BLOQUE I  - Complejidad = O(1)
  BLOQUE II - Complejidad = O(n^2)
  BLOQUE III - Complejidad = O(n^2)

 // Inicio de sacrificio cortoplacista
 for (int d=Linferior;d<Lsuperior;d++) {
  BLOQUE IV - Complejidad = O(n^2)
  BLOQUE V - Complejidad = O(1)
  BLOQUE VI - Complejidad = O(n^2)
  BLOQUE VII - Complejidad = O(n^3log(n)
 }    // Cierro sacrificio cortoplacista

  BLOQUE VIII - Complejidad = O(n)
  BLOQUE IX- Complejidad = O(1)
 }
}
```

= O(1)+ O(n²)+ O(n²)
= O(n²)

+

= O(n)×[O(n²)+O(1)+
O(n²)+O(n³log(n))]
= O(n)×[O(n³log(n))]
= O(n⁴log(n))

+

= O(n) + O(1)
= O(n)

O(n⁴log(n))

Bloque X: Se continúa con el siguiente vector prohibición para volver a ejecutar los Bloques I al IX.

Complejidad = m × Complejidad del Algoritmo = m ×O(n⁴log(n)) = O(n⁴log(n))

Donde, como resultado final, se obtiene:

Complejidad del algoritmo SCA(5)_Multi2Opt = O(n⁴log(n))

Anexo 14. Código del SCA(1)_Uni2Opt y SCA(1)_Uni2Opt_r

Para facilitar el análisis de la complejidad algorítmica, se dividió el código en bloques.

```
package vmc_sca_2opt;

import java.lang.management.ManagementFactory; import java.io.File;
import java.lang.management.MemoryMXBean; import jxl.Workbook;
import org.apache.poi.ss.usermodel.Row; import java.io.IOException;
import java.lang.management.MemoryUsage;
import org.apache.poi.ss.usermodel.CellType; import jxl.Sheet;
import java.io.FileInputStream; import java.util.Iterator;
import org.apache.poi.xssf.usermodel.XSSFWorkbook; import jxl.Cell;
import org.apache.poi.xssf.usermodel.XSSFSheet;

public class VMC_SCA_2Opt {
public static void main(String[] args) throws IOException,
BiffException {

 int dataSize=1024*1024; int contador=0;
 int iteraciones=filas*6; double r=0;
 String archivo="C:\\Users\\Jonathan\\TSPLIB\\fl1577.xlsx";
 FileInputStream fileInStream=new FileInputStream(archivo);
 XSSFWorkbook workBook=new XSSFWorkbook(fileInStream);
 XSSFSheet selSheet=workBook.getSheetAt(0);
 int filas=selSheet.getPhysicalNumberOfRows();
 int matrizpesos[][]=new int[filas][filas];
 int matrizclone[][]=new int[filas][filas];
 int matrizpesos2[][]=new int[filas][filas];
 int matrizclone2[][]=new int[filas][filas];
 int matrizanswer[][]=new int[filas][filas];
 int indFila = -1; int indCol = -1;
 Iterator rowIterator = selSheet.iterator();

while (rowIterator.hasNext()) {
  indFila+= 1;  indCol=-1;
  Row row=(Row) rowIterator.next();
  Iterator cellIterator = row.cellIterator();
  while (cellIterator.hasNext()){
      indCol +=1;
      org.apache.poi.ss.usermodel.Cell cell =
      org.apache.poi.ss.usermodel.Cell) cellIterator.next();
     if(cell.getCellType()==CellType.STRING){
       matrizpesos[indFila][indCol] = Math.toIntExact_
       (Long.parseLong(cell.getStringCellValue()));
     }
     else if(cell.getCellType()==CellType.NUMERIC){
       matrizpesos[indFila][indCol] = Math.toIntExact_
       ((long)cell.getNumericCellValue());
        matrizclone[indFila][indCol]=
        Math.toIntExact((long)cell.getNumericCellValue());
        matrizpesos2[indFila][indCol]=
        Math.toIntExact((long)cell.getNumericCellValue());
        matrizclone2[indFila][indCol]=
        Math.toIntExact((long)cell.getNumericCellValue());
     }
  }
}
```

BLOQUE I — $O(1)$

BLOQUE II — $O(n^2)$

```
//Inicio de Obtengo ruta del VMC
int pos[]=new int [filas]; int pos2[]=new int [filas];
int posnew[]=new int [filas];  int Dsuma[]=new int [filas];
pos[0]=1;  pos2[0]=pos[0];  System.out.println(" ---- ");
for (int i=1; i<filas; i++){ matrizpesos2[i][pos2[0]]=0; }
for (int i=1; i<filas; i++){ int minimo2 = 1000000000;
    for (int j = 1; j < filas; j++){
       if(minimo2>matrizpesos2[pos2[i-1]][j] &
            matrizpesos2[pos2[i-1]][j]!=0){
          minimo2 = matrizpesos2[pos2[i-1]][j];  pos2[i]=j;
       }
    }
     for (int k=1; k<filas; k++){matrizpesos2[k][pos2[i]]=0;}
  pos2[filas-1]=pos2[0];
}
System.out.println("Ruta del VMC original: ");
System.out.println("iteraciones= "+ iteraciones);
for(int i = 0;i < filas;i++){ matrizanswer[0][i]=pos2[i]; }
System.out.println("");    int sumax=0;
for(int i = 0;i < filas-1;i++){ sumax = sumax +
matrizclone[pos2[i]][pos2[i+1]]; }
sumax=sumax+matrizclone[pos2[filas-2]][pos2[0]];Dsuma[0]=sumax;
//Fin de obtengo ruta del VMC
```

BLOQUE III

$O(n^2)$

```
//INICIO DE SACRIFICIO CORTOPLACISTA
for (int d = 1; d < filas-4; d++){int indFila2=-1;
  int indCol2=-1; Iterator rowIterator2=selSheet.iterator();
  while(rowIterator2.hasNext()){indFila2+=1; indCol2=-1;
    Row row2 = (Row)  rowIterator2.next();
    Iterator cellIterator2 = row2.cellIterator();
    while (cellIterator2.hasNext()) { indCol2 +=1;
     org.apache.poi.ss.usermodel.Cell cell =
     (org.apache.poi.ss.usermodel.Cell) cellIterator2.next();
       if(cell.getCellType()==CellType.STRING)
          matrizpesos[indFila2][indCol2]=Math.toIntExact
          (Long.parseLong(cell.getStringCellValue()));
       else if(cell.getCellType()==CellType.NUMERIC)
         matrizpesos[indFila2][indCol2]=
         Math.toIntExact((long)cell.getNumericCellValue());
    }
  }
```

BLOQUE IV

$O(n^2)$

```
int temp1 = matrizpesos[pos2[d-1]][pos2[d]];
matrizpesos[pos2[d-1]][pos2[d]]=0;
  for (int i=1; i<filas; i++){ matrizpesos[i][pos[0]]=0; }
  for (int i=1; i<filas; i++){ int minimo = 1000000000;
     for (int j=1; j<filas; j++){
         if (minimo > matrizpesos[pos[i-1]][j] &
            matrizpesos[pos[i-1]][j]!=0){
             minimo = matrizpesos[pos[i-1]][j]; pos[i]=j;}
     }
     for (int k=1; k<filas; k++){matrizpesos[k][pos[i]]=0;}
     pos[filas-1]=pos[0];
  }  int suma=0;
  for(int i=0;i<filas;i++){  matrizanswer[d][i]=pos[i]; }
  for(int i=0;i<filas-1;i++){  suma=suma+
      matrizclone[pos[i]][pos[i+1]]; }   Dsuma[d]=suma;
  System.out.println("La distancia asociada para el "+d+"_
  punto de apalancamiento es: "+suma);
} //FINALIZO SACRIFICIO CORTOPLACISTA
```

BLOQUE V

$O(n^2)$

```
int posi=0, min= 999999999;
for (int i=0; i<filas-4; i++){
    if (min>Dsuma[i]) { min=Dsuma[i]; posi=i;  }
}
System.out.println("La menor distancia es de:"+Dsuma[posi]);
for (int i = 0; i < filas; i++){
    if (i<filas-1){ posnew[i]=matrizanswer[posi][i];  }
    else if (i==filas-1){ posnew[filas-1]=1;    }
}
for (int i = 0; i < filas; i++){
    System.out.print(posnew[i]+"-");
}
System.out.println("");
System.out.println("El punto de apalancamiento_
corresponde al nodo: "+posi);
```

BLOQUE VI

O(n)

```
//APERTURO CODIGO PARA 2-OPT
int cambio=1; int Compara[]= new int [iteraciones];
for (int cont=0;cont<iteraciones;cont++){
double change,min_change=0; int
b=0,c=0,j,min_i=0,min_j=0,costoactual,costonuevo;
    for(int i=0;i<=filas-4;i++){
       for (j=i+2;j<=filas-2;j++){
        costonuevo = matrizclone[posnew[i]][posnew[j]]+
        matrizclone[posnew[i+1]][posnew[j+1]];
        costoactual = matrizclone[posnew[i]][posnew[i+1]]+
        matrizclone[posnew[j]][posnew[j+1]];
        change = (1+r)*costonuevo-costoactual;
          if (change < min_change){min_change=change;
             b=posnew[j]; c=posnew[i+1]; min_i=i+1; min_j=j;
               }
         }
    }
    posnew[min_i]=b; posnew[min_j]=c;
    int suma2=0; posnew[0]=1;
    for(int i=0;i<filas-1;i++) {
    suma2=suma2+matrizclone[posnew[i]][posnew[i+1]];
    }
  Compara[cont]=suma2;
}
```

BLOQUE VII

$O(n^3 log(n))$

```
System.out.println("minimo= "+ Compara[iteraciones-1] +
", "+ " r = "+ r);
for (int i=0; i<filas; i++){
    System.out.print(posnew[i]+"-")
}
  }  // Fin de public static void
} // Fin de public class VMC_SCA_2Opt
```

BLOQUE VIII

O(n)

Anexo 15. Análisis de la complejidad a priori del SCA(1)_Uni2Opt y SCA(1)_Uni2Opt_r

Se hace hincapié en que el algoritmo SCA(1)_Uni2Opt es un caso particular cuando r = 0 del SCA(1)_Uni2Opt_r. Por lo cual, ambas versiones presentan la misma complejidad según notación Big O.

A continuación, para la determinación de la complejidad algorítmica tanto del SCA(1)_Uni2Opt y el SCA(1)_Uni2Opt_r, según el método de la Big O, se particionó su código, implementado en NetBeans mediante lenguaje de programación Java, en bloques descritos en el Anexo 14, de tal manera que se pueda determinar la complejidad de los mencionados bloques para luego calcular la complejidad total del algoritmo.

Bloque I: Se tiene instrucciones principalmente de asignación y declaración de variables.

Complejidad = O(1)

Bloque II: Se tiene instrucciones para lectura de los elementos de la matriz de costos, por medio de dos While.

Complejidad = $O(n)O(n) = O(n^2)$

Bloque III: Código para obtener ruta del Vecino más Cercano

$$
\begin{aligned}
&Complejidad = O(1)+O(n)+O(n)O(n)+O(n)O(n)+O(1)+O(n)+O(1)+O(n)+O(1)\\
&= O(1) + O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)\\
&= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)\\
&= O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)\\
&= O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)\\
&= O(n^2) + O(n) + O(1) + O(n) + O(1)\\
&= O(n^2) + O(1) + O(n) + O(1)\\
&= O(n^2) + O(n) + O(1)\\
&= O(n^2)
\end{aligned}
$$

Bloque IV: Lectura de elementos de la matriz de costos para el SCA(1).

Complejidad = $O(n) \times O(n) = O(n^2)$

Bloque V: Prohíbe arco según nodo de apalancamiento + Pensamiento del Vecino más cercano = Sacrificio Cortoplacista

$$
\begin{aligned}
Complejidad &= O(1) + O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(n) + O(1) \\
&= O(1) + O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(1) \\
&= O(n^2) + O(1) \\
&= O(n^2)
\end{aligned}
$$

Bloque VI: Selección de la mejor ruta de las construidas por el SCA.

$$Complejidad = O(1) + O(n) + O(1) + O(n) + O(n) + O(1) = O(n)$$

Bloque VII: Ejecución de heurística 2Opt para mejorar la ruta determinada en el Bloque VI (cuya complejidad se determinó en el Anexo 24).

$$Complejidad = O(n^3 log(n))$$

Bloque VIII: Imprime la secuencia de nodos que se debe seguir para la ruta construida y la respectiva distancia recorrida.

$$Complejidad = O(1) + O(n) = O(n)$$

De tal manera, se tiene el siguiente código:

De tal manera que se tiene el siguiente código:

```
public class VMC_SCA_2Opt {

public static void main(String[] args) throws IOException, BiffException {
 BLOQUE I  - Complejidad = O(1)
 BLOQUE II - Complejidad = O(n^2)
 BLOQUE III - Complejidad = O(n^2)

 // Inicio de sacrificio cortoplacista

 for (int d=Linferior;d<Lsuperior;d++) {

 BLOQUE IV - Complejidad = O(n^2)
 BLOQUE V - Complejidad = O(n^2)

 }    // Cierro sacrificio cortoplacista

 BLOQUE VI - Complejidad = O(n)
 BLOQUE VII - Complejidad = O(n^3log(n)
 BLOQUE VIII- Complejidad = O(n)
  }
}
```

O(1) + O(n²)+ O(n²) = O(n²)

+

O(n)×[O(n²) + O(n²)] = O(n) × [O(n²)] = O(n³)

+

O(n)+O(n³log(n))+O(n) = O(n³log(n))+O(n) = O(n³log(n))

O(n³log(n))

Donde, como resultado final, se obtiene:

Complejidad del algoritmo SCA(1)_Uni2Opt = $O(n^2) + O(n^3) + O(n^3 \log(n))$

Complejidad del SCA(1)_Uni2Opt_r y SCA(1)_Uni2Opt = $O(n^3 \log(n))$

Anexo 16. Código del SCA(5)_Uni2Opt

Para facilitar el análisis de la complejidad algorítmica, se dividió el código en bloques.

```
package vmc_sca_2opt;
import java.lang.management.ManagementFactory; import jxl.Workbook;
import java.lang.management.MemoryMXBean; import java.util.Iterator;
import java.lang.management.MemoryUsage; import java.io.IOException;
import java.io.FileInputStream; import jxl.Cell;
import jxl.read.biff.BiffException;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.xssf.usermodel.XSSFSheet; import java.io.File;
import org.apache.poi.xssf.usermodel.XSSFWorkbook; import jxl.Sheet;

public class VMC_SCA_2Opt {

private static String sRuta;

public static void main(String[] args) throws IOException,
BiffException {

int dataSize= 1024*1024;

  Obtener_Ruta(1,0,0,0,0);  Obtener_Ruta(1,1,0,0,0);
  Obtener_Ruta(1,0,1,0,0);  Obtener_Ruta(1,1,1,0,0);
  Obtener_Ruta(0,1,1,1,1);  Obtener_Ruta(1,0,0,1,0);
  Obtener_Ruta(0,1,0,1,1);  Obtener_Ruta(1,1,0,1,0);
  Obtener_Ruta(1,0,1,1,1);  Obtener_Ruta(1,1,0,1,1);
  Obtener_Ruta(1,1,1,0,1);  Obtener_Ruta(1,0,0,0,1);
  Obtener_Ruta(1,1,0,0,1);  Obtener_Ruta(1,0,0,1,1);
  Obtener_Ruta(1,0,1,0,1);  Obtener_Ruta(1,1,1,1,1);

Runtime runtime= Runtime.getRuntime();
System.out.println("MTotal:"+runtime.totalMemory()/dataSize+"MB");
System.out.println("MFree:"+runtime.freeMemory()/dataSize+"MB");
System.out.println("MUsed:"+(double)((runtime.totalMemory()-
runtime.freeMemory())*1000/dataSize)/1000 + "MB");
}

private static String Obtener_Ruta(Integer bn1, Integer bn2, Integer
bn3, Integer bn4, Integer bn5) throws IOException, BiffException{

int dataSize= 1024*1024;
double r=0;
String archivo="C:\\Users\\Jonathan\\TSPLIB\\st70.xlsx";
FileInputStream fileInStream = new FileInputStream(archivo);
XSSFWorkbook workBook = new XSSFWorkbook(fileInStream);
XSSFSheet selSheet = workBook.getSheetAt(0);
int filas = selSheet.getPhysicalNumberOfRows();
int matrizpesos[][] = new int[filas][filas];
int matrizclone[][] = new int[filas][filas];
int matrizpesos2[][] = new int[filas][filas];
int matrizanswer[][] = new int[filas][filas];
int iteraciones = filas*6;
int indFila = -1;
int indCol = -1;
Iterator rowIterator = selSheet.iterator();
```

BLOQUE IX

BLOQUE I — **O(1)**

```
while (rowIterator.hasNext()){ indFila += 1;  indCol =-1;
      Row row = (Row) rowIterator.next();
      Iterator cellIterator = row.cellIterator();
     while (cellIterator.hasNext()){  indCol +=1;
      org.apache.poi.ss.usermodel.Cell cell =
      org.apache.poi.ss.usermodel.Cell) cellIterator.next();
        if(cell.getCellType()==CellType.STRING)
           matrizpesos[indFila][indCol]=
            Math.toIntExact(Long.parseLong(
            cell.getStringCellValue()));
        else if(cell.getCellType()==CellType.NUMERIC)
            matrizpesos[indFila][indCol]=Math.toIntExact
            ((long)cell.getNumericCellValue());
            matrizclone[indFila][indCol]=Math.toIntExact
            ((long)cell.getNumericCellValue());
            matrizpesos2[indFila][indCol]=Math.toIntExact
            ((long)cell.getNumericCellValue());
     }
}
```

BLOQUE II

$O(n^2)$

```
// Inicio de algoritmo de VMC
int pos[]= new int[filas]; int pos2[]= new int[filas];
int posnew[]= new int[filas]; int Dsuma[]=new int[filas];
pos[0]=1; pos2[0]=pos[0];
System.out.println(" ------------- ");
for (int i=1;i<filas;i++)
{
   matrizpesos2[i][pos2[0]]=0;
}
for (int i=1;i<filas;i++)
{
   int minimo2 = 1000000000;
    for (int j = 1; j < filas; j++)
    {
       if (minimo2 > matrizpesos2[pos2[i-1]][j] &
            matrizpesos2[pos2[i-1]][j]!=0){
         minimo2=matrizpesos2[pos2[i-1]][j]; pos2[i]=j;
       }
    }
    for (int k=1;k < filas; k++)
    {
     matrizpesos2[k][pos2[i]]=0;
    }
  pos2[filas-1]=pos2[0];
}
System.out.println("Ruta del VMC original: ");
for(int i=0;i<filas;i++)
{
  matrizanswer[0][i]=pos2[i];
}
System.out.println(""); int sumax=0;
for(int i=0;i<filas-1;i++)
{
  sumax = sumax+matrizclone[pos2[i]][pos2[i+1]];
}
 Sumax = sumax+ matrizclone[pos2[filas-2]][pos2[0]];
 Dsuma[0]=sumax;
//Fin de algoritmo de VMC
```

BLOQUE III

$O(n^2)$

```
//Inicio de Sacrificio Cortoplacista
for (int d = 1; d < filas-4; d++)
{ int indFila2 = -1; int indCol2 = -1;
  Iterator rowIterator2 = selSheet.iterato r
  while (rowIterator2.hasNext())
   { indFila2 += 1; indCol2 =-1;
     Row row2 = (Row) rowIterator2.next();
     Iterator cellIterator2 = row2.cellIterator();
    while (cellIterator2.hasNext())
    {  indCol2 +=1;
     org.apache.poi.ss.usermodel.Cell cell = (org.apache.
      poi.ss.usermodel.Cell) cellIterator2.next();
      if(cell.getCellType()==CellType.STRING)
         matrizpesos[indFila2][indCol2]= Math.toIntExact
         (Long.parseLong(cell.getStringCellValue()));
      else if(cell.getCellType()==CellType.NUMERIC)
         matrizpesos[indFila2][indCol2]= Math.toIntExact
         ((long)cell.getNumericCellValue());
    }
  }

  if(bn1==1){matrizpesos[pos2[d-1]][pos2[d]]=0;}
  if(bn2==1){matrizpesos[pos2[d]][pos2[d+1]]=0;}
  if(bn3==1){matrizpesos[pos2[d+1]][pos2[d+2]]=0;}
  if(bn4==1){matrizpesos[pos2[d+2]][pos2[d+3]]=0;}
  if(bn5==1){matrizpesos[pos2[d+3]][pos2[d+4]]=0;}

  for (int i = 1; i< filas; i++)
  {
      matrizpesos[i][pos[0]]=0;
  }
  for (int i = 1; i < filas; i++)
  {
      int minimo = 1000000000;
      for (int j = 1; j < filas; j++)
      {
         if (minimo > matrizpesos[pos[i-1]][j] &
            matrizpesos[pos[i-1]][j]!=0)
         {
            minimo = matrizpesos[pos[i-1]][j]; pos[i]=j;
         }
      }
      for (int k = 1; k< filas; k++)
      {
           matrizpesos[k][pos[i]]=0;
      }
    pos[filas-1]=pos[0];
  }
  int suma=0;
  for(int i = 0;i<filas;i++)
  {
      matrizanswer[d][i]=pos[i];
  }
  for(int i = 0;i<filas-1;i++)
  {
      suma = suma+matrizclone[pos[i]][pos[i+1]];
  }
  Dsuma[d]=suma;
}
// FINALIZO SCA
```

BLOQUE IV

BLOQUE V

BLOQUE VI

$O(n^2)$

$O(1)$

$O(n^2)$

```
int posi=0, min= 999999999;
for (int i = 0; i < filas-4; i++){
     if (min>Dsuma[i]){min=Dsuma[i]; posi=i;}
}
System.out.println("La menor distancia es:"+Dsuma[posi]);
for (int i = 0; i < filas; i++){
     if (i<filas-1){posnew[i]=matrizanswer[posi][i];}
     else if (i==filas-1){posnew[filas-1]=1; }
}
for (int i = 0; i < filas; i++){
    System.out.print(posnew[i]+"-");
}
System.out.println();
System.out.println("Nodo apalancamiento = "+posi);
```

O(n)

BLOQUE VII

```
//APERTURO CODIGO PARA 2-OPT
int cambio=1;
int Compara[]= new int [iteraciones];
for(int cont=0;cont<iteraciones;cont++){
   double change,min_change= 0; int b=0, c=0;
   int j, min_i=0,min_j=0; int costoactual, costonuevo;
    for(int i=0;i<=filas-4;i++){
        for (j=i+2;j<=filas-2;j++){
          costonuevo = matrizclone[posnew[i]][posnew[j]]+
                       matrizclone[posnew[i+1]][posnew[j+1]];
          costoactual = matrizclone[posnew[i]][posnew[i+1]]+
                        matrizclone[posnew[j]][posnew[j+1]];
          change = (1+r)*costonuevo-costoactual;

            if (change < min_change){
               min_change = change; b=posnew[j];
               c=posnew[i+1]; min_i=i+1; min_j=j;
            }
        }
    }
  posnew[min_i]=b; posnew[min_j]=c; int suma2=0; posnew[0]=1;
    if  (cont==iteraciones-1) {
        for(int i=0;i<filas;i++){
            System.out.print(posnew[i]+"-");
        }
        System.out.print("\n");
    }
    for(int i = 0;i<filas-1;i++){
         suma2 = suma2+matrizclone[posnew[i]][posnew[i+1]];
    }
      Compara[cont]=suma2;
} //CIERRO CODIGO PARA 2-OPT

System.out.println("minimo="+Compara[iteraciones-1]+",r="+r);

return sRuta;
```

BLOQUE VIII

$O(n^3 log(n))$

```
} // Fin de private static String Obtener_Ruta

} // Fin de public class VMC_SCA_2Opt
```

Anexo 17. Análisis de la complejidad del SCA(5)_Uni2Opt

Se hace hincapié en que el algoritmo SCA(5)_Uni2Opt es una ampliación (o extensión) del SCA(1)_Uni2Opt, donde se realizan a lo más cinco sacrificios cortoplacistas en la construcción de una ruta, la cual es guiada por el denominado vector prohibición. Por lo cual, se presenta una complejidad de $m \times O(n^3 \log(n))$, donde "m" representa la cantidad de vectores prohibición definidos y, por tanto, según notación Big O, se tendría una complejidad $O(n^3 \log(n))$.

A continuación, para la determinación de la complejidad algorítmica del SCA(5)_Uni2Opt, según el método de la Big O, se particionó su código, implementado en NetBeans mediante lenguaje de programación Java, en bloques descritos en el Anexo 16, de tal manera que se determine la complejidad de los mencionados bloques, para luego calcular la complejidad total del algoritmo.

Bloque I: Se tiene instrucciones principalmente de asignación y declaración de variables.

Complejidad = O(1)

Bloque II: Se tiene instrucciones para lectura de los elementos de la matriz de costos, por medio de dos While.

Complejidad = $O(1) + O(n)O(n) = O(n^2)$

Bloque III: Código para obtener ruta del Vecino más Cercano.

$$Complejidad = O(1)+O(n)+O(n)O(n)+O(n)O(n)+O(1)+O(n)+O(1)+O(n)+O(1)$$
$$= O(1) + O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(1) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(n) + O(1) + O(n) + O(1)$$
$$= O(n^2) + O(1) + O(n) + O(1)$$

$= O(n^2)$

Bloque IV: Lectura de elementos de la matriz de costos para el SCA(5).

Complejidad = $O(n) \times O(n) = O(n^2)$

Bloque V: Condicional para la activación de los distintos vectores prohibición

Complejidad = O(1)

Bloque VI: Construye rutas en base al Sacrificio Cortoplacista, donde los vectores prohibición que guían la búsqueda fueron definidos en el Bloque V.

$$\begin{aligned}
&\textit{Complejidad} = O(n) + O(n)O(n) + O(n)O(n) + O(1) + O(n) + O(n) + O(1) \\
&= O(n) + O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(1) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(n) + O(1) \\
&= O(n^2) + O(n) + O(1) \\
&= O(n^2) + O(1) \\
&= O(n^2)
\end{aligned}$$

Bloque VII: Se obtiene la mejor ruta de las construidas en el Bloque VI

$$\begin{aligned}
\textit{Complejidad} &= O(1) + O(n) + O(1) + O(n) + O(n) + O(1) \\
&= O(n)
\end{aligned}$$

Bloque VIII: Ejecución de la heurística 2Opt para mejorar la ruta determinada en el Bloque VII (cuya complejidad se determinó en el Anexo 24).

$\textit{Complejidad} = O(n^3 log(n))$

De tal manera que se tiene el siguiente código:

```
private static String Obtener_Ruta(Integer bn1, Integer bn2, Integer
bn3, Integer bn4, Integer bn5) throws IOException, BiffException{

   BLOQUE I  - Complejidad = O(1)
   BLOQUE II - Complejidad = O(n^2)
   BLOQUE III - Complejidad = O(n^2)

 // Inicio de sacrificio cortoplacista
   for (int d=Linferior;d<Lsuperior;d++) {
   BLOQUE IV - Complejidad = O(n^2)
   BLOQUE V - Complejidad = O(1)
   BLOQUE VI - Complejidad = O(n^2)
   }    // Cierro sacrificio cortoplacista

   BLOQUE VII - Complejidad = O(n)
   BLOQUE VIII - Complejidad = O(n^3log(n))
}
```

$= O(1) + O(n^2) + O(n^2)$

$= O(n^2)$

$+$

$= O(n) \times [O(n^2) + O(1) + O(n^2)]$

$= O(n) \times [O(n^2)]$

$= O(n^3)$

$+$

$= O(n) + O(n^3)$

$= O(n^3 log(n))$

$O(n^3 log(n))$

Bloque IX: Se continúa con el siguiente vector prohibición para volver a ejecutar los Bloques I al VIII.

Complejidad = m × Complejidad del Algoritmo = m ×O(n³log(n)) = O(n³log(n))

Donde, como resultado final, se obtiene:

Complejidad del algoritmo SCA(5)_Uni2Opt = O(n³log(n))

Anexo 18. Determinación de criterios para ampliación de selección de instancias simétricas de TSPLIB

Para ampliar nuestras conclusiones a instancias de mayor tamaño, se decidió ampliar la cantidad de instancias a las cuales ejecutamos nuestras versiones Heavy y Soft (ver Tabla 25). Por lo cual, por criterio de conveniencia se decidió seleccionar instancias del Problema del Agente Viajero Simétrico con una cantidad de nodos menor o igual a 1002 y que el tipo de cálculo de los elementos de la matriz de costos se haya realizado según lo indicado:

- **EUC2D:** El método de cálculo de la matriz de costos consiste en la distancia euclidiana, redondeada a valores enteros.
- **MATRIX:** La matriz de costos es brindada por la librería virtual TSPLIB, por lo cual no se tiene errores de redondeo.
- **ATT:** El método de cálculo de la matriz de costos consiste en la distancia pseudo euclidiana, se da exclusivamente para las instancias de att48 y att532. Donde se redondean a valores enteros.

Tabla 25

Detalle de las 65 instancias simétricas TSPLIB seleccionadas

Instancia	Instancia	Instancia	Instancia	Instancia	Instancia
1.gr17	12.Brazil58	23.eil101	34.kroB150	45.pr226	56.att532
2.gr21	13.st70	24.lin105	35.pr152	46.gil262	57.si535
3.gr24	14.eil76	25.pr107	36.u159	47.pr264	58.pa561
4.fri26	15.pr76	26.gr120	37.si175	48.a280	59.u574
5.dantzig42	16.rat99	27.pr124	38.brg180	49.pr299	60.rat575
6.Swiss42	17.kroa100	28.bier127	39.rat195	50.lin318	61.p654
7.att48	18.kroB100	29.ch130	40.d198	51.rd400	62.d657
8.gr48	19.kroC100	30.pr136	41.kroA200	52.fl417	63.u724
9.hk48	20.kroD100	31.pr144	42.kroB200	53.pr439	64.Rat783
10.eil51	21.kroE100	32.ch150	43.ts225	54.Pcb442	65.Pr1002
11.berlin52	22.rd100	33.kroA150	44.tsp225	55.d493	

Anexo 19. Ampliación del análisis para la determinación del criterio de parada

Para determinar un criterio de parada que no requiera hacer un análisis previo para determinar el número de ejecuciones necesarias para la convergencia del algoritmo SCA(1)_Multi2Opt, se planteó realizar un análisis más extenso de las instancias seleccionadas en la librería virtual TSPLIB. Esa extensión del análisis se realiza con el objetivo de poder usarlo como punto de partida para posteriores investigaciones que utilicen a la librería TSPLIB, y poder definir un criterio de parada en general para matrices simétricas, el cual incluso podría extenderse a matrices asimétricas.

A continuación, se elaboró la escala de color del número de ejecuciones 2Opt, dentro del criterio Multi2Opt, con cantidades de 100 a 8000 ejecuciones, con cierta separación de intervalos definida por conveniencia, a criterio propio en la presente investigación, el cual es resumido en la Tabla 26.

Tabla 26

Escala de color de mejor solución disponible según número de ejecuciones 2Opt para el Multi2Opt

Instancia	100	200	300	400	500	600	1000	1100	1500	2000	2100	2500	3000	3500	4000	4500	5000	6000	8000
1.gr17	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085	2085
2.gr21	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707	2707
3.gr24	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294	1294
4.fri26	953	953	953	953	953	953	953	953	953	953	953	953	953	953	953	953	953	953	953
5.dantzig42	699	699	699	699	699	699	699	699	699	699	699	699	699	699	699	699	699	699	699
6.Swiss42	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273	1273
7.att48	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648	10648
8.gr48	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097	5097
9.hk48	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461	11461
10.eil51	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430	430
11.berlin52	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715	7715
12.Brazil58	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400	25400
13.st70	692	688	688	688	688	688	688	688	688	688	688	688	688	688	688	688	688	688	688
14.eil76	552	548	548	548	548	548	548	548	548	548	548	548	548	548	548	548	548	548	548
15.pr76	114151	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254	109254
16.rat99	1286	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258	1258
17.kroa100	22066	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474	21474
18.kroB100	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349	22349
19.kroC100	21387	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162	21162
20.kroD100	22236	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861	21861
21.kroE100	22620	22364	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355	22355
22.rd100	8149	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018	8018

Tabla 26

Escala de color de mejor solución disponible según número de ejecuciones 2Opt para el Multi2Opt

Instancia	100	200	300	400	500	600	1000	1100	1500	2000	2100	2500	3000	3500	4000	4500	5000	6000	8000
23.eil101	643	638	638	638	638	638	638	638	638	638	638	638	638	638	638	638	638	638	638
24.lin105	14775	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484	14484
25.pr107	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303	44303
26.gr120	7587	7224	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069	7069
27.pr124	60542	59990	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246	59246
28.bier127	124260	119553	119384	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928	118928
29.ch130	6433	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269	6269
30.pr136	108412	103745	102117	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384	100384
31.pr144	59714	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143	59143
32.ch150	6927	6765	6602	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563	6563
33.kroA150	30186	28294	27799	27642	27642	27493	27493	27493	27493	27493	27493	27493	27493	27493	27493	27493	27493	27493	27493
34.kroB150	30089	27100	26893	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651	26651
35.pr152	76124	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606	74606
36.u159	45468	43820	42566	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547	42547
37.si175	22025	21847	21673	21509	21484	21441	21441	21441	21441	21441	21441	21441	21441	21441	21441	21441	21441	21441	21441
38.brg180	3870	3330	3060	2820	2710	2480	2170	2160	2010	2000	2000	1970	1970	1970	1970	1970	1970	1970	1970
39.rat195	2608	2488	2399	2397	2371	2354	2354	2354	2354	2354	2354	2354	2354	2354	2354	2354	2354	2354	2354
40.d198	17048	16740	16031	16031	15955	15926	15926	15926	15926	15926	15926	15926	15926	15926	15926	15926	15926	15926	15926
41.kroA200	31323	30255	30115	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591	29591
42.kroB200	34219	31415	31067	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651	30651
43.ts225	139769	135124	132021	131211	127749	127512	127512	127512	127512	127512	127512	127512	127512	127512	127512	127512	127512	127512	127512
44.tsp225	4560	4334	4279	4106	4076	4096	3993	3976	3976	3976	3976	3976	3976	3976	3976	3976	3976	3976	3976

Tabla 26

Escala de color de mejor solución disponible según número de ejecuciones 2Opt para el Multi2Opt

Instancia	100	200	300	400	500	600	1000	1100	1500	2000	2100	2500	3000	3500	4000	4500	5000	6000	8000
45.pr226	87499	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285	81285
46.gil262	2750	2653	2501	2475	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440	2440
47.pr264	55557	53795	51923	51144	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985	50985
48.a280	3037	2897	2765	2738	2721	2702	2670	2670	2664	2664	2664	2664	2664	2664	2664	2664	2664	2664	2664
49.pr299	56211	55167	53496	50453	50453	50170	49217	49217	49217	49217	49217	49217	49217	49217	49217	49217	49217	49217	49217
50.lin318	50211	49349	46894	46734	45030	43990	43450	43450	43450	43450	43450	43450	43450	43450	43450	43450	43450	43450	43450
51.rd400	18075	17491	17470	17234	16586	15924	15825	15796	15663	15641	15641	15641	15641	15641	15641	15641	15641	15641	15641
52.fl417	13592	13431	12843	13047	12748	12641	12139	12139	12092	12092	12092	12092	12092	12092	12092	12092	12092	12092	12092
53.pr439	125111	121182	118022	118187	112530	111756	111756	111756	108569	108569	108569	108569	108569	108569	108569	108569	108569	108569	108569
54.Pcb442	58042	57143	55598	56445	55416	53905	52228	52532	52107	52048	51977	51977	51977	51977	51977	51977	51977	51977	51977
55.d493	39868	38397	38161	37416	36846	36499	36105	35913	35913	35913	35913	35913	35913	35913	35913	35913	35913	35913	35913
56.att532	32923	31980	32458	30810	32016	30452	29850	29077	28588	28543	28543	28416	28416	28416	28416	28416	28416	28416	28416
57.si535	49582	49754	49333	49059	48914	48863	48778	48778	48756	48756	48756	48756	48756	48756	48756	48756	48756	48756	48756
58.pa561	3243	3250	3216	3180	3141	3124	2944	2903	2899	2879	2879	2868	2854	2854	2854	2854	2854	2854	2854
59.u574	46162	44947	43942	43617	42824	42484	41385	39345	38699	38525	38525	38525	38318	38318	38318	38318	38318	38318	38318
60.rat575	8364	8185	8120	7954	7953	7750	7531	7569	7144	7081	7081	7041	7041	7041	7041	7041	7041	7041	7041
61.p654	37456	36455	35981	35568	35558	35567	35404	35254	35254	35254	35254	35254	35254	35254	35254	35254	35254	35254	35254
62.d657	57755	58541	58205	57143	56312	56191	55208	53154	53003	51139	50957	50735	50510	50238	50238	50238	50238	50238	50238
63.u724	50546	49988	49646	49704	49318	48935	47305	46438	44600	43934	43934	43934	43905	43891	43891	43891	43891	43891	43891
64.Rat783	10545	10534	10525	10140	10286	10310	10104	10050	9842	9591	9473	9224	9151	9151	9119	9119	9119	9119	9119
65.Pr1002	312095	316029	334081	310155	306413	307100	299462	296681	295098	278424	278498	275016	269719	269719	269719	269719	269719	269547	269547

Nota. En la escala de color, los números sombreados con rojo corresponden a los menores valores encontrados según el número de ejecuciones 2Opt respectiva.

A partir de la Tabla 26, se construyó la Tabla 27, la cual muestra el número mínimo de ejecuciones para que el Multi2Opt, combinado con el SCA(1), converja a la mejor solución disponible para las 65 instancias TSPLIB seleccionadas en el Anexo 18.

Tabla 27

Cantidad mínima de ejecuciones 2Opt para el Multi2Opt

Instancia	Nodos	TIPO	Cantidad Mínima	Instancia	Nodos	TIPO	Cantidad Mínima	Instancia	Nodos	TIPO	Cantidad Mínima
1.gr17	17	MATRIX	100	23.eil101	101	EUC2D	200	45.pr226	226	EUC2D	200
2.gr21	21	MATRIX	100	24.lin105	105	EUC2D	200	46.gil262	262	EUC2D	500
3.gr24	24	MATRIX	100	25.pr107	107	EUC2D	100	47.pr264	264	EUC2D	500
4.fri26	26	MATRIX	100	26.gr120	120	MATRIX	300	48.a280	280	EUC2D	1500
5.dantzig42	42	MATRIX	100	27.pr124	124	EUC2D	300	49.pr299	299	EUC2D	1000
6.Swiss42	42	MATRIX	100	28.bier127	127	EUC2D	400	50.lin318	318	EUC2D	1000
7.att48	48	ATT	100	29.ch130	130	EUC2D	200	51.rd400	400	EUC2D	2000
8.gr48	48	MATRIX	100	30.pr136	136	EUC2D	400	52.fl417	417	EUC2D	1500
9.hk48	48	MATRIX	100	31.pr144	144	EUC2D	200	53.pr439	439	EUC2D	1500
10.eil51	51	EUC2D	100	32.ch150	150	EUC2D	400	54.Pcb442	442	EUC2D	2100
11.berlin52	52	EUC2D	100	33.kroA150	150	EUC2D	600	55.d493	493	EUC2D	1100
12.Brazil58	58	MATRIX	100	34.kroB150	150	EUC2D	400	56.att532	532	ATT	2500
13.st70	70	EUC2D	200	35.pr152	152	EUC2D	200	57.si535	535	MATRIX	1500
14.eil76	76	EUC2D	200	36.u159	159	EUC2D	400	58.pa561	561	MATRIX	3000
15.pr76	76	EUC2D	200	37.si175	175	MATRIX	600	59.u574	574	EUC2D	3000
16.rat99	99	EUC2D	200	38.brg180	180	MATRIX	2500	60.rat575	575	EUC2D	2500
17.kroa100	100	EUC2D	200	39.rat195	195	EUC2D	600	61.p654	654	EUC2D	1100
18.kroB100	100	EUC2D	100	40.d198	198	EUC2D	600	62.d657	657	EUC2D	3500
19.kroC100	100	EUC2D	200	41.kroA200	200	EUC2D	400	63.u724	724	EUC2D	3500
20.kroD100	100	EUC2D	200	42.kroB200	200	EUC2D	400	64.Rat783	783	EUC2D	5000
21.kroE100	100	EUC2D	300	43.ts225	225	EUC2D	600	65.Pr1002	1002	EUC2D	6000
22.rd100	100	EUC2D	200	44.tsp225	225	EUC2D	1100				

Con la información de la Tabla 27, se determinó una cota superior del número de ejecuciones el Intercambio 2Opt en su versión Best Improvement, donde el valor de la constante "k" $\in Z^+$ es probado por ensayo y error según lo indicado en la Tabla 28. La cota se define como:

Cota=(Número de nodos del grafo)×k, o su equivalente = (Orden del Grafo)×k

Tabla 28

Determinación de constante "k" para cota de # de ejecuciones 2Opt en el Multi2Opt

Nodos	Cantidad Mínima	k= 1	k= 2	k= 3	k= 4	k= 5	k= 6	k= 7	k= 14	k= 20
17	100	17	34	51	68	85	102	119	238	340
21	100	21	42	63	84	105	126	147	294	420
24	100	24	48	72	96	120	144	168	336	480
26	100	26	52	78	104	130	156	182	364	520
42	100	42	84	126	168	210	252	294	588	840
42	100	42	84	126	168	210	252	294	588	840
48	100	48	96	144	192	240	288	336	672	960
48	100	48	96	144	192	240	288	336	672	960
48	100	48	96	144	192	240	288	336	672	960
51	100	51	102	153	204	255	306	357	714	1020
52	100	52	104	156	208	260	312	364	728	1040
58	100	58	116	174	232	290	348	406	812	1160
70	200	70	140	210	280	350	420	490	980	1400
76	200	76	152	228	304	380	456	532	1064	1520
76	200	76	152	228	304	380	456	532	1064	1520
99	200	99	198	297	396	495	594	693	1386	1980
100	100	100	200	300	400	500	600	700	1400	2000
100	200	100	200	300	400	500	600	700	1400	2000
100	200	100	200	300	400	500	600	700	1400	2000
100	200	100	200	300	400	500	600	700	1400	2000
100	200	100	200	300	400	500	600	700	1400	2000
100	300	100	200	300	400	500	600	700	1400	2000
101	200	101	202	303	404	505	606	707	1414	2020
105	200	105	210	315	420	525	630	735	1470	2100
107	100	107	214	321	428	535	642	749	1498	2140
120	300	120	240	360	480	600	720	840	1680	2400
124	300	124	248	372	496	620	744	868	1736	2480
127	400	127	254	381	508	635	762	889	1778	2540
130	200	130	260	390	520	650	780	910	1820	2600

Tabla 28

Determinación de constante "k" para cota de # de ejecuciones 2Opt en el Multi2Opt

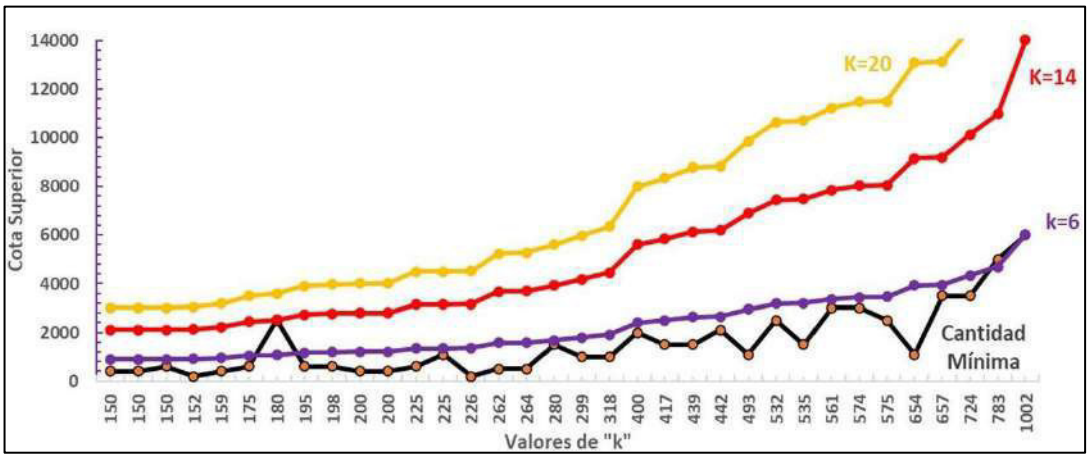
Nodos	Cantidad Mínima	k= 1	k= 2	k= 3	k= 4	k= 5	k= 6	k= 7	k= 14	k= 20
136	400	136	272	408	544	680	816	952	1904	2720
144	200	144	288	432	576	720	864	1008	2016	2880
150	400	150	300	450	600	750	900	1050	2100	3000
150	400	150	300	450	600	750	900	1050	2100	3000
150	600	150	300	450	600	750	900	1050	2100	3000
152	200	152	304	456	608	760	912	1064	2128	3040
159	400	159	318	477	636	795	954	1113	2226	3180
175	600	175	350	525	700	875	1050	1225	2450	3500
180	2500	180	360	540	720	900	1080	1260	2520	3600
195	600	195	390	585	780	975	1170	1365	2730	3900
198	600	198	396	594	792	990	1188	1386	2772	3960
200	400	200	400	600	800	1000	1200	1400	2800	4000
200	400	200	400	600	800	1000	1200	1400	2800	4000
225	600	225	450	675	900	1125	1350	1575	3150	4500
225	1100	225	450	675	900	1125	1350	1575	3150	4500
226	200	226	452	678	904	1130	1356	1582	3164	4520
262	500	262	524	786	1048	1310	1572	1834	3668	5240
264	500	264	528	792	1056	1320	1584	1848	3696	5280
280	1500	280	560	840	1120	1400	1680	1960	3920	5600
299	1000	299	598	897	1196	1495	1794	2093	4186	5980
318	1000	318	636	954	1272	1590	1908	2226	4452	6360
400	2000	400	800	1200	1600	2000	2400	2800	5600	8000
417	1500	417	834	1251	1668	2085	2502	2919	5838	8340
439	1500	439	878	1317	1756	2195	2634	3073	6146	8780
442	2100	442	884	1326	1768	2210	2652	3094	6188	8840
493	1100	493	986	1479	1972	2465	2958	3451	6902	9860
532	2500	532	1064	1596	2128	2660	3192	3724	7448	10640
535	1500	535	1070	1605	2140	2675	3210	3745	7490	10700
561	3000	561	1122	1683	2244	2805	3366	3927	7854	11220
574	3000	574	1148	1722	2296	2870	3444	4018	8036	11480
575	2500	575	1150	1725	2300	2875	3450	4025	8050	11500
654	1100	654	1308	1962	2616	3270	3924	4578	9156	13080
657	3500	657	1314	1971	2628	3285	3942	4599	9198	13140
724	3500	724	1448	2172	2896	3620	4344	5068	10136	14480
783	5000	783	1566	2349	3132	3915	4698	5481	10962	15660
1002	6000	1002	2004	3006	4008	5010	6012	7014	14028	20040

Nota. Las 65 instancias simétricas de TSPLIB se seleccionaron en el Anexo 18

A partir de la Tabla 28, se construyó la Figura 75, la cual muestra el comportamiento de la cota superior en función de algunos valores de “k”.

Figura 75

Constante “k” para cota del número de ejecuciones 2Opt en el Multi2Opt



Nota. Las 65 instancias simétricas de TSPLIB se seleccionaron en el Anexo 18

A partir de la Figura 75, se tomó la decisión de asignar k = 6 para el valor de la Cota Superior, donde la única instancia para la cual no cumple dicha cota es para brg180 (único caso puntual del total de revisados).

Para k ≥ 14, se observa que la cota superior es mayor o igual que la cantidad mínima necesaria para la convergencia del algoritmo en el 100% de las instancias revisadas, pero el consumo de recursos se hace elevado. Por lo cual, se concluye usar un valor de k = 6 y la respectiva cota superior toma la siguiente forma:

Cota Superior → Número de ejecuciones = Orden del grafo × 6

De esta manera, se espera contribuir para futuras investigaciones para problemas simétricos del agente viajero tomados de TSPLIB.

Anexo 20. Resumen de las soluciones de 65 instancias del Problema del Agente Viajero Simétrico de TSPLIB

Con el objetivo de presentar una tabla que permita consolidar una mayor cantidad de instancias del Problema del Agente Viajero Simétrico de TSPLIB. Entonces, a partir del Anexo 18, en las 65 instancias seleccionadas se procedió a ejecutar las distintas versiones de los algoritmos desarrollados, tanto en sus versiones Heavy como Soft, para determinar:

- Las soluciones respectivas y así medir su Error Relativo con respecto a la solución óptima reportada en TSPLIB.

- El Nivel de Eficacia respectivo, al realizar la comparación entre los algoritmos a comparar.

Los resultados fueron consolidados en la Tabla 29 y Tabla 30, las cuales constan de las soluciones brindadas por los siguientes algoritmos:

- Vecino más Cercano (VMC).
- Vecino más Cercano con Sacrificio Cortoplacista Adaptativo (VMC_SCA).
- Vecino más Cercano, mejorado con intercambio 2Opt (VMC_2Opt).
- Sacrificio Cortoplacista Adaptativo de Amplitud 1, mejorado por el Criterio Uni2Opt sin el factor “r” (SCA(1)_Uni2Opt).
- Sacrificio Cortoplacista Adaptativo de Amplitud 1, mejorado por el Criterio Uni2Opt con el factor “r” (SCA(1)_Uni2Opt_r).
- Sacrificio Cortoplacista Adaptativo de Amplitud 5, mejorado por el Criterio Uni2Opt sin el factor “r” (SCA(5)_Uni2Opt).
- Sacrificio Cortoplacista Adaptativo de Amplitud 1, mejorado por el Criterio Multi2Opt sin el factor “r” (SCA(1)_Multi2Opt).
- Sacrificio Cortoplacista Adaptativo de Amplitud 1, mejorado por el Criterio Multi2Opt con el factor “r” (SCA(1)_Multi2Opt_r).
- Sacrificio Cortoplacista Adaptativo de Amplitud 5, mejorado por el Criterio Multi2Opt sin el factor “r” (SCA(5)_Multi2Opt).

Tabla 29

Escala de color de la mejor solución disponible: VMC,VMC_SCA, VMC_2Opt, versiones Soft y Heavy de 65 instancias TSPLIB

Instancia TSPLIB	Óptimo	VMC	VMC _SCA	VMC _2Opt	VERSIONES SOFT			VERSIONES HEAVY		
					SCA(1) _Uni 2Opt	SCA(1) _Uni 2Opt_r	SCA(5) _Uni 2Opt	SCA(1) _Multi 2Opt	SCA(1) _Multi 2Opt_r	SCA(5) _Multi 2Opt
1.a280	2579	3157	3115	2733	2744	2753	2742	2664	2665	2649
2.att48	10628	12861	12240	11010	10648	10648	10648	10648	10648	10648
3.att532	27686	35516	33541	29236	29081	28892	29081	28416	28393	28416
4.berlin52	7542	8980	8172	7877	7793	7792	7793	7715	7715	7657
5.bier127	118282	135737	129184	120460	120724	120724	120724	118928	118928	118928
6.Brazil58	25395	30774	27384	26008	25704	25699	25704	25400	25400	25400
7.brg180	1950	12360	59590	2170	2150	2070	2110	1970	1960	1970
8.ch130	6110	7579	6982	6375	6381	6331	6381	6269	6280	6254
9.ch150	6528	8191	7218	6695	6568	6556	6568	6563	6554	6554
10.d198	15780	18240	18012	16021	16260	16009	15902	15926	15907	15902
11.d493	35002	41665	40312	36245	36793	36713	36699	35913	35771	35759
12.d657	48912	61627	58529	50896	51401	51258	51174	50238	50238	50238
13.dantzig42	699	956	873	768	704	700	704	699	700	699
14.eil51	426	511	489	438	437	430	430	430	429	430
15.eil76	538	642	604	552	567	553	567	548	543	546
16.eil101	629	803	736	657	645	639	647	638	634	632
17.fl417	11861	15013	13859	12339	12375	12369	12375	12092	12071	12092
18.fri26	937	1112	1043	955	973	957	937	953	953	937
19.gil262	2378	3208	2791	2477	2523	2502	2518	2440	2440	2433
20.gr17	2085	2187	2158	2085	2085	2085	2085	2085	2085	2085
21.gr21	2707	3333	3098	2795	2707	2707	2707	2707	2707	2707
22.gr24	1272	1553	1445	1318	1314	1300	1314	1294	1272	1272
23.gr48	5046	6098	5889	5259	5101	5101	5101	5097	5094	5075
24.gr120	6942	9351	8555	7253	7377	7217	7241	7069	7081	7051
25.hk48	11461	13181	12606	11461	12163	11895	11735	11461	11461	11461
26.kroa100	21282	27807	24675	21946	23314	22786	23278	21474	21474	21470
27.kroA150	26524	33633	31969	28053	28139	28107	28075	27493	27446	27243
28.kroA200	29368	35859	33962	29654	29955	29746	29868	29591	29534	29565
29.kroB100	22141	29158	24456	22571	22454	22291	22349	22349	22163	22211
30.kroB150	26130	34499	32053	27679	26651	26591	26651	26651	26439	26532
31.kroB200	29437	36980	35418	31799	32008	31457	31067	30651	30127	30651
32.kroC100	20749	26227	24445	21997	21162	21162	21162	21162	21162	20965
33.kroD100	21294	26947	24254	22535	22420	21945	22330	21861	21640	21853
34.kroE100	22068	27460	25672	23305	22872	22872	22620	22355	22226	22355
35.Lin105	14379	20356	17801	14826	14810	14810	14712	14484	14438	14484

Nota. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios respecto a la cercanía al óptimo.

Tabla 29

Escala de color de la mejor solución disponible: VMC,VMC_SCA, VMC_2Opt, versiones Soft y Heavy de 65 instancias TSPLIB

Instancia TSPLIB	Óptimo	VMC	VMC _SCA	VMC _2Opt	VERSIONES SOFT			VERSIONES HEAVY		
					SCA(1) _Uni 2Opt	SCA(1) _Uni 2Opt_r	SCA(5) _Uni 2Opt	SCA(1) _Multi 2Opt	SCA(1) _Multi 2Opt_r	SCA(5) _Multi 2Opt
36.lin318	42029	54019	50869	44055	44049	44049	44049	43450	43204	43046
37.p654	34643	43457	39221	35895	35254	35235	35254	35254	35266	35254
38.pa561	2763	3422	3329	2959	2907	2876	2892	2854	2859	2848
39.Pcb442	50778	61979	58262	53162	53223	52871	53167	51977	51730	51811
40.pr76	108159	153462	137348	114099	111086	109067	111069	109254	108586	109049
41.pr107	44303	46680	46211	44648	44469	44446	44326	44303	44303	44303
42.pr124	59030	69297	67041	59990	59416	59246	59416	59246	59246	59246
43.pr136	96772	120769	118062	106162	106816	104439	104165	100384	100358	100384
44.pr144	58537	61652	61331	59702	59814	59814	59702	59143	59143	58669
45.pr152	73682	85699	81405	76053	74967	74967	74967	74606	74453	74606
46.pr226	80369	94683	91356	82361	82602	82440	81285	81285	81285	81243
47.pr264	49135	58023	56937	52138	52003	51950	52003	50985	50165	50985
48,pr299	48191	59890	58108	50400	50908	50908	50400	49217	48987	49006
49.Pr439	107217	131281	128691	113207	112213	111894	112213	108569	108568	108569
50.Pr1002	259045	331103	311506	275309	272478	270760	270824	269547	269547	268116
51.rat99	1211	1554	1490	1305	1301	1266	1265	1258	1229	1238
52.rat195	2323	2752	2571	2382	2387	2363	2387	2354	2347	2354
53.rat575	6773	8605	8349	7242	7127	7121	7127	7041	6980	7027
54.Rat783	8806	11054	10487	9283	9210	9179	9184	9119	9105	9111
55.rd100	7910	9938	9485	8439	8452	8431	8452	8018	7965	8018
56.rd400	15281	19183	18549	15788	15872	15851	15872	15631	15641	15631
57.si175	21407	22263	22023	21705	21549	21487	21522	21441	21440	21440
58.si535	48450	50144	49929	48910	48852	48852	48835	48756	48747	48756
59.st70	675	830	790	738	732	728	725	688	686	686
60.Swiss42	1273	1630	1491	1284	1273	1273	1273	1273	1273	1273
61.ts225	126643	152493	144849	127831	130496	127794	130496	127512	126899	127161
62.tsp225	3919	5033	4530	4069	4040	4040	4031	3976	3955	3976
63.u159	42080	54675	53323	42626	45526	44705	42798	42547	42375	42511
64.u574	36905	50459	47076	38973	38569	38550	38569	38318	38188	38318
65.u724	41910	52943	50767	44835	43954	43924	43954	43891	43636	43618

Nota. El degradado de color de las celdas indica mayor o menor cercanía al óptimo reportado, tal que el color rojo indica mayor proximidad al óptimo, mientras el verde indica mayor lejanía y el color ámbar indica valores intermedios respecto a la cercanía al óptimo

Tabla 30

Escala de color del error relativo: VMC,VMC_SCA,VMC_2Opt, versiones Soft y Heavy de 65 instancias simétricas TSPLIB

Instancia TSPLIB	VMC	VMC _SCA	VMC _2Opt	VERSIONES SOFT			VERSIONES HEAVY		
				SCA(1) _Uni 2Opt	SCA(1) _Uni 2Opt_r	SCA(5) _Uni 2Opt	SCA(1) _Multi 2Opt	SCA(1) _Multi 2Opt_r	SCA(5) _Multi 2Opt
1.a280	22.4%	20.8%	6.0%	6.4%	6.7%	6.3%	3.3%	3.3%	2.7%
2.att48	21.0%	15.2%	3.6%	0.2%	0.2%	0.2%	0.2%	0.2%	0.2%
3.att532	28.3%	21.1%	5.6%	5.0%	4.4%	5.0%	2.6%	2.6%	2.6%
4.berlin52	19.1%	8.4%	4.4%	3.3%	3.3%	3.3%	2.3%	2.3%	1.5%
5.bier127	14.8%	9.2%	1.8%	2.1%	2.1%	2.1%	0.5%	0.5%	0.5%
6.Brazil58	21.2%	7.8%	2.4%	1.2%	1.2%	1.2%	0.0%	0.0%	0.0%
7.brg180	534%	2955.9%	11.3%	10.3%	6.2%	8.2%	1.0%	0.5%	1.0%
8.ch130	24.0%	14.3%	4.3%	4.4%	3.6%	4.4%	2.6%	2.8%	2.4%
9.ch150	25.5%	10.6%	2.6%	0.6%	0.4%	0.6%	0.5%	0.4%	0.4%
10.d198	15.6%	14.1%	1.5%	3.0%	1.5%	0.8%	0.9%	0.8%	0.77%
11.d493	19.0%	15.2%	3.6%	5.1%	4.9%	4.8%	2.6%	2.2%	2.2%
12.d657	26.0%	19.7%	4.1%	5.1%	4.8%	4.6%	2.7%	2.7%	2.7%
13.dantzig42	36.8%	24.9%	9.9%	0.7%	0.1%	0.7%	0.0%	0.1%	0.0%
14.eil51	20.0%	14.8%	2.8%	2.6%	0.9%	0.9%	0.9%	0.7%	0.9%
15.eil76	19.3%	12.3%	2.6%	5.4%	2.8%	5.4%	1.9%	0.9%	1.5%
16.eil101	27.7%	17.0%	4.5%	2.5%	1.6%	2.9%	1.4%	0.8%	0.5%
17.fl417	26.6%	16.8%	4.0%	4.3%	4.3%	4.3%	1.9%	1.8%	1.9%
18.fri26	18.7%	11.3%	1.9%	3.8%	2.1%	0.0%	1.7%	1.7%	0.0%
19.gil262	34.9%	17.4%	4.2%	6.1%	5.2%	5.9%	2.6%	2.6%	2.3%
20.gr17	4.9%	3.5%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
21.gr21	23.1%	14.4%	3.3%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
22.gr24	22.1%	13.6%	3.6%	3.3%	2.2%	3.3%	1.7%	0.0%	0.0%
23.gr48	20.8%	16.7%	4.2%	1.1%	1.1%	1.1%	1.0%	1.0%	0.6%
24.gr120	34.7%	23.2%	4.5%	6.3%	4.0%	4.3%	1.8%	2.0%	1.6%
25.hk48	15.0%	10.0%	0.0%	6.1%	3.8%	2.4%	0.0%	0.0%	0.0%
26.kroa100	30.7%	15.9%	3.1%	9.5%	7.1%	9.4%	0.9%	0.9%	0.9%
27.kroA150	26.8%	20.5%	5.8%	6.1%	6.0%	5.8%	3.7%	3.5%	2.7%
28.kroA200	22.1%	15.6%	1.0%	2.0%	1.3%	1.7%	0.8%	0.6%	0.7%
29.kroB100	31.7%	10.5%	1.9%	1.4%	0.7%	0.9%	0.9%	0.1%	0.3%
30.kroB150	32.0%	22.7%	5.9%	2.0%	1.8%	2.0%	2.0%	1.2%	1.5%
31.kroB200	25.6%	20.3%	8.0%	8.7%	6.9%	5.5%	4.1%	2.3%	4.1%
32.kroC100	26.4%	17.8%	6.0%	2.0%	2.0%	2.0%	2.0%	2.0%	1.0%
33.kroD100	26.5%	13.9%	5.8%	5.3%	3.1%	4.9%	2.7%	1.6%	2.6%
34.kroE100	24.4%	16.3%	5.6%	3.6%	3.6%	2.5%	1.3%	0.7%	1.3%
35.lin105	41.6%	23.8%	3.1%	3.0%	3.0%	2.3%	0.7%	0.4%	0.7%

Tabla 30

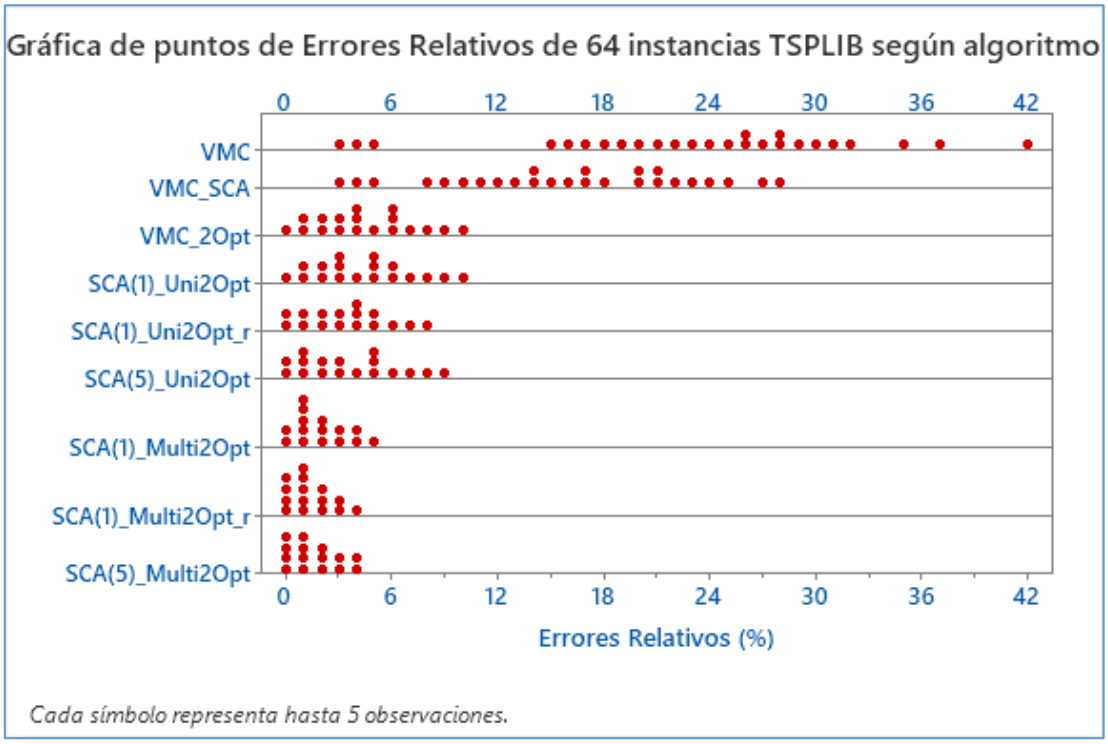
Escala de color del error relativo: VMC,VMC_SCA,VMC_2Opt, versiones Soft y Heavy de 65 instancias simétricas TSPLIB

Instancia TSPLIB	VMC	VMC _SCA	VMC _2Opt	VERSIONES SOFT			VERSIONES HEAVY		
				SCA(1) _Uni 2Opt	SCA(1) _Uni 2Opt_r	SCA(5) _Uni 2Opt	SCA(1) _Multi 2Opt	SCA(1) _Multi 2Opt_r	SCA(5)_ Multi 2Opt
36.lin318	28.5%	21.0%	4.8%	4.8%	4.8%	4.8%	3.4%	2.8%	2.4%
37.p654	25.4%	13.2%	3.6%	1.8%	1.7%	1.8%	1.8%	1.8%	1.8%
38.pa561	23.9%	20.5%	7.1%	5.2%	4.1%	4.7%	3.3%	3.5%	3.1%
39.Pcb442	22.1%	14.7%	4.7%	4.8%	4.1%	4.7%	2.4%	1.9%	2.0%
40.pr76	41.9%	27.0%	5.5%	2.7%	0.8%	2.7%	1.0%	0.4%	0.8%
41.pr107	5.4%	4.3%	0.8%	0.4%	0.3%	0.1%	0.0%	0.0%	0.0%
42.pr124	17.4%	13.6%	1.6%	0.7%	0.4%	0.7%	0.4%	0.4%	0.4%
43.pr136	24.8%	22.0%	9.7%	10.4%	7.9%	7.6%	3.7%	3.7%	3.7%
44.pr144	5.3%	4.8%	2.0%	2.2%	2.2%	2.0%	1.0%	1.0%	0.2%
45.pr152	16.3%	10.5%	3.2%	1.7%	1.7%	1.7%	1.3%	1.0%	1.3%
46.pr226	17.8%	13.7%	2.5%	2.8%	2.6%	1.1%	1.1%	1.1%	1.1%
47.pr264	18.1%	15.9%	6.1%	5.8%	5.7%	5.8%	3.8%	2.1%	3.8%
48.pr299	24.3%	20.6%	4.6%	5.6%	5.6%	4.6%	2.1%	1.7%	1.7%
49.pr439	22.4%	20.0%	5.6%	4.7%	4.4%	4.7%	1.3%	1.3%	1.3%
50.Pr1002	27.8%	20.3%	6.3%	5.2%	4.5%	4.5%	4.1%	4.1%	3.5%
51.rat99	28.3%	23.0%	7.8%	7.4%	4.5%	4.5%	3.9%	1.5%	2.2%
52.rat195	18.5%	10.7%	2.5%	2.8%	1.7%	2.8%	1.3%	1.0%	1.3%
53.rat575	27.0%	23.3%	6.9%	5.2%	5.1%	5.2%	4.0%	3.1%	3.8%
54.Rat783	25.5%	19.1%	5.4%	4.6%	4.2%	4.3%	3.6%	3.4%	3.5%
55.rd100	25.6%	19.9%	6.7%	6.9%	6.6%	6.9%	1.4%	0.7%	1.4%
56.rd400	25.5%	21.4%	3.3%	3.9%	3.7%	3.9%	2.3%	2.4%	2.3%
57.si175	4.0%	2.9%	1.4%	0.7%	0.4%	0.5%	0.2%	0.2%	0.2%
58.si535	3.5%	3.1%	0.9%	0.8%	0.8%	0.8%	0.6%	0.6%	0.6%
59.st70	23.0%	17.0%	9.3%	8.4%	7.9%	7.4%	1.9%	1.6%	1.6%
60.Swiss42	28.0%	17.1%	0.9%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
61.ts225	20.4%	14.4%	0.9%	3.0%	0.9%	3.0%	0.7%	0.2%	0.4%
62.tsp225	28.4%	15.6%	3.8%	3.1%	3.1%	2.9%	1.5%	0.9%	1.5%
63.u159	29.9%	26.7%	1.3%	8.2%	6.2%	1.7%	1.1%	0.7%	1.0%
64.u574	36.7%	27.6%	5.6%	4.5%	4.5%	4.5%	3.8%	3.5%	3.8%
65.u724	26.3%	21.1%	7.0%	4.9%	4.8%	4.9%	4.7%	4.12%	4.08%

Con la información de la Tabla 30, se construyó la Figura 76, la cual muestra la variación del Error Relativo.

Figura 76

Gráfica de puntos del error relativo: VMC, VMC_SCA, VMC_2Opt, versiones Soft y Heavy



Nota. Se tomaron las instancias de la Tabla 30, donde se omitió el Error Relativo del Vecino más Cercano (VMC) y del Vecino más Cercano con Sacrificio Cortoplacista Adaptativo (VMC_SCA) de la instancia brg180, debido a que presentaron un Error Relativo muy elevado, el cual no ayudaría a la visualización de la Figura.

En la Figura 76 se puede observar que los Errores Relativos de los algoritmos en sus versiones Heavy presentan menor dispersión para el total de instancias revisadas; en segundo lugar, los Errores Relativos de las versiones Soft, mientras que la mayor dispersión de Errores Relativos se da para las soluciones del Vecino más Cercano. A partir de la Tabla 30 se construyó la Tabla 31, la cual resume los valores mínimos, máximos, la media, la desviación estándar y el coeficiente de variación del conjunto de soluciones obtenidas para las 65 instancias del problema del agente viajero simétrico de TSPLIB, según el algoritmo utilizado.

Tabla 31

Estadísticos para describir al error relativo según algoritmo Soft, Heavy, Vecino más Cercano (VMC), VMC_SCA, VMC_2Opt

Clasificación del algoritmo	Descripción del algoritmo	Error Relativo				
		Mínimo	Máximo	Media (μ)	Desviación Estándar (σ)	Coeficiente de Variación (CV)(**)
Versiones Precedentes*	VMC	3.5%	533.8%	31.3%	0.64	2.04
	VMC_SCA	2.9%	2955.9%	61.4%	3.65	5.94
	VMC_2OPT	0.0%	11.3%	4.2%	0.03	0.59
SOFT	SCA(1)_Uni2Opt	0.0%	10.4%	3.9%	0.03	0.66
	SCA(1)_Uni2Opt_r	0.0%	7.9%	3.2%	0.02	0.69
	SCA(5)_Uni2Opt	0.0%	9.4%	3.3%	0.02	0.69
HEAVY	SCA(1)_Multi2Opt	0.0%	4.7%	1.7%	0.01	0.72
	SCA(1)_Multi2Opt_r	0.0%	4.1%	1.4%	0.01	0.82
	SCA(5)_Multi2Opt	0.0%	4.1%	1.5%	0.01	0.81

(*) Versiones a partir de las cuales se combinaron los criterios para desarrollar las versiones Soft y Heavy

(**) Para su cálculo, se utilizó todos los decimales de la media y desviación estándar para luego redondear el cociente a dos decimales

Nota: Los estadísticos usados consistieron en el mínimo, máximo, media, desviación estándar y coeficiente de variación del Error Relativo, los cuales son mostrados con dos decimales como máximo, para las 65 instancias TSPLIB seleccionadas en el Anexo 18.

La Tabla 31 muestra la variación del “Error Relativo” de las soluciones obtenidas por los distintos algoritmos utilizados en la presente investigación (en sus versiones Heavy y Soft), así como también las soluciones de las versiones en las que no combinaron el Sacrificio Cortoplacista con el Intercambio 2Opt: Vecino más Cercano [VMC], Vecino más Cercano con heurística 2Opt [VMC_2Opt], Vecino más Cercano con Sacrificio Cortoplacista Adaptativo [VMC_SCA].

Anexo 21. Resumen del tiempo empleado para determinar las soluciones de 65 instancias del Problema del Agente Viajero Simétrico de TSPLIB

En los Anexos 10 a 17 se hace hincapié en la determinación de la complejidad algorítmica, medida con base en su complejidad temporal: cantidad de operaciones elementales necesarias para encontrar una solución hasta el cumplimiento del criterio de parada.

Por otra parte, para los interesados en el uso respectivo de los algoritmos desarrollados en la presente investigación y saber cuál de ellos utilizar según el número de nodos, se presenta la Tabla 32, la cual es resultado de la medición del tiempo (en segundos, minutos y horas) en nuestro computador, que presentó las siguientes características durante el desarrollo de la presente investigación.

Procesador: Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz

RAM Instalada: 8.00 GB (7.88 GB utilizable)

Tipo de Sistema: Sistema operativo de 64 bits, procesador x64

Disco Sólido Kingston de 256 GB

Windows 10 Home Single Language

Notar que estos tiempos son referenciales solo aplicables para el computador en referencia. Para hacer una correcta medición para realizar comparaciones correctas se utiliza la notación Big O, la cual se detalla en:

- Anexos 11 y 13 para las versiones Heavy
- Anexos 15 y 17 para las versiones Soft

Tabla 32

Tiempo de ejecución de algoritmos según instancia TSPLIB

Instancia TSPLIB	VMC	VMC_ SCA	VMC_ 2Opt	SCA(1)_ Uni2Opt	SCA(5)_ Uni2Opt	SCA(1)_ Multi2Opt	SCA(5)_ Multi2Opt
1.gr17	0.6 s	0.7 s	0.6 s	0.9 s	1.4 s	0.7 s	1.4 s
2.gr21	0.8 s	0.8 s	0.7 s	0.9 s	1.7 s	0.7 s	1.5 s
3.gr24	0.6 s	0.6 s	0.6 s	0.8 s	1.5 s	0.7 s	1.5 s
4.fri26	0.7 s	0.7 s	0.7 s	0.8 s	1.5 s	0.8 s	1.9 s
5.dantzig42	3.1 s	2.9 s	2.5 s	2.7 s	29.2 s	2.8 s	28 s
6.Swiss42	0.8 s	0.9 s	0.8 s	1.0 s	2.5 s	1.0 s	3.8 s
7.att48	0.8 s	1.0 s	0.8 s	1.1 s	3.1 s	1.2 s	4.9 s
8.gr48	0.7 s	1.1 s	0.7 s	1.0 s	2.8 s	1.3 s	4.3 s
9.hk48	0.8 s	0.9 s	0.7 s	0.9 s	3.0 s	1.2 s	4.1 s
10.eil51	0.8 s	1.2 s	0.8 s	1.1 s	3.6 s	1.4 s	4.7 s
11.berlin52	2.8 s	2.8 s	2.9 s	2.9 s	31.6 s	3.0 s	39.8 s
12.Brazil58	0.8 s	1.1 s	1.0 s	1.1 s	4.2 s	1.6 s	6.9 s
13.st70	0.9 s	1.6 s	0.9 s	1.3 s	6.6 s	2.3 s	11.2 s
14.eil76	1.0 s	1.3 s	1.0 s	1.5 s	6.9 s	2.3 s	13.8 s
15.pr76	0.9 s	1.4 s	1.0 s	1.5 s	6.8 s	2.2 s	13.6 s
16.rat99	1.0 s	1.6 s	1.0 s	2.0 s	12.8 s	3.9 s	27.3 s
17.kroa100	1.2 s	2.2 s	1.1 s	2.0 s	13.3 s	3.9 s	33.9 s
18.kroB100	1.0 s	2.0 s	1.1 s	2.0 s	13.2 s	4.0 s	33.3 s
19.kroC100	1.0 s	1.9 s	1.2 s	2.0 s	13.0 s	3.9 s	33.2 s
20.kroD100	1.2 s	1.8 s	1.0 s	2.0 s	13.0 s	3.8 s	33.9 s
21.kroE100	1.0 s	1.9 s	1.1 s	2.0 s	13.1 s	4.0 s	32.2 s
22.rd100	1.0 s	1.8 s	1.4 s	2.0 s	13.5 s	4.2 s	33.4 s
23.eil101	2.6 s	3.0 s	3.2 s	3.8 s	42.6 s	5.9 s	1.1 min
24.lin105	2.0 s	3.2 s	2.2 s	3.1 s	24.8 s	4.2 s	51.9 s
25.pr107	1.1 s	2.5 s	1.1 s	2.1 s	15.8 s	5.1 s	43.1 s
26.gr120	1.0 s	2.2 s	1.3 s	2.4 s	21.5 s	6.6 s	1.1 min
27.pr124	1.1 s	2.8 s	1.3 s	2.8 s	23.6 s	7.4 s	1.3 min
28.bier127	1.2 s	3.2 s	1.2 s	2.8 s	24.5 s	7.8 s	1.5 min
29.ch130	1.1 s	2.8 s	1.3 s	2.9 s	27.6 s	5.4 s	1.6 min
30.pr136	1.2 s	3.3 s	1.4 s	3.2 s	31.5 s	10.0 s	1.8 min
31.pr144	1.4 s	3.1 s	1.5 s	3.8 s	35.5 s	7.2 s	2.3 min
32.ch150	1.3 s	3.3 s	1.6 s	3.7 s	38.4 s	8.4 s	2.6 min
33.kroA150	1.4 s	3.6 s	1.7 s	3.7 s	39.7 s	8.7 s	2.6 min

Tabla 32

Tiempo de ejecución de algoritmos según instancia TSPLIB

Instancia TSPLIB	VMC	VMC_ SCA	VMC_ 2Opt	SCA(1)_ Uni2Opt	SCA(5)_ Uni2Opt	SCA(1)_ Multi2Opt	SCA(5)_ Multi2Opt
34.kroB150	1.4 s	4.0 s	1.5 s	4.0 s	40.4 s	8.34 s	2.5 min
35.pr152	1.3 s	3.7 s	1.8 s	4.3 s	41.6 s	8.74 s	2.7 min
36.u159	1.5 s	4.4 s	1.8 s	5.0 s	53.6 s	10.45 s	6.5 min
37.si175	1.1 s	4.6 s	1.4 s	4.7 s	58.2 s	13.83 s	4.8 min
38.brg180	1.5 s	5.8 s	1.8 s	6.0 s	1.1 min	15.02 s	5.3 min
39.rat195	1.7 s	6.8 s	2.3 s	7.2 s	1.5 min	20.62 s	9.5min
40.d198	2 s	8.3 s	2.3 s	7.9 s	1.6 min	41.72 s	8.1 min
41.kroA200	2.2 s	8.2 s	2.2 s	7.9 s	1.7 min	41.16 s	8.2 min
42.kroB200	1.9 s	7.5 s	2.1 s	7.8 s	1.5 min	42.23 s	8.0 min
43.ts225	1.9 s	10.7 s	2.2 s	10.7 s	2.2 min	1 min 1s	12.5 min
44.tsp225	2 s	10.2 s	2.3 s	10.2 s	2.2 min	1 min 1s	12.9 min
45.pr226	2.2 s	10.1 s	2.7 s	10.1 s	2.3 min	1 min 15s	13.3 min
46.gil262	2.6 s	14.4 s	3.5 s	15.1 s	3.7 min	1 min 59s	27.4 min
47.pr264	3 s	15.3 s	3.9 s	15.8 s	3.8 min	2 min 1s	23.5 min
48.a280	3.4 s	17.7 s	4.8 s	19.5 s	4.6 min	2 min 29s	29.6 min
49.pr299	3.7 s	21.3 s	4.6 s	23.2 s	5.6 min	3 min 7s	43.3 min
50.lin318	2.4 s	23.7 s	3.4 s	25.7 s	6.5 min	4 min 16s	47.6 min
51.rd400	3.1 s	46.7 s	5.0 s	49.4 s	12.8 min	10min 56s	2hrs 15min
52.fl417	5.2 s	55.1 s	6.6 s	57.0 s	15.0 min	12min 38s	2hrs 33min
53.pr439	3.5 s	1.0 min	5.6 s	1.0 min	17.5 min	15min 41s	3hrs 3min
54.Pcb442	3.9 s	1.1 min	5.9 s	1.1 min	18.3 min	15min 49s	3hrs 17min
55.d493	5.3 s	1.5 min	8.3 s	1.6 min	26.5 min	19min 34s	5hrs 5min
56.att532	5 s	1.8 min	9.3 s	2.0 min	30.6 min	15min 3s	6hrs 25min
57.si535	2.5 s	1.7 min	7.4 s	1.9 min	30.5 min	17min 50s	6hrs 45min
58.pa561	3.8 s	2.0 min	9.2 s	2.0 min	38.5 min	21min 37s	6hrs 38min
59.u574	4.6 s	2.2 min	10.5 s	2.4 min	40.6 min	24min 51s	> 6hrs
60.rat575	5.4 s	2.2 min	11.2 s	2.4 min	48.5 min	40min 20s	> 6hrs
61.p654	6.1 s	3.7 min	14.8 s	4.3 min	1hr 1min	41min 17s	>15hrs
62.d657	6.5 s	4.0 min	15.5 s	3.8 min	1hr 13min	41min 26s	15hrs 39min
63.u724	7 s	5.0 min	19.1 s	5.4 min	1hr 27min	1hr 33min	>16hrs
64.Rat783	7.9 s	9.5 min	24.3 s	10.8 min	2hrs 2min	1hr 27min	>17hrs
65.Pr1002	5.9 s	12.5min	40.0 s	17.6min	3hrs 4min	4hrs 3min	>24hrs

Nota: El tiempo aproximado de las versiones con factor r, se calcula **multiplicando** el número de valores de **r** probados por el tiempo de ejecución de la versión sin factor.

LEYENDA		
	Tiempo < 1min	Tiempo muy aceptable
	1 <= tiempo <= 31min	Tiempo aceptable
	31 < tiempo <= 61min	Tiempo ligeramente aceptable
	> 61 minutos	Tiempo poco aceptable

A partir de la Tabla 32 y su leyenda, se procedió a la categorización de los algoritmos desarrollados en la presente investigación según el tiempo que emplearon para determinar una solución en nuestro computador clasificándolos por la cantidad de nodos, estos son resumidos en la Tabla 33.

Tabla 33

Categorización de algoritmos según tiempo y orden del grafo

Algoritmo	Categoría		Detalle
VMC		Tiempo muy aceptable	Hasta 1002 nodos
VMC_ SCA		Tiempo muy aceptable	hasta 420 nodos
		Tiempo aceptable	421 hasta 1002 nodos
VMC_ 2Opt		Tiempo muy aceptable	hasta 1002 nodos
SCA(1)_ Uni2Opt		Tiempo muy aceptable	hasta 420 nodos
		Tiempo aceptable	421 hasta 1002 nodos
SCA(5)_ Uni2Opt		Tiempo muy aceptable	hasta 170 nodos
		Tiempo aceptable	De 170 hasta 442 nodos
		Tiempo ligeramente aceptable	De 443 hasta 650 nodos
		Tiempo poco aceptable	De 651 hasta 1002 nodos
SCA(1)_ Multi2Opt		Tiempo muy aceptable	Hasta 200 nodos
		Tiempo aceptable	De 201 hasta 550 nodos
		Tiempo ligeramente aceptable	De 551 hasta 670 nodos
		Tiempo poco aceptable	De 671 hasta 1002 nodos
SCA(5)_ Multi2Opt		Tiempo muy aceptable	hasta 100 nodos
		Tiempo aceptable	De 101 hasta 250 nodos
		Tiempo ligeramente aceptable	De 251 hasta 345 nodos
		Tiempo poco aceptable	De 346 hasta 1002 nodos

Nota: La actual clasificación es referencial, pues solo se cumple para las 65 instancias revisadas y en nuestro computador con las características mencionadas.

A partir de la tabla 33, se resalta en que la clasificación de tiempos es solo referencial para futuras investigaciones que decidan usar las mismas instancias que en la presente investigación. Esto se debe a que el mismo tamaño de instancias pueden consumir más recursos computacionales si los elementos de la matriz de costos ocupan una mayor memoria en el computador. Por ejemplo, si las variables son definidas y leídas como del tipo “entero” o “con decimales”, podría generar un mayor consumo de memoria y, por tanto, mayor tiempo en determinar una solución.

Anexo 22. Selección de Vectores Prohibición para la aplicación del Algoritmo Sacrificio Cortoplacista Adaptativo de Amplitud 5 - SCA(5)

Durante las distintas pruebas de ensayo y error en la determinación de las soluciones para las distintas instancias seleccionadas del problema del agente viajero simétrico de TSPLIB, se observó que ciertos vectores prohibición producían los mismos resultados entre sí. Razón por la cual se realizó un análisis en base a distintas ejecuciones con el algoritmo SCA(5) (sin 2Opt) para algunas instancias, a partir del cual se construyó la Figura 77.

Figura 77

Agrupación del total de vectores prohibición para el SCA(5)

(1 0 0 0 0) (0 1 0 0 0) (0 0 1 0 0) (0 0 0 1 0) (0 0 0 0 1) ——— misma solución
(1 1 0 0 0) (0 1 1 0 0) (0 0 1 1 0) (0 0 0 1 1) ——— misma solución
(1 0 1 0 0) (0 1 0 1 0) (0 0 1 0 1) ——— misma solución
(1 1 1 0 0) (0 1 1 1 0) (0 0 1 1 1) ——— misma solución
(1 1 1 1 0) (0 1 1 1 1) ——— misma solución
(1 0 0 1 0) (0 1 0 0 1) ——— misma solución
(1 0 1 1 0) (0 1 0 1 1) ——— misma solución
(1 1 0 1 0) (0 1 1 0 1) ——— misma solución
(1,0,1,1,1) ——————————— solución independiente de un grupo
(1 1 0 1 1) ——————————— solución independiente de un grupo
(1 1 1 0 1) ——————————— solución independiente de un grupo
(1 0 0 0 1) ——————————— solución independiente de un grupo
(1 1 0 0 1) ——————————— solución independiente de un grupo
(1 0 0 1 1) ——————————— solución independiente de un grupo
(1 0 1 0 1) ——————————— solución independiente de un grupo
(1 1 1 1 1) ——————————— solución independiente de un grupo
(0 0 0 0 0) ——————————— solución del Vecino más Cercano

Recordemos que la cantidad de vectores prohibición está definida según el tamaño de la amplitud, siendo esta igual a $2^{Amplitud}$; razón por la cual, en la Figura 77 se definieron en total 32 vectores prohibición y se observa que ciertos grupos de Vectores Prohibición producen la misma solución entre sí, con lo cual, al aplicar uno de tales vectores, sería equivalente a aplicar el resto perteneciente al mismo grupo.

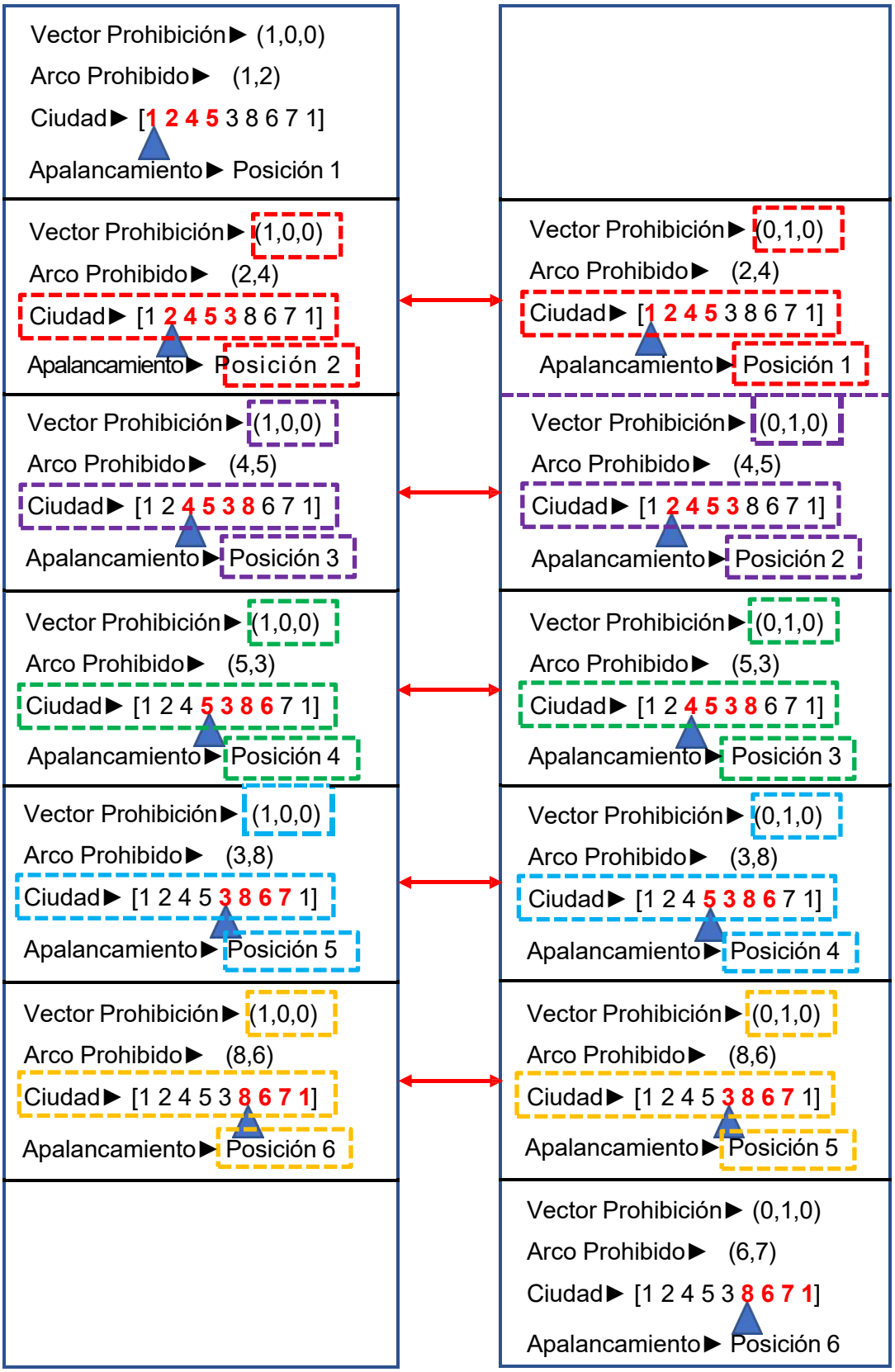
Por medio de este razonamiento, logramos reducir de un total de 32 vectores prohibición (para una amplitud de 5) a solo 16 vectores prohibición, seleccionando uno representativo por grupo equivalente por cada fila de la Figura 77 y se eliminó la solución del vector (0,0,0,0,0) por ser la aplicación del algoritmo del Vecino más Cercano, de tal manera que se usaron solo los siguientes vectores prohibición tanto para el SCA(5)_Uni2Opt como para el SCA(5)_Multi2Opt.

Vector Prohibición 1: (1 0 0 0 0)
Vector Prohibición 2: (1 1 0 0 0)
Vector Prohibición 3: (1 0 1 0 0)
Vector Prohibición 4: (1 1 1 0 0)
Vector Prohibición 5: (1 1 1 1 0)
Vector Prohibición 6: (1 0 0 1 0)
Vector Prohibición 7: (1 0 1 1 0)
Vector Prohibición 8: (1 1 0 1 0)
Vector Prohibición 9: (1,0,1,1,1)
Vector Prohibición 10: (1 1 0 1 1)
Vector Prohibición 11: (1 1 1 0 1)
Vector Prohibición 12: (1 0 0 0 1)
Vector Prohibición 13: (1 1 0 0 1)
Vector Prohibición 14: (1 0 0 1 1)
Vector Prohibición 15: (1 0 1 0 1)
Vector Prohibición 16: (1 1 1 1 1)

Para entender mejor este funcionamiento, se construyó la Figura 78, la cual muestra, de manera intuitiva, la igualdad de dichos vectores prohibición.

Figura 78

Comportamiento del Vector Prohibición para el SCA(3)



En la Figura 78 se observa que los mismos arcos prohibidos generan la misma solución sobre la misma ruta inicial (la cual proviene de la aplicación del Vecino más Cercano); aquí radica la casi equivalencia de aplicar, por ejemplo, el vector prohibición (1,0,0) o el (0,1,0), donde la única diferencia está en los extremos del primer y último punto de apalancamiento. Lo mismo aplica para un vector prohibición de amplitud 5, es decir, de la forma (* * * * *), donde cada "*" representa un valor binario (0 ó 1).

Por lo anterior, vectores prohibición tales como: (1,0,0,0,0), (0,1,0,0,0), (0,0,1,0,0), (0,0,0,1,0), (0,0,0,0,1), presentarían la misma solución en la mayoría de las instancias revisadas; esto debido a que existe la posibilidad de presentar una solución diferente para el primer o último punto de apalancamiento, el cual no necesariamente es el mejor entre los otros puntos de apalancamiento.

Para determinar qué rango de puntos de apalancamiento se debe elegir sin afectar de manera significativa la búsqueda de la mejor solución disponible, se presenta a continuación el Anexo 23.

Anexo 23. Determinación del rango de puntos de apalancamiento para la aplicación del Sacrificio Cortoplacista Adaptativo

De la Figura 78 del Anexo 22, con el razonamiento del pensamiento sistémico en el que se basa el Sacrificio Cortoplacista Adaptativo, los cambios no son vistos en el corto plazo. Debido a esto, bajo este razonamiento, se considera que eliminar el último punto de apalancamiento no afecta de manera significativa la búsqueda de la mejor solución para las instancias del Problema de Agente Viajero Simétrico revisadas.

Para analizar que dicho razonamiento se cumple en al menos la mayor parte de las 65 instancias seleccionadas del Problema del Agente Viajero Simétrico de TSPLIB (para detalle de las 65 instancias, ver Anexo 18), se construyó la Tabla 34, la cual muestra que el punto de apalancamiento es encontrado antes del último nodo de la instancia revisada.

De la Tabla 34, también se observó que la mayor distancia entre el punto de apalancamiento y la cantidad de nodos es seis (para la instancia “11.berlin52” evaluada en el SCA(1)_Multi2Opt), razón por la cual, para definir el rango de los puntos de apalancamiento para la aplicación del Sacrificio Cortoplacista Adaptativo en los algoritmos desarrollados en la presente investigación, se usó:

LInferior = 1 y

LSuperior = Número de nodos del grafo- 6

Tabla 34

Puntos de apalancamiento según algoritmo SCA(1)_Uni2Opt y SCA(1)_Multi2Opt

Instancia TSPLIB	# Nodos	SCA(1)_ Uni2Opt	SCA(1)_ Multi2Opt	Instancia TSPLIB	# Nodos	SCA(1)_ Uni2Opt	SCA(1)_ Multi2Opt	Instancia TSPLIB	# Nodos	SCA(1)_ Uni2Opt	SCA(1)_ Multi2Opt
1.gr17	17	PA= 3	PA= 1	23.eil101	101	PA= 16	PA= 29	45.pr226	226	PA= 29	PA= 76
2.gr21	21	PA= 5	PA= 1	24.lin105	105	PA= 53	PA= 74	46.gil262	262	PA= 89	PA= 167
3.gr24	24	PA= 14	PA= 4	25.pr107	107	PA= 60	PA= 5	47.pr264	264	PA= 109	PA= 105
4.fri26	26	PA= 8	PA= 6	26.gr120	120	PA= 22	PA= 14	48.a280	280	PA= 7	PA= 120
5.dantzig42	42	PA= 21	PA= 36	27.pr124	124	PA= 13	PA= 16	49.pr299	299	PA= 198	PA= 125
6.Swiss42	42	PA= 14	PA= 1	28.bier127	127	PA= 13	PA= 66	50.lin318	318	PA= 144	PA= 105
7.att48	48	PA= 20	PA= 20	29.ch130	130	PA= 24	PA= 11	51.rd400	400	PA= 149	PA= 121
8.gr48	48	PA= 3	PA= 2	30.pr136	136	PA= 5	PA= 53	52.fl417	417	PA= 112	PA= 345
9.hk48	48	PA= 12	PA= 6	31.pr144	144	PA= 47	PA= 105	53.pr439	439	PA= 245	PA= 414
10.eil51	51	PA= 24	PA= 32	32.ch150	150	PA= 37	PA= 119	54.Pcb442	442	PA= 211	PA= 139
11.berlin52	52	PA= 18	PA= 46	33.kroA150	150	PA= 4	PA= 56	55.d493	493	PA= 143	PA= 94
12.Brazil58	58	PA= 11	PA= 13	34.kroB150	150	PA= 6	PA= 6	56.att532	532	PA= 186	PA= 83
13.st70	70	PA= 40	PA= 8	35.pr152	152	PA= 19	PA= 49	57.si535	535	PA= 338	PA= 47
14.eil76	76	PA= 33	PA= 53	36.u159	159	PA= 76	PA= 13	58.pa561	561	PA= 1	PA= 136
15.pr76	76	PA= 14	PA= 40	37.si175	175	PA= 63	PA= 129	59.u574	574	PA= 27	PA= 103
16.rat99	99	PA= 31	PA= 52	38.brg180	180	PA= 121	PA= 163	60.rat575	575	PA= 37	PA= 42
17.kroa100	100	PA= 25	PA= 4	39.rat195	195	PA= 143	PA= 13	61.p654	654	PA= 306	PA= 305
18.kroB100	100	PA= 33	PA= 1	40.d198	198	PA= 16	PA= 81	62.d657	657	PA= 82	PA= 160
19.kroC100	100	PA= 12	PA= 6	41.kroA200	200	PA= 81	PA= 123	63.u724	724	PA= 72	PA= 409
20.kroD100	100	PA= 66	PA= 35	42.kroB200	200	PA= 29	PA= 130	64.Rat783	783	PA= 228	PA= 400
21.kroE100	100	PA= 19	PA= 3	43.ts225	225	PA= 113	PA= 69	65.Pr1002	1002	PA= 589	PA= 250
22.rd100	100	PA= 26	PA= 48	44.tsp225	225	PA= 52	PA= 121				

Anexo 24. Complejidad a posteriori del algoritmo VMC_2Opt

A continuación, se muestra el código en NetBeans, en lenguaje de programación Java, del algoritmo heurístico del Vecino más Cercano, a partir del cual se obtiene una solución para ser punto de partida para mejorarse mediante la aplicación de la heurística 2Opt (la versión con la cual se trabajó en la presente investigación), donde la combinación de la aplicación de ambos algoritmos se denominó VMC_2Opt.

Por esa razón, se realizó un análisis a posteriori de la complejidad para las 65 instancias del Problema del Agente Viajero Simétrico seleccionadas de TSPLIB, indicadas en la Tabla 28 del Anexo 19, de tal manera que, al buscar una función que acote (f(n)) superiormente a la función que describe la cantidad de operaciones elementales del algoritmo (g(n)), según la definición brindada por Guerequeta y Vallecillo (2019, p.7):

$$O(f)= \{g: \mathbf{N}\rightarrow[0,\infty) / \exists c \in \mathbf{R}, c>0, \exists\ n_0 \in N \bullet g(n) \leq cf(n)\ \forall n \geq n_0\}$$

Es decir, se debe buscar una función (f(n)), que, a partir de un determinado parámetro de entrada ($n \geq n_0$), acote la función que describe la cantidad de operaciones elementales (g(n)).

A continuación, se detalla el código en lenguaje Java del algoritmo VMC_2Opt:

```
package vcm_2opt_completov2;
import java.lang.management.ManagementFactory; import jxl.Cell;
import java.lang.management.MemoryMXBean;
import java.lang.management.MemoryUsage; import java.io.IOException;
import jxl.read.biff.BiffException; import java.io.FileInputStream;
import org.apache.poi.ss.usermodel.CellType;
import org.apache.poi.ss.usermodel.Row; import java.util.Iterator;
import org.apache.poi.xssf.usermodel.XSSFSheet; import java.io.File;
import org.apache.poi.xssf.usermodel.XSSFWorkbook; import jxl.Sheet;

public class VCM_2OPT_COMPLETOv2 {

public static void main(String[] args) throws IOException,
BiffException {

     int dataSize= 1024*1024; long contador=0, TInicio, TFin;
     Double tiempo; TInicio = System.currentTimeMillis();
     double r=0; contador++;
     String archivo="C:\\Users\\Jonathan\\Documents\\d657.xlsx";
     FileInputStream fileInStream = new FileInputStream(archivo);
     XSSFWorkbook workBook = new XSSFWorkbook(fileInStream);
     XSSFSheet selSheet = workBook.getSheetAt(0);

int iFilas=selSheet.getPhysicalNumberOfRows();contador++;contador++;
int iteraciones = iFilas*6 ; contador++;contador++;
int indiceFila = -1;contador++; int indiceCol = -1;contador++;

long matrizpesos[][]=new long[iFilas][iFilas];
long matrizclone[][]=new long[iFilas][iFilas];
Iterator rowIterator = selSheet.iterator();

//LECTURA DE ELEMENTOS DE MATRIZ EXCEL
while (rowIterator.hasNext())
{contador++;
      indiceFila += 1; contador++;    indiceCol =-1; contador++;
      Row row = (Row) rowIterator.next();
      Iterator cellIterator = row.cellIterator();
    while (cellIterator.hasNext()) {contador++;
      indiceCol +=1; contador++;
      org.apache.poi.ss.usermodel.Cell cell =
      org.apache.poi.ss.usermodel.Cell) cellIterator.next();
      contador++;
      if(cell.getCellType()==CellType.STRING){
           matrizpesos[indiceFila][indiceCol]= Long.parseLong
           (cell.getStringCellValue()); contador=contador+3;
           matrizclone[indiceFila][indiceCol]= Long.parseLong
           (cell.getStringCellValue());contador=contador+3;       }
      else if(cell.getCellType()==CellType.NUMERIC) {contador++;
           matrizpesos[indiceFila][indiceCol]=
           long)cell.getNumericCellValue(); contador=contador+3;
           matrizclone[indiceFila][indiceCol]=
           (long)cell.getNumericCellValue(); contador=contador+3;
      }
    }contador++;
}contador++;
//TERMINA DE LEER ELEMENTOS DE MATRIZ EXCEL
```

```
//INICIO DEL VMC
int pos[]= new int [iFilas]; int Dsum[]=new int [iteraciones];
pos[0]=1;    contador++;contador++;
contador++;
for (int i = 1; i< iFilas; i++){  contador++;contador++;
    matrizpesos[i][pos[0]]=0; contador=contador+4;
} contador++;

contador++;
for (int i = 1; i < iFilas; i++){contador++;contador++;
    long minimo = 999999999;contador++;
      contador+
      for (int j = 1; j < iFilas; j++){contador++;contador++;
           contador=contador+11;
           if (minimo > matrizpesos[pos[i-1]][j] &
           matrizpesos[pos[i-1]][j]!=0){
              minimo = matrizpesos[pos[i-1]][j]; contador=contador+5;
              pos[i]=j;contador++;contador++;
           }
      }contador++;

      contador++;
      for (int k = 1; k< iFilas; k++){  contador++;contador++;
            matrizpesos[k][pos[i]]=0; contador=contador+4;
      }contador++;
      pos[iFilas-1]=pos[0]; contador=contador+4;
} contador++;

int suma=0;contador++;
contador++
for(int i = 0;i<iFilas-1;i++){contador++;contador++;contador++;
   suma = suma+Math.toIntExact(matrizclone[pos[i]][pos[i+1]]);
   contador=contador+7;
}contador++;contador++;
//FIN DEL VMC

//APERTURO CODIGO PARA 2-OPT
contador++;
for (int ax=0;ax<iteraciones;ax++){  contador++;contador++;
     int cambio=1;contador++; int Compara[]= new int [iteraciones];
     double min_change= 0, change; contador++;
     int b=0, c=0, costoactual, costonuevo ;contador++;contador++;
     int j, min_i=0,min_j=0;contador++;contador++;
    contador++;
    for(int i=0;i<=iFilas-4;i++){contador++;contador++;contador++;
         contador++;contador++;
         for (j=i+2;j<=iFilas-2;j++){  contador=contador+3;
           costonuevo = Math.toIntExact(matrizclone[pos[i]][pos[j]]+
           matrizclone[pos[i+1]][pos[j+1]]); contador=contador+12;
           costoactual=Math.toIntExact(matrizclone[pos[i]][pos[i+1]]
           + matrizclone[pos[j]][pos[j+1]]); contador=contador+12;
           change=(1+r)*costonuevo-costoactual; contador=contador+4;
           contador++;
              if (change <min_change){
                 min_change = change; contador++;
                 b=pos[j];c=pos[i+1]; contador=contador+5;
                 min_i=i+1;min_j=j;    contador=contador+3;
              }
         }contador++;contador++;
    }contador++;contador++;
```

```
     pos[min_i]=b; pos[min_j]=c; contador=contador+4;
     int suma2=0;  pos[0]=1; contador=contador+3;

    contador++;
    for(int i = 0;i<iFilas-1;i++){ contador=contador+3;
        suma2 =suma2+Math.toIntExact(matrizclone[pos[i]][pos[i+1]]);
            contador=contador+7;
    }contador++;contador++;
      Dsum[ax]=suma2;contador++;contador++;
} contador++;
//FIN DE 2OPT

//ORDENO POR BURBUJA Y ENCUENTRO LA MENOR DISTANCIA
int posi=0, min= 999999999;contador++;contador++;
contador++;
for (int i = 0; i < iteraciones; i++){  contador++;contador++;
      contador++;contador++;
      if (min > Dsum[i]) {
          min = Dsum[i]; contador++;contador++;
          posi=i;        contador++;
      }
} contador++;
//FIN DE ORDENO POR BURBUJA Y ENCUENTRO LA MENOR DISTANCIA

//IMPRIMO RUTA Y DISTANCIA ASOCIADA A RUTA DETERMINADA
System.out.println("minimo= "+min);contador++;
contador++;
for (int i = 0; i < iFilas; i++){  contador++;contador++;
       System.out.print(pos[i]+"-");  contador++;
}contador++;
//FIN DE IMPRIMO RUTA Y DISTANCIA ASOCIADA A RUTA DETERMINADA

Runtime runtime= Runtime.getRuntime();  System.out.println("");
System.out.println("MemTotal:"+runtime.totalMemory()/dataSize+"MB");
System.out.println("MemLibre:"+runtime.freeMemory()/dataSize+"MB");
System.out.println("MemUsada:"+(double)((runtime.totalMemory()-
runtime.freeMemory())*1000/dataSize)/1000 + "MB");
System.out.println("La cantidad de tiempo consumido fue de:
"+contador+" operaciones");

TFin = System.currentTimeMillis();  tiempo = (double)(TFin-TInicio);
System.out.println("Segundos es: "+tiempo/1000 +" segundos");

  } //FIN DE PUBLIC CLASS VCM_2OPT_COMPLETOv2
} //FIN DE PUBLIC STATIC VOID MAIN
```

A continuación, se presenta la cantidad de operaciones elementales (para ver definición de operación elemental, ver numeral 2.2.21, inciso “d”), consolidadas en la Tabla 35 (para función de acotación de la forma $f(n) = n^k$) y en la Tabla 36 (para función de acotación de la forma $f(n) = n^k \log(n)$).

Tabla 35

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = n^k

Instancia TSPLIB	Nodos (n)	# Operaciones (g(n))	333333×f(n) / f(n)=n	% Error para f(n)=n	11111×f(n) / f(n)=n^2	% Error para f(n)=n^2	99×f(n) / f(n)=n^3	% Error para f(n)=n^3	6×f(n) / f(n)=n^4	% Error para f(n)=n^4
1.gr17	17	457017	5666661	**1140%**	3211079	**603%**	486387	**6%**	501126	**10%**
2.gr21	21	866373	6999993	**708%**	4899951	**466%**	916839	**6%**	1166886	**35%**
3.gr24	24	1296520	7999992	**517%**	6399936	**394%**	1368576	**6%**	1990656	**54%**
4.fri26	26	1650835	8666658	**425%**	7511036	**355%**	1740024	**5%**	2741856	**66%**
5.dantzig42	42	7014227	13999986	**100%**	19599804	**179%**	7334712	**5%**	18670176	**166%**
6.Swiss42	42	7013310	13999986	**100%**	19599804	**179%**	7334712	**5%**	18670176	**166%**
7.att48	48	10486506	15999984	**53%**	25599744	**144%**	10948608	**4%**	31850496	**204%**
8.gr48	48	10486126	15999984	**53%**	25599744	**144%**	10948608	**4%**	31850496	**204%**
9.hk48	48	10485660	15999984	**53%**	25599744	**144%**	10948608	**4%**	31850496	**204%**
10.eil51	51	12585500	16999983	**35%**	28899711	**130%**	13132449	**4%**	40591206	**223%**
11.berlin52	52	13344251	17333316	**30%**	30044144	**125%**	13920192	**4%**	43869696	**229%**
12.Brazil58	58	18537035	19333314	**4%**	37377404	**102%**	19316088	**4%**	67898976	**266%**
13.st70	70	32641474	23333310	**-29%**	54443900	**67%**	33957000	**4%**	144060000	**341%**
14.eil76	76	41803601	25333308	**-39%**	64177136	**54%**	43458624	**4%**	200173056	**379%**
15.pr76	76	41808978	25333308	**-39%**	64177136	**54%**	43458624	**4%**	200173056	**379%**
16.rat99	99	92577093	32999967	**-64%**	108898911	**18%**	96059601	**4%**	576357606	**523%**
17.kroa100	100	95412245	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**
18.kroB100	100	95416264	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**
19.kroC100	100	95417057	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**
20.kroD100	100	95410725	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**
21.kroE100	100	95414079	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**
22.rd100	100	95413683	33333300	**-65%**	111110000	**16%**	99000000	**4%**	600000000	**529%**

Tabla 35

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = n^k

Instancia TSPLIB	Nodos (n)	# Operaciones (g(n))	333333×f(n) / f(n)=n	% Error para f(n)=n	11111×f(n) / f(n)=n^2	% Error para f(n)=n^2	99×f(n) / f(n)=n^3	% Error para f(n)=n^3	6×f(n) / f(n)=n^4	% Error para f(n)=n^4
23.eil101	101	98307022	33666633	**-66%**	113343311	**15%**	101999799	**4%**	624362406	**535%**
24.lin105	105	110484739	34999965	**-68%**	122498775	**11%**	114604875	**4%**	729303750	**560%**
25.pr107	107	116926558	35666631	**-69%**	127209839	**9%**	121279257	**4%**	786477606	**573%**
26.gr120	120	165040929	39999960	**-76%**	159998400	**-3%**	171072000	**4%**	1244160000	**654%**
27.pr124	124	182131337	41333292	**-77%**	170842736	**-6%**	188755776	**4%**	1418528256	**679%**
28.bier127	127	195695679	42333291	**-78%**	179209319	**-8%**	202789917	**4%**	1560867846	**698%**
29.ch130	130	209909187	43333290	**-79%**	187775900	**-11%**	217503000	**4%**	1713660000	**716%**
30.pr136	136	240392876	45333288	**-81%**	205509056	**-15%**	249030144	**4%**	2052612096	**754%**
31.pr144	144	285421341	47999952	**-83%**	230397696	**-19%**	295612416	**4%**	2579890176	**804%**
32.ch150	150	322675359	49999950	**-85%**	249997500	**-23%**	334125000	**4%**	3037500000	**841%**
33.kroA150	150	322671676	49999950	**-85%**	249997500	**-23%**	334125000	**4%**	3037500000	**841%**
34.kroB150	150	322670424	49999950	**-85%**	249997500	**-23%**	334125000	**4%**	3037500000	**841%**
35.pr152	152	335756488	50666616	**-85%**	256708544	**-24%**	347668992	**4%**	3202768896	**854%**
36.u159	159	384392167	52999947	**-86%**	280897191	**-27%**	397948221	**4%**	3834773766	**898%**
37.si175	175	512678805	58333275	**-89%**	340274375	**-34%**	530578125	**3%**	5627343750	**998%**
38.brg180	180	557955011	59999940	**-89%**	359996400	**-35%**	577368000	**3%**	6298560000	**1029%**
39.rat195	195	709573619	64999935	**-91%**	422495775	**-40%**	734072625	**3%**	8675403750	**1123%**
40.d198	198	742853788	65999934	**-91%**	435595644	**-41%**	768476808	**3%**	9221721696	**1141%**
41.kroA200	200	765616476	66666600	**-91%**	444440000	**-42%**	792000000	**3%**	9600000000	**1154%**
42.kroB200	200	765610662	66666600	**-91%**	444440000	**-42%**	792000000	**3%**	9600000000	**1154%**
43.ts225	225	1090480869	74999925	**-93%**	562494375	**-48%**	1127671875	**3%**	15377343750	**1310%**
44.tsp225	225	1090500136	74999925	**-93%**	562494375	**-48%**	1127671875	**3%**	15377343750	**1310%**

Tabla 35

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = n^k

Instancia TSPLIB	Nodos (n)	# Operaciones (g(n))	333333×f(n) / f(n)=n	% Error para f(n)=n	11111×f(n) / f(n)=n^2	% Error para f(n)=n^2	99×f(n) / f(n)=n^3	% Error para f(n)=n^3	6×f(n) / f(n)=n^4	% Error para f(n)=n^4
45.pr226	226	1105087195	75333258	**-93%**	567505436	**-49%**	1142774424	**3%**	15652546656	**1316%**
46.gil262	262	1722414006	87333246	**-95%**	762703484	**-56%**	1780488072	**3%**	28271992416	**1541%**
47.pr264	264	1762192361	87999912	**-95%**	774392256	**-56%**	1821574656	**3%**	29145194496	**1554%**
48.a280	280	2102699651	93333240	**-96%**	871102400	**-59%**	2173248000	**3%**	36879360000	**1654%**
49.pr299	299	2560839111	99666567	**-96%**	993334511	**-61%**	2646359001	**3%**	47955232806	**1773%**
50.lin318	318	3081040469	105999894	**-97%**	1123588764	**-64%**	3183585768	**3%**	61356380256	**1891%**
51.rd400	400	6134386035	133333200	**-98%**	1777760000	**-71%**	6336000000	**3%**	153600000000	**2404%**
52.fl417	417	6950647365	138999861	**-98%**	1932080679	**-72%**	7178659587	**3%**	181424305926	**2510%**
53.pr439	439	8110449761	146333187	**-98%**	2141323031	**-74%**	8375847381	**3%**	222848303046	**2648%**
54.Pcb442	442	8277993032	147333186	**-98%**	2170689404	**-74%**	8548737912	**3%**	229002554976	**2666%**
55.d493	493	11488393207	164333169	**-99%**	2700517439	**-76%**	11862492543	**3%**	354436898406	**2985%**
56.att532	532	14437658052	177333156	**-99%**	3144679664	**-78%**	14906308032	**3%**	480615507456	**3229%**
57.si535	535	14683194282	178333155	**-99%**	3180245975	**-78%**	15159907125	**3%**	491548503750	**3248%**
58.pa561	561	16930646865	186999813	**-99%**	3496865031	**-79%**	17479289619	**3%**	594295847046	**3410%**
59.u574	574	18135668531	191333142	**-99%**	3660807836	**-80%**	18722803176	**3%**	651326607456	**3491%**
60.rat575	575	18230604228	191666475	**-99%**	3673574375	**-80%**	18820828125	**3%**	655877343750	**3498%**
61.p654	654	26827852412	217999782	**-99%**	4752352476	**-82%**	27692900136	**3%**	1097645859936	**3991%**
62.d657	657	27198958376	218999781	**-99%**	4796052039	**-82%**	28075745907	**3%**	1117925155206	**4010%**
63.u724	724	36400751556	241333092	**-99%**	5824119536	**-84%**	37570838976	**3%**	1648562873856	**4429%**
64.Rat783	783	46047665340	260999739	**-99%**	6812031879	**-85%**	47524820013	**3%**	2255268731526	**4798%**
65.Pr1002	1002	96516661838	333999666	**-100%**	11155488444	**-88%**	99595188792	**3%**	6048144192096	**6166%**

Tabla 36

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación f(n) = n^k × log(n)

Instancia TSPLIB	Nodos (n)	Operaciones (n×6)	(333333)×f(n) / f(n)=nlog(n)	% Error para f(n)=nlog(n)	9999×f(n) / f(n)=n^2log(n)	% Error para f(n)=n^2log(n)	79×f(n) / f(n)=n^3logn(n)	% Error para f(n)=n^3log(n)
1.gr17	17	457017	6972536.92	**1426%**	3951068.69	**765%**	477570.449	**4%**
2.gr21	21	866373	9255525.81	**968%**	6478809.76	**648%**	967360.758	**12%**
3.gr24	24	1296520	11041678.9	**752%**	8833263.61	**581%**	1507323.18	**16%**
4.fri26	26	1650835	12263090.1	**643%**	10627915.8	**544%**	1964696.15	**19%**
5.dantzig42	42	7014227	22725467.3	**224%**	31815367.9	**354%**	9500800.18	**35%**
6.Swiss42	42	7013310	22725467.3	**224%**	31815367.9	**354%**	9500800.18	**35%**
7.att48	48	10486506	26899832.9	**157%**	43039345.3	**310%**	14688614.6	**40%**
8.gr48	48	10486126	26899832.9	**157%**	43039345.3	**310%**	14688614.6	**40%**
9.hk48	48	10485660	26899832.9	**157%**	43039345.3	**310%**	14688614.6	**40%**
10.eil51	51	12585500	29028664	**131%**	49348284.6	**292%**	17894360.4	**42%**
11.berlin52	52	13344251	29744028.2	**123%**	51555851.6	**286%**	19061420.1	**43%**
12.Brazil58	58	18537035	34092907.1	**84%**	65912360.5	**256%**	27181211.1	**47%**
13.st70	70	32641474	43052244.5	**32%**	100454333	**208%**	49996621.6	**53%**
14.eil76	76	41803601	47647230	**14%**	120705230	**189%**	65224930.2	**56%**
15.pr76	76	41808978	47647230	**14%**	120705230	**189%**	65224930.2	**56%**
16.rat99	99	92577093	65855895.6	**-29%**	217322499	**135%**	152972664	**65%**
17.kroa100	100	95412245	66666600	**-30%**	222220000	**133%**	158000000	**66%**
18.kroB100	100	95416264	66666600	**-30%**	222220000	**133%**	158000000	**66%**
19.kroC100	100	95417057	66666600	**-30%**	222220000	**133%**	158000000	**66%**
20.kroD100	100	95410725	66666600	**-30%**	222220000	**133%**	158000000	**66%**
21.kroE100	100	95414079	66666600	**-30%**	222220000	**133%**	158000000	**66%**
22.rd100	100	95413683	66666600	**-30%**	222220000	**133%**	158000000	**66%**

Tabla 36

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación $f(n) = n^k \times log(n)$

Instancia TSPLIB	Nodos (n)	Operaciones (n×6)	(333333)×f(n) / f(n)=nlog(n)	% Error para f(n)=nlog(n)	9999×f(n) / f(n)=n^2log(n)	% Error para f(n)=n^2log(n)	79×f(n) / f(n)=n^3logn(n)	% Error para f(n)=n^3log(n)
23.eil101	101	98307022	67478752.1	**-31%**	204440377	**108%**	163139291	**66%**
24.Lin105	105	110484739	70741554.7	**-36%**	222813837	**102%**	184842562	**67%**
25.pr107	107	116926558	72381282.4	**-38%**	232320914	**99%**	196400509	**68%**
26.gr120	120	165040929	83167166.7	**-50%**	299372159	**81%**	283833190	**72%**
27.pr124	124	182131337	86528009.8	**-52%**	321852330	**77%**	315318074	**73%**
28.bier127	127	195695679	89060935.1	**-54%**	339288570	**73%**	340442266	**74%**
29.ch130	130	209909187	91604120.3	**-56%**	357220701	**70%**	366902350	**75%**
30.pr136	136	240392876	96720333.8	**-60%**	394579895	**64%**	423979037	**76%**
31.pr144	144	285421341	103601296	**-64%**	447513291	**57%**	509142034	**78%**
32.ch150	150	322675359	108804454	**-66%**	489571571	**52%**	580200332	**80%**
33.kroA150	150	322671676	108804454	**-66%**	489571571	**52%**	580200332	**80%**
34.kroB150	150	322670424	108804454	**-66%**	489571571	**52%**	580200332	**80%**
35.pr152	152	335756488	110546631	**-67%**	504042733	**50%**	605315046	**80%**
36.u159	159	384392167	116673931	**-70%**	556479553	**45%**	699063874	**82%**
37.si175	175	512678805	130843755	**-74%**	686861709	**34%**	949681281	**85%**
38.brg180	180	557955011	135316215	**-76%**	730635221	**31%**	1039067191	**86%**
39.rat195	195	709573619	148852101	**-79%**	870698582	**23%**	1341445311	**89%**
40.d198	198	742853788	151579751	**-80%**	900294583	**21%**	1408381624	**90%**
41.kroA200	200	765616476	153401846	**-80%**	920319957	**20%**	1454250957	**90%**
42.kroB200	200	765610662	153401846	**-80%**	920319957	**20%**	1454250957	**90%**
43.ts225	225	1090480869	176413512	**-84%**	1190673321	**9%**	2116633491	**94%**
44.tsp225	225	1090500136	176413512	**-84%**	1190673321	**9%**	2116633491	**94%**

Tabla 36

Cantidad de operaciones elementales según instancia simétrica TSPLIB para función de acotación $f(n) = n^k \times log(n)$

Instancia TSPLIB	Nodos (n)	Operaciones (n×6)	(333333)×f(n) / f(n)=nlog(n)	% Error para f(n)=nlog(n)	9999×f(n) / f(n)=n^2log(n)	% Error para f(n)=n^2log(n)	79×f(n) / f(n)=n^3logn(n)	% Error para f(n)=n^3log(n)
45.pr226	226	1105087195	177342658	**-84%**	1202264188	**9%**	2146737155	**94%**
46.gil262	262	1722414006	211198102	**-88%**	1659852737	**-4%**	3435906785	**99%**
47.pr264	264	1762192361	213100932	**-88%**	1687592297	**-4%**	3519994494	**100%**
48.a280	280	2102699651	228401188	**-89%**	1918380039	**-9%**	4243881035	**102%**
49.pr299	299	2560839111	246741648	**-90%**	2213053472	**-14%**	5227976403	**104%**
50.lin318	318	3081040469	265257009	**-91%**	2530301345	**-18%**	6357258766	**106%**
51.rd400	400	6134386035	346940985	**-94%**	4162879657	**-32%**	13156015316	**114%**
52.fl417	417	6950647365	364198547	**-95%**	4555672772	**-34%**	15009253737	**116%**
53.pr439	439	8110449761	386680255	**-95%**	5092074790	**-37%**	17661590737	**118%**
54.Pcb442	442	8277993032	389758491	**-95%**	5167685942	**-38%**	18046330405	**118%**
55.d493	493	11488393207	442524068	**-96%**	6544283015	**-43%**	25490568116	**122%**
56.att532	532	14437658052	483394513	**-97%**	7714212640	**-47%**	32424535336	**125%**
57.si535	535	14683194282	486555938	**-97%**	7808449690	**-47%**	33005713184	**125%**
58.pa561	561	16930646865	514055541	**-97%**	8650698250	**-49%**	38342863862	**126%**
59.u574	574	18135668531	527871281	**-97%**	9089043551	**-50%**	41219298818	**127%**
60.rat575	575	18230604228	528935808	**-97%**	9123239396	**-50%**	41446459604	**127%**
61.p654	654	26827852412	613795335	**-98%**	12041472254	**-55%**	62219692519	**132%**
62.d657	657	27198958376	617046199	**-98%**	12160776543	**-55%**	63124390929	**132%**
63.u724	724	36400751556	690149550	**-98%**	14988564220	**-59%**	85737165632	**136%**
64.Rat783	783	46047665340	755271065	**-98%**	17739560916	**-61%**	109742576215	**138%**
65.Pr1002	1002	96516661838	1002288817	**-99%**	30125819076	**-69%**	238493808020	**147%**

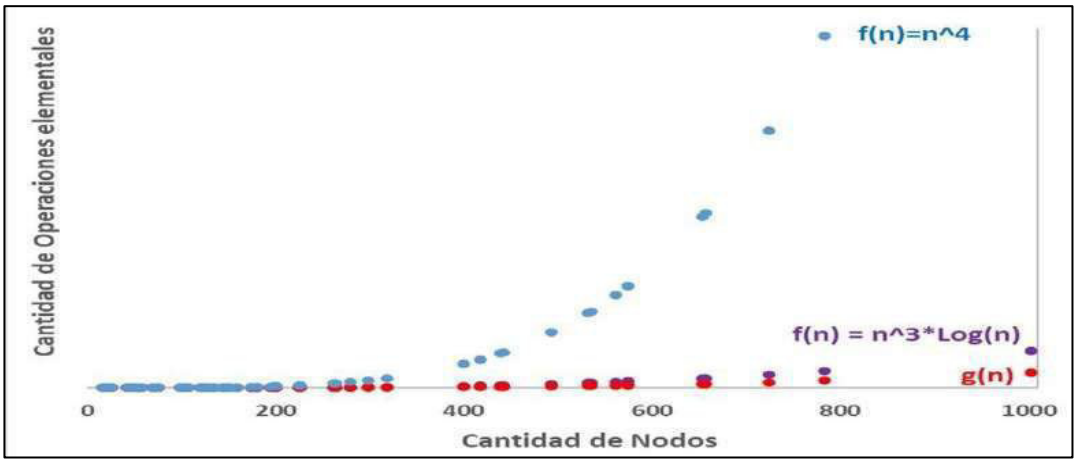
Donde:

- g(n): función que describe la cantidad de operaciones elementales según instancia TSPLIB seleccionada.
- f(n): función que, al ser multiplicada por una constante, acota a g(n) a partir de $n \geq n_0$
- c: constante multiplicativa

En la Tabla 35 se puede observar que la función de acotación corresponde a $f(n) = n^4$ y de la Tabla 36, se verifica que la función de acotación corresponde a $f(n) = n^3 \times \log(n)$. Así, al presentar dos funciones f(n) que acotan a la función g(n), la cual representa la cantidad de operaciones elementales del VMC_2Opt según el tamaño de instancia. Por ello, seleccionamos la "f(n)" que acote de mejor manera a g(n), tal como se muestra en la Figura 79.

Figura 79

Funciones f(n) que acotan superiormente al VMC_2Opt



De la Figura 79 se observa que, de las dos funciones que acotan a g(n): $f(n) = n^4$ y $f(n) = n^3\log(n)$, la función que acota mejor a g(n) corresponde a la función $f(n) = n^3\log(n)$. Por lo anterior, la complejidad de nuestra versión del algoritmo heurístico 2Opt (VMC_2Opt) es de $O(n^3\log(n))$.

Anexo 25. Descripción de Vectores Prohibición para el SCA(5)_Multi2Opt

Para determinar los vectores prohibición, del total de 16 vectores definidos en el Anexo 22, que permitan encontrar las mejores soluciones para el algoritmo SCA(5)_Multi2Opt, se realizaron distintas ejecuciones para las 65 instancias seleccionadas, las cuales se presentan concisamente en la Tabla 37.

Tabla 37

Vectores prohibición de la mejor solución disponible para el SCA(5)_Multi2Opt según instancia simétrica TSPLIB

Instancia	Óptimo	# Operaciones	Vector Prohibición	SCA(5)_Multi2Opt
1.gr17	2085	82475483	Todos los 16 vectores prohibición	2085
2.gr21	2707	211928345	Todos los 16 vectores prohibición	2707
3.gr24	1272	379597201	(1,1,0,1,0)(1,1,0,1,1)	1272
4.fri26	937	536225202	(1,0,1,1,1)(1,1,0,1,1)(1,1,1,0,1)(1,0,0,0,1)(1,1,0,0,1)(1,0,0,1,1)(1,0,1,0,1)(1,1,1,1,1)	937
5.dantzig42	699	4076150254	Todos los 16 vectores prohibición	699
6.Swiss42	1273	4075227193	Todos los 16 vectores prohibición	1273
7.att48	10628	7101196506	Todos los 16 vectores prohibición	10648
8.gr48	5046	6656932148	(1,1,0,0,1)(1,0,0,0,1)	5075
9.hk48	11461	7100452016	Todos los 16 vectores prohibición	11461
10.eil51	426	9126670697	Todos los 16 vectores prohibición - {(1,0,1,0,0)(1,0,1,0,1)}	430
11.berlin52	7542	9890571135	(1,0,0,1,1)(1,1,0,1,1)(1,1,0,1,0)(1,0,0,1,0)	7657
12.Brazil58	25395	15519680579	Todos los 16 vectores prohibición	25400
13.st70	675	33598008072	(1,0,0,1,1)(1,1,0,0,1)(1,0,0,0,1)(1,0,0,1,0)	686
14.eil76	538	47041778118	(1,0,0,1,1)(1,0,1,1,1)(0,1,0,1,1)(1,0,0,1,0)	546
15.pr76	108159	47046825948	(1,0,0,1,1)(1,0,0,1,0)	109049
16.rat99	1211	1.38247E+11	(1,0,1,0,1)(1,0,0,0,1)	1238

Tabla 37

Vectores prohibición de la mejor solución disponible para el SCA(5)_Multi2Opt según instancia simétrica TSPLIB

Instancia	Óptimo	# Operaciones	Vector Prohibición	SCA(5)_Multi2Opt
17.kroa100	21282	1.44008E+11	(1,0,0,1,1)(1,0,0,1,0)	21470
18.kroB100	22141	1.44013E+11	(1,0,1,0,1)(1,1,1,0,1)(1,0,1,0,0)	22211
19.kroC100	20749	1.44014E+11	(1,0,0,1,1)(1,0,0,1,0)	20965
20.kroD100	21294	1.44008E+11	(1,1,1,1,1)(1,0,1,1,1)(1,1,0,1,0)(0,1,1,1,1)(1,1,1,0,0)(1,1,0,0,0)	21853
21.kroE100	22068	1.44012E+11	(1,1,0,0,0)(1,0,0,0,0)	22355
22.rd100	7910	1.4401E+11	Todos los 16 vectores prohibición	8018
23.eil101	629	1.49954E+11	(1,1,1,1,1)(1,0,1,0,1)(1,0,0,1,1)(1,1,0,0,1)(1,0,0,0,1)(1,1,1,0,1)(1,1,0,1,1)(1,0,1,1,1)	632
24.lin105	14379	1.756E+11	Todos los 16 vectores prohibición	14484
25.pr107	44303	1.8958E+11	Todos los 16 vectores prohibición	44303
26.gr120	6942	3.0192E+11	(1,0,0,0,1)	7051
27.pr124	59030	3.44843E+11	Todos los 16 vectores prohibición	59246
28.bier127	118282	3.79917E+11	Todos los 16 vectores prohibición	118928
29.ch130	6110	4.17591E+11	(0,1,0,1,1)(0,1,1,1,1)(1,1,1,0,0)(1,0,1,0,0)	6254
30.pr136	96772	5.0131E+11	(1,1,0,1,0)(0,1,0,1,1)(1,0,0,1,0)(0,1,1,1,1)(1,1,1,0,0)(1,0,1,0,0)(1,1,0,0,0)(1,0,0,0,0)	100384
31.pr144	58537	6.3175E+11	(1,1,1,1,1)(1,0,1,0,1)(1,1,1,0,1)(1,0,1,1,1)(0,1,0,1,1)(0,1,1,1,1)(1,1,1,0,0)(1,0,1,0,0)	58669
32.ch150	6528	7.45175E+11	(1,1,0,1,0)(1,0,0,1,0)	6554

Tabla 37

Vectores prohibición de la mejor solución disponible para el SCA(5)_Multi2Opt según instancia simétrica TSPLIB

Instancia	Óptimo	# Operaciones	Vector Prohibición	SCA(5)_Multi2Opt
33.kroA150	26524	7.45172E+11	(1,0,0,1,1)(1,0,0,0,1)(1,0,0,1,0)	27243
34.kroB150	26130	7.45175E+11	(1,1,0,0,1)(1,0,0,0,1)	26532
35.pr152	73682	7.86139E+11	Todos los 16 vectores prohibición	74606
36.u159	42080	9.43064E+11	(1,1,1,1,1)(1,0,1,1,1)(0,1,0,1,1)(0,1,1,1,1)(1,1,1,0,0)(1,0,1,0,0)	42511
37.si175	21407	1.38906E+12	(1,1,0,0,1)(1,0,0,0,1)	21440
38.brg180	1950	1.55637E+12	(1,1,1,1,1)(1,0,1,0,1) (1,1,1,0,1)(1,0,1,1,1)	1970
39.rat195	2323	2.14959E+12	(1,0,1,0,1)(1,0,1,1,1)(1,0,0,0,0)	2354
40.d198	15780	2.28607E+12	(1,0,0,1,0)(1,1,0,1,0)(1,1,0,1,1)	15902
41.kroA200	29368	2.38063E+12	(1,1,1,0,1)(1,1,1,0,0)(1,1,0,0,0)(1,1,0,0,1)	29565
42.kroB200	29437	2.38061E+12	Todos los 16 vectores prohibición - {(1,0,1,0,0)(0,1,0,1,1)(1,0,1,0,1)}	30651
43.ts225	126643	3.82697E+12	(1,0,1,0,0)(1,1,1,0,0)(0,1,1,1,1)(0,1,0,1,1)(1,0,1,1,1)	127161
44.tsp225	3919	3.82702E+12	(1,0,0,0,0)(1,1,0,0,0)(1,0,0,1,0)(1,1,0,1,0)(1,1,0,1,1)(1,0,0,1,1)	3976
45.pr226	80369	3.89591E+12	(1,0,0,1,0)(1,1,0,1,1)(1,1,1,0,1)(1,0,0,0,1)(1,1,0,0,1)(1,0,0,1,1)(1,1,1,1,1)	81243
46.gil262	2378	7.06438E+12	(1,0,0,0,1)(1,1,1,0,1)(1,1,0,0,1)(1,0,1,0,1)	2433
47.pr264	49135	7.28392E+12	todos - {(1,0,0,1,0)(1,1,0,1,0)(1,0,1,1,1)(1,1,0,1,1)(1,0,0,1,1)(1,1,1,1,1)}	50985
48.a280	2579	9.22969E+12	(1,1,1,1,1)(1,1,0,1,1)(1,1,0,1,0)(0,1,1,1,1)	2649

Tabla 37

Vectores prohibición de la mejor solución disponible para el SCA(5)_Multi2Opt según instancia simétrica TSPLIB

Instancia	Óptimo	# Operaciones	Vector Prohibición	SCA(5)_Multi2Opt
49.pr299	48191	1.20192E+13	(1,1,1,0,0)(0,1,1,1,1)(1,1,1,0,1)	49006
50.lin318	42029	1.53974E+13	Todos - {(1,0,0,0,0)(1,0,1,0,0)(1,0,0,1,0)(0,1,0,1,1)(1,0,0,0,1)(1,0,1,0,1)}	43046
51.rd400	15281	3.87047E+13	(1,0,0,0,1)(1,1,0,0,1)	15631
52.fl417	11861	4.57454E+13	(1,0,0,0,0)(1,1,0,0,0)(1,0,0,1,0)(1,1,0,1,0)(1,1,0,1,1)	12092
53.pr439	107217	5.62336E+13	Todos los 16 vectores prohibición - {(1,1,1,0,0)(1,0,1,0,0)}	108569
54.Pcb442	50778	5.77924E+13	(1,1,0,0,0)(1,1,1,0,0)(0,1,1,1,1)(1,1,0,1,0)(1,1,0,1,1)(1,1,1,0,1)(1,1,0,0,1)	51811
55.d493	35002	8.95803E+13	(1,0,0,1,0)(1,1,0,1,0)(1,1,0,1,1)(1,0,0,1,1)	35759
56.att532	27686	1.21586E+14	(1,0,0,0,0)(1,1,0,0,0)(1,1,1,0,1)(1,0,0,0,1)(1,1,0,0,1)(1,0,1,0,1)	28416
57.si535	48450	1.24359E+14	Todos los 16 vectores prohibición	48756
58.pa561	2763	1.50437E+14	(1,1,0,0,0)(1,1,1,0,0)(0,1,1,1,1)(1,1,0,1,0)(1,1,0,1,1)(1,1,1,0,1)(1,1,0,0,1)	2848
59.u574	36905	1.64916E+14	(1,0,0,0,0)(1,1,0,0,0)(0,1,1,1,1)(0,1,0,1,1)(1,0,1,1,1)(1,1,1,1,1)	38318
60.rat575	6773	1.66071E+14	(1,0,1,1,1)(1,1,0,1,1)(1,1,1,0,1)(1,0,0,0,1)(1,1,0,0,1)(1,0,0,1,1)(1,0,1,0,1)(1,1,1,1,1)	7027
61.p654	34643	2.78298E+14	Todos los 16 vectores prohibición	35254
62.d657	48912	2.83454E+14	Todos los 16 vectores prohibición	50238
63.u724	41910	4.18371E+14	Todos los 16 vectores prohibición - {(1,0,0,0,0), (1,0,1,0,0), (1,0,0,0,1), (1,0,1,0,1)}	43618
64.Rat783	8806	5.72717E+14	(1,0,0,1,1) (1,1,0,1,1)	9111
65.Pr1002	259045	1.53862E+15	(1,0,0,1,0)(1,1,0,1,0)(1,1,0,1,1)(1,0,0,1,1)	268116

Anexo 26. Determinación de la “Amplitud (A)” del Sacrificio Cortoplacista Adaptativo

Se experimentó con valores de amplitud entre 1 y 5 para la versión SCA(A)_Multi2Opt, registrando el mejor costo alcanzado por cada valor y eligiendo la amplitud que obtuvo los menores costos para cada ruta. Los resultados se presentan en la Tabla 38 y Figura 80.

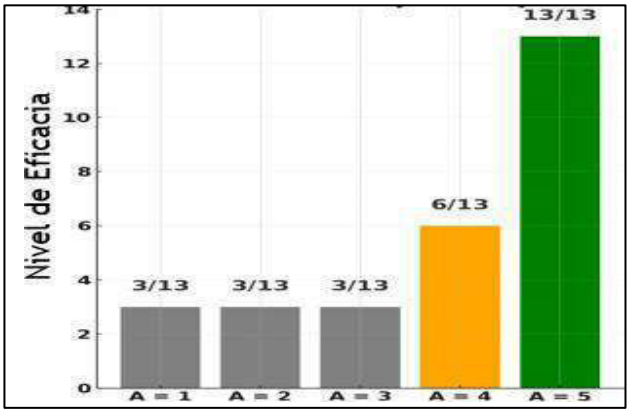
Tabla 38

Determinación de la amplitud del sacrificio cortoplacista adaptativo a utilizar

N°	Instancia TSPLIB	Óptimo	Mejor costo alcanzado por valor de “A”				
			A = 1	A = 2	A = 3	A = 4	A = 5
1	Swiss42	1273	1273	1273	1273	1273	1273
2	eil50	425	429	429	429	429	429
3	eil51	426	430	430	430	430	430
4	st70	675	688	688	687	686	686
5	Eil75	535	547	547	547	546	542
6	Eil76	538	548	548	548	546	546
7	Rat99	1211	1258	1249	1249	1249	1238
8	KroA100	21282	21474	21474	21474	21470	21287
9	KroA150	26524	27493	27493	27493	27243	27243
10	Krob150	26130	26651	26651	26651	26651	26532
11	d198	15780	15926	15926	15926	15902	15902
12	tsp225	3919	3976	3976	3976	3976	3976
13	Pcb442	50778	51977	51811	51811	51811	51811

Figura 80

Nivel de eficacia por cada valor de amplitud(A) evaluado



Anexo 27. Diagrama de flujo del algoritmo VMC_2Opt_r

A partir del desarrollo de las versiones Soft y Heavy de nuestros algoritmos, con el criterio Uni2Opt o Multi2Opt, con influencia o no del factor "r", surgió un algoritmo el cual no se encuentra basado en el sacrificio o MultiSacrificio Cortoplacista (no usa el SCA(1) ni el SCA(A)), pero sí usa el enfoque del factor de perturbación que modifica el umbral para realizar el Intercambio 2Opt (ver diagrama de flujo en la Figura 81).

Si bien este algoritmo no formó parte del estudio original de nuestra investigación, se consideró su relevancia para ser mencionado y analizado en futuros trabajos de investigación, al cual hemos denominado algoritmo heurístico del VMC_2Opt_r, cuya nomenclatura se explica a continuación:

- VMC: Por usar el algoritmo del Vecino más Cercano para construir una primera ruta, la cual se mejorará en una siguiente etapa.

- 2Opt: Por usar el algoritmo heurístico del Intercambio 2Opt para mejorar la ruta anterior, donde se puede usar la política BI o FI.

- "r": Por la adición de la componente lineal "(1 + r)", la cual está en función del denominado factor de perturbación "r" dentro del criterio del algoritmo Intercambio 2Opt.

El algoritmo VMC_2Opt_r funciona como una metaheurística poblacional, generando una solución por cada valor de "r" evaluado, los cuales son calculados por medio de la siguiente fórmula recursiva:

$r_n \leftarrow r_{n-1} + (LSuperior - LInferior) / NIntervalos \quad / n = 0, 1, 2, 3, \ldots, NIntervalos.$

Donde:

- r_0 = LInferior;
- La cantidad de valores de "r" es igual a "NIntervalos+1".
- LSuperior es el mayor valor permisible para "r" y definido por el usuario.
- LInferior es el menor valor permisible para "r" y definido por el usuario.

Figura 81

Diagrama de flujo para el algoritmo VMC_2Opt_r

