



Universidad Autónoma de Querétaro

Facultad de Informática

Doctorado en Ciencias de la Computación

ALGORITMOS DE RUTEO EN SUPERFICIES DE
EMBALDOSADOS RECTILÍNEOS

Tesis

Que como parte de los requisitos para obtener el Grado de
Doctor en Ciencias de la Computación

Presenta:

Fidel González Gutiérrez, M. C.

Dirigido por:

Dr. Arturo González Gutiérrez

SINODALES

Dr. Arturo González Gutiérrez
Presidente

Dr. Saúl Tovar Arriaga
Secretario

Dra. Adela Becerra Chávez
Vocal

Dra. Amanda Montejano Cantoral
Suplente

Dr. Francisco Antonio Castillo Velásquez
Suplente

Centro Universitario Querétaro, QRO
Agosto 2022
México.

*Por tanto, al Rey de los siglos, inmortal, invisible, al
único y sabio Dios, sea honor y gloria por los siglos de
los siglos. Amén.*

1o. Timoteo 1:17 RVR1960

*Primeramente quiero expresar mi agradecimiento a mi Dios por este logro
alcanzado.*

*Así mismo quiero agradecer de manera especial a mi madre Lupita por su
apoyo y porque siempre tiene un consejo sabio para mí.*

A mis hermanos y grandes amigos Arturo y Abraham gracias por todo su apoyo.

Reconocimientos

Quiero expresar mi gratitud a mi director de tesis, **Dr. Arturo González Gutiérrez**, por su guía y apoyo total para el desarrollo de este trabajo de investigación. Su experiencia y conocimiento, así como su tiempo y comentarios valiosos me permitieron alcanzar los resultados que aquí presento.

A los profesores del programa doctoral y particularmente a los miembros del sínodo doctoral, agradezco sus enseñanzas que redundaron en mi formación académica y finalmente en el enriquecimiento de este trabajo.

Mis agradecimientos al **Dr. Saúl Tovar Arriaga** por su tiempo dedicado en la revisión de este trabajo, a la **Dra. Adela Becerra Chávez** por sus atinadas observaciones en los resultados gráficos que se presentaron, a la **Dra. Amanda Montejano Cantoral** por sus recomendaciones pertinentes y observaciones en la formalización matemática y al **Dr. Francisco Antonio Castillo Velásquez** por su tiempo en la revisión de este trabajo.

Un agradecimiento especial a mi maestro, compañero y amigo **Dr. Jaime Rangel Mondragón†** de quien aprendí a disfrutar las matemáticas y computación, como un mundo fascinante y maravilloso de esta creación. Su ejemplo y tenacidad por este maravilloso campo del conocimiento siempre estarán presentes.

Agradezco a la Universidad Autónoma de Querétaro por el apoyo recibido para llevar a cabo mis estudios doctorales y el desarrollo de esta investigación.

Resumen

Se estudia el problema de ruteo en superficies de embaldosados rectilíneos. Formalmente el problema consiste en, dado un rectángulo R dividido en rectángulos de tamaño 1×2 (dominós), encontrar un conjunto de aristas sobre la periferia de R y dominós, interconectadas y libres de ciclos (árbol o corredor) que conecten todos los dominós desde un punto de acceso sobre la periferia de R , y cuya longitud total sea la mínima. Así planteado, es un subproblema del problema del Corredor de Longitud Mínima (MLC), el cual es NP-completo, y en consecuencia no existe algoritmo eficiente que produzca soluciones óptimas. Por lo que el objetivo es diseñar algoritmos de ruteo en superficies de embaldosados rectilíneos utilizando una metodología basada en la técnica de diseño greedy que produzcan un corredor de longitud total mínima. Para ello, se diseñan familias de instancias de embaldosados para el análisis experimental de los algoritmos de ruteo. Se presentan varias técnicas para la enumeración de embaldosados en general, a la vez que se utiliza una metodología de acuerdo a la técnica de backtracking para la generación de familias de embaldosados con dominós no isomorfos y libres de corte de tamaño 6×5 , 6×7 , 6×8 , 6×9 , 6×10 y 7×8 . Sobre dichas familias, se lleva a cabo el análisis experimental de cuatro heurísticas, midiendo su rendimiento y la solución, bajo los criterios de conectar primero: dominós más cercanos, dominós más alejados, vértices compartidos (mejorado), vértices internos (usando reducción). Las heurísticas se implementan en ©Mathematica versión 11.3.0.0 en una computadora MacBook Pro con un CPU de 2.6 GHz Intel Core i7 con 6 núcleos y 16 GB de memoria RAM. Como resultado del análisis experimental se establece que existen cotas inferior y superior para la longitud total del corredor, entre las cuales se encuentra la solución óptima. Asimismo, se establece que la

heurística basada en el criterio de conectar todos los vértices internos, de acuerdo a los algoritmos de Kruskal y Prim, y enseguida llevar a cabo un proceso de poda para eliminar conexiones redundantes de dominós, produce los corredores de menor longitud total dentro de las cotas inferior y superior en el menor tiempo de ejecución. Los resultados se encuentran publicados en las revistas indexadas *Revista de la Ingeniería Industrial* [González Gutiérrez et al., 2018] y *Visum Mundi* [González Gutiérrez et al., 2019].

(Palabras Clave: Embaldosados, Problemas NP-Complejos, Problema del Corredor de Longitud Mínima, Heurísticas Greedy, Algoritmos de ruteo.)

Abstract

The routing problem in rectilinear tiling surfaces is studied. Formally, the problem consists of, given a rectangle R divided into rectangles of size 1×2 (dominoes), find a set of edges on the periphery of R and dominoes, interconnected and free of cycles (tree or corridor) that connect all the dominoes from an access point on the periphery of R , and whose total length is the minimum. Thus stated, it is a sub-problem of the Minimum Length Corridor (MLC) problem, which is NP-complete, and consequently there is no efficient algorithm that produces optimal solutions. Therefore, the objective is to design routing algorithms on rectilinear tiling surfaces using a methodology based on the greedy design technique that produce a corridor of minimum total length. For this, families of tiling instances are designed for the experimental analysis of the routing algorithms. Several techniques are presented for the enumeration of tilings in general, while a methodology is used according to the backtracking technique for the generation of non-isomorphic and cut-free families of tilings of size 6×5 , 6×7 , 6×8 , 6×9 , 6×10 and 7×8 . On these families, the experimental analysis of four heuristics is carried out, measuring their performance and the solution, under the criteria of connecting first: closest dominoes, furthest dominoes, shared vertices (improved), internal vertices (using reduction). The heuristics are implemented in ©Mathematica version 11.3.0.0 on a MacBook Pro computer: 2.6 GHz Intel Core *i7* CPU, 6 cores, and 16 GB of RAM. As a result of the experimental analysis, it is established that there are lower and upper bounds for the total length of the corridor, among which the optimal solution resides. Likewise, it is established that the heuristic based on the criterion of connecting all the internal vertices, according to the Kruskal and Prim algorithms, and then carrying out a pruning process to eliminate redundant connections of

dominoes, produces the shortest corridors within the lower and upper bounds in the shortest execution time. The results are published in the indexed journals *Revista de la Ingeniería Industrial* [González Gutiérrez et al., 2018] y *Visum Mundi* [González Gutiérrez et al., 2019].

(Keywords: Tiling, NP-Complete Problems, Minimum Length Corridor Problem, Greedy Heuristics, Routing Algorithms.)

Índice general

Reconocimientos

Resumen

Abstract

Índice **II**

Índice de Figuras **VI**

Índice de Tablas **XIV**

1. Introducción. **2**

1.1. Prólogo. 2

1.2. Descripción del problema. 3

1.2.1. Definición del Problema MLC. 4

1.2.2. Modelación matemática discreta. 7

1.3. Justificación. 15

1.4. Hipótesis. 17

1.5. Objetivos. 18

1.5.1. Objetivo general. 18

1.5.2. Objetivos específicos. 19

2. Marco de referencia. **21**

2.1. Embaldosados. 21

2.2. Estructura de Grafo y Árbol. 24

2.2.1. Modelación matemática. 24

2.2.2. Técnicas de procesamiento.	28
2.3. Algoritmos Eficientes de Procesamiento.	31
2.3.1. Algoritmo de Dijkstra.	31
2.3.2. Algoritmos de Prim y Kruskal.	34
2.4. Complejidades Algorítmicas.	39
2.5. El problema del agente viajero.	41
2.6. El problema del “profesor perfecto”.	43
3. Algoritmos de enumeración y generación de embaldosados con dominós.	45
3.1. Modelado de embaldosados.	45
3.2. Metodología.	50
3.2.1. Generación de embaldosados.	53
3.2.2. Generando embaldosados no-isomorfos.	55
3.2.3. Generando embaldosados libres de cortes.	58
4. Algoritmos de ruteo	62
4.1. Instancia geométrica	64
4.1.1. Caracterización de la instancia	64
4.1.2. Modelo discreto de la instancia	66
4.1.3. Matriz de incidencia del grafo	70
4.2. Diseño de algoritmos para el cálculo del Corredor de Longitud Mínima (MLC)	71
4.2.1. Heurística 1: Algoritmo de Búsqueda Voraz – Dominós más ceranos	72
4.2.2. Heurística 2: Algoritmo de Búsqueda Voraz – Dominós más lejanos.	79
4.2.3. Heurística 3: Algoritmo de Vértices Compartidos	87

4.2.4.	Heurística 4: Algoritmo de Vértices Compartidos Mejorado	94
4.2.5.	Heurística 5: Algoritmo de Árbol Extendido de Costo Mínimo – Reducción	103
4.2.6.	Heurística 6: Algoritmo de Árbol Extendido de Costo Mínimo con Vértices Internos – Reducción	112
5.	Resultados.	122
5.1.	Embaldosados no isomorfos libres de corte.	123
5.1.1.	Generando todos los embaldosados.	124
5.1.2.	Generando embaldosados no-isomorfos.	125
5.1.3.	Generando embaldosados libres de cortes.	127
5.1.4.	Embaldosados con una y dos líneas de corte.	128
5.1.5.	Midiendo tiempos de ejecución.	130
5.1.6.	Selección de instancias geométricas.	131
5.2.	Corredores de longitud mínima.	132
5.2.1.	Vértices de Acceso y Cotas para Corredores en Mallas y Dominós.	132
5.2.2.	Características de Instancias para Experimentación.	146
5.2.3.	Análisis de Corredores con Vértice de Acceso en Esquina generados por las Heurísticas.	150
5.2.4.	Análisis de Corredores con Vértice de Acceso en Borde pero no en Esquinas generados por las Heurísticas.	170
5.2.5.	Tiempos de procesamiento por heurística.	187
6.	Conclusiones y trabajos futuros.	200
6.1.	Conclusiones.	200
6.2.	Líneas de investigación.	204

Bibliografía	207
Referencias	211
Apéndice	212
A. Publicaciones en Revistas Indexadas	212
B. Materiales: Datos, Figuras, Programas, CDF.	213

Índice de Figuras

1.1. Árbol extendido de longitud mínima.	3
1.2. Corredores para la misma instancia geométrica.	5
1.3. Corredor óptimo para una instancia geométrica del problema <i>TRA-MLC</i>	6
1.4. Árbol de longitud mínima donde cada habitación incide a un vértice del árbol.	7
1.5. Grafo no dirigido conexo $G = (V, E)$	8
1.6. Grafo no dirigido conexo ponderado $G = (V, E, w)$	9
1.7. Generalización del árbol extendido de costo mínimo.	11
1.8. Generalizaciones del problema de Árbol de Steiner.	13
1.9. Instancia de 5000×5000	14
1.10. Modelado de la instancia por una red.	15
2.1. Ejemplos de embaldosados.	22
2.2. Ciudad de Königsberg.	25
2.3. Modelación de los puentes de Königsberg con un grafo.	26
2.4. Grafo Conexo Ponderado.	28
2.5. Árbol Extendido de Costo Mínimo.	28
2.6. Grafo de tránsito vehicular de la ciudad de Querétaro (Universidad, 5 de febrero, Corregidora y Constituyentes).	37
2.7. Peso de la Ruta Corta: Centro Universitario UAQ y Alameda.	38
2.8. Tour para 13,509 ciudades de Estados Unidos de América.	42
3.1. Los 11 embaldosados del rectángulo 4×3	46
3.2. Indexación del rectángulo 2×4	50

4.1. Corredores en un instancia R con 23 rectángulos rectilíneos.	63
4.2. Instancia Geométrica R de tamaño 5000×5000	64
4.3. Modelo discreto de la instancia geométrica R	67
4.4. Matriz de incidencia del grafo $G = (V, E, w)$	71
4.5. Rectángulos 4 y 10 alcanzados por el vértice raíz $v = 18$ del árbol solución.	73
4.6. Rectángulo 8 cercano al vértice raíz $v = 18$ a través del vértice 17 y la arista $\{18, 17\}$	74
4.7. Corredor de longitud mínima de 11,500: mejor caso.	78
4.8. Longitud del corredor 14,000 y Punto de acceso 5: peor caso.	79
4.9. Rectángulos 4 y 10 alcanzados por el vértice raíz $v = 18$ del árbol solución.	80
4.10. Rectángulo 1 más alejado con distancia mínima al vértice raíz $v = 18$ a través del vértice 10 y las aristas $\{\{18, 17\}, \{17, 16\}, \{16, 15\}, \{15, 14\}, \{14, 10\}\}$ alcanzando también los rectángulos 5, 7, 8 y 9.	81
4.11. Corredor de longitud mínima de 11,000: mejor caso.	86
4.12. Longitud del corredor 14,500 y Punto de acceso 3: peor caso.	87
4.13. Vértices con mayor incidencia: vértice 6 incidente sobre los rectángulos 1, 2 y 5; vértice 12 incidente sobre los rectángulos 3, 6 y 8; vértice 20 incidente sobre los rectángulos 7, 9 y 11; vértice 22 incidente sobre los rectángulos 10, 12 y 13; vértice 4 incidente sobre los rectángulos 3 y 4; vértice 24 incidente sobre los rectángulos 11 y 14.	88
4.14. Corredor de longitud mínima de 11,500: mejor caso.	93
4.15. Longitud del corredor 14,000 y Punto de acceso 5: peor caso.	94

4.16. Vértices internos con mayor grado de incidencia: vértice 6 incidente sobre los rectángulos 1, 2 y 5; vértice 12 incidente sobre los rectángulos 3, 6 y 8; vértice 20 incidente sobre los rectángulos 7, 9 y 11; vértice 22 incidente sobre los rectángulos 10, 12 y 13; vértice 13 incidente sobre los rectángulos 3, 4 y 8; vértice 25 incidente sobre los rectángulos 9, 11 y 14.	96
4.17. Rectángulos externos e internos de la instancia R	96
4.18. Instancia geométrica R dividida en rectángulos.	97
4.19. Vértices y Aristas del Árbol Interno.	100
4.20. Mejor caso del Corredor de Longitud Mínima.	102
4.21. Peor caso del Corredor de Longitud Mínima.	102
4.22. Árbol extendido de costo mínimo de la instancia geométrica.	104
4.23. Eliminación del vértice hoja 1 y su arista $\{1, 2\}$, seguido por el vértice hoja 2 y su arista $\{2, 6\}$ del árbol extendido de costo mínimo de la instancia geométrica.	104
4.24. Corredor de longitud mínima de 12, 500: mejor caso.	110
4.25. Corredor de longitud mínima de 15, 000: peor caso.	111
4.26. Árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.	113
4.27. Eliminación del vértice hoja 10 y su aristas $\{10, 6\}$, seguido por el vértice hoja 25 y su arista $\{25, 20\}$ del árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.	113
4.28. Árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.	116
4.29. Corredor de longitud mínima de 12, 500: mejor caso.	118
4.30. Corredor de longitud mínima de 14, 500: peor caso.	119

5.1. Embaldosados del cuadrado de 4×4	125
5.2. Embaldosados no-isomorfos del cuadrado 4×4	126
5.3. Embaldosados sin líneas de corte de un rectángulo 5×6	127
5.4. Caso rectángulo 7×6 : 32 embaldosados no isomorfos sin líneas de corte.	128
5.5. Embaldosados con una línea de corte de área menor o igual a 25.	128
5.6. Embaldosados con dos líneas de corte de área menor o igual a 16.	129
5.7. Vértices en esquina y de borde en una instancia de tamaño 6×5 de dominós.	133
5.8. Instancia de Malla de tamaño $n \times m$	134
5.9. Árbol solución de la Instancia de Malla I_M de tamaño $n \times m$	135
5.10. Instancias de dominós I_D de tamaño 6×5	141
5.11. Instancia de malla I'_M sobre dominós.	142
5.12. Instancias de dominós I_D	143
5.13. Instancias de malla I'_M sobre dominós.	144
5.14. Ejemplos de embaldosados.	149
5.15. Distribución de corredores en instancias de tamaño 6×5 con lon- gitud dentro y fuera de las cotas.	152
5.16. Distribución de corredores en instancias de tamaño 6×5 con lon- gitud dentro de los límites de las cotas.	153
5.17. Distribución de corredores en instancias de tamaño 6×5 con lon- gitud mayor a la cota superior.	153
5.18. Distribución de corredores en instancias de tamaño 6×7 con lon- gitud dentro y fuera de las cotas.	155
5.19. Distribución de corredores en instancias de tamaño 6×7 con lon- gitud dentro de los límites de las cotas.	156

5.20. Distribución de corredores en instancias de tamaño 6×7 con longitud mayor a la cota superior.	156
5.21. Distribución de corredores en instancias de tamaño 6×8 con longitud dentro y fuera de las cotas.	158
5.22. Distribución de corredores en instancias de tamaño 6×8 con longitud dentro de los límites de las cotas.	159
5.23. Distribución de corredores en instancias de tamaño 6×8 con longitud mayor a la cota superior.	159
5.24. Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro y fuera de las cotas.	161
5.25. Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro de las cotas.	162
5.26. Distribución de corredores en instancias de tamaño 6×9 con longitudes mayor a la cota superior.	162
5.27. Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro y fuera de las cotas.	164
5.28. Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro de los límites de las cotas.	165
5.29. Distribución de corredores en instancias de tamaño 6×10 con longitudes mayor a la cota superior.	166
5.30. Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro y fuera de las cotas.	168
5.31. Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro de los límites de las cotas.	169
5.32. Distribución de corredores en instancias de tamaño 7×8 con longitudes mayor a la cota superior.	170

5.33. Distribución de corredores en instancias de tamaño 6×5 con longitud dentro de las cotas.	173
5.34. Distribución de corredores en instancias de tamaño 6×7 con longitud dentro y fuera de las cotas.	174
5.35. Distribución de corredores en instancias de tamaño 6×7 con longitud dentro de los límites de las cotas.	175
5.36. Distribución de corredores en instancias de tamaño 6×7 con longitud mayor a la cota superior.	175
5.37. Distribución de corredores en instancias de tamaño 6×8 con longitud dentro y fuera de las cotas.	177
5.38. Distribución de corredores en instancias de tamaño 6×8 con longitud dentro de las cotas.	178
5.39. Distribución de corredores en instancias de tamaño 6×8 con longitud mayor a la cota superior.	178
5.40. Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro y fuera de las cotas.	180
5.41. Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro de las cotas.	181
5.42. Distribución de corredores en instancias de tamaño 6×9 con longitudes mayor a la cota superior.	181
5.43. Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro y fuera de las cotas.	183
5.44. Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro de las cotas.	184
5.45. Distribución de corredores en instancias de tamaño 6×10 con longitudes mayor a la cota superior.	184

5.46. Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro y fuera de las cotas.	186
5.47. Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro de las cotas.	187
5.48. Distribución de corredores en instancias de tamaño 7×8 con longitudes mayor a la cota superior.	187
5.49. Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.	188
5.50. Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.	189
5.51. Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.	189
5.52. Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.	190
5.53. Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.	190
5.54. Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.	191
5.55. Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.	191
5.56. Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.	191
5.57. Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.	192
5.58. Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.	192

5.59. Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.	192
5.60. Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.	192
5.61. Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.	193
5.62. Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.	193
5.63. Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.	193
5.64. Comparación de Tiempos Promedios del Algoritmo en la genera- ción de las familias de corredores para las cuatro heurísticas.	196
5.65. Comparación de Tiempos Promedios en la generación de las fa- milias de corredores para las cuatro heurísticas.	197
5.66. Comparación de Tiempos Promedios del Algoritmo en la genera- ción de corredores para las familias de tamaño 6×5 , 6×7 , 6×8 , 6×9 y 6×10 para las cuatro heurísticas.	198
5.67. Comparación de Tiempos Promedios del Algoritmo en la genera- ción de corredores para las familias de tamaño 7×6 y 7×8 para las cuatro heurísticas.	198

Índice de Tablas

1.1. Variaciones del Problema MLC.	4
1.2. Algoritmos de solución.	9
2.1. Baldosas y embaldosados.	23
3.1. Valores de $t(n, m)$ para $n, m \leq 10$	47
4.1. Caracterización de instancia de Fig. 4.2.	65
4.2. Rectángulos ordenados.	66
4.3. Vértices de los Rectángulos.	68
4.4. Vértices de las cuatro esquinas de los rectángulos.	69
4.5. Vértices adyacentes y pesos de las aristas.	70
4.6. Corredores de Longitud Mínima.	77
4.7. Corredores de Longitud Mínima.	85
4.8. Corredores de Longitud Mínima.	92
4.9. Información del Árbol Extendido de Costo Mínimo Modificado. . .	100
4.10. Información del Árbol Extendido de Costo Mínimo.	101
4.11. Corredores de Longitud Mínima.	108
4.12. Información del Árbol Extendido de Costo Mínimo.	116
4.13. Información del Árbol Extendido de Costo Mínimo.	117
5.1. Número de embaldosados no-isomorfos para $n, m \leq 7$	126
5.2. Número de Embaldosados libres de líneas de corte.	127
5.3. Número de Embaldosados con una línea de corte.	129
5.4. Número de Embaldosados con dos líneas de corte.	130
5.5. Tiempo de ejecución (seg).	131

5.6. Cantidad de embaldosados y tiempo de generación (seg).	131
5.7. Límite de Cota Inferior y Superior en I_D con Vértice de Acceso en Esquina.	144
5.8. Límite de Cota Inferior y Superior en I_D con Vértice de Acceso en Borde pero no en Esquina.	145
5.9. Cantidad de Embaldosados por Tamaño de Instancia.	147
5.10. Distribución de Corredores de acuerdo al número de nodos en la periferia.	148
5.11. Cantidad y porcentaje de corredores en instancias de tamaño 6×5 por longitud para cada heurística.	151
5.12. Cantidad y porcentaje de corredores en instancias de tamaño 6×7 por longitud para cada heurística.	154
5.13. Cantidad y porcentaje de corredores en instancias de tamaño 6×8 por longitud para cada heurística.	157
5.14. Cantidad y porcentaje de corredores en instancias de tamaño 6×9 por longitud para cada heurística.	160
5.15. Cantidad y porcentaje de corredores en instancias de tamaño 6×10 por longitud para cada heurística.	163
5.16. Cantidad y porcentaje de corredores en instancias de tamaño 7×8 por longitud para cada heurística.	167
5.17. Cantidad y porcentaje de corredores en instancias de tamaño 6×5 por longitud para cada heurística.	172
5.18. Cantidad y porcentaje de corredores en instancias de tamaño 6×7 por longitud para cada heurística.	174
5.19. Cantidad y porcentaje de corredores en instancias de tamaño 6×8 por longitud para cada heurística.	176

5.20. Cantidad y porcentaje de corredores en instancias de tamaño 6×9 por longitud para cada heurística.	179
5.21. Cantidad y porcentaje de corredores en instancias de tamaño 6×10 por longitud para cada heurística.	182
5.22. Cantidad y porcentaje de corredores en instancias de tamaño 7×8 por longitud para cada heurística.	185
5.23. Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 1 en la generación de la familia de corredores.	194
5.24. Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 2 en la generación de la familia de corredores.	194
5.25. Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 3 en la generación de la familia de corredores.	195
5.26. Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 4 en la generación de la familia de corredores.	195

**”Lo que sabemos es una gota, lo que ignoramos,
un inmenso océano. La admirable disposición y
armonía del universo no ha podido sino salir del
plan de un Ser omnisciente y omnipotente”**

Isaac Newton (1643-1727)

Físico, teólogo, inventor, alquimista y matemático inglés

Introducción.

1.1. Prólogo.

El presente trabajo de tesis está orientado al análisis y solución, desde un punto de vista algorítmico, de problemas de optimización en redes. Aquí, la solución siempre obedece al criterio de maximizar, digamos ganancias, o minimizar, por ejemplo pérdidas. El enfoque de solución de dichos problemas de optimización que se utiliza es mediante procedimientos consistentes en pasos lógicos ordenados, y particularmente se concentra en aquellos problemas que pueden ser representados por medio del modelo matemático llamado red. Se precisa entonces qué es una red y posteriormente se presenta dos problemas relevantes en este proyecto de investigación.

Una red, que también suele llamarse grafo o gráfica, consiste básicamente en un conjunto de puntos, vértices o nodos, y un conjunto de líneas, aristas o arcos que unen ciertos vértices. Este modelo matemático llamado grafo provee de un modelo computacional que permite representar la estructura subyacente de una amplia gama de problema de optimización, en los cuales se pretende imponer una función objetivo a minimizar (o maximizar en otros contextos).

1.2.1. Definición del Problema MLC.

Dado el par (R, P) , donde R es un rectángulo particionado en un conjunto P de polígonos rectilíneos P_1, P_2, \dots, P_r , encontrar un conjunto S de segmentos de línea que se encuentran a lo largo de las aristas de R y de los polígonos rectilíneos en P . Los segmentos en S forman un árbol o *corredor*, e incluye al menos un punto a lo largo de las aristas límite de R , llamado punto de acceso, y al menos un punto del límite a lo largo de las aristas de cada uno de los polígonos rectilíneos. La suma de la longitud de los segmentos de línea en S es llamada la longitud o simplemente longitud de S y se denota por $L(S)$.

En la Tabla 1.1 se concentran algunas de las variaciones del problema MLC.

Tabla 1.1: Variaciones del Problema MLC.

Problema	Características
MLC	Rectángulo particionado en polígonos rectilíneos.
$MLC - R$	Rectángulo particionado en rectángulos.
$TRA - MLC$	El problema MLC incluyendo la esquina superior derecha del rectángulo R como punto de acceso (Top Right Access).
$TRA - MLC - R$	El problema $MLC - R$ incluyendo la esquina superior derecha del rectángulo R como punto de acceso.
MLC_f	El problema MLC permitiendo un conjunto o bosque de corredores parciales en lugar de un solo corredor.
$MA - MLC_f$	El problema MLC_f con múltiples puntos de acceso. Cada corredor parcial tiene su nodo raíz como punto de acceso localizado a lo largo de R . Cuando la esquina superior derecha de R es el único punto de acceso, el problema $MA - MLC_f$ corresponde al problema $TRA - MLC$.

El problema del Corredor de Longitud Mínima ($MLC - Minimum Length Corridor$) consiste en obtener un conjunto de segmentos de líneas que conecten un vértice localizado sobre la periferia de R con al menos un punto de cada

polígono rectilíneo resultante de la partición de R , de tal manera que la suma de la longitud de sus segmentos de línea sea la mínima posible. [Gonzalez, 2014]

En la Figura 1.2 se muestran dos instancias: la primera corresponde a una solución factible aunque no la óptima, la segunda corresponde a un corredor óptimo para la misma instancia geométrica.

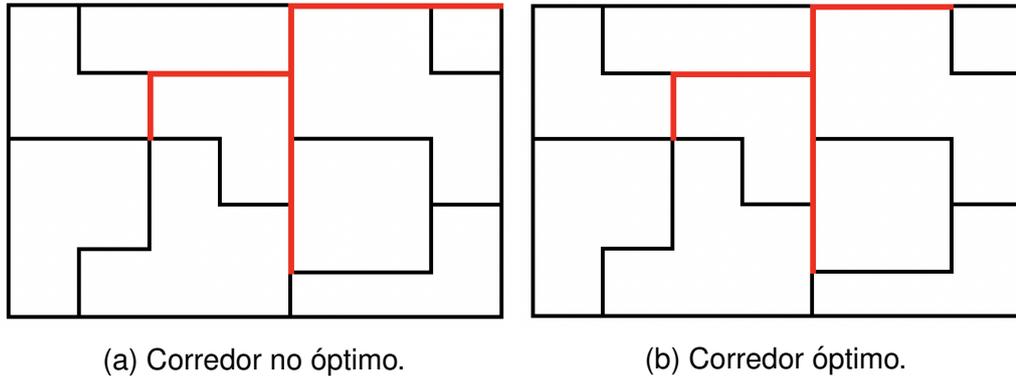


Figura 1.2: Corredores para la misma instancia geométrica.

Una versión restringida del problema MLC propuesto por Eppstein es cuando los polígonos rectilíneos son rectángulos, llamado el problema $MLC-R$ [Eppstein, 2001]. Este problema, bajo las restricciones de un rectángulo particionado en rectángulos, cae en la categoría de los problemas NP-completos, lo cual fue demostrado [Gonzalez-Gutierrez and Gonzalez, 2007] y presentado el primer algoritmo de aproximación de razón constante de tiempo polinomial [Gonzalez-Gutierrez and Gonzalez, 2010].

El problema $TRA - MLC$ (Top Right Access Point Minimum Length Corridor) consiste en encontrar un corredor de longitud mínima sobre un rectángulo particionado en polígonos y un punto de acceso localizado en la esquina superior derecha del rectángulo. La Figura 1.3 muestra el caso de un corredor óptimo para este tipo de instancias. Priyadarsini *et al* demuestran que el problema $TRA - MLC$ es NP-Completo [Priyadarsini, 2007].

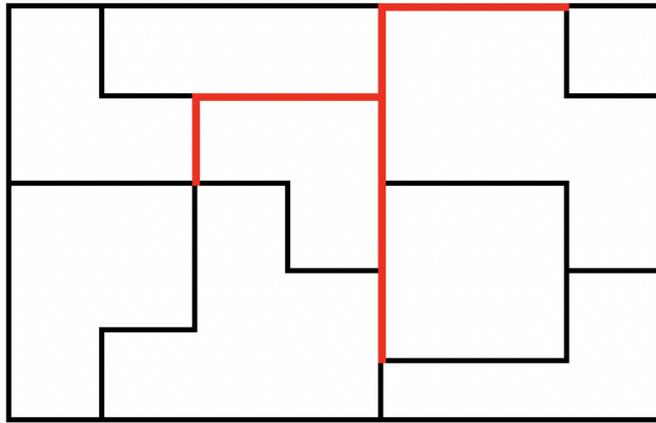


Figura 1.4: Árbol de longitud mínima donde cada habitación incide a un vértice del árbol.

El problema MLC_k es una generalización del problema $MLC - R$ donde los rectángulos son c -gonos rectilíneos para un valor de c donde $c \leq k$ y k es una constante. Se presenta el primer algoritmo de aproximación de tiempo polinomial de relación constante para la generalización del Problema del Agente Viajero (GTSP) basado en la técnica de restricción y aproximación de relajación [Gonzalez-Gutierrez and Gonzalez, 2010].

1.2.2. Modelación matemática discreta.

El modelo matemático discreto que subyace en la representación de la instancia geométrica del corredor de longitud mínima es la estructura de datos *grafo*. Esta estructura matemática consiste en dos conjuntos; uno de nodos o vértices V y el otro de arcos o aristas E . Cinco características importantes en esta estructura en el contexto del problema son:

- (a) Un grafo es *dirigido* cuando sus aristas o arcos tienen una dirección en la cual debe de ser recorrido, mientras que un grafo es *no - dirigido* cuando

sus arista o arcos pueden ser recorridas en ambas direcciones.

- (b) Un grafo es *conexo* si para cada par de vértices de V existe un conjunto de aristas en E que definen una ruta que los conecta.
- (c) Un grafo es *planar* cuando sus aristas no se cruzan en otros puntos diferentes a sus nodos.
- (d) Un grafo es *aciclico* cuando esta libre de ciclos o lazos.
- (e) Un grafo es *ponderado* si a cada arista del grafo se le asigna un peso $w \in \mathbb{R}^+$, el cual puede representar distancia, costo o tiempo.

En la Figura 1.5 se presenta un grafo $G = (V, E)$ el cual tiene $|V| = 5$ nodos y $|E| = 7$ aristas; los conjuntos de nodos y aristas son $V = \{a, b, c, d, e\}$ y $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}, \{e, b\}, \{e, c\}\}$. Mientras que en la Figura 1.6 se representa el caso de un grafo ponderado $G = (V, E, w)$ donde $w \in \mathbb{R}^+$ el cual tiene $|V| = 5$ nodos y $|E| = 7$ aristas; el conjunto de nodos $V = \{a, b, c, d, e\}$ y de aristas $E = \{\{a, b\}, \{b, c\}, \{c, d\}, \{d, e\}, \{e, a\}, \{e, b\}, \{e, c\}\}$; los pesos de cada arista son $w(a, b) = 8$, $w(b, c) = 5$, $w(c, d) = 6$, $w(d, e) = 1$, $w(e, a) = 9$, $w(e, b) = 2$, $w(e, c) = 3$.

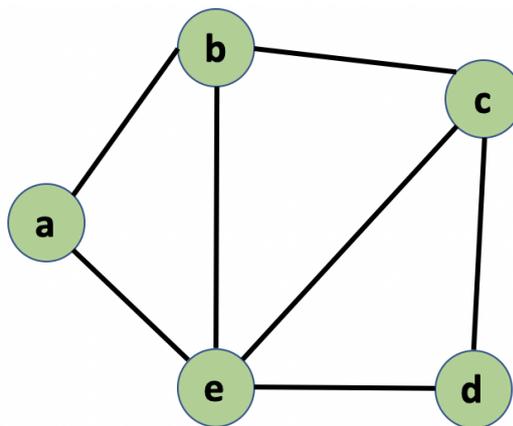


Figura 1.5: Grafo no dirigido conexo $G = (V, E)$.

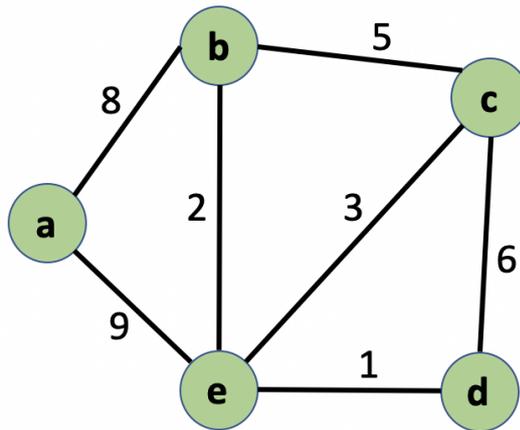


Figura 1.6: Grafo no dirigido conexo ponderado $G = (V, E, w)$.

En la Tabla 1.2 se muestran tres problemas y sus algoritmos, para los cuales, dado un grafo $G = (V, E, w)$ con una cantidad de nodos $|V| = n$, construir un subgrafo o árbol $G' = (V', E', w')$ donde $V' \subseteq V$, $1 < |V'| \leq n$, $E' \subseteq E$.

Tabla 1.2: Algoritmos de solución.

$ V' $	Problema: Algoritmo de solución	Complejidad
2	Shortest Path: Dijkstra, Bellman-Ford	Polinomial
3 a $(n - 1)$	Steiner Tree Problem	NP-Completo
n	Minimum-Cost Spanning Tree: Kruskal, Prim	Polinomial

El **primer caso** corresponde al subgrafo con $n = 2$ nodos, para el cual el algoritmo de Dijkstra o el algoritmo de Bellmand-Ford generan una ruta óptima en tiempo polinomial. Para el **segundo caso**, el subgrafo tiene entre 3 y $n - 1$ nodos para los cuales se busca un árbol que los interconecte. La solución se basa en el Problema de Árboles de Steiner, el cual es un problema NP-completo. En el **último caso** cuando el subgrafo tiene $|V'| = n$ nodos, se calcula el Árbol Extendido de Costo Mínimo a través de los algoritmos de Kruskal o Prim que tienen una complejidad polinomial.

Se pueden emplear los algoritmos del segundo y tercer caso para generar soluciones a problemas modelados con grafos en donde se obtenga una gene-

realización del problema de árbol extendido de costo mínimo o del problema del grupo de árbol de Steiner (GST - Group Steiner Tree).

La generalización del problema de Árbol Extendido de Costo Mínimo puede ser definida como:

Entrada: Un grafo ponderado no dirigido conexo $G = (V, E, w)$, donde el peso $w : E \rightarrow \mathbb{R}^+$ es una función y sus vértices están separados en particiones $\{S_1, S_2, \dots, S_k\}$, donde $V = S_1 \cup S_2 \cup \dots \cup S_k$.

Salida: Un árbol extendido de costo mínimo $T(S) = (V', E')$, donde las aristas $E' \subseteq E$ y los nodos $V' \subseteq V$ y $\forall i (V' \cap S_i) \neq \emptyset$, tal que al menos un terminal de cada conjunto S_i está en el árbol $T(S)$ y la longitud total de las aristas $\sum_{e \in E'} w(e)$ es mínima.

En la Figura 1.7 se presenta un ejemplo de la generalización de un árbol extendido de costo mínimo de un grafo conexo $G = (V, E, w)$; con $|V| = 22$ nodos y $|E| = 31$ aristas. Los nodos se agrupan en cuatro particiones $S_1 = \{1, 2, 5, 6, 7\}$, $S_2 = \{3, 4, 8, 9, 12, 13, 14\}$, $S_3 = \{10, 11, 17, 18, 19, 20\}$ y $S_4 = \{15, 16, 21, 22\}$, donde $V = S_1 \cup S_2 \cup S_3 \cup S_4$.

El problema consiste en obtener un árbol que conecte al menos un nodo de cada partición, cuya longitud total sea la mínima. Como se observa en la Figura 1.7 el árbol $T(S)$ conecta los nodos 7, 11, 12, 15 de las particiones S_1, S_3, S_2, S_4 , respectivamente para lo cual $\sum_{e \in E'} w(e)$ es mínima.

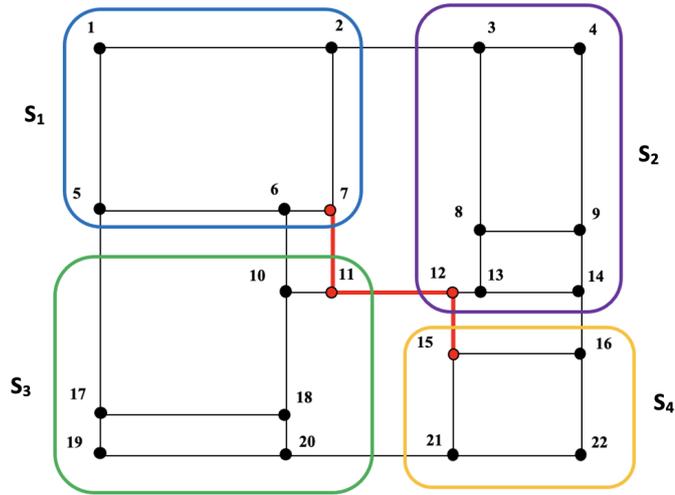


Figura 1.7: Generalización del árbol extendido de costo mínimo.

El problema del grupo de árbol de Steiner (GST, de Group Steiner Tree) puede verse como una generalización del problema del MLC. El problema GST puede definirse como [Gonzalez-Gutierrez and Gonzalez, 2007]:

Entrada: Un grafo ponderado no dirigido conexo $G = (V, E, w)$, donde $w : E \rightarrow \mathbb{R}^+$ es una función de peso; un subconjunto $S \subseteq V$, de terminales; y una partición $\{S_1, S_2, \dots, S_k\}$ de S .

Salida: Un árbol $T(S) = (V', E')$, donde $E' \subseteq E$ y $V' \subseteq V$, tal que al menos un terminal de cada conjunto S_i está en el árbol $T(S)$ y la longitud total de las aristas $\sum_{e \in E'} w(e)$ es mínima.

Una versión más general del problema GST es cuando $\{S_1, S_2, \dots, S_k\}$ de S no es una partición, sino que $\forall i S_i \subseteq S$, donde un vértice puede ser miembro de más de un conjunto S_i . Esta versión del problema GST se denomina Cobertura de Tarea de Árbol (TEC, de Tree Errand Cover).

En las Figuras 1.8a y 1.8b se presentan dos instancias para el problema

GST y el problema TEC respectivamente; se considera un grafo ponderado no dirigido conexo $G = (V, E, w)$ con $|V| = 22$ nodos y $|E| = 31$ aristas.

En el caso del problema GST, el conjunto V de vértices del grafo se encuentran en cinco particiones $S_1 = \{1, 2, 5, 6, 7\}$, $S_2 = \{3, 4, 8, 9\}$, $S_3 = \{10, 11\}$, $S_4 = \{17, 18, 19, 20\}$ y $S_5 = \{12, 13, 14, 15, 16, 21, 22\}$; se considera $S \subseteq V$, donde $S = S_1 \cup S_4 \cup S_5$. El problema consiste en obtener un árbol $T(S) = (V', E')$ que conecte un nodo de cada partición S_1, S_4, S_5 y que la longitud total de las aristas $\sum_{e \in E'} w(e)$ sea mínima. Como puede observarse en la Figura 1.8a los nodos del árbol $7 \in S_1$, $18 \in S_4$ y $12 \in S_5$ con los nodos de Steiner 10 y 11.

Para el problema TEC, considérese que V , el conjunto de vértices del grafo se encuentran distribuidos en cuatro subconjunto:

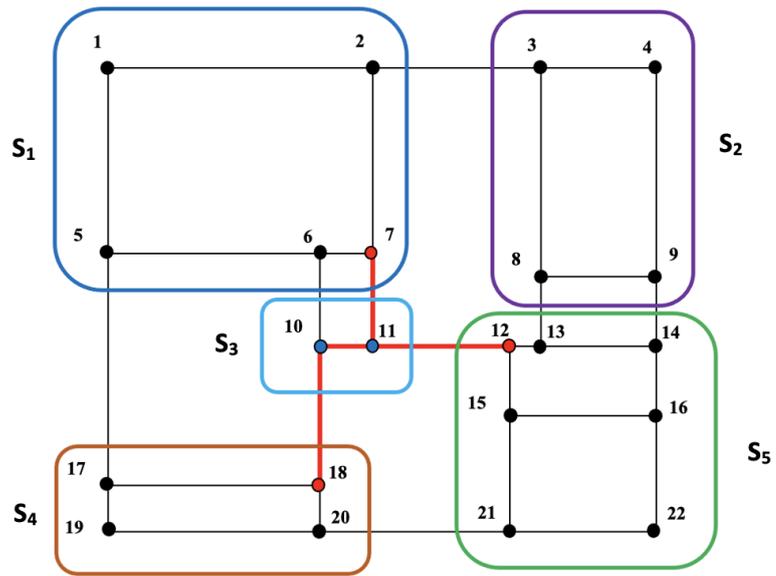
$$S_1 = \{1, 2, 3, 5, 6, 7, 8\},$$

$$S_2 = \{2, 3, 4, 7, 8, 9\},$$

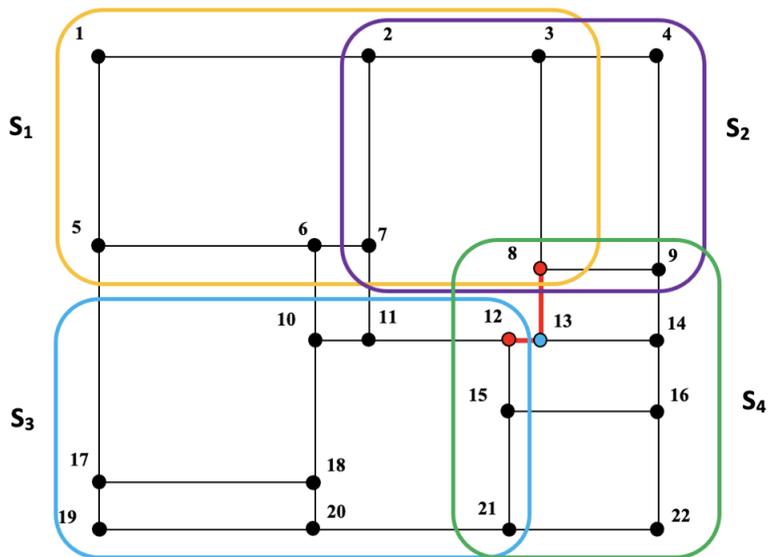
$$S_3 = \{10, 11, 12, 15, 17, 18, 19, 20, 21\} \text{ y}$$

$$S_4 = \{8, 9, 12, 13, 14, 15, 16, 21, 22\}$$

El problema consiste en obtener un árbol $T(S) = (V', E')$ que conecte un nodo de cada subconjunto S_1, S_2, S_3, S_4 y que la longitud total de las aristas $\sum_{e \in E'} w(e)$ sea mínima. Como puede observarse en la Figura 1.8b los nodos del árbol requeridos de cada subconjunto son $8 \in S_1, S_2, S_4$ y $12 \in S_3$ y el nodo de Steiner 13. En este caso es factible considerar los nodos que son comunes a dos o más subconjuntos con la finalidad de cubrir la mayor cantidad de subconjuntos con el menor número de nodos y aristas para conectar los nodos mas cercanos minimizando la distancia entre los nodos.



(a) Ejemplo problema GST.



(b) Ejemplo problema TEC.

Figura 1.8: Generalizaciones del problema de Árbol de Steiner.

Una red N es un grafo con valores numéricos en cada una de sus aristas que representan el peso de la misma. Resulta de especial interés construir un subgrafo conexo T de N de tal manera que T se extienda a todos los vértices de la red N considerando las aristas que resulten en un subgrafo T que tengan

el menor peso total posible. El subgrafo T es llamado árbol extendido de costo mínimo [Bondy and Murty, 2010].

Considere la Figura 1.9, correspondiente a una instancia de un rectángulo de 5000×5000 dividido en rectángulos más pequeños de diferentes tamaños.

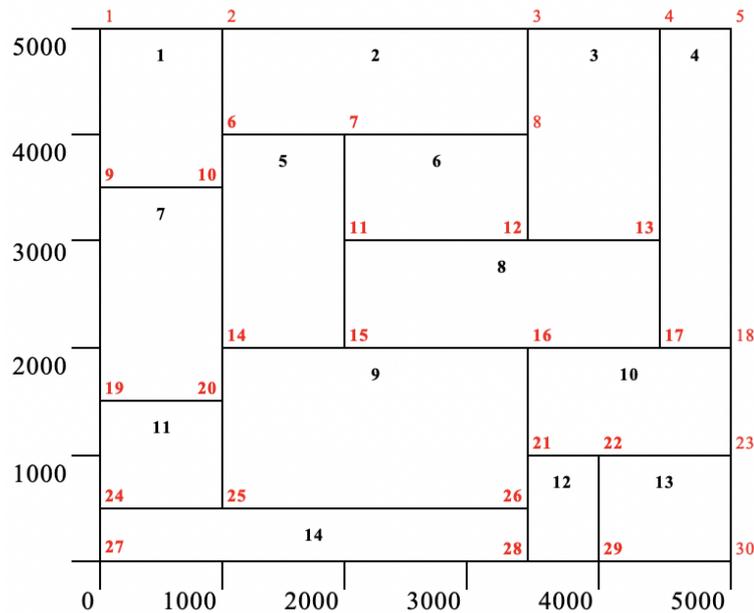


Figura 1.9: Instancia de 5000×5000 .

El problema que nos interesa resolver aquí es construir a partir de un vértice de acceso localizado en la periferia de la instancia, un árbol que toque al menos un vértice de cada rectángulo más pequeño y que tenga la distancia mínima, lo cual se desarrolla con mas detalla en los capítulos 3, 4 y 5.

Una forma interesante de visualizar esta instancia es considerando que cada rectángulo se puede representar por un nodo en una red, como se puede apreciar en la Figura 1.10, mientras que las adyacencias de los rectángulos se representa por las aristas. Así por ejemplo, el rectángulo 8 de la Figura 1.9 es representado por el *nodo* 8 en la Figura 1.10. Los rectángulos 5, 6, 3, 4, 10 y 9 son adyacentes al rectángulo 8, en el modelo de la red el *nodo* 8 esta conectado con

los nodos 5, 6, 3, 4, 10 y 9 a través de las aristas $(8, 5)$, $(8, 6)$, $(8, 3)$, $(8, 4)$, $(8, 10)$ y $(8, 9)$ que representan la adyacencia de los rectángulos.

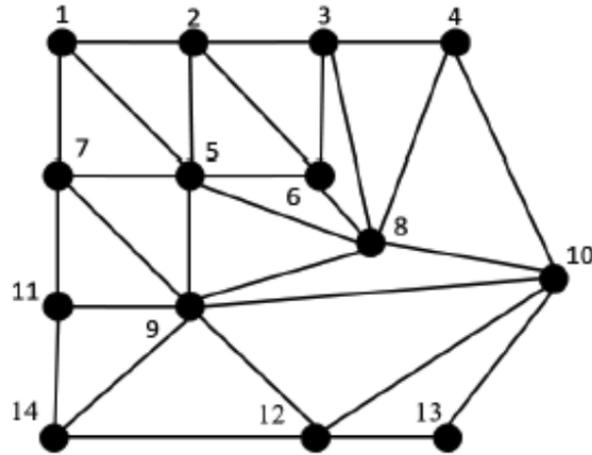


Figura 1.10: Modelado de la instancia por una red.

Este modelo de red de la instancia geométrica se pueden observar a través del grado de incidencia de cada nodo cuantos rectángulos son adyacentes y que comparten además nodos en la instancia que pueden ser comunes a 2 o más rectángulos.

1.3. Justificación.

Este trabajo de tesis se basa en los problemas del agente viajero y el profesor perfecto con el propósito de abordar el diseño e implementación de algoritmos de aproximación como solución al problema MLC, usando familias de instancias consistentes en superficies de embaldosados rectilíneos; particularmente aquellas conformadas por rectángulos (dominós) de dimensiones 1×2 . Con los resultados obtenidos a través del análisis comparativo de los algoritmos con estas instancias, se evaluará el comportamiento de los algoritmos para ser

utilizados en problemas de ruteo. La aplicación de estos algoritmos en diferentes áreas es importante; por ejemplo, en las grandes ciudades se pueden implementar para buscar alternativas de ruteo en sistemas de transporte público, así como en transporte privado para la distribución de bienes y servicios.

Así mismo, se aborda el problema de optimización consistente en encontrar un ordenamiento circular en un conjunto de puntos en el plano cartesiano, es decir el último punto se une al primero en el ordenamiento, tal que la longitud total del ciclo resultante sea la mínima. El problema es un caso más general que el problema de ciclos Hamiltonianos, y se conoce como el problema del agente viajero. En virtud de que este problema es intratable computacionalmente, esto es, pertenece a la categoría de problemas NP-Duros, se resuelve a través de algoritmos de aproximación de razón constante. En particular, se puede usar el algoritmo basado en árboles extendidos de costo mínimo, donde los recorridos pre, in, y post orden del árbol producido dictan el ordenamiento circular, y que resulta en una razón de aproximación constante, donde la solución aproximada es a lo más dos veces el valor de la solución óptima. El ordenamiento de puntos es entonces tratado para obtener la ruta óptima entre cada par de puntos consecutivos conforme a la métrica de distancia Manhattan sobre el grafo que representa el embalosado rectilíneo.

Este problema es de gran relevancia en el área de aplicación del modelado de redes de transporte dinámicas para la construcción de sistemas de transporte inteligentes. En la dirección de abordar el problema de redes de transporte dinámicas, uno de los problemas importantes es el diseño de algoritmos óptimos y eficientes para obtener la trayectoria entre dos puntos a través de un grafo ponderando. Así mismo, con el diseño de trayectorias fijas de enrutamiento de transporte público, para ello se diseñarán algoritmos eficientes que conecten

zonas rectilíneas que pueden representar zonas geográficas de demandas de servicios de transporte de modo tal que la longitud total de las trayectorias sea la mínima. Particularmente cuando se buscan soluciones óptimas y eficientes en la distribución de productos a través de una red de transportación tanto privada como pública.

El desarrollo de esta tesis se enmarca en el trabajo académico y de investigación del Cuerpo Académico Optimización y Computación Avanzada, orientado fuertemente en el sentido de contribuir con el fortalecimiento de las Líneas de Generación y/o Aplicación del Conocimiento que cultiva el cuerpo Académico de la Facultad de Ingeniería, a saber, la línea de Algoritmos, Optimización y Redes impulsando la innovación y la tecnología que contribuyan a la solución de problemas del Estado, y en particular de la ciudad de Querétaro. Asimismo, los resultados experimentales contribuirán a la solución de problemas básicos relevantes de las ciencias computacionales [Gonzalez-Gutierrez and Gonzalez, 2007], [Gonzalez-Gutierrez and Gonzalez, 2010].

1.4. Hipótesis.

Muchos de los problemas de ruteo, como el problemas del agente viajero, el profesor perfecto y el corredor de longitud mínima (MLCP), pertenecen a la categoría de problemas NP-completos, para los cuales se conjetura que no existe algoritmo eficiente; es decir, de orden de complejidad $O(n^k)$ para cualquier $k \in \mathbb{Z}^+$ y n correspondiente al tamaño de la instancia, tal que produzca soluciones óptimas [Applegate et al., 2011]. Recientemente, Gonzalez-Gutierrez A. *et al* demostraron que el tercer problema y muchos relacionados a el son NP-completos [Gonzalez-Gutierrez and Gonzalez, 2007].

En este contexto se plantea la siguiente hipótesis:

Existen algoritmos eficientes que producen soluciones óptimas o cuasi-óptimas para familias de instancias consistentes en superficies de embaldosados rectilíneos, como aquellas conformadas por rectángulos (dominós) de dimensiones 1×2 y las conformadas por cuadrículas regulares. Estas familias de instancias constituyen casos particulares al problema del corredor de longitud mínima mencionado.

1.5. Objetivos.

1.5.1. Objetivo general.

Diseñar algoritmos de ruteo en superficies de embaldosados rectilíneos utilizando estrategias greedy para optimizar el procesamiento en la solución de problemas de ruteo óptimo en sistemas de transporte tanto público como privado.

En el contexto del objetivo general: el qué de la investigación u objeto de estudio consiste en el diseño de los algoritmos de ruteo en superficies de embaldosados rectilíneos; el cómo de la investigación o método de solución consiste en el uso de técnicas o estrategias de diseño de algoritmos como Greedy (o voraz); y el para qué o resultados concretos consistente en la solución de problemas de ruteo para el estudio y diseño de sistemas de transporte público y privado.

1.5.2. Objetivos específicos.

1. Diseñar instancias de superficies geométricas de embaldosados rectilíneos no isomorfos libres de cortes para la evaluación de algoritmos de ruteo.
2. Diseñar soluciones algorítmicas eficientes al problema consistente en encontrar un árbol solución tal que la longitud total del recorrido del árbol resultante sea la mínima.
3. Realizar pruebas y análisis experimental de algoritmos de ruteo.
4. Construir un prototipo para la simulación de los problemas de ruteo en superficies divididas en polígonos rectilíneos.

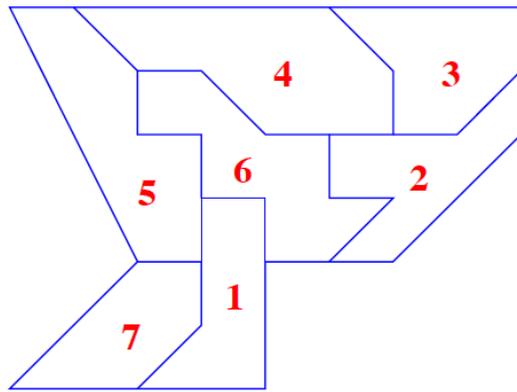
”Para mí, las matemáticas, las ciencias de la computación y las artes están relacionadas de una manera increíble. Todas son expresiones creativas”

*Sebastian Thrun (1967-)
Científico de la computación y profesor alemán*

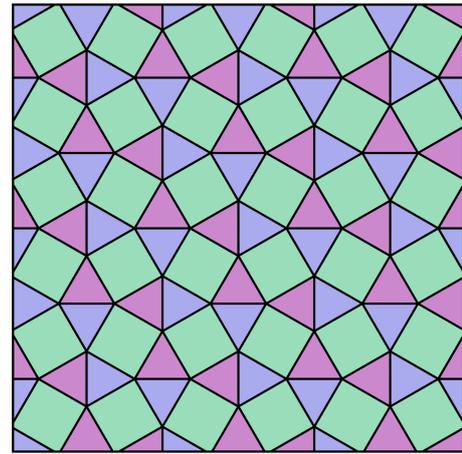
Marco de referencia.

2.1. Embaldosados.

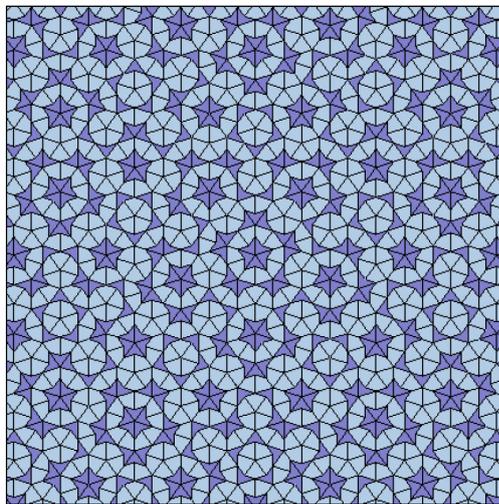
En el campo de la geometría, los embaldosados (tilings) son una generalización para llenar un espacio euclidiano de dos dimensiones utilizando objetos geométricos de la misma forma y tamaño con la finalidad de cubrir dicho espacio. En algunos casos de embaldosados, el espacio a cubrir y los objetos geométricos usados pueden ser irregulares, pero el objetivo es lograr cubrir el espacio completamente sin que los objetos se traslapen o que existan espacios sin ser cubiertos. Para llevar a cabo este proceso se permite voltear y girar las piezas con la finalidad de lograr el objetivo. En la Figura 2.1 se presentan diferentes tipos de embaldosados con objetos geométricos regulares e irregulares [Ardila and Stanley, 2004].



(a) Polígonos irregulares.



(b) Polígonos regulares.



(c) Penrose.



(d) Caballeros.

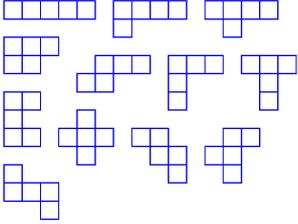
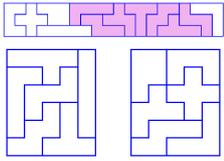
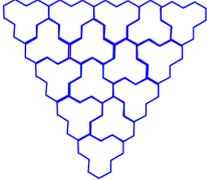
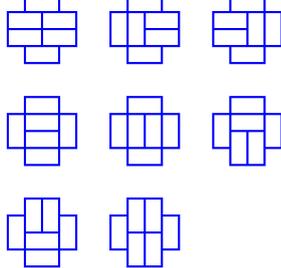
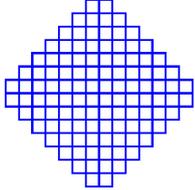
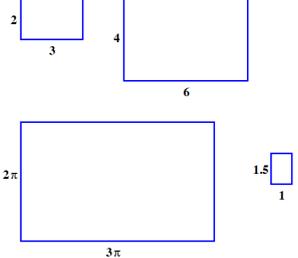
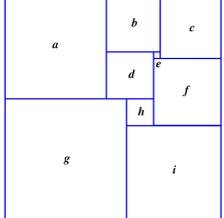
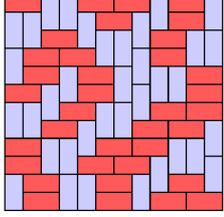
Figura 2.1: Ejemplos de embaldosados.

En este problema en particular convergen dos áreas importantes de las matemáticas computacionales: la geometría y la combinatoria; en donde resulta de particular interés saber cuántas instancias diferentes de embaldosados existen y además saber cuáles son. A través del desarrollo matemático combinatorial se conoce la cantidad de embaldosados que se pueden construir sobre una superficie, mientras que el uso de algoritmos eficientes y técnicas de programación permiten la generación de los embaldosados. En la Tabla 2.1 se presenta

algunos ejemplos de baldosas (tiles) y superficies geométricas regulares, así como algunos casos de embaldosados. [Ardila and Stanley, 2004]

El último caso representa la instancia geométrica que se va a considerar en este trabajo de tesis .

Tabla 2.1: Baldosas y embaldosados.

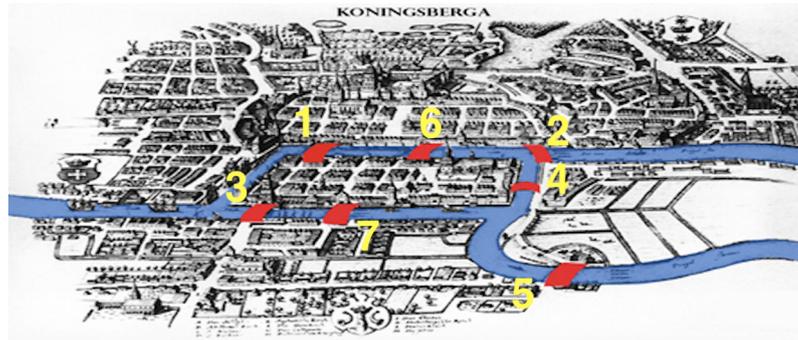
Tipo	Baldosa	Superficie	Embalosado
Pentominos		Rectángulo de área 60	
Tribone		Triángulos	
Diamante azteca		Rombo	
Rectángulos		Rectángulos	
Dominós		Rectángulos	

2.2. Estructura de Grafo y Árbol.

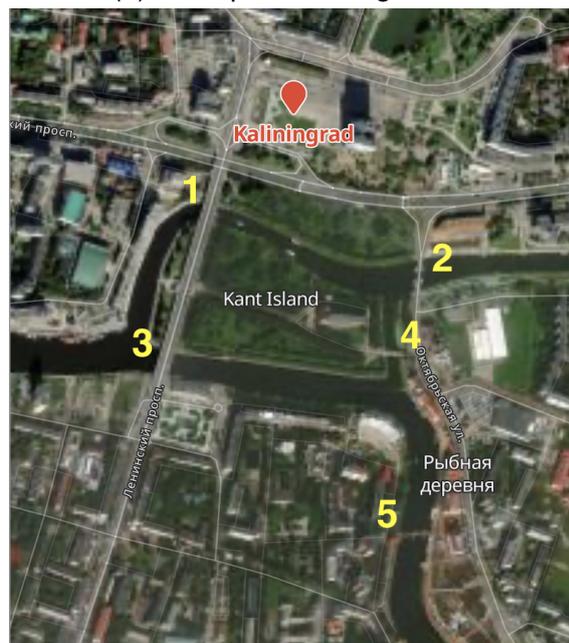
2.2.1. Modelación matemática.

En las ciencias computacionales se utilizan dos estructuras de datos que son importantes para modelar problemas: *los grafos* y *los árboles*. La Teoría de Grafos es una rama de las matemáticas discretas en donde se puede encontrar una extensa cantidad de aplicaciones en la ingeniería química, eléctrica, industrial, de tránsito, logística, entre otras.

El primer artículo sobre grafos fue escrito por el matemático y físico suizo Leonhard Euler (1707-1783), que fue publicado en 1736 por la Academia de Ciencias de San Petersburgo. Euler fue motivado a estudiar los grafos por el problema de los puentes de Königsberg (Leningrado, Rusia). En la ciudad de Königsberg existe una isla, llamada Kneiphoff, limitada por dos ramas del río Pregel en la que se construyeron siete puentes para cruzar las ramas del río. Actualmente solamente existen cinco puentes que conectan la isla con el resto de la ciudad, como se puede observar en la Figura 2.2.



(a) Siete puentes originales.



(b) Cinco puentes actuales.

Figura 2.2: Ciudad de Könisberg.

Euler planteo la siguiente pregunta: *¿Puede un habitante de Könisberg realizar un recorrido por la ciudad iniciando en su casa y atravesar cada puente exactamente una vez y regresar a su casa?* Él no solamente dio respuesta a este cuestionamiento sino estableció una metodología para resolver una clase más general de problemas.

El modelado con grafos del problema que trató Euler se muestra en la Figura 2.3, el cual corresponde a un grafo no dirigido debido a que sus aristas

no tienen un sentido definido. Cada vértice tiene un grado que corresponde al número de aristas que inciden sobre él. El nodo A tiene tres aristas que inciden sobre él; por lo que el $\text{grad}(A) = 3$ mientras que el $\text{grad}(B) = 5$, $\text{grad}(C) = 3$ y $\text{grad}(D) = 3$, observando que para cada vértice el grado es impar.

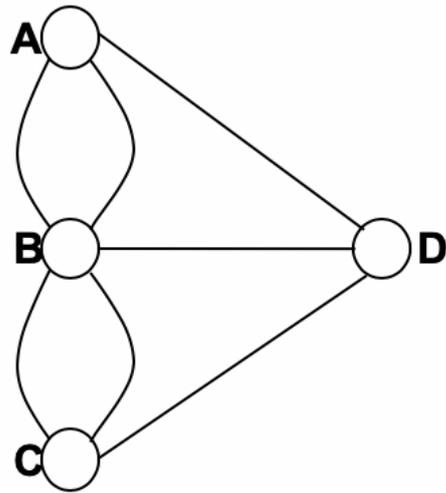


Figura 2.3: Modelación de los puentes de Königsberg con un grafo.

En un grafo se dice que existe un *circuito de Euler* si se recorren todas las aristas del grafo exactamente una vez con un recorrido cerrado; es decir, que se inicia y termina en el mismo vértice. Mientras que una *trayectoria de Euler* se recorre cada arista del grafo exactamente una vez con un recorrido abierto; es decir, que se inicia y termina en vértices diferentes. El problema de los puentes de Königsberg no tiene un circuito y tampoco una trayectoria, lo cual puede ser demostrado por los siguientes resultados [Hunter, 2009], [Epp, 2012].

Teorema 2.1 (Circuito de Euler). *Si todos los vértices de un grafo conexo G tienen grado par, entonces el grafo G tiene un circuito de Euler.*

Corolario 2.1 (Trayectoria de Euler). *Sea G un grafo y dos vértices distintos $v, w \in V$. Existe una trayectoria de Euler de v a w si y solo si G es conexo, v y w*

tienen grado impar y todos los otros vértices de G tienen grado par.

Otra área importante dentro de las matemáticas discretas es la Teoría de Árboles, particularmente un *árbol* extendido (spanning tree) de un grafo conexo no dirigido es un subgrafo acíclico conexo que contiene todos los vértices del grafo. Si las aristas del grafo tienen asignado un costo, se obtiene un árbol extendido de costo mínimo (minimum spanning tree). El Problema de Árbol Extendido de Costo Mínimo (Minimum Spanning Tree Problem) es interesante en el campo de las ciencias computacionales y encuentra aplicaciones en diversas áreas del conocimiento tales como la arqueología, biología, sociología y otras ciencias. El problema consiste en obtener un árbol que conecte todos los vértices de un grafo minimizando el costo total de las aristas del árbol. Una gran variedad de problemas, como el descrito, pueden ser modelados y solucionados mediante algoritmos greedy, los cuales hacen una elección óptima a nivel local con la esperanza de que esta elección conduzca a una solución global óptima [Levitin, 2012], [Cormen et al., 2013].

La Figura 2.4 muestra un grafo conexo ponderado no dirigido $G = (V, E, w)$ con $|V| = 9$ vértices y $|E| = 14$ aristas ponderadas; así mismo en la Figura 2.5 se muestra el árbol generado por el algoritmo de Prim o Kruskal mediante las aristas marcadas, conformando el árbol extendido de costo mínimo $T = (V, E')$ con $|V| = 9$ vértices y $|E'| = 8$ aristas donde $E' \subset E$.

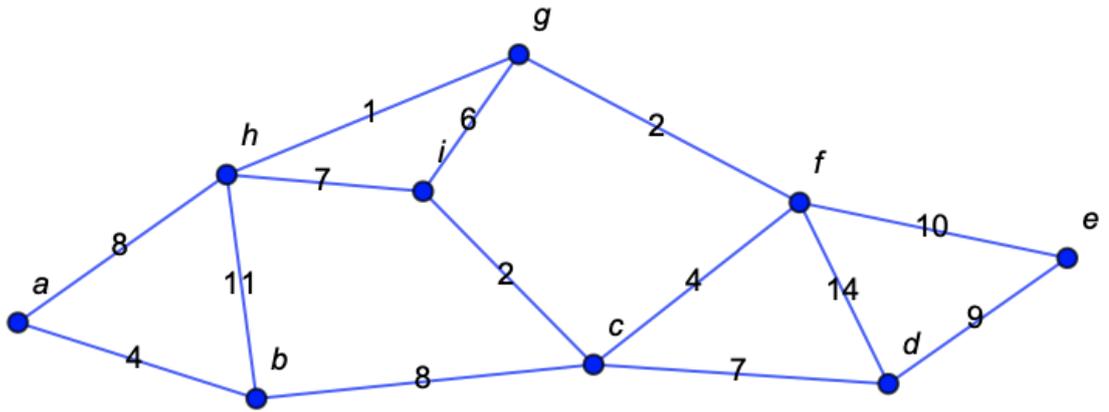


Figura 2.4: Grafo Conexo Ponderado.

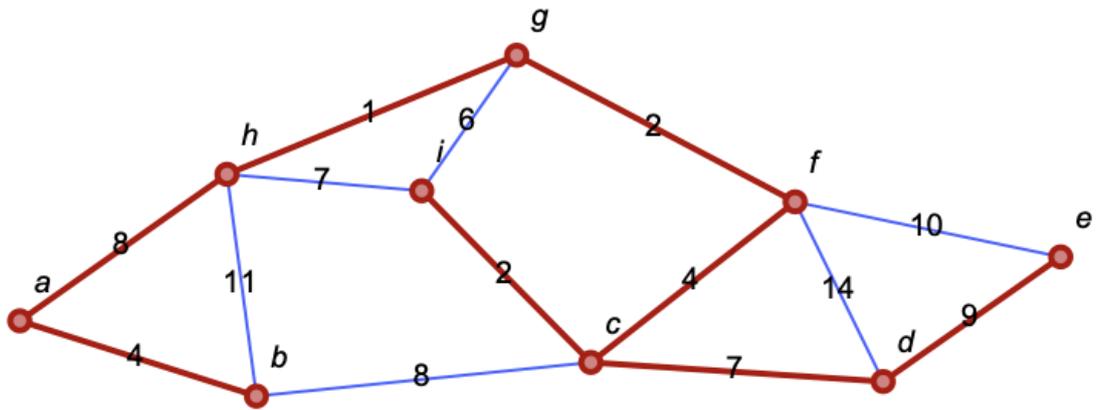


Figura 2.5: Árbol Extendido de Costo Mínimo.

2.2.2. Técnicas de procesamiento.

El desarrollo de técnicas y algoritmos de procesamiento para estructuras discretas de árboles y grafos surgen en la investigación de operaciones, una rama de las matemáticas. La finalidad de estas técnicas es optimizar los resultados en la búsqueda de rutas cortas o recorridos en grafos y multígrafos [Evans and Minieka, 1992].

Por su importancia en las aplicaciones, se han desarrollado algoritmos

basados en grafos con la finalidad de optimizar una función objetivo, esto es maximizarla o minimizarla, dependiendo del caso. Algoritmos como el de *Dijkstra* o *Bellman – Ford* son utilizados para encontrar la ruta más corta en un grafo dirigido conexo ponderado, o bien, los algoritmos de *Kruskal* o *Prim* para encontrar el árbol extendido de costo mínimo de un grafo conexo ponderado, son ejemplos de algoritmos de optimización aplicados a los modelos de redes basados en grafos [Hunter, 2009].

Considere un grafo $G = (V, E, w)$ conexo dirigido ponderado libre de lazos; donde V es el conjunto de nodos del grafo y E es el conjunto de aristas $e = (a, b)$. A cada arista se le asigna un número real positivo llamado el peso de la arista e , y el cual es denotado por $w(e)$ ó $w(a, b)$, para la función de peso $w : E \rightarrow \mathbb{R}^+$. Si $x, y \in V$ pero $(x, y) \notin E$ entonces $w(x, y) = \infty$. Para cualquier arista $e \in E$ el peso w puede representar:

- (a) La longitud de la arista que conecta el nodo a al nodo b .
- (b) El tiempo que se toma viajar directamente del vértice a al vértice b .
- (c) El costo de viajar de a hasta b a lo largo de la arista (a, b) .

La implementación de estas estructuras en la computadora determinan el marco y los algoritmos del método de optimización que facilitan representación y procesamiento. Dos tipos de problemas se pueden abordar [Grimaldi, 2004]:

- (a) La ruta más corta desde un vértice origen $v_0 \in V$ y todos los demás vértices $v \neq v_0 \in V$ en un grafo dirigido conexo libre de lazos (*single-source shortest-path*).
- (b) El árbol extendido para un grafo o multígrafo donde la suma de los pesos de las aristas en el árbol es mínima (*spanning tree*).

En el problema de *la ruta más corta desde un vértice fuente* es adecuado para resolver tres variantes del problema [Cormen et al., 2013]:

- (a) *La ruta más corta a un sólo nodo destino*: encuentra la ruta más corta a un vértice destino $t \in V$ a partir de cualquier vértice $v \neq t$ en V .
- (b) *La ruta más corta entre un par de vértices*: encuentra la ruta corta entre dos vértices dados de u, v .
- (c) *Las rutas más cortas para cada par de vértices*: encuentra la ruta corta de u a v para todos los pares de vértices $u, v \in V$.

Para dos vértices $a, b \in V$, $d(a, b)$ representa el costo o distancia total de la ruta más corta del nodo a al nodo b en el grafo G . Si no existe la ruta del nodo a al nodo b entonces $d(a, b) = \infty$. Para algún nodo $a \in V$, $d(a, a) = 0$. El peso de la ruta $p = \langle v_0, \dots, v_k \rangle$ es la suma de todos los pesos de las aristas que la conforman, como se muestra en la ecuación 2.1

$$d(v_0, v_k) = d(p) = \sum_{i=1}^k w(v_{i-1}, v_i) \quad (2.1)$$

El problema de *la ruta más corta desde un vértice fuente* requiere fijar un nodo fuente $v_0 \in V$ a partir del cual se encontrará para todos los nodos $v_k \in V$:

- (a) El peso $d(p)$ de la ruta corta $p = \langle v_0, \dots, v_k \rangle$.
- (b) La ruta corta $p = \langle v_0, \dots, v_k \rangle$ para todos los nodos $v_k \in V$, si el peso es finito.

El peso de la ruta corta del nodo v_0 al nodo v_k está definido por la ecuación

2.2.

$$\delta(v_0, v_k) = \begin{cases} \min(d(p) : v_0 \xrightarrow{p} v_k) & \text{si hay una ruta de } v_0 \text{ a } v_k \\ \infty & \text{en otro caso} \end{cases} \quad (2.2)$$

2.3. Algoritmos Eficientes de Procesamiento.

2.3.1. Algoritmo de Dijkstra.

El matemático y físico holandés Edsger Wybe Dijkstra (1930-2002) diseñó en 1959 un algoritmo que resuelve el problema de rutas cortas sobre un grafo ponderado $G = (V, E, w)$, donde cada arista $(u, v) \in E$ tienen un peso positivo $w(u, v) \geq 0$. A través de éste, se puede encontrar la distancia corta de un vértice fuente s a todos los otros vértices del grafo, así como la ruta corta dirigida para cada uno de estos vértices [Grimaldi, 2004], [Rosen, 2004].

El tiempo de ejecución del algoritmo de Dijkstra, usando un arreglo lineal para la cola de prioridad Q , es $O(|V|^2 + |E|) = O(|V|^2)$. Pero si se implementa la cola de prioridad Q como una estructura de datos heap binaria, el tiempo de ejecución es $O((|E| + |V|) \log(|V|)) = O(|E| \log(|V|))$. Más aún, si $|E| = O(|V|)$, entonces el algoritmo se ejecuta en tiempo $O(|V| \log |V|)$, el cual es mejor que $O(|V|^2)$.

El algoritmo de Dijkstra asume que todos los pesos de las aristas son números positivos [Cormen et al., 2013], [Weiss, 2014] y mantiene un conjunto S de vértices, donde la ruta óptima y su longitud total desde el nodo origen s hasta

el vértice $v \in S$ ya han sido determinados. Para todos los vértices $v \in S$, se tiene entonces que $d[v] = \delta(s, v)$, donde $\delta(s, v) = \min\{d(p) \mid s \xrightarrow{p} v\}$ si existe una ruta p de s a v , con una longitud total $d(p)$, de otro modo $\delta(s, v) = \infty$.

Para resolver el problema de las rutas óptimas entre cada par de vértices en la red, se utiliza el algoritmo Floyd-Warshall [Weiss, 2014] basado en la Programación Dinámica y resuelve dicho problema en un tiempo de ejecución $O(|V|^3)$. Sin embargo, asumiendo que $|E| = O(|V|)$, otra manera más eficiente consiste en ejecutar el algoritmo de Dijkstra para cada uno de los nodos del grafo; es decir, $|V|$ veces produciendo las rutas óptimas para cada par de vértices en un tiempo $O(|V||E| \log |V|) = O(|V|^2 \log |V|)$, lo cual es mejor que el tiempo $O(|V|^3)$ que consume el algoritmo basado en programación dinámica.

El Algoritmo 1 consiste en el procedimiento $DIJKSTRA(V, E, w, v_0)$ que genera las etiquetas $(L(v), v_a)$ en cada uno de los nodos del grafo, donde $L(v)$ es la distancia mínima que se recorre para llegar al vértice v y v_a corresponde al vértice anterior en la ruta para alcanzar el nodo v . El algoritmo requiere de una inicialización de variables: n con la cantidad de nodos $|V|$ del grafo, un índice i con el valor cero, un conjunto S_i que contiene el nodo fuente v_0 , la etiqueta del nodo v_0 es $(L(v_0), v_a)$ con el valor $(0, -)$ y el resto de los nodos v del grafo, la etiqueta $(L(v), v_a)$ con el valor $(\infty, -)$ (Líneas 1-7 del Algoritmo 1).

Algoritmo 1 Algoritmo de Dijkstra.

procedure DIJKSTRA(V, E, w, v_0)

```
1:  $n := |V|$ 
2:  $i := 0$ 
3:  $S_i := \{v_0\}$ 
4:  $(L(v_0), v_a) := (0, -)$  ▷ Etiqueta de  $v_0$ 
5: for all  $v \in \{V - v_0\}$  do
6:    $(L(v), v_a) := (\infty, -)$  ▷ Etiqueta de  $\forall v \in \{V - v_0\}$ 
7: end for
8: if  $n = 1$  then
9:   return  $V := \{v_0\}$  ▷ El problema base está solucionado
10: else
11:    $u := v_0$ 
12:   while  $i < n - 1$  do
13:      $minimo := \infty$ 
14:      $\overline{S}_i := V - S_i$ 
15:     for all  $v \in \overline{S}_i$  do
16:       if  $L(u) + w(u, v) < L(v)$  then
17:          $(L(v), v_a) := (L(u) + w(u, v), u)$ 
18:       end if
19:       if  $L(v) < minimo$  then
20:          $minimo := L(v)$ 
21:          $nodo := v$ 
22:       end if
23:     end for
24:      $u := nodo$ 
25:      $S_{i+1} := S_i \cup \{nodo\}$ 
26:      $i := i + 1$ 
27:   end while
28: end if
29: return Etiquetas  $(L(v), v_a)$  del grafo
```

Si el grafo tiene solamente un nodo $n = 1$ el algoritmo regresa como solución el nodo inicial v_0 (Líneas 8-9 del Algoritmo 1). De otra manera, se asigna a u el nodo fuente v_0 (Líneas 10-11 del Algoritmo 1). Es necesario procesar los $n - 1$ nodos restantes del grafo para obtener las etiquetas $(L(v), v_a)$ de cada uno a través de una estructura de ciclo While (Líneas 12-27 del Algoritmo 1). Se inicializa la variable *minimo* con un valor grande y se determinan los nodos

que no han sido procesados mediante la actualización del conjunto $\overline{S}_i := V - S_i$ re-etiquetando los nodos $\forall v \in \overline{S}_i$ que tengan una ruta con un peso menor con $(L(v), v_a)$ y se identifica cuál es el nodo más cercano a v asignando la información de la distancia a la variable *minimo* y el vértice alcanzado a la variable *nodo* (Líneas 13-23 del Algoritmo 1). Posteriormente el vértice alcanzado *nodo* se asigna a u y se agrega al conjunto S_{i+1} , y se incrementa el índice i (Líneas 24-26 del Algoritmo 1). Una vez finalizado el ciclo el procedimiento regresa las etiquetas de todos los nodos (Línea 29 del Algoritmo 1).

2.3.2. Algoritmos de Prim y Kruskal.

El Algoritmo de Prim tiene una complejidad computacional del orden de $O(E+V \log V)$ y el Algoritmo de Kruskal tiene complejidad del orden de $O(E \log V)$ ambos generan un árbol extendido de costo mínimo a partir de un grafo conexo ponderado [Levitin, 2012] [Cormen et al., 2013].

El Algoritmo 2 corresponde al procedimiento $PRIM(V, E, w, u)$ para obtener el árbol extendido de costo mínimo T de un grafo ponderado conexo $G(V, E, w)$ a partir del nodo inicial u . Se inicializan las variables n con la cantidad de nodos $|V|$, el conjunto de aristas del árbol $|T|$ con vacío y el conjunto de nodos del árbol TV con el nodo inicial u (Líneas 1-3 del Algoritmo 2). Mientras el conjunto de aristas $E \neq \emptyset$ y $|T| \neq n - 1$ el algoritmo entrará en un ciclo (Líneas 4-14 del Algoritmo 2) para buscar la arista e de menor costo para $u \in TV \wedge v \notin TV$ (Línea 5 del Algoritmo 2). En caso de que el conjunto de aristas e este vacío \emptyset el algoritmo terminará el ciclo (Líneas 6-8 del Algoritmo 2). Si se encuentra una arista y no se forme un ciclo se procede a eliminar la arista del conjunto E y agregarla al conjunto T , mientras que el nodo v se agrega al conjunto TV del árbol extendido

(Líneas 9-13 del Algoritmo 2). El algoritmo regresará el conjunto de aristas T del árbol cuando tengan $n - 1$ aristas, en otro caso no es posible construirlo (Líneas 15-19 del Algoritmo 2).

Algoritmo 2 Algoritmo de Prim.

procedure $PRIM(V, E, w, u)$

```

1:  $n := |V|$ 
2:  $T := NULL$ 
3:  $TV := \{u\}$ 
4: while  $E \neq \emptyset \wedge |T| \neq n - 1$  do
5:    $e := \{(u, v)\}$  arista de mínimo costo tal que  $u \in TV \wedge v \notin TV$ 
6:   if  $e = \emptyset$  then
7:     BREAK
8:   end if
9:   if  $T \cup e$  no crea un ciclo then
10:     $E := E - e$ 
11:     $T := T \cup e$ 
12:     $TV := TV \cup \{v\}$ 
13:   end if
14: end while
15: if  $|T| = n - 1$  then
16:   return  $T$  es el Árbol Extendido de Costo Mínimo
17: else
18:   return Grafo no conexo, no tiene un Árbol Extendido de Costo Mínimo
19: end if

```

El procedimiento $KRUSKAL(V, E, w)$ corresponde al Algoritmo 3 para obtener el árbol extendido de costo mínimo T de un grafo ponderado conexo $G(V, E, w)$. Se requiere inicializar las siguientes variables: n con la cantidad de nodos $|V|$ del grafo y el conjunto de aristas del árbol $|T|$ con nulo o vacío (Líneas 1-2 del Algoritmo 3). Mientras el conjunto de aristas del árbol $|T| < n - 1$ el algoritmo entrará en un ciclo (Líneas 3-9 del Algoritmo 3) buscando la arista e de menor costo y eliminándola del conjunto de aristas del grafo E (Líneas 4-5 del Algoritmo 3). Si no se forme un ciclo se incorporará la arista al conjunto de aristas T del árbol extendido (Líneas 6-8 del Algoritmo 3). El algoritmo regresará el conjunto de aristas T del árbol extendido de costo mínimo (Línea 10 del Algoritmo

3).

Algoritmo 3 Algoritmo de Kruskal.

procedure *KRUSKAL*(V, E, w)

```
1:  $n := |V|$ 
2:  $T := NULL$ 
3: while  $|T| < n - 1$  do
4:    $e := \{(u, v)\}$  arista de mínimo costo
5:    $E := E - e$ 
6:   if  $T \cup e$  no crea un ciclo then
7:      $T := T \cup e$ 
8:   end if
9: end while
10: return  $T$  es el Árbol Extendido de Costo Mínimo
```

Con estos antecedentes podemos modelar redes de transporte a través de un grafo conexo dirigido, en donde los vértices representan las intersecciones, las aristas representan segmentos de camino entre intersecciones, y los pesos de las aristas representan distancias recorridas entre dos intersecciones. En la Figura 2.6 se muestra el grafo dirigido correspondiente a la Red de Tráfico Vehicular de la ciudad de Querétaro delimitado por las avenidas Universidad, 5 de febrero, Corregidora y Constituyentes, en donde se muestran los sentidos de las calles y avenidas dentro del límite propuesto. El grafo tiene un conjunto de 411 vértices y un conjunto de 736 aristas [Gonzalez-Gutierrez and Gonzalez, 2010] [Huerta J., 2010].

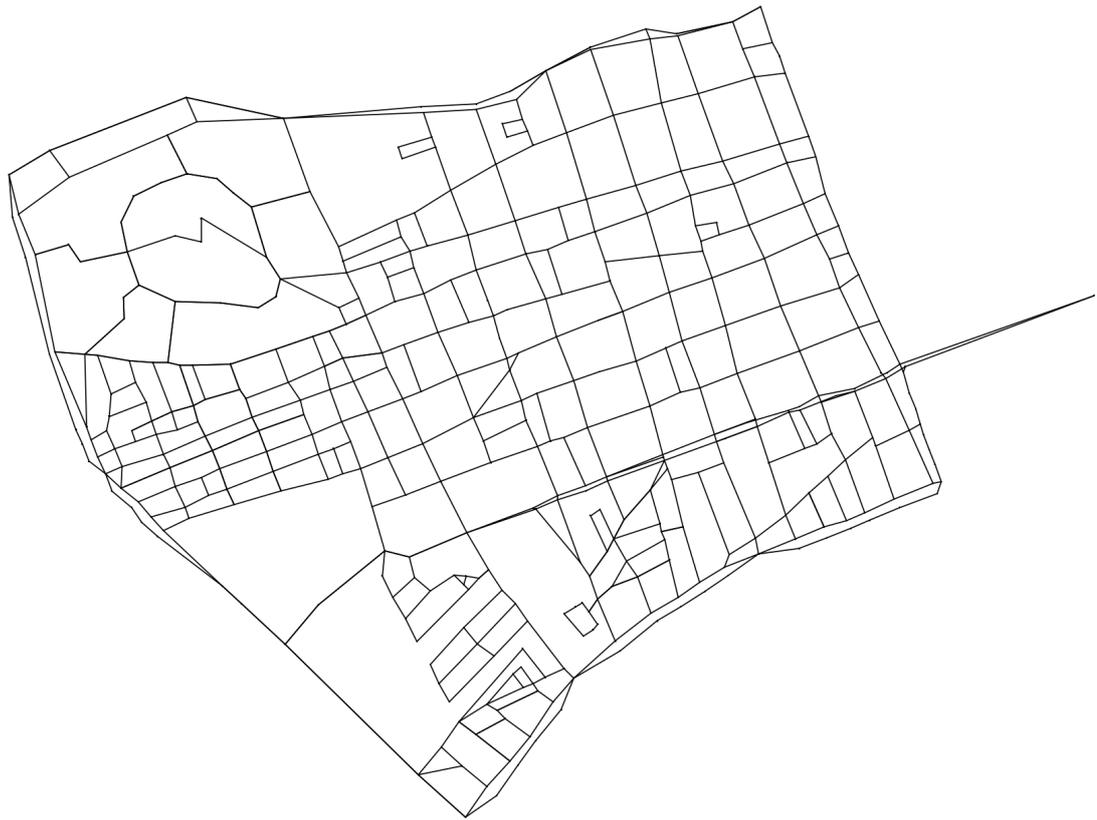


Figura 2.6: Grafo de tránsito vehicular de la ciudad de Querétaro (Universidad, 5 de febrero, Corregidora y Constituyentes).

El modelado y simulación del grafo de la Red de Tránsito Vehicular de la ciudad de Querétaro se realizó en el lenguaje de programación funcional *Mathematica*® a través del cual se obtuvieron datos importantes de la red, en la Figura 2.7 se puede apreciar la ruta corta que dibuja la aplicación en los dos sentidos (rojo y azul) para ir del Centro Universitario UAQ a la Alameda.

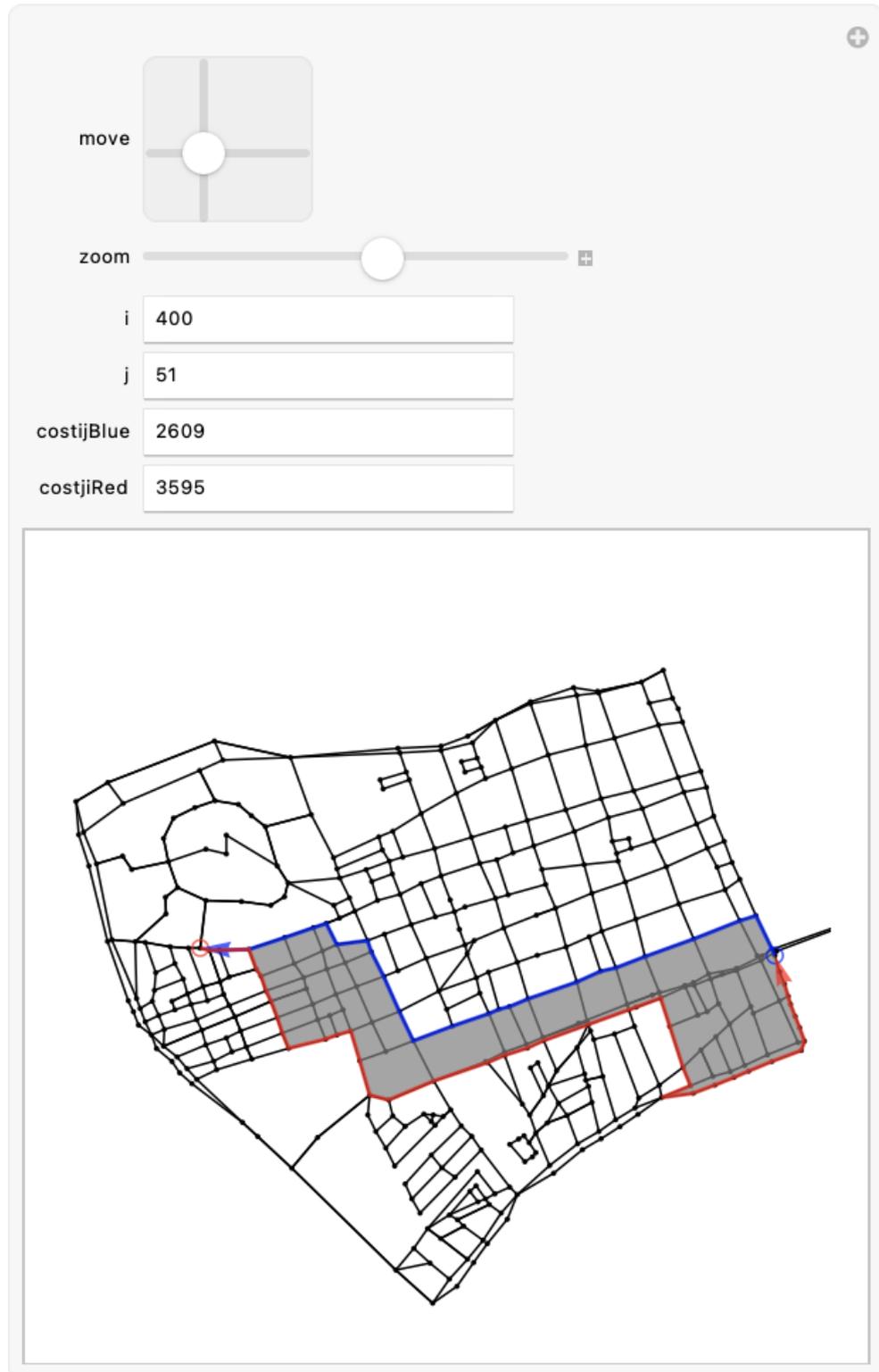


Figura 2.7: Peso de la Ruta Corta: Centro Universitario UAQ y Alameda.

2.4. Complejidades Algorítmicas.

La eficiencia de un algoritmo se mide por el tiempo de ejecución y la cantidad de memoria que requiere una computadora para el procesamiento y almacenamiento de información de una instancia. Existen problemas que no tienen una solución de algorítmica eficiente. Un algoritmo es eficiente si existe un polinomio $p(n)$ de tal manera que el algoritmo pueda resolver cualquier instancia de tamaño n en un tiempo $O(p(n))$; esto es, un algoritmo de tiempo polinomial.

Definition 2.1 (Complejidad Límite Superior). *Dado un problema P decimos que el tiempo de complejidad del límite superior de P es $O(g(n))$ si existe un algoritmo para P cuyo tiempo de ejecución es $O(g(n))$.*

Definition 2.2 (Complejidad Límite Inferior). *Dado un problema P decimos que el tiempo de complejidad del límite inferior de P es $\Omega(g(n))$ si existe un algoritmo para P que tiene un tiempo de ejecución $\Omega(g(n))$.*

Definition 2.3 (Límites de Complejidad). *Dado un problema P decimos que el tiempo de complejidad de P es $\Theta(g(n))$ si su límite superior es $O(g(n))$ y el límite inferior es $\Omega(g(n))$.*

El orden de crecimiento del tiempo de ejecución de un algoritmo nos da una característica de la eficiencia del mismo, y permite comparar el rendimiento relativo de algoritmos alternos. Para ello, se utilizan las notaciones Ω , O y Θ para medir la complejidad de los algoritmos. Por lo que, un algoritmo que requiere $f(n)$ número de pasos para resolver una instancia \mathcal{X} de un problema P , se dice tener una complejidad Ω , O y Θ , si cumple con las siguientes condiciones:

$$f(n) = \Omega(g(n)) \Leftrightarrow \exists c, n_0 \ni \forall n f(n) \geq cg(n), n \geq n_0 \quad (2.3)$$

$$f(n) = O(g(n)) \Leftrightarrow \exists c, n_0 \ni \forall n f(n) \leq cg(n), n \geq n_0 \quad (2.4)$$

$$f(n) = \Theta(g(n)) \Leftrightarrow f(n) = \Omega(g(n)) \wedge f(n) = O(g(n)) \quad (2.5)$$

En general, se puede expresar la solución de un problema en términos de la relación $P \subseteq I \times S$ donde I es el conjunto de instancias del problema P y S el conjunto de soluciones de P . Podemos considerar un predicado $p(x, y)$ el cual es verdadero si y solo si $(x, y) \in P$. En algún caso se requiere verificar que para una instancia $x \in I$ satisface una condición dada a través de un predicado unitario específico $\pi(x)$; por ejemplo, si queremos verificar si un programa es sintácticamente correcto. En este caso, la relación P se reduce a la siguiente función $f : I \rightarrow S$, donde S es el conjunto binario $S = YES, NO$ (o $S = 0, 1$) y el problema se transforma en un *problema de decisión* (o *reconocimiento*) [Ausiello et al., 2003].

Se puede considerar también el caso de un *problema de búsqueda* donde para alguna instancia $x \in I$, una solución $y \in S$ tiene que ser devuelta después de ser verificado $(x, y) \in P$. Este caso incluye problemas como por ejemplo el encontrar una ruta en un grafo entre dos nodos. Expresado de otra manera, dada una instancia $x \in I$ se está interesado en encontrar la “*mejor*” solución y^* (de acuerdo a una métrica) entre todas las soluciones $y \in S$ tal que $(x, y) \in P$ es verificado. Este tipo de problemas son llamados problemas de optimización y frecuentemente surgen en diversas aplicaciones importantes.

2.5. El problema del agente viajero.

Suponga usted que el encargado de una gran compañía que se dedica a la distribución de productos en la ciudad cuenta con una cartera de clientes a lo largo y ancho de la ciudad, ha ubicado su almacén estratégicamente. La pregunta que el encargado de planeación del itinerario de sus embarques se hace es: ¿cuál es el orden de distribución de los productos de modo que se minimicemos tiempo, distancia, o ambos, los cuales por supuesto impactan finalmente en los costos de distribución? Ahora se introduce un problema clásico de las ciencias computacionales que ha recibido una gran atención de los investigadores por muchos años y se conoce como el problema del Agente Viajero [Gutin and Punnen, 2007] [Cook, 2014]. En otras palabras, se trata de un agente viajero que quiere visitar a todos sus clientes, cada uno exactamente una sola vez. Probablemente visitar a un cliente y en otro momento regresar al mismo sitio donde lo visitó no sea bien visto por la empresa ya que podría significar que el agente es ineficiente, y su acción redunde en gastos infructuosos. Así que el agente debe visitar cada cliente exactamente una vez, recorriendo la mínima distancia posible. En palabras más técnicas, queremos la ruta cuya longitud sea la óptima, es decir la mínima.

Técnicamente, el problema del agente viajero consiste en un problema difícil o intratable computacionalmente hablando. Esto significa que se conjetura que no existe un algoritmo, y finalmente un programa de computadora, que resuelva cualquier instancia del problema del agente viajero en un tiempo de cómputo razonable, que siempre produzca soluciones óptimas.

Los problemas que no cuentan con un algoritmo de tiempo polinomial, se clasifican como problemas NP-Completos. Los algoritmos que usan técnicas

greedy son utilizados para construir soluciones algorítmicas de problemas NP-Completos.

El Problema del Agente Viajero (Traveling Salesperson Problem, TSP) es modelado como un grafo completo conexo no-dirigido y el objetivo es que el agente realice un *tour* o ciclo hamiltoniano, visitando cada ciudad exactamente una vez y terminando en la ciudad donde inicia su recorrido. El costo de viajar de la ciudad i a la ciudad j es $c(i, j) \in \mathbb{Z}^+$, y el agente desea realizar un *tour* cuyo costo total sea mínimo, el cual consiste en la suma de costos de todas las aristas que conforman el *tour* [Garey and Johnson, 2002] [Cormen et al., 2013] [Gonzalez-Gutierrez and Gonzalez, 2010].

La Figura 2.8 muestra la solución al Problema del Agente Viajero, solucionado por Applegate *et al* en 1998, encontrando un *tour* óptimo de las 13,509 ciudades en Estados Unidos con una población mayor a 500 habitantes [Cook, 2014] [Applegate et al., 2011] .

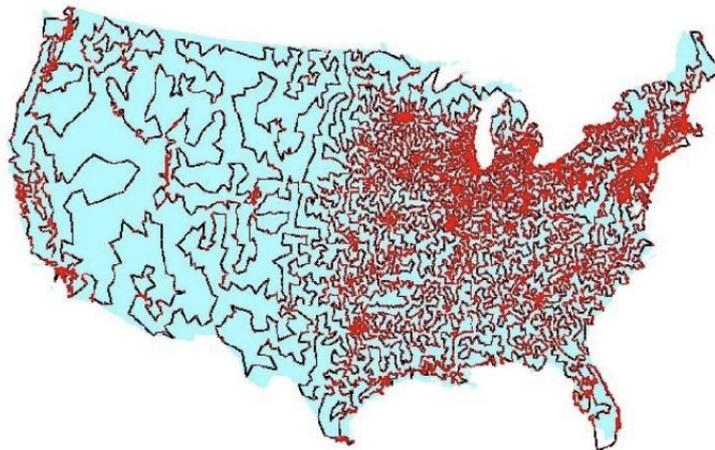


Figura 2.8: Tour para 13,509 ciudades de Estados Unidos de América.

2.6. El problema del “profesor perfecto”.

Considere ahora Usted un problema interesante al que se le ha referido como el problema del profesor Perfecto, quien pretende realizar todas sus tareas de manera óptima. Así que un día, la esposa del profesor Perfecto le llama y le dice: *“Amor, quiero que a mediodía que vienes de la Universidad a casa a comer, pases por favor a alguna de las tiendas OSSO, para comprar una bolsa de hielo, después no olvides pasar a cargar gasolina al auto”*. Y sigue la lista: *“pasas por un cajero automático a retirar efectivo, y finalmente no olvides el ramo de flores que tanto me has prometido”*. Como el profesor Perfecto no le gusta malgastar su tiempo (y gasolina), procede a construir mediante la ayuda de un programa de computadora su recorrido desde la Universidad pasando a una tienda OSSO, una gasolinera, un cajero automático y una tienda donde vendan flores. Como se sabe, hay una gran cantidad de tiendas OSSO, así como de gasolineras en la ciudad, decenas de cajeros automáticos, y por supuesto muchas tiendas donde pueda adquirir un ramo de flores. La pregunta que plantea el profesor Perfecto es ¿cuál es la ruta, desde la Universidad a mi casa, de modo que atienda todos los encargos, con una longitud total óptima, es decir, mínima, y cuál es el orden en que debo atender cada encargo? Este problema consiste en un problema más general que aquel del Agente Viajero mencionado anteriormente.

Así que, si el problema del Agente Viajero es difícil computacionalmente hablando, también lo es el problema que plantea el Profesor Perfecto, ya que éste consiste en una generalización del problema del Agente Viajero [Cook, 2014] [Gutin and Punnen, 2007] [Paschos, 2014].

**”Los grandes algoritmos son como la poesía de la
computación”**

*John G. F. Francis Sullivan (1934-)
Científico de la computación inglés*

Algoritmos de enumeración y generación de embaldosados con dominós.

3.1. Modelado de embaldosados.

Dado un conjunto finito de baldosas, la tarea de embaldosar un plano es, sorprendentemente, un problema indecidible [Berger, 1966]. El problema correspondiente para regiones finitas resulta también ser un problema NP-completo [Read, 1980]. Incluso limitando el estudio a embaldosados rectangulares con dominós, su dificultad computacional sigue siendo NP-completo. El objetivo es mostrar una breve descripción acerca de métodos para resolver el problema de enumeración y generación de embaldosados, pero principalmente presentar algoritmos basados en la técnica recursiva *backtracking* para la generación computacional de embaldosados. Para ello, primeramente se definen algunos términos.

Sea un rectángulo, de altura n y anchura m , con n y m enteros positivos, un rectángulo $n \times m$. El embaldosado de un rectángulo es una partición del rectángulo en dominós (rectángulos 1×2) no superpuestos. Se presentan algoritmos para generar varios casos de embaldosados con rectángulos de área a lo

más 49; por ejemplo, la Figura 3.1 muestra los 11 embaldosados de un rectángulo 4×3 .

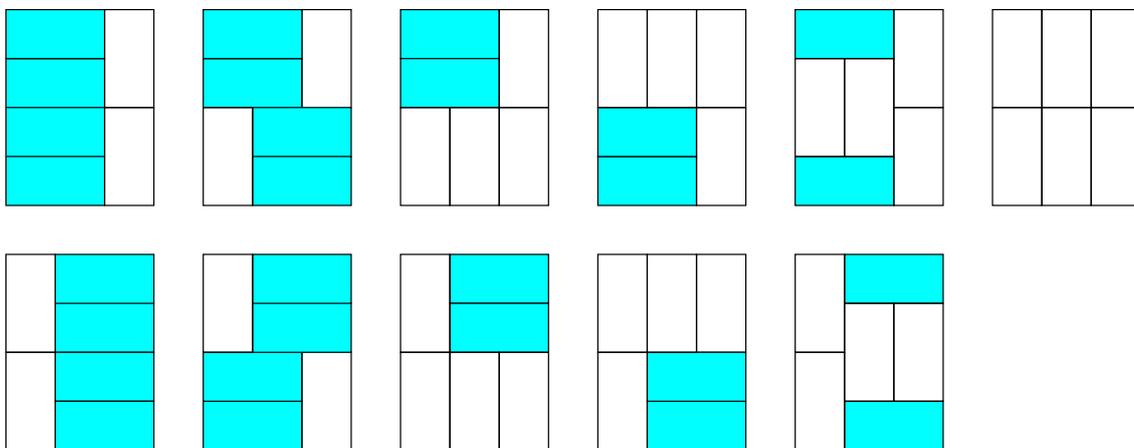


Figura 3.1: Los 11 embaldosados del rectángulo 4×3 .

Una cota superior para el número $t(n, m)$ de embaldosados de un rectángulo $n \times m$ se puede obtener considerando todas las permutaciones

$$\frac{(2u)!}{2^u \times u!} \quad (3.1)$$

del multiconjunto $\{1, 1, 2, 2, \dots, \frac{nm}{2}, \frac{nm}{2}\}$, con $u = \frac{nm}{2}$, teniendo en cuenta las repeticiones debido al re-etiquetado de los índices. Se puede lograr una mejora adicional a dicha cota considerando solo aquellas configuraciones en las que dos números iguales se encuentran a una distancia Manhattan de 1.

El primer resultado significativo sobre enumeración de embaldosados aparece en el contexto del modelado de dímeros con grafos (un dímero es un polímero con dos átomos [Kenyon and Okounkov, 2005], [Klarner and Pollack, 1980]) propuesto en 1961 por Kasteleyn [Kasteleyn, 1961], [Percus, 1971], quien en-

contró para $t(n, m)$ la siguiente ecuación, caracterizada por la función coseno:

$$t^2(n, m) = \prod_{i=1}^n \prod_{j=1}^m 2 \left| \cos\left(\frac{i\pi}{n+1}\right) + \sqrt{-1} \cos\left(\frac{j\pi}{m+1}\right) \right| \quad (3.2)$$

En general, una cobertura de dímerno (o emparejamiento perfecto) de un grafo G es una colección de aristas que cubren todos los vértices exactamente una vez, es decir cada vértice es el punto final de una sola arista de la colección. Así el problema de enumeración de embaldosados es equivalente al problema de encontrar un emparejamiento perfecto en el grafo, que a su vez puede ser conectado al problema de encontrar el permanente de una matriz.

La Tabla 3.1 muestra los valores correspondientes a $t(n, m)$ para un rectángulo de área a lo más de 100. Los valores de 0 que aparecen en la Tabla 3.1 se deben al hecho de que es imposible embaldosar rectángulos de área impar; este hecho está determinado en (3.2), ya que, cuando $n \times m$ es impar, los valores de $i = \frac{n+1}{2}$ y $j = \frac{m+1}{2}$ producen un factor cero, el cual reduce el producto total a 0. Todavía más interesante resulta la aparición de los números de Fibonacci en el segundo renglón (y segunda columna), lo que sugiere el hecho de que el número de Fibonacci F_{m+1} (F_{n+1}) es igual al número de embaldosados de un rectángulo $2 \times m$ ($n \times 2$, respectivamente).

Tabla 3.1: Valores de $t(n, m)$ para $n, m \leq 10$.

n	m									
	1	2	3	4	5	6	7	8	9	10
1	0	1	0	1	0	1	0	1	0	1
2	1	2	3	5	8	13	21	34	55	89
3	0	3	0	11	0	41	0	153	0	571
4	1	5	11	36	95	281	781	2245	6336	18061
5	0	8	0	95	0	1183	0	14824	0	185921
6	1	13	41	281	1183	6728	31529	167089	817991	4213133
7	0	21	0	781	0	31529	0	1292697	0	53175517
8	1	34	153	2245	14824	167089	1292697	12988816	108435745	1031151241
9	0	55	0	6336	0	817991	0	108435745	0	14479521761
10	1	89	571	18061	185921	4213133	53175517	1031151241	14479521761	258584046368

Los valores que resultan de (3.2) provienen, en la mayoría de los casos, de la multiplicación de números que no son en general números enteros o incluso racionales. Al final, en la multiplicación se tienen resultados enteros de una manera no trivial, lo que conduce a resultados difíciles de demostrar. Por ejemplo, al usar (3.2) se pueden demostrar los siguientes corolarios:

$$\prod_{i=1}^n [2\text{Cos}(\frac{i\pi}{n+1})] \in \{-1, 0, 1\} \quad (3.3)$$

$$64\text{Cos}^2(\frac{\pi}{9}) \times \text{Cos}^2(\frac{2\pi}{9}) \times \text{Sen}^2(\frac{\pi}{18}) = 1 \quad (3.4)$$

$$\prod_{i=1}^m [3 + 2\text{Cos}(\frac{2\pi i}{m+1})] = F_{m+1}^2 \quad (3.5)$$

Para cualquier n constante, el número $t(n, m)$ satisface una ecuación diferencial lineal (homogénea con coeficientes constantes), o lo que es lo mismo, la función generadora $\sum_{m=0}^{\infty} t(n, m)x^m$ representa una función racional.

Los kernels correspondientes a los valores de m de 1 a 6 son

$$(0, 1), (1, 1), (0, 4, 0, -1), (1, 5, 1, -1), (0, 15, 0, -32, 0, 15, 0, -1), \text{ y} \\ (1, 20, 10, -38, -10, 20, -1, -1),$$

respectivamente. Estos pueden ser usados para obtener expresiones explícitas para $t(n, m)$; por ejemplo, las siguientes:

$$t(n, 1) = \frac{1}{2}(1 + (-1)^n) \quad (3.6)$$

$$t(n, 2) = \frac{1}{2}(Fibonacci(n) + Lucas(n)) \quad (3.7)$$

$$t(n, 3) = \frac{((2 - \sqrt{3})^{\frac{n+1}{2}} + (2 + \sqrt{3})^{\frac{n+1}{2}})(1 + e^{n\pi\sqrt{-1}})}{2\sqrt{6}} \quad (3.8)$$

$$t(n, 4) = \frac{-r_1^{n+1} - r_2^{n+1} + s_1^{n+1} + s_2^{n+1}}{4^{n+1}\sqrt{29}} \quad (3.9)$$

La última expresión para $t(n, 4)$ se obtiene resolviendo la ecuación de recurrencia $t(n) = t(n-1) + 5t(n-2) + t(n-3) - t(n-4)$, con las condiciones de frontera $t(1) = 1, t(2) = 5, t(3) = 11$ y $t(4) = 36$; donde $r_{1,2} = 1 - \sqrt{29} \pm \sqrt{2(7 - \sqrt{29})}$ y $s_{1,2} = 1 + \sqrt{29} \pm \sqrt{2(7 + \sqrt{29})}$. Alternativamente, $t(n, 4)$ se puede expresar como el siguiente producto de matrices:

$$t(n, 4) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{n-1} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \quad (3.10)$$

Aunque de ninguna manera obvia de acuerdo con Klarner *et al* y Lovász, $t(2n, 2n)$ es siempre un cuadrado perfecto o dos veces un cuadrado perfecto [Klarner and Pollack, 1980], [Lovász, 1979]. Además, n es la potencia más alta de 2 dividiendo $t(2n, 2n)$. Pachter usó un método diferente que Kasteleyn para llegar al mismo resultado casi al mismo tiempo [Temperley and Fisher, 1961],

[Pachter, 1997], [Kasteleyn, 1967]. Ambas investigaciones demuestran que $t(2n, 2n)$ converge a C^{4n^2} , donde $C = e^{G/\pi} \approx 1.3385$ (G es la constante de Catalan 0.915965).

3.2. Metodología.

Para obtener la cantidad $t(n, m)$ de embañosados, se puede utilizar el permanente de la *matriz de Kasteleyn* asociada al rectángulo $n \times m$ que se desea embañosar con dominós [Lieb, 1967], [Strehl, 2001], [Klarner and Pollack, 1980]. Primero, se caracteriza el rectángulo $n \times m$, dividiéndolo en cuadrados negros y blancos, de tal manera que no existan dos cuadrados adyacentes con el mismo color en el mismo renglón y en la misma columna. Se procede ahora a etiquetar los cuadrados por renglón de izquierda a derecha y de abajo hacia arriba con un número entero positivo k en orden creciente, usando dos veces dicho número como etiqueta, de modo que para cada etiqueta k existen dos cuadrados de diferente color. Por ejemplo, la Figura 3.2 muestra la caracterización del rectángulo 2×4 , así como la indexación de los cuadrados conforme al procedimiento de etiquetado establecido.

3	3	4	4
1	1	2	2

Figura 3.2: Indexación del rectángulo 2×4 .

La matriz de Kasteleyn K , correspondiente al rectángulo partido en los cuadrados negros y blancos etiquetados, se define como una matriz cuadrada binaria con sus renglones indexados por el número asociado a los cuadrados negros y sus columnas indexadas con el número asociado a los cuadrados blancos. Es decir, la entrada $K_{i,j}$ refiere al cuadrado negro i y cuadrado blanco j del rectángulo partido en cuadrados negros y blancos. La entrada $K_{i,j} = 0$ si el cuadrado negro i no es vecino del cuadrado blanco j , de otro modo, $K_{i,j} = 1$. Por ejemplo, la matriz de Kasteleyn K asociada al rectángulo 2×4 es la siguiente:

$$K = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \quad (3.11)$$

Finalmente, $t(2,4) = \text{Permanente}(K) = 5$.

Ya que el cálculo del determinante de una matriz de tamaño $n \times m$ demanda menor tiempo de cómputo que el cálculo de su permanente, resulta conveniente transformar el problema a otro, donde $t(n,m)$ se pueda calcular mediante el uso del determinante de una matriz. Así, se puede cambiar el numeral 1 a $\sqrt{-1}$ ó mantener ese mismo valor de 1 de acuerdo con la relación que exista entre el cuadrado negro i y el cuadrado blanco j . Si el cuadrado negro i se relaciona horizontalmente con el cuadrado blanco j , entonces la entrada $K_{i,j}$ que originalmente vale 1 se cambia por el valor $\sqrt{-1}$. De otro modo, si el cuadrado negro i se relaciona verticalmente con el cuadrado blanco j , entonces se mantiene el valor de 1 de la entrada $K_{i,j}$. Bajo esta consideración la matriz previa K en nuestro

ejemplo cambia a

$$K^{mod} = \begin{bmatrix} \sqrt{-1} & 0 & 1 & 0 \\ \sqrt{-1} & \sqrt{-1} & 0 & 1 \\ 1 & 0 & \sqrt{-1} & \sqrt{-1} \\ 0 & 1 & 0 & \sqrt{-1} \end{bmatrix} \quad (3.12)$$

Entonces el determinante de K^{mod} es igual al número $t(2, 4)$ de embaldosados [Montroll, 1964], [Propp, 2001], [Read, 1980]. Kasteleyn arribó a (2.1) usando el producto de los eigenvalores de K^{mod} .

Multiplicando cada valor de la entrada $K_{i,j}^{mod}$ de la matriz K^{mod} por la variable $d_{i,j}$, la cual representan al dominó desplegado a lo largo del cuadro negro i y el cuadro blanco j , el determinante de la matriz resultante hace una descripción explícita de los cinco embaldosados enumerados por $t(2, 4)$:

$$Det \begin{bmatrix} \sqrt{-1}d_{11} & 0 & d_{13} & 0 \\ \sqrt{-1}d_{21} & \sqrt{-1}d_{22} & 0 & d_{24} \\ d_{31} & 0 & \sqrt{-1}d_{33} & \sqrt{-1}d_{34} \\ 0 & d_{42} & 0 & \sqrt{-1}d_{44} \end{bmatrix} = d_{13}d_{24}d_{31}d_{42} +$$

$$d_{11}d_{24}d_{33}d_{42} + d_{13}d_{21}d_{34}d_{42} + d_{13}d_{22}d_{31}d_{44} + d_{11}d_{22}d_{33}d_{44} \quad (3.13)$$

3.2.1. Generación de embaldosados.

La generación de todos los embaldosados se puede también lograr utilizando el método de *backtracking*, en el que un embaldosado parcial se extiende exhaustivamente hasta obtener una solución parcial válida, o una solución (no válida) que no se puede extender. En este último punto el algoritmo retrocede para extender embaldosados parciales válidos obtenidos previamente. Para implementar el método de *backtracking*, los embaldosados se representan mediante una matriz de índices asociados a cada uno de los cuadrados que forman un dominó. Por ejemplo, el cuarto embaldosado de la Figura 3.1 corresponde a la matriz.

$$K = \begin{bmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 4 & 4 & 5 \\ 6 & 6 & 5 \end{bmatrix} \quad (3.14)$$

La visualización de cada embaldosado se puede construir dibujando una línea vertical enseguida de cada índice que sea diferente a su índice adyacente en el mismo renglón, y una línea horizontal por cada índice diferente a su adyacente en la misma columna.

El Algoritmo 4 consiste en el procedimiento $EMBALDOSADOS(n, m)$ que llama al procedimiento recursivo $BACK(patron, d)$ del Algoritmo 5 para generar todos los embaldosados de tamaño $n \times m$, insertando en cada llamada recursiva el dominó de índice $d + 1$. La matriz que representa al embaldosado es llamada *patron* en el Algoritmo 4 (Línea 3). La condición para realizar el *backtracking* está basada en la posibilidad de tener un candidato (celda

vacía) ya sea en la dirección vertical u horizontal (Líneas 7 y 13 del Algoritmo 5, respectivamente). El orden en el que la siguiente posición se selecciona para desplegar un dominó es primeramente por columna y posteriormente por renglón (Líneas 9-10 y 15-16 del Algoritmo 5, respectivamente), considerando el rectángulo como una matriz de tamaño $n \times m$. El Algoritmo 5 usa las funciones $Posicion(patron, 0)$ y $Primera(PosicionesCero)$. El resultado que devuelve la primera función corresponde al conjunto de todas las posiciones vacías de la matriz $patron$, esto es, todas las posiciones que contengan entradas ceros. Dicho conjunto se asigna a la variable $PosicionesCero$. La segunda función retorna la entrada de la matriz correspondiente al primer elemento de $PosicionesCero$. La función $Dimensiones(patron)$ (Linea 1 del Algoritmo 5) regresa la altura n y el ancho m del embañosado $patron$. Las variables $solv$ y $solh$ (Linea 2 del Algoritmo 5) son conjuntos que contendrán la matriz $patron$ correspondiente al embañosado generado de manera vertical u horizontal (Líneas 11 y 17 del Algoritmo 5, respectivamente), y al finalizar, el Algoritmo 5 regresará como solución la unión de los conjuntos $solv$ y $solh$.

Algoritmo 4 Generación de todos los embañosados de tamaño $n \times m$.

procedure $EMBALDOSADOS(n, m)$

```

1: for  $i := 1, n$  do
2:   for  $j := 1, m$  do
3:      $patron[i, j] := 0$ 
4:   end for
5: end for
6:  $d := 0$ 
7: return  $BACK(patron, d)$ 

```

Algoritmo 5 Generación recursiva de embaldosados usando *backtrack*.

```
procedure BACK(patron, d)
1: (n, m) := Dimensiones(patron)
2: solv :=  $\emptyset$ , solh :=  $\emptyset$ 
3: PosicionesCero := Posicion(patron, 0)
4: if PosicionesCero =  $\emptyset$  then return patron
5: else
6:   (i, j) := Primera(PosicionesCero)
7:   if (i + 1 ≤ n) ∧ (patron(i + 1, j) = 0) then
8:     patronAux := patron
9:     patronAux[i, j] := d + 1
10:    patronAux[i + 1, j] := d + 1
11:    solv := solv ∪ BACK(patronAux, d + 1)
12:  end if
13:  if (j + 1 ≤ m) ∧ (patron(i, j + 1) = 0) then
14:    patronAux := patron
15:    patronAux[i, j] := d + 1
16:    patronAux[i, j + 1] := d + 1
17:    solh := solh ∪ BACK(patronAux, d + 1)
18:  end if
19: end if
20: return solv ∪ solh
```

En la siguiente subsección se organiza el conjunto de embaldosados en clases de equivalencias, donde los embaldosados son isomorfos bajo rotaciones y/o reflexiones en la misma clase.

3.2.2. Generando embaldosados no-isomorfos.

El número de embaldosados no-isomorfos se puede calcular mediante el Teorema de Burnside [Grimaldi, 2004], como sigue.

Teorema 3.1. *Teorema de Burnside*

Sea T el conjunto de todos los embaldosados de área $n \times m$ y G el grupo de permutaciones actuando sobre T . El número de clases de equivalencia, en las cuales T es particionado por la acción de G , está dado por $\frac{1}{|G|} \sum_{\pi \in G} \psi(\pi)$, donde

$\psi(\pi)$ es el número de configuraciones en T que permanecen invariantes bajo la acción de la permutación π .

Para obtener solamente los embaldosados no isomorfos, se consideran las siguientes permutaciones o transformaciones sobre un embaldosado: reflexión r_v sobre un eje vertical central; reflexión r_h sobre un eje horizontal central; reflexión r_{D_1}, r_{D_2} sobre ejes diagonales con desplazamiento NO-SE y NE-SO, respectivamente; y rotaciones π_1, π_2, π_3 y π_0 en sentido antihorario de 90, 180, 270 y 360 grados, respectivamente, sobre un eje perpendicular al embaldosado que pasa por el centro del mismo. Finalmente, bajo la acción del grupo G , conformado por las transformaciones $r_v, r_h, r_{D_1}, r_{D_2}, \pi_1, \pi_2, \pi_3$ y π_0 , sobre T , se pueden obtener las clases de equivalencia del conjunto T de todos los embaldosados de tamaño $n \times m$. Cada clase es cerrada bajo la acción del grupo G de movimientos rígidos, de modo que cualquier embaldosado miembro de una clase puede ser considerado como el representante de todos los embaldosados de su clase.

La representación de embaldosados como matrices permite utilizar estas transformaciones, las cuales corresponden simplemente a operaciones sobre esas matrices. Por lo tanto, considerando que un embaldosado es representado por una matriz m : la acción de r_v sobre m , $r_v(m)$, corresponde al *reverso* de cada renglón de m ; $\pi_1(m)$ corresponde a la matriz *transpuesta* de la matriz resultante $r_v(m)$, es decir $\pi_1(m) = \text{transpuesta}(r_v(m))$; $r_h(m) = \pi_1(\text{transpuesta}(m))$; $\pi_2(m) = \pi_1(\pi_1(m))$; $\pi_3(m) = \pi_1(\pi_2(m))$; $\pi_0(m) = m$; $r_{D_1}(m) = \text{transpuesta}(m)$; y $r_{D_2}(m) = \pi_3(r_v(m))$.

Los Algoritmos 6 y 7 producen el conjunto de representantes de cada clase de equivalencia. El procedimiento $NONISO(n, m)$ del Algoritmo 6 produce, para cada embaldosado del conjunto u de embaldosados de tamaño $n \times m$,

generados inicialmente por la función $EMBALDOSADOS(n, m)$ (Línea 2 del Algoritmo 6), todos los embaldosados que son isomorfos al embaldosado t , usando el procedimiento $CLASE(t)$ del Algoritmo 7, y asigna éstos a $embaldosadosIsom$. El conjunto de embaldosados isomorfos de t son eliminados de u (Línea 6 del Algoritmo 6), y el conjunto $embaldosadosNoIsom$ acumula el embaldosado representativo t (Línea 7 del Algoritmo 6).

Algoritmo 6 Encontrar los embaldosados no isomorfos.

procedure $NONISO(n, m)$

```

1:  $embaldosadosNoIsom := \emptyset$ 
2:  $u := EMBALDOSADOS(n, m)$ 
3: while  $u \neq \emptyset$  do
4:    $t := Primera(u)$ 
5:    $embaldosadosIsom := CLASE(t)$ 
6:    $u := u - embaldosadosIsom$ 
7:    $embaldosadosNoIsom := embaldosadosNoIsom \cup \{t\}$ 
8: end while
9: return  $embaldosadosNoIsom$ 

```

Algoritmo 7 Encontrar la clase de embaldosados isomorfos.

procedure $CLASE(t)$

```

1:  $(n, m) := Dimensiones(t)$ 
2:  $sol := \{t, \pi_2(t), r_h(t), r_v(t)\}$ 
3: if  $n = m$  then
4:    $sol := sol \cup \{\pi_1(t), \pi_3(t), r_{D_1}(t), r_{D_2}(t)\}$ 
5: end if
6: return  $sol$ 

```

El procedimiento $CLASE(t)$ del Algoritmo 7 genera todos los embaldosados isomorfos para el embaldosado representativo t de tamaño $n \times m$, aplicando apropiadamente y según corresponda las transformaciones $r_v, r_h, r_{D_1}, r_{D_2}, \pi_1, \pi_2, \pi_3$ y π_0 sobre t . Para un embaldosado cuadrado, es decir, para el cual $m = n$, todas las transformaciones son válidas, mientras que para el caso donde $m \neq n$, solamente las transformaciones r_v, r_h, π_2, π_0 son aplicables, las cuales producen embaldosados de tamaño $n \times m$.

En la siguiente subsección se tratará una versión más restringida del problema de embaldosados, donde no sólo se seleccionan los embaldosados no-isomorfos, sino también aquellos embaldosados para los cuales no existe un conjunto de segmentos de línea a lo largo de las aristas de los dominós que atraviesen completamente al embaldosado de manera horizontal o vertical. Esto es, el embaldosado no se puede cortar horizontal o verticalmente sin cortar algún dominó. Así que, cuando se corta un embaldosado de este tipo ya sea horizontal o verticalmente siempre es el caso que se producen dos partes, cada una de las cuales no son embaldosados con dominós. A cada instancia de esta familia se le llama embaldosado libre de líneas de cortes. Cuando se tratan estas instancias desde el punto de vista de problemas de ruteo y se busca establecer una conexión entre cualquier par de dominós a lo largo de los segmentos de línea definidos por las aristas de los dominós, la longitud total de dicha conexión dada por la suma de las longitudes de cada segmento de línea resulta mayor que si existieran líneas de corte horizontales o verticales.

3.2.3. Generando embaldosados libres de cortes.

Una familia importante de embaldosados es la de aquellos que son libres de líneas de cortes. Una línea de corte en un rectángulo con baldosas tipo dominó es una línea que corta al rectángulo en dos piezas sin cortar ningún dominó. Un embaldosado está libre de líneas de corte si no contiene una línea de corte. Graham estableció el siguiente teorema, el cual generaliza todos los embaldosados libres de líneas de corte [Graham, 1981].

Teorema 3.2. *Teorema de Graham*

Sea R un rectángulo $n \times m$ de área mayor que 2. Entonces R tiene un embaldosado con línea de corte si y solo si $m \times n$ es par, $m \geq 5$, $n \geq 5$ y; n y m no son

ambos iguales a 6.

El procedimiento $LIBREDECORTE(t)$ del Algoritmo 8 decide si una instancia t de embaledado es libre de líneas de corte o no, de acuerdo al Teorema 3.2. Este procedimiento se aplica al conjunto de embaledados generados por el Algoritmo 6. Los procedimientos $CORTEV(t)$ y $CORTEH(t)$ (Algoritmos 9 y 10, respectivamente) cuentan el número de líneas de corte verticales y horizontales, respectivamente.

Algoritmo 8 Verificación de la existencia de líneas de corte.

procedure $LIBREDECORTE(t)$

```
1:  $(n, m) := Dimensiones(t)$ 
2: if  $mod(n \times m, 2) = 0 \wedge CORTEV(t) + CORTEH(t) = 0$  then
3:   return Yes
4: else
5:   return No
6: end if
```

Algoritmo 9 Contar líneas de corte verticales.

procedure $CORTEV(t)$

```
1:  $(n, m) := Dimensiones(t)$ 
2:  $lineasCorte := 0$ 
3: for  $col := 1, (m - 1)$  do
4:    $cont := 0$ 
5:   for  $ren := 1, n$  do
6:     if  $(t[ren, col] - t[ren, col + 1]) \neq 0$  then
7:        $cont := cont + 1$ 
8:     end if
9:   end for
10:  if  $(cont = n)$  then
11:     $lineasCorte := lineasCorte + 1$ 
12:  end if
13: end for
14: return  $lineasCorte$ 
```

Algoritmo 10 Contar líneas de corte horizontales.

procedure *HCUT*(*t*)

```
1: (n, m) := Dimensiones(t)
2: lineasCorte := 0
3: for ren := 1, (n - 1) do
4:   cont := 0
5:   for col := 1, m do
6:     if (t[ren, col] - t[ren + 1, col] ≠ 0) then
7:       cont := cont + 1
8:     end if
9:   end for
10:  if (cont = m) then
11:    lineasCorte := lineasCorte + 1
12:  end if
13: end for
14: return lineasCorte
```

En este capítulo se han presentado, como una contribución importante de este trabajo, los algoritmos para la generación de embaldosados no-isomorfos y libres de corte usados en la generación de un conjunto de familias de embaldosados de tamaño 6×5 , 6×7 , 6×8 , 6×9 , 6×10 y 7×8 (Ver capítulo 5). En el siguiente capítulo se presentan los algoritmos de ruteo que posteriormente se prueban utilizando las familias de instancias generadas.

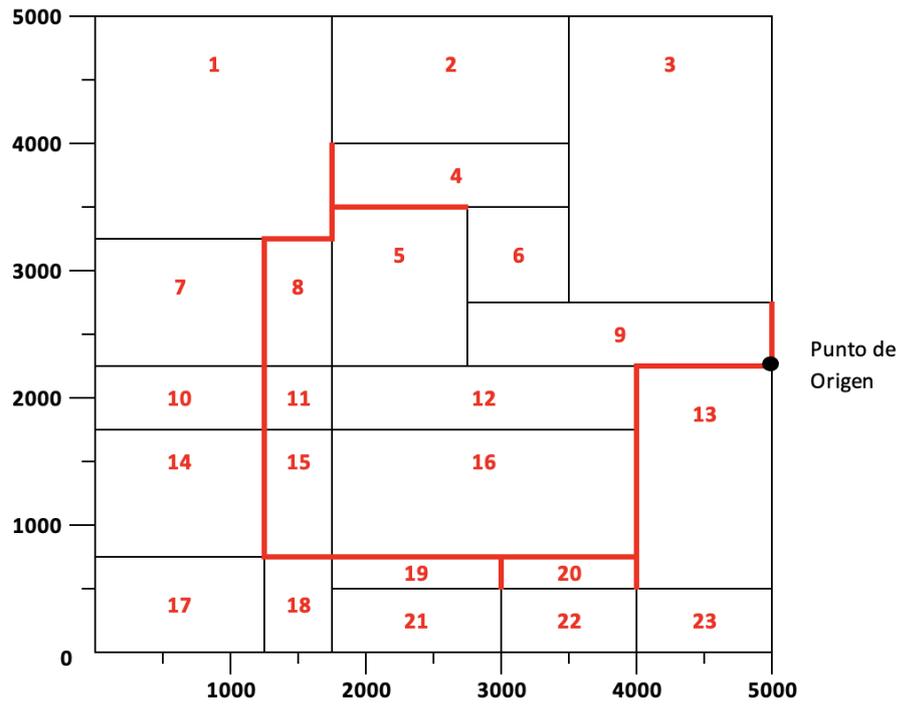
”Un algoritmo debe ser visto para ser creído”

*Donald Ervin Knuth (1938-)
Experto en ciencias de la computación y profesor
emérito estadounidense*

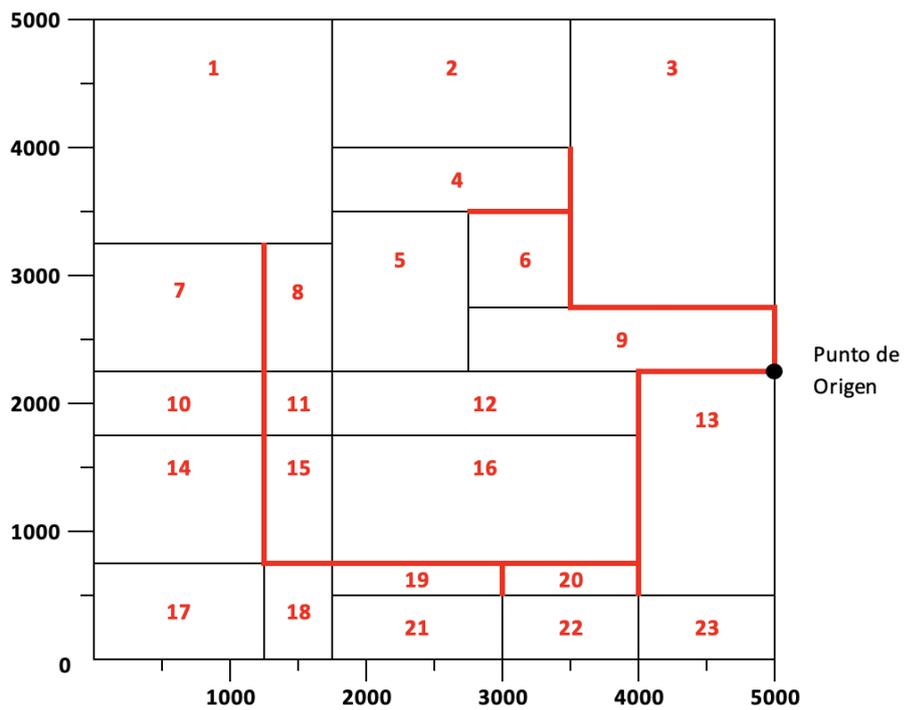
Algoritmos de ruteo

En este capítulo se presentan los algoritmos propuestos para la generación del corredor de longitud mínima y se realiza un análisis de las heurísticas, implementadas en el lenguaje de programación *Mathematica*®, utilizando instancias de *embaldosados no – isomorfos libres de corte* para la obtención de los *Corredores de Longitud Mínima (MLC)*.

El problema del Corredor de Longitud Mínima (MLC - Minimum Length Corridor) es una generalización del Problema del Agente Viajero, el cual consiste en encontrar un conjunto de segmentos de líneas que conectan un vértice localizado sobre la periferia de una instancia geométrica R con al menos un vértice de cada rectángulo rectilíneo de la partición de R , de tal manera que la suma de la longitud de sus segmentos de línea sea la mínima posible. En la Figura 4.1 se muestran dos corredores de una instancia R de tamaño 5000×5000 particionada en 23 rectángulos cuyo punto de acceso se localiza en la periferia vertical derecha de R . La Figura 4.1a el corredor tiene una longitud de 11,000, mientras que el de la Figura 4.1b es de 12,250 [Gonzalez-Gutierrez and Gonzalez, 2007], [Gonzalez, 2014].



(a) Longitud del corredor 11,000.



(b) Longitud del corredor 12,250.

Figura 4.1: Corredores en un instancia R con 23 rectángulos rectilíneos.

4.1. Instancia geométrica

Considere la instancia geométrica R que se presenta en la Figura 4.2, la cual consiste en un cuadrado de tamaño 5000×5000 y que se encuentra dividida en rectángulos de diferentes tamaños.

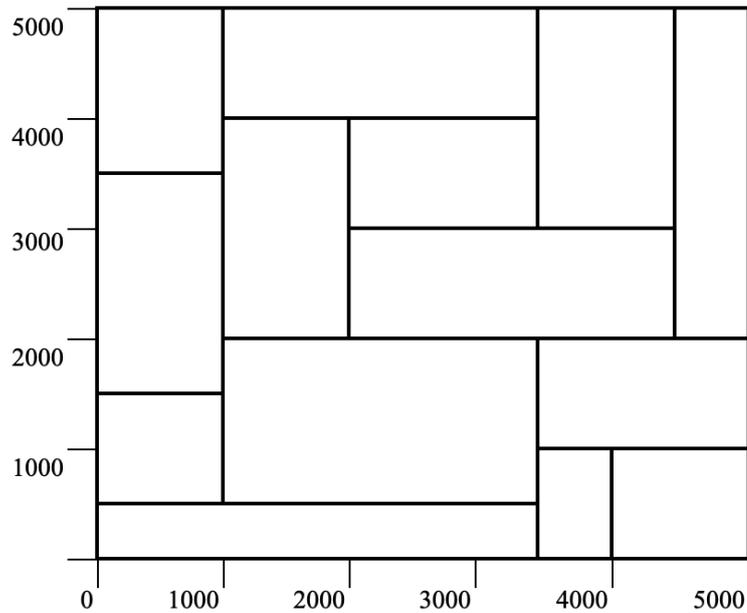


Figura 4.2: Instancia Geométrica R de tamaño 5000×5000 .

4.1.1. Caracterización de la instancia

En la Tabla 4.1 se presentan las características de la instancia geométrica de la Figura 4.2. Cada rectángulo de R se especifica mediante las coordenadas (x, y) de las esquinas superior izquierda e inferior derecha. Finalmente, se indica el número de vértice considerado como el punto de acceso.

Tabla 4.1: Caracterización de instancia de Fig. 4.2.

Cantidad rectángulos		Dimensión Instancia	
14		5000	5000
Coordenadas			
Superior Izquierda		Inferior Derecha	
<i>abscisa (x)</i>	<i>ordenada (y)</i>	<i>abscisa (x)</i>	<i>ordenada (y)</i>
4 000	1 000	5 000	0
0	5 000	1 000	3 500
2 000	4 000	3 500	3 000
1 000	5 000	3 500	4 000
2 000	3 000	4 500	2 000
4 500	5 000	5 000	2 000
0	3 500	1 000	1 500
1 000	4 000	2 000	2 000
3 500	2 000	5 000	1 000
0	500	3 500	0
3 500	1 000	4 000	0
1 000	2 000	3 500	500
0	1 500	1 000	500
3 500	5 000	4 500	3 000
Punto de Acceso			7

Los rectángulos en que se divide la instancia se ordenan de izquierda a derecha y de arriba hacia abajo, tomando en cuenta las coordenadas de la esquina superior izquierda (x, y) de cada rectángulo p_i . Así que, se dice que p_i precede a p_j si se cumple el criterio de ordenamiento expresado a través de la siguiente ecuación.

$$\forall i \forall j ((p_i(y) = p_j(y) \wedge p_i(x) < p_j(x)) \vee (p_i(y) > p_j(y))), 1 \leq i < j \leq m \quad (4.1)$$

Este criterio de ordenamiento de izquierda a derecha y de arriba hacia abajo produce la enumeración de orden indicado en la Tabla 4.2.

Tabla 4.2: Rectángulos ordenados.

Rectángulo	Coordenadas			
	Superior Izquierda		Inferior Derecha	
	<i>abscisa (x)</i>	<i>ordenada (y)</i>	<i>abscisa (x)</i>	<i>ordenada (y)</i>
1	0	5 000	1 000	3 500
2	1 000	5 000	3 500	4 000
3	3 500	5 000	4 500	3 000
4	4 500	5 000	5 000	2 000
5	1 000	4 000	2 000	2 000
6	2 000	4 000	3 500	3 000
7	0	3 500	1 000	1 500
8	2 000	3 000	4 500	2 000
9	1 000	2 000	3 500	500
10	3 500	2 000	5 000	1 000
11	0	1 500	1 000	5 00
12	3 500	1 000	4 000	0
13	4 000	1 000	5 000	0
14	0	500	3 500	0

4.1.2. Modelo discreto de la instancia

El modelo discreto de la instancia geométrica R se presenta en la Figura 4.3 a través del uso de un grafo acíclico conexo ponderado con $|V| = 30$ vértices y $|E| = 43$ aristas.

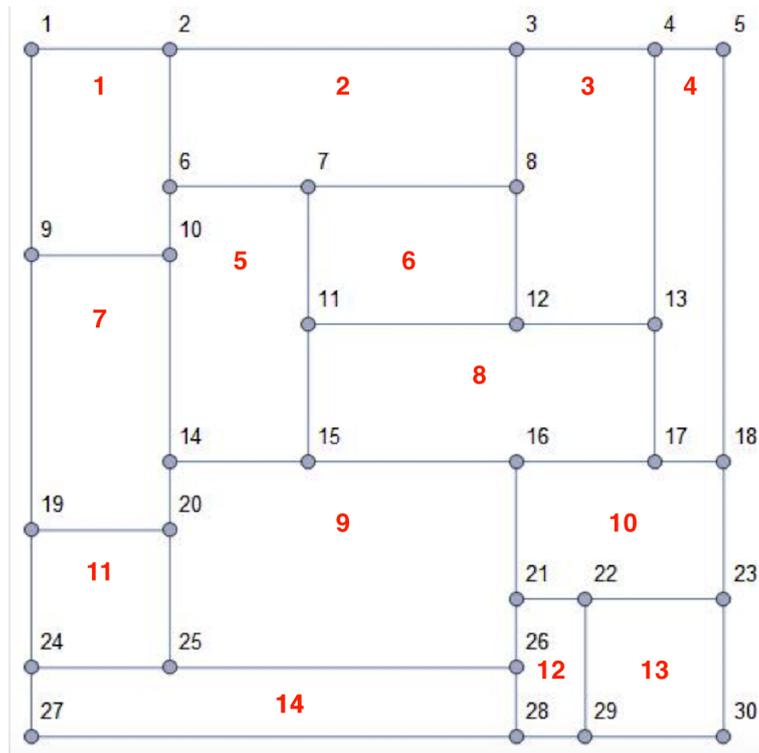


Figura 4.3: Modelo discreto de la instancia geométrica R .

Algunas características que son importantes destacar del modelo discreto de la instancia geométrica son:

- (a) Los vértices que se localizan en el perímetro de la instancia R de tamaño 5000×5000 se consideran vértices de acceso, a través de los cuales se inicia la construcción del corredor de longitud mínima. El conjunto de vértices de acceso para nuestro ejemplo, siguiendo el sentido horario, corresponde al conjunto $\{1, 2, 3, 4, 5, 18, 23, 30, 29, 28, 27, 24, 19, 9\}$.
- (b) En la Tabla 4.2 se muestran el número asignado a cada rectángulo y sus coordenadas de las esquinas superior izquierda e inferior derecha. Se puede observar que en los rectángulos del 1 al 4 se utilizó el primer criterio de ordenación $p_i(y) = p_j(y) \wedge p_i(x) < p_j(x)$, ya que los cuatro rectángulos tienen el mismo valor en la ordenada $y = 5000$ mientras que los valores de las

abscisas x se encuentran ordenados de menor a mayor respectivamente. En el caso de los rectángulos 7 y 8, se aplica el criterio $p_i(y) > p_j(y)$, tomando en cuenta que no existen otros rectángulos con el mismo valor en la ordenada y , y de esta manera se ordenan de menor a mayor.

- (c) Cada rectángulo tiene un conjunto de vértices en su contorno (ver Tabla 4.3), y algunos de estos vértices son compartidos entre dos, tres o hasta cuatro rectángulos. Por ejemplo, el vértice 11 es compartido por los rectángulos 5, 6 y 8; mientras que el nodo 18 es compartido por los rectángulos 4 y 10.

Tabla 4.3: Vértices de los Rectángulos.

Rectángulo	Vértices
1	1, 2, 6, 10, 9
2	2, 3, 8, 7, 6
3	3, 4, 13, 12, 8
4	4, 5, 18, 17, 13
5	6, 7, 11, 15, 14, 10
6	7, 8, 12, 11
7	9, 10, 14, 20, 19
8	11, 12, 13, 17, 16, 15
9	14, 15, 16, 21, 26, 25, 20
10	16, 17, 18, 23, 22, 21
11	19, 20, 25, 24
12	21, 22, 29, 28, 26
13	22, 23, 30, 29
14	24, 25, 26, 28, 27

En la Tabla 4.4 se puede apreciar el conjunto de vértices correspondientes a las cuatro esquinas de cada rectángulo. Por ejemplo, para el rectángulo 1 los cuatro vértice del rectángulo superior izquierdo y derecho, así como inferior izquierdo y derecho son el 1, 2, 10 y 9. Asimismo, se tiene para cada nodo del grafo sus vértices adyacentes y el peso de la arista que los conecta. Cabe señalar que

una buena cantidad de vértices del grafo son de grado 3 (Tabla 4.5).

Tabla 4.4: Vértices de las cuatro esquinas de los rectángulos.

Rectángulo	Vértices Superiores		Vértices Inferiores	
	Izquierda	Derecha	Izquierda	Derecha
1	1	2	10	9
2	2	3	8	6
3	3	4	13	12
4	4	5	18	17
5	6	7	15	14
6	7	8	12	11
7	9	10	20	19
8	11	13	17	15
9	14	16	26	25
10	16	18	23	21
11	19	20	25	24
12	21	22	29	28
13	22	23	30	29
14	24	26	28	27

Tabla 4.5: Vértices adyacentes y pesos de las aristas.

Vértice	Coordenada		Vértice adyacente	Peso arista	Vértice adyacente	Peso arista	Vértice adyacente	Peso arista
	x	y						
1	0	5000	2	1,000	9	1,500	—	—
2	1000	5000	1	1,000	3	2,500	6	1,000
3	3500	5000	2	2,500	4	1,000	8	1,000
4	4500	5000	3	1,000	5	500	13	2,000
5	5000	5000	4	500	18	3,000	—	—
6	1000	4000	2	1,000	7	1,000	10	500
7	2000	4000	6	1,000	8	1,500	11	1,000
8	3500	4000	3	1,000	7	1,500	12	1,000
9	0	3500	1	1,500	10	1,000	19	2,000
10	1000	3500	6	500	9	1,000	14	1,500
11	2000	3000	7	1,000	12	1,500	15	1,000
12	3500	3000	8	1,000	11	1,500	13	1,000
13	4500	3000	4	2,000	12	1,000	17	1,000
14	1000	2000	10	1,500	15	1,000	20	500
15	2000	2000	11	1,000	14	1,000	16	1,500
16	3500	2000	15	1,500	17	1,000	21	1,000
17	4500	2000	13	1,000	16	1,000	18	500
18	5000	2000	5	3,000	17	500	23	1,000
19	0	1500	9	2,000	20	1,000	24	1,000
20	1000	1500	14	500	19	1,000	25	1,000
21	3500	1000	16	1,000	22	500	26	500
22	4000	1000	21	500	23	1,000	29	1,000
23	5000	1000	18	1,000	22	1,000	30	1,000
24	0	500	19	1,000	25	1,000	27	500
25	1000	500	20	1,000	24	1,000	26	2,500
26	3500	500	21	500	25	2,500	28	500
27	0	0	24	500	28	3,500	—	—
28	3500	0	26	500	27	3,500	29	500
29	4000	0	22	1,000	28	500	30	1,000
30	5000	0	23	1,000	29	1,000	—	—

4.1.3. Matriz de incidencia del grafo

En la Figura 4.4 se presenta la matriz de incidencia correspondiente al modelado discreto de la instancia R . Los puntos en la matriz indican las incidencias de las aristas entre los diferentes nodos del grafo. En esta representación gráfica se puede observar la simetría que existe en los nodos conectados.

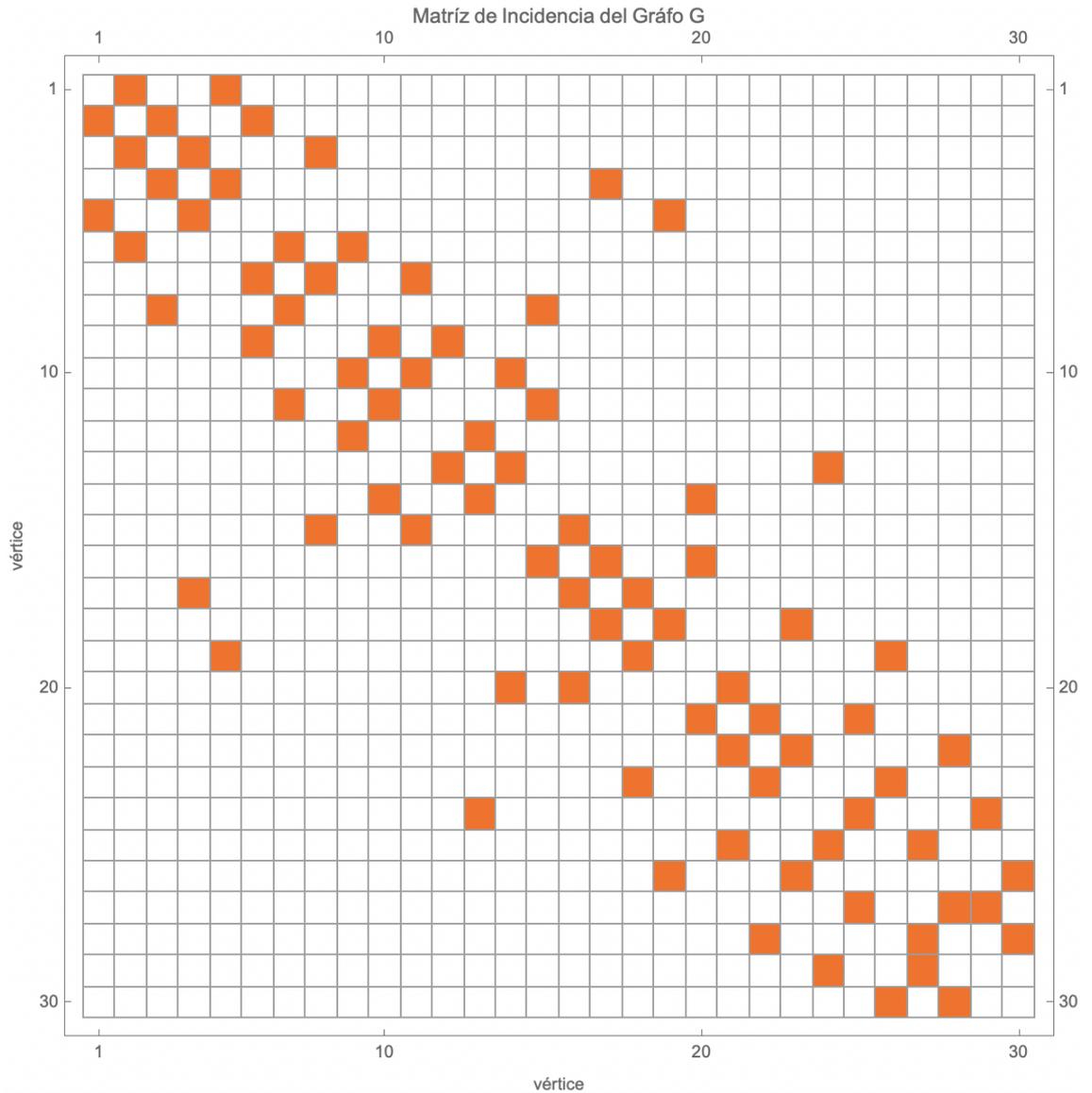


Figura 4.4: Matriz de incidencia del gráfico $G = (V, E, w)$.

4.2. Diseño de algoritmos para el cálculo del Corredor de Longitud Mínima (MLC)

El objetivo es obtener el *Corredor de Longitud Mínima (MLC)* usando el algoritmo que calcula el árbol extendido de costo mínimo modificado a partir de

un instancia geométrica R dividida en m rectángulos rectilíneos, dado un vértice de acceso v_s .

Se diseñaron e implementaron las siguientes seis heurísticas para la obtención del árbol solución al problema del MLC :

- (1) Algoritmo de Búsqueda Voraz – Dominós más cercanos.
- (2) Algoritmo de Búsqueda Voraz – Dominós más alejados.
- (3) Algoritmo de Vértices Compartidos.
- (4) Algoritmo de Vértices Compartidos Mejorado.
- (5) Algoritmo de Árbol Extendido de Costo Mínimo – Reducción.
- (6) Algoritmo de Árbol Extendido de Costo Mínimo con Vértices Internos – Reducción.

4.2.1. Heurística 1: Algoritmo de Búsqueda Voraz – Dominós más cercanos

En esta primera solución algorítmica construye el árbol incluyendo los rectángulos más cercanos a los nodos que conforman la solución parcial hasta alcanzar todos los rectángulos en al menos uno de sus vértices. Los pasos principales son:

1. Se inicia el árbol solución con un vértice raíz $root$, esto es, el nodo de acceso en la periferia de la instancia geométrica representada por un grafo.
2. Se marcan los rectángulos que comparten este vértice como rectángulos alcanzados, como se muestra en la Figura 4.5.

3. A partir del vértice *root* se calcula la ruta más corta a todos los vértices de cada rectángulo no alcanzado de la instancia.
4. Se selecciona entre todas estas rutas cortas la que tenga la menor longitud; es decir, la ruta del rectángulo más cercano.
5. Se marca el rectángulo alcanzado y la ruta óptima como se muestra en la Figura 4.6.
6. Considerando todos los vértices de la solución parcial construida, se busca la ruta más corta a todos los vértices de cada rectángulo no alcanzado de la instancia.
7. Regresar al paso 4 hasta que se hayan alcanzado todos los rectángulos de la instancia geométrica.

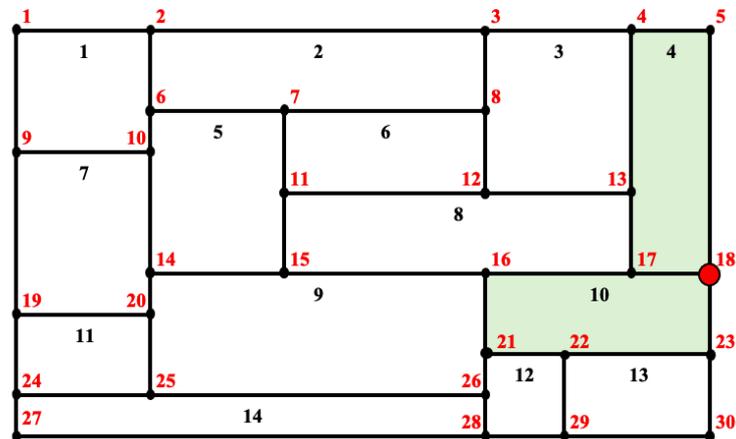


Figura 4.5: Rectángulos 4 y 10 alcanzados por el vértice raíz $v = 18$ del árbol solución.

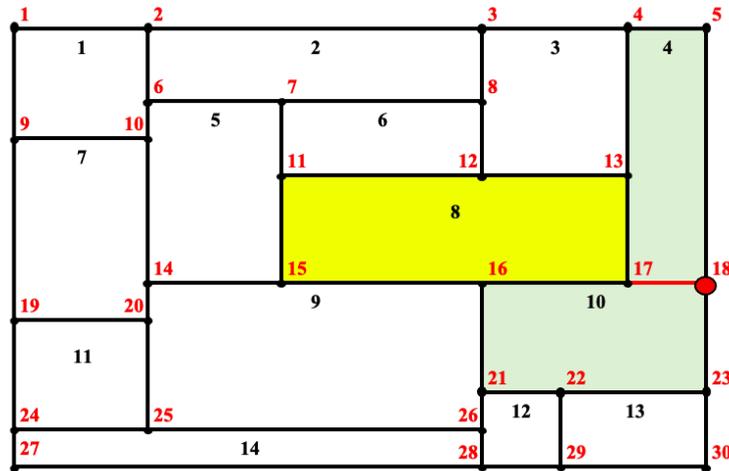


Figura 4.6: Rectángulo 8 cercano al vértice raíz $v = 18$ a través del vértice 17 y la arista $\{18, 17\}$.

La solución propuesta anterior para el problema MLC se basa en la técnica de diseño *voraz*, y se presenta en el Algoritmo 11, el cual consiste en el procedimiento $MLC - H1(V_G, E_G, R_G, root)$. El procedimiento requiere como datos de entrada correspondiente a la instancia geométrica: el conjunto de nodos V_G y aristas E_G del grafo, el conjunto de rectángulos R_G en que se divide la instancia, el nodo raíz del árbol $root$ que servirá de acceso,

Las conjuntos V'_T , E'_T y R'_T de nodos, aristas y rectángulos respectivamente, se inicializan con $NULL$ y se emplearán para guardar la solución parcial que conforma el corredor hasta el momento construido (Línea 1 del Algoritmo 11). El conjunto V_r de nodos del rectángulo r se inicializa con los vértices que se encuentran en la periferia de cada rectángulo r (Línea 3 del Algoritmo 11). Los rectángulos que son alcanzados por el nodo $root$, se agregan al conjunto R'_T de rectángulos del árbol T , y se eliminan del conjunto R_G de rectángulos del grafo G (Líneas 2-7 del Algoritmo 11). Así mismo el nodo $root$ se agrega al conjunto V'_T de vértices del árbol T y se elimina del conjunto V_G de vértices del grafo G (Línea 8 del Algoritmo 11).

Mientras que existan rectángulos, esto es $R_G \neq \emptyset$, que no hayan sido alcanzados (Líneas 9-30 del Algoritmo 11), para cada uno de los rectángulos del conjunto R_G se inicializan las variables correspondientes al peso mínimo de la ruta encontrada min_weight_path y la ruta mínima min_path con un valor grande y $NULL$, respectivamente (Línea 10 del Algoritmo 11). Se realiza una búsqueda de la ruta más corta entre cada nodo u de la solución parcial y cada nodo v de cada rectángulo r usando el algoritmo de Dijkstra, esto es, $min\{\forall r \in R_G \forall u \in V'_T \forall v \in V_r DIJSKTRA(u, v)\}$ (Línea 14 del Algoritmo 11). Para la selección de la ruta mínima, si se cumple que el peso de la ruta $w(path)$ es menor que min_weight_path (Línea 15 del Algoritmo 11), se actualizan las variables min_weight_path y min_path por $w(path)$ y $path$, respectivamente, obteniéndose el rectángulo r más cercano a la solución parcial hasta el momento construida (Líneas 11-20 del Algoritmo 11).

A través de las funciones $V(min_path)$ y $E(min_path)$ se obtienen los conjuntos de nodos y aristas de la ruta corta entre los nodos u y v respectivamente (Líneas 21-22 del Algoritmo 11), los cuales se agregan a los conjuntos V'_T y E'_T de nodos y aristas del árbol T ; mientras que en los conjuntos V_G y E_G de nodos y aristas del grafo G se eliminan (Líneas 21-22 del Algoritmo 11). En consecuencia, los rectángulos $r \in R_G$ que comparten algún vértice de $path$ son agregados al conjunto R'_T de rectángulos del árbol T y se eliminan del conjunto R_G de rectángulos del grafo G (Líneas 23-29 del Algoritmo 11).

Finalmente el corredor obtenido por el algoritmo se obtendrá con la información que entrega a través de las variables V'_T y E'_T , correspondientes a los nodos y aristas del árbol T (Línea 31 del Algoritmo 11).

Algoritmo 11 Algoritmo de Búsqueda Voraz - Dominós más cercanos.

procedure *MLC* – $H1(V_G, E_G, R_G, root)$

```
1:  $V'_T := NULL, E'_T := NULL, R'_T := NULL$ 
2: for all  $r \in R_G$  do
3:    $V_r := \{v \mid v \text{ es un nodo de la periferia de } r\}$ 
4:   if  $root \in V_r$  then
5:      $R'_T := R'_T \cup \{r\}, R_G := R_G - \{r\}$ 
6:   end if
7: end for
8:  $V'_T := V'_T \cup \{root\}, V_G := V_G - \{root\}$ 
9: while  $R_G \neq \emptyset$  do
10:   $min\_weight\_path := \infty, min\_path := NULL$ 
11:  for all  $r \in R_G$  do
12:    for all  $u \in V'_T$  do
13:      for all  $v \in V_r$  do
14:         $path := DIJKSTRA(u, v)$ 
15:        if  $w(path) < min\_weight\_path$  then
16:           $min\_weight\_path := w(path), min\_path := path$ 
17:        end if
18:      end for
19:    end for
20:  end for
21:   $V'_T := V'_T \cup V(min\_path), E'_T := E'_T \cup E(min\_path)$ 
22:   $V_G := V_G - V(min\_path), E_G := E_G - E(min\_path)$ 
23:  for all  $r \in R_G$  do
24:    for all  $v \in V(min\_path)$  do
25:      if  $v \in V_r$  then
26:         $R'_T := R'_T \cup \{r\}, R_G := R_G - \{r\}$ 
27:      end if
28:    end for
29:  end for
30: end while
31: return  $V'_T, E'_T$ 
```

Se realizaron pruebas con la instancia mostrada en la Figura 4.3. Los vértices localizados en el perímetro del instancia R como puntos de acceso fueron 1, 2, 3, 4, 5, 18, 23, 30, 29, 28, 27, 24, 19 y 9. En la Tabla 4.6 se concentra la información de los corredores de longitud mínima obtenidos por el Algoritmo 11 que incluye el punto de acceso, vértices del árbol, aristas del árbol y longitud total del

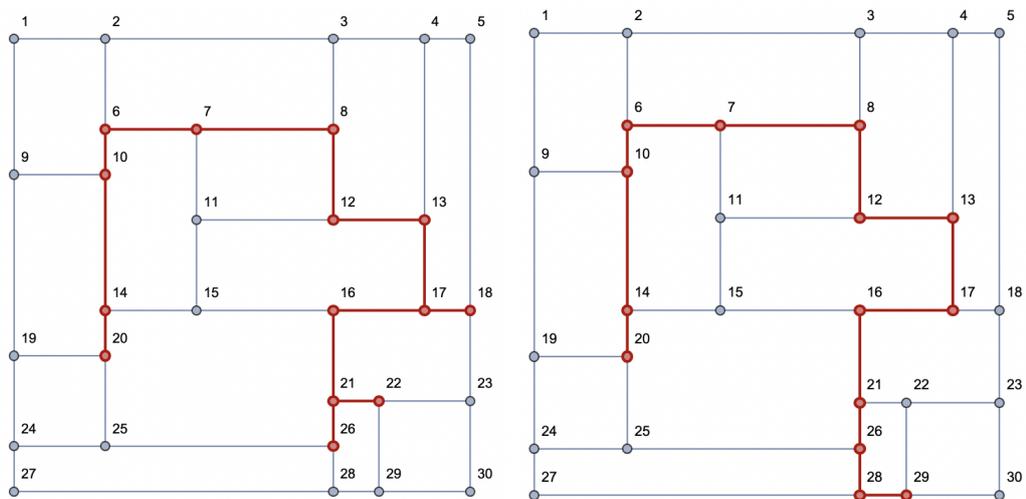
corredor.

Tabla 4.6: Corredores de Longitud Mínima.

Punto Acceso	Corredor		
	Vértices	Aristas	Longitud
1	1, 2, 6, 10, 7, 11, 15, 8, 16, 17, 21, 22, 26, 14, 20	{1,2}, {2,6}, {6,10}, {6,7}, {7,11}, {11,15}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}, {21,26}, {15,14}, {14,20}	13 000
2	2, 6, 10, 7, 11, 15, 8, 16, 17, 21, 22, 26, 14, 20	{2,6}, {6,10}, {6,7}, {7,11}, {11,15}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}, {21,26}, {15,14}, {14,20}	12 000
3	3, 4, 8, 12, 7, 6, 10, 14, 20, 25, 13, 17, 18, 23, 22	{3,4}, {3,8}, {8,12}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	13 500
4	4, 3, 8, 12, 7, 6, 10, 14, 20, 25, 13, 17, 18, 23, 22	{4,3}, {3,8}, {8,12}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	13 500
5	5, 4, 3, 8, 12, 7, 6, 10, 14, 20, 25, 13, 17, 18, 23, 22	{5,4}, {4,3}, {3,8}, {8,12}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	14 000
18	18, 17, 13, 12, 8, 16, 21, 22, 26, 7, 6, 10, 14, 20	{18,17}, {17,13}, {13,12}, {12,8}, {17,16}, {16,21}, {21,22}, {21,26}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}	11 500
23	23, 18, 17, 13, 12, 8, 16, 21, 26, 7, 6, 10, 14, 20	{23,18}, {18,17}, {17,13}, {13,12}, {12,8}, {17,16}, {16,21}, {21,26}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}	12 000
30	30, 23, 18, 17, 13, 12, 8, 16, 21, 26, 7, 6, 10, 14, 20	{30,23}, {23,18}, {18,17}, {17,13}, {13,12}, {12,8}, {17,16}, {16,21}, {21,26}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}	13 000
29	29, 28, 26, 21, 16, 17, 13, 12, 8, 7, 6, 10, 14, 20	{29,28}, {28,26}, {26,21}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}	11 500
28	28, 26, 21, 22, 16, 17, 13, 12, 8, 7, 6, 10, 14, 20	{28,26}, {26,21}, {21,22}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}	11 500
27	27, 24, 19, 20, 14, 15, 11, 7, 6, 8, 16, 17, 21, 22	{27,24}, {24,19}, {19,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,6}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}	12 500
24	24, 19, 20, 14, 15, 11, 7, 6, 8, 16, 17, 21, 22	{24,19}, {19,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,6}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}	12 000
19	19, 20, 14, 15, 11, 7, 6, 24, 8, 16, 17, 21, 22	{19,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,6}, {19,24}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}	12 000
9	9, 10, 6, 7, 11, 15, 8, 16, 17, 21, 22, 26, 14, 20	{9,10}, {10,6}, {6,7}, {7,11}, {11,15}, {7,8}, {15,16}, {16,17}, {16,21}, {21,22}, {21,26}, {15,14}, {14,20}	12 000

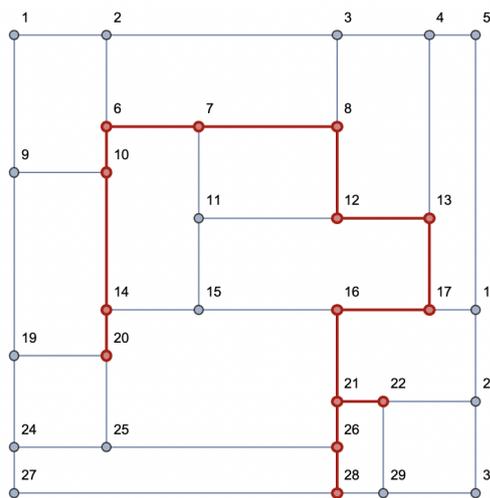
Los tres mejores casos corresponden a los puntos de acceso 18, 29 y 28 con una longitud total de 11,500; mientras que el peor caso fue el obtenido para el nodo 5 con una longitud total de 14,000.

En la Figura 4.7 se muestran los resultados para los casos de los corredores con longitud total de 11,500 construidos a partir de los puntos de acceso 18, 29 y 28, respectivamente. Mientras que en la Figura 4.8 se muestran el caso con mayor longitud del corredor de 14,000.



(a) Punto de acceso 18.

(b) Punto de acceso 29.



(c) Punto de acceso 28.

Figura 4.7: Corredor de longitud mínima de 11,500: mejor caso.

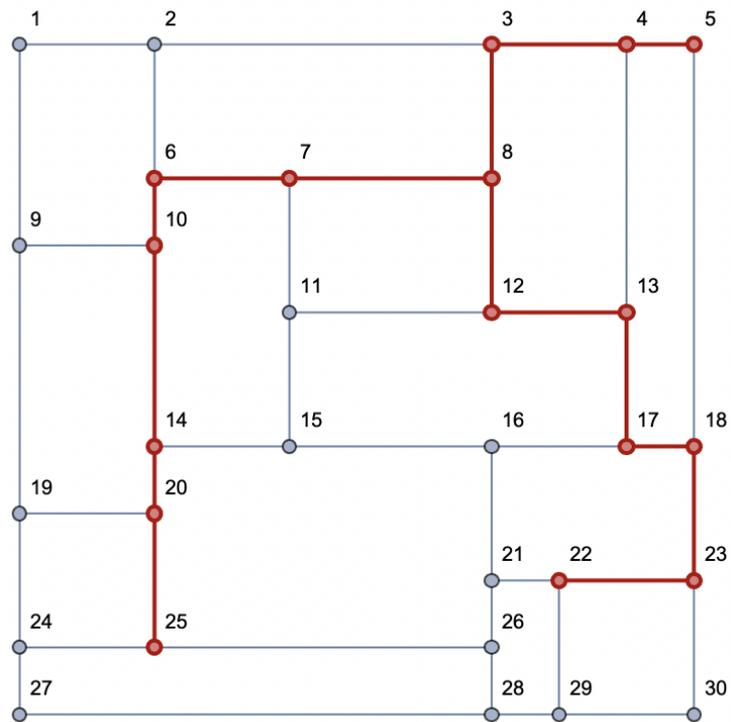


Figura 4.8: Longitud del corredor 14,000 y Punto de acceso 5: peor caso.

4.2.2. Heurística 2: Algoritmo de Búsqueda Voraz – Dominós más lejanos.

En esta segunda solución algorítmica se buscará ir construyendo el árbol incluyendo los rectángulos más lejanos con distancia mínima a partir de los nodos que conforman el árbol solución, hasta alcanzar todos los rectángulos en al menos uno de sus vértices. Los pasos principales son:

1. Se inicia el árbol solución con un vértice raíz *root* en la periferia de la instancia geométrica, representada por un grafo.
2. Se marcan los rectángulos que comparten este vértice como rectángulos alcanzados, como se muestra en la Figura 4.9.

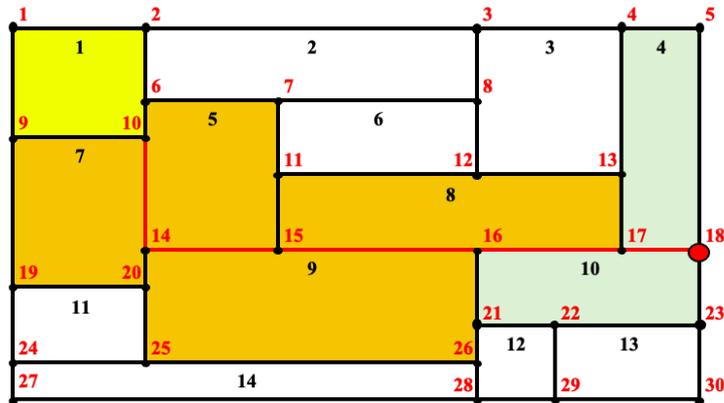


Figura 4.10: Rectángulo 1 más alejado con distancia mínima al vértice raíz $v = 18$ a través del vértice 10 y las aristas $\{\{18, 17\}, \{17, 16\}, \{16, 15\}, \{15, 14\}, \{14, 10\}\}$ alcanzando también los rectángulos 5, 7, 8 y 9.

La solución propuesta para el problema MLC se basa en una técnica de diseño de *algoritmo voraz*, resultando el Algoritmo 12, el cual consiste en el procedimiento $MLC - H2(V_G, E_G, R_G, root)$. El procedimiento requiere como datos de entrada correspondiente a la instancia geométrica: el conjunto de nodos V_G y aristas E_G del grafo, el conjunto de rectángulos R_G en que se divide la instancia, el nodo raíz del árbol $root$ que servirá de acceso.

Los conjuntos V'_T , E'_T y R'_T de nodos, aristas y rectángulos respectivamente, se inicializan con $NULL$ y se emplearán para guardar la solución parcial que conforman el corredor hasta el momento construido (Línea 1 del Algoritmo 12). El conjunto V_r de nodos del rectángulo r se inicializa con los vértices que se encuentran en la periferia de cada rectángulo r (Línea 3 del Algoritmo 12). Los rectángulos que son alcanzados por el nodo $root$, se agregan al conjunto R'_T de rectángulos del árbol T , y se eliminan del conjunto R_G de rectángulos del grafo G (Líneas 2-7 del Algoritmo 12). Así mismo el nodo $root$ se agrega al conjunto V'_T de vértices del árbol T y se elimina del conjunto V_G de vértices del grafo G (Línea 8 del Algoritmo 12).

Mientras que existan rectángulos, esto es $R_G \neq \emptyset$, que no hayan sido alcanzados (Líneas 9-36 del Algoritmo 12) para cada uno de los rectángulos del conjunto R_G se inicializan las variables correspondiente al peso mínimo de la ruta encontrada min_weight_path y el peso mínimo de la ruta encontrada al rectángulo más alejado $min_weight_path_far$ con un valor grande y con un valor de 0, respectivamente. Los conjuntos de la ruta mínima cercana min_path_near y la ruta mínima lejana min_path_far con un valor $NULL$ respectivamente (Líneas 10-11 del Algoritmo 12). Se realiza una búsqueda de la ruta más corta entre los nodos del árbol u y los nodos v del rectángulo usando el algoritmo de Dijkstra bajo la siguiente condición $\forall r \in R_G \forall u \in V'_T \forall v \in V_r DIJSKTRA(u, v)$ (Línea 15 del Algoritmo 12). Para la selección de la ruta mínima se debe cumplir la condición que el peso de la ruta $w(path)$ sea menor al valor que tiene la variable min_weight_path (Línea 16 del Algoritmo 12), realizando una actualización de las variables min_weight_path y min_path_near por $w(path)$ y $path$, respectivamente, obtenida para el rectángulo r (Líneas 12-21 del Algoritmo 12). Se compara el peso de la ruta mínima cercana $w(min_path_near)$ si es mayor que $min_weight_path_far$, si fuera el caso encontramos el peso y ruta del rectángulo más alejado de los nodos del árbol y se actualizan el peso mínimo de la ruta encontrada al rectángulo más alejado $min_weight_path_far$ con el peso de la ruta mínima cercana, así como la ruta mínima lejana min_path_far con la ruta mínima cercana min_path_near (Líneas 22-25 del Algoritmo 12).

A través de las funciones $V(min_path_far)$ y $E(min_path_far)$ se obtienen los conjunto de nodos y aristas de la ruta corta del rectángulo más lejano entre los nodos u y v respectivamente (Líneas 27-28 del Algoritmo 11), los cuales se agregan a los conjuntos V'_T y E'_T de nodos y aristas del árbol T ; mientras que en los conjuntos V_G y E_G de nodos y aristas del grafo G se eliminan (Líneas 27-28

del Algoritmo 11). En consecuencia los rectángulos $r \in R_G$ que comparten algún vértice de *min_path_far* son agregados al conjunto de rectángulos del árbol R'_T y se eliminan del conjunto de rectángulos del grafo R_G (Líneas 29-35 del Algoritmo 12).

Finalmente el corredor obtenido por el algoritmo se obtendrá con la información que entrega a través de las variables V'_T y E'_T correspondientes a los nodos y aristas del árbol (Línea 37 del Algoritmo 12).

Algoritmo 12 Algoritmo de Búsqueda Voraz - Dominós más lejanos.

procedure *MLC* – $H2(V_G, E_G, R_G, root)$

```
1:  $V'_T := NULL, E'_T := NULL, R'_T := NULL$ 
2: for all  $r \in R_G$  do
3:    $V_r := \{v \mid v \text{ es un nodo de la periferia de } r\}$ 
4:   if  $root \in V_r$  then
5:      $R'_T := R'_T \cup \{r\}, R_G := R_G - \{r\}$ 
6:   end if
7: end for
8:  $V'_T := V'_T \cup \{root\}, V_G := V_G - \{root\}$ 
9: while  $R_G \neq \emptyset$  do
10:   $min\_weight\_path := \infty, min\_weight\_path\_far := 0,$ 
11:   $min\_path\_near := NULL, min\_path\_far := NULL$ 
12:  for all  $r \in R_G$  do
13:    for all  $u \in V'_T$  do
14:      for all  $v \in V_r$  do
15:         $path := DIJKSTRA(u, v)$ 
16:        if  $w(path) < min\_weight\_path$  then
17:           $min\_weight\_path := w(path)$ 
18:           $min\_path\_near := path$ 
19:        end if
20:      end for
21:    end for
22:    if  $w(min\_path\_near) > min\_weight\_path\_far$  then
23:       $min\_weight\_path\_far := w(min\_path\_near)$ 
24:       $min\_path\_far := min\_path\_near$ 
25:    end if
26:  end for
27:   $V'_T := V'_T \cup V(min\_path\_far), E'_T := E'_T \cup E(min\_path\_far)$ 
28:   $V_G := V_G - V(min\_path\_far), E_G := E_G - E(min\_path\_far)$ 
29:  for all  $r \in R_G$  do
30:    for all  $v \in V(min\_path\_far)$  do
31:      if  $v \in V_r$  then
32:         $R'_T := R'_T \cup \{r\}, R_G := R_G - \{r\}$ 
33:      end if
34:    end for
35:  end for
36: end while
37: return  $V'_T, E'_T$ 
```

La Tabla 4.7 concentra la información de los corredores de longitud mínima obtenidos por el Algoritmo 12 que incluye el punto de acceso, vértices del árbol,

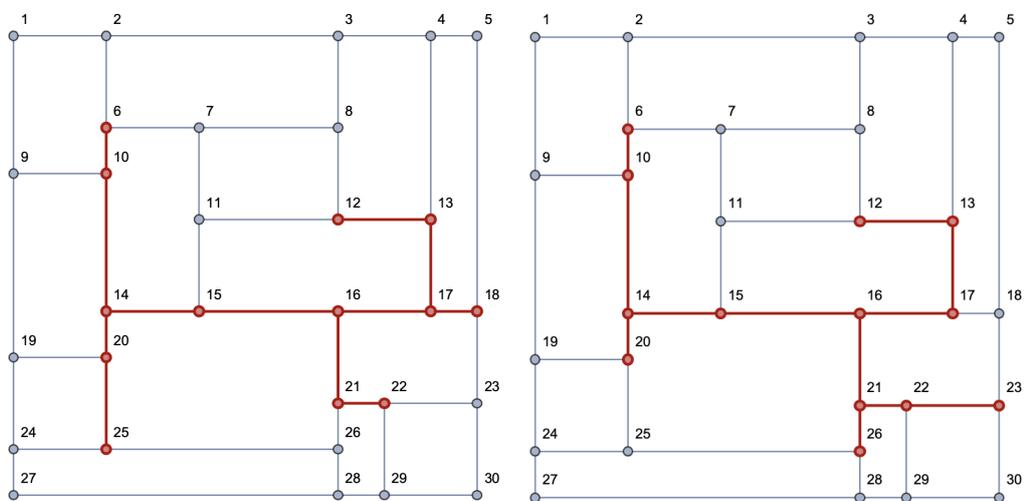
aristas del árbol y longitud total del corredor.

Tabla 4.7: Corredores de Longitud Mínima.

Punto Acceso	Corredor		
	Vértices	Aristas	Longitud
1	1, 2, 6, 7, 11, 15, 16, 21, 22, 8, 14, 20, 17, 26	{1,2}, {2,6}, {6,7}, {7,11}, {11,15}, {15,16}, {16,21}, {21,22}, {7,8}, {15,14}, {14,20}, {16,17}, {21,26}	12 500
2	2, 6, 7, 11, 15, 16, 21, 22, 8, 14, 20, 17, 26	{2,6}, {6,7}, {7,11}, {11,15}, {15,16}, {16,21}, {21,22}, {7,8}, {15,14}, {14,20}, {16,17}, {21,26}	11 500
3	3, 4, 13, 17, 16, 21, 26, 2, 25, 6, 8, 10, 22	{3,4}, {4,13}, {13,17}, {17,16}, {16,21}, {21,26}, {3,2}, {26,25}, {2,6}, {3,8}, {6,10}, {21,22}	14 500
4	4, 13, 17, 16, 15, 14, 20, 10, 21, 22, 12, 6, 26	{4,13}, {13,17}, {17,16}, {16,15}, {15,14}, {14,20}, {14,10}, {16,21}, {21,22}, {13,12}, {10,6}, {21,26}	12 000
5	5, 18, 17, 16, 15, 14, 20, 10, 11, 21, 6, 4, 22, 26	{5,18}, {18,17}, {17,16}, {16,15}, {15,14}, {14,20}, {14,10}, {15,11}, {16,21}, {10,6}, {5,4}, {21,22}, {21,26}	13 000
18	18, 17, 16, 15, 14, 10, 20, 25, 13, 12, 21, 6, 22	{18,17}, {17,16}, {16,15}, {15,14}, {14,10}, {14,20}, {20,25}, {17,13}, {13,12}, {16,21}, {10,6}, {21,22}	11 000
23	23, 22, 21, 16, 15, 14, 10, 17, 13, 12, 6, 20, 26	{23,22}, {22,21}, {21,16}, {16,15}, {15,14}, {14,10}, {16,17}, {17,13}, {13,12}, {10,6}, {14,20}, {21,26}	11 000
30	30, 29, 28, 26, 25, 20, 14, 10, 6, 7, 8, 3, 4, 11, 21	{30,29}, {29,28}, {28,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}, {6,7}, {7,8}, {8,3}, {3,4}, {7,11}, {26,21}	14 000
29	29, 28, 26, 25, 20, 14, 10, 6, 7, 8, 3, 4, 11, 21	{29,28}, {28,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}, {6,7}, {7,8}, {8,3}, {3,4}, {7,11}, {26,21}	11 500
28	28, 26, 25, 20, 14, 10, 6, 7, 8, 3, 4, 11, 21, 22	{28,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}, {6,7}, {7,8}, {8,3}, {3,4}, {7,11}, {26,21}, {21,22}	13 000
27	27, 24, 19, 20, 14, 15, 11, 12, 16, 21, 22, 10, 13, 6	{27,24}, {24,19}, {19,20}, {20,14}, {14,15}, {15,11}, {11,12}, {15,16}, {16,21}, {21,22}, {14,10}, {12,13}, {10,6}	12 500
24	24, 25, 20, 14, 15, 11, 12, 16, 21, 22, 10, 13, 6	{24,25}, {25,20}, {20,14}, {14,15}, {15,11}, {11,12}, {15,16}, {16,21}, {21,22}, {14,10}, {12,13}, {10,6}	12 000
19	19, 20, 25, 26, 21, 22, 16, 17, 13, 9, 10, 6, 7	{19,20}, {20,25}, {25,26}, {26,21}, {21,22}, {21,16}, {16,17}, {17,13}, {19,9}, {9,10}, {10,6}, {6,7}	13 000
9	9, 10, 14, 15, 16, 21, 22, 17, 13, 12, 6, 20, 26	{9,10}, {10,14}, {14,15}, {15,16}, {16,21}, {21,22}, {16,17}, {17,13}, {13,12}, {10,6}, {14,20}, {21,26}	11 000

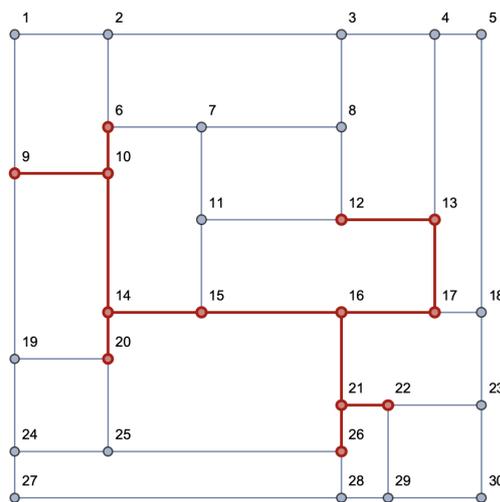
Los tres mejores casos corresponden a los puntos de acceso 18, 23 y 9 con una longitud total de 11,000; mientras que el peor caso fue obtenido para el nodo 3 con una longitud total de 14,500.

En la Figura 4.11 se muestran los casos de los corredores con longitud total de 11,000 iniciando a partir de los puntos de acceso 18, 23 y 9, respectivamente. Mientras que en la Figura 4.12 se muestran el caso con mayor longitud del corredor de 14,500.



(a) Punto de acceso 18.

(b) Punto de acceso 23.



(c) Punto de acceso 9.

Figura 4.11: Corredor de longitud mínima de 11,000: mejor caso.

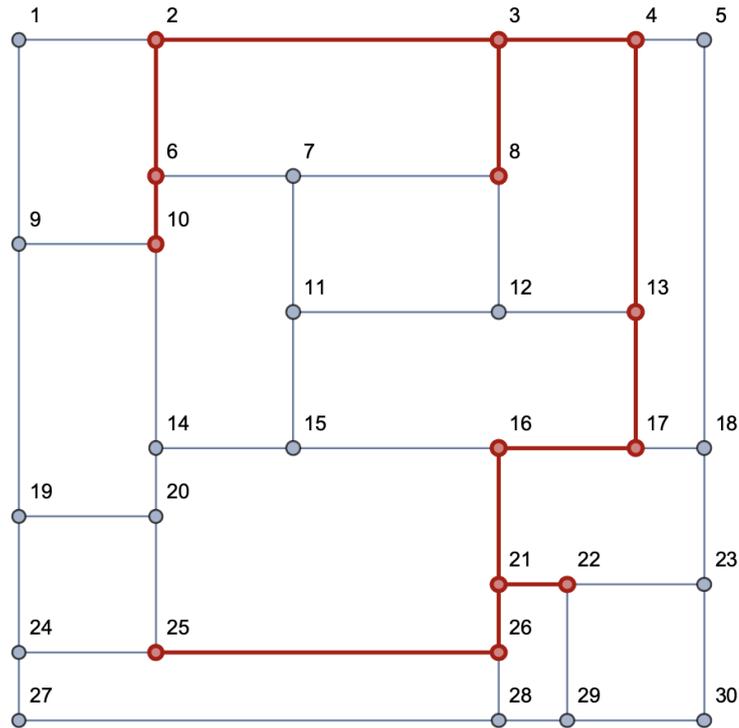


Figura 4.12: Longitud del corredor 14,500 y Punto de acceso 3: peor caso.

4.2.3. Heurística 3: Algoritmo de Vértices Compartidos

En esta tercera solución algorítmica, el árbol se construye a partir de los vértices de mayor grado de incidencia sobre los rectángulos y se selecciona la ruta más corta para conectarlos construyendo el árbol solución, hasta alcanzar todos los rectángulos en al menos uno de sus vértices. Los pasos principales son:

1. Ordenar los vértices del grafo de acuerdo al grado de incidencia sobre los rectángulos no alcanzados; es decir, cuántos rectángulos comparten el vértice.
2. Seleccionar el vértice de mayor grado como se muestra en la Figura 4.13.

3. Marcar los rectángulos que comparten este vértice como rectángulos alcanzados como se muestra en la Figura 4.9.
4. Mientras no se hayan alcanzado todos los rectángulos de la instancia, regresar al paso 1.
5. Con los dos vértices de mayor grado construir la ruta más corta entre los dos vértices.
6. Conectar el vértice de mayor grado de incidencia no conectado con el vértice más cercano que pertenece a la solución parcial.
7. Mientras existan vértices de mayor grado no conectados, repetir el paso 6.

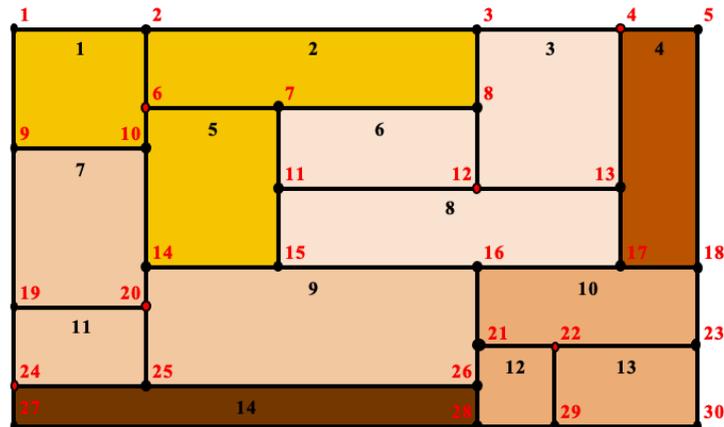


Figura 4.13: Vértices con mayor incidencia: vértice 6 incidente sobre los rectángulos 1, 2 y 5; vértice 12 incidente sobre los rectángulos 3, 6 y 8; vértice 20 incidente sobre los rectángulos 7, 9 y 11; vértice 22 incidente sobre los rectángulos 10, 12 y 13; vértice 4 incidente sobre los rectángulos 3 y 4; vértice 24 incidente sobre los rectángulos 11 y 14.

La solución propuesta para el problema MLC se presenta en el Algoritmo 13 el cual consiste en el procedimiento $MLC - H3(V_G, E_G, R_G)$. El procedimiento requiere como datos de entrada correspondientes a la instancia geométrica: el

conjunto de nodos V_G y aristas E_G del grafo, y el conjunto de rectángulos R_G en que se divide la instancia.

Los conjuntos V'_T , E'_T y V_C de nodos, aristas y vértices compartidos respectivamente se inicializan con $NULL$ y se emplean para guardar la solución parcial que conforma el corredor hasta el momento construido. (Línea 1 del Algoritmo 13).

Mientras que existan rectángulos, es decir $R_G \neq \emptyset$, que no hayan sido alcanzados (Líneas 2-23 del Algoritmo 13) para cada uno de los rectángulos del conjunto R_G se inicializan las variables correspondientes al grado de vértice máximo $grado_vertice_max$ con un valor de 0, el rectángulo seleccionado r con el primer rectángulo del conjunto R_G y el conjunto de vértices del rectángulo V_r se inicializa con los vértices que se encuentran en la periferia de cada rectángulo r respectivamente (Líneas 3-4 del Algoritmo 13).

Para todos los vértices v del conjunto V_r se obtiene el grado de incidencia sobre los rectángulos, para seleccionar el que tiene el mayor grado. La función $grado(v, R_G)$ devuelve el grado de incidencia sobre los rectángulos del vértice v en el conjunto de rectángulos R_G de la instancia geométrica. Se asigna el grado y vértice en las variables $grado_vertice_max$ y v_{sel} , respectivamente (Líneas 5-9 del Algoritmo 13). Si el grado del vértice seleccionado es igual a 1 se busca un vértice del rectángulo seleccionado que tenga el mayor grado de incidencia tomando en cuenta el conjunto de todos los rectángulos de la instancia geométrica R_G con la finalidad de alcanzar una mayor cantidad de rectángulos. Los valores del grado y vértice son asignados a las variables $grado_vertice_max$ y v_{sel} , respectivamente (Líneas 10-16 del Algoritmo 13). Se actualiza el conjunto R_G de rectángulos de la instancia eliminando aquellos rectángulos que comparten el vértice seleccionado v_{sel} (Líneas 17-21 del Algoritmo 13). Finalmente se actualiza el conjunto de

vértices compartidos v_{comp} agregando el vértice seleccionado v_{sel} (Línea 22 del Algoritmo 13).

Posteriormente el conjunto V'_T de vértices de la solución parcial del árbol T se le asigna el primer vértice con el mayor grado de incidencia del conjunto V_C y se actualiza el conjunto V_C de vértices compartidos eliminando el de mayor grado de incidencia (Línea 24 del Algoritmo 13). Mientras el conjunto $V_C \neq \emptyset$ existirán vértices compartidos para agregados al árbol solución parcial T (Líneas 25-37 del Algoritmo 13) se realizarán las siguientes operaciones: se inicializa la variable min_weight_path con un valor grande ∞ (Línea 26 del Algoritmo 13); se emplea el algoritmo de *DIJKSTRA* para calcular la ruta corta entre los vértices u del conjunto V'_T del árbol solución T y el vértice v del conjunto V_C de vértices con el mayor grado de incidencia, si el peso $w(DIJKSTRA(u, v))$ es menor que min_weight_path se ha encontrado una ruta mínima entre los vértices u y v , se asigna el resultado de la ruta a la variable p_m (Líneas 27-33 del Algoritmo 13).

A través de las funciones $V(p_m)$ se obtiene el conjunto de nodos de la ruta corta entre los vértices u y v , se actualizan los conjuntos de nodos compartidos V_C eliminando los vértices de del conjunto $V(p_m)$, así como los conjuntos de nodos y aristas del árbol agregando los vértices de la ruta $V(p_m)$ a V'_T y las aristas de la ruta p_m al conjunto E'_T (Línea 34 del Algoritmo 13).

Finalmente la información que entrega el algoritmo a través de las variables V'_T y E'_T correspondientes a los nodos y aristas del corredor (Línea 38 del Algoritmo 13).

Algoritmo 13 Algoritmo de Vértices Compartidos.

procedure *MLC* – *H3*(V_G, E_G, R_G)

```
1:  $V'_T := NULL, E'_T := NULL, V_C := NULL$ 
2: while  $R_G \neq \emptyset$  do
3:    $grado\_vertice\_max := 0, r := First[R_G], V_r := \{v \mid v \text{ nodo periferia de } r\}$ 
4:   for all  $v \in V_r$  do
5:     if  $grado(v, R_G) > grado\_vertice\_max$  then
6:        $grado\_vertice\_max := grado(v, R_G), v_{sel} := v$ 
7:     end if
8:   end for
9:   if  $grado\_vertice\_max = 1$  then
10:    for all  $v \in V_r$  do
11:      if  $grado(v, R_G) > grado\_vertice\_max$  then
12:         $grado\_vertice\_max := grado(v, R_G), v_{sel} := v$ 
13:      end if
14:    end for
15:   end if
16:   for all  $r \in R_G$  do
17:     if  $v_{sel} \in V_r$  then
18:        $R_G := R_G - \{r\}$ 
19:     end if
20:   end for
21:    $V_C := V_C \cup \{v_{sel}\}$ 
22: end while
23:  $V'_T := First[V_C], V_C := V_C - V'_T$ 
24: while  $V_C \neq \emptyset$  do
25:    $min\_weight\_path := \infty$ 
26:   for all  $u \in V'_T$  do
27:     for all  $v \in V_C$  do
28:       if  $w(DIJKSTRA(u, v)) < min\_weight\_path$  then
29:          $p_m := DIJKSTRA(u, v)$ 
30:       end if
31:     end for
32:   end for
33:    $V_C := V_C - V(p_m), V'_T := V'_T \cup V(p_m), E'_T := E'_T \cup \{p_m\}$ 
34: end while
35: return  $V'_T, E'_T$ 
```

La Tabla 4.8 concentra la información de los corredores de longitud mínima obtenidos por el Algoritmo 13 que incluye el punto de acceso, vértices del árbol, aristas del árbol y longitud total del corredor.

Tabla 4.8: Corredores de Longitud Mínima.

Punto Acceso	Corredor		
	Vértices	Aristas	Longitud
1	1, 2, 6, 10, 14, 20, 25, 7, 11, 12, 13, 17, 18, 23, 22	{1,2}, {2,6}, {6,10}, {10,14}, {14,20}, {20,25}, {6,7}, {7,11}, {11,12}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	13 500
2	2, 6, 10, 14, 20, 25, 7, 11, 12, 13, 17, 18, 23, 22	{2,6}, {6,10}, {10,14}, {14,20}, {20,25}, {6,7}, {7,11}, {11,12}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	12 500
3	3, 8, 7, 6, 10, 14, 20, 25, 12, 13, 17, 18, 23, 22	{3,8}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {8,12}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	12 500
4	4, 13, 12, 8, 7, 6, 10, 14, 20, 25, 17, 18, 23, 22	{4,13}, {13,12}, {12,8}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {13,17}, {17,18}, {18,23}, {23,22}	13 500
5	5, 4, 13, 12, 8, 7, 6, 10, 14, 20, 25, 17, 18, 23, 22	{5,4}, {4,13}, {13,12}, {12,8}, {8,7}, {7,6}, {6,10}, {10,14}, {14,20}, {20,25}, {13,17}, {17,18}, {18,23}, {23,22}	14 000
18	18, 17, 13, 12, 23, 22, 21, 26, 25, 20, 14, 10, 6	{18,17}, {17,13}, {13,12}, {18,23}, {23,22}, {22,21}, {21,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}	11 500
23	23, 18, 17, 13, 12, 22, 21, 26, 25, 20, 14, 10, 6	{23,18}, {18,17}, {17,13}, {13,12}, {23,22}, {22,21}, {21,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}	11 500
30	30, 23, 18, 17, 13, 12, 22, 21, 26, 25, 20, 14, 10, 6	{30,23}, {23,18}, {18,17}, {17,13}, {13,12}, {23,22}, {22,21}, {21,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}	12 500
29	29, 22, 21, 16, 17, 13, 12, 26, 25, 20, 14, 10, 6	{29,22}, {22,21}, {21,16}, {16,17}, {17,13}, {13,12}, {21,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}	12 000
28	28, 29, 22, 21, 16, 17, 13, 12, 26, 25, 20, 14, 10, 6	{28,29}, {29,22}, {22,21}, {21,16}, {16,17}, {17,13}, {13,12}, {28,26}, {26,25}, {25,20}, {20,14}, {14,10}, {10,6}	12 500
27	27, 24, 25, 26, 21, 22, 16, 17, 13, 12, 20, 14, 10, 6	{27,24}, {24,25}, {25,26}, {26,21}, {21,22}, {21,16}, {16,17}, {17,13}, {13,12}, {25,20}, {20,14}, {14,10}, {10,6}	12 500
24	24, 25, 26, 21, 22, 16, 17, 13, 12, 20, 14, 10, 6	{24,25}, {25,26}, {26,21}, {21,22}, {21,16}, {16,17}, {17,13}, {13,12}, {25,20}, {20,14}, {14,10}, {10,6}	12 000
19	19, 20, 25, 26, 21, 22, 16, 17, 13, 12, 14, 10, 6	{19,20}, {20,25}, {25,26}, {26,21}, {21,22}, {21,16}, {16,17}, {17,13}, {13,12}, {20,14}, {14,10}, {10,6}	12 000
9	9, 10, 14, 20, 25, 6, 7, 11, 12, 13, 17, 18, 23, 22	{9,10}, {10,14}, {14,20}, {20,25}, {10,6}, {6,7}, {7,11}, {11,12}, {12,13}, {13,17}, {17,18}, {18,23}, {23,22}	12 500

Los dos mejores casos corresponden a los puntos de acceso 18 y 23 con una longitud total de 11,500; mientras que el peor caso fue obtenido para el nodo 5 con una longitud total de 14,000.

En la Figura 4.14 se muestran los casos de los corredores con longitud total de 11,500 iniciando a partir de los puntos de acceso 18 y 23, respectivamente. Mientras que en la Figura 4.15 se muestran el caso con mayor longitud del corredor de 14,000.

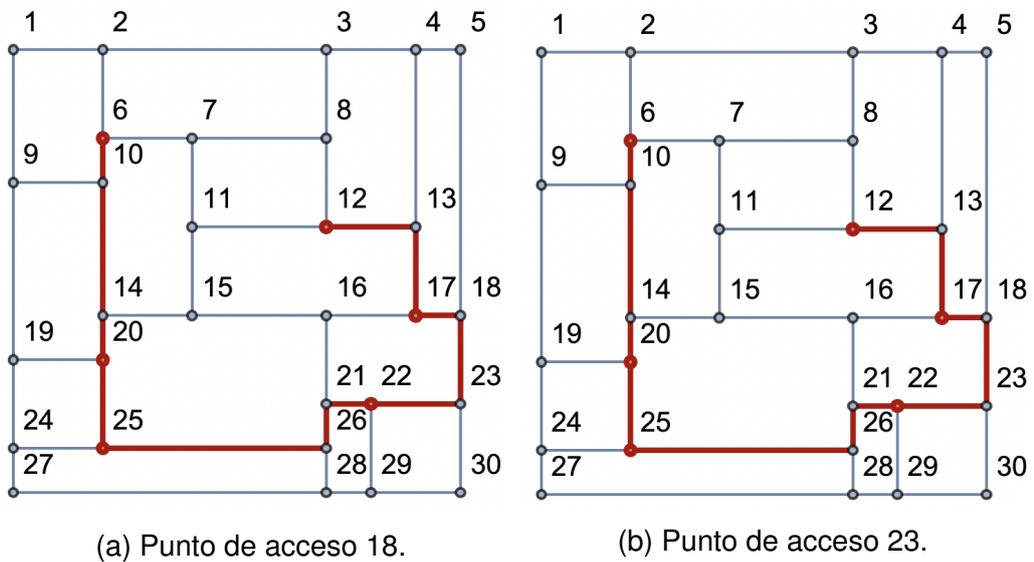


Figura 4.14: Corredor de longitud mínima de 11,500: mejor caso.

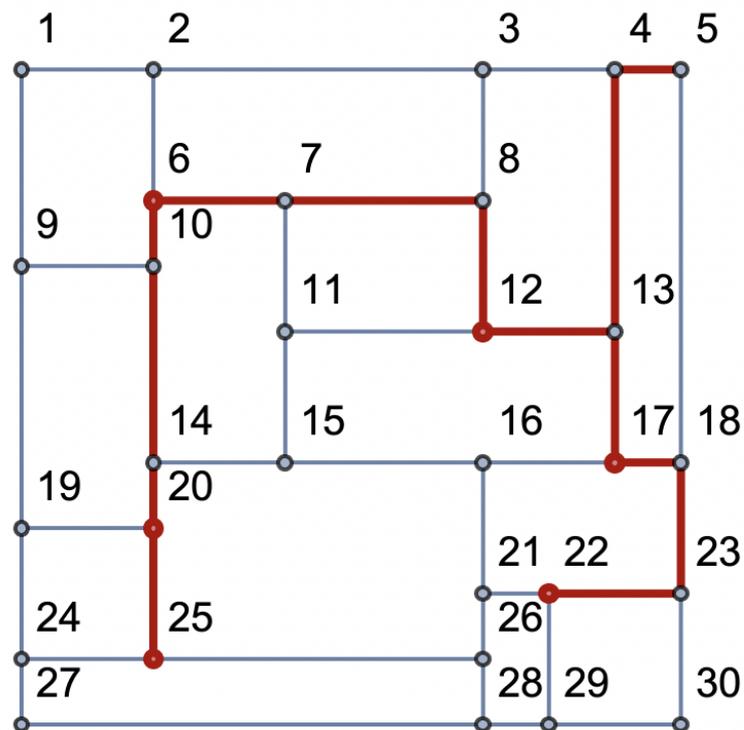


Figura 4.15: Longitud del corredor 14,000 y Punto de acceso 5: peor caso.

Como puede observar al incluir los nodos de la periferia la solución construya el corredor de longitud mínima utilizando algunos de los nodos, por tal motivo se presenta una mejora al algoritmo considerando solamente los vértices internos del grafo en la siguiente sección. La implementación del Algoritmo 15 no se llevo a cabo sobre la familia de dóminos.

4.2.4. Heurística 4: Algoritmo de Vértices Compartidos Mejorado

En esta cuarta solución algorítmica, el árbol se construye con los nodos que tienen el mayor grado de incidencia sobre los rectángulos internos de la instancia, seleccionando la mejor ruta corta para conectarlos construyendo el

árbol solución, hasta alcanzar todos los rectángulos en al menos uno de sus vértices. Los pasos principales serían:

1. Eliminar los vértices que se localizan en la periferia de la instancia geométrica.
2. Ordenar los vértices internos del grafo de acuerdo al grado de incidencia sobre los rectángulos no alcanzados; es decir, cuántos rectángulos comparten el vértice.
3. Seleccionar el vértice interno de mayor grado.
4. Marcar los rectángulos que comparten este vértice como rectángulos alcanzados como se muestra en la Figura 4.16.
5. Mientras no se hayan alcanzado todos los rectángulos de la instancia, regresar al paso 1.
6. Con los dos vértices internos de grado mayor construir la ruta más corta entre los dos vértices.
7. Conectar el vértice interno de mayor grado de incidencia no conectado con el vértice más cercano que pertenece a la solución parcial.
8. Mientras existan vértices internos de mayor grado no conectados, repetir el paso 6.

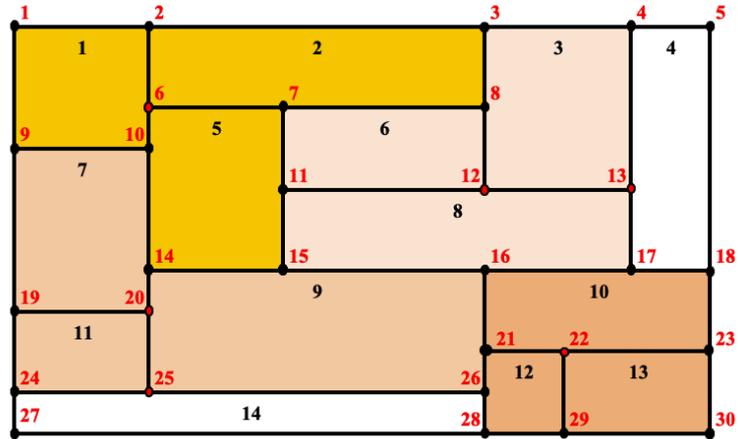


Figura 4.16: Vértices internos con mayor grado de incidencia: vértice 6 incidente sobre los rectángulos 1, 2 y 5; vértice 12 incidente sobre los rectángulos 3, 6 y 8; vértice 20 incidente sobre los rectángulos 7, 9 y 11; vértice 22 incidente sobre los rectángulos 10, 12 y 13; vértice 13 incidente sobre los rectángulos 3, 4 y 8; vértice 25 incidente sobre los rectángulos 9, 11 y 14.

En la Figura 4.17 se puede observar los rectángulos externos que se encuentran en los límites de la instancia geométrica identificados en color verde y etiquetados con los números 1, 2, 3, 4, 7, 10, 11, 12, 13, 14, mientras que los rectángulos internos de la instancia geométrica están identificados en color blanco y etiquetados con los números 5, 6, 8, 9.

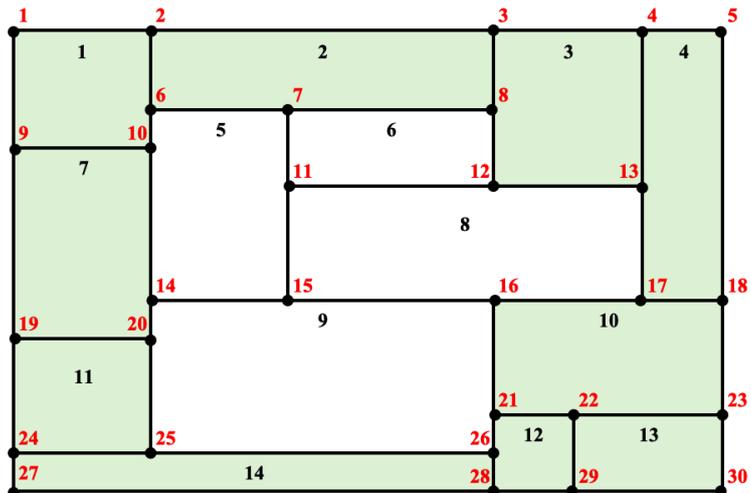
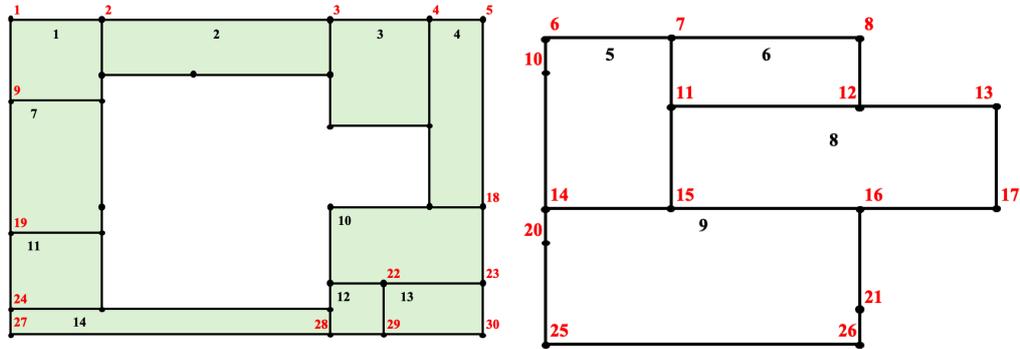


Figura 4.17: Rectángulos externos e internos de la instancia R .

En la Figura 4.18a podemos observar los rectángulos que se encuentran en los límites de la instancia y los vértices de la periferia 1, 2, 3, 4, 5, 9, 18, 19, 23, 24, 27, 28, 29 y 30 que se van a eliminar; mientras que en la Figura 4.18b se puede observar los rectángulos y los vértices internos de la instancia.



(a) Rectángulos externos y vértices de la perifería.

(b) Rectángulos y vértices internos.

Figura 4.18: Instancia geométrica R dividida en rectángulos.

La cuarta propuesta de solución algorítmica al problema MLC es una versión mejorada del Algoritmo 13, se presenta en el Algoritmo 14 la cual consiste en el procedimiento $MLC - H4(V_G, E_G, P_G, root)$. El procedimiento requiere como datos de entrada correspondiente a la instancia geométrica: el conjunto de nodos V_G y aristas E_G del grafo, el conjunto de rectángulos R_G en que se divide la instancia, el nodo raíz del árbol $root$ que serviría de acceso.

Para el Algoritmo 14 se inicializa el conjunto de nodos del árbol solución V'_T con el vértice raíz $root$ (Línea 1 del Algoritmo 14). Se identifica los vértices que se encuentran en la periferia de la instancia geométrica (Líneas 2-11 del Algoritmo 14). La función $lim_inst_geom(r)$ se aplica a todos los rectángulos r del conjunto R_G devolviendo un valor booleano de $False$ si el rectángulo es interno y $True$ si el rectángulo es externo (Línea 3 del Algoritmo 14). La función $VPeriferia(v)$ valida todos los vértices v de los rectángulos externos que se encuentran en

el conjunto V_r ; si el resultado es *False* no es un vértice de la periferia de la instancia geométrica y si es *True* corresponde a un vértice de la periferia de la instancia geométrica, y se agrega al conjunto V_P de los vértices de la periferia de la instancia geométrica (Líneas 5-6 del Algoritmo 14).

Algoritmo 14 Algoritmo de Vértices Compartidos Mejorado.

procedure *MLC* – $H4(V_G, E_G, R_G, root)$

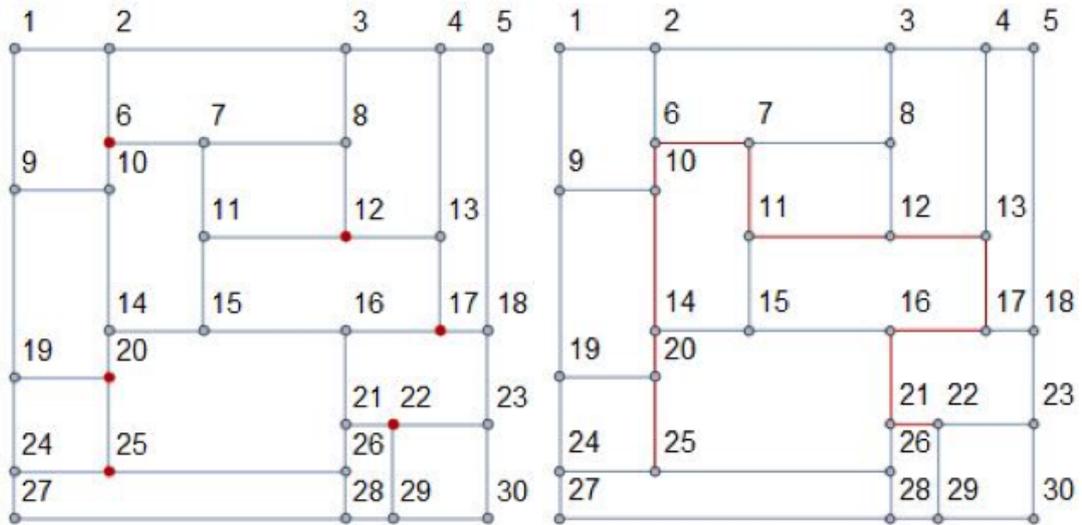
```
1:  $V'_T := \{root\}$ ,  $E'_T := NULL$ ,  $V_C := NULL$ ,  $V_P := NULL$ 
2: for all  $r \in R_G$  do
3:   if  $lim\_inst\_geom(r)$  then
4:      $V_r := \{v \mid v \text{ nodo de } r\}$ 
5:     for all  $v \in V_r$  do
6:       if  $VPeriferia(v)$  then
7:          $V_P := V_P \cup \{v\}$ 
8:       end if
9:     end for
10:  end if
11: end for
12: while  $R_G \neq \emptyset$  do
13:    $grado\_vertice\_max := 0$ ,  $r := First[R_G]$ ,  $V_r := \{v \mid v \text{ nodo periferia de } r\}$ 
14:   for all  $v \in V_r$  do
15:     if  $(v \notin V_P) \wedge (grado(v, R_G) > grado\_vertice\_max)$  then
16:        $grado\_vertice\_max := grado(v, R_G)$ ,  $v_{sel} := v$ 
17:     end if
18:   end for
19:   if  $grado\_vertice\_max = 1$  then
20:     for all  $u \in V_r$  do
21:       if  $grado(u, R_G) > grado\_vertice\_max$  then
22:          $grado\_vertice\_max := grado(u, R_G)$ ,  $v_{sel} := u$ 
23:       end if
24:     end for
25:   end if
26:   for all  $r \in R_G$  do
27:     if  $v_{sel} \in V_r$  then
28:        $R_G := R_G - \{r\}$ 
29:     end if
30:   end for
31:    $V_C := V_C \cup \{v_{sel}\}$ 
32: end while
33: while  $V_C \neq \emptyset$  do
34:    $min\_weight\_path := \infty$ 
35:   for all  $u \in V'_T$  do
36:     for all  $v \in V_C$  do
37:       if  $w(DIJKSTRA(u, v)) < min\_weight\_path$  then
38:          $p_m := DIJKSTRA(u, v)$ 
39:       end if
40:     end for
41:   end for
42:    $V_C := V_C - V(p_m)$ ,  $V'_T := V'_T \cup V(p_m)$ ,  $E'_T := E'_T \cup \{p_m\}$ 
43: end while
44: return  $V'_T, E'_T$ 
```

La Tabla 4.9 concentra la información del árbol extendido de costo mínimo obtenido por el Algoritmo 14 que incluye los vértices del árbol, las aristas del árbol y la longitud total del árbol extendido de costo mínimo interno de la instancia.

Tabla 4.9: Información del Árbol Extendido de Costo Mínimo Modificado.

Vértices	Aristas	Longitud
6, 12, 17, 20, 22, 25	{6, 7}, {7, 11}, {11, 12}, {12, 13}, {13, 17}, {17, 16}, {16, 21}, {16, 22}, {6,10}, {10, 14}, {14, 20}, {20, 25}	11 500

En la Figura 4.19a, se muestran los vértices (en rojo) con el mayor grado de incidencia $grad(v) = 3$. En la Figura 4.19b, se presenta el árbol interno correspondiente a la instancia geométrica, y que constituye un árbol extendido de costo mínimo del subgrafo.



(a) Nodos Compartidos entre rectángulos. (b) Árbol de Longitud Mínima de Nodos Compartidos.

Figura 4.19: Vértices y Aristas del Árbol Interno.

En el Tabla 4.10 se presenta los resultados para todos los casos de vértices de acceso que se encuentran en el perímetro de la instancia geométrica. Se puede observar que la trayectoria para los vértices compartidos es común a

todos los corredores de longitud mínima con una longitud de 11,500; a esta trayectoria se le agrega la ruta corta entre el vértice de acceso y el nodo más cercano que conforman el árbol extendido, así mismo a la longitud del árbol se le adiciona la longitud de las aristas adicionales de los vértices de acceso para determinar la totalidad de la longitud del árbol extendido de costo mínimo modificado.

Tabla 4.10: Información del Árbol Extendido de Costo Mínimo.

Árbol interno de la instancia		Elementos adicionales al árbol interno			Longitud
Aristas	Peso	Punto acceso	Arista	Peso	total
{6,10}, {10, 14}, {14, 20}, {20, 25}, {6, 7}, {7, 11}, {11, 12}, {12, 13}, {13, 17}, {22, 21}, {21, 16}, {16, 17}	11 500	1	{1, 2}, {2, 6}	2 000	13 500
		2	{2, 6}	1 000	12 500
		3	{3, 8}, {8, 12}	2 000	13 500
		4	{4, 13}	2 000	13 500
		5	{5, 4}, {4, 13}	2 500	14 000
		9	{9, 10}	1 000	12 500
		18	{18, 17}	500	12 000
		19	{19, 20}	1 000	12 500
		23	{23, 22}	1 000	12 500
		24	{24, 25}	1 000	12 500
		27	{27, 24}, {24, 25}	1 500	13 000
		28	{28, 26}, {26, 21}	1 000	12 500
		29	{29, 22}	1 000	12 500
		30	{30, 23}, {23, 22}	2 000	13 500

Los resultados obtenidos a través del algoritmo en la generación del corredor de longitud mínima proporciono una solución como el mejor caso teniendo una longitud de 12,000 teniendo como vértice de acceso el nodo 18; mientras que el peor caso también fue solamente un resultado con una longitud de 14,000 con el vértice de acceso en el nodo 5. En las Figuras 4.20a y 4.20b se muestran los vértices compartidos requeridos y el corredor del mejor caso de solución respectivamente.

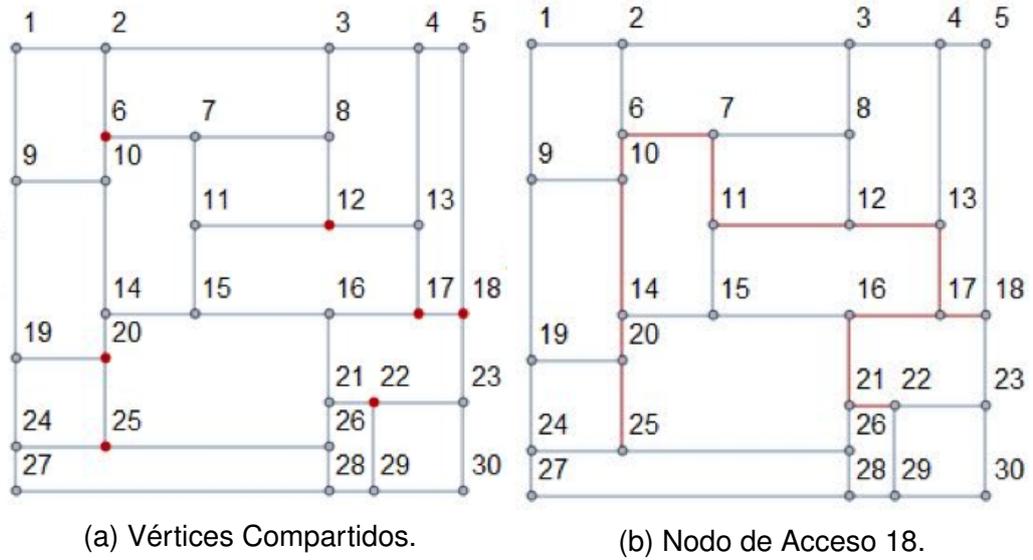


Figura 4.20: Mejor caso del Corredor de Longitud Mínima.

En las Figuras 4.21a y 4.21b se muestra el conjunto de vértices compartidos requeridos y el corredor de longitud mínima correspondiente a la peor solución obtenida por el algoritmo.

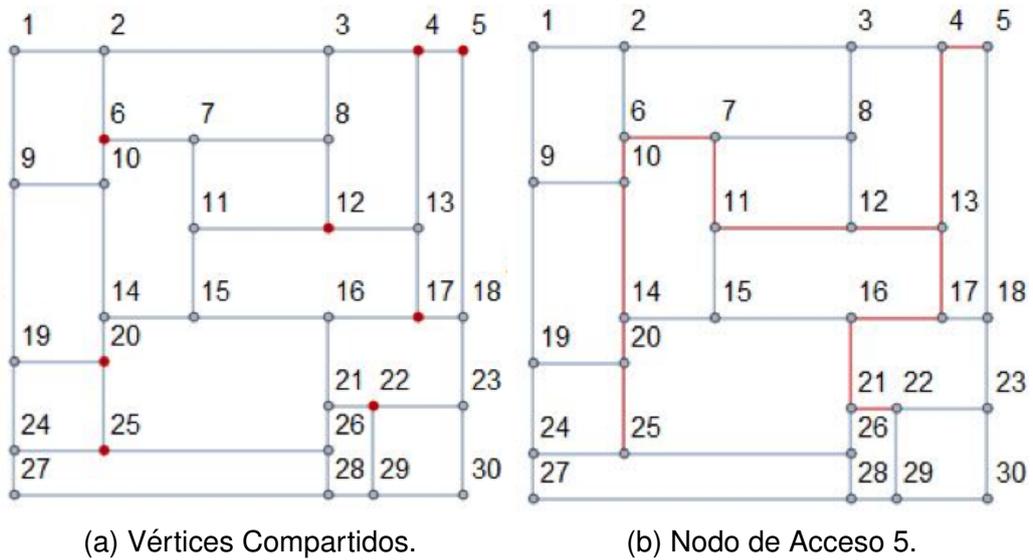


Figura 4.21: Peor caso del Corredor de Longitud Mínima.

La heurística implementada corresponde a un algoritmo de aproximación,

obteniendo las siguientes longitud derivado de la suma de los pesos de las aristas del árbol extendido de costo mínimo modificado: (a) 1 de 12 000, (b) 7 de 12 500, (c) 1 de 13 000, (d) 4 de 13 500 y (e) 1 de 14 000.

La heurística implementada corresponde a un algoritmo que da soluciones casi óptimas. Para nuestro caso ejemplo, se han obtenido soluciones que van desde 12 000 y hasta 14 000, como longitud total del árbol extendido de costo mínimo modificado.

4.2.5. Heurística 5: Algoritmo de Árbol Extendido de Costo Mínimo – Reducción

En esta quinta solución algorítmica, se construye el árbol extendido de costo mínimo considerando la totalidad de vértices del grafo, en donde se van recortando los vértices hoja y sus aristas del árbol considerando el siguiente criterio para todos los vértices hojas del árbol solución mientras sea posible: si los rectángulos que son alcanzados por el vértice hoja del árbol solución son también alcanzados por otros vértices del árbol solución entonces se puede eliminar el vértice hoja y la arista correspondiente que la conecta al árbol. Los pasos principales son:

1. Construir el árbol extendido de costo mínimos del grafo que representa la instancia geométrica usando el algoritmo de *KRUSKAL* como se muestra en la Figura 4.22
2. Ordenar los vértices hojas no verificados de manera ascendente.
3. Si el rectángulo que contiene el primer vértice hoja tiene otro vértice que lo conecte al árbol solución entonces eliminar el vértice hoja y la arista del

vértice hoja, como se muestra en la Figura 4.23.

4. Mientras existan vértices hoja sin validar regresar al paso 2.

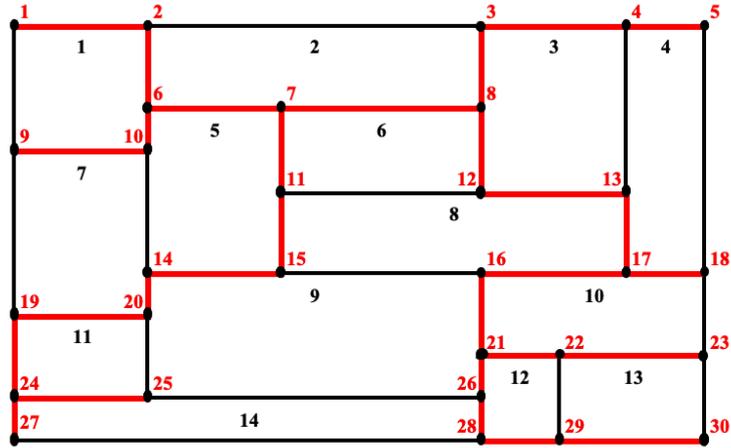


Figura 4.22: Árbol extendido de costo mínimo de la instancia geométrica.

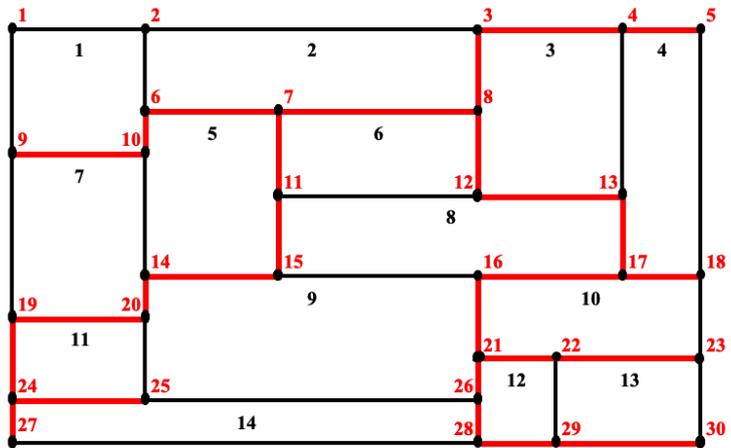


Figura 4.23: Eliminación del vértice hoja 1 y su arista $\{1, 2\}$, seguido por el vértice hoja 2 y su arista $\{2, 6\}$ del árbol extendido de costo mínimo de la instancia geométrica.

La solución propuesta para el problema MLC se presenta en el Algoritmo 15 el cual consiste en el procedimiento $MLC - H5(V_G, E_G, w_G)$. El procedimiento requiere como datos de entrada correspondiente a la instancia geométrica: el conjunto de vértices V_G , aristas E_G y pesos de las aristas w_G del grafo G .

Los conjuntos V'_T , E'_T y w'_T de nodos, aristas y pesos, respectivamente, del árbol extendido de costo mínimo, se inicializan con el resultado del algoritmo de $KRUSKAL(V_G, E_G, w_G)$ (Línea 1 del Algoritmo 15). Se inicializan las variables *eliminar_arista* con *True* para identificar cuando un nodo y arista se han removidos del árbol; así mismo se hace una copia de los vértices, aristas y pesos del árbol solución en *vertices*, *aristas*, *peso* (Líneas 2-3 del Algoritmo 15).

La eliminación de vértices hoja y aristas se realizarán mientras exista la posibilidad de contar con un árbol solución factible al problema MLC (Líneas 4-23 del Algoritmo 15). Primeramente se inicializa el conjunto *hojas* con *NULL* y se procede a realiza una prueba a todos los vértices del árbol con la función *hojaQ(v)*, la función devuelve un valor booleano, si el resultado es *True* significa que el vértice del árbol extendido T es una hoja y se agrega el vértice v al conjunto *hojas* (Líneas 5-10 del Algoritmo 15). Posteriormente los vértices hoja del árbol se ordenan de acuerdo al peso de la arista de mayor a menor con la función *sortW* (Línea 11 del Algoritmo 15).

La siguiente etapa del algoritmo consiste en verificar si el árbol, con la eliminación del vértice hoja y su arista, todavía corresponde a una solución factible. La función *findE(v)* devuelve la arista que conecta al vértice hoja v con el árbol solución parcial (Línea 13 del Algoritmo 15. La función *instanciaMLC* reporta si el grafo definido por $vertices - \{v\}, aristas - \{e\}, peso - \{w(e)\}$, alcanza o no a todos los rectángulos de la partición. En caso de que el valor retornado por la función sea *True* se procede a actualizar los conjuntos *vertices*, *aristas*, *peso* eliminando el nodo hoja v , la arista e que conecta al vértice hoja y el peso de la arista $w(e)$ del árbol, respectivamente; y la variable *eliminar_arista* toma el valor de *True*, saliendo del ciclo *For* para actualizar nuevamente el conjunto de nodos hojas del nuevo árbol y regresando a la Línea 4 del Algoritmo 15. En ca-

so de que el valor retornado por la función sea *False*, se le asigna a la variable *eliminar_arista* un valor de *False*, permaneciendo en el ciclo *For*. Si no se pudiera eliminar ningún nodo del conjunto *hojas* terminara la ejecución del ciclo *For* y *While* (Líneas 12-22 del Algoritmo 15). El árbol solución definido por los conjuntos de vértices V'_T , aristas E'_T y pesos w'_T del árbol se actualizan con los conjuntos *vertices*, *aristas* y *pesos*, respectivamente (Línea 23 del Algoritmo 15). El algoritmo regresa la información del árbol extendido de costo mínimo como solución del problema MLC en los conjuntos de vértices V'_T , aristas E'_T y pesos w'_T (Línea 24 del Algoritmo 15).

Algoritmo 15 Algoritmo de Árbol Extendido de Costo Mínimo – Reducción.

procedure *MLC* – $H5(V_G, E_G, w_G)$

```

1:  $(V'_T, E'_T, w'_T) := KRUSKAL(V_G, E_G, w_G)$ 
2: eliminar_arista := True
3: vertices :=  $V'_T$ , aristas :=  $E'_T$ , peso :=  $w'_T$ 
4: while eliminar_arista do
5:   hojas := NULL
6:   for all  $v \in \textit{vertices}$  do
7:     if hojaQ( $v$ ) then
8:       hojas := hojas  $\cup$   $\{v\}$ 
9:     end if
10:  end for
11:  hojas := sortW(hojas)
12:  for all  $v \in \textit{hojas}$  do
13:     $e := \textit{findE}(v)$ 
14:    if instanciaMLC(vertices –  $\{v\}$ , aristas –  $\{e\}$ , peso –  $\{w(e)\}$ ) then
15:      vertices := vertices –  $\{v\}$ , aristas := aristas –  $\{e\}$ 
16:      peso := peso –  $\{w(e)\}$ , eliminar_arista := True
17:      BREAK
18:    else
19:      eliminar_arista := False
20:    end if
21:  end for
22: end while
23:  $V'_T := \textit{vertices}$ ,  $E'_T := \textit{aristas}$ ,  $w'_T := \textit{peso}$ 
24: return  $V'_T, E'_T, w'_T$ 

```

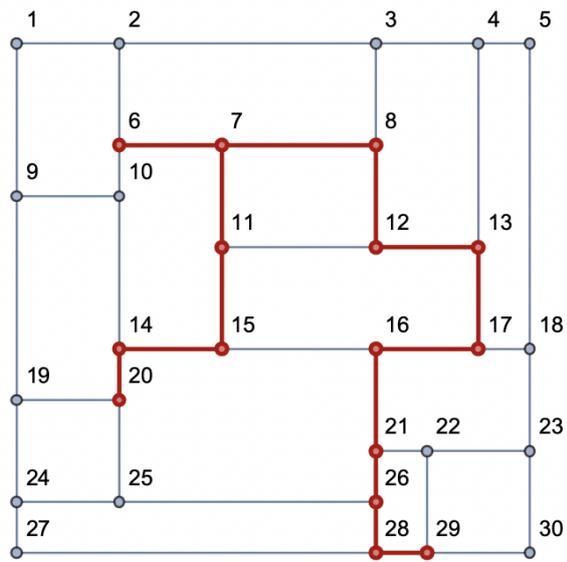
La Tabla 4.11 concentra la información de los corredores de longitud mínima obtenidos por el Algoritmo 15 que incluye el punto de acceso, vértices del árbol, aristas del árbol y longitud total del corredor.

Tabla 4.11: Corredores de Longitud Mínima.

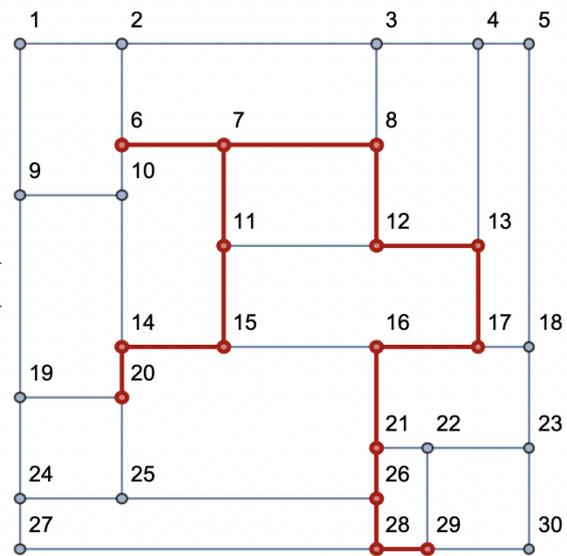
Punto Acceso	Corredor		
	Vértices	Aristas	Longitud
1	1, 2, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{1,2}, {2,6}, {6,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,11}, {11,15}, {15,14}, {14,20}	14 500
2	2, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{2,6}, {6,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,11}, {11,15}, {15,14}, {14,20}	13 500
3	3, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{3,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	13 500
4	4, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{4,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	14 500
5	4, 5, 6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{5,4}, {4,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	15 000
18	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 26, 28, 29	{18,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {17,13}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	13 000
23	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 23, 26, 28, 29	{23,22}, {22,21}, {21,26}, {26,28}, {28,29}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	14 000
30	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29, 30	{30,29}, {29,28}, {28,26}, {26,21}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	13 500
29	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{29,28}, {28,26}, {26,21}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	12 500
28	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{28,28}, {28,26}, {26,21}, {21,16}, {16,17}, {17,13}, {13,12}, {12,8}, {8,7}, {7,11}, {11,15}, {15,14}, {14,20}, {7,6}	12 500
27	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 24, 25, 26, 27, 28, 29	{27,24}, {24,25}, {25,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,6}	15 000
24	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 24, 25, 26, 28, 29	{24,25}, {25,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,6}	14 500
19	6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 25, 26, 28, 29	{19,20}, {20,14}, {14,15}, {15,11}, {11,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,6}	13 500
9	6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 26, 28, 29	{9,10}, {10,6}, {6,7}, {7,8}, {8,12}, {12,13}, {13,17}, {17,16}, {16,21}, {21,26}, {26,28}, {28,29}, {7,11}, {11,15}, {15,14}, {14,20}	14 000

Los dos mejores casos corresponden a los puntos de acceso 28 y 29 con una longitud total de 12,500; mientras que los dos peores casos fueron obtenidos para los nodos de acceso 5 y 27 con una longitud total de 15,000.

En la Figura 4.24 se muestran los casos de los corredores con longitud total de 12,500 iniciando a partir de los puntos de acceso 28 y 29, respectivamente. Mientras que en la Figura 4.25 se muestran los casos con mayor longitud total de 15,000 iniciando a partir de los puntos de acceso 5 y 27, respectivamente.

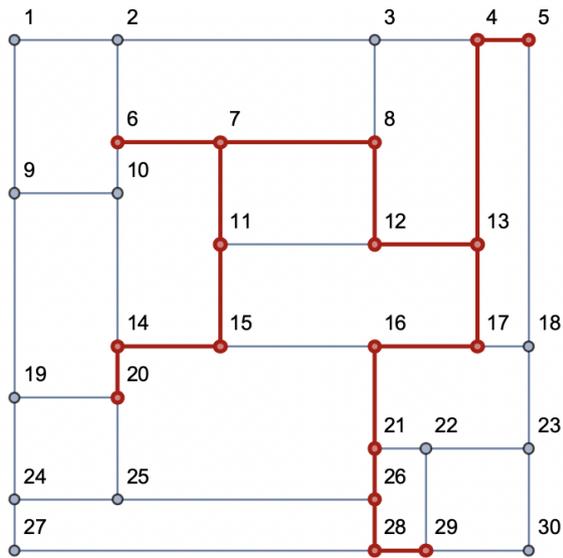


(a) Punto de acceso 28.

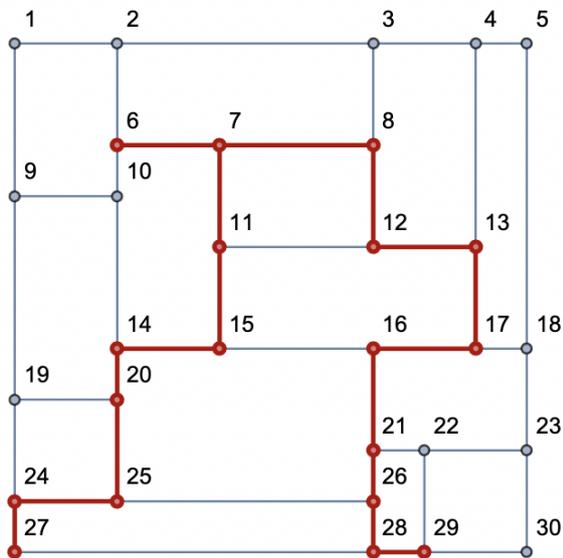


(b) Punto de acceso 29.

Figura 4.24: Corredor de longitud mínima de 12,500: mejor caso.



(a) Punto de acceso 5.



(b) Punto de acceso 27.

Figura 4.25: Corredor de longitud mínima de 15,000: peor caso.

Al considerar la totalidad de vértices de la instancia geométrica para la construcción del árbol extendido de costo mínimo en el proceso de recortar el árbol la mayoría de los nodos de la periferia son eliminados, por tal motivo se presenta en la siguiente sección una mejora al algoritmo considerando solamente los vértices internos del grafo para la construcción y recorte sobre el árbol

extendido de costo mínimo del subgrafo de la instancia geométrica. La implementación del Algoritmo 15 no se llevo a cabo sobre la familia de d6minos.

4.2.6. Heurística 6: Algoritmo de rbol Extendido de Costo Mnimo con Vrtices Internos – Reducci3n

En esta sexta soluci3n algortmica se excluyen los vrtices de la periferia y las aristas que incidan sobre estos vrtices, y se construye el rbol extendido de costo mnimo del subgrafo interno. Se construye el rbol soluci3n mediante el recorte de los vrtices hojas y sus aristas del rbol extendido de costo mnimo del subgrafo, eliminando los vrtices hoja mientras el rbol soluci3n alcance todos los rectngulos de la instancia geométrica. Los pasos principales son:

1. Eliminar todos los vrtices que se localizan en la periferia de la instancia geométrica.
2. Eliminar todas las aristas que incidan en los vrtices eliminados de la periferia.
3. Construir el rbol extendido de costo mnimos con los vrtices internos del subgrafo usando el algoritmo de Kruskal como se muestra en la Figura 4.26.
4. Ordenar los vrtices hojas de manera ascendente.
5. Si el rectngulo que contiene el vrtice hoja tiene otro vrtice que lo conecte al rbol soluci3n entonces eliminar el vrtice hoja y la arista del vrtice hoja, como se muestra en la Figura 4.27.
6. Mientras existan vrtices hoja sin validar regresar al paso 4.

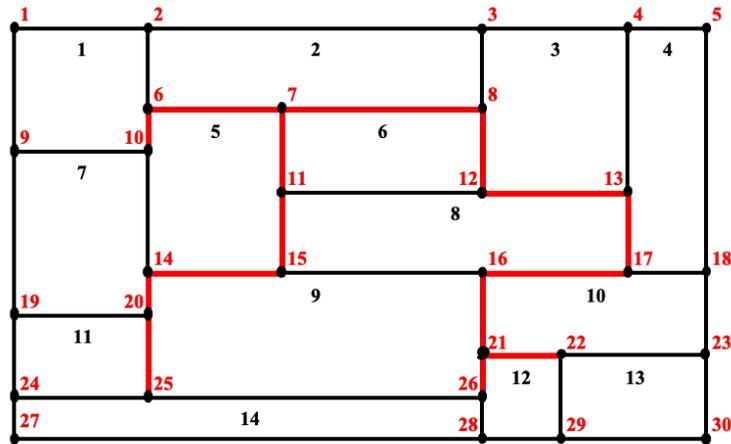


Figura 4.26: Árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.

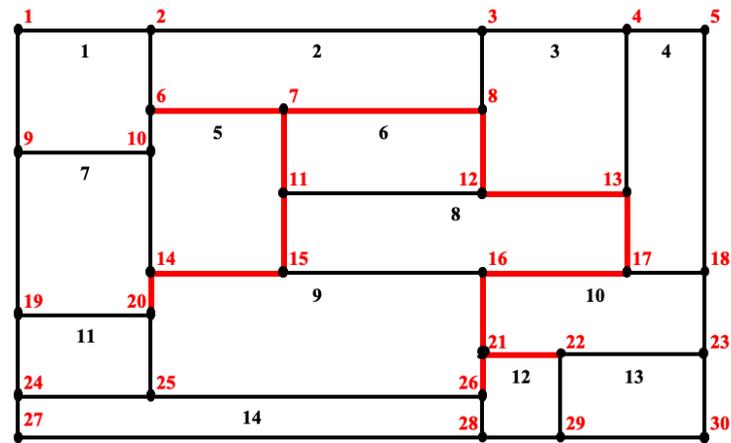


Figura 4.27: Eliminación del vértice hoja 10 y su aristas $\{\{10,6\}\}$, seguido por el vértice hoja 25 y su arista $\{25,20\}$ del árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.

La solución propuesta al problema MLC se presenta en el Algoritmo 16, el cual consiste en el procedimiento $MLC - H6(V_G, E_G, w_G, P_G)$. El procedimiento requiere como datos de entrada correspondiente a la instancia geométrica: el conjunto V_G de vértices, el conjunto E_G de aristas, los pesos de las aristas w_G del grafo y el conjunto P_G de rectángulos.

El Algoritmo 16 es una versión mejorada del Algoritmo 15, al que se le han agregado una líneas previas de procesamiento para eliminar de la instancia los vértices de la periferia y las aristas de las que ellos penden (Líneas 1-17 del Algoritmo 16). Se realiza una copia de los conjuntos V_G , E_G , w_G , R_G en los conjuntos V'_G , E'_G , w'_G y R'_G de vértices, aristas, pesos del grafo y rectángulos de la instancia geométrica (Línea 1 del Algoritmo 16).

Para eliminar los vértices de la periférica de la instancia geométrica y las aristas que conectan estos vértices con otros vértices del grafo se utilizando dos funciones. La primer función $lim_inst_geom(r)$ (Línea 3 del Algoritmo 16) devuelve el valor booleano *True* si el rectángulo r es externo; es decir, r se encuentra en la frontera. En otro caso, devuelve el valor *False*, lo cual significa que el rectángulo es interno. La segunda función $periferia_inst(v)$ (Línea 6 del Algoritmo 16) devuelve un valor booleano *True* si el vértice v se encuentra en la periferia de la instancia geométrica, mientras que devuelve *False* si es un vértice interno. Todos los rectángulos r del conjunto R_G que sean rectángulos externos y que sus vértices u sean parte de los vértices de la periferia de la instancia geométrica se eliminan del conjunto V'_G de vértices del grafo, las aristas (u, v) que conectan el vértice u con cualquier vértice v del conjunto V'_G de vértices del grafo se eliminan del conjunto E'_G de aristas del grafo, así como el peso de las aristas $w(u, v)$ se eliminaran del conjunto w'_G de pesos de las aristas del grafo (Líneas 2-17 del Algoritmo 16). Considerando los nuevos conjuntos V'_G , E'_G y w'_G de vértices, de aristas y de los pesos de las aristas del grafo, se ejecuta el algoritmo de *KRUSKAL* para la generación del árbol extendido de costo mínimo de los nodos internos del subgrafo de la instancia geométrica (Línea 18 del Algoritmo 16).

Algoritmo 16 Algoritmo de Árbol Extendido de Costo Mínimo con Vértices Inter-
nos – Reducción.

procedure *MLC* – $H6(V_G, E_G, w_G, R_G)$

```
1:  $V'_G := V_G, E'_G := E_G, w'_G := w(G), R'_G := R_G$ 
2: for all  $r \in R'_G$  do
3:   if lim_inst_geom( $r$ ) then
4:      $V_r := \{u \mid u \text{ nodo de } r\}$ 
5:     for all  $u \in V_r$  do
6:       if periferia_inst( $u$ ) then
7:          $V'_G := V'_G - \{v\}$ 
8:         for all  $v \in V'_G$  do
9:           if  $\{u, v\} \in E'_G$  then
10:             $E'_G := E'_G - \{u, v\}, w'_G := w'_G - w(\{u, v\})$ 
11:          end if
12:        end for
13:      end if
14:    end for
15:     $R'_G := R'_G - \{r\}$ 
16:  end if
17: end for
18:  $(V'_T, E'_T, w'_T) := KRUSKAL(V'_G, E'_G, w'_G)$ 
19: eliminar_arista := True
20: vertices :=  $V'_T$ , aristas :=  $E'_T$ , peso :=  $w'_T$ 
21: while eliminar_arista do
22:   hojas := NULL
23:   for all  $v \in \text{vertices}$  do
24:     if hojaQ( $v$ ) then
25:       hojas := hojas  $\cup \{v\}$ 
26:     end if
27:   end for
28:   hojas := sortW(hojas)
29:   for all  $v \in \text{hojas}$  do
30:      $e := \text{findE}(v)$ 
31:     if instanciaMLC(vertices –  $\{v\}$ , aristas –  $\{e\}$ , peso –  $\{w(e)\}$ ) then
32:       vertices := vertices –  $\{v\}$ , aristas := aristas –  $\{e\}$ 
33:       peso := peso –  $\{w(e)\}$ , eliminar_arista := True
34:       BREAK
35:     else
36:       eliminar_arista := False
37:     end if
38:   end for
39: end while
40:  $V'_T := \text{vertices}, E'_T := \text{aristas}, w'_T := \text{peso}$ 
41: return  $V'_T, E'_T, w'_T$ 
```

La Tabla 4.12 contiene la información del árbol extendido de costo mínimo obtenido por el Algoritmo 16 que incluye los vértices del árbol, las aristas del árbol y la longitud del árbol extendido de costo mínimo interno de la instancia.

Tabla 4.12: Información del Árbol Extendido de Costo Mínimo.

Vértices	Aristas	Longitud
6, 7, 8, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22, 26	{6, 7}, {7, 8}, {8, 12}, {12, 13}, {13, 17}, {17, 16}, {16, 21}, {21, 22}, {21, 26}, {7, 11}, {11, 15}, {15, 14}, {14, 20}	12 000

En la Figura 4.28, se muestran los vértices y aristas (en rojo) del árbol extendido de costo mínimo de los nodos internos de la instancia geométrica.

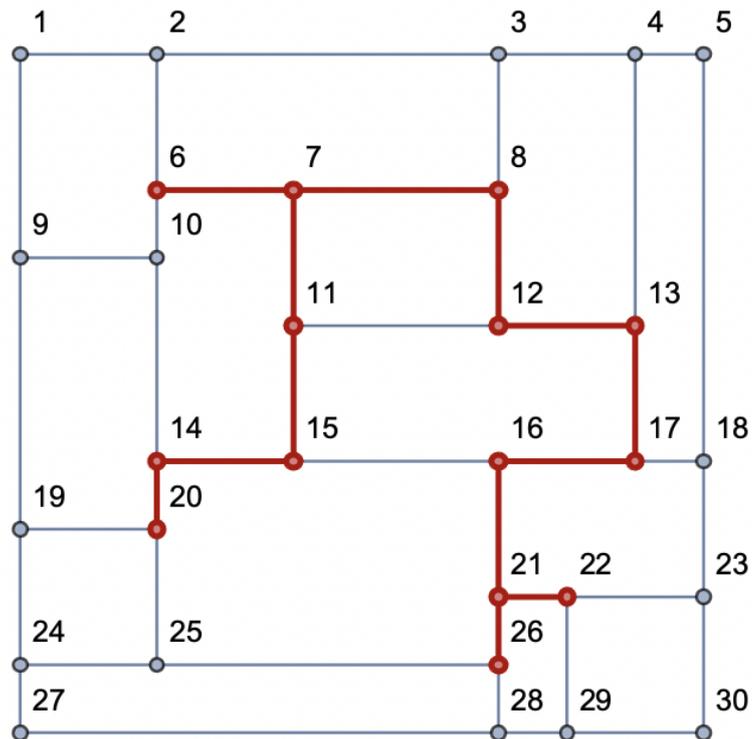


Figura 4.28: Árbol extendido de costo mínimo del subgrafo interno de la instancia geométrica.

En el Tabla 4.13 se presenta los resultados para todos los casos de vérti-

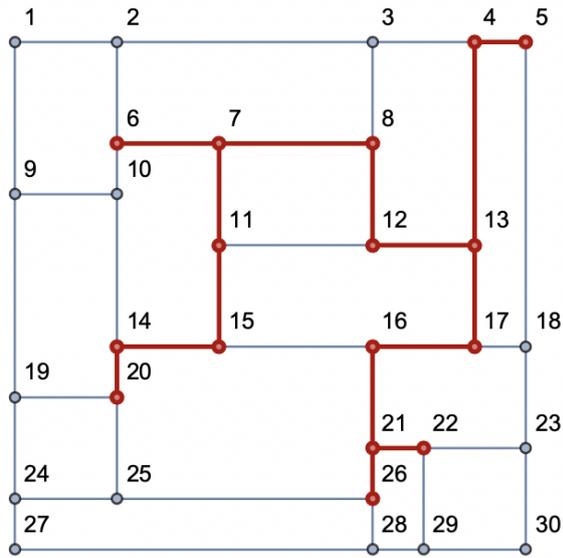
ces de acceso que se encuentran en el perímetro de la instancia geométrica. Se puede observar que el árbol interno de la instancia es común a todos los corredores de longitud mínima con una longitud de 12,000; a esta trayectoria se le agrega la ruta corta entre el vértice de acceso y el nodo más cercado que conforman el árbol extendido, así mismo a la longitud del árbol se le adiciona la longitud de las aristas adicionales de los vértices de acceso para determinar la totalidad de la longitud del árbol extendido de costo mínimo modificado.

Tabla 4.13: Información del Árbol Extendido de Costo Mínimo.

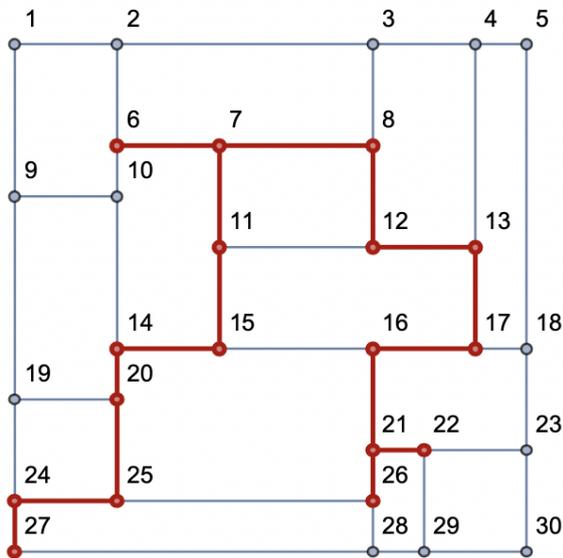
Árbol interno de la instancia		Elementos adicionales al árbol interno			Longitud
Aristas	Peso	Punto acceso	Arista	Peso	total
{6, 7}, {7, 8}, {8, 12}, {12, 13}, {13, 17}, {17, 16}, {16, 21}, {21, 22}, {21, 26}, {7, 11}, {11, 15}, {15, 14}, {14, 20}	12 000	1	{1, 2}, {2, 6}	2 000	14 000
		2	{2, 6}	1 000	13 000
		3	{3, 8}	1 000	13 000
		4	{4, 13}	2 000	14 000
		5	{5, 4}, {4, 13}	2 500	14 500
		9	{9, 10}, {10, 6}	1 500	13 500
		18	{18, 17}	500	12 500
		19	{19, 20}	1 000	13 000
		23	{23, 22}	1 000	13 000
		24	{24, 25}, {25, 20}	2 000	14 000
		27	{27, 24}, {24, 25}, {25, 20}	2 500	14 500
		28	{28, 26}	500	12 500
		29	{29, 22}	1 000	13 000
		30	{30, 23}, {23, 22}	2 000	14 000

Los dos mejores casos corresponden a los puntos de acceso 18 y 28 con una longitud total de 12,500; mientras que los dos peores casos fueron obtenidos para los nodos de acceso 5 y 27 con una longitud total de 14,500.

En la Figura 4.29 se muestran los casos de los corredores con longitud total de 12,500 iniciando a partir de los puntos de acceso 18 y 28, respectivamente. Mientras que en la Figura 4.30 se muestran los casos con mayor longitud total de 14,500 iniciando a partir de los puntos de acceso 5 y 27, respectivamente.



(a) Punto de acceso 5.



(b) Punto de acceso 27.

Figura 4.30: Corredor de longitud mínima de 14,500: peor caso.

En este capítulo se han presentado, como una contribución importante de este trabajo, los algoritmos de ruteo consistentes en seis heurísticas diseñadas de acuerdo a la técnica de programación greedy e implementados en el lenguaje de programación funcional *Mathematica*® para la generación de corredores de longitud mínima. En el siguiente capítulo se presentan los resultados así como

su análisis de los algoritmos de ruteo sobre las instancias consistentes en em-
baldosados no-isomorfos y libres de corte de tamaño 6×5 , 6×7 , 6×8 , 6×9 ,
 6×10 y 7×8 .

”Una de las cosas que debe hacer un científico de la computación es distinguir entre los problemas específicos de la ciencia de la computación y el uso de las computadoras en la sociedad”

Edsger Wybe Dijkstra (1930-2002)
Científico de la computación holandés

Resultados.

Los resultados obtenidos en esta tesis doctoral se encuentran enmarcados en un trabajo de investigación básica y aplicada. Consiste en el análisis experimental de algoritmos en contraste con su análisis asintótico. Los resultados experimentales permitieron también el planteamiento de algoritmos con mejores razones de aproximación. Esto tiene un impacto en la práctica así como en la teoría cuando se trabaja en el modelado y simulación de problemas, de los que se busca determinar su complejidad computacional [Gaylord and Wellin, 1995].

Una variedad de técnicas de diseño de algoritmos se puede encontrar en la literatura, entre ellas las estrategias greedy, divide y vencerás, programación dinámica, programación lineal, entre otras [Hastings, 2006], [Singh, 2015]. En esta tesis se ha usado la técnica greedy para la solución de problemas que por su naturaleza involucra una gran cantidad de información. En particular, aquellas redes que modelan los sistemas de transporte, en las cuales el tamaño de la instancia consiste en grafos con miles de vértices y aristas.

El problema en particular que se estudio en esta investigación refiere al de ruteo en superficies divididas por aristas rectilíneas, donde se busca construir un árbol que conecte cada una de las regiones de manera óptima, es decir, un árbol cuya longitud total sea la mínima. Para esto se diseñaron e implementaron algoritmos para conocer la cantidad de instancias geométricas así como la gene-

ración de cada uno de los embaldosados rectilíneos no-isomorfos sin líneas de corte, con la finalidad de poder llevar a cabo un estudio comparativo de los tiempos de ejecución de los algoritmos desarrollados bajo las diferentes estrategias de diseño mencionadas anteriormente.

Así mismo, se plantea acotar el valor óptimo $f^*(I_D)$ de la longitud del corredor en una instancia geométrica I_D de embaldosados con dominós a través del establecimiento de una cota inferior y una cota superior.

Primeramente, se generan las familias completas de instancias de embaldosados no-isomorfos libres de corte. Enseguida se correo el banco de los diferentes programas, para llevar a cabo el análisis comparativo de los algoritmos correspondientes.

5.1. Embaldosados no isomorfos libres de corte.

El cálculo del determinante ha sido optimizado en muchos sistemas, al punto que la velocidad del cálculo del permanente de una matriz es típicamente 80 veces más lento que el cálculo de su determinante. Por ello, resulta deseable transformar la solución del problema basada en el cálculo del permanente por otra solución basada en el cálculo del determinante.

Por otro lado, el cálculo numérico del determinante de una matriz cuadrada de números reales de tamaño 5000×5000 es bastante rápido, y suele tomar no más de 3 segundos. Este tiempo corresponde aproximadamente al que toma el cálculo del determinante de una matriz cuadrada de números enteros de tamaño 530×530 . Sin embargo, el cómputo simbólico necesario para el cálculo del determinante de la matriz de Kasteleyn con las variables adicionales es sig-

nificativamente lento (Ecuación 3.13). Por ejemplo, para una matriz de tamaño 20×20 toma alrededor de 3 segundos debido a la cantidad de términos involucrados. La generación simbólica similar al ejemplo en Ecuación 3.13, de todos los embaldosados del rectángulo 7×6 , tardó 1.225 minutos.

Los tiempos de ejecución mencionados aquí fueron medidos en una computadora MacBook Pro con un CPU de 2.6 GHz Intel Core i7 con 6 núcleos y 16 GB de memoria RAM, usando el lenguaje de programación de alto nivel ©Mathematica versión 11.3.0.0.

5.1.1. Generando todos los embaldosados.

Todos los embaldosados de un cuadrado de 4×4 generados por los Algoritmos 4 y 5 se muestran en la Figura 5.1. De todos los embaldosados, se puede observar, por ejemplo, que el primero es esencialmente el mismo que el último embaldosado, ya que este último puede ser obtenido a partir del primero aplicándole una operación de rotación de 90 grados en sentido horario o antihorario sobre un eje perpendicular al plano alrededor del centro del cuadrado. Por lo tanto, el conjunto de embaldosados de tamaño 4×4 en la Figura 5.1 incluye embaldosados que son isomorfos bajo operaciones de rotación y/o reflexión. En la siguiente subsección se organiza el conjunto de embaldosados en clases de equivalencia, donde los embaldosados son isomorfos bajo rotaciones y/o reflexiones en la misma clase.

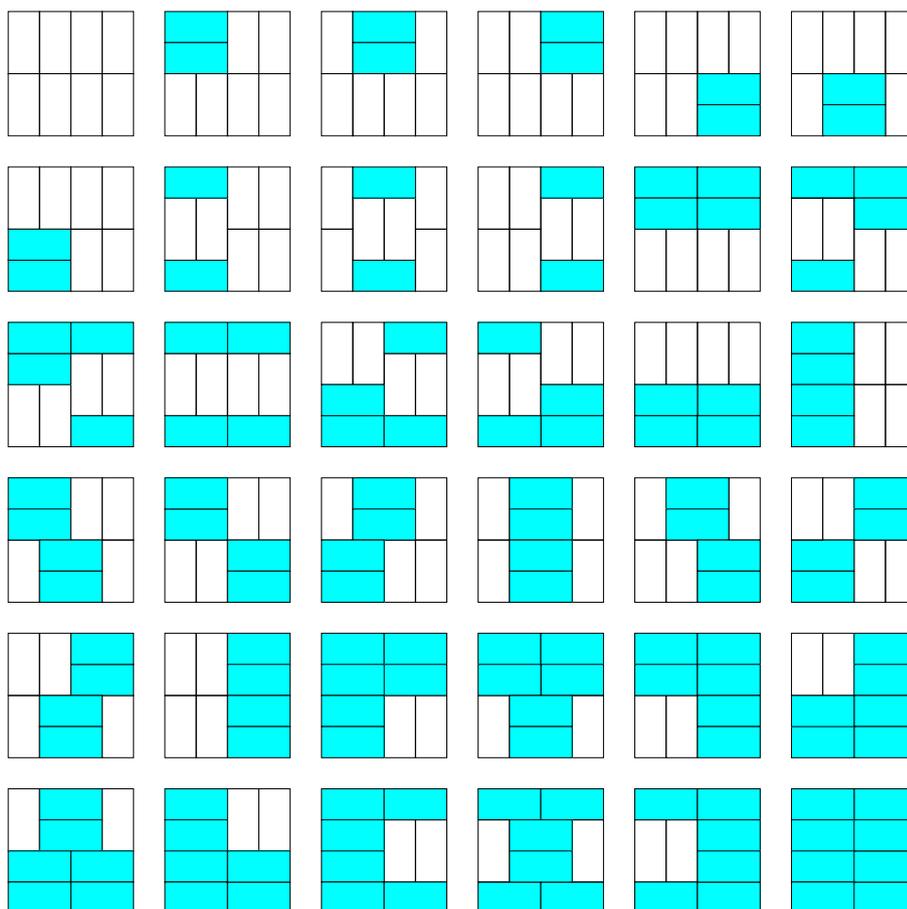


Figura 5.1: Embaldosados del cuadrado de 4×4 .

5.1.2. Generando embaldosados no-isomorfos.

Cuando se aplican las transformaciones descritas en la subsección 3.2.2 sobre el conjunto de embaldosados de tamaño 4×4 de la Figura 5.1, el número de embaldosados se reduce de 36 a 9 embaldosados no-isomorfos o clases de equivalencia. Más precisamente, el número de embaldosados invariantes bajo la acción de cada una de las rotaciones y reflexiones $r_v, r_h, r_{D_1}, r_{D_2}, \pi_1, \pi_2, \pi_3$ y π_0 corresponden a 12, 12, 0, 0, 2, 8, 2 y 36 respectivamente. Entonces, de acuerdo con el Teorema 3.1, el número de embaldosados no-isomorfos es $\frac{12+12+2+8+2+36}{8} = 9$ para un cuadrado 4×4 , los cuales se muestran en la Figura 5.2.

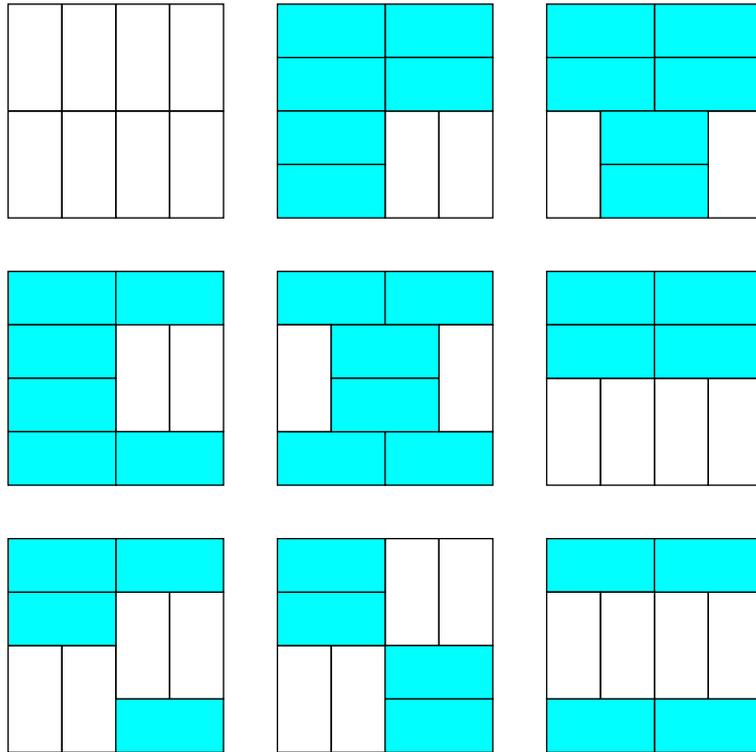


Figura 5.2: Embaldosados no-isomorfos del cuadrado 4×4 .

El número de embaldosados no-isomorfos para $n, m \leq 7$ se muestran en la Tabla 5.1, la cual tomó 30.0931 segundos para calcularla, usando el Algoritmo 6.

Tabla 5.1: Número de embaldosados no-isomorfos para $n, m \leq 7$.

n	m						
	1	2	3	4	5	6	7
1	0	1	0	1	0	1	0
2	1	1	2	4	5	9	12
3	0	2	0	5	0	14	0
4	1	4	5	9	33	98	230
5	0	5	0	33	0	329	0
6	1	9	14	98	329	930	8121
7	0	12	0	230	0	8121	0

5.1.3. Generando embaldosados libres de cortes.

En la Figura 5.3 se muestran los 2 únicos embaldosados libres de líneas de corte de un rectángulo 5×6 , que resultan de la selección de embaldosados, a partir de los embaldosados generados por el Algoritmo 6.

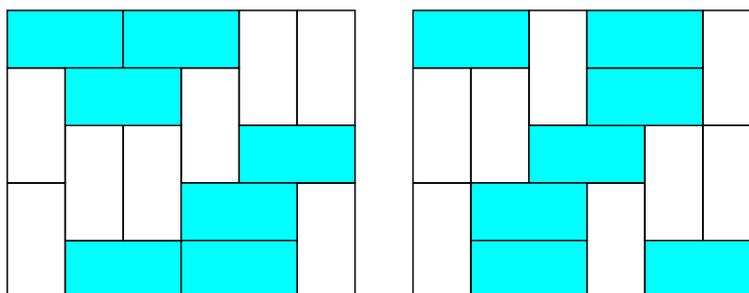


Figura 5.3: Embaldosados sin líneas de corte de un rectángulo 5×6 .

La Tabla 5.2 contiene el resultado de la búsqueda de embaldosados libres de líneas de cortes, la cual tomó 31.2482 segundos para generarla. La Figura 5.4 contiene los 32 embaldosados no-isomorfos libres de líneas de corte de un rectángulo 7×6 . Observe que el número de embaldosados se reduce de 31529 embaldosados en total a 8121 embaldosados no-isomorfos, y de ahí a 32 embaldosados no-isomorfos sin líneas de corte.

Tabla 5.2: Número de Embaldosados libres de líneas de corte.

n	m						
	1	2	3	4	5	6	7
1	0	1	0	0	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	0	0	0	0	0	2	0
6	0	0	0	0	2	0	32
7	0	0	0	0	0	32	0

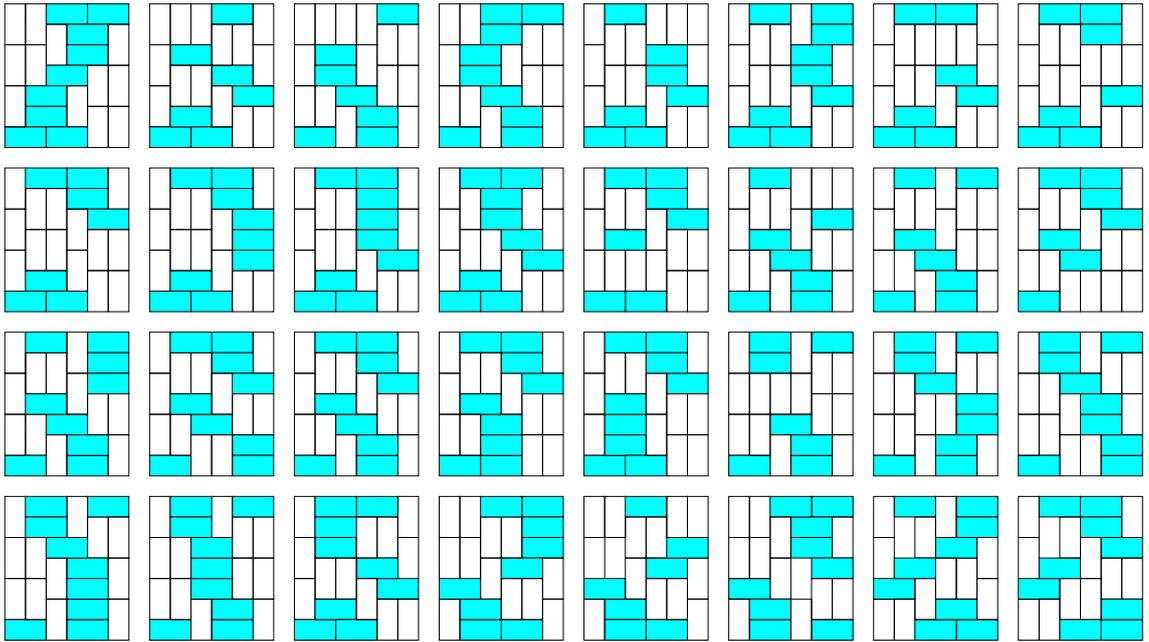


Figura 5.4: Caso rectángulo 7×6 : 32 embaldosados no isomorfos sin líneas de corte.

5.1.4. Embaldosados con una y dos líneas de corte.

Los Algoritmos 9 y 10 pueden ser modificados para reportar aquellas instancias de embaldosados con una o dos líneas de corte.

Existen ocho embaldosados con solo una línea de corte de área menor o igual a 25 que se muestran en la Figura 5.5. En la Tabla 5.3 se muestran la cantidad de embaldosados con una línea de corte en rectángulos con una área menor a 49.

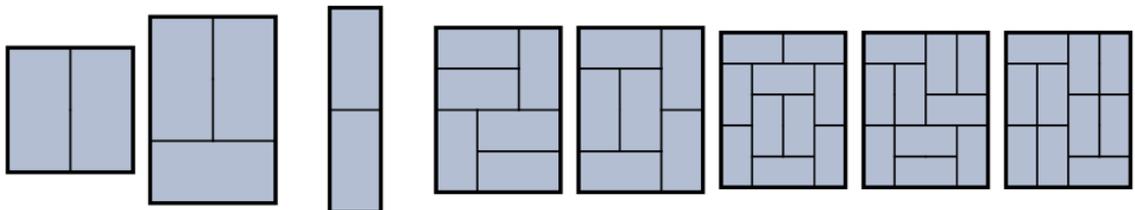
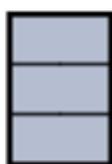


Figura 5.5: Embaldosados con una línea de corte de área menor o igual a 25.

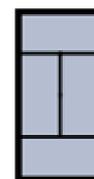
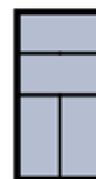
Tabla 5.3: Número de Embaldosados con una línea de corte.

n	m						
	1	2	3	4	5	6	7
1	0	0	0	1	0	0	0
2	0	1	1	0	0	0	0
3	0	1	0	2	0	2	0
4	1	0	2	0	3	0	4
5	0	0	0	3	0	34	0
6	0	0	2	0	34	14	473
7	0	0	0	4	0	473	0

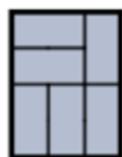
Existen diez embaldosados con dos líneas de corte de área menor o igual a 16, y se muestran en la Figura 5.6. En la Tabla 5.4 se reporta la cantidad de embaldosados con dos líneas de corte de área hasta 49.



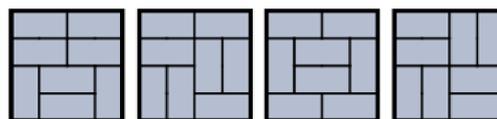
(a) Embaldosado de 3×2 de área 6.



(b) Embaldosados de 4×2 de área 8.



(c) Embaldosados de 4×3 de área 12.



(d) Embaldosados de 4×4 de área 16.

Figura 5.6: Embaldosados con dos líneas de corte de área menor o igual a 16.

Tabla 5.4: Número de Embaldosados con dos líneas de corte.

n	m						
	1	2	3	4	5	6	7
1	0	0	0	0	0	1	0
2	0	0	1	3	2	0	0
3	0	1	0	2	0	6	.0
4	0	3	2	4	12	15	29
5	0	2	0	12	0	105	.0
6	1	0	6	15	105	149	1740
7	0	0	0	29	0	1740	0

5.1.5. Midiendo tiempos de ejecución.

En la Tabla 5.5 se muestran los tiempos de ejecución t_d y t_b para el cálculo del determinante de la matriz de Kasteleyn con variables, y del proceso de backtracking, respectivamente, para la generación y enumeración de embaldosados de ocho instancias de rectángulos $8 \times m$, para $1 \leq m \leq 8$.

En ambos métodos el tiempo de ejecución es exponencial. Sin embargo, la razón t_d/t_b , que denota que tan lento es el cálculo del determinante con respecto al método de backtracking, crece conforme el tamaño de la instancia es mayor. Por ejemplo, para la instancia de tamaño 8×7 el método de backtracking es casi 32 veces más rápido que el del cálculo del determinante de la matriz de Kasteleyn con variables.

El caso del rectángulo 8×8 para el cálculo del determinante no representa dificultad en lo que se refiere a tiempo de ejecución. Sin embargo, no se reporta porque presenta un problema de desbordamiento de memoria.

Tabla 5.5: Tiempo de ejecución (seg).

Tamaño	t_a	t_b	t_a/t_b
8 × 1	0.001596	0.000262	6.09
8 × 2	0.021464	0.003165	6.78
8 × 3	0.162972	0.014392	11.32
8 × 4	2.49997	0.185034	13.51
8 × 5	26.7338	1.17374	22.78
8 × 6	412.918	15.2201	27.13
8 × 7	4,049.34	126.607	31.98
8 × 8	——	1,461.02	——

5.1.6. Selección de instancias geométricas.

Las pruebas experimentales de las heurísticas implementadas se llevan a cabo sobre seis familias de instancias consistentes en embaldosados con dominós no-isomorfos libres de corte. En la Tabla 5.6 se reportan el tamaño de cada una de las familias de instancias, así como la cantidad de embaldosados de cada familia antes y después de ser filtrados. Asimismo se reporta el tiempo requerido en la generación de cada familia de instancias.

Tabla 5.6: Cantidad de embaldosados y tiempo de generación (seg).

Tamaño Instancia	Embaldosados		Tiempo (seg)
	Total	Filtrados	
6 × 5	1 183	2	0.033
6 × 7	31 529	32	0.590
6 × 8	167 089	18	979.357
6 × 9	817 991	418	24 285.600
6 × 10	4 213 133	415	671 181.000
7 × 8	1 292 697	3 391	62 842.300

En el Apéndice B se ofrecen los conjuntos de embaldosados filtrados para

cada familia de instancias representados a través de grafos, los cuales se utilizaron en la evaluación experimental de las heurísticas implementadas. A través de los algoritmos se obtuvo el corredor de longitud mínima. Para cada instancia, se consideran las cuatro esquinas de la instancia geométrica como los puntos de acceso para la construcción del corredor.

5.2. Corredores de longitud mínima.

El problema general consiste en encontrar un corredor de longitud mínima a partir de un vértice de acceso v sobre un rectángulo de tamaño $n \times m$, el cual está dividido en rectángulos pequeños de diferentes tamaños. Se ha demostrado que es un problema NP-completo, y en consecuencia se establece la conjetura que no hay un algoritmo eficiente de tiempo polinomial que genere una solución óptima.

5.2.1. Vértices de Acceso y Cotas para Corredores en Mallas y Dominós.

En esta sección se considera dos posibilidades para la elección de un nodo de acceso. Como se puede observar en la Figura 5.7 correspondiente a un embaldosado con dominós de 6×5 , los vértices localizados sobre el perímetro de la instancia se definirán de la siguiente manera:

1. Vértices en esquina: Son los cuatro vértices localizados en las esquinas de la instancia, los cuales se pueden identificar en la Figura 5.7 por el color rojo.

2. Vértices de borde: Son los vértices localizados en el perímetro de la instancia excepto las esquinas, los cuales se pueden identificar en la Figura 5.7 por el color azul.

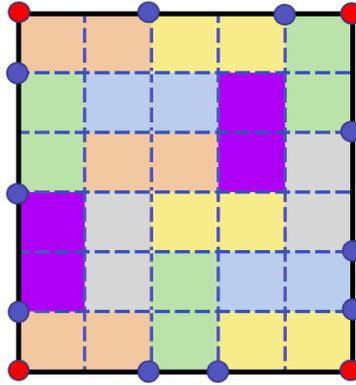


Figura 5.7: Vértices en esquina y de borde en una instancia de tamaño 6×5 de dominós.

Instancias Geométricas de Mallas.

Un caso particular de este problema es una instancia de malla I_M , la cual consiste en un rectángulo de tamaño $n \times m$, donde n y m son números enteros, dividido en cuadrados de 1×1 , como se muestra en la Figura 5.8.

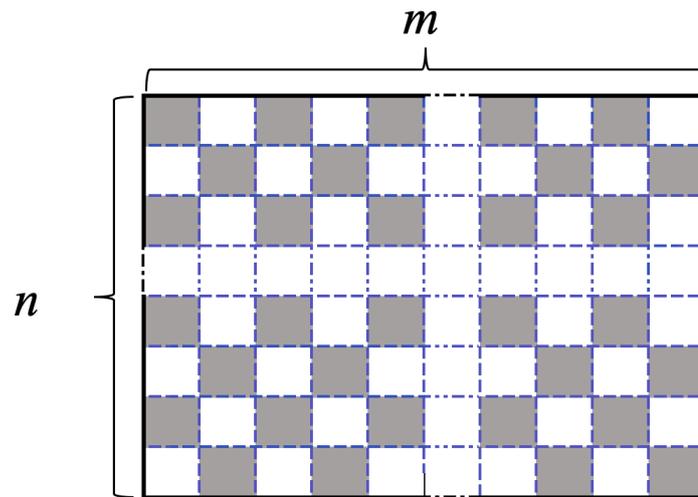


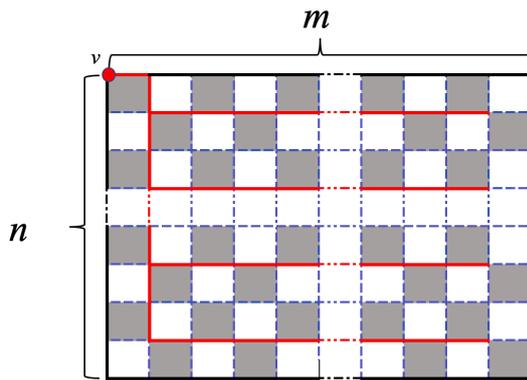
Figura 5.8: Instancia de Malla de tamaño $n \times m$.

Para obtener el corredor de longitud mínima, considere el vértice de la esquina superior izquierda como el nodo de acceso v , que corresponde a la raíz del árbol. Las ramas se extienden de manera vertical y horizontal sobre la malla con la finalidad de construir el árbol solución. Considere los siguientes tres casos de instancias de malla I_M de tamaño $n \times m$:

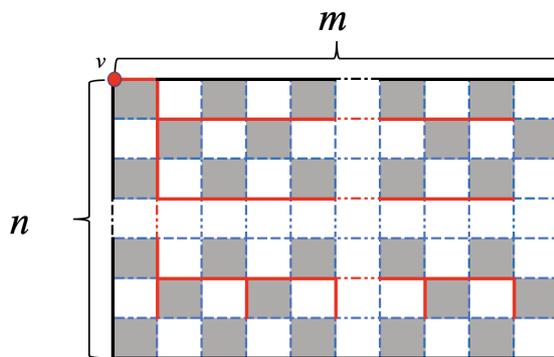
Caso 1: n par, m par o impar.

Caso 2: n impar, m par.

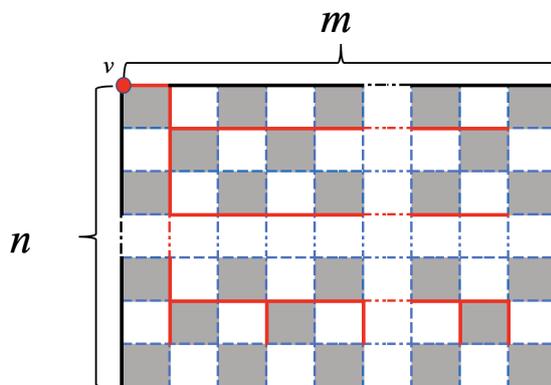
Caso 3: n, m impares.



(a) Caso 1: n par, m par o impar.



(b) Caso 2: n impar, m par.



(c) Caso 3: n, m impares.

Figura 5.9: Árbol solución de la Instancia de Malla I_M de tamaño $n \times m$.

Caso 1: n par, m par o impar.

En este caso, la solución está conformada parcialmente por exactamente $\frac{n}{2}$ segmentos horizontales. El primer segmento horizontal se despliega sobre las aristas horizontales inferiores (horizontales superiores) de los cuadrados que se encuentran en la primera (segunda, respectivamente) fila de la instancia, a partir del vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la segunda columna y hasta el vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la $m - \text{ésima}$ columna. De manera similar y alternada, se despliegan segmentos horizontales que cubren los cuadrados de la tercera y cuarta fila, y así sucesivamente hasta trazar el segmento horizontal que alcanza la filas $n - 1$ y n de cuadrados. La longitud de cada segmento horizontal es $m - 2$. La longitud parcial total de los $\frac{n}{2}$ segmentos horizontales es $\frac{n}{2} \times (m - 2)$. Enseguida se traza un segmento de línea (vertical) a lo largo de las aristas de contacto entre los cuadrados de la primera y segunda columna, desde la parte superior del rectángulo $n \times m$ y hasta la esquina superior derecha (esquina superior izquierda) del último cuadrado de la primera columna (de la segunda columna, respectivamente). Este segmento de línea une todas las ramas horizontales. Finalmente, se traza un segmento de línea de longitud unitaria desde la esquina superior izquierda del rectángulo $n \times m$ hasta alcanzar el segmento de línea vertical. En la Figura 5.9a se presenta el árbol solución óptimo para la malla I_M de tamaño $n \times m$, para n par, m par o impar, donde se puede observar que todos los cuadrados de 1×1 son tocados al menos por una arista del árbol solución. La longitud total de la solución es $\frac{n}{2} \times (m - 2) + n$.

Caso 2: n impar, m par.

En este caso, la solución está conformada parcialmente por exactamente $\lfloor \frac{n}{2} \rfloor$ segmentos horizontales. El primer segmento horizontal se despliega sobre las aristas horizontales inferiores (horizontales superiores) de los cuadrados que se encuentran en la primera (segunda, respectivamente) fila de la instancia, a partir del vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la segunda columna y hasta el vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la $m - \text{ésima}$ columna. De manera similar y alternada, se despliegan segmentos horizontales que cubren los cuadrados de la tercera y cuarta fila, y así sucesivamente hasta trazar el segmento horizontal que alcanza la filas $(n - 2)$ y $(n - 1) - \text{ésima}$ de cuadrados. La longitud de cada segmento horizontal es $m - 2$. La longitud parcial total de los $\frac{n}{2}$ segmentos horizontales es $\lfloor \frac{n}{2} \rfloor \times (m - 2)$. Enseguida se traza un segmento de línea (vertical) a lo largo de las aristas de contacto entre los cuadrados de la primera y segunda columna, desde la parte superior del rectángulo $n \times m$ y hasta la esquina superior derecha (esquina superior izquierda) del último cuadrado de la primera columna (de la segunda columna, respectivamente). Este segmento de línea une todas las ramas horizontales. Finalmente, para alcanzar los $m - 2$ cuadros de la última fila de la instancia (cuadros 3 al m) se trazan $\frac{m-2}{2}$ segmentos de línea verticales de longitud unitaria a partir del último segmento horizontal de la instancia hasta alcanzar la esquina superior derecha (esquina superior izquierda) de los cuadrados de la $n - \text{ésima}$ fila, los cuales serán desplegados para alcanzar el tercer y cuarto cuadrado, luego el quinto y sexto cuadrado, y finalmente el $(m - 1)$ y $m - \text{ésimo}$ cuadrado de la última fila. Finalmente, se traza un segmento de línea de longitud unitaria desde la esquina superior izquierda del rectángulo $n \times m$ hasta alcanzar el segmento de línea

vertical. En la Figura 5.9b se presenta el árbol solución óptimo para la malla I_M de tamaño $n \times m$, para n impar y m par, donde se puede observar que todos los cuadrados de 1×1 son tocados al menos por una arista del árbol solución. La longitud total de la solución es $\lfloor \frac{n}{2} \rfloor \times (m - 2) + n + \frac{m-2}{2}$.

Caso 3: n, m impares.

En este caso, la solución está conformada parcialmente por exactamente $\lfloor \frac{n}{2} \rfloor$ segmentos horizontales. El primer segmento horizontal se despliega sobre las aristas horizontales inferiores (horizontales superiores) de los cuadrados que se encuentran en la primera (segunda, respectivamente) fila de la instancia, a partir del vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la segunda columna y hasta el vértice inferior izquierdo (superior izquierdo) del primero (segundo, respectivamente) cuadrado de la $m - \text{ésima}$ columna. De manera similar y alternada, se despliegan segmentos horizontales que cubren los cuadrados de la tercera y cuarta fila, y así sucesivamente hasta trazar el segmento horizontal que alcanza la filas $(n - 2)$ y $(n - 1) - \text{ésima}$ de cuadrados. La longitud de cada segmento horizontal es $m - 2$. La longitud parcial total de los $\frac{n}{2}$ segmentos horizontales es $\lfloor \frac{n}{2} \rfloor \times (m - 2)$. Enseguida se traza un segmento de línea (vertical) a lo largo de las aristas de contacto entre los cuadrados de la primera y segunda columna, desde la parte superior del rectángulo $n \times m$ y hasta la esquina superior derecha (esquina superior izquierda) del último cuadrado de la primera columna (de la segunda columna, respectivamente). Este segmento de línea une todas las ramas horizontales. Finalmente, para alcanzar los $m - 2$ cuadros de la última fila de la instancia (cuadros 3 al m) se trazan $\lceil \frac{m-2}{2} \rceil$ segmentos de línea verticales de longitud unitaria a partir del último segmento horizontal de la instancia hasta alcanzar la esquina supe-

rior derecha (esquina superior izquierda) de los cuadrados de la n –ésima fila, los cuales serán desplegados para alcanzar el tercer y cuarto cuadrado, luego el quinto y sexto cuadrado, y finalmente el $(m - 2)$ y $(m|1)$ –ésimo cuadrado de la última fila. Nótese que el cuadrado localizado en la esquina inferior derecha de la instancia no ha sido alcanzado, por lo que es necesario extender la solución con un segmento de línea unitario a partir del último segmento horizontal. Finalmente, se traza un segmento de línea de longitud unitaria desde la esquina superior izquierda del rectángulo $n \times m$ hasta alcanzar el segmento de línea vertical. En la Figura 5.9c se presenta el árbol solución óptimo para la malla I_M de tamaño $n \times m$, para n, m impares, donde se puede observar que todos los cuadrados de 1×1 son tocados al menos por una arista del árbol solución. La longitud total de la solución es $\lfloor \frac{n}{2} \rfloor \times (m - 2) + n + \lceil \frac{m-2}{2} \rceil$.

Con base en lo anterior se plantea el siguiente teorema para instancias de malla I_M de tamaño $n \times m$.

Teorema 5.1. *Longitud total del corredor para instancias de cuadrados.*

Sea I_M una instancia de malla de tamaño $n \times m$. Existe un corredor de longitud mínima con un vértice de esquina v como nodo de acceso de I_M cuya longitud total óptima es $f^(I_M) = \lceil \frac{n \times m}{2} \rceil$.*

Demostración. Consideremos el **caso 1 para n par y m par o impar**. De la discusión anterior correspondiente a la Figura 5.9a, se tiene que:

$$\begin{aligned} \frac{n}{2} \times (m - 2) + n &= \frac{n \times m}{2} \\ &= \left\lceil \frac{n \times m}{2} \right\rceil \end{aligned} \tag{5.1}$$

Consideremos el **caso 2 para n impar y m par**. De la discusión anterior correspondiente a la Figura 5.9b, se tiene que:

$$\begin{aligned}
 \left\lfloor \frac{n}{2} \right\rfloor \times (m-2) + n + \frac{m-2}{2} &= \frac{n-1}{2} \times (m-2) + n + \frac{m-2}{2} \\
 &= \frac{(m-2)}{2} \times n + n \\
 &= n \times \frac{m}{2} \\
 &= \left\lfloor \frac{n \times m}{2} \right\rfloor
 \end{aligned} \tag{5.2}$$

Consideremos el **caso 3 donde n y m son impares**. De la discusión anterior correspondiente a la Figura 5.9c, se tiene que:

$$\begin{aligned}
 \left\lfloor \frac{n}{2} \right\rfloor \times (m-2) + n + \left\lceil \frac{m-2}{2} \right\rceil &= \frac{n-1}{2} \times (m-2) + n + \frac{m-1}{2} \\
 &= \frac{(n-1) \times m}{2} - \frac{2 \times (n-1)}{2} + n + \frac{m-1}{2} \\
 &= \frac{n \times m}{2} - \frac{m}{2} + 1 + \frac{m}{2} - \frac{1}{2} \\
 &= \frac{n \times m}{2} + \frac{1}{2} \\
 &= \left\lceil \frac{n \times m}{2} \right\rceil
 \end{aligned} \tag{5.3}$$

□

Instancias Geométricas de Dominós.

El segundo caso de estudio corresponde a la familia de instancias geométricas de dominós no-isomorfos y libres de corte I_D . A partir de una instancia de

malla I_M de tamaño $n \times m$ con una área par y realizando un embaldosado con dominós de tamaño 1×2 y 2×1 se pueden construir instancias geométricas como la que se presenta en la Figura 5.10 para el caso de una instancia de malla I_M de tamaño 6×5 .

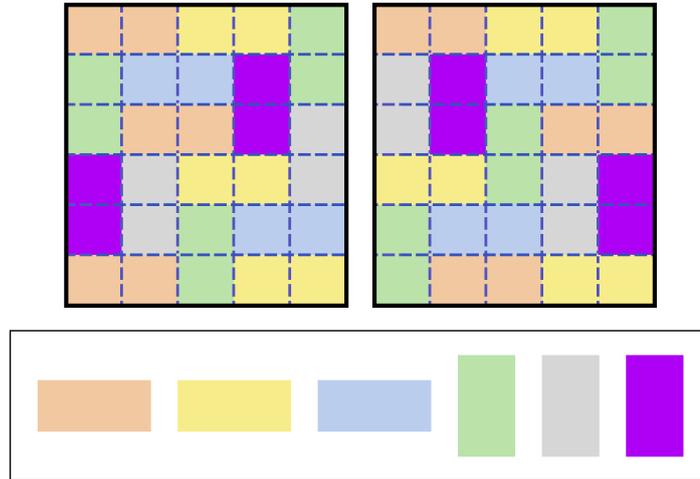


Figura 5.10: Instancias de dominós I_D de tamaño 6×5 .

Determinar la longitud óptima del corredor para una instancia de dominó I_D no es sencillo, debido a que la naturaleza del problema es NP-completo y la configuración de los embaldosados es irregular para cada instancias de tamaño $n \times m$. Sin embargo, se puede calcular una cota inferior y una cota superior, entre las cuales se encuentra la función óptima $f^*(I_D)$, que corresponda a la longitud óptima del corredor en una instancia de dominós I_D de tamaño $n \times m$.

Consideremos primeramente el caso de una instancia de malla I_M de tamaño $n \times m$ con un vértice de acceso v en una de sus esquinas, la cual es embaldosada con dominós de tamaño 1×2 y 2×1 . Si la instancia se reduce 1 unidad tanto en la base como en la altura, tendremos una instancia de malla I'_M de tamaño $(n - 1) \times (m - 1)$ con una función óptima $f^*(I'_M) = \left\lceil \frac{(n-1) \times (m-1)}{2} \right\rceil$. Como se puede apreciar en la Figura 5.11 el árbol solución de la instancia de

malla I'_M solamente alcanza a tocar algunos dominós que se encuentran en el extremo derecho y abajo de la instancia geométrica, mientras que otros dominós nunca son alcanzados. Esto nos indica que el valor de $f^*(I'_M)$ está por debajo de $f^*(I_D)$.

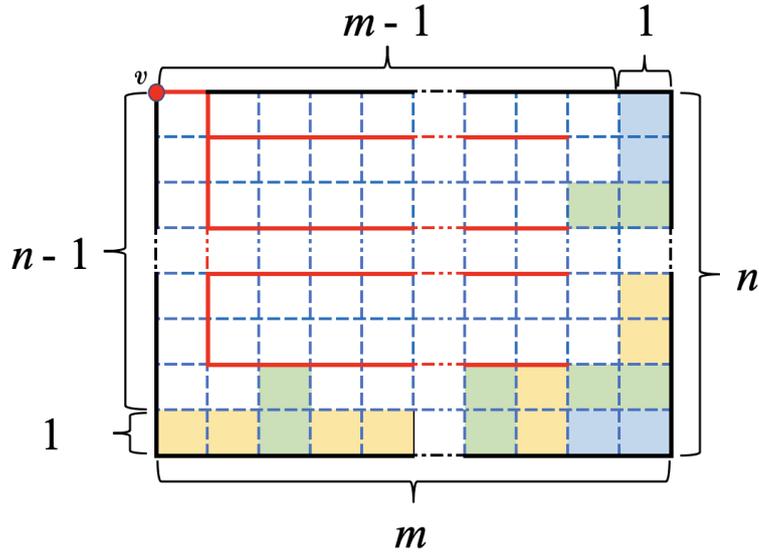


Figura 5.11: Instancia de malla I'_M sobre dominós.

Por otro lado, considerando la instancia de malla I_M de tamaño $n \times m$ se puede observar en la Figura 5.12 que el valor $f^*(I_M)$ correspondiente a la longitud total óptima del corredor de la instancia I_M es mayor que la necesaria para conectar algunos dominós, por lo que es posible reducir dicha longitud total, y en consecuencia $f^*(I_D)$ tendría un valor inferior a $f^*(I_M)$.

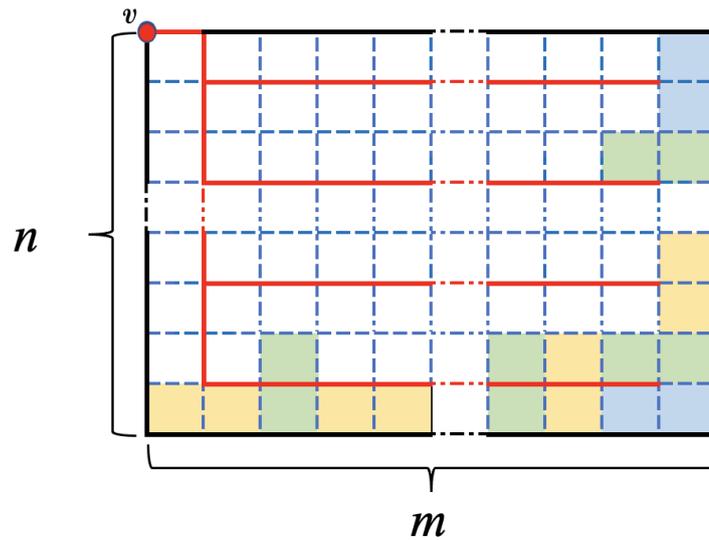


Figura 5.12: Instancias de dominós I_D .

A continuación se establece formalmente el teorema que define las cotas inferior y superior para instancias de dominós I_D con vértices de acceso en cualquiera de sus esquinas.

Teorema 5.2. *Teorema de Cota Inferior y Superior para Dominós con Vértice de Acceso en cualquiera de sus Esquina*

Sea I_D una instancia de dominós de tamaño $n \times m$ de área par y un vértice de acceso en esquina v . La función óptima $f^*(I_D)$ satisface:

$$\left\lfloor \frac{(n-1) \times (m-1)}{2} \right\rfloor \leq f^*(I_D) \leq \left\lceil \frac{n \times m}{2} \right\rceil$$

El rango de valores de $f^*(I_D)$ para los corredores de longitud mínima en algunas instancias con vértice de acceso en esquina se muestra en la Tabla 5.7.

Tabla 5.7: Límite de Cota Inferior y Superior en I_D con Vértice de Acceso en Esquina.

Instancia	Cota Inferior	Cota Superior
6×5	10	15
6×7	15	21
6×8	17	24
6×9	20	27
6×10	22	30
7×8	21	28

Consideremos ahora el caso de una instancia de malla I_M de tamaño $n \times m$ con un vértice de acceso v en el borde de la instancia pero no en esquina, la cual es embaldosada con dominós de tamaño 1×2 y 2×1 . Si la instancia se reduce en 2 unidades en la base y en la altura, tendremos una instancia de malla I'_M de tamaño $(n-2) \times (m-2)$ con una función óptima $f^*(I'_M) = \left\lceil \frac{(n-2) \times (m-2)}{2} \right\rceil$. Como se puede apreciar en la Figura 5.13 el árbol solución de la instancia de malla I'_M no alcanza a tocar ningún dominó que se encuentran en el extremo derecho y abajo de la instancia geométrica. Esto nos indica que $f^*(I'_M)$ es menor que $f^*(I_D)$.

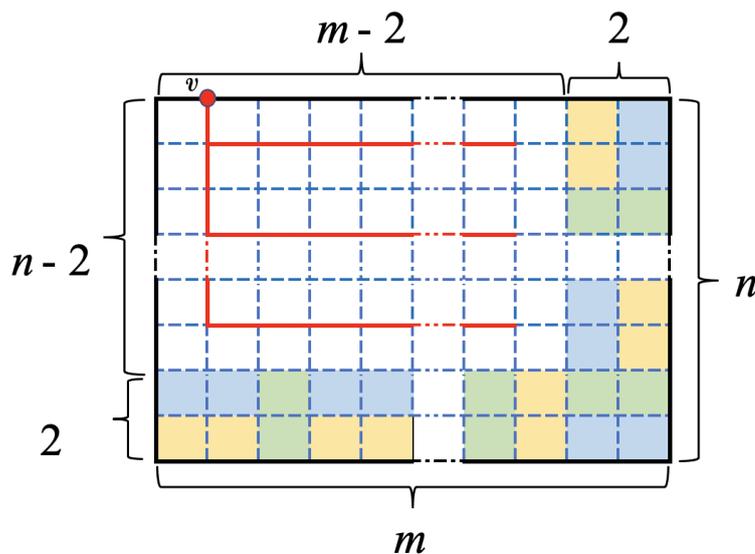


Figura 5.13: Instancias de malla I'_M sobre dominós.

Por otro lado, considerando la instancia de malla I_M de tamaño $n \times m$ se puede observar en la Figura 5.12 que el valor $f^*(I_M)$ correspondiente a la longitud total óptima del corredor de la instancia I_M es mayor que la necesaria para conectar algunos dominós, por lo que es posible reducir dicha longitud total, y en consecuencia $f^*(I_D)$ tendría un valor inferior a $f^*(I_M)$.

A continuación se plantean el teorema que define la cota inferior y la cota superior para instancias de dominós I_D con vértices de acceso en borde pero no en esquina.

Teorema 5.3. *Teorema de Cota Inferior y Superior para Dominós con Vértice de Acceso en Borde pero no en Esquina*

Sea I_D una instancia de dominós de tamaño $n \times m$ de área par y un vértice de acceso en borde pero no en esquina v . La función óptima $f^(I_D)$ satisface:*

$$\left\lfloor \frac{(n-2) \times (m-2)}{2} \right\rfloor < f^*(I_D) \leq \left\lceil \frac{n \times m}{2} \right\rceil$$

El rango de valores de $f^*(I_D)$ para los corredores de longitud mínima en algunas instancias con vértice de acceso en borde pero no en esquina se muestra en la Tabla 5.8.

Tabla 5.8: Límite de Cota Inferior y Superior en I_D con Vértice de Acceso en Borde pero no en Esquina.

Instancia	Cota Inferior	Cota Superior
6×5	6	15
6×7	10	21
6×8	12	24
6×9	14	27
6×10	16	30
7×8	15	28

5.2.2. Características de Instancias para Experimentación.

Se han implementado cuatro heurísticas para abordar el problema del Corredor de Longitud Mínima (MLCP – Minimum Length Corridor Problem) aplicado sobre los conjuntos de familias de embaldosados rectilíneos con dominós, los cuales son:

- (1) Algoritmo de Búsqueda Voraz – Dominós más cercanos (correspondiente al Algoritmo 11 de la sección 4.2.1).
- (2) Algoritmo de Búsqueda Voraz – Dominós más alejados (correspondiente al Algoritmo 12 de la sección 4.2.2).
- (3) Algoritmo de Vértices Compartidos Mejorado (correspondiente al Algoritmo 14 de la sección 4.2.4).
- (4) Algoritmo de Árbol Extendido de Costo Mínimo con Vértices Internos – Reducción (correspondiente al Algoritmo 16 de la sección 4.2.6).

El análisis experimental de los cuatro algoritmos se lleva a cabo sobre un conjunto de seis familias de instancias de embaldosados con dominós. Para ello, y como se discutió en la sección 3.2.2, se eligieron para cada instancia los embaldosados no-isomorfos, es decir los representantes de cada clase, que son libres de líneas de corte. De ello resultan las instancias de embaldosados filtradas (ver Tabla 5.9).

Tabla 5.9: Cantidad de Embaldosados por Tamaño de Instancia.

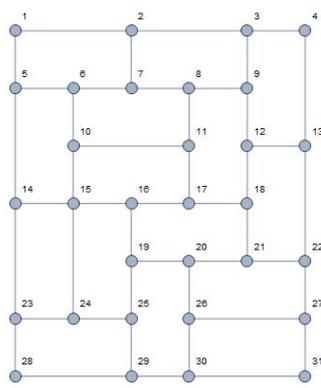
Tamaño Instancia	Cantidad de Embaldosados	
	Total	Filtrados
6 × 5	1 183	2
6 × 7	31 529	32
6 × 8	167 089	18
6 × 9	817 991	418
6 × 10	4 213 133	415
7 × 8	1 292 697	3 391

Para llevar a cabo el diseño de experimentos, primeramente se dimensiona la cantidad total de corredores para el conjunto de las seis familias de instancias. El número de corredores para cada uno de los embaldosados por dominós depende de la cantidad de nodos en la periferia. Para cada embaldosado, se construirán corredores asociados a cada nodo de la periferia considerado como nodo de acceso. Así por ejemplo, de acuerdo con la Tabla 5.10, se tienen un total de 418 embaldosados de dominós no-isomorfos libres de líneas de corte para la instancia de tamaño 6 × 9. Los embaldosados se han clasificado de acuerdo al número de nodos en la periferia. En este caso, se tienen 111 embaldosados con 18 nodos en la periferia, haciendo un total de $4 \times 111 = 444$ corredores cuyo nodo de acceso corresponde a cada una de las esquinas de la instancia. Asimismo, se tiene un total de $14 \times 111 = 1554$ corredores, cuyo nodo de acceso corresponde al conjunto de nodos de la periferia excepto las esquinas. En total, 1998 corredores a generar por cada algoritmo bajo análisis.

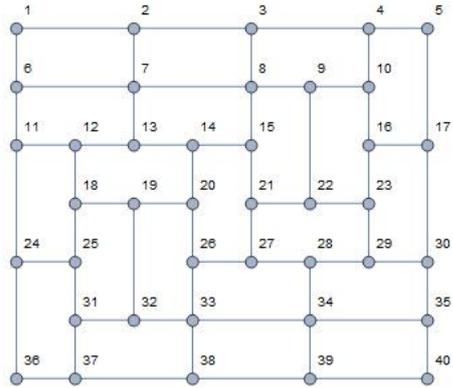
Tabla 5.10: Distribución de Corredores de acuerdo al número de nodos en la periferia.

Tamaño	Instancias Geométricas			Cantidad de Corredores		
	Total de Embaldosados	Embaldosados	Nodos en la Periferia	Esquinas	Bordes	Total
6 × 5	2	2	14	8	20	28
6 × 7	32	16	16	64	192	256
		16	17	64	208	272
	Subtotal	32		128	400	528
6 × 8	18	18	18	72	252	324
6 × 9	418	111	18	444	1 554	1 998
		213	19	852	3 195	4 047
		94	20	376	1 504	1 880
	Subtotal	418		1 672	6 253	7 925
6 × 10	415	242	20	968	3 872	4 840
		173	21	692	2 941	3 633
	Subtotal	415		1 660	6 813	8 473
7 × 8	3 391	528	18	2 112	7 392	9 504
		1 492	19	5 968	22 380	28 348
		1 131	20	4 524	18 096	22 620
		240	21	960	4 080	5 040
	Subtotal	3 391		13 564	51 948	65 512

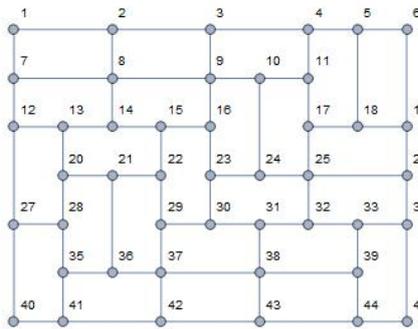
En la Figura 5.14 se presentan seis ejemplos de embaldosados, cada uno perteneciente a las instancias $I_{6 \times 5}$, $I_{6 \times 7}$, $I_{6 \times 8}$, $I_{6 \times 9}$, $I_{6 \times 10}$ e $I_{7 \times 8}$.



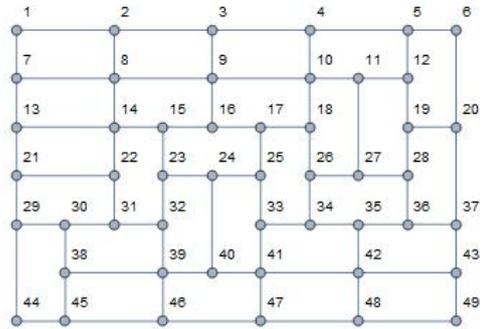
(a) $I_{6 \times 5}$



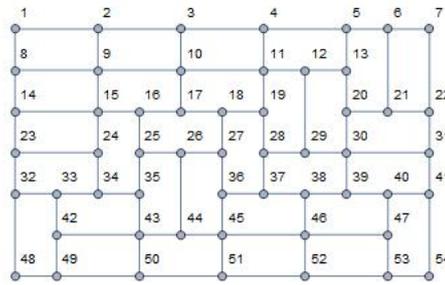
(b) $I_{6 \times 7}$



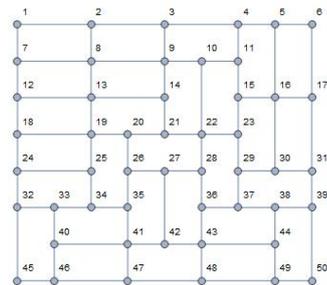
(c) $I_{6 \times 8}$



(d) $I_{6 \times 9}$



(e) $I_{7 \times 8}$



(f) $I_{7 \times 8}$

Figura 5.14: Ejemplos de embaledos.

5.2.3. Análisis de Corredores con Vértice de Acceso en Esquina generados por las Heurísticas.

En esta sección se presentan los resultados obtenidos con las heurísticas implementadas aplicadas al conjunto de instancias geométricas seleccionadas en el Capítulo 3 y presentadas en la Tabla 5.10. Primeramente se tomaron como nodos de acceso los vértices de las esquinas de cada embaldosado, correspondientes a las cuatro esquinas de las instancias geométricas. Se validaron los resultados de la longitud del corredor tomando en cuenta la cota inferior y la cota superior de acuerdo a su tamaño $n \times m$ planteadas en la sección 5.2.1 y resumidas en la Tabla 5.7.

Los resultados para cada tamaño de instancia $n \times m$, se presentan de la siguiente manera:

- 1 Tabla con la cantidad y porcentaje de corredores construidos por las heurísticas, distribuidos de acuerdo a su longitud.
- 2 Gráfica de porcentajes de corredores con longitudes entre la cota inferior y superior, así como aquellos corredores con una longitud mayor a la cota superior.
- 3 Gráfica con la distribución de cantidades de corredores por longitud entre la cota inferior y superior.
- 4 Gráfica con la distribución de cantidades de corredores por longitud mayor a la cota superior.

Instancias de tamaño 6×5 .

En la Tabla 5.11 se presenta la distribución de 8 corredores en 2 instancias de tamaño 6×5 . De acuerdo al Teorema 5.2, las cotas inferior y superior para esta instancia $I_{6 \times 5}$ tienen un valor de 10 y 15 unidades de longitud, respectivamente. La heurística H1 generó los 8 (100%) corredores dentro de las cotas, con una longitud entre 13 y 15, la heurística H3 generó los 8 (100%) corredores dentro de las cotas, con una longitud entre 14 y 15 y la heurística H4 generó los 8 (100%) corredores dentro de las cotas, con una longitud entre 12 y 14. La heurística H2 solamente generó 5 (72.5%) corredores dentro de cotas con una longitud entre 13 y 15, mientras que los corredores con una longitud fuera de la cota superior, es decir, mayor a 15 fueron 3 (37.5%).

Tabla 5.11: Cantidad y porcentaje de corredores en instancias de tamaño 6×5 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
10	0	0.0	0	0.0	0	0.0	0	0.0
11	0	0.0	0	0.0	0	0.0	0	0.0
12	0	0.0	0	0.0	0	0.0	1	12.5
13	1	12.5	2	25.0	0	0.0	5	62.5
14	6	75.0	2	25.0	4	50.0	2	25.0
15	1	12.5	1	12.5	4	50.0	0	0.0
16	0	0.0	2	25.0	0	0.0	0	0.0
17	0	0.0	1	12.5	0	0.0	0	0.0
Total	8	100.0	8	100.0	8	100.0	8	100.0

Se puede observar en la gráfica de la Figura 5.15 que las heurísticas H1, H3 y H4 el 100% de sus corredores están dentro de los límites de las cotas mientras que la heurística H2 tiene 2 (25%) corredores de longitud 16 y 1 (12.5%) corredor de longitud 17, las cuales son mayores a la cota superior.

*Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro y fuera de cotas
Cota Inferior: 10 y Cota Superior: 15
2 Instancias de Tamaño 6 x 5, Total de corredores 8*

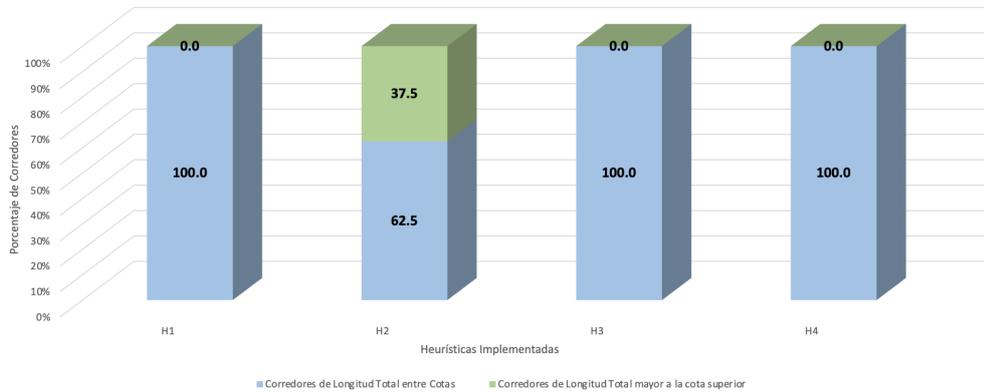


Figura 5.15: Distribución de corredores en instancias de tamaño 6×5 con longitud dentro y fuera de las cotas.

En la gráfica de la Figura 5.16 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior. Para la instancia $I_{6 \times 5}$, la longitud total del corredor va de 12 a 15 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 10 y 15. Obsérvese que la heurística H4 produce la mejor solución correspondiente a un corredor cuya longitud total es de 12 unidades de longitud, mayor a la cota inferior de 10.

La gráfica de la Figura 5.17 es similar a la de la Figura 5.16, solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior. Obsérvese que la heurística H2 produce la peor solución correspondiente a un corredor cuya longitud total es 17 unidades de longitud, mayor a la cota superior de 15. Con una longitud total de 16 unidades de longitud la heurística H2 generó 2 corredores.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 10 y Cota Superior: 15
2 Instancias de Tamaño 6 x 5, Total de corredores: 8

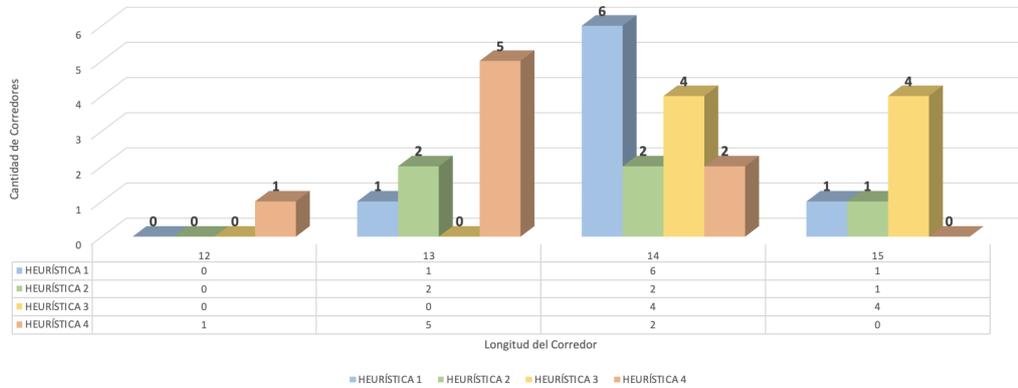


Figura 5.16: Distribución de corredores en instancias de tamaño 6×5 con longitud dentro de los límites de las cotas.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 15
2 Instancias de Tamaño 6 x 5, Total de corredores: 8

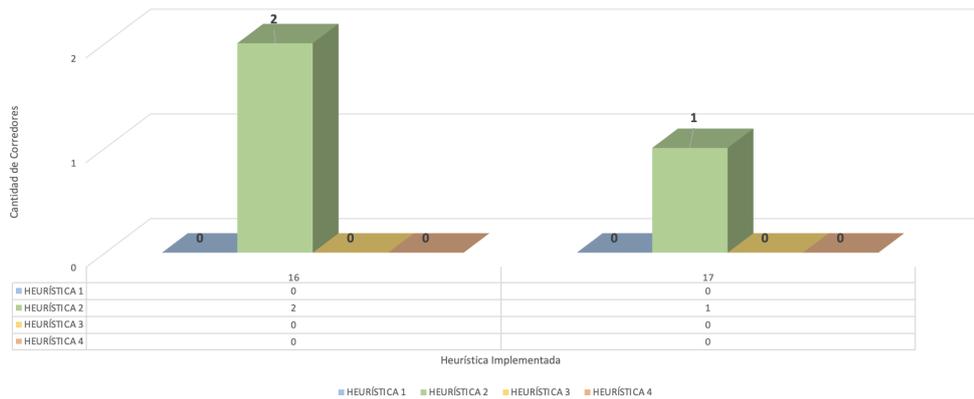


Figura 5.17: Distribución de corredores en instancias de tamaño 6×5 con longitud mayor a la cota superior.

Instancias de tamaño 6×7 .

En la Tabla 5.12 se presenta la distribución de 128 corredores en 32 instancias de tamaño 6×7 . De acuerdo al Teorema 5.2, las cotas inferior y superior para esta instancia $I_{6 \times 7}$ tienen un valor de 15 y 21 unidades de longitud, respectivamente. La heurística H3 generó los 128 (100%) corredores dentro de las cotas

con una longitud entre 17 y 21, la heurística H1 solamente generó 127 (99.2%) corredores dentro de cotas con una longitud entre 16 y 21, la heurística H2 solamente generó 113 (88.3%) corredores dentro de cotas con una longitud entre 17 y 21, mientras que la heurística H4 solamente generó 125 (97.7%) corredores dentro de cotas con una longitud entre 17 y 21. Los corredores con una longitud fuera de la cota superior, es decir, mayor a 21 fueron: 1 (0.8%) corredor por la heurística H1, 15 (11.7%) corredores por la heurística H2 y 3 (2.3%) corredores por la heurística H4.

Tabla 5.12: Cantidad y porcentaje de corredores en instancias de tamaño 6×7 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
15	0	0.0	0	0.0	0	0.0	0	0.0
16	3	2.3	0	0.0	0	0.0	0	0.0
17	6	4.7	5	3.9	12	9.4	9	7.0
18	26	20.3	22	17.2	40	31.2	38	29.7
19	45	35.1	35	27.4	55	43.0	44	34.4
20	29	22.7	32	25.0	20	15.6	25	19.6
21	18	14.1	19	14.8	1	0.8	9	7.0
22	1	0.8	12	9.4	0	0.0	3	2.3
23	0	0.0	3	2.3	0	0.0	0	0.0
Total	128	100.0	128	100.0	128	100.0	128	100.0

Se puede observar en la gráfica de la Figura 5.18 que la heurística H3 el 100% de sus corredores están dentro de los límites de las cotas mientras que las heurísticas H1 tiene 1 (0.8%) corredor de longitud 22, H2 tiene 12 (9.4%) corredores de longitud 22 y 3 (2.3%) corredores de longitud 23, y H4 tiene 3 (2.3%) corredor de longitud 22, las cuales son mayores a la cota superior.

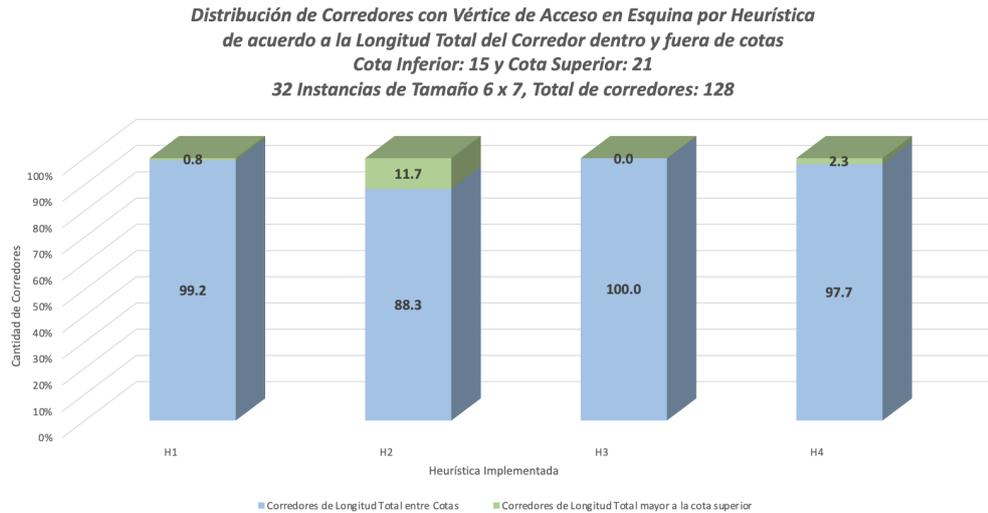


Figura 5.18: Distribución de corredores en instancias de tamaño 6×7 con longitud dentro y fuera de las cotas.

En la gráfica de la Figura 5.19 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior. Para la instancia $I_{6 \times 7}$ la longitud total del corredor va de 16 hasta 21 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 15 y 21. Obsérvese que la heurística H1 produce la mejor solución correspondiente a 3 corredores cuya longitud total es de 16 unidades de longitud, mayor a la cota inferior de 15.

La gráfica de la Figura 5.20 es similar a la de la Figura 5.19, solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior. Obsérvese que la heurística H2 produce la peor solución correspondiente a 3 corredores cuya longitud total es 23 unidades de longitud, mayor a la cota superior de 21. Con una longitud total de 22 unidades de longitud la heurística H1 generó 1 corredor, la heurística H2 generó 12 corredores y la heurística H4 generó 3 corredores. Con una longitud total de 23 unidades de longitud la heurística H2

generó 3 corredores.

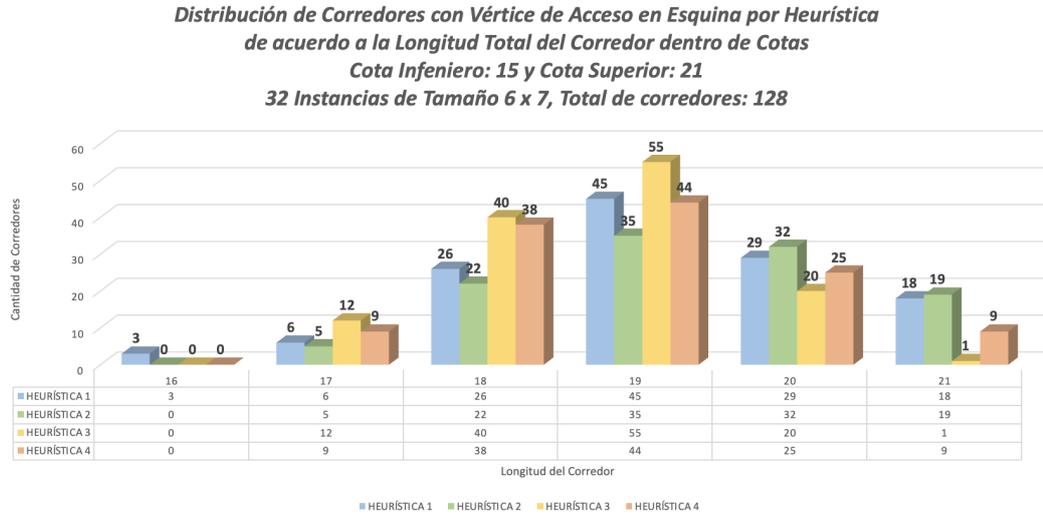


Figura 5.19: Distribución de corredores en instancias de tamaño 6×7 con longitud dentro de los límites de las cotas.

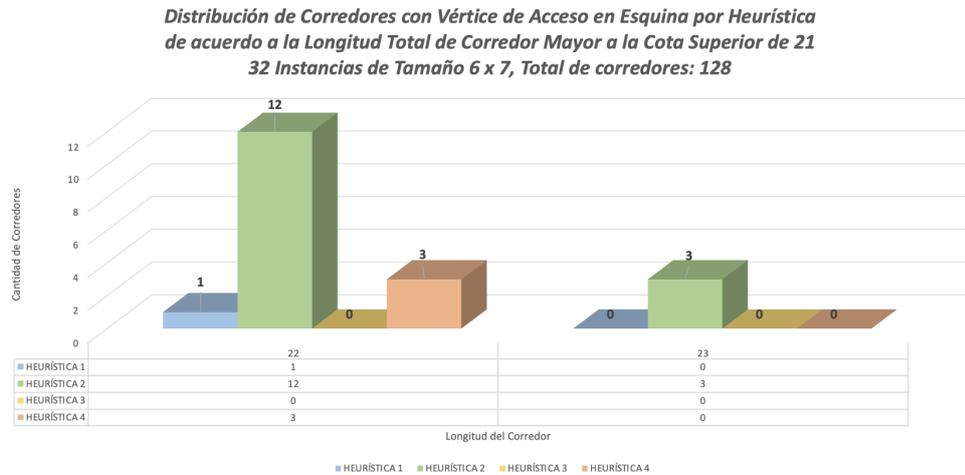


Figura 5.20: Distribución de corredores en instancias de tamaño 6×7 con longitud mayor a la cota superior.

Instancias de tamaño 6×8 .

En la Tabla 5.13 se presenta la distribución de 72 corredores en 18 instancias de tamaño 6×8 . De acuerdo al Teorema 5.2, las cotas inferior y superior

para esta instancia $I_{6 \times 8}$ tienen un valor de 17 y 24 unidades de longitud, respectivamente. La heurística H1 generó los 72 (100%) corredores dentro de las cotas, con una longitud entre 20 y 24, la heurística H3 generó los 72 (100%) corredores dentro de las cotas, con una longitud entre 20 y 23, la heurística H2 solamente generó 69 (95.8%) corredores dentro de cotas con una longitud entre 19 y 24, mientras que la heurística H4 solamente generó 68 (94.4%) corredores dentro de cotas con una longitud entre 19 y 24. Los corredores con una longitud fuera de la cota superior, es decir, mayor a 24 fueron: 3 (4.2%) corredores por la heurística H2 y 4 (5.6%) corredores por la heurística H4.

Tabla 5.13: Cantidad y porcentaje de corredores en instancias de tamaño 6×8 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
17	0	0.0	0	0.0	0	0.0	0	0.0
18	0	0.0	0	0.0	0	0.0	0	0.0
19	0	0.0	1	1.4	0	0.0	1	1.4
20	6	8.4	5	6.9	16	22.2	15	20.8
21	24	33.3	14	19.4	26	36.1	24	33.3
22	24	33.3	20	27.8	20	27.8	17	23.6
23	14	19.4	18	25.0	10	13.9	4	5.6
24	4	5.6	11	15.3	0	0.0	7	9.7
25	0	0.0	0	0.0	0	0.0	1	1.4
26	0	0.0	3	4.2	0	0.0	3	4.2
Total	72	100.0	72	100.0	72	100.0	72	100.0

Se puede observar en la gráfica de la Figura 5.21 que las heurísticas H1 y H3 el 100% de sus corredores están dentro de los límites de las cotas mientras que la heurística H2 tiene 3 (4.2%) corredores de longitud 26, mientras que la heurística H4 tiene 1 (1.4%) corredor de longitud 25 y 3 (4.2%) corredores de longitud 26.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro y fuera de cotas
Cota Inferior: 17 y Cota Superior: 24
18 Instancias de Tamaño 6 x 8, Total de corredores: 72

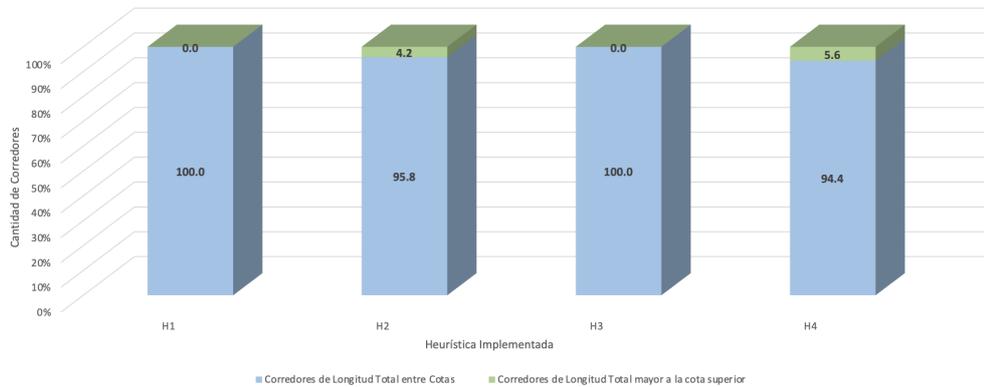


Figura 5.21: Distribución de corredores en instancias de tamaño 6×8 con longitud dentro y fuera de las cotas.

En la gráfica de la Figura 5.22 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentran entre las cotas inferior y superior. Para la instancia $I_{6 \times 8}$ la longitud total del corredor va de 19 hasta 24 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 17 y 24. Obsérvese que las heurísticas H2 y H4 producen la mejor solución correspondiente a 1 corredor cuya longitud es de 19 unidades de longitud, mayor a la cota inferior de 17.

La gráfica de la Figura 5.22 es similar a la de la Figura 5.23 solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior. Obsérvese que las heurísticas H2 y H4 producen la peor solución correspondiente a 3 corredores cuya longitud total es 26 unidades de longitud, mayor a la cota superior de 24. Con una longitud total de 25 unidades de longitud la heurística H4 generó 1 corredor.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 17 y Cota Superior: 24
18 Instancias de Tamaño 6 x 8, Total de corredores: 72

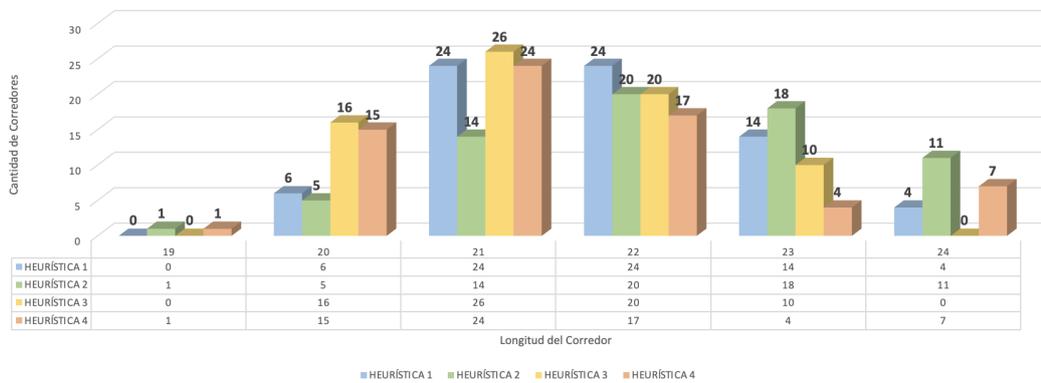


Figura 5.22: Distribución de corredores en instancias de tamaño 6 x 8 con longitud dentro de los límites de las cotas.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 24
18 Instancias de Tamaño 6 x 8, Total de corredores: 72

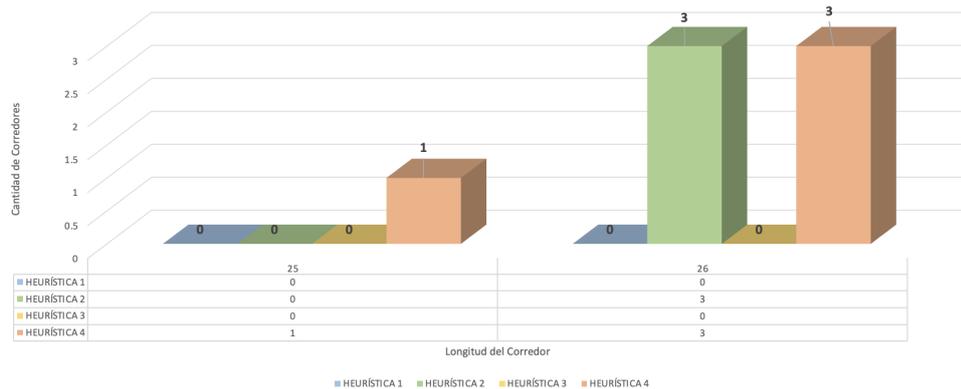


Figura 5.23: Distribución de corredores en instancias de tamaño 6 x 8 con longitud mayor a la cota superior.

Instancias de tamaño 6 x 9.

En la Tabla 5.14 se presenta la distribución de 1 672 corredores en 418 instancias de tamaño 6 x 9. De acuerdo al Teorema 5.2, las cotas inferior y superior para esta instancia $I_{6 \times 9}$ tienen un valor de 20 y 27 unidades de longitud, respectivamente. La heurística H3 generó los 1 672 (100 %) corredores dentro de

cotas con una longitud entre 21 y 27, la heurística H1 solamente generó 1 656 (99.0%) corredores dentro de cotas con una longitud entre 21 y 27, la heurística H2 solamente generó 1 577 (94.7%) corredores dentro de cotas con una longitud entre 21 y 27, mientras que la heurística H4 solamente generó 1 625 (97.2%) corredores dentro de cotas con una longitud entre 20 y 27. Los corredores con una longitud fuera de la cota superior, es decir, mayor a 27 fueron: 16 (1.0%) corredores por la heurística H1, 95 (5.7%) corredores por la heurística H2 y 47 (2.8%) corredores por la heurística H4.

Tabla 5.14: Cantidad y porcentaje de corredores en instancias de tamaño 6×9 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
20	0	0.0	0	0.0	0	0.0	3	0.2
21	16	0.9	4	0.2	10	0.6	19	1.1
22	77	4.6	70	4.2	230	13.8	132	7.9
23	282	16.9	245	14.7	540	32.3	343	20.5
24	505	30.2	420	25.1	574	34.3	465	27.8
25	455	27.2	391	23.4	270	16.1	348	20.8
26	234	14.0	291	17.4	46	2.8	205	12.3
27	87	5.2	156	9.3	2	0.1	110	6.6
28	15	0.9	77	4.6	0	0.0	42	2.5
29	1	0.1	16	0.9	0	0.0	5	0.3
30	0	0.0	1	0.1	0	0.0	0	0.0
31	0	0.0	1	0.1	0	0.0	0	0.0
Total	1 672	100.0						

Se puede observar en la gráfica de la Figura 5.24 que la heurística H3 el 100% de sus corredores están dentro de los límites de las cotas mientras que las heurísticas H1 tiene 15 (0.9%) corredores de longitud 28 y 1 (0.1%) corredor de longitud 29; la heurística H2 tiene 77 (4.6%) corredores de longitud 28, 16 (0.9%) corredores de longitud 29, 1 (0.1%) corredores de longitud 30 y 1 (0.1%) corredores de longitud 31; y la heurística H4 tiene 42 (2.5%) corredores de longitud 28 y 5 (0.3%) corredor de longitud 17, las cuales son mayores a la

cota superior.

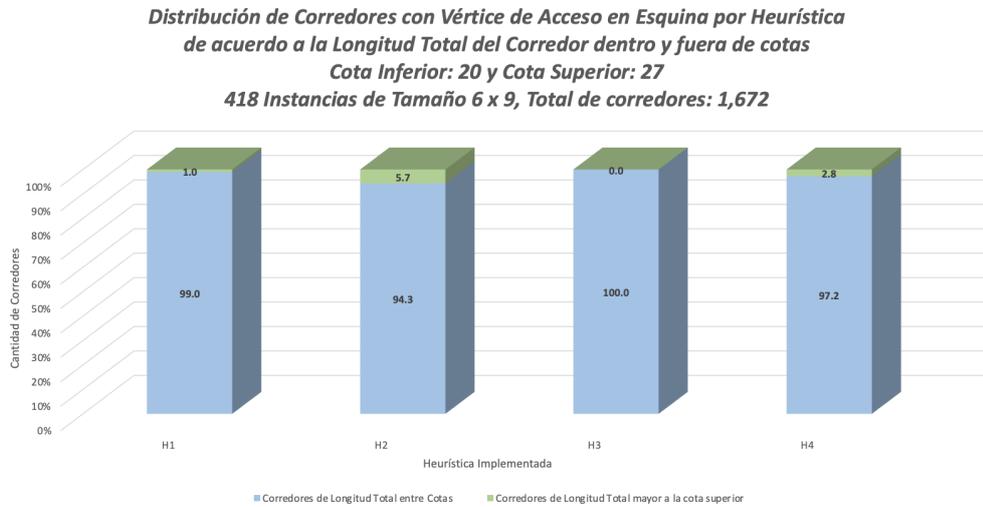


Figura 5.24: Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.25 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior. Para la instancia $I_{6 \times 9}$ la longitud total del corredor va 20 hasta 27 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 20 y 27. Obsérvese que la heurística H4 produce la mejor solución correspondiente a 3 corredores cuya longitud total es de 20 unidades de longitud, igual a la cota inferior de 20.

La gráfica de la Figura 5.26 es similar a la de la Figura 5.25, solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior. Obsérvese que la heurística H2 produce la peor solución correspondiente a 1 corredor cuya longitud total es 31 unidades de longitud, mayor a la cota superior de 27. Con una longitud total de 28 unidades de longitud la heurística H1 generó 15 corredores, la heurística H2 generó 77 corredores y la heurística H4 generó 42

corredores. Con una longitud total de 29 unidades de longitud la heurística H1 generó 1 corredor, la heurística H2 generó 16 corredores y la heurística H4 generó 5 corredores. Con una longitud total de 30 unidades de longitud la heurística H1 generó 1 corredor. Con una longitud total de 31 unidades de longitud la heurística H1 generó 1 corredor.

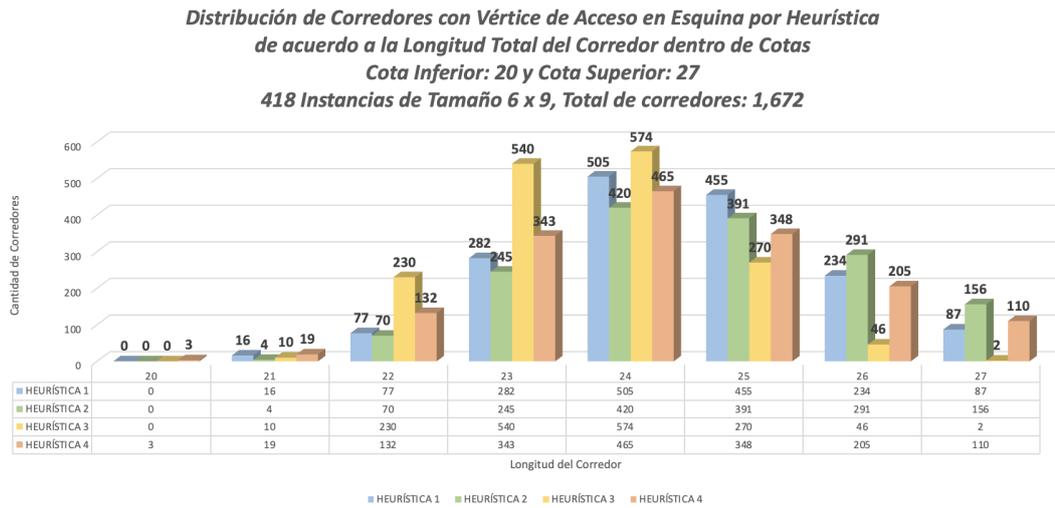


Figura 5.25: Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro de las cotas.

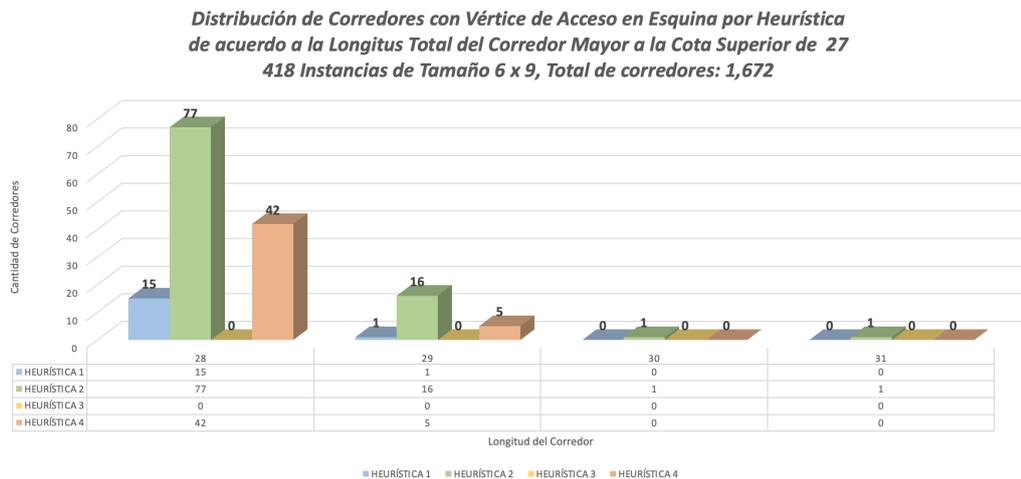


Figura 5.26: Distribución de corredores en instancias de tamaño 6×9 con longitudes mayor a la cota superior.

Instancias de tamaño 6×10 .

En la Tabla 5.15 se presenta la distribución de 1 660 corredores en 415 instancias de tamaño 6×10 . De acuerdo al Teorema 5.2, las cotas inferior y superior para esta instancia $I_{6 \times 10}$ tienen un valor de 22 y 30 unidades de longitud, respectivamente. La heurística H3 generó los 1 660 (100 %) corredores dentro de las cotas con una longitud entre 23 y 30, la heurística H1 solamente generó 1 655 (99.7 %) corredores dentro de cotas con una longitud entre 23 y 30, la heurística H2 solamente generó 1 546 (93.1 %) corredores dentro de cotas con una longitud entre 23 y 30, mientras que la heurística H4 solamente generó 1 618 (97.5 %) corredores dentro de cotas con una longitud entre 23 y 30. Los corredores con una longitud fuera de la cota superior, es decir, mayor a 30 fueron: 5 (0.3 %) corredores por la heurística H1, 114 (6.9 %) corredores por la heurística H2 y 42 (2.5 %) corredores por la heurística H4.

Tabla 5.15: Cantidad y porcentaje de corredores en instancias de tamaño 6×10 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
22	0	0.0	0	0.0	0	0.0	0	0.0
23	4	0.2	1	0.1	13	0.8	7	0.4
24	20	1.2	34	2.0	133	8.0	88	5.3
25	178	10.7	174	10.5	403	24.3	247	14.9
26	408	24.6	318	19.2	566	34.1	401	24.2
27	486	29.3	319	19.2	364	21.9	340	20.5
28	369	22.2	314	18.9	147	8.9	299	18.0
29	154	9.3	246	14.8	30	1.8	143	8.6
30	36	2.2	140	8.4	4	0.2	93	5.6
31	5	0.3	73	4.4	0	0.0	17	1.0
32	0	0.0	28	1.7	0	0.0	22	1.3
33	0	0.0	9	0.6	0	0.0	3	0.2
34	0	0.0	4	0.2	0	0.0	0	0.0
Total	1 660	100.0						

Se puede observar en la gráfica de la Figura 5.27 que la heurística H3 el 100 % de sus corredores están dentro de los límites de las cotas mientras que las heurísticas H1 tiene 5 (0.3 %) corredores de longitud 31; H2 tiene 73 (4.4 %) corredores de longitud 31, 28 (1.7 %) corredores de longitud 32, 9 (0.6 %) corredores de longitud 33 y 4 (0.2 %) corredor de longitud 34; H4 tiene 17 (1.0 %) corredores de longitud 31, 22 (1.3 %) corredores de longitud 32 y 3 (0.2 %) corredor de longitud 33, las cuales son mayores a la cota superior.

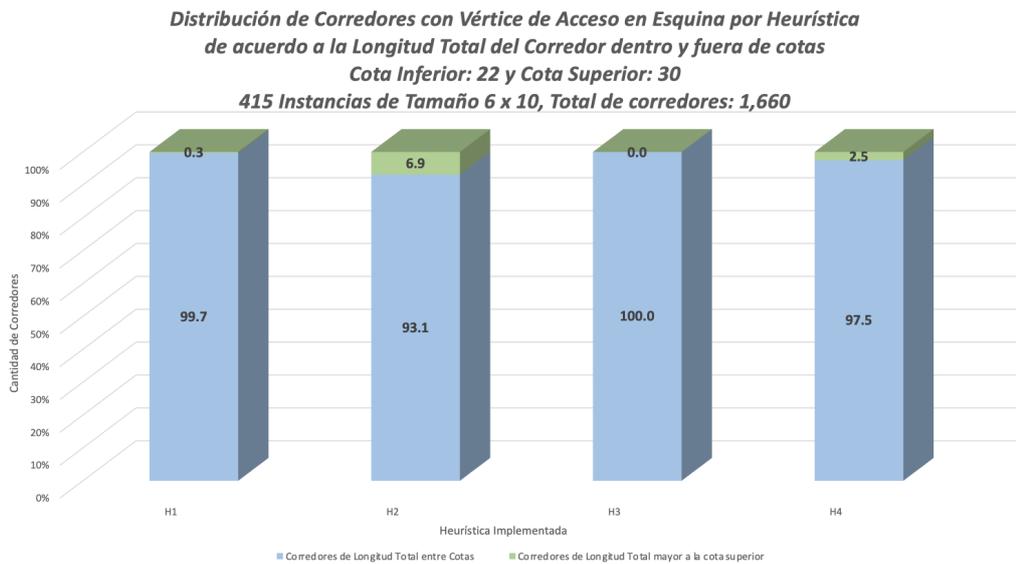


Figura 5.27: Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.28 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior. Para la instancia $I_{6 \times 10}$ la longitud total del corredor va de 23 hasta 30 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 22 y 30. Obsérvese que la mejor solución es producida por todas las heurísticas con corredores cuya longitud total es de 23 unidades de longitud, mayor a la cota inferior de 22. La heurística H1 generó 4 corredores, la heurística H2 generó 1 corredor, la heurística H3

generó 13 corredores, mientras que la heurística H4 generó 7 corredores.

La gráfica de la Figura 5.29 es similar a la de la Figura 5.28 solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior. Obsérvese que la heurística H2 produce la pero solución correspondiente a 4 corredores cuya longitud total es 34 unidades de longitud, mayor que la cota superior de 30. Con una longitud total de 31 unidades de longitud la heurística H1 generó 5 corredores, la heurística H2 generó 73 corredores y la heurística H4 generó 17 corredores. Con una longitud total de 32 unidades de longitud la heurística H2 genero 28 corredores y la heurística H4 genero 22 corredores. Con una longitud total de 33 unidades de longitud la heurística H2 generó 9 corredores y la heurística H4 3 corredores. Con una longitud total de 34 unidades de longitud la heurística H2 generó 4 corredores.

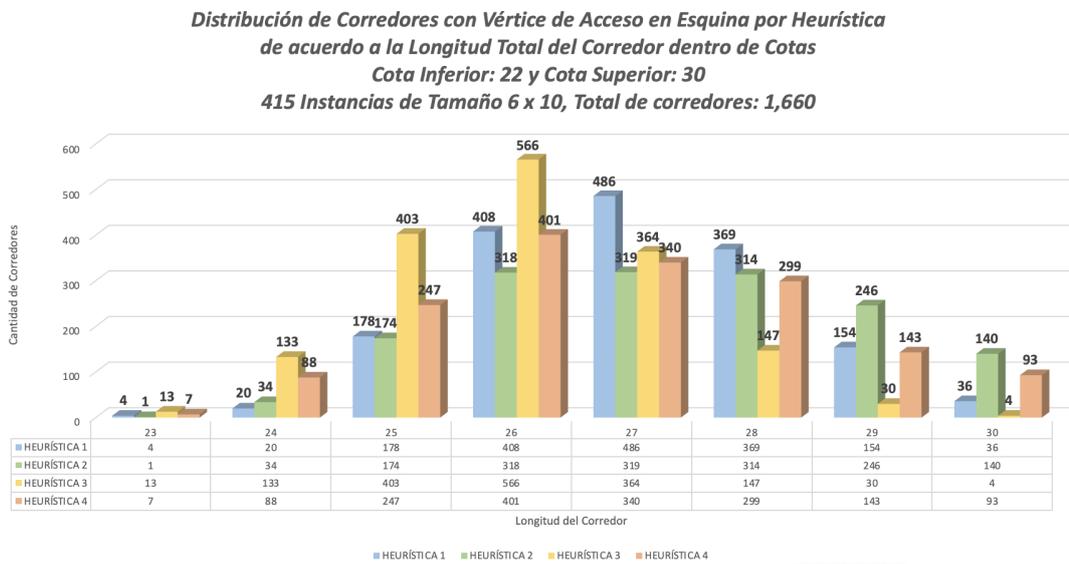


Figura 5.28: Distribución de corredores en instancias de tamaño 6 x 10 con longitudes dentro de los límites de las cotas.

*Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 30
415 Instancias de Tamaño 6 x 10, Total de corredores: 1,660*

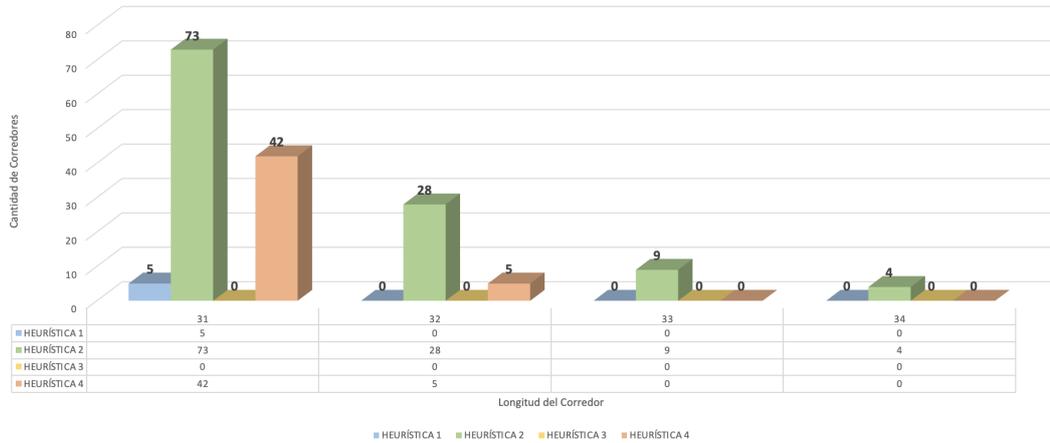


Figura 5.29: Distribución de corredores en instancias de tamaño 6×10 con longitudes mayor a la cota superior.

Instancias de tamaño 7×8 .

En la Tabla 5.16 se presenta la distribución de 13 564 corredores en 3 391 instancias de tamaño 7×8 . De acuerdo al Teorema 5.2, las cotas inferior y superior para esta instancia $I_{7 \times 8}$ tienen un valor de 21 y 28 unidades de longitud, respectivamente. La heurística H3 generó los 13 564 (100%) corredores dentro de las cotas con una longitud entre 21 y 28, la heurística H1 solamente generó 13 551 (99.6%) corredores dentro de cotas con una longitud entre 21 y 28, la heurística H2 solamente generó 12 780 (94.2%) corredores dentro de cotas con una longitud entre 22 y 28, mientras que la heurística H4 solamente generó 12 881 (95.0%) corredores dentro de cotas con una longitud entre 21 y 28. Los corredores con una longitud fuera de la cota superior, es decir, mayor a 28 fueron: 53 (0.4%) corredores por la heurística H1, 784 (5.8%) corredores la heurística H2 y 683 (5.0%) corredores por la heurística H4.

Tabla 5.16: Cantidad y porcentaje de corredores en instancias de tamaño 7×8 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
21	9	0.1	0	0.0	67	0.5	11	0.1
22	204	1.5	106	0.8	934	6.9	210	1.6
23	1 343	9.9	970	7.1	3 231	23.8	949	7.0
24	3 430	25.3	2 427	17.9	4 574	33.7	1 996	14.7
25	4 253	31.3	3 040	22.4	3 175	23.4	2 986	22.0
26	29 75	21.9	2 855	21.1	1 236	9.1	3 243	23.9
27	1 055	7.8	2 132	15.7	321	2.4	2 241	16.5
28	242	1.8	1 250	9.2	26	0.2	1 245	9.2
29	49	0.37	583	4.31	0	0.0	542	4.0
30	4	0.03	151	1.1	0	0.0	119	0.86
31	0	0.0	37	0.3	0	0.0	20	0.13
32	0	0.0	11	0.08	0	0.0	2	0.01
33	0	0.0	2	0.01	0	0.0	0	0.0
Total	13 564	100.0						

Se puede observar en la gráfica en la Figura 5.30 que la heurística H3 el 100 % de sus corredores están dentro de los límites de las cotas mientras que las heurísticas H1 tiene 49 (0.37 %) corredores de longitud 29 y 4 (0.03 %) corredores de longitud 30; la heurística H2 tiene 583 (4.31 %) corredores de longitud 29, 151 (1.1 %) corredores de longitud 30, 37 (0.3 %) corredores de longitud 31, 11 (0.08 %) corredores de longitud 32 y 2 (0.01 %) corredor de longitud 33; y la heurística H4 tiene 542 (4.0 %) corredores de longitud 29, 119 (0.86 %) corredores de longitud 30, 20 (0.13 %) corredores de longitud 31 y 2 (0.01 %) corredor de longitud 32.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro y fuera de cotas
Cota Inferior: 21 y Cota Superior: 28
3,391 Instancias de Tamaño 7 x 8, Total de corredor: 13,564

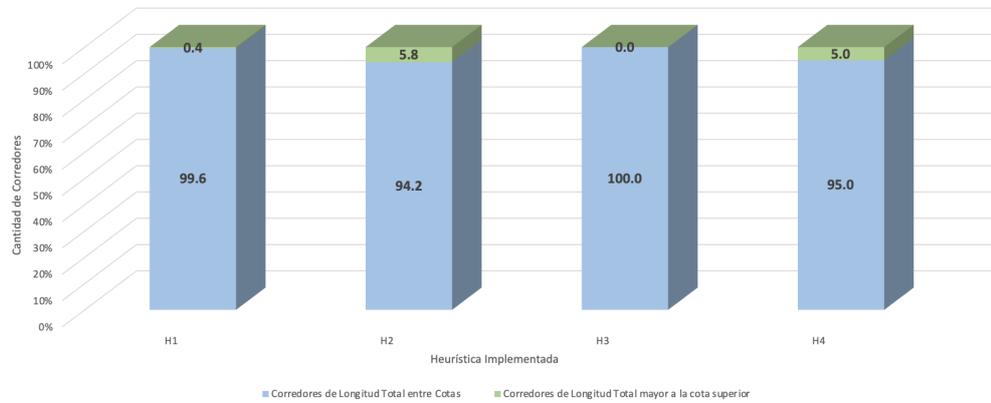


Figura 5.30: Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.31 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior. Para la instancia $I_{7 \times 8}$ la longitud total del corredor va de 21 hasta 28 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior, que correspondientemente son 21 y 28. Obsérvese que las heurísticas H1, H3 y H4 producen la mejor solución correspondiente a 9, 67 y 11 corredores respectivamente cuya longitud total es de 21 unidades de longitud, igual a la cota inferior de 21.

La gráfica de la Figura 5.32 es similar a la de la Figura 5.31 solamente que en este caso se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total, que es mayor a la cota superior de 28. Obsérvese que la heurística H2 produce la peor solución correspondiente a 2 corredores cuya longitud total es 33 unidades de longitud, mayor a la cota superior de 28. Con una longitud total de 29 unidades de longitud la heurística H1 generó 49 corredores, la heurística H2 generó 583 corredores y la heurística

H4 generó 542 corredores. Con una longitud total de 30 unidades de longitud la heurística H1 generó 4 corredores, la heurística H2 generó 151 corredores y la heurística H4 generó 119 corredores. Con una longitud total de 31 unidades de longitud la heurística H2 generó 37 corredores y la heurística H4 generó 20 corredores. Con una longitud total de 32 unidades de longitud la heurística H2 generó 11 corredores y la heurística H4 generó 2 corredores. Con una longitud total de 33 unidades de longitud la heurística H2 generó 2 corredores.

Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 21 y Cota Superior: 28
3,391 Instancias de Tamaño 7 x 8, Total de corredores: 13,564

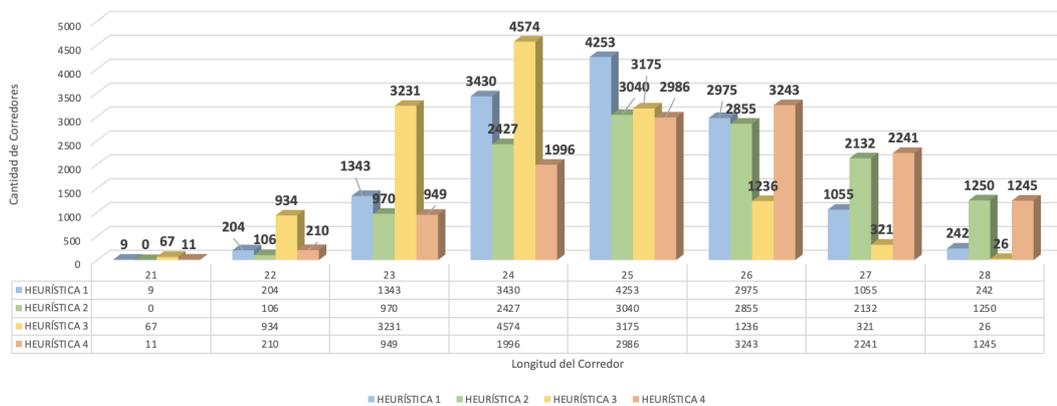


Figura 5.31: Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro de los límites de las cotas.

*Distribución de Corredores con Vértice de Acceso en Esquina por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 28
3,391 Instancias de Tamaño 7 x 8, Total de corredores: 13,564*

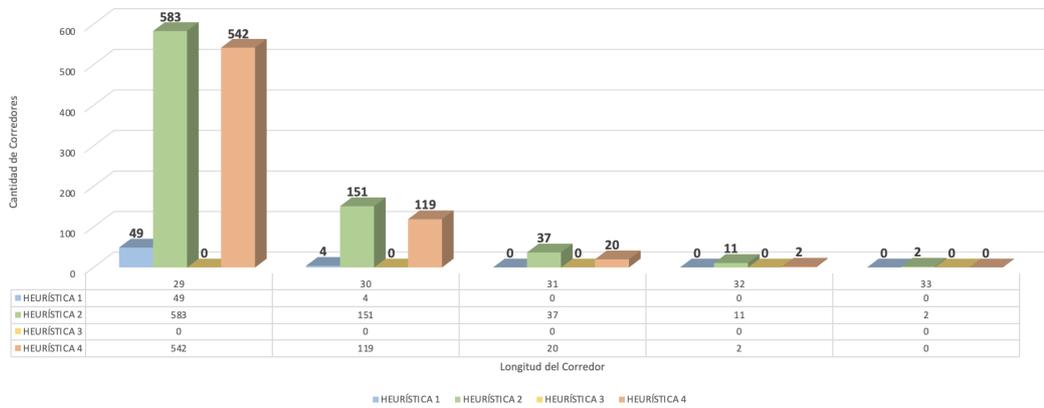


Figura 5.32: Distribución de corredores en instancias de tamaño 7×8 con longitudes mayor a la cota superior.

5.2.4. Análisis de Corredores con Vértice de Acceso en Borde pero no en Esquinas generados por las Heurísticas.

En esta sección se presentan los resultados obtenidos con las heurísticas implementadas aplicadas al conjunto de instancias geométricas seleccionadas en el Capítulo 3 y presentadas en la Tabla 5.10. En este análisis experimental se tomaron como nodos de acceso los vértices del borde de cada embaledado, excepto los vértices de las esquinas. Se validaron los resultados de la longitud total del corredor tomando en cuenta la cota inferior y la cota superior de acuerdo a su tamaño $n \times m$ planteadas en la sección 5.2.1 y presentadas en la Tabla 5.8.

Los resultados para cada instancia de tamaño $n \times m$ se presentan de la siguiente manera:

- 1 Tabla con la cantidad y porcentaje de corredores construidos por las heurísticas, distribuidos de acuerdo a su longitud total.

- 2 Gráfica de porcentajes de corredores con longitud total entre la cota inferior y superior, así como aquellos corredores con una longitud total mayor a la cota superior.
- 3 Gráfica con la distribución de cantidades de corredores por longitud total entre la cota inferior y superior.
- 4 Gráfica con la distribución de cantidades de corredores por longitud total mayor a la cota superior.

Instancias de tamaño 6×5 .

La instancia de tamaño 6×5 consiste en 2 configuraciones, con 10 nodos en el borde sin considerar las esquinas en cada una de ellas. Ello produce 20 diferentes corredores, cada uno teniendo un vértice del borde sin considerar las esquinas como nodo de acceso. Los 20 corredores producidos por las cuatro heurísticas tienen una longitud total dentro de las cotas inferior y superior de 6 y 15, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.17).

Tabla 5.17: Cantidad y porcentaje de corredores en instancias de tamaño 6×5 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
6	0	0.0	0	0.0	0	0.0	0	0.0
7	0	0.0	0	0.0	0	0.0	0	0.0
8	0	0.0	0	0.0	0	0.0	0	0.0
9	0	0.0	0	0.0	0	0.0	0	0.0
10	0	0.0	0	0.0	0	0.0	0	0.0
11	0	0.0	4	20.0	0	0.0	6	30.0
12	6	30.0	8	40.0	0	0.0	11	55.0
13	11	55.0	4	20.0	14	70.0	3	15.0
14	3	15.0	4	20.0	6	30.0	0	0.0
15	0	0.0	0	0.0	0	0.0	0	0.0
Total	20	100.0	20	100.0	20	100.0	20	100.0

En la gráfica de la Figura 5.33 se presenta la distribución de corredores producidos por cada heurística de acuerdo a su longitud total que se encuentra entre las cotas inferior y superior, que correspondientemente son 6 y 15. Para la instancia $I_{6 \times 5}$, la longitud total del corredor va de 11 a 14 unidades de longitud. Dicho rango está dentro de las cotas inferior y superior. Obsérvese que las heurísticas H2 y H4 producen la mejor solución correspondientemente a 4 y 6 corredores de longitud total de 11 unidades de longitud. Cabe mencionar que las cuatro heurísticas producen corredores cuyas longitudes totales no son mayores a la cota superior.

*Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 6 y Cota Superior: 15
2 Instancias de Tamaño 6 x 5, Total de corredores: 20*

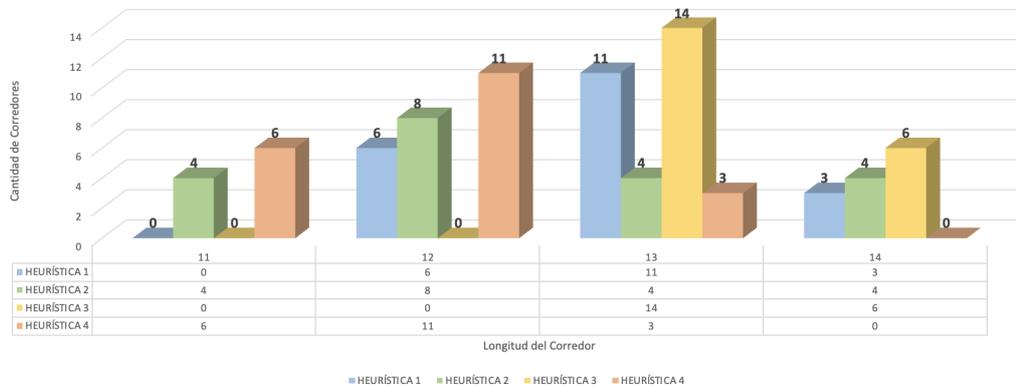


Figura 5.33: Distribución de corredores en instancias de tamaño 6×5 con longitud dentro de las cotas.

Instancias de tamaño 6×7 .

La instancia de tamaño 6×7 consiste en 16 configuraciones con 12 nodos en el borde y 16 configuraciones con 13 nodos en el borde, sin considerar las esquinas en cada una de ellas. Ello produce 400 diferentes corredores, cada uno teniendo un vértice del borde sin considerar las esquinas como nodo de acceso. Los 400 corredores producidos por las heurísticas H3 y H4 tienen una longitud total dentro de las cotas inferior y superior de 10 y 21, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.18). Sin embargo, algunos corredores producidos por las heurísticas H1 y H2 tienen una longitud total que excede la cota superior (ver Figura 5.34).

Tabla 5.18: Cantidad y porcentaje de corredores en instancias de tamaño 6×7 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
10	0	0.0	0	0.0	0	0.0	0	0.0
11	0	0.0	0	0.0	0	0.0	0	0.0
12	0	0.0	0	0.0	0	0.0	0	0.0
13	0	0.0	0	0.0	0	0.0	0	0.0
14	0	0.0	0	0.0	0	0.0	0	0.0
15	8	2.0	8	2.0	6	1.5	0	0.0
16	24	6.0	57	14.3	44	11.0	51	12.7
17	87	21.7	144	36.0	133	33.3	129	32.3
18	141	35.3	118	29.5	155	38.7	120	30.0
19	90	22.5	57	14.3	58	14.5	72	18.0
20	46	11.5	11	2.7	4	1.0	24	6.0
21	2	0.5	2	0.5	0	0.0	4	1.0
22	2	0.5	1	0.3	0	0.0	0	0.0
23	0	0.0	2	0.5	0	0.0	0	0.0
Total	400	100.0	400	100.0	400	100.0	400	100.0

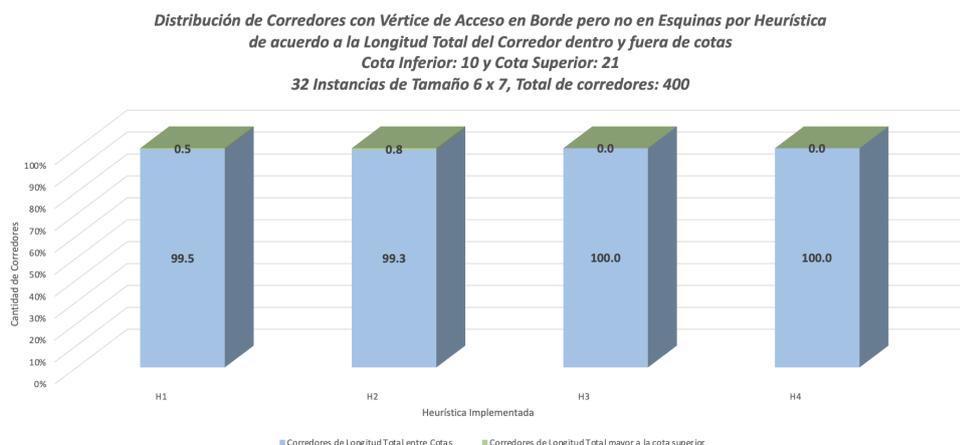


Figura 5.34: Distribución de corredores en instancias de tamaño 6×7 con longitud dentro y fuera de las cotas.

En la gráfica de la Figura 5.35 se puede apreciar la distribución de los corredores generados por las heurísticas en un rango de longitud de 15 hasta 21. La heurística H1, H2 y H3 aportan la mejor solución para 8, 8 y 6 corredores con una longitud de 15 respectivamente. En la gráfica de la Figura 5.36 se presenta

la cantidad de corredores que tienen una longitud mayor a la cota superior. Con una longitud de 22 la heurística H1 generó 2 corredor y la heurística H2 generó 1, mientras que de una longitud de 23 solamente la heurística H2 generó 2.

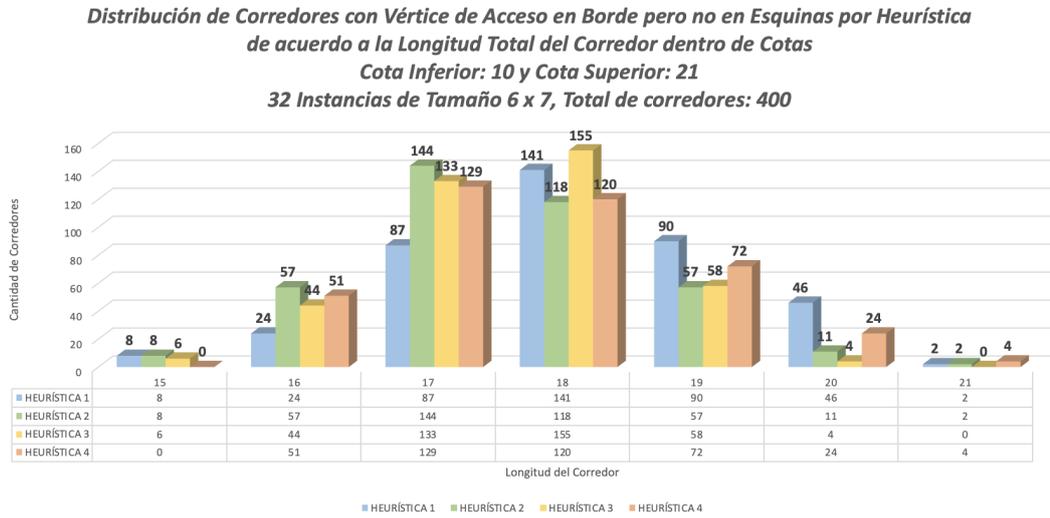


Figura 5.35: Distribución de corredores en instancias de tamaño 6×7 con longitud dentro de los límites de las cotas.

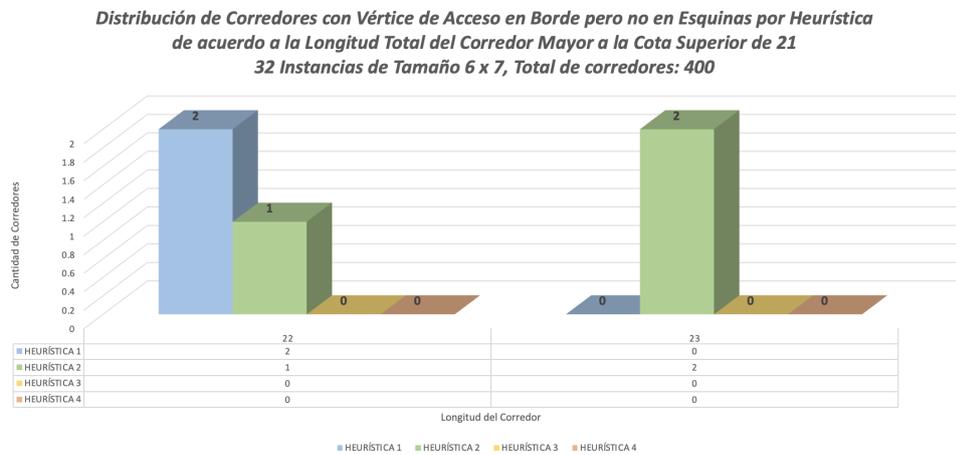


Figura 5.36: Distribución de corredores en instancias de tamaño 6×7 con longitud mayor a la cota superior.

Instancias de tamaño 6×8 .

La instancia de tamaño 6×8 consiste en 18 configuraciones con 14 nodos en el borde, sin considerar las esquinas en cada una de ellas. Ello produce 252 diferentes corredores, cada uno teniendo un vértice del borde sin considerar las esquinas como nodo de acceso. Los 252 corredores producidos por las heurísticas H1 y H3 tienen una longitud total dentro de las cotas inferior y superior de 12 y 24, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.19). Sin embargo, algunos corredores producidos por las heurísticas H2 y H4 tienen una longitud total que excede la cota superior (ver Figura 5.37).

Tabla 5.19: Cantidad y porcentaje de corredores en instancias de tamaño 6×8 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
12	0	0.0	0	0.0	0	0.0	0	0.0
13	0	0.0	0	0.0	0	0.0	0	0.0
14	0	0.0	0	0.0	0	0.0	0	0.0
15	0	0.0	0	0.0	0	0.0	0	0.0
16	0	0.0	0	0.0	0	0.0	0	0.0
17	0	0.0	0	0.0	0	0.0	0	0.0
18	1	0.4	14	5.5	7	2.8	15	5.9
19	26	10.3	58	23.0	74	29.4	69	27.4
20	72	28.6	68	27.0	84	33.3	72	28.6
21	82	32.5	65	25.8	59	23.4	45	17.8
22	60	23.8	29	11.5	25	9.9	26	10.3
23	11	4.4	10	4.0	3	1.2	11	4.4
24	0	0.0	4	1.6	0	0.0	9	3.6
25	0	0.0	0	0.0	0	0.0	5	2.0
26	0	0.0	1	0.4	0	0.0	0	0.0
27	0	0.0	3	1.2	0	0.0	0	0.0
Total	252	100.0	252	100.0	252	100.0	252	100.0

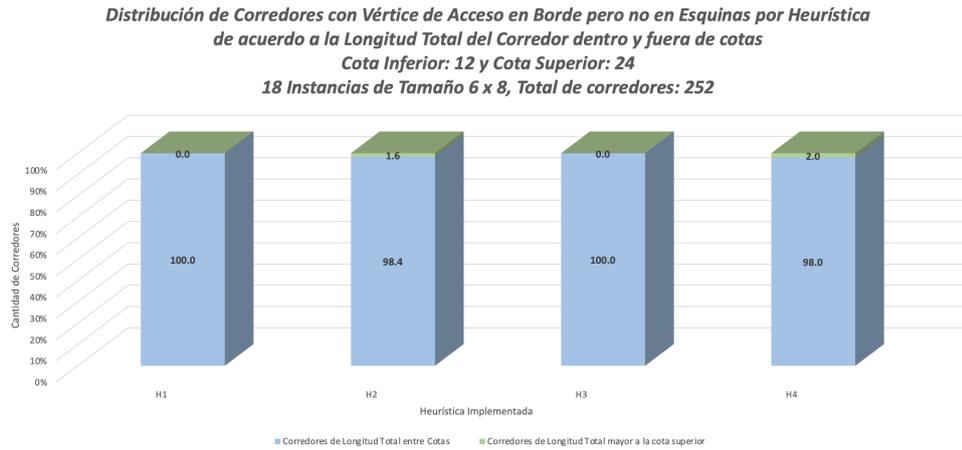


Figura 5.37: Distribución de corredores en instancias de tamaño 6×8 con longitud dentro y fuera de las cotas.

En la gráfica de la Figura 5.38 se puede apreciar la distribución de los corredores generados por las heurísticas en un rango de longitud de 18 hasta 24. Las cuatro heurísticas H1, H2, H3 y H4 aportan la mejor solución para 1, 14, 7 y 15 corredores con una longitud de 18 respectivamente. En la gráfica de la Figura 5.39 se presenta la cantidad de corredores que tienen una longitud mayor a la cota superior. Con una longitud de 25 la heurística H4 generó 5 corredores, con una longitud de 26 la heurística H2 generó 1 corredor, mientras que con una longitud de 27 la heurística H2 generó 3 corredores.

*Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 12 y Cota Superior: 24
18 Instancias de Tamaño 6 x 8, Total de corredores: 252*

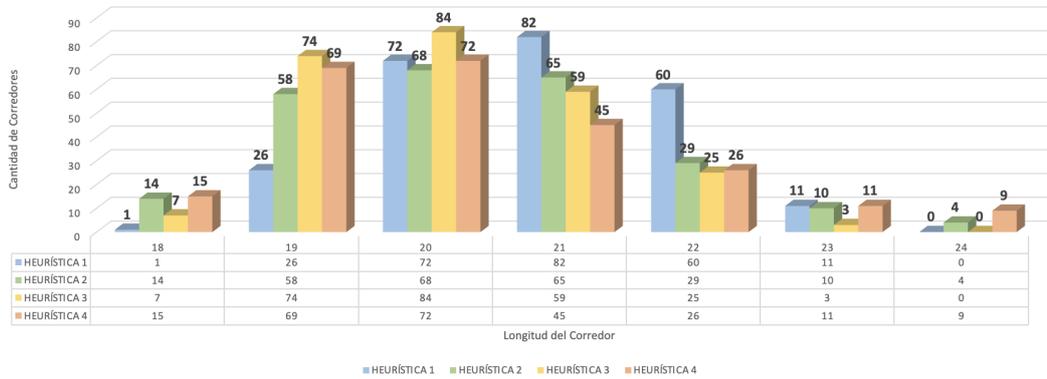


Figura 5.38: Distribución de corredores en instancias de tamaño 6×8 con longitud dentro de las cotas.

*Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 24
18 Instancias de Tamaño 6 x 8, Total de corredores: 252*

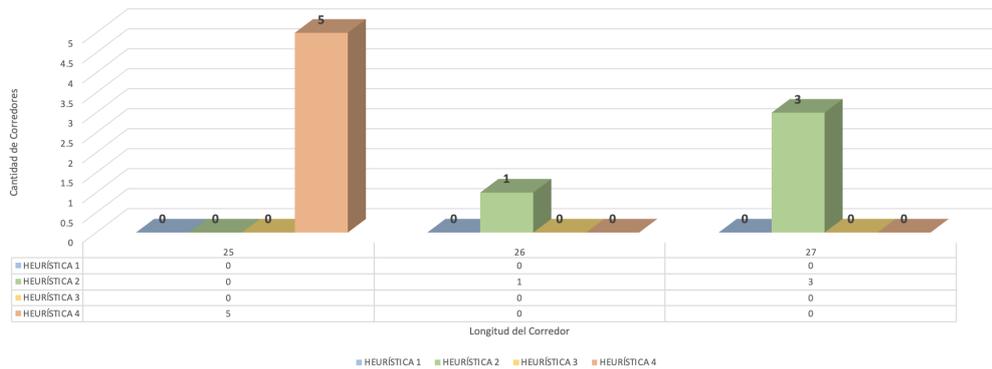


Figura 5.39: Distribución de corredores en instancias de tamaño 6×8 con longitud mayor a la cota superior.

Instancias de tamaño 6×9 .

La instancia de tamaño 6×9 consiste en 111 configuraciones con 14 nodos en el borde, 213 configuraciones con 15 nodos en el borde y 94 configuraciones con 16 nodos en el borde, sin considerar las esquinas en cada una de ellas. Ello produce 6 253 diferentes corredores, cada uno teniendo un vértice del borde sin

considerar las esquinas como nodo de acceso. Los 6 253 corredores producidos por la heurística H3 tienen una longitud total dentro de las cotas inferior y superior de 14 y 27, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.20). Sin embargo, algunos corredores producidos por las heurísticas H1, H2 y H4 tienen una longitud total que excede la cota superior (ver Figura 5.40).

Tabla 5.20: Cantidad y porcentaje de corredores en instancias de tamaño 6×9 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
14	0	0.0	0	0.0	0	0.0	0	0.0
15	0	0.0	0	0.0	0	0.0	0	0.0
16	0	0.0	0	0.0	0	0.0	0	0.0
17	0	0.0	0	0.0	0	0.0	0	0.0
18	0	0.0	0	0.0	0	0.0	0	0.0
19	4	0.06	36	0.58	0	0.0	24	0.38
20	82	1.31	199	3.18	200	3.2	171	2.73
21	341	5.45	906	14.49	1 016	16.25	645	10.32
22	1 214	19.42	1 706	27.28	1 947	31.14	1 414	22.61
23	1 849	29.57	1 716	27.44	1 963	31.39	1 640	26.24
24	1 594	25.49	1 063	17.00	909	14.54	1 130	18.07
25	839	13.42	430	6.88	189	3.02	745	11.91
26	278	4.45	117	1.87	27	0.43	361	5.77
27	48	0.77	37	0.59	2	0.03	113	1.81
28	2	0.03	21	0.34	0	0.0	10	0.16
29	2	0.03	19	0.3	0	0.0	0	0.0
30	0	0.0	3	0.05	0	0.0	0	0.0
Total	6 253	100.0						

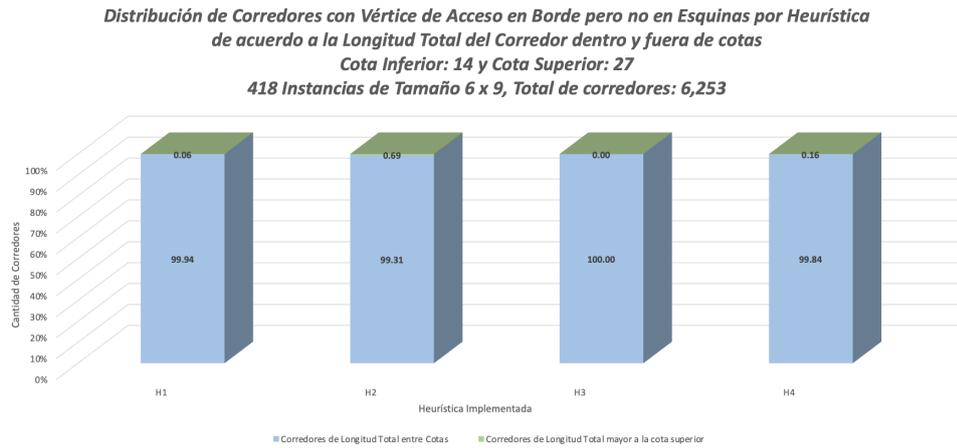


Figura 5.40: Distribución de corredores en instancias de tamaño 6×9 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.41 se puede apreciar la distribución de los corredores generados por las heurísticas en un rango de longitud de 19 hasta 27. Las heurísticas H1, H2 y H4 aportan la mejor solución para 4, 36 y 24 corredores con una longitud de 19 respectivamente. En la gráfica de la Figura 5.42 se presenta la cantidad de corredores que tienen una longitud mayor a la cota superior. Con una longitud de 28 la heurística H1 generó 2 corredores, la heurística H2 generó 21 corredores y la heurística H4 generó 10 corredores. Con una longitud de 29 la heurística H1 generó 2 corredores y la heurística H2 generó 19 corredores. Mientras que de una longitud de 30 la heurística H2 generó 3 corredores.

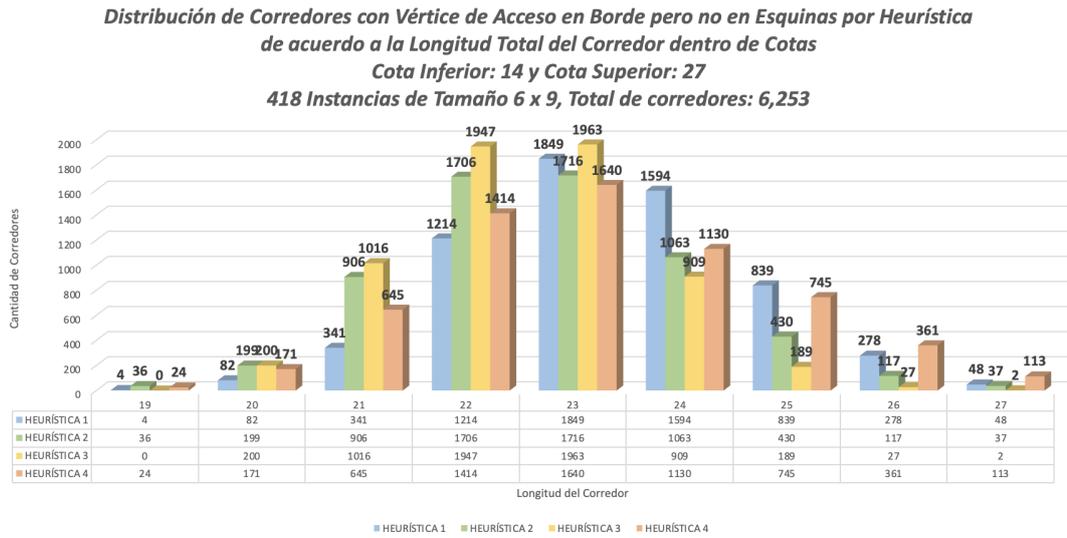


Figura 5.41: Distribución de corredores en instancias de tamaño 6 × 9 con longitudes dentro de las cotas.

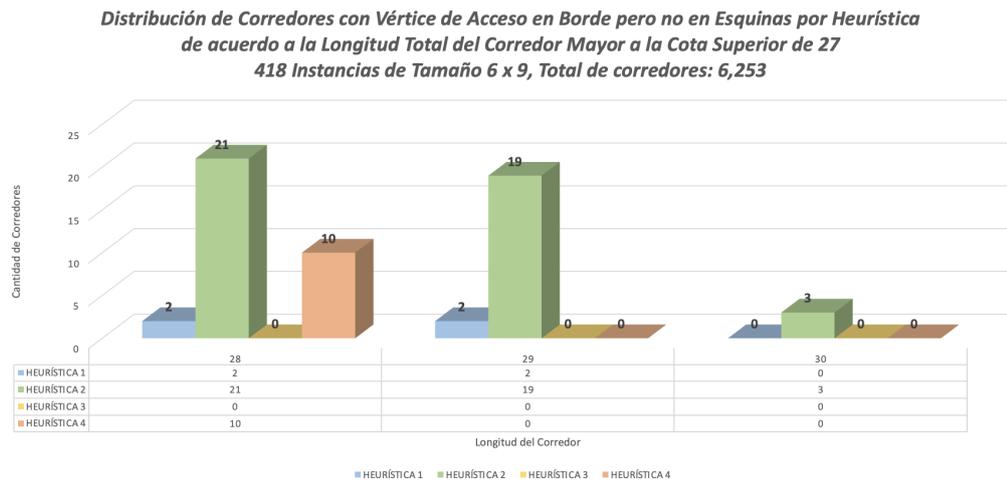


Figura 5.42: Distribución de corredores en instancias de tamaño 6 × 9 con longitudes mayor a la cota superior.

Instancias de tamaño 6 × 10.

La instancia de tamaño 6 × 10 consiste en 242 configuraciones con 16 nodos en el borde y 173 configuraciones con 17 nodos en el borde, sin considerar las esquinas en cada una de ellas. Ello produce 6 813 diferentes corredores,

cada uno teniendo un vértice del borde sin considerar las esquinas como nodo de acceso. Los 6 813 corredores producidos por la heurística H3 tienen una longitud total dentro de las cotas inferior y superior de 16 y 30, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.21). Sin embargo, algunos corredores producidos por las heurísticas H1, H2 y H4 tienen una longitud total que excede la cota superior (ver Figura 5.43).

Tabla 5.21: Cantidad y porcentaje de corredores en instancias de tamaño 6×10 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
16	0	0.0	0	0.0	0	0.0	0	0.0
17	0	0.0	0	0.0	0	0.0	0	0.0
18	0	0.0	0	0.0	0	0.0	0	0.0
19	0	0.0	0	0.0	0	0.0	0	0.0
20	0	0.0	0	0.0	0	0.0	0	0.0
21	0	0.00	13	0.19	7	0.10	0	0.00
22	26	0.38	95	1.39	170	2.50	119	1.75
23	124	1.82	548	8.04	760	11.2	523	7.68
24	741	10.88	1 357	19.92	1 906	27.90	1 182	17.35
25	1 692	24.83	1 756	25.78	1 996	29.30	1 606	23.57
26	2 033	29.84	1 545	22.68	1 329	19.50	1 368	20.09
27	1 453	21.33	794	11.65	500	7.30	1 011	14.84
28	578	8.48	374	5.49	120	1.80	621	9.11
29	143	2.10	145	2.13	25	0.40	221	3.24
30	21	0.31	77	1.13	0	0.00	99	1.45
31	2	0.03	46	0.68	0	0.00	55	0.81
32	0	0.00	35	0.51	0	0.00	8	0.11
33	0	0.00	21	0.31	0	0.00	0	0.00
34	0	0.00	6	0.09	0	0.00	0	0.00
35	0	0.00	1	0.01	0	0.00	0	0.00
Total	6 813	100.0						

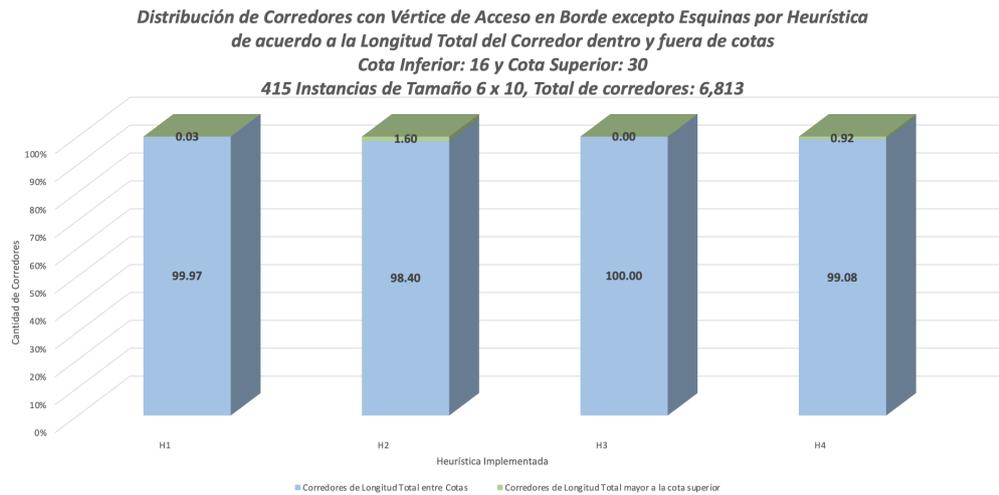


Figura 5.43: Distribución de corredores en instancias de tamaño 6×10 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.44 se puede apreciar la distribución de los corredores generados por las heurísticas en un rango de longitud de 21 hasta 30. Las heurísticas H2 y H3 aportan la mejor solución para 13 y 7 corredores con una longitud de 21 respectivamente. En la gráfica de la Figura 5.45 se presenta la cantidad de corredores que tienen una longitud mayor a la cota superior. Con una longitud de 31 la heurística H1 generó 2 corredores, la heurística H2 generó 46 corredores y la heurística H4 generó 55 corredores. Con una longitud de 32 la heurística H2 generó 35 corredores y la heurística H4 generó 8 corredores. Con una longitud de 33 la heurística H2 generó 21 corredores. Con una longitud de 34 la heurística H2 generó 6 corredores. Mientras que de una longitud de 35 la heurística H2 generó 1 corredor.

Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 16 y Cota Superior: 30
415 Instancias de Tamaño 6 x 10, Total de corredores: 6,813

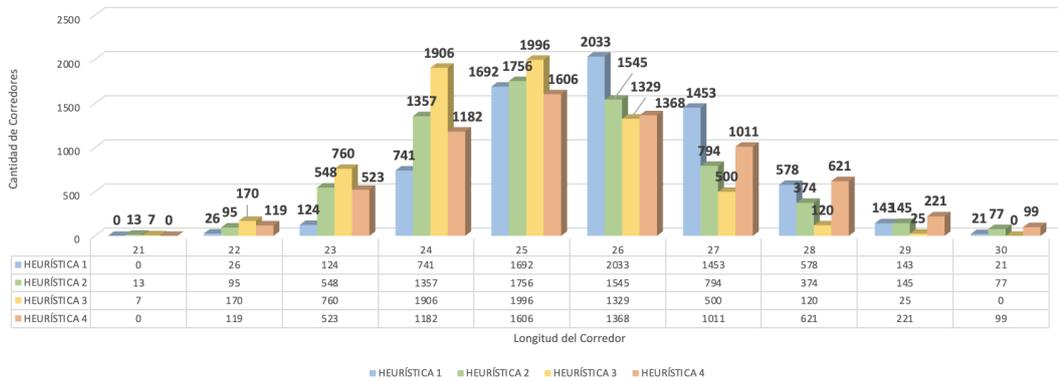


Figura 5.44: Distribución de corredores en instancias de tamaño 6 x 10 con longitudes dentro de las cotas.

Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor Mayor a la Cota Superior de 30
415 Instancias de Tamaño 6 x 10, Total de corredores: 6,813

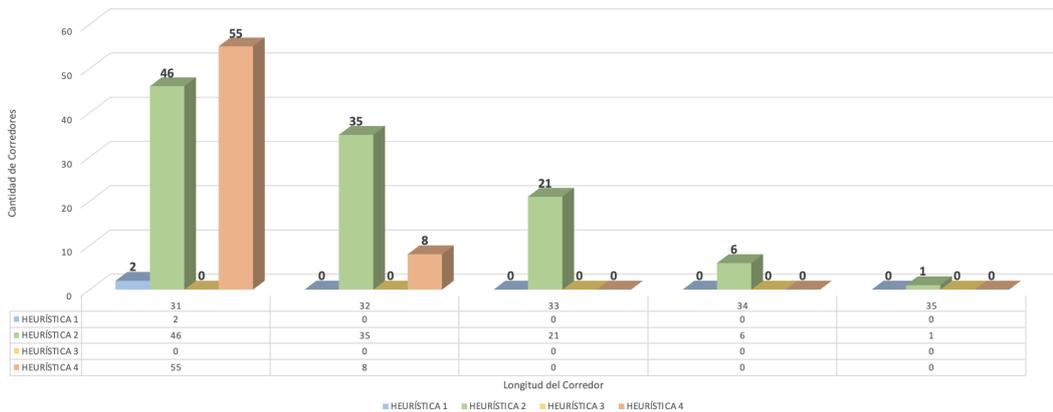


Figura 5.45: Distribución de corredores en instancias de tamaño 6 x 10 con longitudes mayor a la cota superior.

Instancias de tamaño 7 x 8.

La instancia de tamaño 7x8 consiste en 528 configuraciones con 14 nodos en el borde, 1 492 configuraciones con 15 nodos en el borde, 1 131 configuraciones con 16 nodos en el borde y 240 configuraciones con 17 nodos en el borde,

sin considerar las esquinas en cada una de ellas. Ello produce 51 948 diferentes corredores, cada uno teniendo un vértice del borde sin considerar las esquinas como nodo de acceso. Los 51 948 corredores producidos por la heurística H3 tienen una longitud total dentro de las cotas inferior y superior de 15 y 28, respectivamente, para esta instancia, de acuerdo al Teorema 5.3 (ver Tabla 5.22). Sin embargo, algunos corredores producidos por las heurísticas H1, H2 y H4 tienen una longitud total que excede la cota superior (ver Figura 5.46).

Tabla 5.22: Cantidad y porcentaje de corredores en instancias de tamaño 7×8 por longitud para cada heurística.

Longitud	H1		H2		H3		H4	
	Cantidad	%	Cantidad	%	Cantidad	%	Cantidad	%
15	0	0.0	0	0.0	0	0.0	0	0.0
16	0	0.0	0	0.0	0	0.0	0	0.0
17	0	0.0	0	0.0	0	0.0	0	0.0
18	0	0.0	0	0.0	0	0.0	0	0.0
19	0	0.0	2	0.004	13	0.03	0	0.0
20	64	0.12	248	0.477	747	1.44	89	0.171
21	967	1.86	2 828	5.444	5 406	10.40	1 258	2.422
22	5 238	10.08	9 025	17.373	13 954	26.86	4 148	7.985
23	12 472	24.01	13 700	26.373	16 442	31.65	8 564	16.486
24	15 470	29.78	12 373	23.818	10 274	19.78	12 161	23.41
25	11 382	21.91	7 381	14.208	4 051	7.80	11 494	22.126
26	4 640	8.93	3 334	6.418	950	1.83	7 850	15.111
27	1 286	2.48	1 421	2.735	106	0.20	4 276	8.231
28	330	0.64	728	1.4	5	0.01	1 573	3.028
29	79	0.15	511	0.985	0	0.0	429	0.826
30	20	0.04	220	0.424	0	0.0	92	0.177
31	0	0.0	66	0.127	0	0.0	12	0.023
32	0	0.0	42	0.081	0	0.0	2	0.004
33	0	0.0	25	0.048	0	0.0	0	0.0
34	0	0.0	31	0.06	0	0.0	0	0.0
35	0	0.0	11	0.021	0	0.0	0	0.0
36	0	0.0	2	0.004	0	0.0	0	0.0
Total	51 948	100.0						

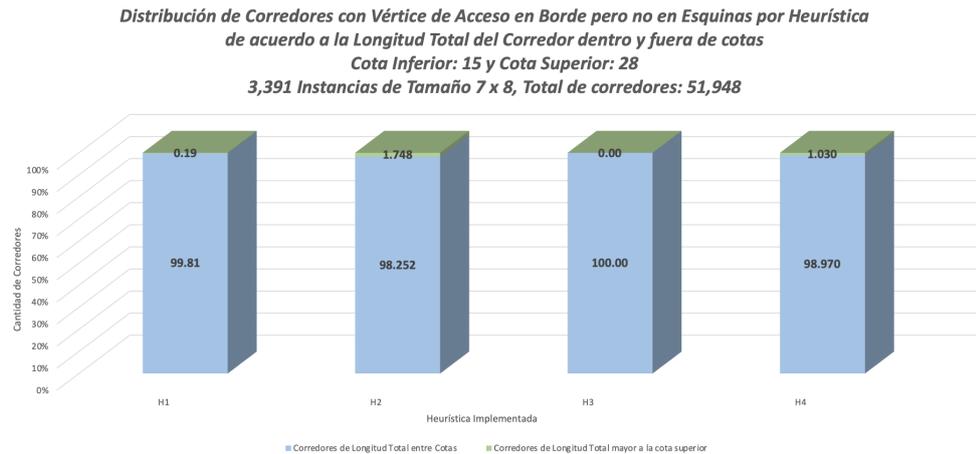


Figura 5.46: Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro y fuera de las cotas.

En la gráfica de la Figura 5.47 se puede apreciar la distribución de los corredores generados por las heurísticas en un rango de longitud de 19 hasta 28. Las heurísticas H2 y H3 aportan la mejor solución para 2 y 13 corredores con una longitud de 19 respectivamente. En la gráfica de la Figura 5.48 se presenta la cantidad de corredores que tienen una longitud mayor a la cota superior. Con una longitud de 29 la heurística H1 generó 79 corredores, la heurística H2 generó 511 corredores y la heurística H4 generó 429 corredores. Con una longitud de 30 la heurística H1 generó 20 corredores, la heurística H2 generó 220 corredores y la heurística H4 generó 92 corredores. Con una longitud de 31 la heurística H2 generó 66 corredores y la heurística H4 generó 12 corredores. Con una longitud de 32 la heurística H2 generó 42 corredores y la heurística H4 generó 2 corredores. Con una longitud de 33 la heurística H2 generó 25 corredores. Con una longitud de 34 la heurística H2 generó 31 corredores. Con una longitud de 35 la heurística H2 generó 11 corredores. Mientras que de una longitud de 36 la heurística H2 generó 2 corredores.

Distribución de Corredores con Vértice de Acceso en Borde pero no en Esquinas por Heurística de acuerdo a la Longitud Total del Corredor dentro de Cotas
Cota Inferior: 15 y Cota Superior: 28
3,391 Instancias de Tamaño 7 x 8, Total de corredores: 51,948

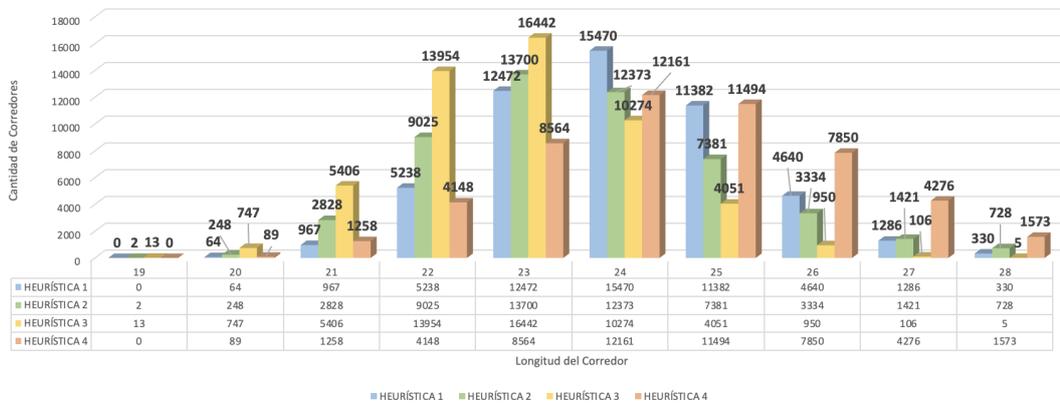


Figura 5.47: Distribución de corredores en instancias de tamaño 7×8 con longitudes dentro de las cotas.

Distribución de Corredores con Vértice de Acceso en Borde por Heurística Fuera Cota: $length(Corredor) > 28$
3,391 Instancias de Tamaño 7 x 8, Total de corredores: 51,948

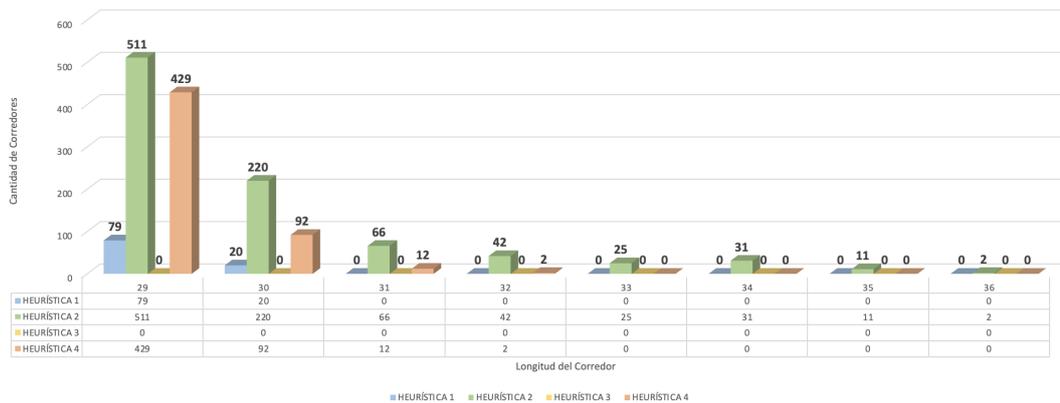


Figura 5.48: Distribución de corredores en instancias de tamaño 7×8 con longitudes mayor a la cota superior.

5.2.5. Tiempos de procesamiento por heurística.

Se probaron las cuatro heurísticas para las seis familias de instancias de tamaños 6×5 , 6×7 , 6×8 , 6×9 , 6×10 y 7×8 . Para ello, las heurísticas se implementaron en el lenguaje de programación de alto nivel ©Mathematica versión

11.3.0.0 en una computadora MacBook Pro con un CPU de 2.6 GHz Intel Core i7 con 6 núcleos y 16 GB de memoria RAM. Para cada configuración de empaquetados correspondiente a una familia de instancias se construyeron corredores para cada nodo de acceso localizado en la periferia. De dichos resultados, se tomaron los tiempos de ejecución mínimo t_{min} y máximo t_{max} . Asimismo, se calculó el promedio de los tiempo de ejecución en la generación de corredores para cada nodo de acceso de la periferia de la instancia $\overline{t_{prom}}$.

A continuación se muestran, para cada familia de instancias, los tiempos mínimo t_{min} , máximo t_{max} y promedio $\overline{t_{prom}}$ de cada configuración. Las Figuras 5.49-5.51, 5.52-5.54, 5.55-5.57, 5.58-5.60 y 5.61-5.63 corresponden a las instancias 6×7 , 6×8 , 6×9 , 6×10 y 7×8 , respectivamente.

Es importante destacar que la heurística 3 ofrece consistentemente el mayor rendimiento para todas las seis familias de instancias, esto es, el tiempo de ejecución mínimo, a la vez que ofrece los valores para la longitud total de corredor dentro de las cotas inferior y superior que se mencionó en la Sección 5.2.1 en los Teoremas 5.2 y 5.3 con respecto a las demás heurísticas.

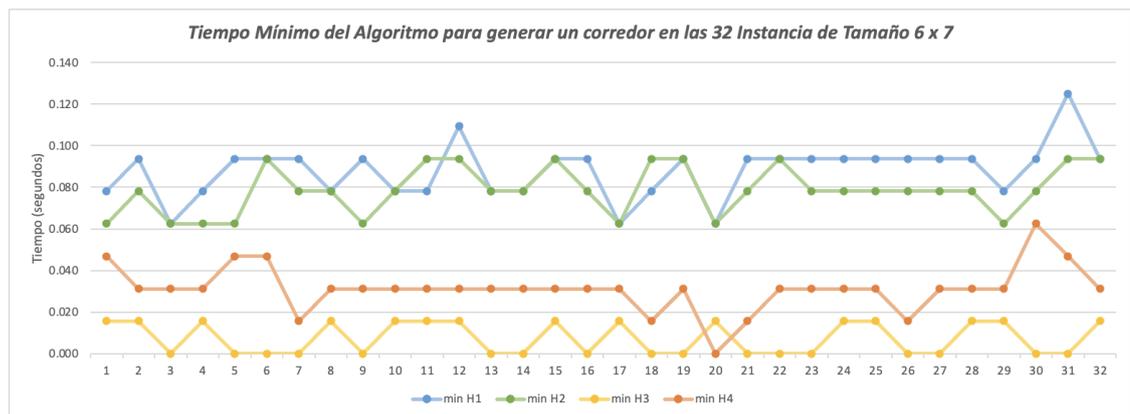


Figura 5.49: Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.

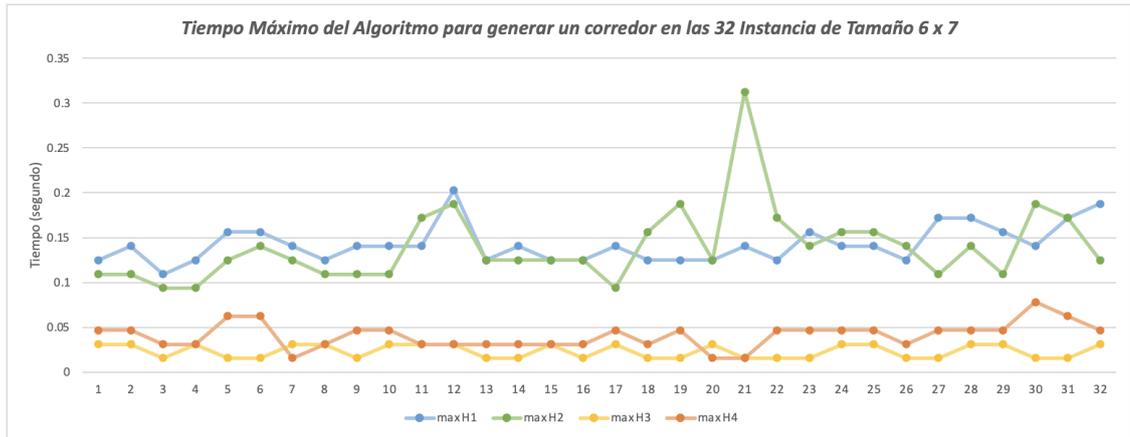


Figura 5.50: Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.

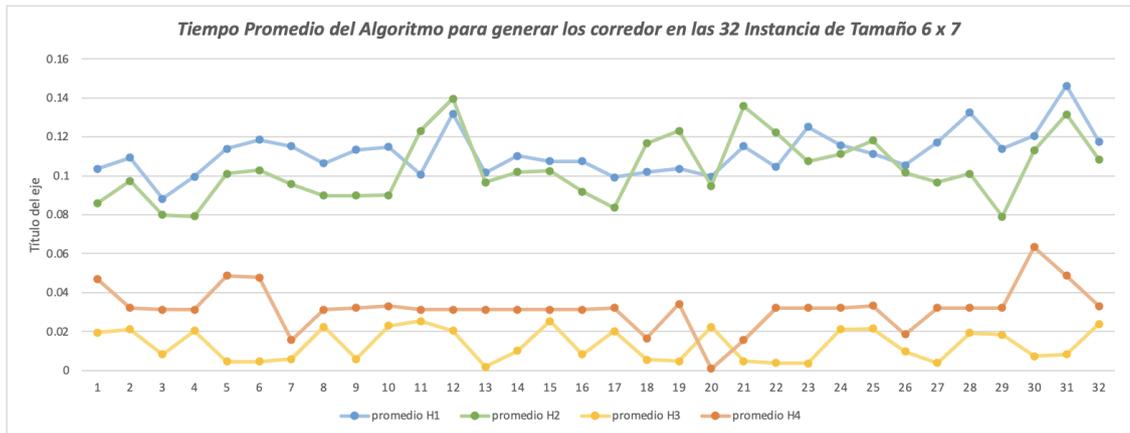


Figura 5.51: Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×7 para las cuatro heurísticas.

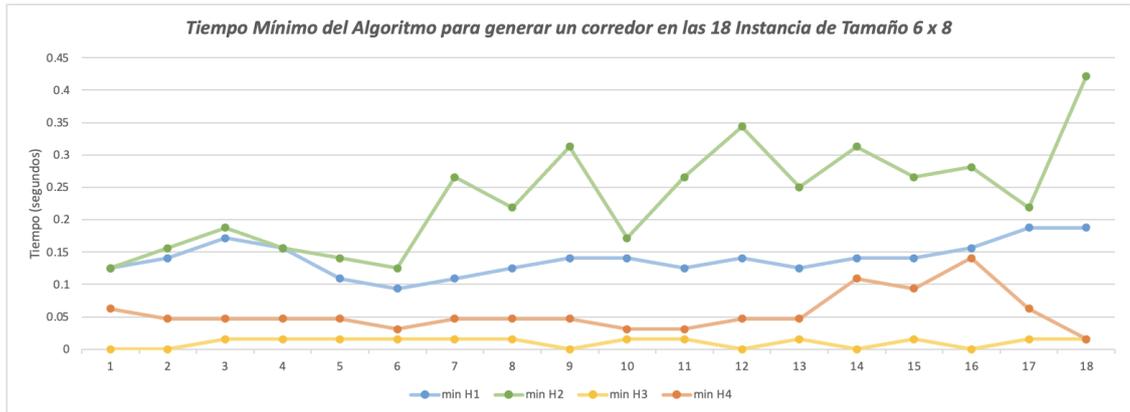


Figura 5.52: Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.

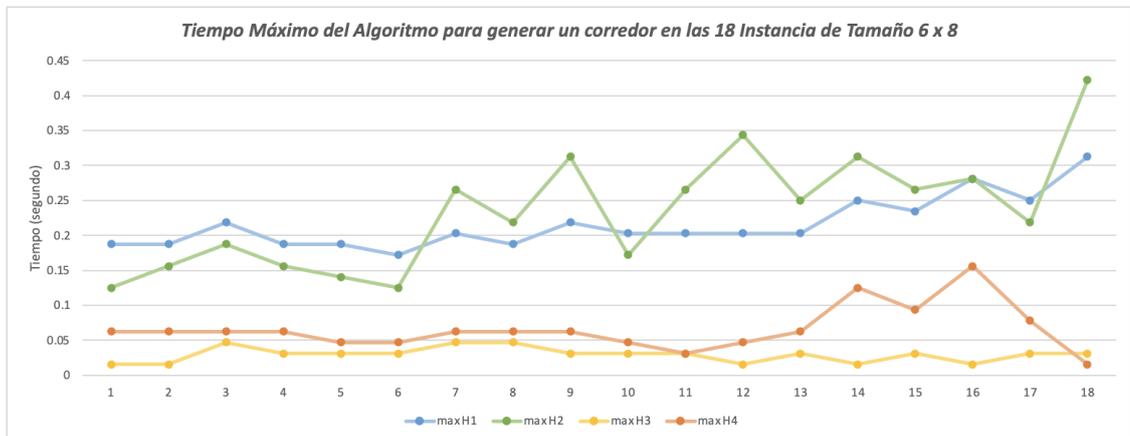


Figura 5.53: Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.

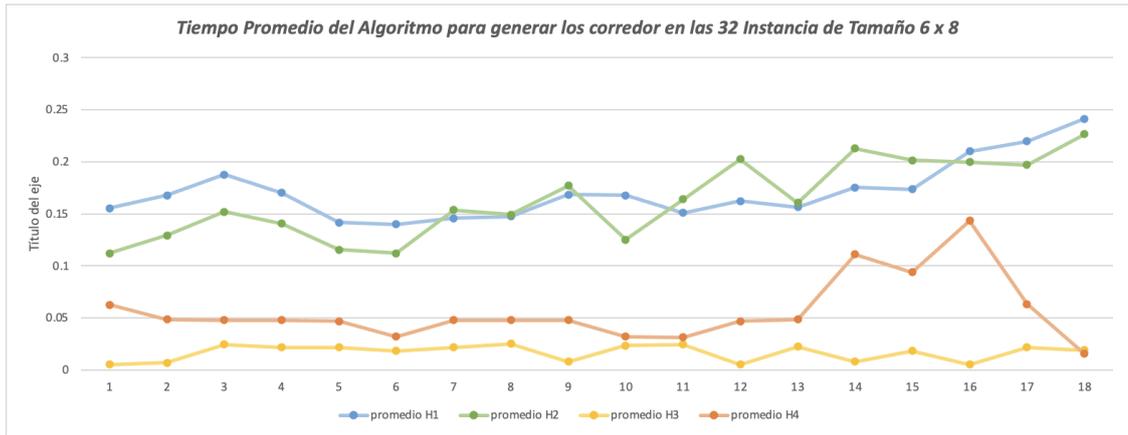


Figura 5.54: Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×8 para las cuatro heurísticas.

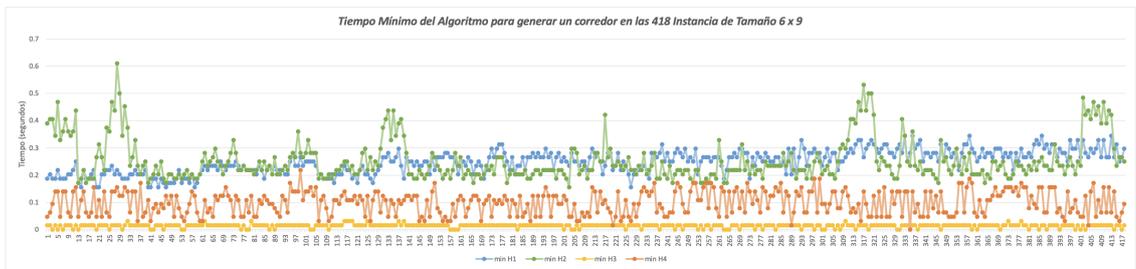


Figura 5.55: Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.

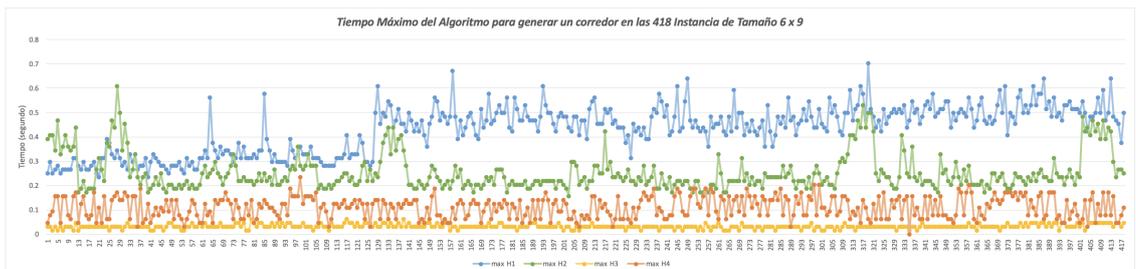


Figura 5.56: Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.

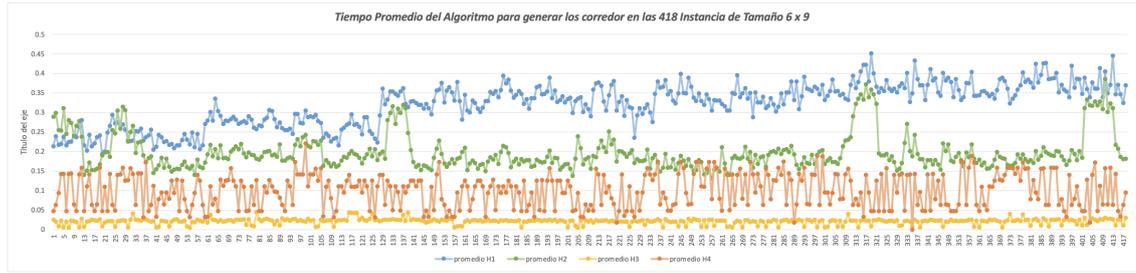


Figura 5.57: Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×9 para las cuatro heurísticas.

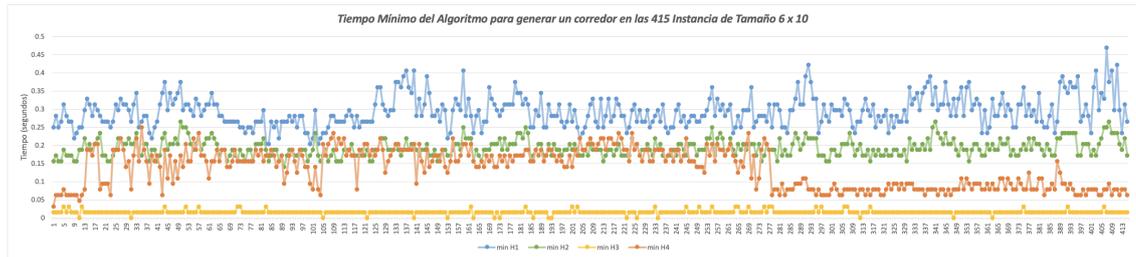


Figura 5.58: Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.

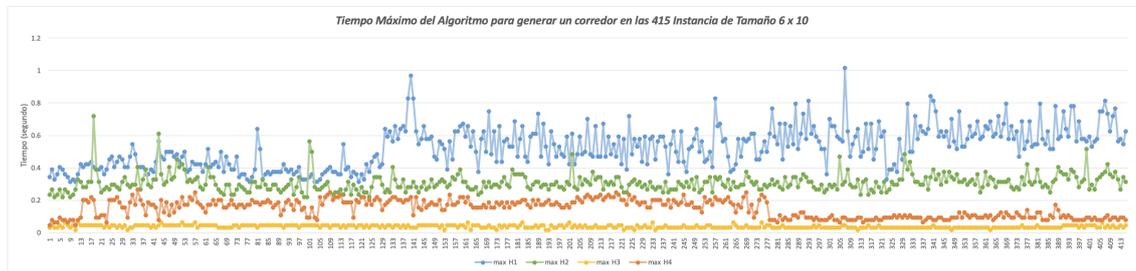


Figura 5.59: Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.

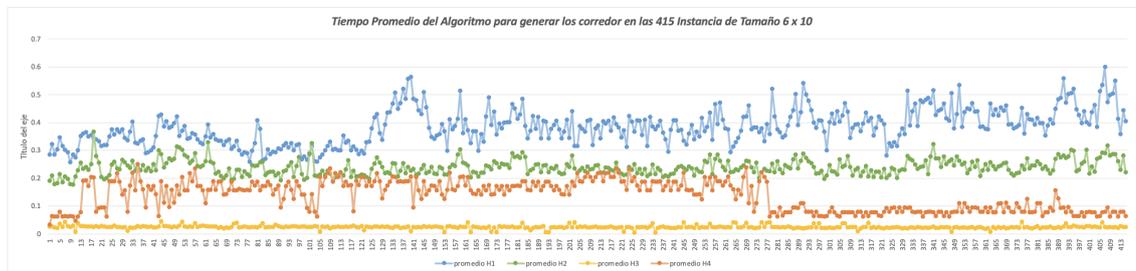


Figura 5.60: Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 6×10 para las cuatro heurísticas.

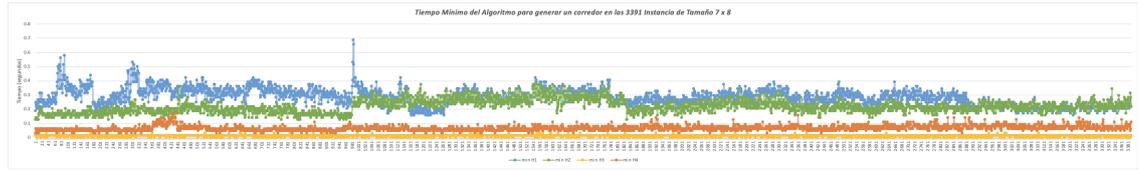


Figura 5.61: Tiempo Mínimo del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.



Figura 5.62: Tiempo Máximo del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.

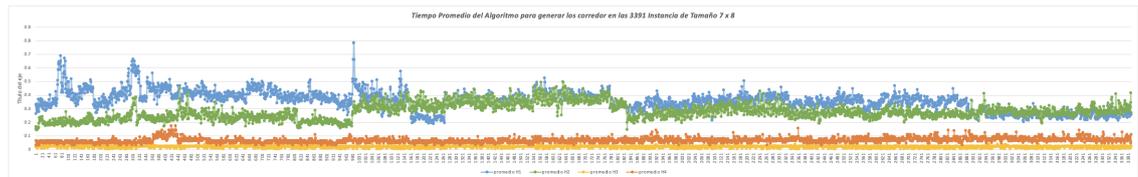


Figura 5.63: Tiempo Promedio del Algoritmo en la generación de un corredor en Instancias de tamaño 7×8 para las cuatro heurísticas.

En las Tablas 5.23, 5.24, 5.25 y 5.26 se presenta un análisis estadístico de los tiempos de ejecución obtenidos en la generación de las familias de las instancias por las cuatro heurísticas implementadas H1, H2, H3 y H4 respectivamente.

Para las seis familias de instancias de tamaños 6×5 , 6×7 , 6×8 , 6×9 , 6×10 y 7×8 se obtuvieron el promedio de los tiempos mínimo $\overline{t_{min}}$, máximo $\overline{t_{max}}$ y promedio $\overline{t_{prom}}$ de las configuraciones de cada familia de instancias; así como la desviación estándar de los tiempos mínimo $\sigma_{\overline{t_{min}}}$, máximo $\sigma_{\overline{t_{max}}}$ y promedio $\sigma_{\overline{t_{prom}}}$ de las configuraciones de cada familia de instancias. Se puede observar que la heurística H3 los tiempos de ejecución para la generación de los corredores en

su configuración para todas las familias de instancias geométricas tiene una dispersión baja entre sus tiempos por la desviación estándar obtenida. La heurística H4 es la siguiente en tener una dispersión baja mientras que las heurísticas H1 y H2 son las que presentan una mayor dispersión en los tiempos de ejecución para la obtención de los corredores en las configuraciones de las familias de instancias geométricas, se puede observar que esta va creciendo en ambos casos de acuerdo al incremento del tamaño de la familia de la instancia geométrica.

Tabla 5.23: Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 1 en la generación de la familia de corredores.

HEURÍSTICA 1						
Familia	\bar{t}_{min}	$\sigma_{\bar{t}_{min}}$	\bar{t}_{max}	$\sigma_{\bar{t}_{max}}$	\bar{t}_{prom}	$\sigma_{\bar{t}_{prom}}$
6 x 5	0.03125	0	0.0703125	0.011048543	0.041852679	0.003945908
6 x 7	0.087890625	0.01301243	0.142578125	0.020906709	0.11161894	0.011445638
6 x 8	0.139756944	0.025405841	0.216145833	0.036839308	0.17115162	0.027789344
6 x 9	0.249215012	0.041867199	0.432378888	0.097232777	0.321331826	0.053984152
6 x 10	0.293260542	0.042297316	0.525301205	0.125728754	0.383576718	0.062964317
7 x 8	0.284332608	0.058018867	0.450180625	0.104243353	0.355631283	0.07133202

Tabla 5.24: Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 2 en la generación de la familia de corredores.

HEURÍSTICA 2						
Familia	\bar{t}_{min}	$\sigma_{\bar{t}_{min}}$	\bar{t}_{max}	$\sigma_{\bar{t}_{max}}$	\bar{t}_{prom}	$\sigma_{\bar{t}_{prom}}$
6 x 5	0.03125	0	0.0625	0	0.044084821	0.000789182
6 x 7	0.078613281	0.011560167	0.139648438	0.042176765	0.103426847	0.016304856
6 x 8	0.118923611	0.024050733	0.234375	0.082679728	0.162808642	0.036920101
6 x 9	0.156100478	0.029772098	0.252691388	0.074459098	0.197992902	0.045175107
6 x 10	0.192281627	0.023065114	0.307304217	0.05394166	0.23925447	0.027330741
7 x 8	0.218630198	0.044099394	0.361992222	0.082606902	0.278218828	0.056445705

Tabla 5.25: Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 3 en la generación de la familia de corredores.

HEURISTICA 3						
Familia	$\overline{t_{min}}$	$\sigma_{\overline{t_{min}}}$	$\overline{t_{max}}$	$\sigma_{\overline{t_{max}}}$	$\overline{t_{prom}}$	$\sigma_{\overline{t_{prom}}}$
6 x 5	0	0	0.015625	0	0.004464286	0.001578363
6 x 7	0.007324219	0.00792199	0.0234375	0.007937508	0.013316761	0.008239993
6 x 8	0.010416667	0.007579238	0.029513889	0.010568743	0.016685957	0.007773
6 x 9	0.013494318	0.006636509	0.032633074	0.009173139	0.020933014	0.007121574
6 x 10	0.016114458	0.005353104	0.03814006	0.009477089	0.024924761	0.005939146
7 x 8	0.013026209	0.006396885	0.029743254	0.007549819	0.02012911	0.006776363

Tabla 5.26: Promedio (\bar{t}) y Desviación Estándar (σ) de los Tiempos Mínimos, Máximos y Totales de la Heurística 4 en la generación de la familia de corredores.

HEURISTICA 4						
Familia	$\overline{t_{min}}$	$\sigma_{\overline{t_{min}}}$	$\overline{t_{max}}$	$\sigma_{\overline{t_{max}}}$	$\overline{t_{prom}}$	$\sigma_{\overline{t_{prom}}}$
6 x 5	0.015625	0	0.0234375	0.011048543	0.016741071	0.001578363
6 x 7	0.03125	0.011225331	0.041015625	0.014171286	0.032226563	0.011285941
6 x 8	0.055555556	0.030422038	0.065972222	0.032403152	0.056375386	0.030836602
6 x 9	0.099058014	0.04316043	0.110720694	0.043680488	0.100067035	0.043159105
6 x 10	0.136445783	0.052000108	0.14875753	0.052509443	0.137338826	0.052102803
7 x 8	0.065748489	0.01752059	0.077152757	0.018792722	0.066765917	0.017553339

En la Figura 5.64 se presenta el comportamiento que tiene el promedio de los tiempos mínimo $\overline{t_{min}}$, máximo $\overline{t_{max}}$ y promedio $\overline{t_{prom}}$ de las configuraciones de cada familia de instancias observando una mayor diferencia en el promedio de los tiempos mínimo $\overline{t_{min}}$ y máximo $\overline{t_{max}}$ en las heurísticas H1 y H2, mientras que las heurísticas H3 y H4 presentan una menor diferencia, en particular la heurística H3.

**PROMEDIO DE TIEMPOS DE EJECUCIÓN EN LA GENERACIÓN DE CORREDORES
POR CONJUNTO DE FAMILIAS Y HEURÍSTICAS
TIEMPO MÍNIMO, MÁXIMO Y PROMEDIO**

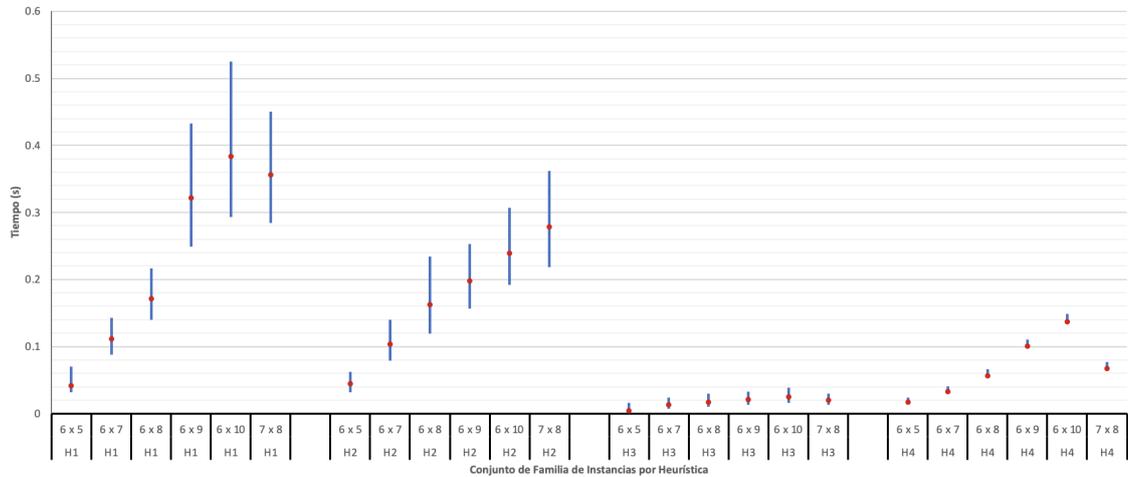


Figura 5.64: Comparación de Tiempos Promedios del Algoritmo en la generación de las familias de corredores para las cuatro heurísticas.

La Figura 5.65 se presenta el comportamiento que tiene el promedio de los tiempos mínimo $\overline{t_{min}}$, máximo $\overline{t_{max}}$ y promedio $\overline{t_{prom}}$ para cada familia de instancias haciendo la comparación con los resultados obtenidos por cada una de las heurísticas implementadas.

**PROMEDIO DE TIEMPOS DE EJECUCIÓN EN LA GENERACIÓN DE CORREDORES POR FAMILIA
CON DIFERENTES HEURÍSTICAS
TIEMPO MÍNIMO, MÁXIMO Y PROMEDIO**

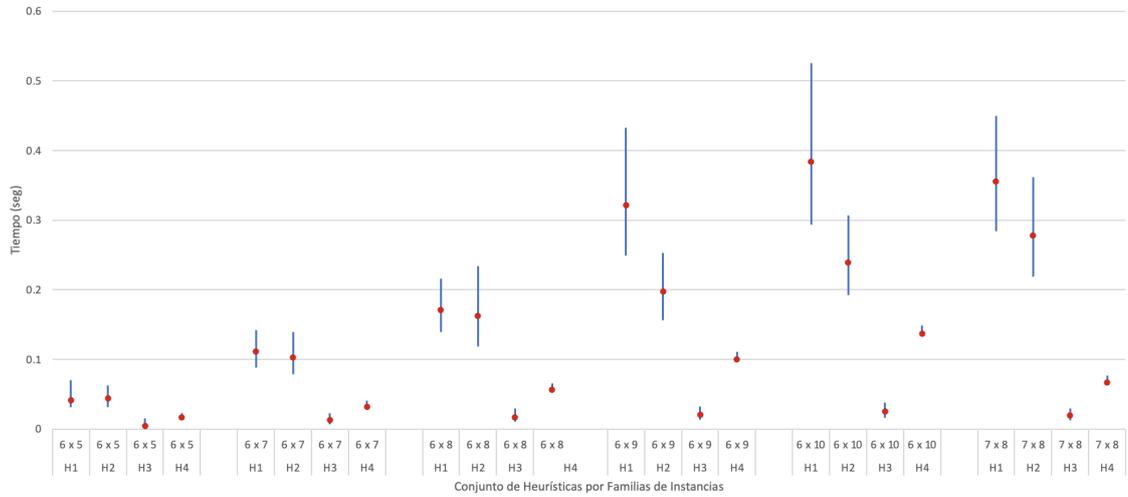


Figura 5.65: Comparación de Tiempos Promedios en la generación de las familias de corredores para las cuatro heurísticas.

Mientras que las Figuras 5.66 y 5.67 presentan la comparación para familias de embalados con un valor de $n = 6$ y $n = 7$ respectivamente considerando que hay mayor similitud entre estas familias.

PROMEDIO DE TIEMPOS DE EJECUCIÓN EN LA GENERACIÓN DE CORREDORES DE LAS FAMILIAS DE TAMAÑO 6X5, 6X7, 6X8, 6X9 Y 6X10 POR HEURÍSTICAS TIEMPO MÍNIMO, MÁXIMO Y PROMEDIO

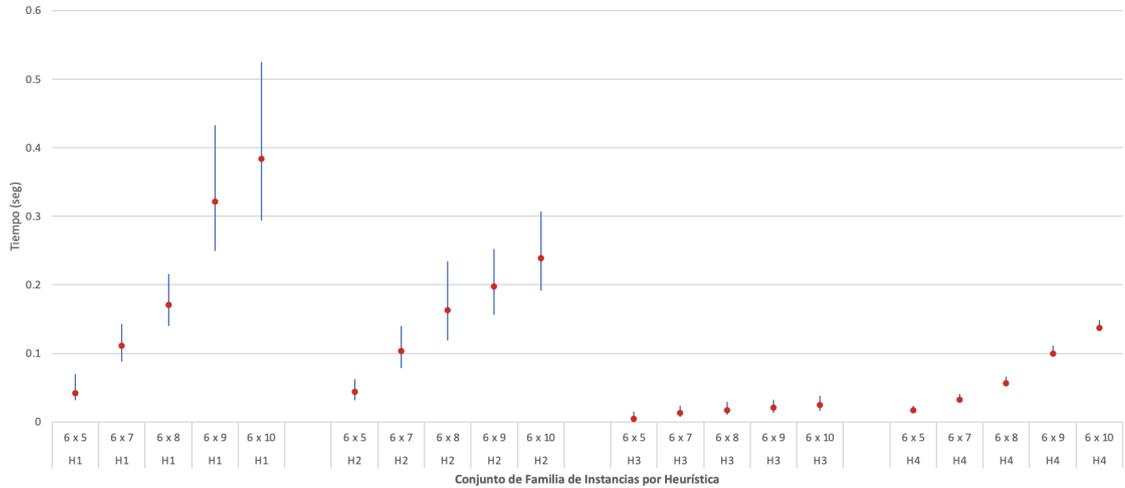


Figura 5.66: Comparación de Tiempos Promedios del Algoritmo en la generación de corredores para las familias de tamaño 6×5 , 6×7 , 6×8 , 6×9 y 6×10 para las cuatro heurísticas.

PROMEDIO DE TIEMPOS DE EJECUCIÓN EN LA GENERACIÓN DE CORREDORES DE LAS FAMILIAS DE TAMAÑO 7X6 Y 7X8 POR HEURÍSTICAS TIEMPO MÍNIMO, MÁXIMO Y PROMEDIO

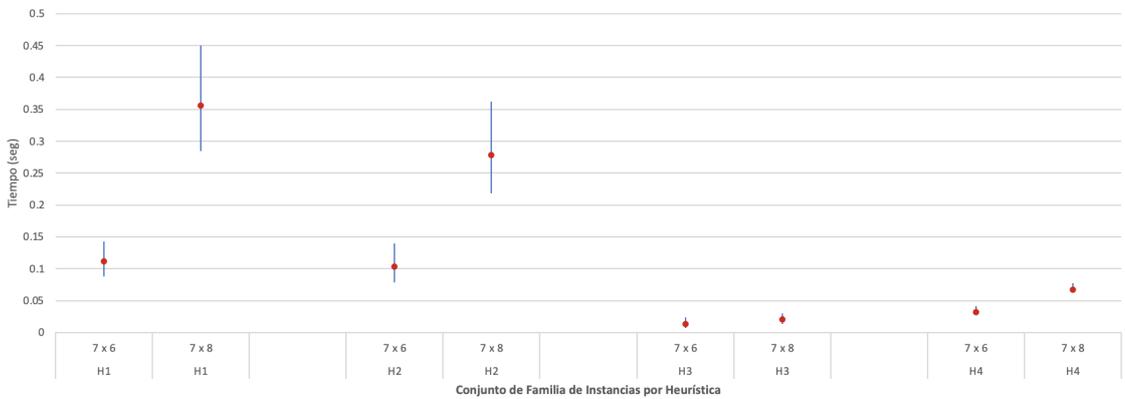


Figura 5.67: Comparación de Tiempos Promedios del Algoritmo en la generación de corredores para las familias de tamaño 7×6 y 7×8 para las cuatro heurísticas.

”Un gran descubrimiento resuelve un gran problema, pero hay una pizca de descubrimiento en la solución de cualquier problema. Tu problema puede ser modesto, pero si es un reto a tu curiosidad y trae a juego tus facultades inventivas, y si lo resuelves por tus propios métodos, puedes experimentar la tensión y disfrutar del triunfo del descubrimiento.”

*George Pólya (1887-1985)
Matemático húngaro*

Conclusiones y trabajos futuros.

6.1. Conclusiones.

En el desarrollo de esta investigación para encontrar una solución cuasi-óptima al problema del corredor de longitud mínima en rectángulos particionado en dominós de tamaño $n \times m$ se ha contribuido con resultados en dos áreas importantes de las ciencias de la computación: Combinatoria y Algoritmos.

Se construyó un conjunto de instancias geométricas de embaldosados de tamaño $n \times m$ con características especiales, estas son: no-isomorfos y libres de corte, con la finalidad de contar con un conjunto de datos experimentales para evaluar la eficiencia de los algoritmos de ruteo en superficies de embaldosados rectilíneos.

Dos aspectos son importantes al estudiar la combinatoria de un problema; primero, saber cuántas posibles soluciones existen para un problema. En segundo lugar, saber cuáles son esas posibles soluciones para un problema.

En la literatura podemos encontrar aportaciones de diversos autores que han determinado la forma para calcular cuantos embaldosados se pueden generar. El cálculo del número $t(n, m)$ de embaldosados de un rectángulo $n \times m$, incluyendo isomorfos bajo la operación de rotación a lo largo de los dos ejes de

simetría, los dos ejes diagonales y rotaciones alrededor de su centro, muestran que su número crece exponencialmente de acuerdo a (Ecuación 3.1). La cantidad de embaledos así como la caracterización simbólica de cada uno de ellos puede calcularse mediante el determinante de una matriz que surge al bicolorar (blanco y negro) el rectángulo, y sustituir algunas de las entradas por un número complejo multiplicado por una variable. Sin embargo, este procedimiento es muy lento comparado con el enfoque de backtracking para generar, eliminar isomorfos, y seleccionar embaledos libres de líneas de corte.

El diseño y la implementación de algoritmos utilizando la técnica de backtracking representa una gran aportación en este trabajo, contribuyendo con una herramienta computacional para la generación de las instancias geométricas de embaledos de tamaño $n \times m$. Con dicha herramienta, no solamente se puede saber cuántos embaledos existen, sino también conocer la configuración geométrica exacta de cada uno de ellos. Esto último, de mucha importancia para conformar un conjunto completo de instancias de prueba para los algoritmos de estudio. Adicionalmente, y dado que la cantidad de embaledos crece exponencialmente en número, se han desarrollado algoritmos para filtrar dicho conjunto. Esto permite el mejor manejo de instancias ya que se seleccionan solamente aquellas que son esencialmente únicas (no-isomorfas) y con una configuración tal que, para efectos de ruteo, no permitan conexiones simples como las que aportarían las líneas de corte horizontal y/o vertical sobre la instancia.

De acuerdo a los resultados que se presentan en la sección 5.2 sobre el desempeño de las cuatro heurísticas implementadas para la generación de los corredores de longitud mínima de las familias de instancias de embaledos con dómicos $I_{6 \times 5}$, $I_{6 \times 7}$, $I_{6 \times 8}$, $I_{6 \times 9}$, $I_{6 \times 10}$ y $I_{7 \times 8}$, podemos considerar dos aspectos importantes en los resultados: Longitud del Corredor y Tiempo de ejecución.

Se puede observar que la heurística H3 es la que proporciona los mejores resultados con corredores dentro de las cotas inferior y superior establecidos para los nodos de acceso en esquinas y en bordes (sin considerar las esquinas) de acuerdo a los Teoremas 5.2 y 5.3.

En la sección previa se puede observar que en el caso de los corredores con nodo de acceso en alguna esquina, las heurísticas H1, H2 y H4 generan corredores con una longitud total menor al mejor resultado que genera la heurística H3, como a continuación se detalla. Para la familia de instancias de embaledosados con d6minos $I_{6 \times 5}$, la heurística H1 gener6 1 corredor (12.5 %), la heurística H2 gener6 2 corredores (25 %) y la heurística H4 gener6 6 corredores (75 %). Para la familia de instancias de embaledosados con d6minos $I_{6 \times 7}$, la heurística H1 gener6 3 corredores (2.3 %). Para la familia de instancias de embaledosados con d6minos $I_{6 \times 8}$, la heurística H2 gener6 1 corredor (1.4 %) y la heurística H4 gener6 1 corredor (1.4 %). Para la familia de instancias de embaledosados con d6minos $I_{6 \times 9}$, la heurística H4 gener6 3 corredores (0.2 %). Como se puede observar, en general, son porcentajes bajos de corredores generados por las heurísticas H1, H2 y H4 que son mejores a los generados por H3, a excepci6n de la H4 para $I_{6 \times 5}$.

Por otro lado, la heurística H1 es la que genera un porcentaje m6s bajo de corredores con una longitud total mayor a la cota superior para cuatro familias de instancias de embaledosados con d6minos: $I_{6 \times 7}$ gener6 1 corredor(0.8 %), $I_{6 \times 9}$ gener6 16 corredores (1.0 %), $I_{6 \times 10}$ gener6 5 corredores (0.3 %) y $I_{7 \times 8}$ gener6 53 corredores (0.4 %). La heurística H2 es la que genera un porcentaje m6s alto de corredores con una longitud total mayor a la cota superior para las seis familias de instancias de embaledosados con d6minos: $I_{6 \times 5}$ gener6 3 corredores (37.5 %), $I_{6 \times 7}$ gener6 15 corredores (11.7%), $I_{6 \times 8}$ gener6 3 corredores (4.2%), $I_{6 \times 9}$ ge-

neró 95 corredores (5.7%), $I_{6 \times 10}$ generó 114 corredores (6.9%) y $I_{7 \times 8}$ generó 784 corredores (5.8%). La heurística H4 es la que genera un porcentaje medio de corredores con una longitud total mayor a la cota superior para cinco familias de instancias de embaldosados con dómicos: $I_{6 \times 7}$ generó 3 corredores (2.3%), (excepto $I_{6 \times 8}$ que generó 4 corredores (5.6%)), $I_{6 \times 9}$ generó 47 corredores (2.8%), $I_{6 \times 10}$ generó 42 corredores (2.5%) y $I_{7 \times 8}$ generó 683 corredores (5.0%).

En el caso de los corredores con nodo de acceso en el borde (sin considerar las esquinas), las heurísticas H1, H2 y H4 generan corredores con una longitud total menor al mejor resultado que genera la heurística H3. Para la familia de instancias de embaldosados con dómicos $I_{6 \times 5}$ la heurística H1 generó 6 corredores (30%), la heurística H2 generó 12 corredores (60%) y la heurística H4 generó 17 corredores (85%) teniendo un porcentaje alto. Para la familia de instancias de embaldosados con dómicos $I_{6 \times 9}$ la heurística H1 generó 4 corredores (0.06%), la heurística H2 generó 36 corredores (0.58%) y la heurística H4 generó 24 corredores (0.38%) teniendo un porcentaje bajo.

Por otro lado, la heurística H1 es la que genera un porcentaje más bajo de corredores con una longitud total mayor a la cota superior para cuatro familias de instancias de embaldosados con dómicos: $I_{6 \times 7}$ generó 2 corredores (0.5%), $I_{6 \times 9}$ generó 4 corredores (0.06%), $I_{6 \times 10}$ generó 2 corredores (0.03%) y $I_{7 \times 8}$ generó 99 corredores (0.19%). La heurística H2 es la que genera un porcentaje más alto de corredores con una longitud total mayor a la cota superior para las seis familias de instancias de embaldosados con dómicos: $I_{6 \times 7}$ generó 3 corredores (0.8%), $I_{6 \times 8}$ generó 4 corredores (1.6%), $I_{6 \times 9}$ generó 43 corredores (0.69%), $I_{6 \times 10}$ generó 109 corredores (1.6%) y $I_{7 \times 8}$ generó 908 corredores (1.939%). La heurística H4 es la que genera un porcentaje medio de corredores con una longitud total mayor a la cota superior para tres familias de instancias de embaldosados con dómicos:

(excepto $I_{6 \times 8}$ que generó 5 corredores (2 %)), $I_{6 \times 9}$ generó 10 corredores (0.16 %), $I_{6 \times 10}$ generó 63 corredores (0.92 %) y $I_{7 \times 8}$ generó 535 corredores (1.03 %).

Con respecto al tiempo de ejecución para las cuatro heurísticas implementadas, se puede observar de manera general que para cada familia de instancias y sus configuraciones que el desempeño en tiempo de ejecución de manera creciente de las cuatro heurísticas implementadas fue primero la heurística H3, después la heurística H4, posteriormente la heurística H2 y finalmente la heurística H1.

De modo que la heurística H3 presenta el mejor rendimiento para la generación de los corredores a partir de cualquier nodo de acceso en la periferia de cada configuración para todas las familias de embaledados. Es recomendable su uso en aquellas aplicaciones en línea donde es importante tener una respuesta rápida.

Las cuatro heurísticas pueden ser ejecutadas en paralelo cuando no es una prioridad el tiempo de ejecución, seleccionando al final de la ejecución de los programas el corredor con la longitud total mínima.

6.2. Líneas de investigación.

Se sugieren varias líneas de investigación de acuerdo a las técnicas usadas y resultados obtenidos en este trabajo. Primeramente, la metodología se puede extender para baldosas de diferentes tamaños.

Por otro lado, los algoritmos desarrollados en este trabajo también son útiles en la generación de instancias de embaledados de rectángulos con dominós para medir el rendimiento de algoritmos propuestos por otros autores para

resolver el Problema del Corredor de Longitud Mínima (*Minimum-Length Corridor Problem* – MLCP), el cual se ha demostrado que es un problema NP-completo [Gonzalez-Gutierrez and Gonzalez, 2007]. La familia de instancias de embalados con dominós representa un subproblema del problema MLC. Han demostrado [Gonzalez-Gutierrez and Gonzalez, 2010] que existe un algoritmo con una constante de aproximación de 30 para las instancias de particiones con rectángulos de diferentes tamaños. Así que, parece posible diseñar un algoritmo eficiente que produzca soluciones óptimas, o por lo menos un algoritmo con una razón de aproximación constante menor inclusive que 30 para el problema MLC con instancias de embalados con dominós.

A futuro se pretenden diseñar una mayor variedad de heurísticas, utilizando diferentes técnicas de diseño de algoritmos para encontrar mejores soluciones al problema del MLC en instancias geométricas. Así mismo, se habrán de considerar otro tipo de instancias geométricas, que por sus características presentan un reto importante en la aplicación de diferentes heurísticas.

Encontrar soluciones al problema del MLCP beneficiará a una amplia gama de aplicaciones que surgen en diferentes campos del conocimiento. Por ejemplo, el ruteo de transporte público, el diseño de circuitos integrados VLSI, los protocolos de evacuación de población debido a desastres naturales, el diseño de rutas de entrega de productos, entre otros. Finalmente, uno de los mayores retos es encontrar algoritmos eficientes de aproximación para un problema NP completo, como el MLCP, lo cual demanda un mayor esfuerzo de investigación.

Como se ha mencionado anteriormente, el problema MLC es NP completo. Por esta razón, no existe un algoritmo de tiempo polinomial que produzca solución óptima para cada instancia del problema. Por lo que se refiere al problema del corredor en embalados rectilíneos, no existe, hasta nuestro mayor

conocimiento, una demostración que establezca que el problema pertenece a la categoría de problemas NP-completos. Por lo que esto otorga una línea de investigación interesante para el futuro, ya que constituye un problema abierto. En este trabajo, en tanto, se ha demostrado que la solución óptima $f^*(I_D)$ para una instancia de dominó I_D , para el caso donde los nodos de acceso corresponden a los vértices de las esquinas de la instancia, se encuentra entre las cotas indicadas en la ecuación 6.1 y para el caso de nodos de acceso de borde (no esquinas) se encuentra entre las cotas indicadas en la ecuación 6.2.

$$\left\lfloor \frac{(n-1) \times (m-1)}{2} \right\rfloor \leq f^*(I_D) \leq \left\lceil \frac{n \times m}{2} \right\rceil \quad (6.1)$$

$$\left\lfloor \frac{(n-2) \times (m-2)}{2} \right\rfloor < f^*(I_D) \leq \left\lceil \frac{n \times m}{2} \right\rceil \quad (6.2)$$

A través de este criterio se puede llevar a cabo la evaluación de heurísticas en instancias geométricas de embaldosados con dominós.

Bibliografía

- [Applegate et al., 2011] Applegate, D., Bixby, R. E., Chvátal, V., and Cook, W. (2011). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.
- [Ardila and Stanley, 2004] Ardila, F. and Stanley, R. P. (2004). *Tilings*. Cambridge: MIT.
- [Ausiello et al., 2003] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., and Protasi, M. (2003). *Complexity and Approximation*. Springer Verlag.
- [Berger, 1966] Berger, R. (1966). The undecidability of the domino problem. *Mem. Amer. Math. Soc.*, 66:72.
- [Bodlaender et al., 2009] Bodlaender, H. L., Feremans, C., Grigoriev, A., Peninkx, E., Sitters, R., and Wolle, T. (2009). On the minimum corridor connection problem and other generalized geometric problems. *Computational Geometry*, 42(9):939–951.
- [Bondy and Murty, 2010] Bondy, J. A. and Murty, U. S. R. (2010). *Graph Theory. Graduate Texts in Mathematics*. Springer.
- [Cook, 2014] Cook, W. J. (2014). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.
- [Cormen et al., 2013] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2013). *Introduction to Algorithms*. International Edition. MIT Press.

- [Demaine and O'Rourke, 2000] Demaine, E. D. and O'Rourke, J. (2000). *Open Problems from CCCG 2000, Canadian Conference on Computational Geometry*. Canadian Conference on Computational Geometry.
- [Epp, 2012] Epp, S. S. (2012). *Matemáticas discretas con aplicaciones*. Cengage Learning Editores, S. A. de C. V. México.
- [Eppstein, 2001] Eppstein, D. (2001). Some open problems in graph theory and computational geometry.
- [Evans and Minieka, 1992] Evans, J. R. and Minieka, E. (1992). *Optimization algorithms for networks and graphs*. Dekker. New York.
- [Garey and Johnson, 2002] Garey, M. R. and Johnson, D. S. (2002). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.
- [Gaylord and Wellin, 1995] Gaylord, R. J. and Wellin, P. R. (1995). Computer simulations with mathematica. explorations in complex physical and biological systems. *Computer in Physics*, 10(4):297.
- [Gonzalez, 2014] Gonzalez, T. F. (2014). *Programming Assignment for CS130B: Data Structures and Algorithms*. University of California at Santa Barbara.
- [Gonzalez-Gutierrez and Gonzalez, 2007] Gonzalez-Gutierrez, A. and Gonzalez, T. F. (2007). Complexity of the minimum-length corridor problem. *Computational Geometry*, 37(2):72 – 103.
- [Gonzalez-Gutierrez and Gonzalez, 2010] Gonzalez-Gutierrez, A. and Gonzalez, T. F. (2010). Approximating corridors and tours via restriction and relaxation techniques. *ACM Trans. Algorithms*, 6(3):1 – 36.

- [González Gutiérrez et al., 2018] González Gutiérrez, F., González Gutiérrez, A., Elena, V. H. M., and Guillermo, D. D. (2018). Análisis experimental para el problema del corredor de longitud mínima en embaldosados rectilíneos. *Revista de la Ingeniería Industrial*, 12:30–39.
- [González Gutiérrez et al., 2019] González Gutiérrez, F., González Gutiérrez, A., Elena, V. H. M., and Guillermo, D. D. (2019). Generación del árbol extendido de costo mínimo modificado usando el algoritmo de vértices compartidos. *Visum Mundi*, 3:114–121.
- [Graham, 1981] Graham, R. L. (1981). *Fault-free Tilings of Rectangles*. Springer US, Boston, MA.
- [Grimaldi, 2004] Grimaldi, R. P. (2004). *Discrete and Combinatorial Mathematics: An Applied Introduction*. Pearson Addison Wesley.
- [Gutin and Punnen, 2007] Gutin, G. and Punnen, A. P. (2007). *The Traveling Salesman Problem and Its Variations*. Springer.
- [Hastings, 2006] Hastings, K. J. (2006). *Introduction to the Mathematics of Operations Research with Mathematica*. Chapman & Hall/CRC.
- [Huerta J., 2010] Huerta J., H. M. (2010). *Optimización de Rutas en Redes de Tráfico Vehicular*. Universidad Autónoma de Querétaro.
- [Hunter, 2009] Hunter, D. J. (2009). *Essentials of Discrete Mathematics*. Jones and Bartlett Publishers. Massachusetts, USA.
- [Kasteleyn, 1961] Kasteleyn, P. W. (1961). The statistics of dimers on a lattice: I. the number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225.

- [Kasteleyn, 1967] Kasteleyn, P. W. (1967). *Graph Theory and Theoretical Physics*, chapter Graph theory and crystal physics, pages 43–110. Academic Press, New York.
- [Kenyon and Okounkov, 2005] Kenyon, R. and Okounkov, A. (2005). What is ... a dimer? *Notices of the AMS*, 52(3):342–343.
- [Klarner and Pollack, 1980] Klarner, D. and Pollack, J. (1980). Domino tilings of rectangles with fixed width. *Discrete Mathematics*, 32(1):45–52.
- [Levitin, 2012] Levitin, A. (2012). *Introduction to the Design and Analysis of Algorithms*. Addison-Wesley. New Jersey.
- [Lieb, 1967] Lieb, E. H. (1967). Residual entropy of square ice. *Phys. Rev.*, 162:162–172.
- [Lovász, 1979] Lovász, L. (1979). *Combinatorial Problems and Exercises*. North Holland Publishing and Akadémiai Kiadó.
- [Montroll, 1964] Montroll, E. (1964). *Lattice Statistics*. John Wiley and Sons.
- [Pachter, 1997] Pachter, L. (1997). Combinatorial approaches and conjectures for 2-divisibility problems concerning domino tilings of polyominoes. *Electr. J. Comb.*, 4(1):Research paper R29, 10 p.
- [Paschos, 2014] Paschos, V. T. (2014). *Paradigms of Combinatorial Optimization: Problems and New Approaches Mathematics and Statistics*. Wiley-ISTE.
- [Percus, 1971] Percus, J. K. (1971). *Combinatorial Methods*, volume 4. Springer, New York.
- [Priyadarsini, 2007] Priyadarsini, P. L. K. (2007). An alternative proof for the np-completeness of top right access point-minimum length corridor problem. *International Journal of Mathematical, Physical and Engineering Sciences*, 2(2).

- [Propp, 2001] Propp, J. (2001). A reciprocity theorem for domino tilings. *The Electronic Journal of Combinatorics [electronic only]*, 8(1):Research paper R18, 9 p.
- [Read, 1980] Read, R. C. (1980). A note on tiling rectangles with dominoes. *Fibonacci Quarterly*, 18(1):24–27.
- [Rosen, 2004] Rosen, K. H. (2004). *Matemática discreta y sus aplicaciones*. McGraw Hill. Madrid.
- [Singh, 2015] Singh, V. (2015). *An Introduction to the Analysis of Algorithms*. Amazon Digital Service, Inc.
- [Strehl, 2001] Strehl, V. (2001). Counting domino tilings of rectangles via resultants. *Advances in Applied Mathematics*, 27(2-3):597–626.
- [Temperley and Fisher, 1961] Temperley, H. and Fisher, M. (1961). Dimer problem in statistical mechanics - an exact result. *Philos. Mag.*, 8(6):1061–1063.
- [Weiss, 2014] Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++*. Pearson.

A. Publicaciones en Revistas Indexadas

González Gutiérrez Fidel, González Gutiérrez Arturo, Vázquez Huerta Ma. Elena, Díaz Delgado Guillermo (2018). Análisis experimental para el problema del corredor de longitud mínima en embaldosados rectilíneos. *Revista de la Ingeniería Industrial*, Vol. 12, No. 1, pp 30-39. ISSN 1940-2163 online.

González Gutiérrez Fidel, González Gutiérrez Arturo, Vázquez Huerta Ma. Elena, Díaz Delgado Guillermo (2019). Generación del árbol extendido de costo mínimo modificado usando el algoritmo de vértices compartidos. *Visum Mundi*, Vol. 3, No. 2, pp. 114-121. ISSN 2572-8458 online.

B. Materiales: Datos, Figuras, Programas, CDF.

En el repositorio <https://github.com/FGL3zGtz/Material> se han colocado datos, figuras, programas y aplicaciones CDF.

Los datos corresponden a las representaciones matriciales de las familias de embaledosados. Las figuras consisten en: 1) la representación en forma de grafo de las configuraciones de embaledosados; 2) los trazos de los corredores de longitud mínima de las familias de embaledosados no isomorfos y libres de corte. Los programas escritos en *Mathematica*® corresponden a los algoritmos para: 1) la generación de configuraciones de embaledosados; 2) para la generación de corredores de longitud mínima. Las aplicaciones CDF (Computable Document Format) son demostraciones interactivas que le permiten al usuario visualizar los ruteos producidos por una heurística seleccionada sobre instancias de dominós también especificadas por el usuario.

Para tener acceso al material del repositorio enviar un correo electrónico al siguiente e-mail: fglez@uaq.mx para solicitar password de acceso.