Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Electrónica

Algoritmo para Automatizar el Enrutamiento de Señales en una Placa de Circuito Impreso Basado en Operaciones Morfológicas

Documento de tesis sometido a consideración para optar por el grado académico de Maestría en Electrónica con Énfasis en Sistemas Embebidos

Luis Carlos Álvarez Mata

Cartago, 8 de septiembre de 2022

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Maestría Académica en Electrónica
Trabajo Final de Graduación
Tribunal Evaluador
Acta de Aprobación de Tesis

Defensa del Trabajo Final de Graduación
Requisito para optar por el título de Máster en Ingeniería Electrónica
Grado Académico de Magister Scientiae

El Tribunal Evaluador aprueba la defensa del Trabajo Final de Graduación denominado **"Algoritmo para Automatizar el Enrutamiento de Señales en una Placa de Circuito Impreso Basado en Operaciones Morfológicas",** realizado por **Luis Carlos Álvarez Mata** Carné: **200902144**, y hace constar que cumple con las normas establecidas por la Unidad Interna de Posgrados de la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

RENATO RIMOLO DONADIO (FIRMA)
Firmado digitalmente por RENATO RIMOLO DONADIO (FIRMA)
Fecha: 2022.09.14 01:51:41 -06'00'

Dr. Renato Rimolo Donadío
Profesor Lector

EDUARDO ADOLFO CANESSA MONTERO (FIRMA)
Firmado digitalmente por EDUARDO ADOLFO CANESSA MONTERO (FIRMA)
Fecha: 2022.09.15 09:59:58 -06'00'

M.Sc. Eduardo Canessa Montero
Profesor Lector

M.Sc. Álvaro Camacho Mora
Evaluador Independiente

JUAN CARLOS ROJAS FERNANDEZ (FIRMA)
Digitally signed by JUAN CARLOS ROJAS FERNANDEZ (FIRMA)
Date: 2022.09.15 13:43:09 -06'00'

Dr. Juan Carlos Rojas Fernández
Director del Tesis

Cartago, 26 de agosto del 2022

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Luis Carlos Álvarez Mata

Cartago, 8 de septiembre de 2022

Céd: 3 0439 0959

Resumen

El enrutamiento de señales consiste en resolver el problema de encontrar conexiones entre los componentes electrónicos colocados en una placa de circuito impreso (PCB, por sus siglas en inglés). Hoy en día se tienen circuitos integrados (IC, por sus siglas en inglés) de alta gama con más de 1000 pines en paquetes de tamaños muy pequeños, donde escapar las señales se vuelva una tarea muy compleja debido al espacio existente entre pines.

Realizar el enrutamiento de escape en las etapas de diseño de un IC de alta gama puede llegar a consumir meses de trabajo por parte de los expertos en el diseño de PCBs. Se realizan diversas iteraciones con propuestas de trazados, con el objetivo de llegar a tener un mapa de pines lo suficientemente maduro para obtener trazos los cuales permitan la mejor conexión entre todas las señales.

Al mismo tiempo, los diseñadores de PCB están realizando el enrutamiento general donde buscan comprender si el orden en el que están escapando las señales del IC son razonables para realizar su interconexión con los demás componentes. Por ello, los enrutadores automáticos son una forma de mejorar el proceso del diseño de un PCB enfocándose en el enrutamiento de escape y general.

Existen investigaciones las cuales proponen algoritmos que cubren parcialmente el problema del enrutamiento automático, así como soluciones comerciales las cuales no son lo suficientemente robustas para realizar trazos aceptables.

En la siguiente tesis, se propone el diseño e implementación de un algoritmo para automatizar el enrutamiento de señales en un PCB, basado en las operaciones morfológicas de dilatación y adelgazamiento. Dando una ruta de conexión para un grupo de señales entre dos componentes electrónicos, ajustándose a las restricciones necesarias del diseño.

Esta investigación cubre los escenarios tanto del enrutamiento de escape como del enrutamiento general en una capa, para grupos de señales de terminación individual, siendo capaz de evitar obstáculos y realizando una optimización en el uso del área y el largo de los trazos.

Palabras clave: circuito impreso, PCB, enrutamiento, automático, operadores morfológicos, dilatación, adelgazamiento, Dijkstra

Abstract

Signal routing involves solving the problem of finding connections between electronic components placed on a printed circuit board (PCB). Today there are high-end integrated circuits (ICs) with more than 1000 pins in truly small packages, where escaping signals becomes a very complex task due to the space between pins.

Carrying out escape routing in the design stages of a high-end IC can consume months of work for PCB design experts. Various iterations are made with layout proposals, with the aim of having a pins map mature enough to obtain paths that allow the best connection between all signals.

At the same time, PCB designers are doing general routing where they seek to understand if the order in which IC signals are escaping is reasonable for interfacing with other components. For this reason, auto-routers are a way to improve the PCB design process by focusing on escape and general routing.

There are investigations that propose algorithms that partially cover the automatic routing problem, as well as commercial solutions which are not robust enough to perform acceptable traces.

The following thesis proposes the design and implementation of an algorithm to automate signal routing in a PCB based on the morphological operations of dilation and thinning. The mentioned gives a connection route for a group of signals between two electronic components adjusting to the necessary restrictions of the design.

This research work covers the scenarios of escape and general routing in one layer, for groups of single-ended signals, being able to avoid obstacles, and performing optimization in the use of the area and the length of the traces.

Keywords: printed circuit, PCB, routing, automatic, morphological operators, dilation, thinning, Dijkstra.

a mis queridos padres

Agradecimientos

Esta tesis de maestría se requirió de mucho esfuerzo y dedicación por parte del autor y del director de tesis, pero no hubiese sido posible su finalización sin la cooperación desinteresada de todas y cada una de las personas que a continuación citaré las cuales han sido un soporte han sido un soporte muy fuerte en momentos claves.

Agradecerle a mi familia por siempre apoyarme en cada paso que di, por siempre creer en mí, por su incondicionalidad y todo ese amor que me motivaron a nunca bajar la cabeza y siempre buscar más. Especialmente a mis queridos padres, Carlos y Adela por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar. Me han dado todo lo que soy como persona, mis valores, mis principios, mi carácter, mi empeño, mi perseverancia, mi coraje para conseguir mis objetivos.

Un agradecimiento muy especial a mi compañera, amiga y pareja, Hillary ya que siempre estuvo dispuesta para brindarme su ayuda desinteresada, su asesoramiento y consejos a lo largo de esta travesía.

A mis amigos, han sido parte muy importante de esta etapa de mi vida, por todos los momentos de apoyo en las buenas y las malas, las largas noches y estar para uno cuando más se les necesitaba. Los lazos creados después de este tiempo son invaluables; esta etapa termina pero la amistad perdura.

De igual manera mi más sincero agradecimiento a mi director de tesis, Dr. Juan Carlos Rojas por toda la colaboración, dedicación y apoyo que ha prestado para culminar este trabajo. Gracias por todas las sugerencias, ideas, la dirección que brindó y muy especialmente por esa confianza ofrecida durante este proceso.

Finalmente, un agradecimiento general a todas las personas que de una u otra forma cooperaron para que yo alcanzara esta meta.

Luis Carlos Álvarez Mata

Cartago, 8 de septiembre de 2022

Índice general

Índice de figuras

Lista de abreviaciones

Abreviaciones

DDR4	Double Data Rate 4.
DIMM	Dual In-line Memory Module.
FHD	Full High Definition.
GB	Gigabyte.
GPU	Graphics Processing Unit.
HP	Hewlett-Packard.
IC	Integrated Circuit.
JSON	JavaScript Object Notation.
MCTS	Monte Carlo tree search.
PCB	Printed Circuit Board.
RAM	Random-Access Memory.
RDIMM	Registered Memory Module.
REP	Rectangle Escape Problem.
SDRAM	Synchronous Dynamic Random-Access Memory.
SoC	System-on-Chip.
SSD	Solid-State Drive.
VLSI	Very Large-Scale Integration.

Capítulo 1

Introducción

Las placas de circuito impreso (PCB, por sus siglas en inglés) son actualmente la forma más común de fabricar circuitos electrónicos y juegan un papel muy importante en el rendimiento general de un sistema, debido a que soportan y brindan conectividad entre los componentes electrónicos . Componentes como los circuitos integrados (IC, por sus siglas en inglés), representan la mayor complejidad de interconexión en un PCB, debido principalmente a factores como las técnicas de empaquetado y al aumento en el número de pines de generación a generación.

Actualmente, se tienen ICs de alta gama con más de 1000 pines los cuales tienen un espaciamiento entre pines de menos de 0.5mm debido a que se busca tener un tamaño de paquete lo más pequeño posible. Un ejemplo de un IC con estas características se muestra en la Figura 1.1, el cual cuenta con 1377 pines en un paquete de un tamaño de 26.5 mm x 18.5 mm con un espaciamiento entre pines de 0.43 mm. Existen múltiples restricciones relacionadas con análisis térmico, diseños mecánicos, restricciones de fabricación, además de integridad de potencia y señal con recursos de enrutamiento muy limitados. Esto hace que la huella de un IC sea una entidad muy densa y el enrutamiento de escape manual se convierta en una tarea compleja en un entorno que debe cumplir con múltiples requerimientos [4, 5, 22].

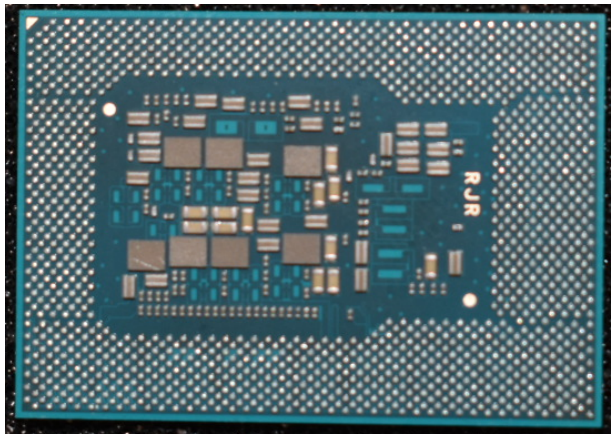


Figura 1.1: Fotografía del lado inferior del procesador Intel Core i5 - 1020Y. [34]

Como se muestra en la Figura 1.2, el enrutamiento de una señal se puede dividir en varias áreas. Para este ejemplo, se tiene la interconexión entre dos ICs, por lo tanto se debe dar un enrutamiento de escape para cada uno. El enrutamiento de escape es el que se realiza en el área que esta debajo el IC, el cual debe ir a través de los pines y buscar el borde del paquete. El enrutamiento general es el que se da entre los extremos del enrutamiento de escape.

El enrutamiento con obstáculos está presente tanto en el área de escape como en el área general. Esto se debe a que un obstáculo puede estar dado por múltiples componentes de un PCB. Se puede definir un obstáculo como cualquier componente que se encuentre en el PCB con el cual el trazo a enrutar no está conectado lógicamente. Para esta investigación se considera como obstáculos los pines y los trazos.

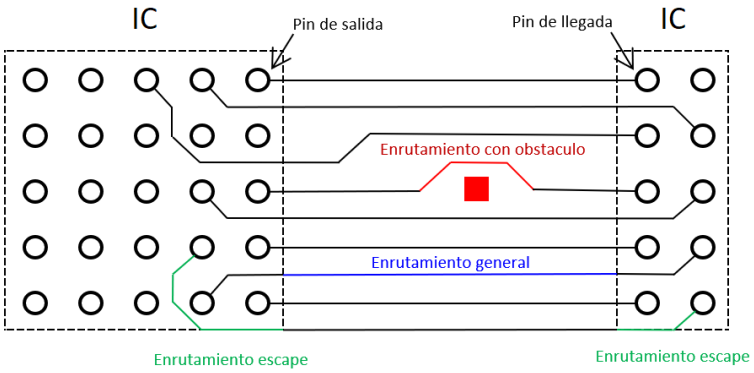


Figura 1.2: Áreas de enrutamiento.

Realizar el enrutamiento de escape en las etapas de diseño de un IC de alta gama puede llegar a consumir meses de trabajo por parte de los expertos en el diseño de PCBs, quienes ejecutan múltiples iteraciones sobre la creación del trazado de las señales con el objetivo de llegar a tener un mapa de pines lo suficientemente maduro para tener un enrutamiento de escape razonable. Pero no solo se realiza el escape, al mismo tiempo se está realizando un enrutamiento general, para comprender si el orden en que las señales están escapando del IC es eficiente para realizar su interconexión con los componentes. Estos componentes, pueden ser desde conectores de video, datos, memoria y otros ICs tan complejos como unidades de procesamiento gráficos (GPU, por sus siglas en ingles). En la Figura 1.2 se muestran las áreas de enrutamiento.

Completar todas estas conexiones manualmente es una tarea que se vuelve tediosa, la cual consume un 80% del tiempo por cada diseño. El experto en PCB debe de diseñar y verificar el correcto escape de las señales, ejecutando varios diseños en paralelo. Eliminar el proceso manual es requerido y los enrutadores automáticos son la opción para resolver problemas de enrutamiento de escape y general, obteniendo una solución global y óptima. Por lo tanto, es deseable tener algoritmos de enrutamiento automático que puedan resolver casos realistas con alta densidad de señales [5, 16, 22].

La mayoría de las soluciones propuestas en la literatura, no están atacando el problema del enrutamiento automático de forma completa. Dichas propuestas que utilizan el escape

simultaneo como recurso no abarcan de forma detallada el enrutamiento general, y dan por sentado que al tener una conciencia del orden de señales entre los dos ICs el enrutamiento general será directo y simple. Además, utilizan ejemplos sencillos en cuadrículas con un tamaño máximo de 50x50, que dan como resultados trazos directos y cortos los cuales no tienen mayor complejidad y no evalúan el potencial de sus algoritmos para casos realistas con alta densidad de pines y señales.

Esta investigación cubre los escenarios del enrutamiento de escape y general en una capa, para grupos de señales de terminación individual entre dos componentes electrónicos, evitando obstáculos y optimizando el uso del área y el largo de los trazos. Se utilizan operadores morfológicos para resolver el problema planteado, dividiendo el desarrollo en dos etapas. La primer etapa se enfoca en el trazado individual de cada señal, donde se enrutará entre un punto de salida y de llegada, realizando un enrutamiento de escape y general, como un solo trazo y tomando en cuenta la existencia de obstáculos. La segunda etapa, se enfoca en la conexión del grupo de señales, utilizando como base la etapa 1, busca el orden óptimo para obtener la mayor cantidad de señales enrutadas. Todo esto verificado por medio de 3 casos de estudio, los cuales cubren desde escenarios mínimos hasta enrutamientos de una implementación comercial.

Capítulo 2

Objetivos

2.1 Objetivo General

Como objetivo general se propuso:

- Desarrollar un algoritmo que enrute un grupo de señales de terminación individual en una capa entre dos componentes electrónicos ajustándose a los requerimientos mínimos.

2.2 Objetivos específicos

Los objetivos específicos se enfocan en atacar cada área de enrutamiento, las cuales se observan en la Figura 1.2. El algoritmo desarrollado busca cumplir cada uno de estos objetivos específicos, los cuales son:

1. Resolver el enrutamiento de escape y general entre un pin de salida y de llegada.
2. Realizar un enrutamiento que sea capaz de evitar los obstáculos.
3. Obtener un enrutamiento que cumpla con los requerimientos mínimos.
4. Optimizar el uso del área y el largo de los trazos.
5. Obtener un enrutamiento que siga las características deseadas.

2.3 Criterios de evaluación

Realizar una evaluación adecuada es clave en la investigación, ya que por medio del planteamiento de criterios de evaluación se puede determinar que el algoritmo está cumpliendo con los objetivos planteados. La evaluación tendrá dos aristas: requerimientos mínimos y características deseadas.

Estos criterios de evaluación se crearon por medio del conocimiento de experto, buscando tener una métrica repetible y así garantizar que el algoritmo cumple con los restricciones de diseño para cada uno de los casos de prueba.

2.3.1 Requerimientos mínimos

Los requerimientos mínimos son puntos que siempre se deben efectuar, estos garantizan el cumplimiento físico de la interconexión de un trazo en un PCB, lo cual se traduce en la obtención de una interconexión razonable y válida. Dichos requerimientos son:

- El trazo debe ir entre un pin A y un pin B.
- El trazo debe evitar obstáculos dejando un espaciamiento mínimo.
- El trazo no se debe traslapar con otros trazos y pines de diferente conexión.
- Se debe tener un espaciamiento mínimo entre cada trazo con diferente conexión.
- Se debe tener un espaciamiento mínimo entre trazo y pin con diferente conexión.

2.3.2 Características deseadas

Las características deseadas darán como resultado un trazo efectivo y se pueden usar para la toma de decisiones entre las opciones de trazos generados. Estas características son:

- Trazo con menor cantidad de curvas.
- Trazo con menor longitud.
- Trazo con mejor utilización del espacio.

2.4 Alcance y limitaciones

El campo de la tecnología es un entorno en constante cambio. Por lo que esta investigación ha tomado numerosos supuestos para aclarar y delimitar los alcances y resultados de la tesis. Cualquier consideración fuera de lo que se está definiendo en los objetivos, características deseadas y esta sección, no se considera dentro del alcance de esta tesis.

2.4.1 Hardware utilizado

Para el desarrollo y la verificación del algoritmo desarrollado para esta investigación se utilizó una computadora portátil con las siguientes características:

- Marca: HP
- Modelo: Elitebook 850G
- Procesador: Intel Core i7-10610U

- Memoria RAM: 16 GB
- Almacenamiento: 250 GB SSD
- Monitor: 15.6 pulgadas, FHD pantalla táctil
- Sistema operativo: Windows 10 64-bit

2.4.2 Ambiente de desarrollo

Como ambiente de desarrollo se utilizó el sistema operativo Windows 10 y el lenguaje de programación Python en su versión 3.9.6. El alcance de esta investigación se basa en la creación de un algoritmo, por lo tanto se optó por utilizar un ambiente de desarrollo el cual cuenta con las bibliotecas necesarias para el adecuado manejo y procesamiento de los datos del PCB. Las bibliotecas que se utilizaron como soporte son:

- Opencv 4.5.3: se utiliza para almacenar las imágenes del enrutamiento.
- Numpy 1.21.1: es el encargado de la creación y manipulación de la matriz que contiene el enrutamiento y los pines.
- Json 2.0.9: da acceso a la información del mapeo del PCB que esta almacenada en archivos con este formato.

Como herramienta de comparación se utilizó Allegro PCB Designer en si versión 17.4. La selección de esta herramienta se debe a que se cuenta con la licencia y acceso a bibliotecas de símbolos, lo cual facilita el desarrollo de la comparaciones realizadas.

2.4.3 Abstracción del enrutamiento

El algoritmo desarrollado hace una abstracción del enrutamiento, para esto utiliza una cuadrícula. Por lo tanto, se realiza una conversión de dimensiones físicas de los componentes de un PCB en micrómetros a un espacio de la cuadrícula, donde una posición es equivalente a $33.78um$. Para la simplificación del algoritmo, se define que el ancho de trazo y el espaciamiento mínimo es de una posición en la cuadrícula.

Capítulo 3

Estudio bibliográfico

Se han desarrollado diversas soluciones para atacar la problemática del enrutamiento automático, propuestas que van desde los estudios académicos en áreas específicas del enrutamiento hasta soluciones que se pueden encontrar en herramientas comerciales del diseño de PCBs. Por lo tanto, se va a dividir el estudio bibliográfico en dos secciones: estudios académicos y soluciones comerciales.

3.1 Soluciones comerciales

La industria ha identificado la necesidad de tener herramientas que realicen enrutamientos de forma automática. Por lo tanto, hay compañías las cuales ofrecen dentro de sus paquetes de software un soporte para este tipo de procesamiento. Entre estas herramientas están:

- Allegro PCB Editor por Cadence.
- Altium Designer por Altium Limited.
- Proteus PCB Design por Labcenter Electronics
- Eagle PCB Design por Autodesk.

3.1.1 Allegro PCB Editor

Allegro es una de las herramientas más utilizadas en la industria para realizar diseños de PCB. Esta herramienta incluye dentro sus opciones más avanzadas el *autoconect*, nombre por el cual Cadence llama a su herramienta para realizar un enrutamiento automático. El *autoconect* puede realizar enrutamientos automáticos de una forma rápida y precisa pero esto dependerá de la complejidad del diseño. Por lo tanto se toma esta herramienta como referencia para ser comparada con el algoritmo propuesto [1].

Para casos donde se tenga un enrutamiento de escape con una densidad alta de señales y pines, *autoconect* puede llegar a dar soluciones no viables. Para mostrar esto se utilizó uno de los casos de prueba el cual arrojó como resultado un enrutamiento erróneo, donde se dan casos como la superposición de dos señales. Un ejemplo de esto se muestra en la Figura 3.1.

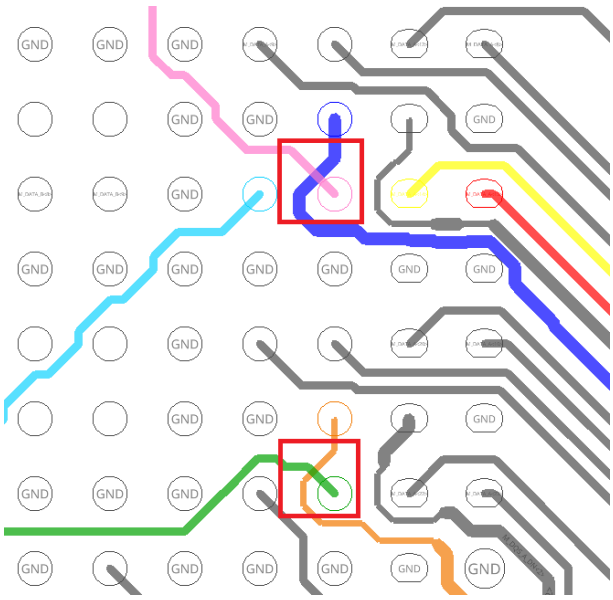


Figura 3.1: Ejemplo de un mal enrutamiento utilizando el software Allegro de Cadence, donde se enruta señales sobre otras señales.

3.1.2 Altium Designer

Altium Designer dispone de un enrutador automático, el cual utiliza una arquitectura topológica para mapear el espacio de enrutamiento sin restricción geométrica. En lugar de utilizar la información de coordenadas del espacio de trabajo al dividirlo en una cuadrícula, el algoritmo construye un mapa utilizando solo las posiciones relativas de los obstáculos en el espacio, sin referencia a sus coordenadas. Lo hace triangulando el espacio entre obstáculos adyacentes creando un mapa. Este mapa, es luego utilizado para crear una ruta desde el punto de inicio hasta el final tomando en cuenta los obstáculos. El mapa es independiente de la forma de los obstáculos y el espacio se puede atravesar en cualquier ángulo; donde el enrutamiento obtenido no se limita a caminos puramente verticales u horizontales [2].

3.1.3 Proteus PCB Software

Proteus es otra herramienta que posee un enrutador automático, el cual utiliza algoritmos de reducción de conflictos basados en costos probados para maximizar las tasas de finalización [1].

3.1.4 Eagle PCB Design

Autodesk recomienda su enrutador automático para realizar optimización de la colocación, descubrir cuellos de botella o buscar ideas diferentes para realizar la solución. Y deja muy en claro que fuera de estos tres usos no se debe confiar en el enrutador automático como un reemplazo al diseño manual que puede realizar un experto [3].

3.1.5 Comparación entre soluciones comerciales

Para realizar la comparación entre soluciones comerciales solo se dispone de la información que sus autores muestran en sus sitios webs. A continuación se resumen los tipos de algoritmos utilizados por las soluciones propuesta por Altium y Proteus.

Altium Designer describe que utiliza un algoritmo el cual construye un mapa utilizando solo las posiciones relativas de los obstáculos en el espacio, sin referencia a sus coordenadas. Por otro lado, Proteus PCB Software tiene algoritmos de reducción de conflictos basados en costos probados para maximizar las tasas de finalización. Eagle PCB Design no indica que tipo de algoritmos utiliza y solo indica que no es una solución tan robusta para confiar en el enrutamiento automático como un reemplazo al enrutamiento manual.

Para esta comparación se hará énfasis en el software de Cadence, Allegro PCB Editor, debido a que es la herramienta más utilizadas en la industria para desarrollar diseños de PCBs. Allegro es capaz de cubrir el enrutamiento de escape y general, donde para realizar una verificación de su robustez se toma un caso de prueba, el cual también se probará con el algoritmo propuesto.

La Figura 3.2 muestra una solución de un IC de alta gama donde se enrutan sus señales a un conector de memoria. Se puede apreciar en los trazos resaltados con colores que estos no siguen un flujo correcto y se enrutan alrededor creando una solución no viable. Además, se observan deficiencias como el enrutamiento de una señal sobre otra, como se muestra en la Figura 3.1, donde se muestra en dos casos el enrutamiento de una señal sobre otra.

3.2 Estudios académicos

La academia ha desarrollado diversos esfuerzos en investigaciones los cuales cubren soluciones completas así como casos específicos. Estas investigaciones se pueden categorizar en 7 áreas diferentes:

- Enrutamiento de escape.

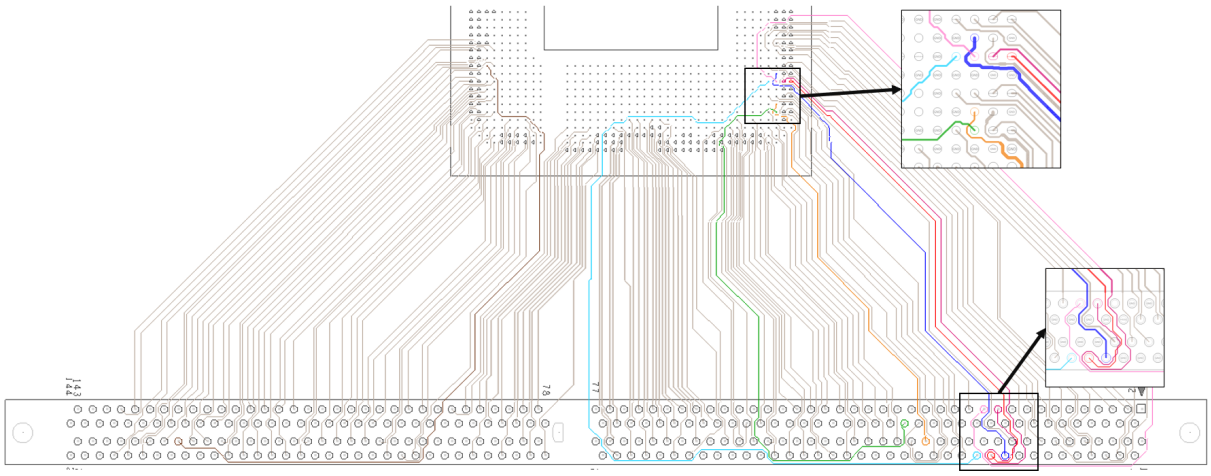


Figura 3.2: Ejemplo de enrutamiento entre un IC de alta gama y un conector de memoria utilizando el software Allegro de Cadence.

- Enrutamiento general.
- Ángulo de enrutamiento.
- Planificación de buses.
- Asignación de capas.
- Distancia entre componentes.
- Enrutamiento con vías.

3.2.1 Enrutamiento de escape

Enrutamiento de escape es el problema más estudiado debido a que es el de más complejidad, donde por ejemplo [4, 5, 18, 22, 23, 25, 26, 30, 36, 37, 38] cubren este tema y se centran en los ICs que utilizan de alta densidad de pines colocados de forma matricial. Al ser una problemática tan abordada se clasifican en 4 categorías:

- Escape desordenado.
- Escape ordenado.
- Escape simultáneo.
- Escape rectangular.

Escape desordenado

Este enrutamiento va desde los pines en un IC, hasta el borde del mismo IC sin ninguna restricción en el orden de enrutamiento de los pines, este tema esta abordado por [36, 37].

Escape ordenado

El escape ordenado también considera solo una matriz de pines. Sin embargo, requiere que la ruta de escape se ajuste a un orden específico a lo largo del borde del IC. El escape ordenado es ampliamente desarrollado por [25, 37].

Escape simultáneo

Considere el enrutamiento de escape de dos ICs al mismo tiempo. Se requiere que el orden de las rutas de escape en las dos matrices coincida para proporcionar una topología plana para el enrutamiento general posterior. La figura 3.3 puede ayudar a comprender este enfoque, muestra que dos conjuntos de pines que deben estar conectados entre sí, donde un pin de cada IC se escapa al mismo tiempo. Los trabajos anteriores sobre este tema son [4, 5, 22, 30, 37].

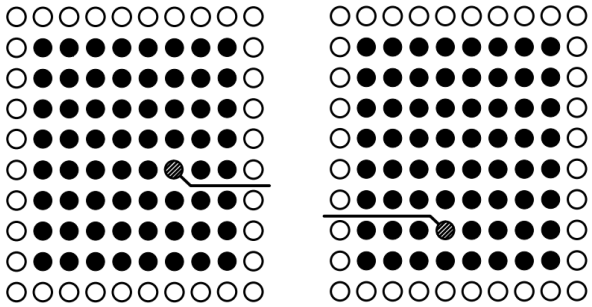


Figura 3.3: Escape simultáneo, muestra una representación de dos ICs con una matriz de pines. [5]

Escape rectangular

En el problema de escape de rectángulos (REP), dada una región rectangular R y un conjunto S de rectángulos que permanecen dentro de R, donde cada rectángulo tiene que hacer coincidir las señales con uno de los cuatro límites de R. El objetivo de REP es determinar una dirección de escape para cada rectángulo en S, de modo que se minimice la densidad máxima resultante sobre la región R. La figura 3.4 muestra cuatro rectángulos de proyección para el grupo de señales, con un ejemplo de ruta de escape hacia el límite derecho. El trabajo anterior sobre este tema incluye [18, 23]

3.2.2 Enrutamiento general

Otros autores se centraron en el problema del enrutamiento general, el que ocurre después del enrutamiento de escape y que es necesario para interconectar los elementos existentes en la PCB. Como resumen, estos algoritmos intentan enrutar una señal de un lado a otro,

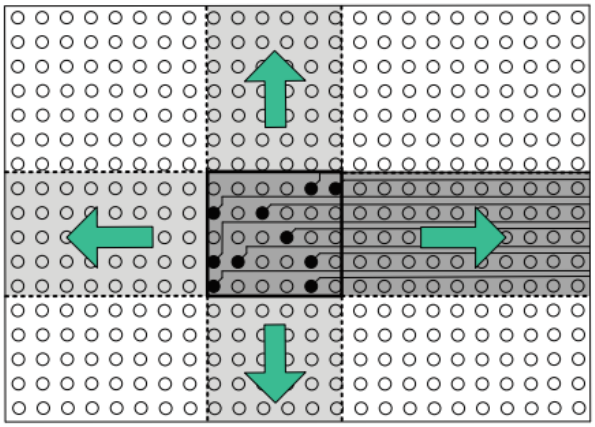


Figura 3.4: Escape rectangular. [23]

pero atacan los casos de enrutamiento específicos.

Por ejemplo, [17, 28, 39] resuelve el problema de la coincidencia de longitudes. [28] propone un algoritmo para enrutar redes con restricciones de área mínima y longitud máxima. Este algoritmo se extiende luego al caso en el que las restricciones mínimas se dan como límites de longitud exactos. Otra solución que también propone la coincidencia de longitudes es [17], pero añade un extra al solventar el enrutamiento cuando hay obstáculos por medio de un algoritmo heurístico rápido, Figura 3.5.

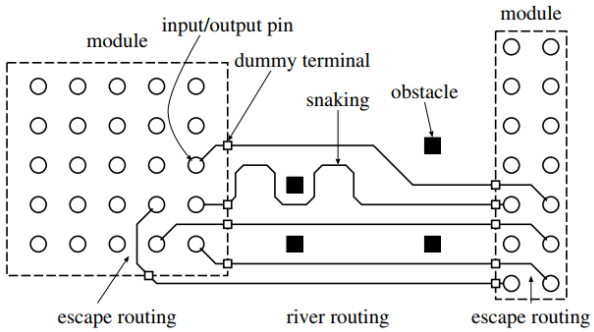


Figura 3.5: Un ejemplo de enrutamiento de PCB con obstáculos y coincidencia de longitudes.[17]

Un enfoque muy particular se puede ver en la propuesta de [11], donde se realizan verificaciones paulatinas del cumplimiento de las reglas de diseño. Para eso, utiliza un algoritmo de búsqueda de ruta de enrutamiento topológico.

También hay propuestas donde el objetivo es mejorar la velocidad de ejecución del algoritmo. Como lo describe [14], un algoritmo de enrutamiento conocido como laberinto incremental ultrarrápido el cual realiza una búsqueda incremental a través de la ruptura de equivalencia explícita para evitar la acumulación ciega de esfuerzos de búsqueda du-

plicados.

Una propuesta enfocada en el enrutamiento de paquetes es descrita por [21], caso de estudio que también se puede exportar al diseño de PCB. Plantea un primer algoritmo que calcula la ruta más corta y un segundo algoritmo que realiza una búsqueda aleatoria paralela para resolver rutas en conflicto.

3.2.3 Ángulo de enrutamiento

En [16], se propone un método cuasi-newtoniano basado en programación no lineal para resolver un problema de enrutamiento sin cuadrícula de ángulos que busca aplicarse en el enrutamiento principal que se puede aplicar tanto para escape como para enrutamiento principal. Este método minimiza la longitud total de las señales, la separación para dos rutas y el ángulo de curvatura en una ruta. Además, está el caso de [36] que tiene la capacidad de enrutamiento diagonal (45 grados) donde el modelo de flujo de red utiliza crea un mosaico que contiene cinco nodos, N-nodo en el norte, E-nodo en el este, S- nodo en el sur, W-nodo en el oeste y nodo C en el centro con lo que se plantea la estrategia de señales de escape, ver Figura 3.6.

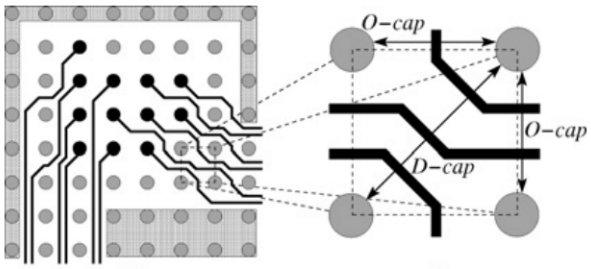


Figura 3.6: Ejemplo del problema de rutas de escape con capacidad diagonal. [36]

3.2.4 Planificación de buses

La planificación de buses es uno de los pasos más largos en el enrutamiento de PCB. Consiste en asignar buses a múltiples capas de la PCB y enrutarlos planos en cada capa. La congestión del enrutamiento entre los componentes a bordo y los límites de longitud mínimo-máximo de los buses también deben considerarse durante el enrutamiento [19, 35]. Incluso este enfoque es aplicado en el diseño de VLSI donde [7] presenta un método de enrutamiento de laberinto bajo un esquema concurrente y jerárquico para buses.

3.2.5 Asignación de capa

Tener diseños de PCB multicapa es muy común en los sistemas más robustos. En [8] se formula el problema de asignación de capas multicapa al introducir la noción de gráfico

de continuación de conflicto extendido que abstrae las relaciones de conectividad de un diseño dado para la asignación de capa problema . Además, [24] propone un algoritmo basado en bifurcaciones y límites que resuelve de manera óptima el problema de asignación de capas del enrutamiento de escape del bus. Esta problemática no es abordada en la solución, pero es un área qué podría ser estudiada en el futuro.

3.2.6 Distancia entre componentes

La ubicación de los componentes es un paso crítico para asegurar un enrutamiento exitoso a nivel de PCB. La distancia entre componentes nos permitirá variar los retornos en el momento del enrutamiento. El propósito de [33] es estimar correctamente la distancia mínima requerida para que el software de enrutamiento automático enrute correctamente entre dos componentes. Esto se logra mediante el uso de un ataque de fuerza bruta para disminuir progresivamente la distancia entre los componentes utilizando un enfoque bidireccional para encontrar la distancia mínima a la que el software de enrutamiento automático aún puede enrutar con éxito un diseño específico.

3.2.7 Enrutamiento con vías

Un diseño abarrotado con muchas vías puede causar problemas en el enrutamiento de otras señales o en los planos de potencia. En [8] se desarrolla una solución de enrutamiento de PCB de dos capas con la que se obtendrá el número mínimo de rutas. Por otro lado, [20] propone un método para optimizar la asignación de pines y el enrutamiento de escape en presencia de pares diferenciales y uso de vía ciegas para la matriz de pines de cuadrícula.

3.2.8 Comparación entre soluciones académicas

En la solución, se propone un algoritmo el cual cubre el enrutamiento de escape y enrutamiento general. Soluciones como las que presentan [4, 5, 22, 30, 38] cubren estas mismas áreas, pero resolviendo la problemática con diferentes técnicas las cuales tienen en común que utilizan un enfoque de enrutamiento simultáneo.

Un algoritmo de búsqueda heurístico es propuesto por [38], el cual convierte el PCB en un mapa de cuadrícula, ubica cada pin y busca la ruta más corta entre el pin de inicio y final. La estructura propuesta consiste en dos árboles de búsqueda Monte Carlo (MCTS, por sus siglas en inglés), donde el primer MCTS generará una ruta incompleta, un subobjetivo, para el segundo MCTS. El segundo MCTS obtiene dos rutas óptimas por el subobjetivo y las dos rutas óptimas se concatenarán como la ruta óptima final. Para esta investigación presentan ejemplos simples, donde el enrutamiento más complejo enruta 6 señales que se ubican en el borde del IC, como se muestra en la Figura 3.7.

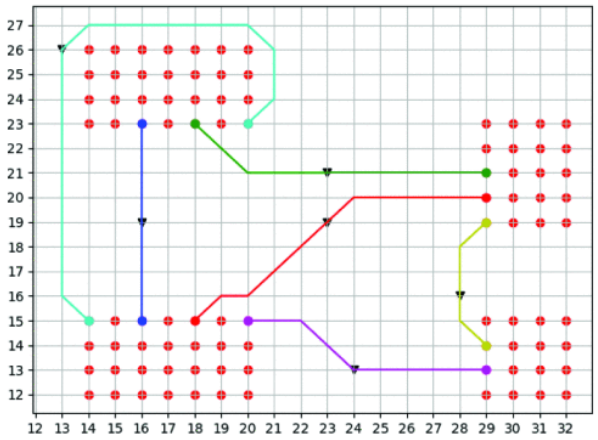


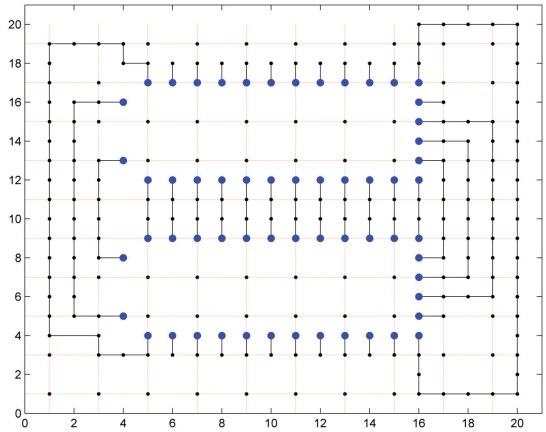
Figura 3.7: Enrutamiento resultante del algoritmo MCTS. [38]

Al igual que [38], se ha convertido el PCB en una cuadrícula, donde se ubican los pines de cada componente con su pin de salida y llegada para cada señal, pero al mismo tiempo en la solución propuesta se busca tener un algoritmo robusto que realice un enrutamiento de escape el cual cubra casos de escape complejos en ICs con una densidad alta de pines.

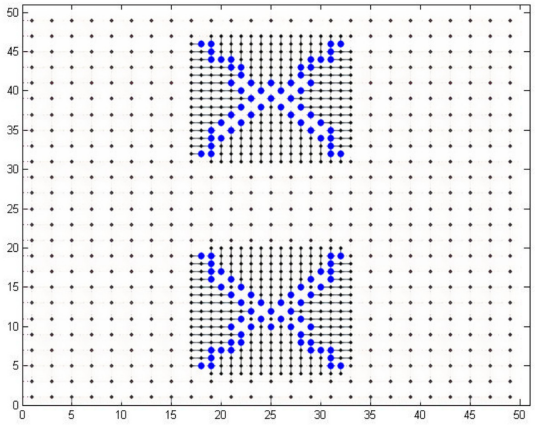
Por otro lado [4, 5, 30] proponen enfoque muy similar de enrutamiento simultáneo, donde por medio de un de flujo de red lineal tratan de resolver esta problemática. En [4] se propone dividir el problema en dos etapas donde en la primera etapa, las señales se escapan de los pines al límite del paquete de los ICs de inicio y final de forma simultánea y en la segunda etapa, estas señales se conectan entre sí. En [5] se muestra una solución con dos algoritmos uno lineal y no lineal. Tanto [4] y [5], presentan resultados de enrutamiento muy simples en cuadrículas pequeñas de 50x50 como tamaño máximo donde muestra la salida de señales con un enrutamiento muy lineal y poco complejo a pesar de tener un caso con 112 pines escapados, como se muestra en la Figura 3.8. Las huellas de los ICs de alta gama poseen patrones de pines complejos y densos, donde no existe la posibilidad de realizar enrutamientos de escape lineales, el algoritmo propuesto busca cubrir estos casos complejos donde el escape de las señales necesita rodear diversos obstáculos.

Continuando con soluciones que presentan un enfoque de enrutamiento simultáneo, [22] define una etapa de enrutamiento al límite de los ICs, utilizan una estructura de grupo, donde agrupan los pines que están ubicados cerca uno del otro y realizan un enrutamiento global para el grupo, para posteriormente realizar un enrutamiento en detalle del grupo, Figura 3.9.

Por último, [30] también presenta un enfoque de enrutamiento simultáneo con flujo de red. Además consideró el número de trazos que pueden pasar entre los pines adyacentes de un IC y aplica un algoritmo de resolución de grafos planos bipartitos para enrutar las señales hasta que estas coincidan. Incluso compara su algoritmo con una herramienta



(a) Pines escapados conectados en una cuadrícula de 20×20 [5].



(b) 112 pines escapados en una cuadrícula de 50x50 [4].

Figura 3.8: Casos de enrutamiento simples en cuadrículas pequeñas

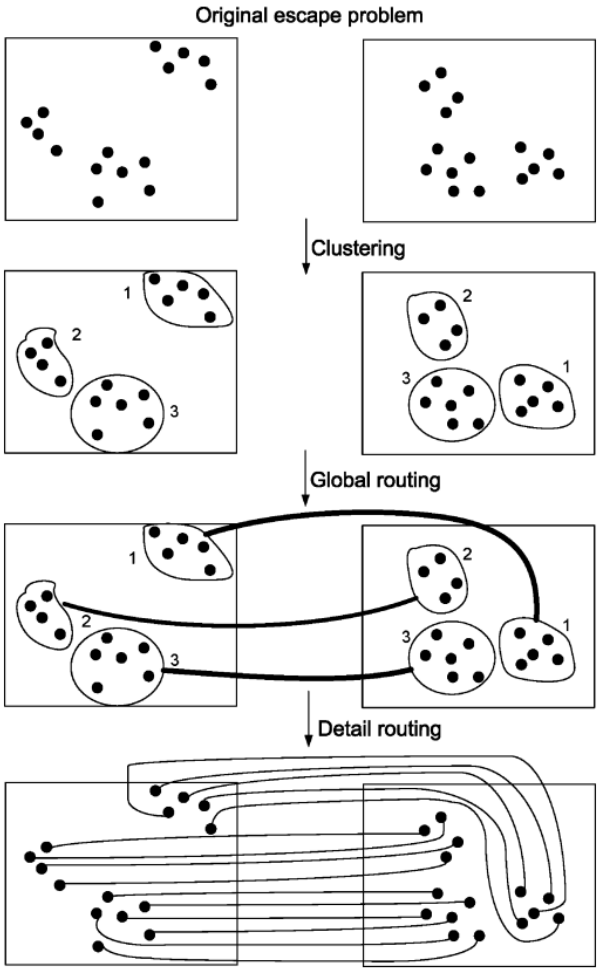


Figura 3.9: Enrutamiento mediante grupos. [4]

comercial enrutando 100 posibles señales entre dos ICs, Figura 3.10. Los casos de prueba propuestos son muy interesantes donde optan por casos más complejos en comparación con las anteriores investigaciones llegando a escapar 100 señales entre ICs de 98x94 y 86x50 pines. Incluso van más allá y realizan comparaciones con una herramienta comercial.

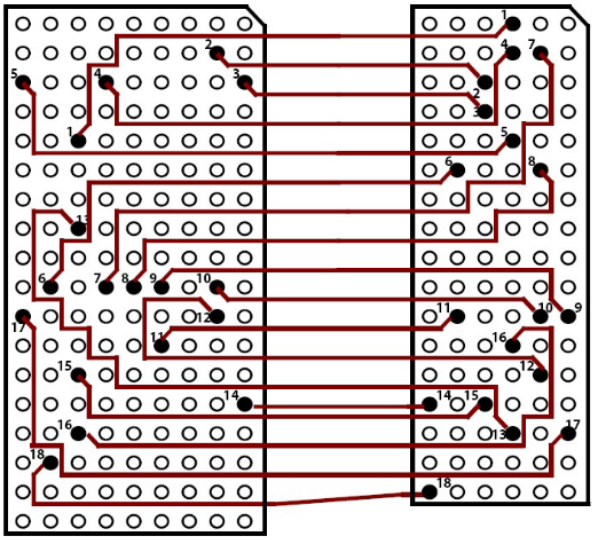


Figura 3.10: Modelo de enrutamiento de escape simultáneo enrutamiento para diferentes tamaños de ICs. [30]

El enrutamiento simultáneo tiene la problemática de que se debe negociar el orden del escape de las señales para ambos ICs, si alguno de los ICs resulta con un orden diferente se debe ir a negociar el orden de nuevo. Esto genera tener un procesamiento para cada IC, es decir un doble procesamiento por cada señal y un incremento en la complejidad del algoritmo. Por lo tanto, en este trabajo de investigación se propuso un algoritmo que genere los caminos posibles para una señal y seleccionar el camino más razonable, realizando sólo una vez el procesamiento para una señal y teniendo una complejidad menor en el desarrollo del algoritmo.

La mayoría de las soluciones no están atacando el problema del enrutamiento automático de forma completa. Las soluciones con una propuesta que utilizan el escape simultáneo como [4, 5, 22, 38], no abarcan de forma detallada el enrutamiento general, y dan por sentado que al tener una conciencia del orden de señales entre los dos ICs el enrutamiento general será directo y simple. Además, utilizan ejemplos sencillos en cuadrículas con un tamaño máximo de 50x50 que dan como resultados trazos directos y cortos los cuales no tienen mayor complejidad y no evalúan el potencial de sus algoritmos para casos realistas con alta densidad de pines y señales.

3.3 Marco Teórico

La morfología matemática se basa en operaciones de teoría de conjuntos, la cual se ha convertido en una herramienta muy poderosa y utilizada para diferentes tareas en el procesamiento digital de imágenes. Dos operaciones básicas de la morfología matemática son las de erosión y dilatación. Para la solución es relevante tener en cuenta los conceptos de sobre el uso de operaciones morfológicas como la dilatación y el adelgazamiento, además del algoritmo de Dijkstra, ya que son utilizados en la propuesta [27].

3.3.1 Dilatación

La dilatación es una operación matemática morfológica donde en una imagen binaria amplía gradualmente los límites de las regiones de los píxeles de primer plano. Por lo tanto, las áreas de píxeles de primer plano aumentan de tamaño mientras que los agujeros dentro de esas regiones se vuelven más pequeños [9, 12, 40].

El operador de dilatación toma dos datos como entradas. La primera es la imagen que se va a dilatar. El segundo es un conjunto (generalmente pequeño) de puntos de coordenadas conocido como elemento estructurante (también conocido como kernel). Es este kernel el que determina el efecto preciso de la dilatación en la imagen [9].

La definición matemática de dilatación en escala de grises es idéntica excepto por la forma en que se deriva el conjunto de coordenadas asociado con la imagen. Como ejemplo de dilatación binaria, se puede tomar un kernel es un cuadrado de 3×3, con el origen en su centro, como se muestra en la Figura 3.11. Se define que los píxeles del primer plano están representados en blanco y los píxeles del fondo en negro [9].

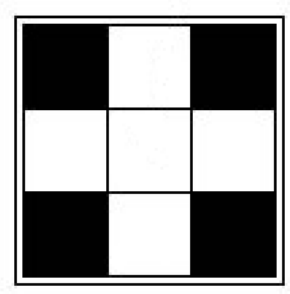
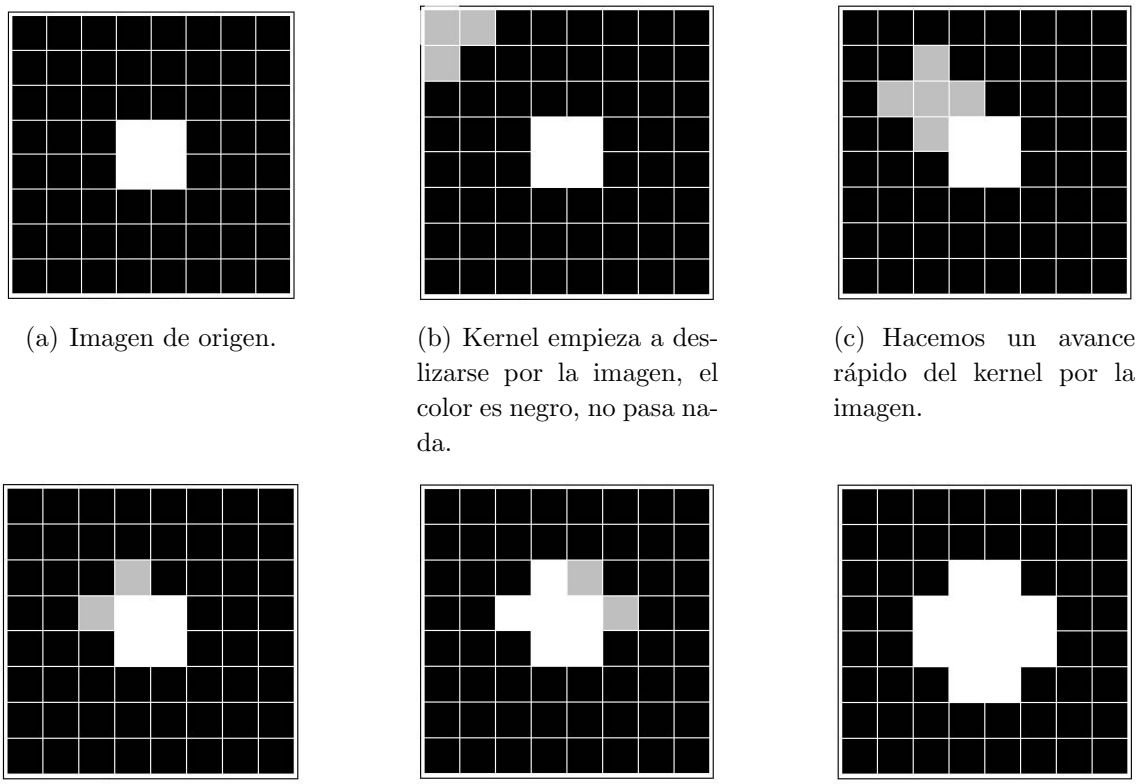


Figura 3.11: Kernel cuadrado de 3×3. [31]

Para calcular la dilatación de una imagen binaria por este kernel, se considera cada uno de los píxeles de la imagen por turno. Para cada píxel de la imagen se superpone el kernel, iniciando por la parte superior de la imagen para que el origen del kernel coincida con la

posición del píxel de entrada. Si el origen del kernel coincide con un píxel de la imagen, los píxeles cubiertos por el kernel toman el valor correspondiente. Sin embargo, si el píxel no coincide con el origen del kernel, se deja en el valor de la imagen. En la Figura 3.12 se observa un ejemplo paso a paso del proceso de dilatación [9].



(a) Imagen de origen.

(b) Kernel empieza a deslizarse por la imagen, el color es negro, no pasa nada.

(c) Hacemos un avance rápido del kernel por la imagen.

(d) El kernel coincide con un píxel blanco, se convierten los píxeles a blanco.

(e) De nuevo, el kernel coincide con un píxel blanco, se convierten los píxeles a blanco.

(f) Imagen de origen dilatada.

Figura 3.12: El kernel se desliza sobre la imagen de origen [31].

Como limitante la dilatación va a llegar a unir cualquier área diferente de cero y en la solución es necesario tomar en cuenta la existencia de posiciones las cuales no son cero o uno, como obstáculos, pines y los trazos de otras señales. Por este motivo se vio la necesidad de crear un algoritmo basado en el algoritmo de dilatación pero que mantuviera los obstáculos existentes.

3.3.2 Adelgazamiento

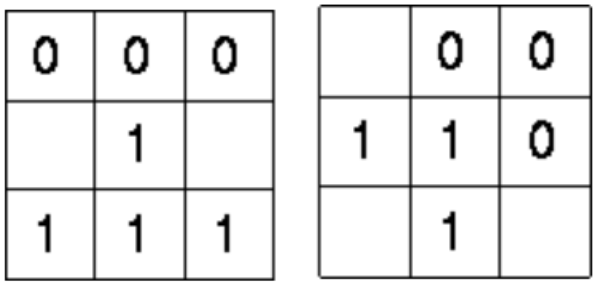
El adelgazamiento es una operación matemática morfológica que se utiliza para eliminar píxeles de primer plano seleccionados de imágenes binarias. Puede usarse para varias aplicaciones, pero es particularmente útil para esqueletizar. El adelgazamiento normalmente

solo se aplica a imágenes binarias y produce otra imagen binaria como salida [10, 40].

El comportamiento de la operación de adelgazamiento está determinado por un kernel. Los kernels de estructuración binaria utilizados para el adelgazamiento son del tipo extendido descrito en la transformación de localización (hit-and-miss transform, en inglés) [10].

La operación de adelgazamiento se calcula traduciendo el origen del kernel a cada posible posición de píxel en la imagen, y en cada una de dichas posiciones comparándola con los píxeles de la imagen subyacente. Si los píxeles de la imagen coinciden con el kernel, entones el píxel se establece como un cero. De lo contrario no se modifica la imagen. Se debe tener en cuenta que el kernel siempre debe tener uno o un espacio en blanco en su origen para que tenga algún efecto [10].

Este efecto se puede lograr mediante el adelgazamiento morfológico iterando hasta la convergencia con los kernel mostrados en la Figura 3.13, y todas sus rotaciones de 90 grados (8 kernels en total) [10].



(a) Kernel 1. (b) Kernel 2.

Figura 3.13: Los kernels para adelgazamiento morfológico. En cada iteración, la imagen se adelgaza primero con el kernel de la izquierda, luego con el de la derecha y luego con las seis rotaciones de 90° restantes de los dos kernels [10].

La Figura 3.14 se muestra un ejemplo de adelgazamiento de una imagen binaria simple [10].

Como limitante, el adelgazamiento no considera puntos de salida o llegada y es posible que estos puntos fueran removidos del adelgazamiento final. Los puntos de salida y llegada son vitales en el desarrollo del algoritmo propuesto debido a que sin estos se generarían caminos sin salida creando soluciones no válidas. Por este motivo, se vio la necesidad de crear un algoritmo basado en el algoritmo de adelgazamiento pero que mantuviera los puntos de salida y llegada. La Figura 3.15, muestra un ejemplo de adelgazamiento, si

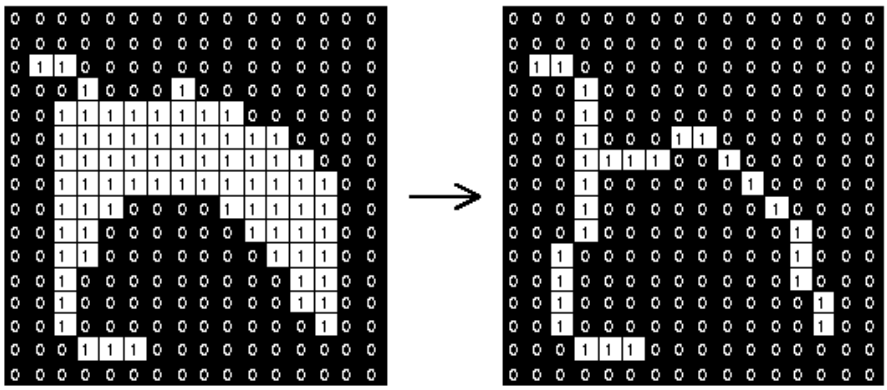


Figura 3.14: Ejemplo de adelgazamiento de una imagen binaria. [10]

tuviéramos los pines de llegada y salida en los vértices de las líneas verdes y amarillas en la imagen adelgazada perderíamos la conexión con estas posiciones.

(a) Imagen de origen [29]. (b) Imagen adelgazada [29].

Figura 3.15: Casos de enrutamiento simples en cuadrículas pequeñas

3.3.3 Algoritmo de Dijkstra

El algoritmo de Dijkstra resuelve el problema de encontrar el camino más corto desde un punto de inicio en un grafo hasta un destino. Por medio de este algoritmo se puede encontrar las rutas más cortas desde una fuente determinada a todos los puntos en un grafo al mismo tiempo.

El algoritmo de Dijkstra inicia en el nodo que elija (el nodo de origen) y analiza el grafo para encontrar la ruta más corta entre ese nodo y todos los demás nodos. El algoritmo realiza un seguimiento de la distancia más corta conocida actualmente desde cada nodo, al nodo de origen y actualiza estos valores si encuentra una ruta más corta. Una vez que el algoritmo ha encontrado la ruta más corta entre el nodo de origen y otro nodo, ese

nodo se marca como *visitado* y se agrega a la ruta. El proceso continúa hasta que todos los nodos se han agregado a la ruta. De esta manera, tenemos una ruta que conecta el nodo de origen con todos los demás nodos siguiendo la ruta más corta posible para llegar a cada nodo.

El algoritmo de Dijkstra sólo puede funcionar con grafos que tengan pesos positivos. Esto se debe a que, durante el proceso, se deben sumar los pesos de los bordes para encontrar el camino más corto. [6, 15, 32].

La Tabla 3.1 y la Figura 3.16 muestran un ejemplo donde se encuentra la distancia más corta del vértice A hacia todos los demás vértices.

Vértice	Distancia	Camino
F	4	A-D-E-F
D	1	A-D
C	3	A-D-E-C
E	2	A-D-E
B	2	A-B

Tabla 3.1: Tabla resultante con la distancia y caminos más cortos del vértice A hacia todos los demás vértices [6].

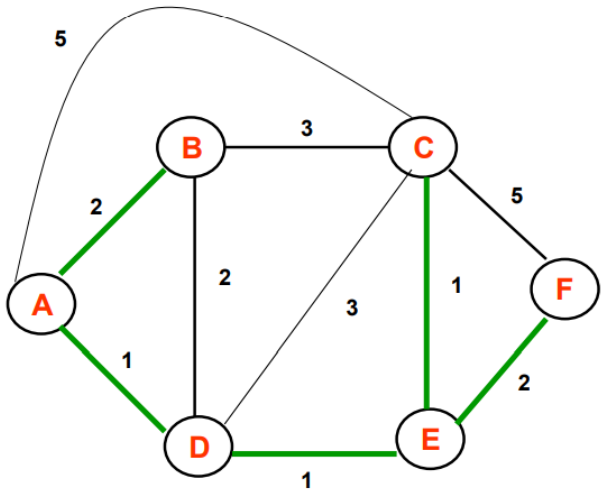


Figura 3.16: Los caminos más cortos desde el vértice A hacia todos los demás vértices [6].

Capítulo 4

Algoritmo de enrutamiento

4.1 Casos de estudio

Para la verificación del algoritmo y poder planear las siguientes etapas de la investigación, se definieron 3 grupos de caso de estudio, los cuales se van a desarrollar a continuación.

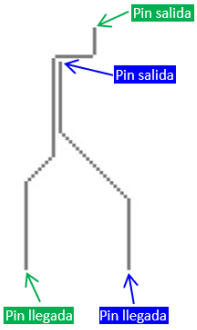
4.1.1 Casos de estudio pequeños

Se propusieron 6 casos de estudio los cuales cubren escenarios mínimos que el algoritmo debe resolver. Estos casos fueron implementados en una cuadrícula de 10x10 donde se enrutaron desde 2 hasta 5 señales, con el objetivo poder crear una prueba de concepto del algoritmo. Por medio de estos casos, se verificaron 3 diferentes propuestas de algoritmos y se puedo seleccionar el mejor algoritmo a desarrollar.

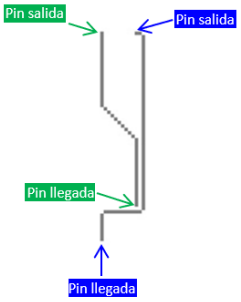
En la Figura 4.1 se pueden observar los resultados de los 6 casos de estudio. En el primer caso se busca verificar qué pasa si se tiene dos señales las cuales van en sentido contrario, es decir donde se deberían cruzar. Para el segundo caso, se extendió el primer caso buscando que se necesitará realizar una dilatación extendida.

El caso tres se realiza el enrutamiento de 3 señales las cuales se cruzan entre sí. Para el caso 4, se combinaron 4 señales las cuales se deben cruzar, pero también es necesario realizar una dilatación extendida.

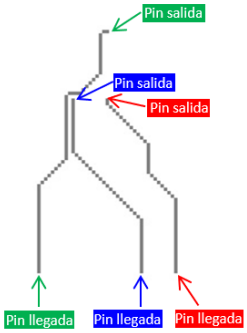
El caso 5 se crea un escenario aún más complejo, donde se tienen 5 señales las cuales se cruzan y se tiene la necesidad de realizar 4 dilataciones extendidas. Como último caso se verifica qué pasa si hay obstáculos.



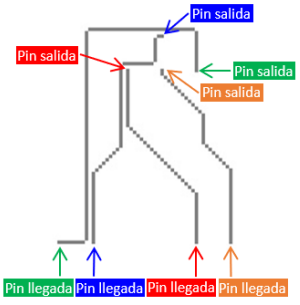
(a) Caso 1. Dos señales con pines cruzados.



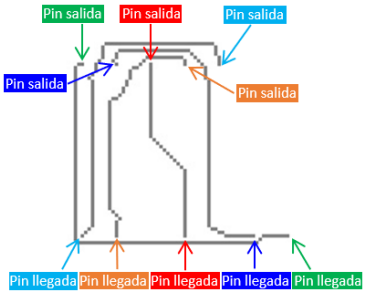
(b) Caso 2. Dos señales con pines cruzados y dilatación extendida.



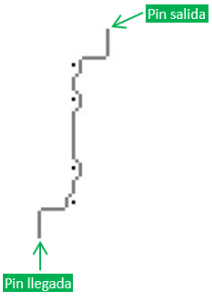
(c) Caso 3. Tres señales con pines cruzados.



(d) Caso 4. Tres señales con pines cruzados y dilatación extendida.



(e) Caso 5. Cinco señales con pines cruzados y dilatación extendida.



(f) Caso 6. Una señal con obstáculos.

Figura 4.1: Casos de estudio pequeños

4.1.2 Caso de estudio mediano

Se propuso un caso de estudio donde se enrutaron 20 señales usando la huella de un IC de alto rendimiento para los pines de salida. En este caso, se utiliza un sistema en chip (SoC, por sus siglas en inglés) para una computadora portátil. Se busca verificar el enrutamiento de escape, el cual es una de las secciones más difíciles de realizar por lo que para los pines de llegada se utilizó una topología simplificada, como se muestra en la Figura 4.2.

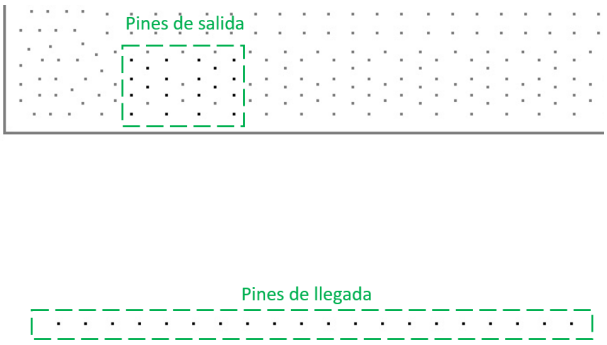


Figura 4.2: Caso de estudio mediano, enrutamiento de 20 pines (color negro) en una huella de un IC de alto rendimiento.

4.1.3 Caso de estudio grande

Se propuso un caso de estudio donde se enrutan 124 señales usando la huella de un IC de alto rendimiento para los pines de salida, en este caso es una unidad central de procesamiento (CPU por sus siglas en inglés) para una computadora de escritorio. Para los pines de llegada se utilizó una huella de un conector de memoria DDR4 SDRAM RDIMM. Para este caso, se busca tener un escenario aún más complejo, como se observa en la Figura 4.3.

4.2 Herramientas de Soporte

Como parte del desarrollo y verificación del algoritmo, fue necesario crear herramientas de soporte las cuales brindaran facilidad y trazabilidad. Para esto se implementó un mapeo de los componentes existentes en el PCB y dos formas de visualizar el enrutamiento resultante.

4.2.1 Mapeo del PCB

El mapeo de los componentes existentes en el PCB fue una de las primeras tareas necesarias para poder exportar los datos del PCB a estructuras, las cuales fueran viables a

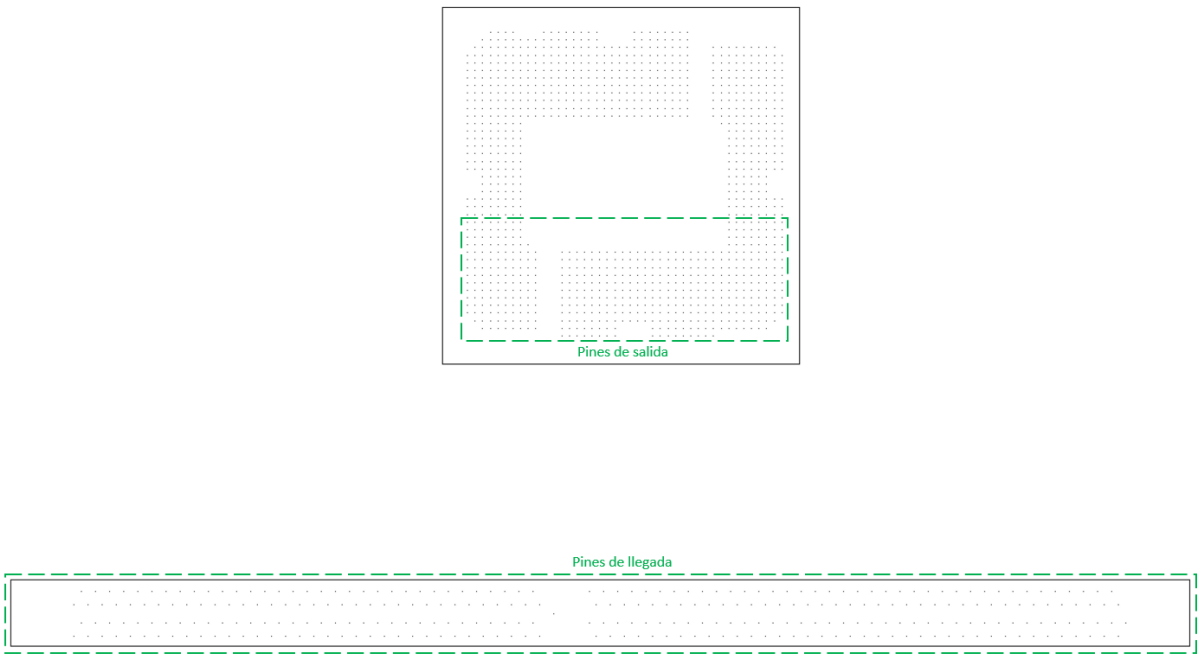


Figura 4.3: Caso de estudio grande, enrutamiento de 124 pines (color negro) saliendo de una huella de un IC de alto rendimiento y llegando a una huella de un conector de memoria

nivel de software para su manipulación. Se va a describir este mapeo iniciando desde una vista general hasta ir al detalle.

El sistema está conformado por dos partes, la física y la lógica. En la parte física, el sistema va a contener una cantidad de componentes. El PCB, que es donde se colocan los componentes y se realiza el enrutamiento. Para la parte lógica se trata de las conexiones que existen entre los componentes.

Cada componente que se coloca en un PCB tiene dos características principales, el tamaño de su paquete y los pines. Para ellos se creó una clase, la cual contiene las dimensiones de los componentes, además de la lista de pines.

Las conexiones entre pines de un componente se exportan de los esquemáticos en una forma lógica, para posteriormente ser representadas en una forma física en el PCB por medio del enrutamiento. El mapeo propuesto contiene el pin de inicio y el pin de llegada.

Por medio de un archivo JSON, se almacena la información descrita anteriormente, como se muestra en la Figura 4.4. Así se pueden exportar los componentes y la interconexión al algoritmo para iniciar con el procesamiento.

```
{
    "pins": [
        {
            "pin_number": "A11",
            "pin_name": "GND",
            "location_x": "64.2042",
            "location_y": "3.06"
        },
    "package": [
        {
            "length": "43.00",
            "width": "43.00",
            "location_x": "52.64",
            "location_y": "0.00"
        },
        {
            "length": "8.00",
            "width": "142.00",
            "location_x": "0.00",
            "location_y": "69.23"
        }
    ],
    "target": [
        {
            "pin_name": "CK_M_DDR0_A_DN",
            "location_x": "69.73",
            "location_y": "74.21"
        },
```


Figura 4.4: Ejemplo del archivo JSON que contiene los datos extraídos del PCB

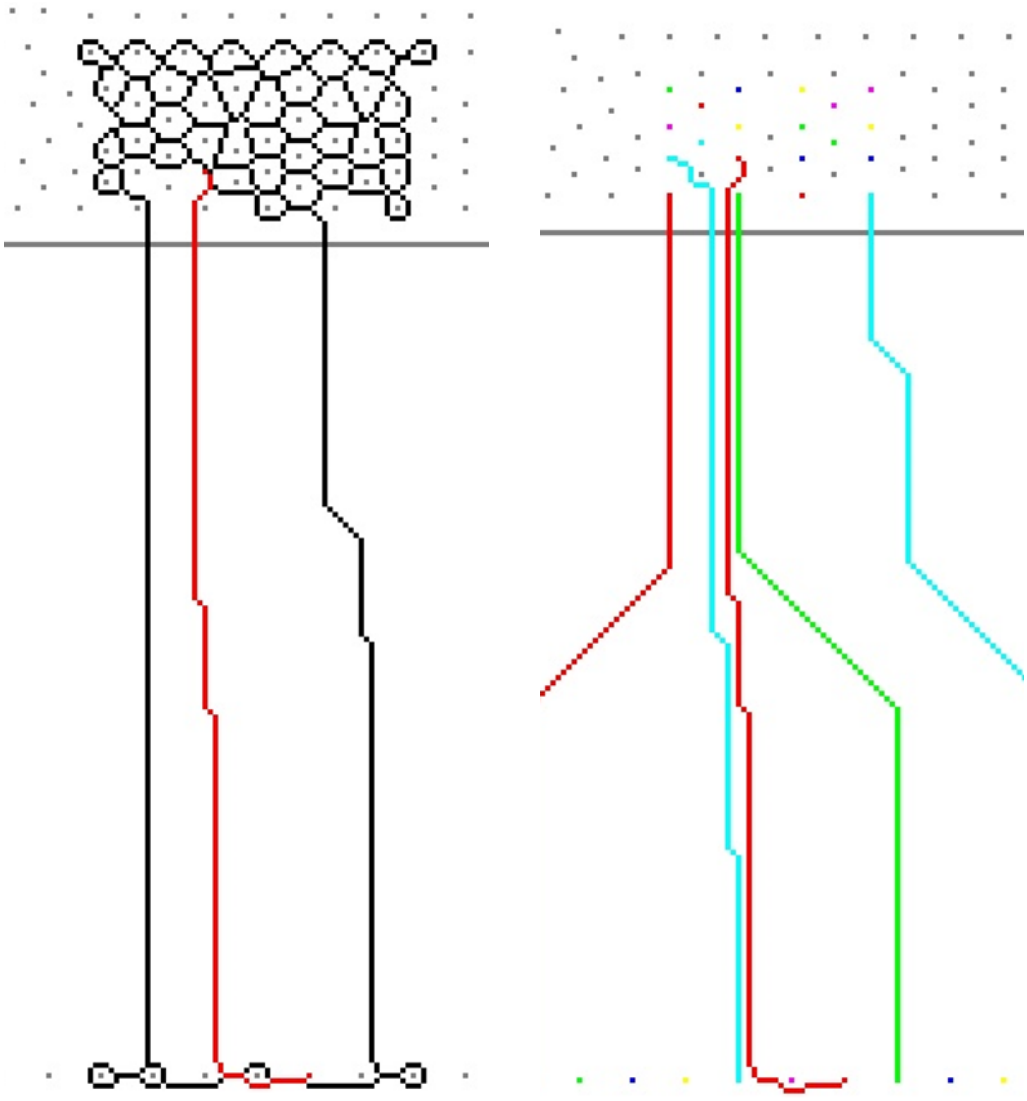
4.2.2 Visualización de los datos

Para la metodología de evaluación es clave poder visualizar los trazos con sus diferentes posibilidades de enrutamiento. Buscando tener diferentes niveles de evaluación, se utilizaron dos métodos para esto. La forma de evaluación de más alto nivel se hizo por medio de imágenes, donde se visualizan dos imágenes por trazo:

- Imagen con los posibles trazos y el trazo seleccionado, Figura 4.5(a).
- Imagen los todos los trazos del grupo de señales enrutadas hasta el momento, Figura 4.5(b).

En la forma de evaluación a un nivel más bajo se exporta la cuadrícula a una hoja de cálculo, dando la posibilidad de visualizar en detalle la información contenida por cada posición de la cuadrícula. Además de utilizar este método para la evaluación del trazo final, también se utilizó para la evaluación de las diferentes etapas del algoritmo.

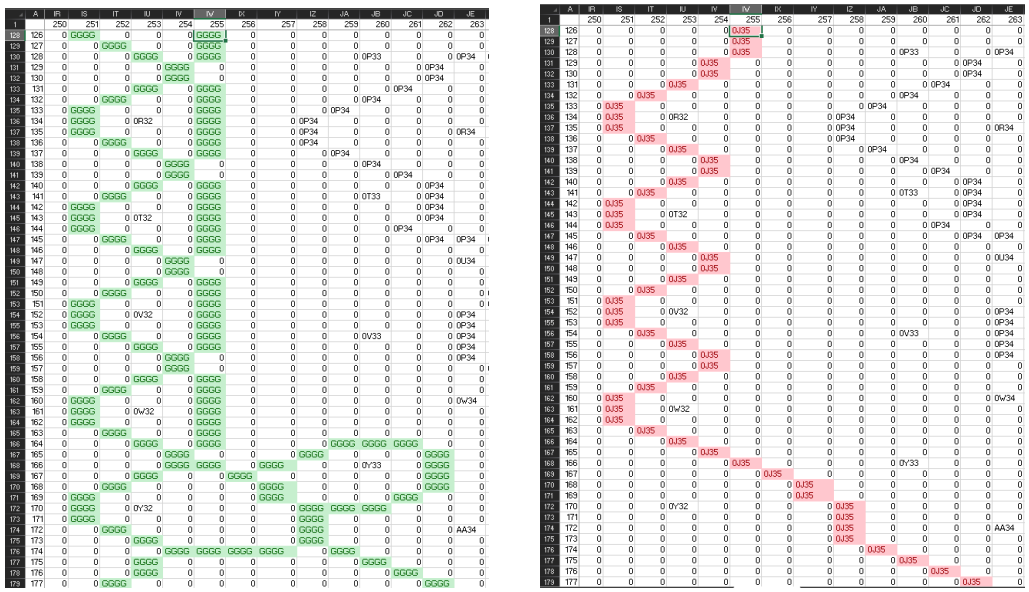
En la Figura 4.6 se observa un ejemplo donde se utilizó este método para la depuración del código. Debido a que el trazo seleccionado, como se muestra en la Figura 4.6(b), no es razonable ya que se selecciona un trazo en zigzag, pero la Figura 4.6(a) muestra que existe una posibilidad de un trazo más directo.



(a) Imagen con los posibles trazos y el trazo seleccionado, color rojo.

(b) Imagen los todos los trazos del grupo de señales enrutados hasta el momento, señal color rojo.

Figura 4.5: Visualización de los datos por medio de imágenes



(a) Imagen con los posibles trazos (verde). (b) Imagen con el trazo seleccionado (rojo).

Figura 4.6: Casos de enrutamiento simples en cuadrículas pequeñas

4.3 Algoritmo propuesto

El algoritmo busca obtener el enrutamiento de un grupo de señales entre dos componentes electrónicos de forma válida y razonable que siga los objetivos y criterios de evaluación definidos en la Sección 2.

Para llevar a cabo el desarrollo del algoritmo de una forma óptima, se toma la decisión de dividir el problema a solucionar en varios subproblemas, donde se obtienen dos etapas como se observa en la Figura 4.7.

Figura 4.7: Etapas del algoritmo propuesto

La etapa 1, es el núcleo del algoritmo, la cual resuelve el enrutamiento de una señal indi-

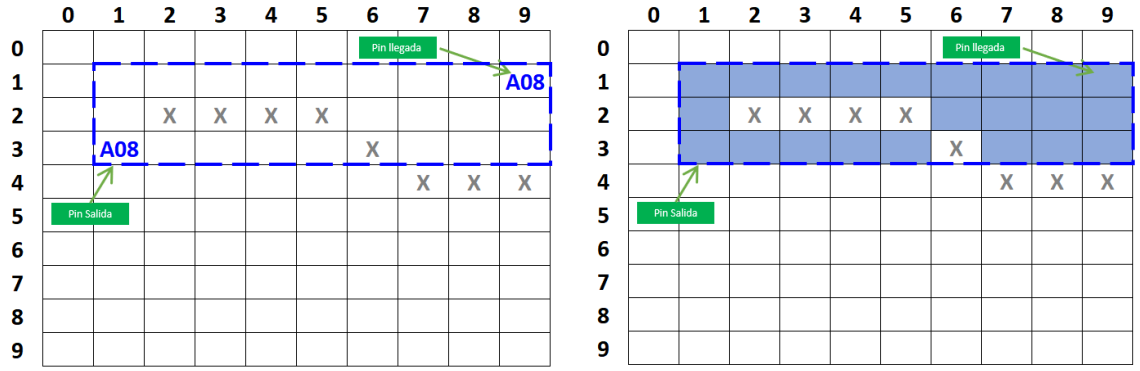
vidual. Esta etapa, es la encargada de resolver el enrutamiento entre dos pines cubriendo el enrutamiento de escape y general, siendo capaz de evitar obstáculos y seguir los requerimientos mínimos. Para realizar esto, la etapa 1 se subdivide en 5 subetapas, como se muestra en la Figura 4.8

Figura 4.8: Diagrama de las etapas del algoritmo.

La etapa 2 envuelve a la etapa 1, la cual resuelve el enrutamiento de un grupo de señales y tiene como propósito optimizar el uso del área y el largo de los trazos, así como enrutar la mayor cantidad de señales posibles. De esta forma, se simplifica el desarrollo y verificación de la solución global al poder realizar algoritmos pequeños con implementaciones simples.

4.3.1 Etapa 1

En el desarrollo de la investigación, se plateo una analogía la cual buscaba dar solución a encontrar el mejor camino de una señal para realizar el enrutamiento. Esta analogía consistió en arrojar un balde de pintura sobre un plano el cual contiene obstáculos, la pintura se iba a ir expandiendo sobre el plano y alrededor de los obstáculos mostrando así posibles caminos. Esta analogía describe en un alto nivel lo que hace un algoritmo de dilatación. En la Figura 4.9 muestra un ejemplo de esta analogía donde se está arrojando pintura azul.



(a) Identificación de los pines de entrada y salida.

(b) Plano con pintura azul extendida alrededor de los obstáculos.

Figura 4.9: Analogía del algoritmo de dilatación.

Trabajando sobre la idea de dilatación, se descubrió que la dilatación daba como resultado múltiples caminos, pero al mismo tiempo no eran caminos eficientes ya que estos podían tomar anchos de más de una posición. Por lo tanto el siguiente reto era disminuir estos caminos no razonables en caminos con un ancho de una posición para que se pudieran

asemejar a un trazo de una señal en un PCB. Fue ahí donde se planteó la pregunta ¿Y qué pasa si erosionamos los caminos hasta que tengan un ancho de una posición? Esta pregunta y además de la investigación sobre operadores morfológicos arrojó la existencia de 3 posibles operadores: erosión, adelgazamiento y esqueletización.

Una versión propia del operador morfológico de adelgazamiento permitió tener caminos con un ancho mínimo de una posición y así poder tener caminos semejantes a los trazos de las señales en un PCB. Así obtuvimos como resultado múltiples caminos para una señal, los cuales son válidos. Para seleccionar un camino final el criterio fue optar por el camino más corto y para ello se utilizó un algoritmo de Dijkstra.

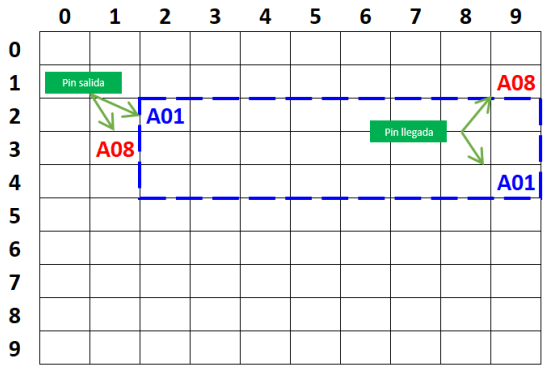
Para poder utilizar el algoritmo de Dijkstra, primero se construye un grafo de pesos en base a los caminos que se crearon por medio de la dilatación y el adelgazamiento. El algoritmo de Dijkstra utiliza el grafo y da como resultado el camino más corto. Este camino se utiliza como base para la última subetapa la cual busca realizar una optimización. El algoritmo de Dijkstra permite obtener el camino más corto, pero se debe construir un grafo con base en los caminos que se construyen por medio de la dilatación y el adelgazamiento.

Dilatación

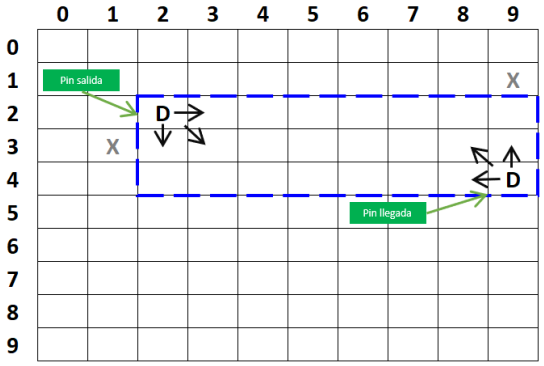
La dilatación es el algoritmo encargado de mostrar todas esas posibles posiciones por donde se pueden enrutar una señal, en la Figura 4.10 muestra un ejemplo simplificado de cómo se lleva a cabo este algoritmo. La dilatación consiste en múltiples iteraciones sobre el área definida, buscando posiciones libres que se puedan usar para el enrutamiento de la señal, pero estas posiciones libres deben ser vecino de una posición ya dilatada. Este algoritmo también indica si el área seleccionada es suficiente para tener interconexión entre el pin de salida y llegada. Si no se llega a tener una interconexión mínima el área de dilatación se extiende y se aplica el mismo algoritmo.

La dilatación inicia con la identificación de la posición de los pines de salida y llegada, donde se delimita un área de trabajo entre estas posiciones, como se observa en la Figura 4.10(a). Una vez definida el área de dilatación, se inicia con el proceso de dilatación convirtiendo los pines de llegada y salida en posiciones dilatadas, Figura 4.10(b). Se inician las iteraciones sobre el área de dilatación y se van dilatando las posiciones vecinas que se encuentran libres, Figura 4.10(c). Finalmente obtendremos un área dilatada, Figura 4.10(d), donde las posiciones marcadas con una "D" son candidatas a ser posibles caminos entre el pin de salida y llegada.

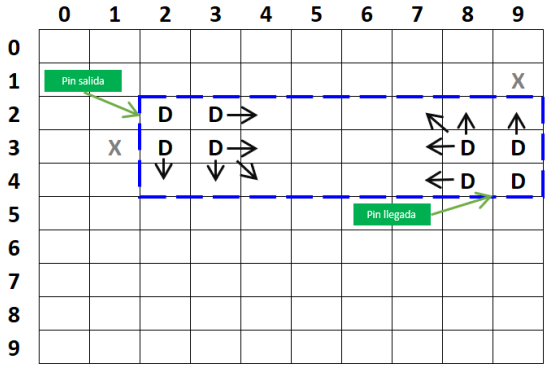
Dependiendo los obstáculos existentes, se puede dar el caso de que el área de dilatación seleccionada no sea suficiente para tener una interconexión entre el pin de salida y llegada. Cuando esto sucede se extiende el tamaño del área de dilatación, este incremento del



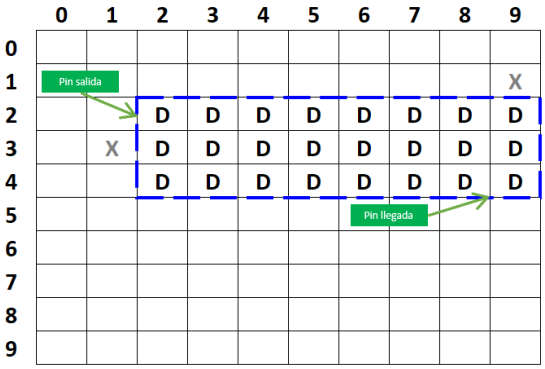
(a) Identificación de los pines de llegada y salida y área de dilatación.



(b) Pines dilatados y evaluación de los vecinos.



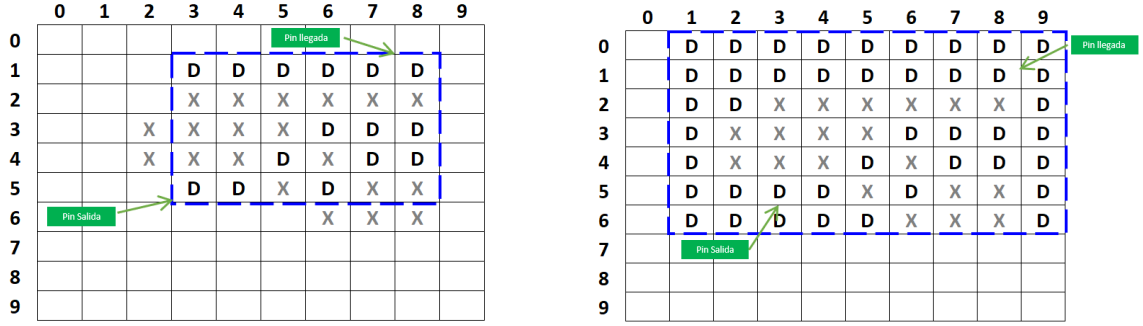
(c) Dilatación de los vecinos de los pines y evaluación de los siguientes vecinos.



(d) Dilatación total del área de trabajo.

Figura 4.10: Ejemplo simplificado del algoritmo de dilatación.

área es configurable y permite obtener más posiciones las cuales pueden ser candidatas a ser posibles caminos. La Figura 4.11 ejemplifica este caso, donde la dilatación inicial no permite tener interconexión entre los pines de salida y llegada, como se muestra en la Figura 4.11(a). El siguiente paso es expandir en una posición el tamaño del área de dilatación para tener candidatos a posibles caminos que den una interconexión entre los pines, como se observa en la Figura 4.11(b).



(a) Dilatación sin posibilidades de interconexión entre los pines de salida y llegada.

(b) Dilatación total en un área extendida.

Figura 4.11: Ejemplo simplificado del algoritmo de dilatación con un área extendida.

En la Figura 4.12 se muestra un ejemplo de dilatación para un caso de estudio mediano el cual presenta una mayor complejidad. Este ejemplo se seguirá utilizando para explicar las diferentes etapas del algoritmo. Para este caso el área de dilatación es la que está dentro del cuadro punteado de color azul, donde se observa la señal dilatada con pequeños cuadros negros, además de múltiples obstáculos en cuadros grises (los obstáculos pueden ser pines y otros trazos).

Adelgazamiento

El objetivo del adelgazamiento es tomar todos los posibles caminos generados en la dilatación y convertirlos en caminos de un ancho mínimo de una posición. Para obtener estos caminos se aplican máscaras sobre cada posición del área dilatada las cuales verifican si es posible liberar dicha posición. Este proceso de aplicación de máscaras consiste en múltiples iteraciones sobre el área de dilatación buscando posiciones las cuales puedan ser convertidas a posiciones libres y así obtener el efecto del adelgazamiento sobre los caminos.

El adelgazamiento para esta solución en específico tiene como requisito no desconectar posibles caminos. Debido a esta razón es que se desarrolló una versión propia del operador morfológico de adelgazamiento ya que este operador no garantiza este requerimiento y se podían tener casos donde se eliminaran caminos válidos. Además, la etapa de adelgazamiento realiza una optimización del área utilizada por cada trazo, así dejando área libre para que pueda ser tomada por otros trazos en caso de ser necesaria.

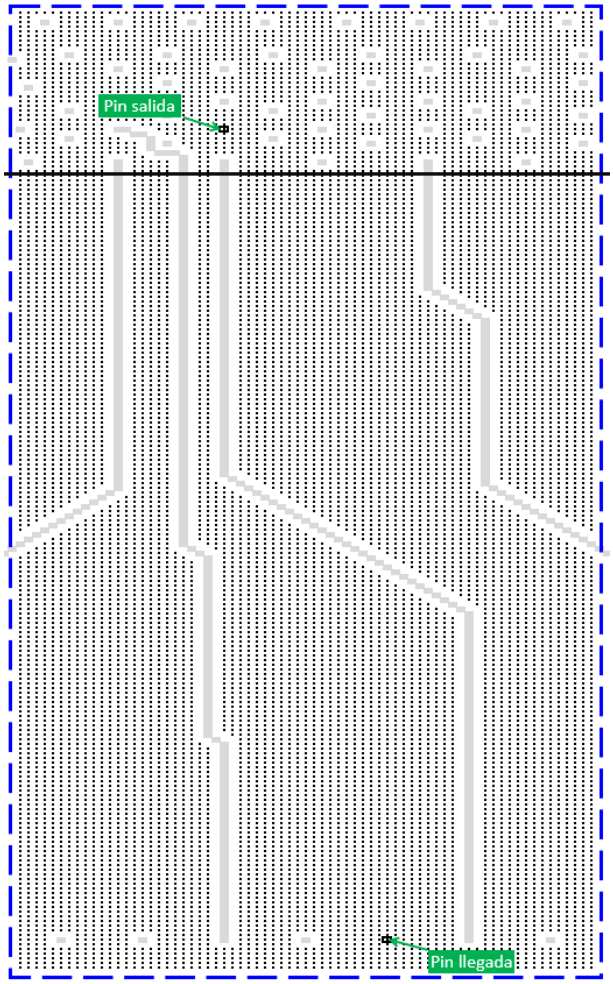


Figura 4.12: Ejemplo de dilatación.

El adelgazamiento se divide en tres etapas, cada etapa aplica diferentes máscaras las cuales fueron optimizadas para atacar de una forma adecuada cada etapa del enrutamiento y así poder obtener caminos mínimos. Incluso estas etapas tratan el área dilatada desde diferentes aristas donde se aplican evaluaciones con y sin sesgo, adelgazamientos con aproximaciones desde afuera hacia adentro o viceversa, e inclusive la verificación de posiciones dilatadas sin un posible camino.

Las máscaras tienen una nomenclatura donde se colocan sobre una posición y evalúan a sus vecinos para decidir si la posición evaluada se puede liberar. Cada posición de la máscara tiene un valor el cual es comparado con el vecino según corresponda. Si una posición de la máscara tiene un "1" indica que debe ser una posición dilatada, cuando se tiene un "0" significa que no deber ser una posición dilatada y al tener una posición con una "X" no se toma en toma en cuenta en la evaluación.

Adelgazamiento etapa 1

La etapa 1 consiste en liberar las posiciones dilatas de afuera hacia dentro. Para ello se utilizan 4 diferentes máscaras, como muestra la Figura 4.13. Estas máscaras liberan las esquinas exteriores del área dilatada. Son máscaras de un tamaño de 3x3 donde la posición a evaluar está en el centro de la máscara.

X	0	0
1	**D**	0
1	1	X

(a) Máscara 1, elimina la esquina superior derecha.

1	1	X
1	**D**	0
X	0	0

(b) Máscara 2, elimina la esquina inferior derecha.

0	0	X
0	**D**	1
X	1	1

(c) Máscara 3, elimina la esquina superior izquierda.

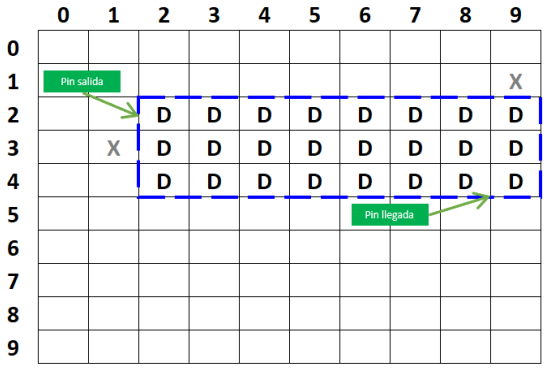
X	1	1
0	**D**	1
0	0	X

(d) Máscara 4, elimina la esquina inferior izquierda

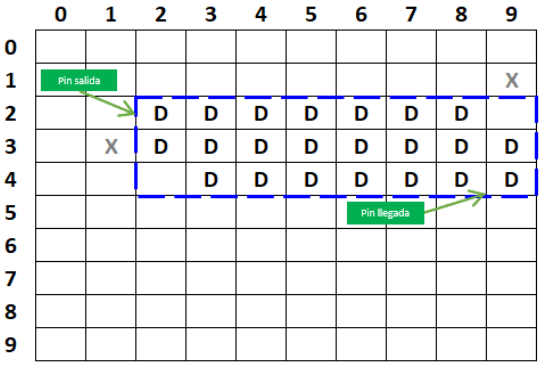
Figura 4.13: Máscaras utilizadas para la etapa 1 del adelgazamiento.

En la Figura 4.14 se puede observar un ejemplo simplificado donde se muestra el paso a paso de la aplicación de la máscara 1 y 4. Se puede observar que se van liberando las posiciones dilatadas iniciando por las esquinas que no son pines, hasta obtener el mínimo camino posible por esta etapa. Este camino es un camino balanceado el cual sitúa una diagonal de ancho de dos posiciones en el centro entre los dos pines.

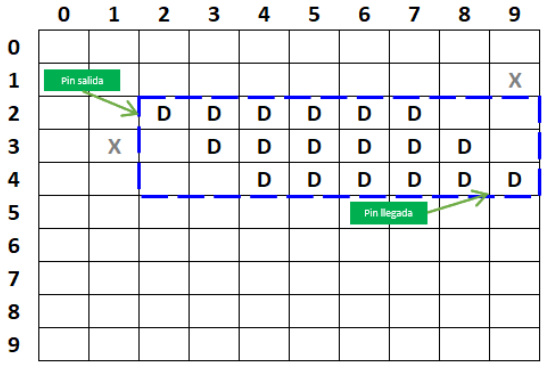
Estas máscaras trabajan muy bien con áreas dilatadas las cuales tienen formas rectangulares, como por ejemplo en el área de enrutamiento general el cual típicamente presenta esta característica. Para esta etapa, la aplicación de cada máscara se realiza sin un sesgo



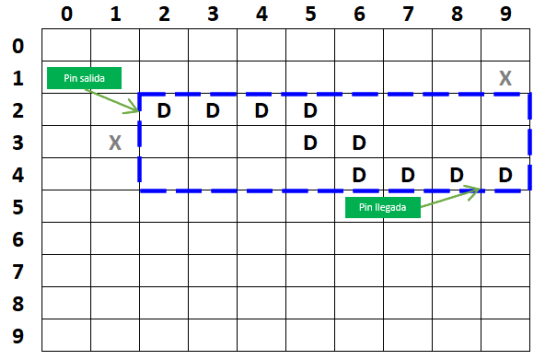
(a) Dilatación total del área de trabajo.



(b) Primera iteración del adelgazamiento aplicando las máscaras 1 y 4.



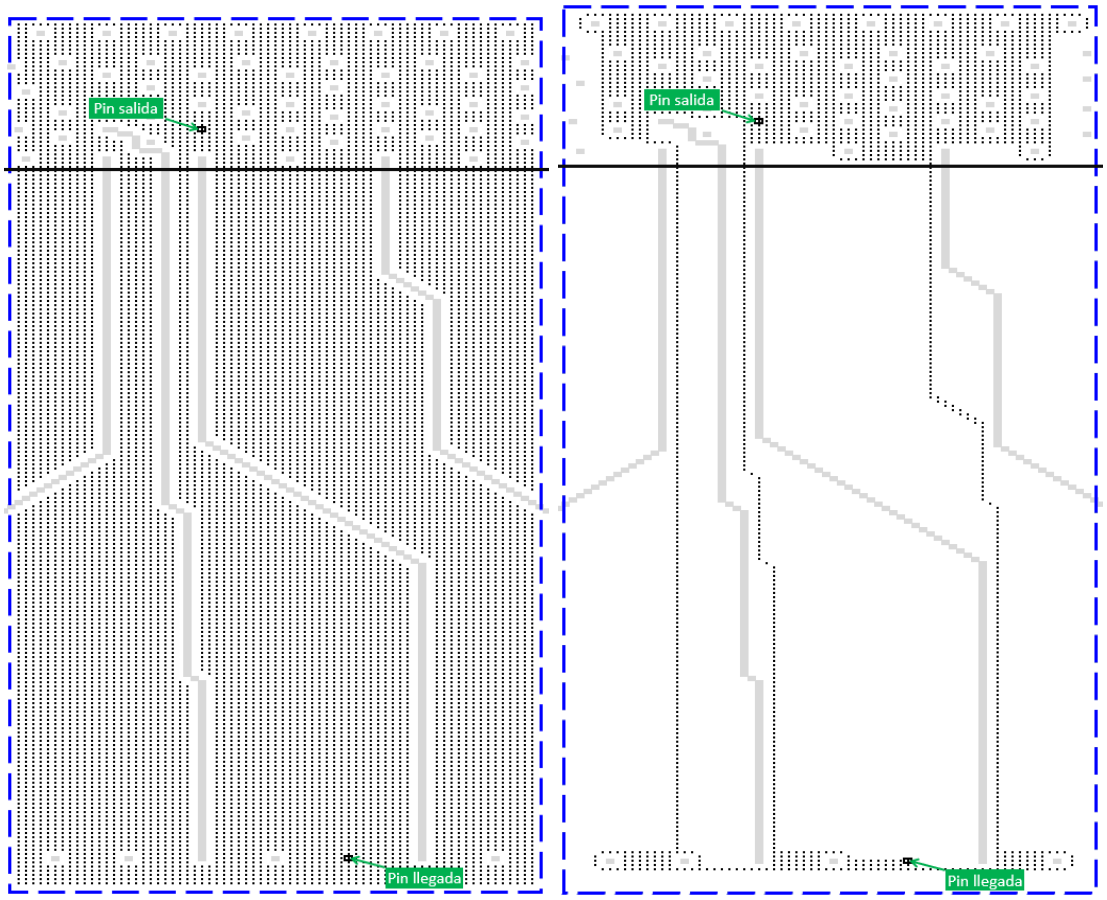
(c) Segunda iteración del adelgazamiento aplicando las máscaras 1 y 4.



(d) Camino mínimo resultante después de aplicar la etapa 1 del algoritmo de adelgazamiento.

Figura 4.14: Ejemplo simplificado del algoritmo de adelgazamiento para la etapa 1.

sobre las liberaciones realizadas dentro de la misma iteración que cubre el área dilatada. Esto ayuda a que la etapa no libere más posiciones de lo debido, garantizando la conectividad entre los caminos. En la Figura 4.15 se puede observar el comportamiento antes descrito, donde el enrutamiento general tiene un adelgazamiento robusto pero el enrutamiento de escape solo obtiene un adelgazamiento ligero en su parte exterior.



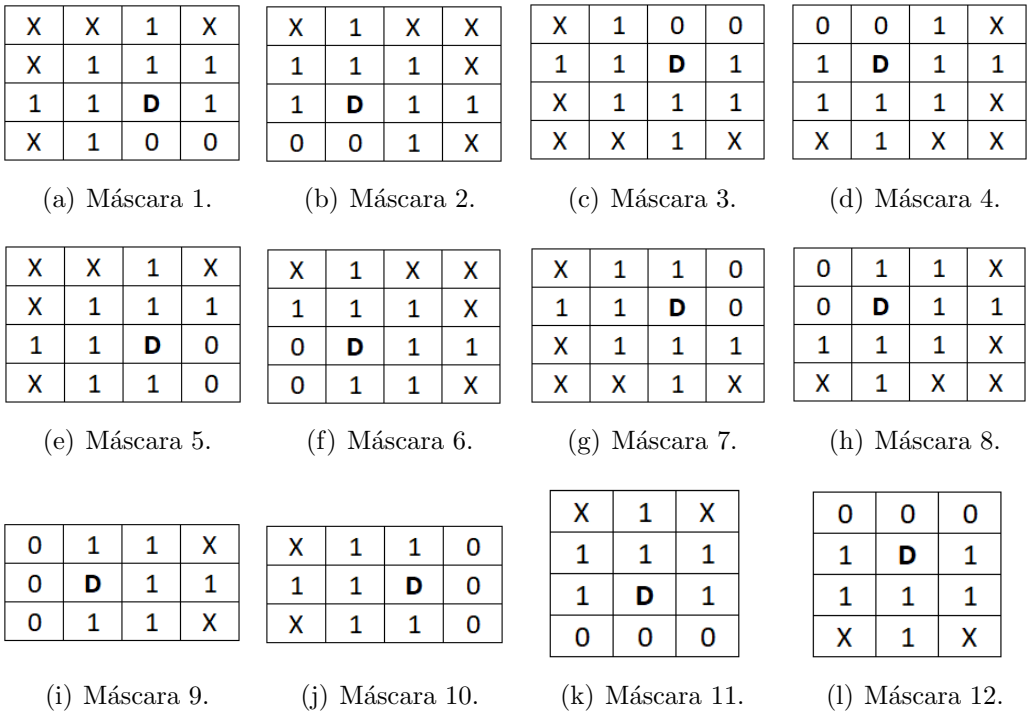
(a) Etapa de dilatación. (b) Etapa 1 de adelgazamiento

Figura 4.15: Comparación entre etapa de dilatación y etapa 1 del adelgazamiento.

Adelgazamiento etapa 2

La etapa 2 consiste en liberar las posiciones dilatadas de adentro hacia afuera. Para ello se utilizan 12 diferentes máscaras, como se muestra en la Figura 4.16. Son máscaras con tamaños variables de 4x4, 4x3 y 3x4 donde la posición a evaluar no está en el centro de la máscara.

Estas máscaras se enfocan en el área del enrutamiento de escape, donde existen muchos obstáculos, realizando un adelgazamiento simétrico que busca no tener un favorecimiento hacia alguna coordenada. Igual que la etapa anterior, en la evaluación de las máscaras se



(a) Máscara 1. (b) Máscara 2. (c) Máscara 3. (d) Máscara 4.

(e) Máscara 5. (f) Máscara 6. (g) Máscara 7. (h) Máscara 8.

(i) Máscara 9. (j) Máscara 10. (k) Máscara 11. (l) Máscara 12.

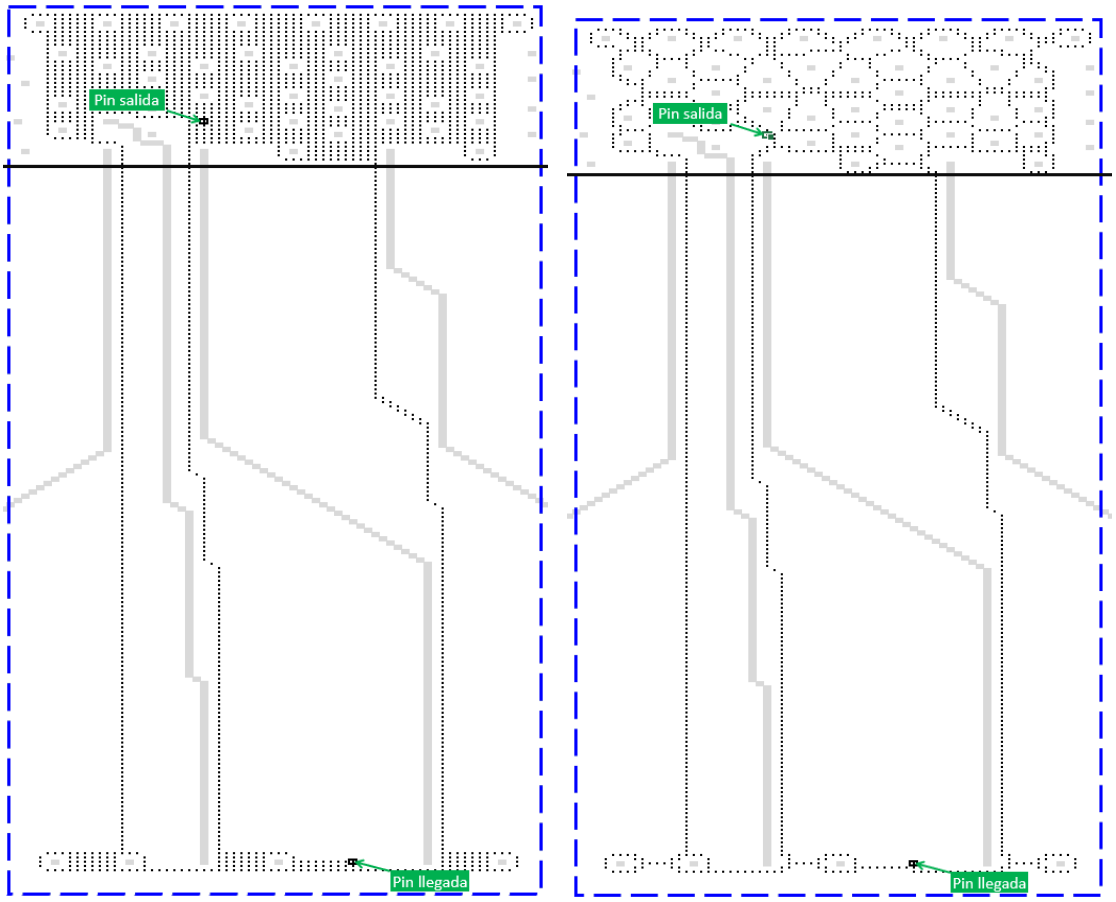
Figura 4.16: Máscaras utilizadas para la etapa 2 del adelgazamiento.

realiza sin sesgo de las liberaciones realizadas dentro de la misma iteración que cubre el área dilatada. La Figura 4.17 se puede observar el comportamiento antes descrito.

Adelgazamiento etapa 3

La etapa 3 consiste en terminar el adelgazamiento del camino, donde se optimizan los caminos adelgazando los de ancho de dos posiciones y eliminando esquinas. Esto permite obtener caminos con ángulos de 45 grados y con un ancho de una posición. Para ello se utilizan 4 diferentes máscaras, observable en la Figura 4.18. Son máscaras de un tamaño de 3x3 donde la posición a evaluar está en el centro de la máscara.

Esta etapa se asemeja mucho a la etapa 1, donde también se quitan esquinas. Pero si se comparan las Figuras 4.13 y 4.18, se puede observar que las máscaras no son iguales y además, la aplicación de cada máscara para la etapa 3 se realiza con sesgo sobre las liberaciones realizadas dentro de la misma iteración que cubre el área dilatada. Esto ayuda a que la etapa no libere más posiciones de lo debido, garantizando la conectividad entre los caminos. En la Figura 4.19 se puede observar el comportamiento antes descrito.



(a) Etapa 1 de adelgazamiento. (b) Etapa 2 de adelgazamiento.

Figura 4.17: Comparación entre etapa 1 del adelgazamiento y la etapa 2 del adelgazamiento.

X	X	X
1	**D**	X
0	1	X

(a) Máscara 1, elimina la esquina superior derecha.

0	1	X
1	**D**	X
X	X	X

(b) Máscara 2, elimina la esquina inferior derecha.

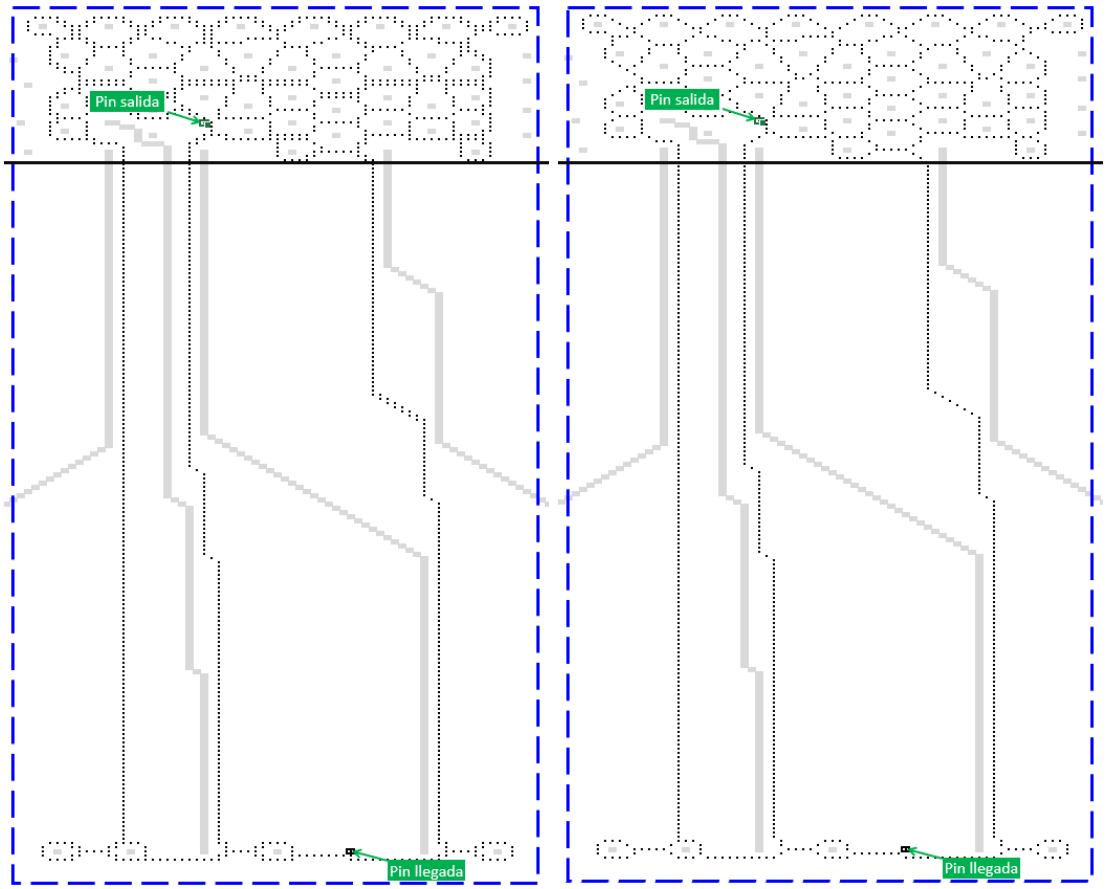
X	X	X
X	**D**	1
X	1	0

(c) Máscara 3, elimina la esquina superior izquierda.

X	1	0
X	**D**	1
X	X	X

(d) Máscara 4, elimina la esquina inferior izquierda

Figura 4.18: Máscaras utilizadas para la etapa 3 del adelgazamiento.



(a) Etapa 2 de adelgazamiento. (b) Etapa 3 de adelgazamiento.

Figura 4.19: Comparación entre etapa 1 del adelgazamiento y la etapa 2 del adelgazamiento.

Construcción del grafo

Esta etapa consiste en tomar los caminos resultantes del adelgazamiento y convertirlos en un grafo. Para esto se realiza un barrido el cual inicia en el pin de salida y va sobre toda el área dilatada donde se cuenta la cantidad de posiciones para cada camino, dando como resultado un peso por cada camino.

Un camino se define como las posiciones dilatadas que entrelazan dos vértices. Un vértice para esta implementación en particular se puede definir como una posición dilatada la cual tiene 3 o más vecinos dilatados. Por lo tanto, los pesos de los caminos irán de vértice a vértice, como en una representación típica de un grafo.

Hay dos formas de asignar pesos, la primera es cuando se tiene una transición vertical u horizontal donde se da un peso de una unidad. La segunda es cuando se tiene una transición en diagonal donde se le da un peso de 1.4 unidades. Esto se realiza suponiendo que cada posición es cuadrada y entendiendo que un movimiento en diagonal es más costoso que un movimiento vertical u horizontal.

A continuación se muestra un ejemplo simplificado de la construcción de un grafo. De este ejemplo se obtiene al final 4 caminos y un vértice. Teniendo los caminos resultantes de la etapa de adelgazamiento, se identifica el pin de salida y llegada. Los pines de salida y llegada son agregados a la lista de vértices y se inicia el procesamiento por el pin de salida, donde se identifican sus vecinos, como se muestra en la Figura 4.20.

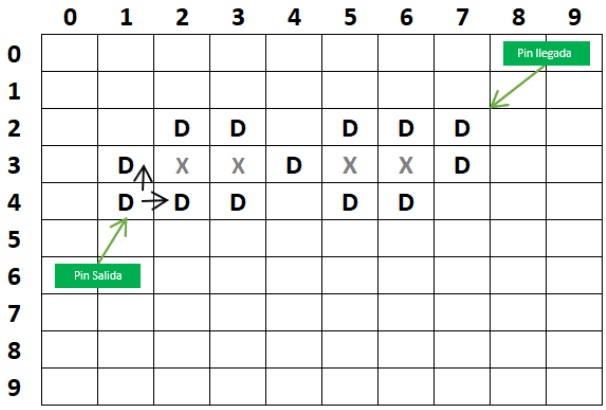


Figura 4.20: Identificación del pin de salida y sus vecinos dilatados.

El pin de salida tiene dos vecinos dilatados, por lo tanto se selecciona un vecino para ser procesado. En este caso el vecino que va hacia el norte y el otro vecino, el que va hacia el este, se encola y se deja como pendiente. El procesamiento de un camino consiste en ir vecino por vecino e ir sumando el peso según el moviente hasta encontrar un vértice, como se muestra en la Figura 4.21. En ese momento el procesamiento del camino se termina,

para este ejemplo seria el camino que va hacia el norte, se identifica el vértice el cual es agregado a la lista de vértices y se inicia con el procesamiento del camino que se dirige al este.

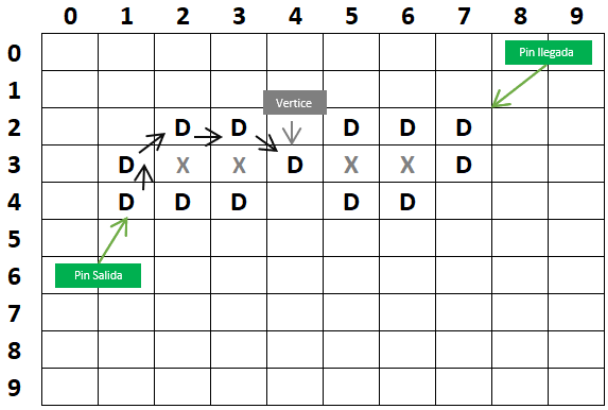


Figura 4.21: Procesamiento del camino norte hasta la identificación del vértice.

El procesamiento del camino que toma dirección al este ocurre de la misma forma que el procesamiento del camino norte, como se muestra en la Figura 4.22 . Una vez procesados los dos posibles caminos que tiene el pin de salida, se realiza el procesamiento de los posibles caminos del pin de llegada, Figura 4.23, ya que este fue el segundo vértice que se encolo y posteriormente se continua con los vértices identificados.

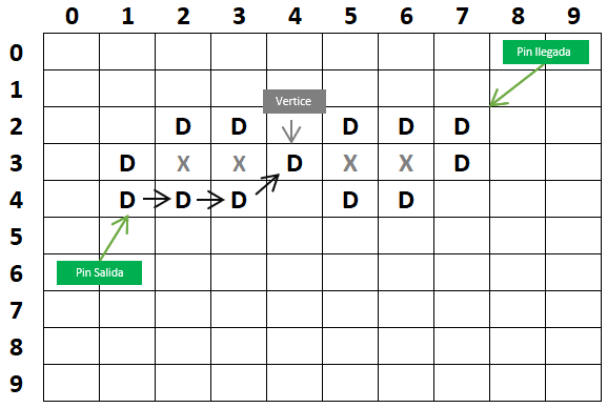


Figura 4.22: Procesamiento del camino este hasta la identificación del vértice.

Durante este procesamiento de las posiciones, se van guardando las posiciones que conforman cada camino entre dos vértices, para así realizar una construcción del camino final de una forma más rápida. En la Figura 4.24 se muestra cómo se realiza el cálculo de los pesos entre posiciones y pesos totales por camino.

En la Figura 4.25 se puede ver la representación del grafo resultante que se muestra en la Figura 4.24.

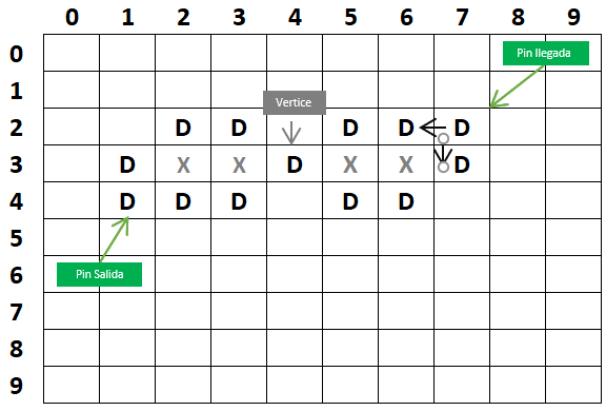


Figura 4.23: Identificación de los vecinos dilatados del pin de llegada.

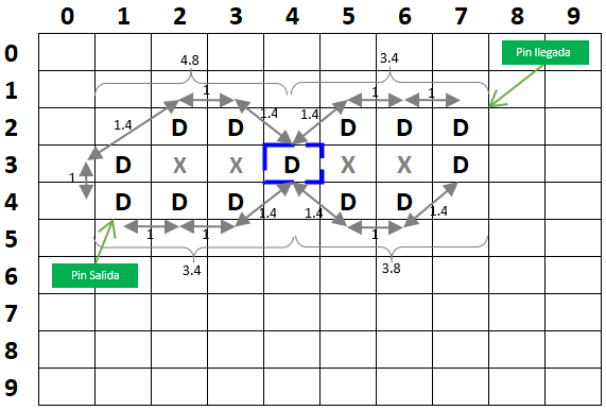


Figura 4.24: Caminos con sus respectivos pesos.

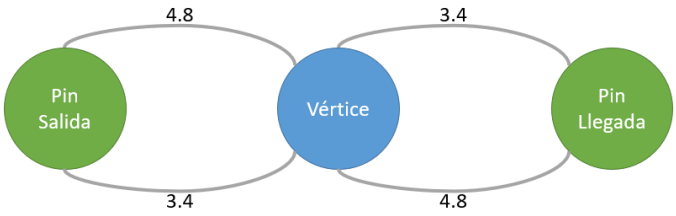


Figura 4.25: Representación del grafo construido.

Selección y construcción del camino

Una vez se tiene un grafo construido se ejecuta el algoritmo de Dijkstra, el cual se basa en una implementación propuesta por [13] la cual fue adaptada para el manejo de las estructuras de datos utilizadas en nuestro algoritmo. Este da como resultado un lista con los vértices que crean el camino más corto.

Teniendo la lista de vértices se recorre tomando el vértice n y el $n+1$. Con esta información se puede recorrer la lista de caminos construida en la Sección 4.3.1 la cual nos da cada posición que conforma el camino entre los vértices n y $n+1$. Así se procede a cambiar el etiquetado de las posiciones dilatadas que van a crear el camino final. Posteriormente se hace un barrido por el área dilatada limpiando las posiciones que no fueron seleccionadas como camino final. La Figura 4.26 muestra un ejemplo del camino resultante después de aplicar el algoritmo de Dijkstra.

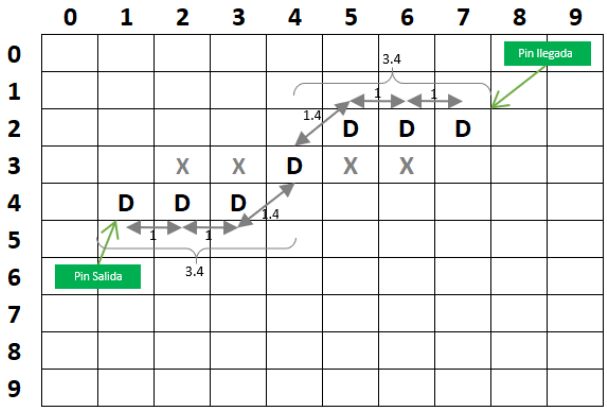


Figura 4.26: Camino mínimo resultante después de aplicar el algoritmo de Dijkstra.

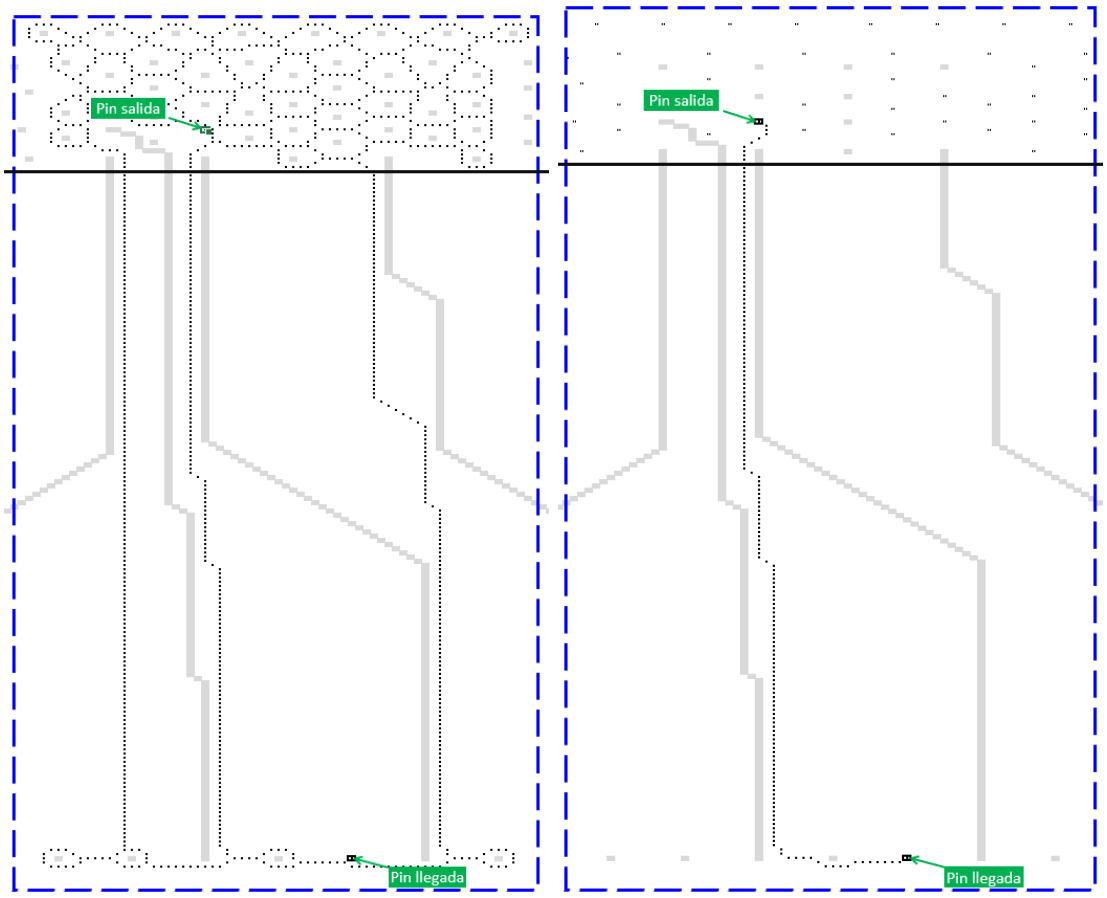
En la Figura 4.27 se observa una comparación del resultado obtenido de la etapa de adelgazamiento, donde se observan múltiples opciones de camino y el resultado final de la selección y construcción del camino final para este trazo.

En la Figura 4.28 se muestra otro ejemplo, donde la igual forma se compara el resultado del adelgazamiento y el resultado final de la selección del camino.

Optimización del camino

Como objetivo se busca obtener caminos más cortos para minimizar el área de enrutamiento. Para esto, se crea un algoritmo iterativo el cual recorre todo el camino buscando optimizaciones en la longitud de este. Para realizar esta optimización en la longitud, el camino se va dividiendo en vecindarios.

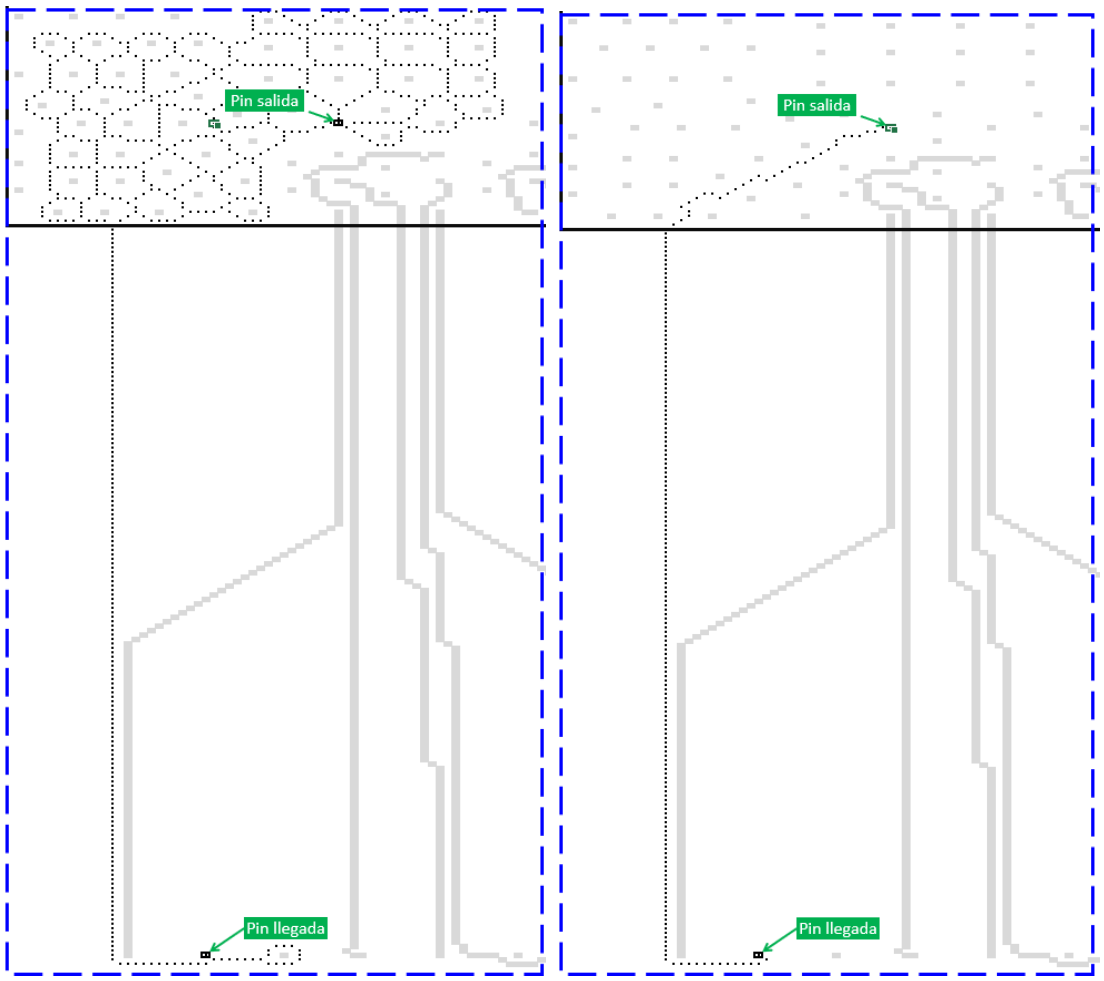
En la Figura 4.29 se puede observar una parte de un camino donde se resalta un vecindario de 10 posiciones. El algoritmo toma la posición de inicio y va creando un camino hipotético, como se muestra en las posiciones azules de la imagen mencionada.



(a) Caminos obtenidos de la etapa de adelgazamiento.

(b) Camino final obtenido a partir de etapa de selección y construcción del camino.

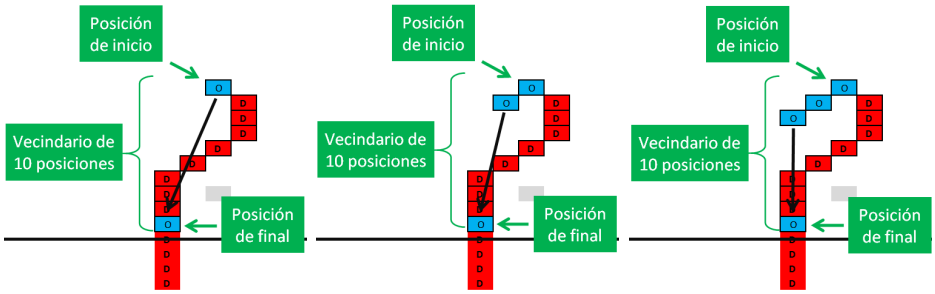
Figura 4.27: Comparación entre etapa 2 del adelgazamiento y la etapa 3 del adelgazamiento.



(a) Caminos obtenidos de la etapa de adelgazamiento.

(b) Camino final obtenido a partir de etapa de selección y construcción del camino.

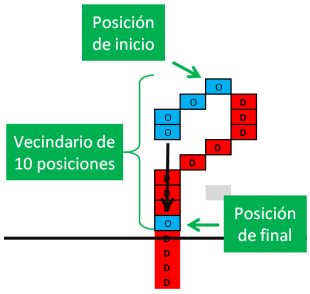
Figura 4.28: Otra comparación entre etapa 2 del adelgazamiento y la etapa 3 del adelgazamiento.



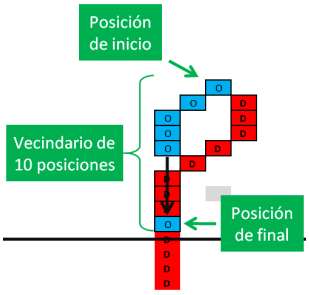
(a) Iteración 1 del camino hipotético.

(b) Iteración 2 del camino hipotético.

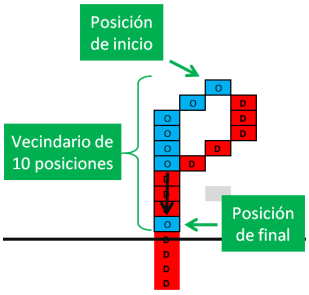
(c) Iteración 3 del camino hipotético.



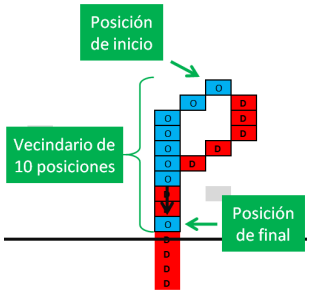
(d) Iteración 4 del camino hipotético.



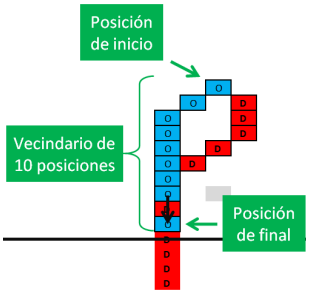
(e) Iteración 5 del camino hipotético.



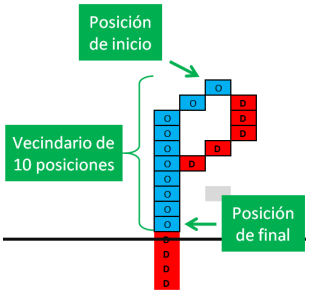
(f) Iteración 6 del camino hipotético.



(g) Iteración 7 del camino hipotético.



(h) Iteración 8 del camino hipotético.



(i) Iteración 9 del camino hipotético.

Figura 4.29: Ejemplo de optimización de un vecindario de 10 posiciones.

El camino hipotético es un trazado lineal con un direccionamiento en múltiplos de 45 grados (0, 45, 90, 135, 180, 225 y 270 grados) que busca obtener la distancia más corta entre la posición de inicio y final del vecindario. Este toma la posición de inicio y se mueve hacia la siguiente vecino más cercano a la posición final y así continua hasta llegar a la posición final cumpliendo con el espacio mínimo. Cabe destacar que si el vecino siguiente más cercano seleccionado está ocupado, ya sea por otro camino o por un pin, el algoritmo termina la optimización para ese vecindario y descarta el camino hipotético. En la Figura 4.30 se muestra los posibles caminos, donde la posición en rojo es el inicio y las posiciones en azul son las posibilidades para crear el camino hipotético.

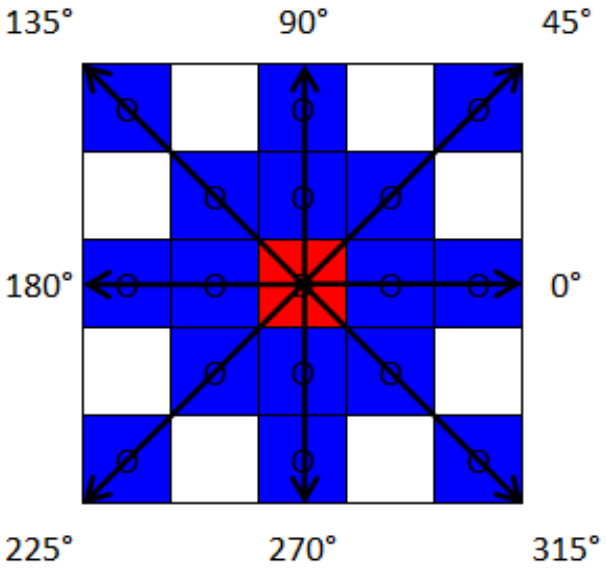
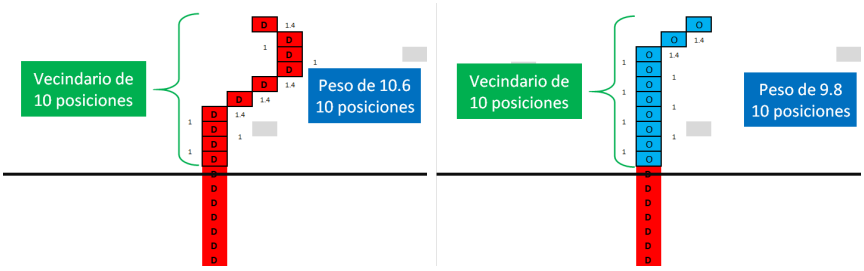


Figura 4.30: Ejemplo con posibles caminos hipotéticos.

Una vez que se tiene el camino hipotético, se da una comparación de los pesos entre el camino no optimizado (posiciones en rojo) y el camino hipotético. En la Figura 4.31 muestra la comparación que se describió, donde el camino no optimizado tiene un peso de 10.6 y el camino hipotético tiene un peso de 9.8. Este es uno de los casos más simples donde la optimización fue de 0.8 y se dio por medio de la mejora de la ubicación del camino. Por lo tanto para este caso, el algoritmo selecciona el camino hipotético y descarta el camino no optimizado.

Cuando el algoritmo optimiza un vecindario se mueve al siguiente, este movimiento tiene una cadencia de una posición para así obtener el mejor resultado en la optimización. En la Figura 4.32 se puede observar una secuencia de imágenes las cuales ejemplifican el comportamiento donde el algoritmo recorre el camino para realizar la optimización con un vecindario de 10 posiciones.

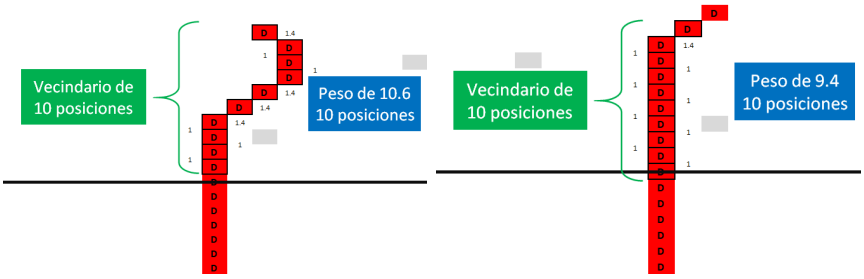
Los vecindarios pueden ser de diferentes tamaños, en los ejemplos anteriores se utilizó un vecindario de 10 posiciones. En la Figura 4.33, se toma un vecindario de 50 posiciones y se presenta un caso donde se da una optimización alta. Se inicia con un vecindario que tiene un peso de 51.4 el cual rodea un pin. Al aplicar la optimización este pasa a tener un



(a) Vecindario de 10 posiciones, no optimizado.

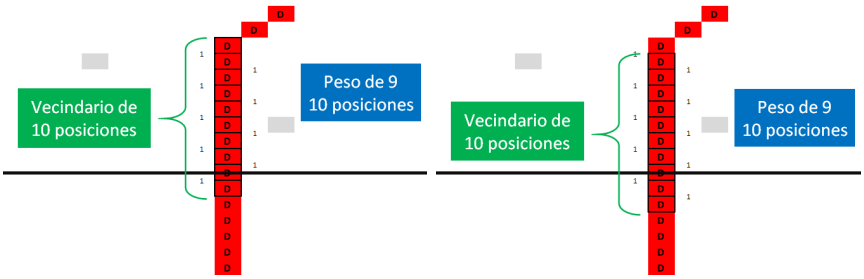
(b) Vecindario de 10 posiciones, optimizado.

Figura 4.31: Comparación de un vecindario de 10 posiciones sin optimizar y optimizado.



(a) Primera vecindario de 10 posiciones.

(b) Segundo vecindario de 10 posiciones.

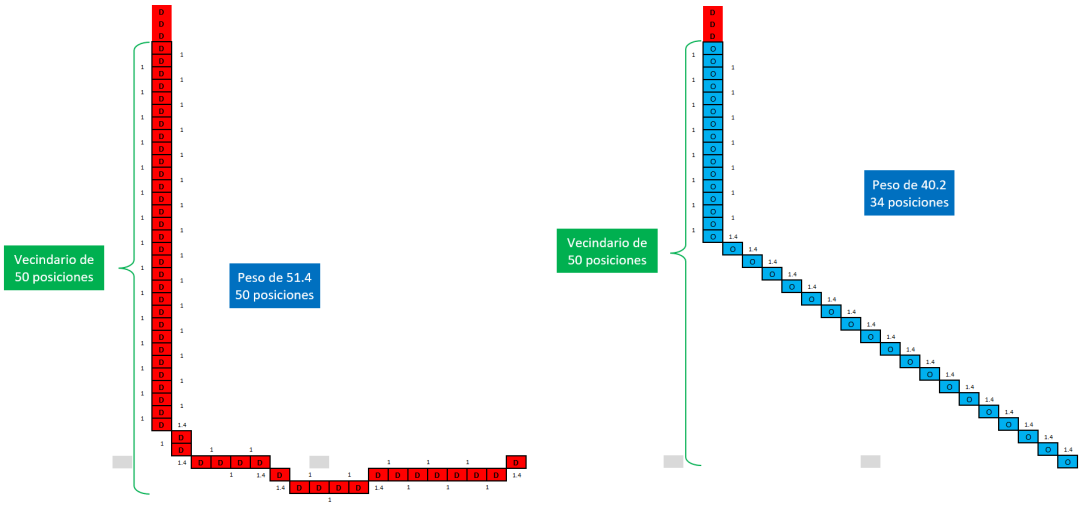


(c) Tercer vecindario de 10 posiciones.

(d) Cuarto vecindario de 10 posiciones.

Figura 4.32: Ejemplo de cómo el algoritmo recorre el camino con un vecindario de 10 posiciones.

peso de 40.2, es decir se dio una mejora en 11.2 en el peso. También se da una disminución de 16 posiciones al pasar de 50 a 34. En otras palabras la optimización no solo pasa por la ubicación del camino, sino también por la reducción del largo del vecindario.



(a) Vecindario de 50 posiciones, no optimizado. (b) Vecindario de 50 posiciones, optimizado.

Figura 4.33: Comparación de un vecindario de 50 posiciones sin optimizar y optimizado.

Se implementó un experimento para determinar cuál es el mejor tamaño de vecindario. Este experimento consistió en enrutar múltiples señales con múltiples tamaños de vecindario, mediante un caso base el cual no tiene ninguna optimización proveniente de la subetapa anterior. Para este experimento cada señal va a tener el mismo ecosistema y solo se varia el tamaño del vecindario, creando casos de enrutamiento controlados donde su única variable es el tamaño de vecindario.

Los tamaños de vecindarios seleccionados variaron desde 4 a 60, con incrementos de 2 unidades. Para cada grupo de señales que fueron enrutadas con el mismo vecindario, se calculó el promedio de la longitud de cada camino, el cual funciona como punto de comparación y así obtener el tamaño de vecindario óptimo.

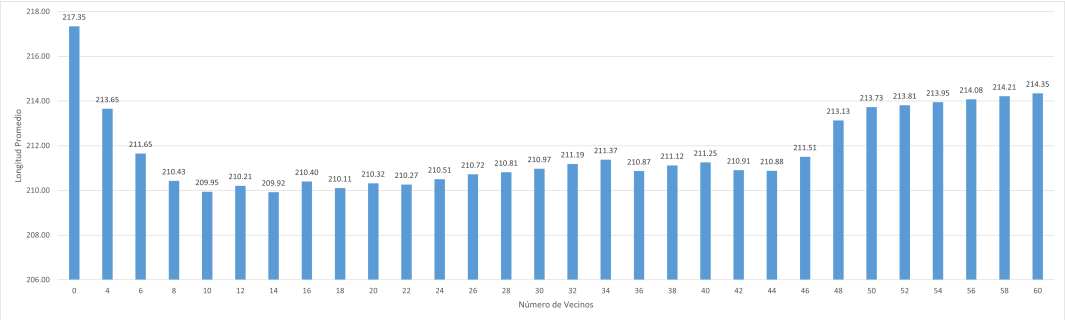
Este experimento fue evaluado con el caso de estudio mediano y el caso de estudio grande, definidos en las Secciones 4.1.2 y 4.1.3, donde para cada caso de estudio se obtuvieron resultados diferentes.

Para el caso del estudio mediano, se observó una disminución de la longitud del camino hasta llegar a un vecindario con tamaño de 10, el cual se mantuvo con muy poca variación hasta un vecindario con un tamaño de 14. A partir de 16, se vió una tendencia a la alza pero es muy leve hasta llegar a 48.

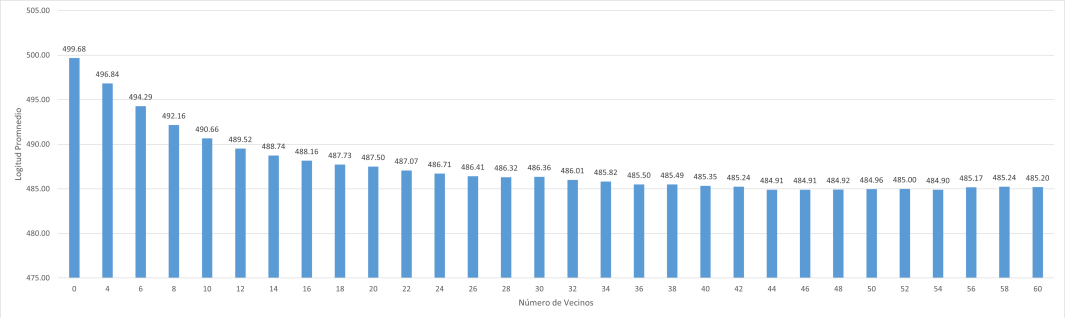
En el caso de estudio grande, se observó una tendencia en la disminución hasta un vecindario con un tamaño de 44, el cual se mantuvo con muy poca variación hasta un vecindario con tamaño de 54.

En la Figura 4.34 se pueden observar los resultados del promedio de la longitud de señales

enrutadas de acuerdo con el número de vecinos- para ambos casos de estudio.



(a) Resultados para el caso de estudio mediano.



(b) Resultados para el caso de estudio grande.

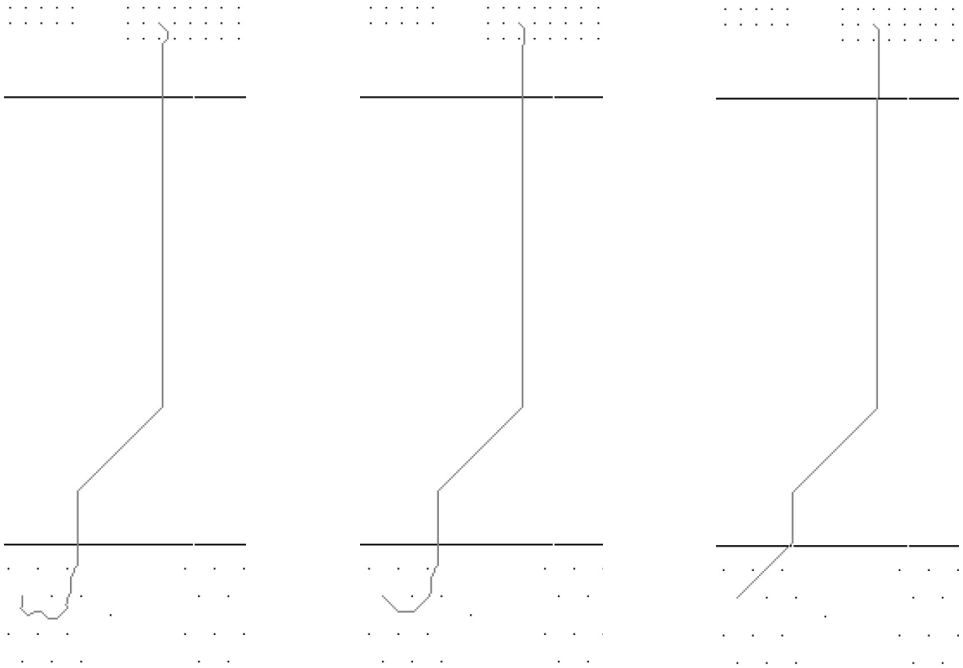
Figura 4.34: Resultados obtenidos al evaluar diferentes tamaños de vecindarios.

En general, se puede observar que los vecindarios grandes como en el de 40 posiciones, su optimización es más significativa en regiones de enrutamiento general o regiones donde haya mucho espacio y la optimización debe ser más gruesa. En la Figura 4.35 se observa una optimización muy significativa en el área sur, al usar un vecindario de 40 posiciones y tener como resultado un trazo completamente lineal que incluso fue capaz de evadir dos pines.

Por otro lado, vecindarios pequeños como el de 10 posiciones, se comportan mejor en las regiones del enrutamiento de escape, donde el espacio es limitado y la optimización debe ser más fina. En la Figura 4.36 se puede visualizar en el área de enrutamiento de escape una mejor optimización cuando se utiliza un vecindario de 10 posiciones.

Al tener resultados buenos tanto con vecindarios grandes como pequeños y observando que estos se comportan mejor según la región de enrutamiento, se decidió implementar una combinación de vecindarios. Se utilizaron dos tamaños de vecindarios, ejecutando primero el de tamaño más grande y después el de tamaño más pequeño. Por lo tanto, en base a los resultados anteriores se evaluaron 3 combinaciones:

- Vecindario de 40 posiciones combinado con un vecindario de 10 posiciones.
- Vecindario de 44 posiciones combinado con un vecindario de 14 posiciones.
- Vecindario de 50 posiciones combinado con un vecindario de 10 posiciones.

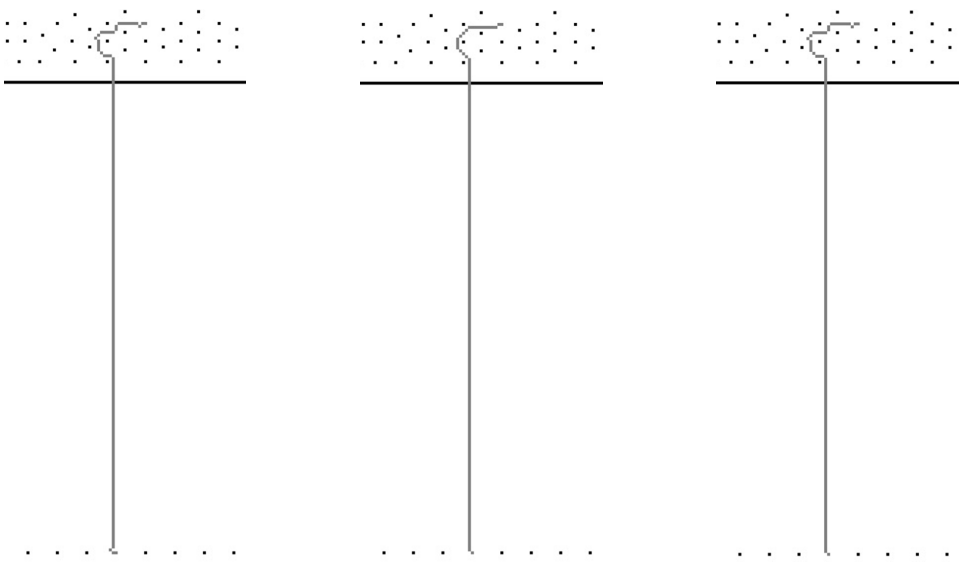


(a) Camino sin optimización.

(b) Camino con una optimización basado en un vecindario de 10 posiciones.

(c) Camino con una optimización basado en un vecindario de 40 posiciones.

Figura 4.35: Ejemplo de optimización para el caso de estudio grande.



(a) Camino sin optimización.

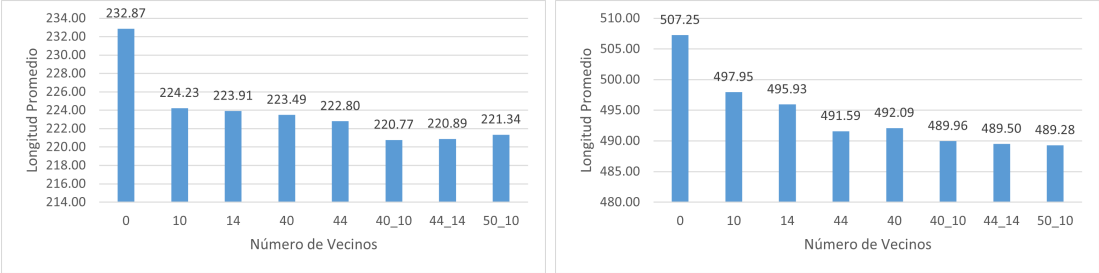
(b) Camino con una optimización basado en un vecindario de 10 posiciones.

(c) Camino con una optimización basado en un vecindario de 40 posiciones.

Figura 4.36: Ejemplo de optimización para el caso de estudio mediano.

El mejor resultado para el caso de estudio mediano se obtuvo de la combinación de vecindarios con tamaños de 40 y 10. Para el caso de estudio grande el mejor resultado se dio con la combinación de vecindarios con tamaños de 50 y 10. En este punto se tenía como semejanza que al utilizar la combinación de dos vecindarios se obtuvo una optimización en la longitud de las señales al compararlo con el mejor resultado de vecindarios de un solo tamaño. Por lo tanto, se optó por utilizar vecindarios combinados.

Se optó por la combinación con tamaños de vecindarios más pequeños, es decir la combinación de 40 y 10. En la Figura 4.37 se pueden observar los resultados del promedio de la longitud de señales enrutadas de acuerdo con la combinación de vecindarios.



(a) Resultados para el caso de estudio mediano. (b) Resultados para el caso de estudio grande.

Figura 4.37: Resultados obtenidos al evaluar diferentes combinaciones de vecindarios.

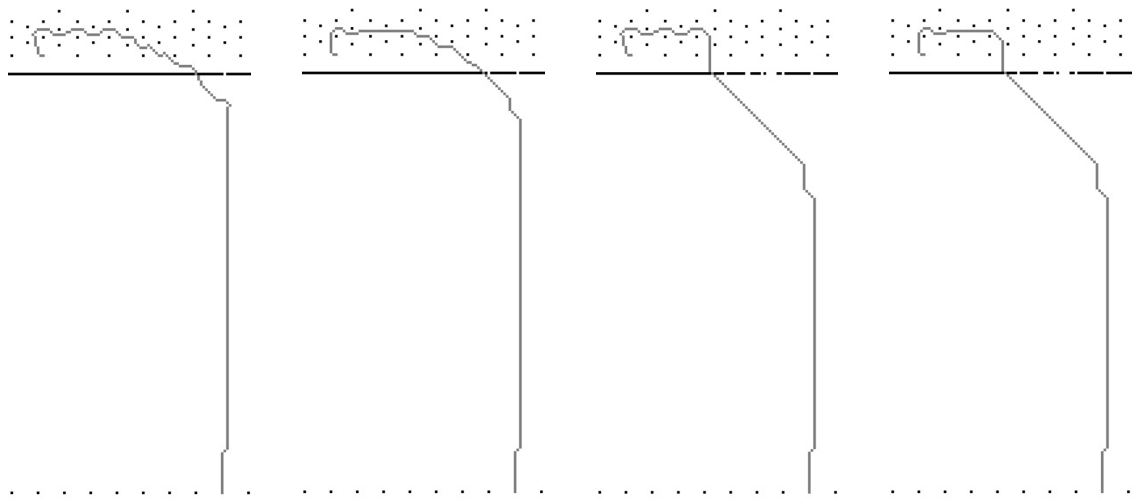
En la Figura 4.38 se puede ver una comparación entre las optimizaciones con vecindarios individuales de 10, 40 y vecindarios combinados de 40 y 10, donde se observa que la optimización con vecindarios combinados da como resultado un camino con un trazo más efectivo y de menor tamaño.

4.3.2 Etapa 2

La etapa 2 resolvió el enrutamiento de grupo de señales y tiene como propósito optimizar el uso del área, minimizar el largo de los trazos y conectar la mayor cantidad de señales posibles. Para lograr este objetivo, se utilizó como base la etapa 1 que da como resultado la conexión de una señal individual de forma óptima, pero esta señal interactúa directamente con las otras señales. El orden en que se procesan las señales afecta directamente el enrutamiento individual y para verificar cual estrategia de ordenamiento es más óptimo se realizaron diferentes algoritmos.

Estrategia 1: Orden de enrutamiento aleatorio

Se realizó el enrutamiento de las señales, dándole a estas un orden aleatorio. Para efecto de comparación, se buscó un caso base, el cual ruteó 48 señales con una longitud promedio de 487.78. En la Figura 4.39 se puede observar el resultado de utilizar un orden aleatorio.



(a) Camino sin optimización.

(b) Camino con una optimización basado en un vecindario de 10 posiciones.

(c) Camino con una optimización basado en un vecindario de 40 posiciones.

(d) Camino con una optimización basado en un vecindario de 40 y 10 posiciones.

Figura 4.38: Comparación de optimización entre vecindarios individuales y vecindarios combinados para el caso de estudio mediano.

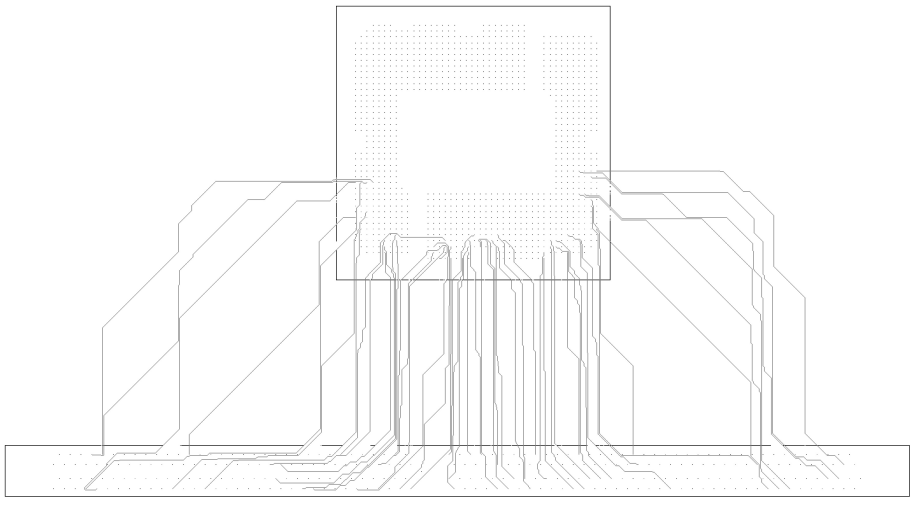


Figura 4.39: Resultado de enrutamiento de grupo utilizando la estrategia 1: orden de enrutamiento aleatorio.

Estrategia 2: Menor distancia entre el punto de inicio y final

Esta estrategia de ordenamiento viene dada por la suposición de que a menor distancia el enrutamiento de la señal será más directo. Para esto se calculó la distancia lineal entre el punto de inicio y el punto final para cada señal a enrutar. Una vez calculada la distancia, se ordenaron las señales de menor a mayor distancia y se enrutaron. En la Figura 4.40 se muestra el resultado al utilizar un orden que prioriza la menor distancia entre el punto de inicio y final, enrutando 57 señales con una distancia promedio de 497.29.

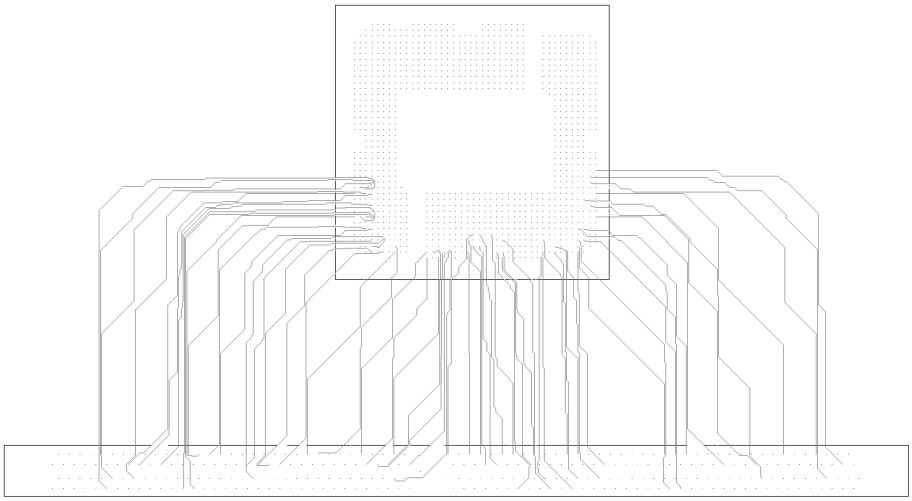


Figura 4.40: Resultado de enrutamiento de grupo utilizando la estrategia 2: menor distancia entre el punto de inicio y final.

Estrategia 3: Centroide más próximo

La motivación detrás de esta estrategia es buscar trazar señales contiguas, iniciando con la señal que este más cercano al centro. Para esto se calcula el centroide para cada grupo de pines, tanto de inicio y de final. Después se calcula para cada señal la distancia lineal entre cada centroide y sus puntos de inicio y final. Estas dos distancias se suman creando un peso, la señal con el menor peso será la señal que se enrutará.

Una vez que se enruta la primer señal, se busca la señal más próxima, calculando la distancia entre la señal enrutada y sus vecinos no enrutados. Para esto se calcula la distancia entre el punto de inicio y final de la señal enrutada y sus vecinos. Así, sumando las dos distancias para crear un peso y se enruta la de menor peso. Esto se repite, hasta que se enruten todas las señales.

En la Figura 4.41 se visualiza el resultado al utilizar la estrategia 3 con el enrutamiento de 69 señales con un promedio de longitud de 532.91, dando una mejora considerable en comparación con la estrategia 2.

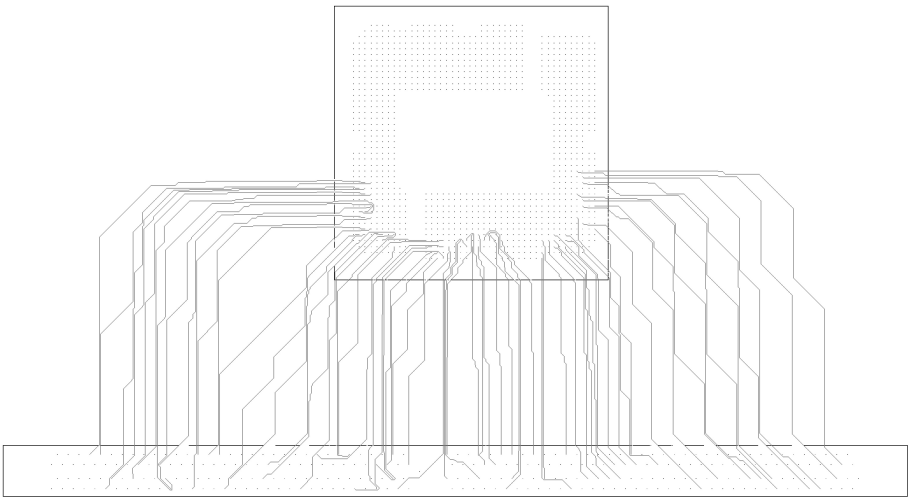


Figura 4.41: Resultado de enrutamiento de grupo utilizando una estrategia de centroide más próximo.

Estrategia 4: Centroide más próximo con menor distancia

Analizando los resultados anteriores, se observa que utilizando la estrategia 2 se dan trazos más directos pero con mayor área sin utilizar entre los trazos. Para la estrategia 3 hay trazos con más curvas, pero se optimiza el área sin utilizar entre los trazos. Por lo tanto, se optó por realizar una combinación entre estas dos estrategias, utilizando el algoritmo de centroide más próximo y agregando el peso de la menor distancia entre el punto de inicio y final, ordenando las señales de menor a mayor peso. Esto dio como resultado el enrutamiento de 67 señales con un promedio de 512.94. En la Figura 4.42 se muestra el enrutamiento.

En comparación con la estrategia 3, no se obtuvo un mejor resultado ya que se enrutaron menos señales. Lo cual llevó a continuar investigando más estrategias para el orden del enrutamiento.

Estrategia 5: Menor distancia más próximo

Al no tener un mejor resultado con la estrategia 4 y siguiendo con el objetivo de optimizar el uso del área, minimizar el largo de los trazos y enrutar la mayor cantidad de señales posibles, se optó por realizar otra estrategia. Esta toma la estrategia 2 en combinación con la estrategia de la señal más próxima. Para ello, se calcula la distancia entre el punto de inicio y final para cada una de las señales a enrutar, se toma la señal con la menor distancia y se enruta. Posteriormente, se calcula la distancia entre los puntos de inicio y final de esta señal enrutada y sus vecinos no enrutados, así enrutando la señal más próxima. En la Figura 4.43 muestra el resultado para esta estrategia donde se enrutan 73 señales con un promedio de longitud de 507.87.

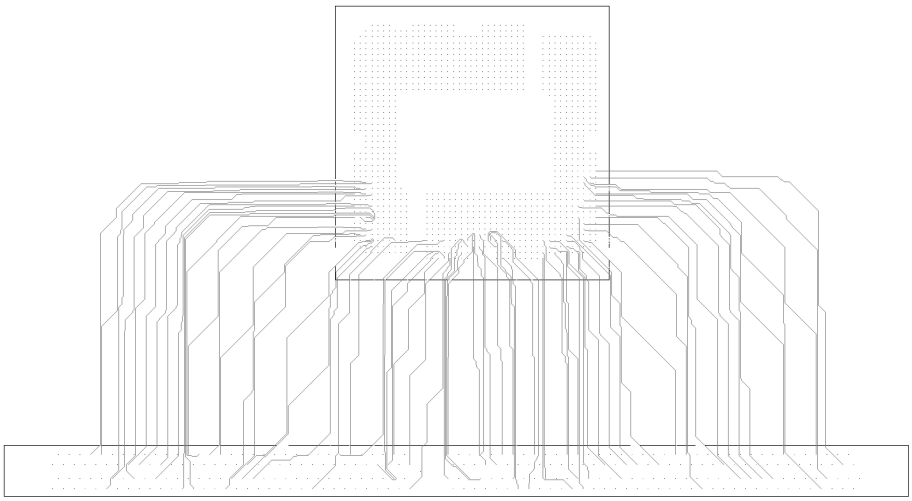


Figura 4.42: Resultado de enrutamiento de grupo utilizando una combinación entre las estrategias 2 y 3.

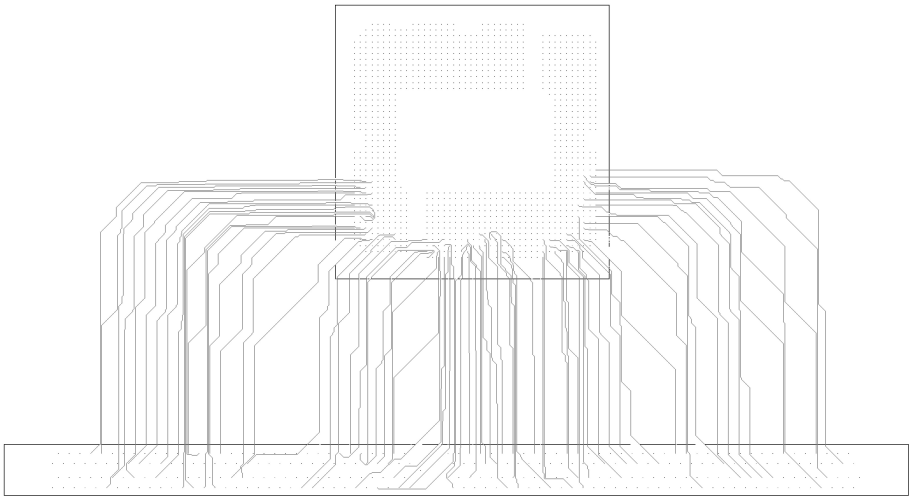


Figura 4.43: Resultado de enrutamiento de grupo utilizando la estrategia 5.

Capítulo 5

Resultados y análisis

Esta capítulo presenta los resultados obtenidos para cada uno de los tres casos de estudio propuestos y un análisis en base a los objetivos y criterios de evaluación. Además, se realiza un análisis de las etapas más críticas del algoritmo y su impacto en el resultado de cada una de ellas. Para finalizar, se ofrece una comparación entre la herramienta comercial Allegro, donde se valoran puntos a favor y en contra del algoritmo propuesto.

5.1 Análisis del algoritmo

5.1.1 Adelgazamiento

El adelgazamiento es una de las subetapas más críticas del algoritmo, ya que esta da como resultado todos los caminos posibles con un ancho mínimo, donde uno de estos caminos será la base del camino final. Como indica la Sección 4.3.1, se realiza la aplicación de diferentes máscaras que buscan adelgazar el camino. En el desarrollo del algoritmo se buscó evitar tener un preferencia hacia alguna coordenada del área dilatada: norte, sur, este u oeste. Por lo tanto, se introdujo adrede un sesgo que da como resultado un camino que se adelgaza hacia el centro del área dilatada, como se muestra en la Figura 5.1. Esto da como resultado un trazo con un espaciamiento equidistante entre sus caminos vecinos.

Este sesgo hacia el centro tiene sentido, cuando se analiza que se quiere tener un adelgazamiento que sea genérico, ajustable en cualquier dirección de enrutamiento y que pueda utilizar tanto en el enrutamiento de escape y general, así obteniendo un algoritmo que no favorece ninguna coordenada. Sin embargo, al realizar el enrutamiento de un grupo de señales se observa que se generan muchas áreas libres, las cuales pueden ser utilizadas por otros trazos. Esto se muestra en la Figura 5.2, donde se observan espacios que quedan con áreas no utilizadas, haciendo que el área de enrutamiento no sea tan densa como podría ser.

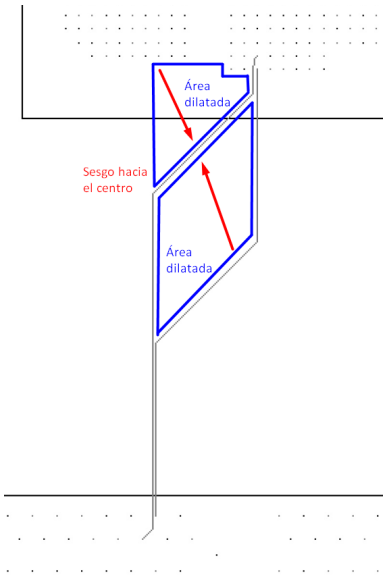


Figura 5.1: Camino resultante del adelgazamiento con sesgo hacía el centro del área de dilatación.

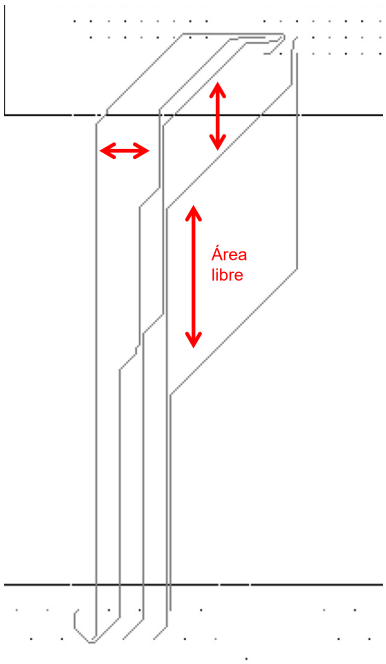


Figura 5.2: Área libre resultante de un camino con sesgo hacia el centro del área de dilatación.

5.1.2 Selección de vecindario

La selección del vecindario para la subetapa de optimización del camino puede crear variaciones en el camino final. Como se mencionó en la Sección 4.3.1, se realizaron experimentos con vecindarios individuales iniciando con un tamaño de 4 hasta 60, así como también pruebas con vecindarios combinados.

Para obtener el trazo más efectivo, lo ideal sería recorrer el camino con todos los tamaños de vecindarios posibles, pero esto haría que computacionalmente sea muy costoso. El algoritmo propuesto recalcula el camino para cada vecindarios, con lo cual, al tener mayor cantidad de vecindarios, mayor será el procesamiento y su tiempo de ejecución. Por esta razón se optó por tener una etapa de optimización con dos vecindarios combinados.

Al analizar los resultados individuales, se observa que hay un grupo de vecindarios, entre 10 y 14, que realiza una mejor optimización para el área de enrutamiento de escape y otro grupo de vecindarios, entre 40 y 50, que realiza una mejor optimización en el área de enrutamiento general. Por lo tanto, se opta por realizar un análisis con vecindarios combinados evaluando 3 combinaciones distintas:

- Vecindario de 40 posiciones combinado con un vecindario de 10 posiciones.
- Vecindario de 44 posiciones combinado con un vecindario de 14 posiciones.
- Vecindario de 50 posiciones combinado con un vecindario de 10 posiciones.

Para definir cuál es la mejor combinación de vecindarios se realizó un análisis del algoritmo, el cual recalcula el camino para cada vecindario, es decir, a mayor tamaño de vecindario, mayor procesamiento. Al analizar los resultados, la mayor variación entre los vecindarios combinados propuestos para el caso de estudio grande es de 0.13%. Se optó por la combinación con tamaños de vecindarios más pequeños, es decir la combinación de 40 y 10. Bajo el panorama descrito, el algoritmo realizó una menor cantidad de procesamiento para encontrar un camino más efectivo con muy poca variación en la optimización final. Teniendo como resultado una mejora de 12.1 y 17.29 la longitud promedio de las señales para los casos de prueba mediano y grande, respecto a las otras dos combinaciones. Así se logró mejorar las características deseadas desarrolladas en la Sección 2.3.

El uso finito de vecindarios tiene como objetivo mejorar el trazado resultante que se obtuvo de la subetapa de adelgazamiento, reduciendo la cantidad de curvas, longitud y el espacio del enrutamiento, pero esta técnica no es ideal ya que utiliza vecindarios estáticos. Se pueden encontrar casos donde la optimización no es la mejor debido a que el tamaño de vecindario limita esto, como por ejemplo en la Figura 5.3, donde se observa que las diagonales de las señales, marcadas con la flecha roja, no se optimizaron más. Esto debido a que se llegó al límite del vecindario, para este caso de 40 posiciones.

Si se quiere buscar una optimización para estas diagonales se deben utilizar vecindarios más grandes a 40 posiciones, donde no se sabe cuál es el límite para optimizar cada uno de los casos. Por lo tanto, una optimización ideal está sujeta al tamaño del vecindario, donde se debería tener un grupo de vecindarios infinitos para cubrir todos los casos posibles.

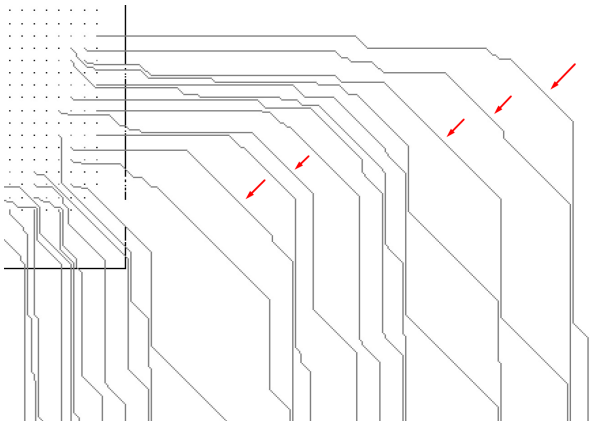
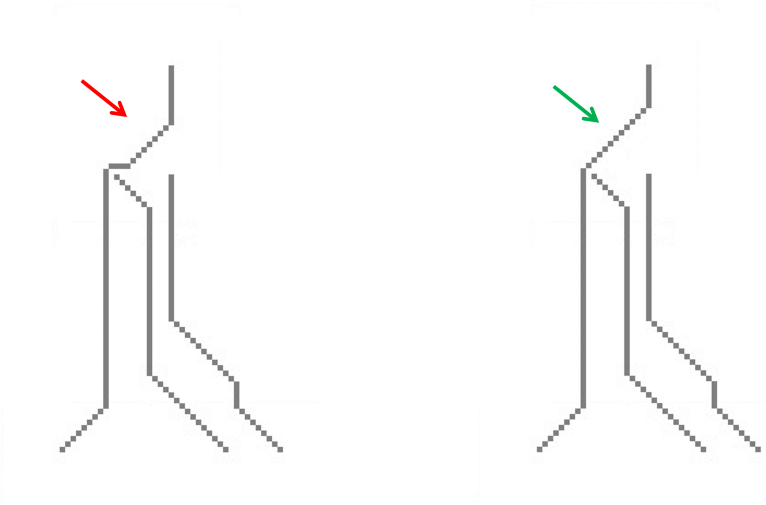


Figura 5.3: Limite de optimización debido al tamaño del vecindario.

Haciendo que la subetapa de optimización redibuje por completo el camino y perdiendo validez las subetapas anteriores.

En la Figura 5.4 se muestra una comparación entre el resultado obtenido con una combinación de vecindarios de 40 y 10 y el resultado ideal. Se evidencia que la diagonal resultante, señalada con la flecha roja, no es ideal y esto se debe a la limitante del vecindario. Por otro lado, la diagonal señalada con la flecha verde es el resultado ideal, pero para llegar a esto se debió ejecutar un vecindario específico para este caso.



(a) Optimización con un vecindario de 40 y 10 posiciones.

(b) Optimización ideal.

Figura 5.4: Caso de estudio pequeño donde se evidencia la limitante del tamaño del vecindario.

5.1.3 Orden del enrutamiento

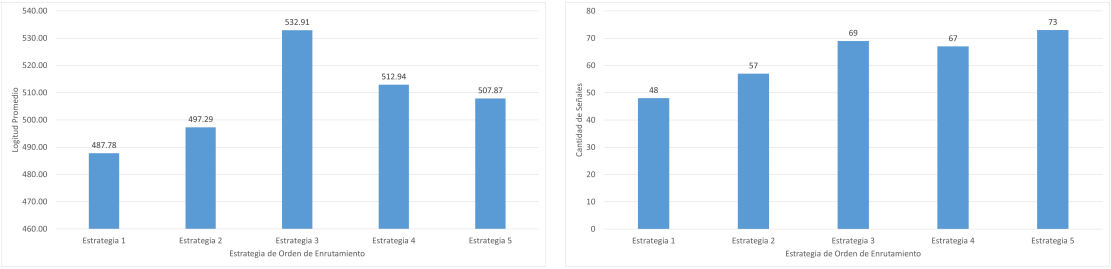
Para el orden de enrutamiento se optó por el desarrollo de cinco estrategias, los cuales tienen como propósito optimizar el uso del área, minimizar el largo de los trazos, y enrutar la mayor cantidad de señales posibles.

En la Figura 5.5(a) se puede visualizar un gráfico comparativo entre los métodos de orden de enrutamiento y la longitud promedio de las señales enrutadas. Dejando de lado la estrategia 1 para este análisis, se observa que la estrategia 2 tiene la menor longitud promedio de las señales de 497.29. Por otro lado la estrategia 3 la que arroja una mayor longitud de promedio de señales, con una longitud de 532.91.

Pero no sólo el criterio de longitud de señales es importante, sino también la cantidad de señales enrutadas. El gráfico de la Figura 5.5(b) compara los métodos de orden de enrutamiento y la cantidad de señales enrutadas.

La estrategia 2 es el que enruta menos señales, con 57. De esto se puede inferir que al enrutar una menor cantidad de señales, se obtiene un promedio de longitud más bajo que las otras estrategias.

La estrategia 5 enruta un total de 73 señales, enrutando entre 4 y 16 señales más que las otras estrategias. Se obtuvo una longitud promedio de 507.87, el cual es más bajo que las estrategias 3 y 4. Por lo tanto, la estrategia 5 es la mejor opción, al poder enrutar la mayor cantidad de señales pero manteniendo una longitud promedio baja.



(a) Longitud promedio de señales enrutadas para cada algoritmo.

(b) Cantidad de señales enrutadas por cada algoritmo.

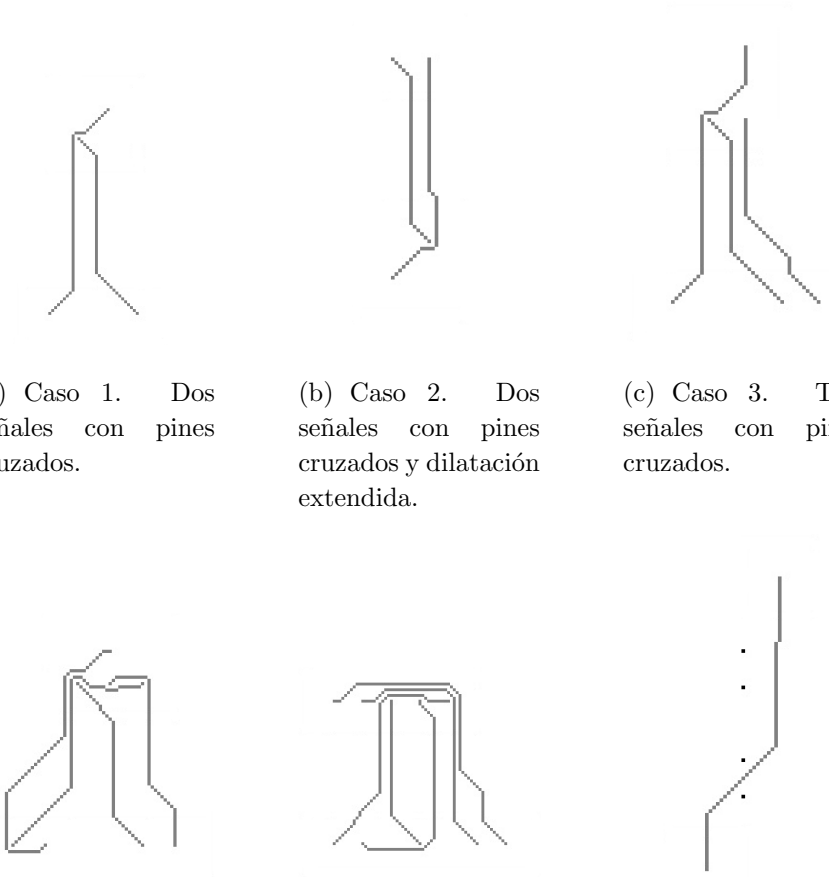
Figura 5.5: Resultados de enrutamiento en grupo para cada algoritmo.

5.2 Resultados de casos de prueba

5.2.1 Casos de estudio pequeños

En la Sección 4.1.1 se propusieron 6 casos de estudio los cuales cubren escenarios mínimos que el algoritmo debe resolver, donde en la Figura 4.1 se mostró el resultado obtenido con la propuesta conceptual del algoritmo.

En la Figura 5.6 se pueden observar los resultados de estos 6 casos de estudio pequeños pero ahora evaluados con la propuesta final del algoritmo. En dicha figura se muestra que aún se siguen cumpliendo con resultados de la prueba de concepto. Además se da una optimización en los trazos dando como resultado un enrutamiento más directo lo cual se traduce en trazos de menor longitud con ángulos de 45 grados.



(a) Caso 1. Dos señales con pines cruzados.

(b) Caso 2. Dos señales con pines cruzados y dilatación extendida.

(c) Caso 3. Tres señales con pines cruzados.

(d) Caso 4. Tres señales con pines cruzados y dilatación extendida.

(e) Caso 5. Cinco señales con pines cruzados y dilatación extendida.

(f) Caso 6. Una señal con obstáculos.

Figura 5.6: Casos de estudio pequeños

5.2.2 Caso de estudio mediano

Para el caso del estudio mediano, se propuso el enrutamiento de 20 señales usando la huella de un SoC de alto rendimiento para los pines de inicio y una topología simplificada para los pines de llegada para centrarse en la optimización del escape.

En la Figura 5.7 se muestra que el algoritmo es capaz de enrutar un total de 14 de 20 señales, donde se evidencia que el algoritmo cumple con los requerimientos mínimos y realiza trazos efectivos dentro de sus limitaciones, como se planteó en la Sección 2.3.

Si se realiza un análisis detallado en la sección del enrutamiento de escape, se puede observar que el algoritmo es capaz de descruzar las señales para poder encontrar un camino viable a los pines de llegada. Esto a pesar de que se tiene una región con una densidad alta de pines. A la vez, el algoritmo está respetando el espaciamiento mínimo y no se traslapa con otro trazo o pin, obteniendo un trazo con reducción de curvas, mejor utilización del espacio y menor longitud.

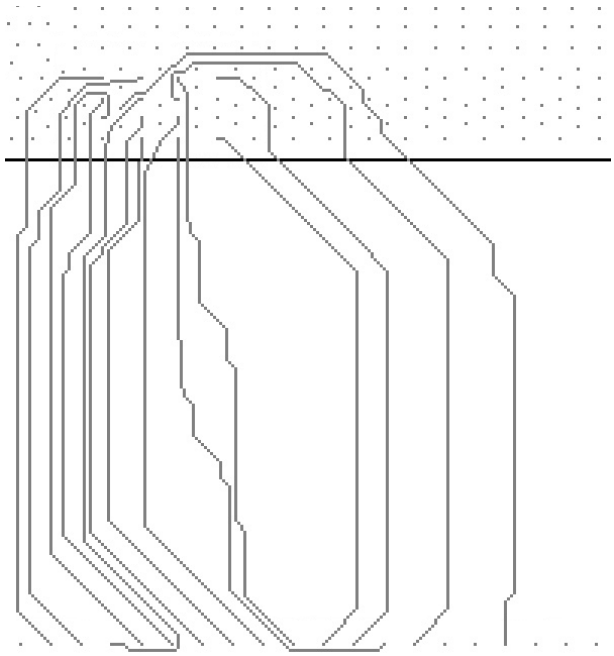


Figura 5.7: Caso de estudio mediano, enrutamiento de 20 pines en una huella de un IC de alto rendimiento.

5.2.3 Caso de estudio grande

Para el caso de estudio grande, la propuesta es enrutar 124 señales usando la huella de un IC de alto rendimiento para los pines de salida y para los pines de llegada se utiliza una huella de un conector de memoria DDR4 SDRAM RDIMM. Así se tiene un caso de estudio muy complejo debido a la cantidad de señales, ubicaciones de los pines y orientación del enrutamiento.

En la Figura 5.8 se observa el enrutamiento obtenido, donde se logran enrutar 73 señales de las 124 propuestas. Se evidencia que el algoritmo es capaz de cumplir con los requerimientos mínimos y realiza trazos efectivos dentro de sus limitaciones tanto la región de enrutamiento de escape y general, como se plantea en la Sección 2.3. Este enrutamiento consume un área de 6600 mm^2.

El IC tiene señales a escapar desde 3 diferentes lados: sur, este y oeste. Las señales que escapan del lado sur siguen un trazado óptimo en su mayoría muy lineal, en algunos casos realizando un descruce de señales en el área del IC o del conector de memoria.

Para las señales que se escapan desde los lados este y oeste, siendo estos casos de enrutamiento de escape más complejos, se observa que primero se da un escape buscado los lados para después tomar un direccionamiento hacia el sur buscando el pin de llegada. Este comportamiento es efectivo y crítico, ya que si el enrutamiento fuera directo hacia el sur, bloquearía muchas señales reduciendo drásticamente el número de señales enrutadas.

Para el enrutamiento general, se observa que las señales que escapan de los lados tienden a crear una diagonal. Esto es efectivo, ya que así se disminuye el área de enrutamiento y la longitud de los trazos.

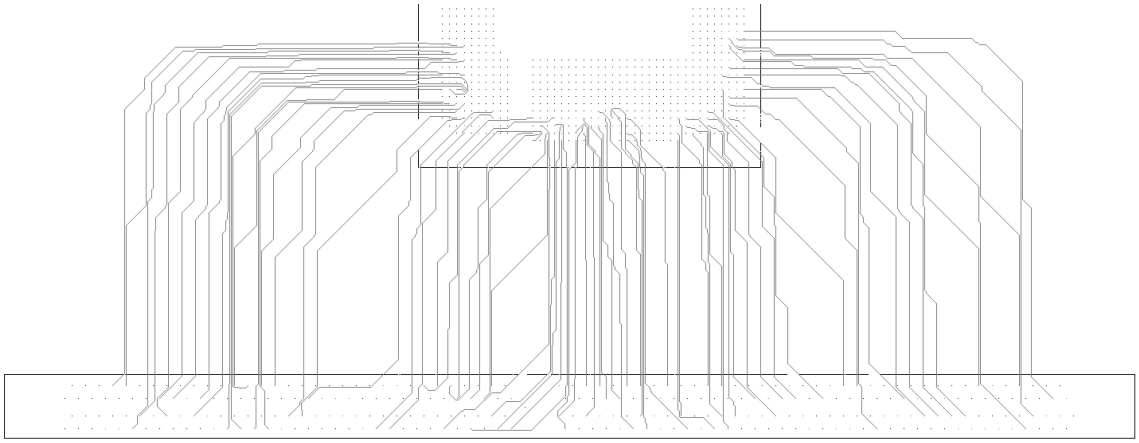


Figura 5.8: Resultado de enrutamiento entre un IC de alta gama y un conector de memoria utilizando el algoritmo propuesto.

5.3 Comparación con herramientas comerciales

En esta sección se compara los resultados obtenidos entre el software Allegro de Cadence, Figura 5.9 y el algoritmo propuesto, Figura 5.8, para un enrutamiento que va de un IC de alta gama y un conector de memoria.

El primer hallazgo al confrontar los algoritmos es que los resultados obtenidos con la herramienta Allegro tienen una mayor cantidad de señales enrutadas con 124 señales y un área de enrutamiento de 5350 mm^2. Por otra parte el algoritmo propuesto solo enrutó 73 señales y utilizó un área de enrutamiento de 6600 mm^2.

En el enrutamiento general se puede ver que Allegro realiza trazos en diagonal más amplios con un espaciamiento mínimo entre señales, lo cual es más eficiente porque reduce el área de enrutamiento y el largo total de la señal. Por otro lado, el algoritmo propuesto realiza una estrategia similar con la aplicación de las diagonales, pero dejando un mayor espaciamiento entre señales, lo cual no es eficiente al consumir más área.

A pesar de que Allegro escapa todas las señales, algunos trazos en el enrutamiento de escape que no siguen un flujo correcto y se enrutan alrededor de los pines internos crean-

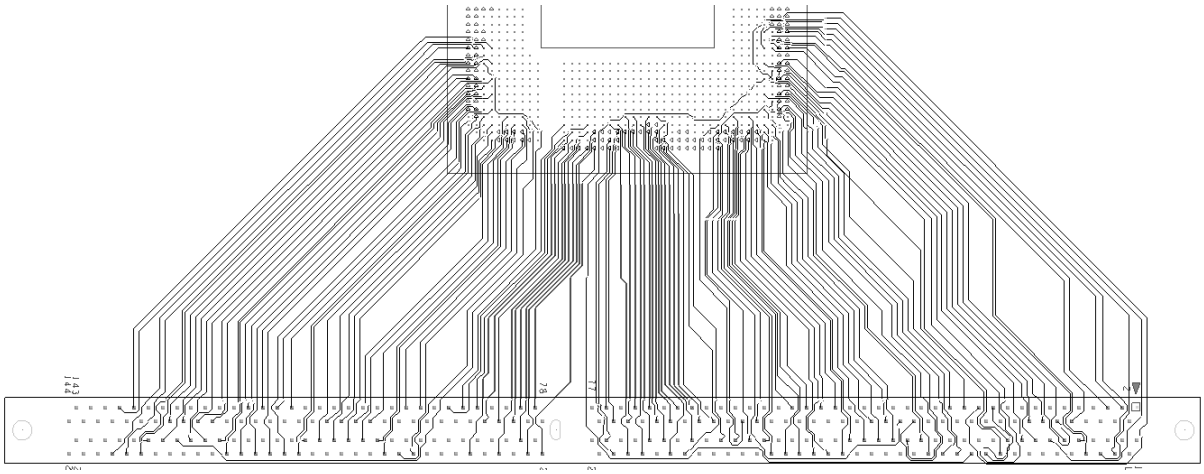
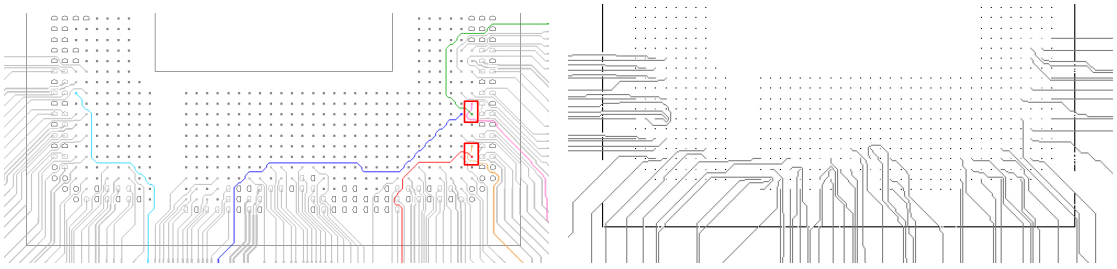


Figura 5.9: Resultado de enrutamiento entre un IC de alta gama y un conector de memoria utilizando el software Allegro de Cadence.

do una solución no viable pues bloquea el enrutamiento de los pines internos, observe los trazos resaltados en color celeste, azul y verde en la Figura 5.10(a). Al realizar esta comparación con la solución propuesta, los trazos siempre buscan el borde más cercano del paquete realizando un mejor enrutamiento de escape, como se muestra en la Figura 5.10(b) no hay señales que vayan por medio de los pines internos. También es importante resaltar que Allegro realiza enrutamientos erróneos donde se dan casos como la superposición de dos señales, creando cortos. Esto se resalta en los cuadros rojos de la Figura 5.10(a).



(a) Enrutamiento de escape con Allegro.

(b) Enrutamiento de escape con el algoritmo propuesto.

Figura 5.10: Comparación del enrutamiento de escape.

Capítulo 6

Conclusiones

El objetivo fundamental de esta investigación es abordar el problema del enrutamiento en un PCB, aportando una nueva solución para automatizar el proceso de enrutamiento que dé como resultado el enrutamiento de un grupo de señales entre dos componentes electrónicos de forma valida y razonable.

La aportación principal de este trabajo consiste en el diseño e implementación de un algoritmo para automatizar el enrutamiento de señales en una placa de circuitos impresos basado en operaciones morfológicas de dilatación y adelgazamiento. Se han escogido estos operadores matemáticos, ya que como puso en manifiesto en esta tesis, son herramientas que ofrecen soluciones para la obtención de caminos que son candidatos para ser utilizados para el enrutamiento. Los algoritmos utilizados, además de no ser explorados por otros autores consultados permiten garantizar la obtención de enrutamientos válidos.

Por medio de la operación matemática morfológica de dilatación se demuestra la viabilidad de un camino entre el pin de llegada y de salida. La dilatación da como resultado si existe al menos un camino mínimo entre el pin de llegada y de salida, así garantizando el cumplimiento de requerimiento mínimo que indica que un trazo debe ir entre un pin A y un pin B.

De existir más de un camino posible, la dilatación da todas esas posiciones disponibles cumpliendo con los requerimientos mínimos de espaciamiento entre obstáculos, pines y trazos. Por lo tanto, desde la subetapa de dilatación se puede concluir que se está cumpliendo con los requerimientos mínimos planteados que debe seguir el algoritmo para asegurar una interconexión física optima de un trazo en un PCB.

Así mismo, la subetapa de adelgazamiento toma todos los posibles caminos generados en la dilatación y los convierte en caminos de un ancho mínimo de una posición. Esto por medio de máscaras que cumplen con el requisito de no desconectar posibles caminos válidos y mantener los requisitos mínimos. Dando como resultado múltiples caminos que son candidatos para seleccionar como el enrutamiento final.

Se diseñó un algoritmo el cual es capaz de tomar como base los caminos mínimos de la

etapa de adelgazamiento y realizar la construcción de grafo con pesos. Así, habilitando el poder utilizar el algoritmo de Dijkstra para poder determinar el camino más corto entre el pin de salida y de llegada.

Utilizando vecindarios de 40 y 10 posiciones, se comprobó la disminución en la cantidad de curvas, longitud de señal y de la utilización del espacio.

La última etapa cubre el orden de enrutamiento donde se demuestra que la estrategia 5 es el que da mejores resultados, esto debido a que fue la estrategia que enrutó la mayor cantidad de señales, con una longitud de 507.87, así dando la menor área de enrutamiento por cantidad de señales enrutadas.

Finalmente, al realizar una comparación contra la herramienta Allegro, se muestra que la solución propuesta crea trazos con un flujo correcto y enrutamientos sin superposición de señales, donde Allegro crea un diseño erróneo para estos casos.

Todos los algoritmos planteados se han diseñado en Python 3.9, han sido probados con los tres casos de estudios planteados los cuales dan robustes al algoritmo verificando desde casos mínimos y simples, hasta casos reales con complejidades altas.

Capítulo 7

Trabajos a futuro

Finalmente quedaría plantear cuáles pueden ser las siguientes pautas de la investigación. Como continuación natural del trabajo desarrollado en esta tesis, un área futura inmediata podría ser la optimización de la subetapa de adelgazamiento. Otra área para desarrollar que resultaría interesante ampliar es el orden del enrutamiento, ya que se trabajó con cinco estrategias de ordenamiento, pero no son los únicos que se pueden desarrollar. También se puede agregar a esta solución otras áreas que ya han sido desarrolladas por otros autores y no fueron abordadas en esta solución, para así obtener un algoritmo más robusto.

7.1 Subetapa de Adelgazamiento

Como se mencionó en secciones anteriores, el adelgazamiento es una de las subetapas más importantes, ya que da como resultado los posibles caminos que van a ser la base para el trazo final. Dentro del trabajo realizado se identificó que es posible realizar una optimización para esta etapa. La optimización consiste en dar un sesgo hacia los trazos de los vecinos. Con esto se garantizaría obtener una optimización en la utilización del área de enrutamiento y generar la hipótesis que se pueden escapar más señales. Un ejemplo de esto se observa en la Figura 7.1, donde el trazo azul sustituiría al segmento del trazo con sesgo hacia el centro.

Esta es una propuesta que no se abordó para esta investigación por motivos de tiempo, pero queda para trabajos futuros.

7.2 Orden de enrutamiento

Una estrategia extra para el orden de enrutamiento que se identificó durante el desarrollo de este trabajo y que se escapó del alcance, fue el de iniciar o darles prioridad a las señales que estas más cercanas al borde de los ICs a enrutar. Así se garantizaría tener trazos más limpios al inicio y dejar espacio para que puedan escapar los trazos que están

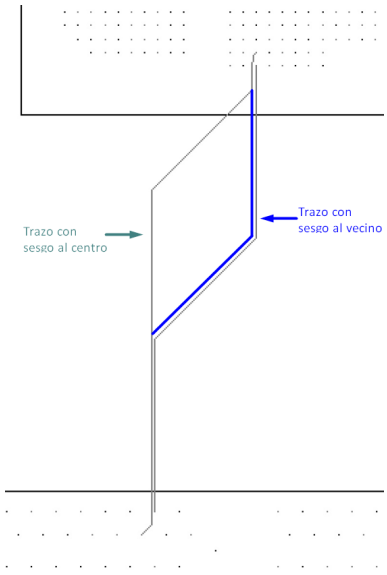


Figura 7.1: Camino con sesgo al trazo vecino (trazo azul).

en la parte interna de los ICs. Esta es una propuesta de un método para el orden de enrutamiento de las señales que incluso se puede combinar con alguna de las desarrollas para esta investigación.

7.3 Otras categorías del enrutamiento automático

Además del enrutamiento de escape y general se pueden abordar otras categorías en el área del enrutamiento automático. Estas podrían ser:

- Asignación de capas.
- Enrutamiento con vías.
- Otros ángulos de enrutamiento.
- Enrutamiento de señales diferenciales.
- Coincidencia de longitudes.
- Manejo de geometrías de trazas y espaciados.

Tener diseños de PCB multicapa es muy común en los sistemas que utilizan ICs de alta gama, donde las señales se pueden distribuir entre las capas del PCB. El enrutamiento multicapa se puede presentar en tanto en el área de escape como en la general y para lograr esto es necesario hacer uso de las vías.

El enrutamiento con vías brinda la posibilidad de realizar transiciones de los trazos entre capas, donde se debe tener en cuenta la estructura de la vía, los espaciamientos entre los diferentes componentes de un PCB y también las capas de disponibles.

En la investigación se utilizó ángulos de 45 grados para los trazados, ya que es el estándar en la industria. Pero también es posible utilizar otros ángulos para el enrutamiento.

Explorar una solución con diversos ángulos podría permitir tener mayor flexibilidad en el diseño y posibles optimizaciones.

Las interfaces de alta velocidad utilizan señales diferenciales para la transmisión de los datos, por lo tanto es un caso que se debe abordar en el enrutamiento automático. Esto debido a que el enrutamiento de una señal diferencial tiene diferentes restricciones en comparación a una señal de terminación individual.

Los grupos de señales, como por ejemplo un canal de memoria, tienen restricciones en la longitud ya que las señales deben tener una sincronización. Garantizar que las longitudes de los trazos están dentro del margen adecuado para realizar una sincronización es muy importante, esto hace que la interfaz tenga un correcto rendimiento.

Las geometrías de los trazos pueden tener restricciones de ancho o espaciamiento según el área de enrutamiento. Tomar en cuenta estos cambios en las geometrías crearan un algoritmo más robusto.

Bibliografía

[1] Auto connect - feature video. URL https://resources.pcb.cadence.com/vidyard-all-players/auto-connect-feature-video.

[2] AutoRoute | altium designer 21 user manual | documentation. URL https://www.altium.com/documentation/altium-designer/pcb-cmd-autorouteautoroute-ad.

[3] Routing & autorouting - PCB layout basics 2 | EAGLE | blog. URL https://www.autodesk.com/products/eagle/blog/routing-autorouting-pcb-layout-basics-2/. Section: EAGLE Academy.

[4] Asad Ali, Anjum Naveed, and Muhammad Zeeshan. A dual model node based optimization algorithm for simultaneous escape routing in PCBs. 7:e499. URL https://doi.org/10.7717/peerj-cs.499.

[5] Asad Ali, Muhammad Zeeshan, and Anjum Naveed. A network flow approach for simultaneous escape routing in PCB. In *2017 14th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT (HONET-ICT)*, pages 78–82. ISSN: 1949-4106.

[6] Said Broumi, Assia Bakal, Mohamed Talea, Florentin Smarandache, and Luige Vladareanu. Applying dijkstra algorithm for solving neutrosophic shortest path problem. In *2016 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 412–416. URL https://ieeexplore.ieee.org/abstract/document/7813483. ISSN: 2325-0690.

[7] Jingsong Chen, Jinwei Liu, Gengjie Chen, Dan Zheng, and Evangeline F. Y. Young. MARCH: MAze routing under a concurrent and hierarchical scheme for buses. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 1–6. Association for Computing Machinery. URL https://doi.org/10.1145/3316781.3317860.

[8] Chin-Chih Chang and J. J. S. Cong. An efficient approach to multilayer layer assignment with an application to via minimization. 18(5):608–620.

[9] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart. Morphology - dilation. URL https://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm.

[10] Robert Fisher, Simon Perkins, Ashley Walker, and Erik Wolfart. Morphology - thinning. URL https://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm.

[11] T. Hama and H. Etoh. Topological routing path search algorithm with incremental routability test. 18(2):142–150.

[12] H. Heijmans, M. Buckley, and H. Talbot. Path-based morphological openings. In *2004 International Conference on Image Processing, 2004. ICIP '04.*, volume 5, pages 3085–3088 Vol. 5. URL https://ieeexplore.ieee.org/abstract/document/1421765. ISSN: 1522-4880.

[13] K Hong. Python tutorial: Dijkstra's shortest path algorithm - 2020. URL https://www.bogotobogo.com/python/python_Dijkstras_Shortest_Path_Algorithm.php.

[14] Tsung-Wei Huang, Pei-Ci Wu, and Martin D. Wong. UI-route: An ultra-fast incremental maze routing algorithm. In *2014 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pages 1–8. ISSN: null.

[15] Adeel Javaid. Understanding dijkstra algorithm. URL https://www.researchgate.net/publication/273264449_Understanding_Dijkstra_Algorithm.

[16] Yukihide Kohira and Atsushi Takahashi. An any-angle routing method using quasi-newton method. In *17th Asia and South Pacific Design Automation Conference*, pages 145–150. ISSN: 2153-6961.

[17] Yukihide Kohira and Atsushi Takahashi. CAFE router: A fast connectivity aware multiple nets routing algorithm for routing grid with obstacles. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 281–286. ISSN: 2153-697X.

[18] Hui Kong, Qiang Ma, Tan Yan, and Martin D.F. Wong. An optimal algorithm for finding disjoint rectangles and its application to PCB routing. In *Design Automation Conference*, pages 212–217. ISSN: 0738-100X.

[19] Hui Kong, Tan Yan, and Martin D.F. Wong. Automatic bus planner for dense PCBs. In *2009 46th ACM/IEEE Design Automation Conference*, pages 326–331. ISSN: 0738-100X.

[20] Seong-I Lei and Wai-Kei Mak. Optimizing pin assignment and escape routing for blind-via-based PCBs. 35(2):246–259. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[21] Étienne Lepercq, Yves Blaquière, and Yvon Savaria. A pattern-based routing algorithm for a novel electronic system prototyping platform. 62:224–237. URL http://www.sciencedirect.com/science/article/pii/S016792601830049X.

[22] Lijuan Luo, Tan Yan, Qiang Ma, Martin D. F. Wong, and Toshiyuki Shibuya. A new strategy for simultaneous escape based on boundary routing. 30(2):205–214. URL https://ieeexplore.ieee.org/abstract/document/5689348. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[23] Qiang Ma, Hui Kong, Martin D. F. Wong, and Evangeline F. Y. Young. A provably good approximation algorithm for rectangle escape problem with application to PCB routing. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pages 843–848. IEEE. URL http://ieeexplore.ieee.org/document/5722308/.

[24] Qiang Ma, Evangeline F. Y. Young, and Martin D. F. Wong. An optimal algorithm for layer assignment of bus escape routing on PCBs. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 176–181. ISSN: 0738-100x.

[25] Man-Fai Yu and W. Wei-Ming Dai. Single-layer fanout routing and routability analysis for ball grid arrays. In *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*, pages 581–586.

[26] Jeffrey McDaniel, Zachary Zimmerman, Daniel Grissom, and Philip Brisk. PCB escape routing and layer minimization for digital microfluidic biochips. 36(1):69–82. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[27] Yosbel Morales Olivera, Josué García Parrado, Pablo E. Reyes Fernández, and Juan V. Lorenzo Ginori. Experiencias en la implementación de las operaciones morfológicas de erosión y dilatación para imágenes binarias empleando vecindades adaptativas. 33(2):34–41. URL http://scielo.sld.cu/scielo.php?script=sci_abstract&pid=S1815-59282012000200005&lng=es&nrm=iso&tlng=es. Publisher: Facultad de Eléctrica, Instituto Superior Politécnico José Antonio Echeverría, Cujae.

[28] M.M. Ozdal and M.D.F. Wong. A provably good algorithm for high performance bus routing. In *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, pages 830–837. ISSN: 1092-3152.

[29] PTC Mathcad Prime 5.0.0.0. Example: Thinning and skeletonization. URL http://support.ptc.com/help/mathcad/en/index.html#page/PTC_Mathcad_Help/example_thinning_and_skeletonization.html.

[30] Kashif Sattar and Aleksandar Ignjatovic. Simultaneous escape routing using network flow optimization. 29(2):86–105. URL http://mjs.um.edu.my/index.php/MJCS/article/view/7006. Number: 2.

[31] Utkarsh Sinha. Mathematical morphology: Basic operations - AI shack. URL https://aishack.in/tutorials/mathematical-morphology/.

[32] M. Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. pages 599–620. URL https://www.infona.pl//resource/bwmeta1.element.baztech-article-BAT5-0013-0005.

[33] Simon Ström and Ali Qhorbani. *Automation of the design process of printed circuit boards : Determining minimum distance required by auto-routing software*. URL http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-251925.

[34] WikiChip LLC. Ice lake y - cores - intel - WikiChip. URL https://en.wikichip.org/wiki/intel/cores/ice_lake_y.

[35] Pei-Ci Wu, Qiang Ma, and Martin D. F. Wong. An ILP-based automatic bus planner for dense PCBs. In *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 181–186. ISSN: 2153-6961.

[36] Tan Yan and Martin D. F. Wong. Correctly modeling the diagonal capacity in escape routing. 31(2):285–293. URL https://ieeexplore.ieee.org/document/6132657. Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[37] Tan Yan and Martin D. F. Wong. Recent research development in PCB layout. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 398–403. ISSN: 1092-3152.

[38] Cong Zhang, Huilin Jin, Jienan Chen, Jinkuan Zhu, and Jinting Luo. A hierarchy MCTS algorithm for the automated PCB routing. In *2020 IEEE 16th International Conference on Control Automation (ICCA)*, pages 1366–1371. URL https://ieeexplore.ieee.org/document/9264558. ISSN: 1948-3457.

[39] Ran Zhang, Tieyuan Pan, Li Zhu, and Takahiro Watanabe. A length matching routing method for disordered pins in PCB design. In *The 20th Asia and South Pacific Design Automation Conference*, pages 402–407. ISSN: 2153-697X.

[40] Yu-Jin Zhang. Mathematical morphology for binary images. In Yu-Jin Zhang, editor, *Handbook of Image Engineering*, pages 1723–1741. Springer. URL https://doi.org/10.1007/978-981-15-5873-3_48.

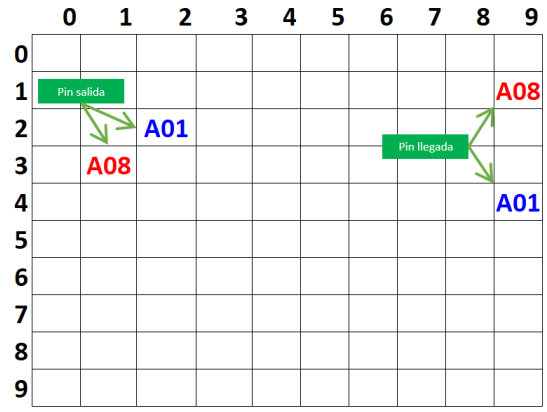
Apéndice A

Algoritmos realizados para el enrutamiento individual de una señal

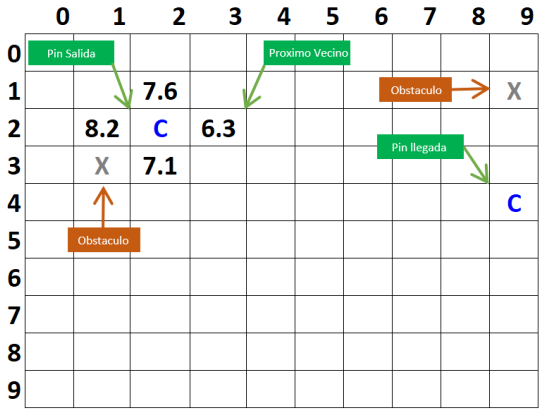
Para llegar a tener una solución viable para el enrutamiento de una señal se desarrollaron 3 algoritmos, los cuales fueron evaluados en cada etapa del desarrollo y dieron como resultado un algoritmo razonable basado en operadores morfológicos y el algoritmo de Dijkstra, el cual se detalló en el Sección 4.3. Por lo tanto para dar contexto al algoritmo final debemos entender que propuestas se realizaron y como estas nos fueron llevando a tener una solución con un enfoque muy diferente al que proponen.

A.1 Algoritmo 1

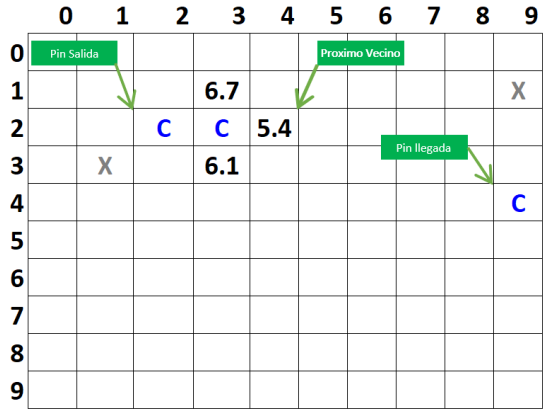
En la primera iteración se buscó realizar un algoritmo voraz el cual diera el camino más corto entre los dos puntos de conexión para un grupo de señales. Para esto se realizó un caso de estudio simplificado donde se enruta dos señales en una cuadricula de 10x10. El algoritmo inicia por el pin de salida buscando moverse a su vecino que este más cercano al pin de llegada, para esto se realizó el cálculo de la distancia euclidiana de cada uno de sus vecinos (norte, sur, este y oeste) y se mueve el vecino con el resultado menor. Estos pasos se vuelven a realizar por cada vecino seleccionado hasta que se llega al pin de llegada. La Figura A.1 resume los pasos descritos.



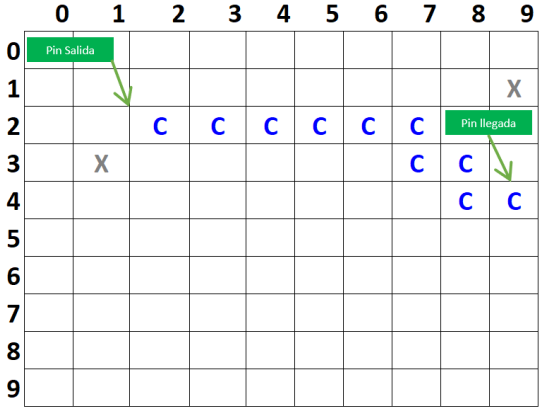
(a) Identificación de los pines de llegada y salida.



(b) Calculo de la distancia euclidiana para los vecinos del pin de salida A01.



(c) Selección del vecino más cercano y cálculo de la distancia euclidiana para la selección de la próxima posición.

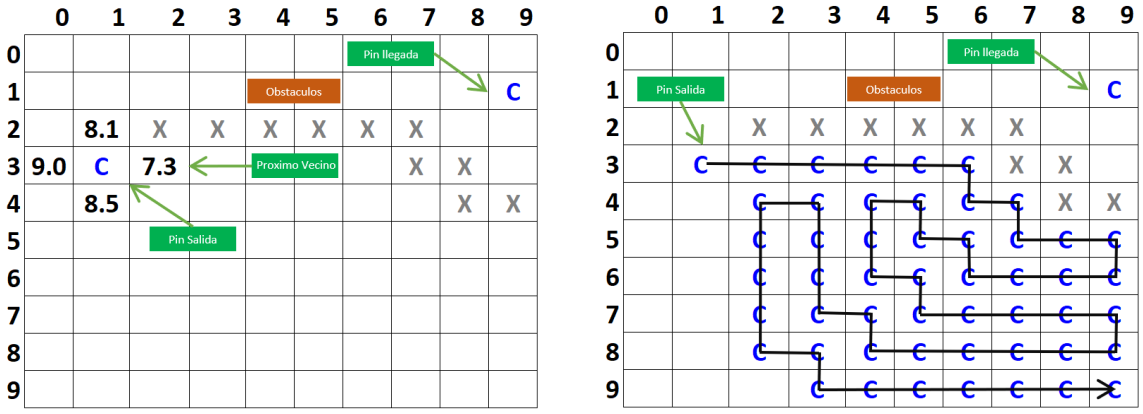


(d) Enrutamiento completo para el pin A01.

Figura A.1: Algoritmo voraz enrutando el pin A01.

Cuando se llegó a verificar el algoritmo voraz con otra señal, en el caso mostrado por la Figura A.2, se puede observar que el inicio del enrutamiento para el pin A08 busca los vecinos más cercanos matemáticamente al pin de llegada, seleccionado el vecino que esta al este, pero también se evidencia que la selección de este vecino no es la correcta, ya que el trazo del pin A01 bloquearía esta selección de vecino, dando como resultado un sin salida para el pin A08.

El algoritmo se limita únicamente a la selección del vecino más cercano al pin de llegada y pierde las otras posibilidades de enrutamiento, donde para este caso la selección correcta debería ser el vecino norte.



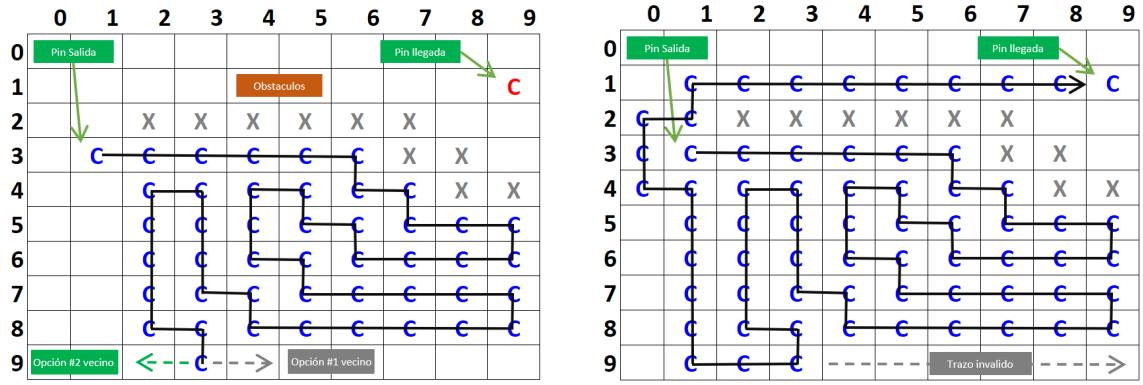
(a) Calculo de la distancia euclidiana para los vecinos del pin de salida A08.

(b) Enrutamiento completo para el pin A08.

Figura A.2: Algoritmo voraz enrutando el pin A08.

A.2 Algoritmo 2

Para la segunda interacción se continua con casos de estudio simplificados enrutando dos señales en una cuadricula de 10x10. Debido a las limitaciones dadas en la iteración uno, se implementó un algoritmo similar, el cual trata de buscar el camino más corto posible por medio de la distancia euclidiana, pero se modificó para que fuera un algoritmo recursivo, así cuando el algoritmo no encuentre el camino correcto pudiera devolverse a su vecino anterior y poder seleccionar la siguiente posibilidad de vecino con la distancia más corta. Esto se ejemplifica en la Figura A.3(a), donde el algoritmo al seleccionar un camino invalido y al llegar al final de este, el algoritmo retrocede hasta donde haya otra opción de vecino y selecciona esta.



(a) Selección se un vecino alternativo.

(b) Enrutamiento completo para el pin A08 con la selección de un vecino alternativo.

Figura A.3: Algoritmo voraz recursivo enrutando el pin A08.

En la etapa de verificación al enrutar el pin A08 se observa que tuvo que retroceder 6 veces para encontrar un trazo valido al pin de llegada. Este trazo final no es razonable, ya que hace muchos giros en el área disponible para encontrar el trazo valido. Al trasladar este caso a una cuadricula más grande de 100x100 se observa un comportamiento similar donde el trazo utilizando el área disponible y como resultado se tiene un incremento exponencial en el tiempo de procesamiento, incluso en otros casos de prueba obteniendo un desborde de la memoria.

A.3 Algoritmo 3

En la tercera iteración se desarrolló un algoritmo el cual cuenta con 5 etapas y es el que se propone como solución final. En la Sección 4.3.1 se desarrolla en detalle esta solución.