Template-based Routing Generator

José de Jesús de la Rosa de la Rosa*
Facultad de Ciencias, Universidad Autónoma de San Luis Potosí.
(Dated: 22 de junio de 2022)

I. INTRODUCCIÓN

Durante años se han propuesto en la literatura técnicas de enrutamiento para la automatización del diseño de circuitos integrados (IC) digitales y analógicos. En ellas, ya se ha cubierto un amplio conjunto de restricciones geométricas para la la mejora de calidad del enrutamiento, pero también se han incluido progresivamente criterios relacionados con el rendimiento. Sin embargo, a medida que el diseño de circuitos mas complejos y de características particulares, como en los circuitos analógicos y de radiofrecuencia, avanzan hacia nuevos nodos tecnológicos, el creciente número de reglas y restricciones de diseño, la resistencia de cables, la congestión y el crecimiento de circuitos parasíticos entre cables, impulsan constantemente las técnicas de enrutamiento automático existentes y mantienen la presión sobre su mejora.

En la literatura se han propuesto técnicas de enrutamiento para la automatización del diseño de circuitos integrados analógicos y de radiofrecuencia (A/RF) [1, 2]. En ellas, ya se ha cubierto un amplio conjunto de restricciones geométricas como sustitutos de la calidad del enrutamiento, pero también se han incluido progresivamente criterios relacionados con el rendimiento. Sin embargo, a medida que el diseño de circuitos A/RF avanza hacia nodos tecnológicos de integración avanzada, el creciente número de reglas ó restricciones de diseño, la resistencia de los cables, la congestión y el crecimiento parasitario entre cables desafían constantemente las técnicas de enrutamiento automático existentes y mantienen la presión sobre su mejora. Afortunadamente, los recientes avances en las capacidades de las estaciones de trabajo modernas permitieron el crecimiento de sofisticados procesos de enrutamiento, incluidos algunos asistidos por los últimos métodos de aprendizaje automático y profundo, que ofrecen soluciones sin precedentes para la automatización de esta tarea. Sin embargo, la correlación entre las estructuras parasíticas inducidas por el enrutamiento y el comportamiento funcional del circuito dista mucho de ser sencilla, también se han propuesto técnicas de síntesis computacionalmente intensivas con inclusión de parásitos y conscientes del diseño, en las que las técnicas de enrutamiento automático desempeñan un papel decisivo.

* a228835@alumnos.uaslp.mx

II. PATH-FINDING ALGORITHM

Una de las principales técnicas es el uso de generadores de enrutamiento basados en plantillas (TbRG, *Template-based routing generators*). La plantilla actúa como una representación del diseño gráfico de la tecnología, generada por el conjunto y características de los dispositivos. Son especialmente útiles cuando se debe migrar un diseño validado previamente diseñado, es decir, un diseño heredado, a otros nodos de tecnología cercanos o se necesitan cambios en el diseño. Esta plantilla puede ser implementada como un grafo de pesos unitarios, conexo y no dirigido, donde se seleccionan un vértice origin y destino, y a través de diferentes algoritmos, encontrar el camino mas corto.

El algoritmo de Dijkstra, también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista. La idea general en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices, cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene. Se trata de una especialización de la búsqueda de costo uniforme y, como tal, no funciona en grafos con aristas de coste negativo (al elegir siempre el nodo con distancia menor, pueden quedar excluidos de la búsqueda nodos que en próximas iteraciones bajarían el costo general del camino al pasar por una arista con costo negativo).

Existe una variante del algoritmo de Dijkstra, llamado algoritmo A*, el cual tiene como objetivo únicamente el camino más corto desde una fuente específica hasta un objetivo específico, y no el árbol de caminos más corto desde una fuente específica hasta todos los objetivos posibles. Para esto es es necesario el uso de una función heucarística. Para el algoritmo de Dijkstra, dado que se genera todo el árbol de la ruta más corta, cada nodo es un objetivo y no puede haber una heurística dirigida a un objetivo específico, obteniendo peor rendimiento que el algoritmo A* cuando la búsqueda del camino mas corto es solo entre dos nodos.

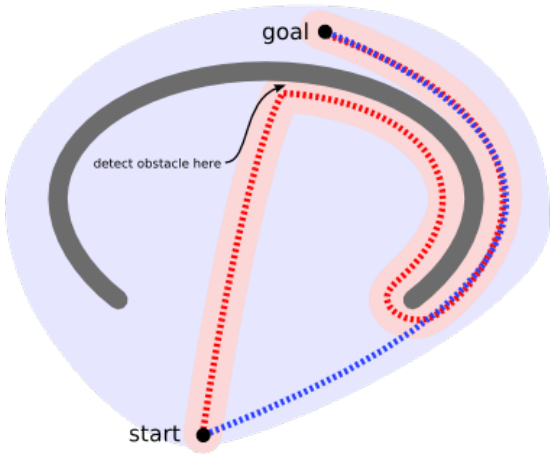

Figura 1. Algoritmo pathfinding.

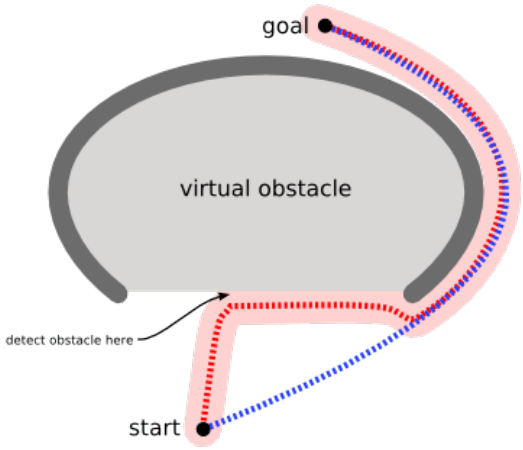

Figura 2. Algoritmo pathfinding.

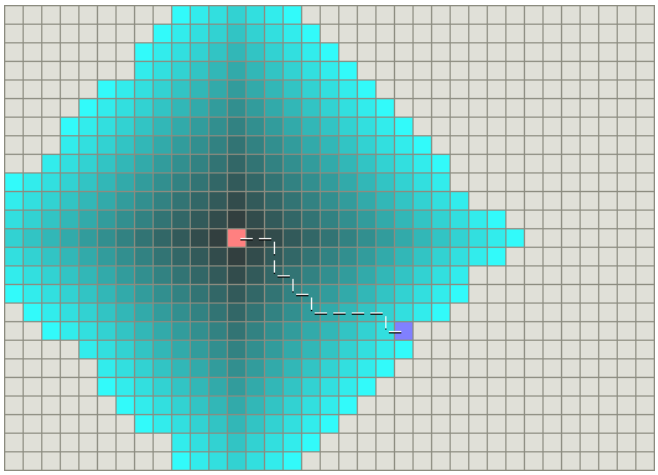

Figura 3. Algoritmo pathfinding.

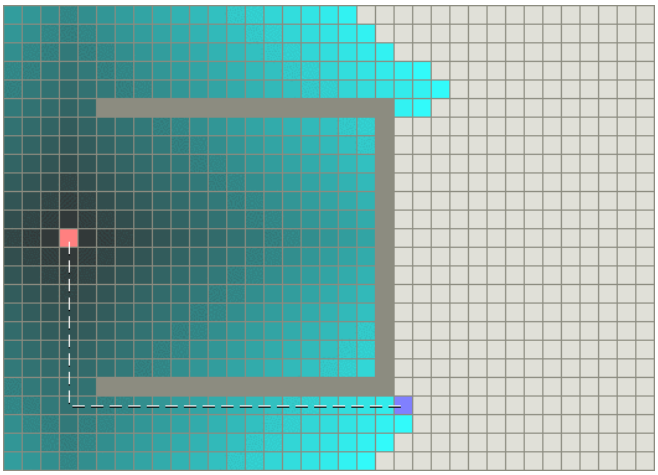

Figura 4. Algoritmo pathfinding.

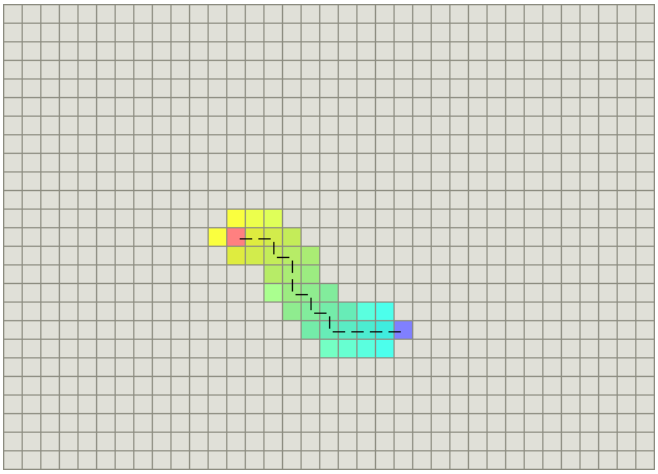

Figura 5. Algoritmo pathfinding.

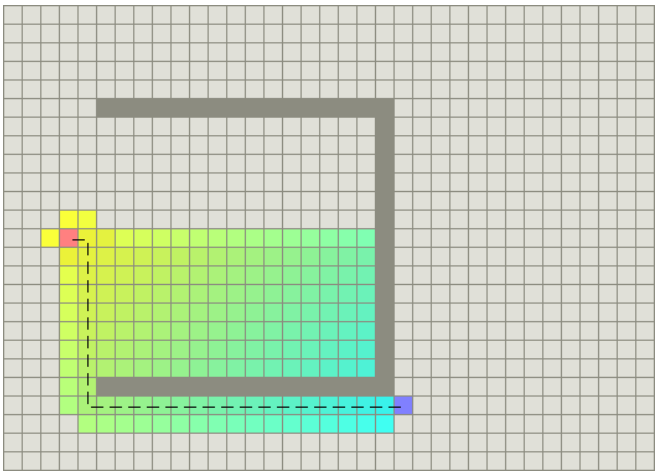

Figura 6. Algoritmo pathfinding.

III. IMPLEMENTACIÓN

IV. TRABAJO A FUTURO

V. CONCLUSIONES

VI. CONCLUSIONES

A pesar de las diferentes arquitecturas de memorias basadas en RRAM reportadas en la literatura, no ha sido reportada una celda que logre una implementación funcional debido los problemas derivados de la variabilidad de los dispositivos.

---

[1] Ahmet Unutulmaz, Günhan Dündar, and Francisco V Fernández. A template router. In *2011 20th European Conference on Circuit Theory and Design (ECCTD)*, pages 334–337. IEEE, 2011.

[2] Ricardo MF Martins and Nuno CC Lourenço. Analog integrated circuit routing techniques: An extensive review. *IEEE Access*, 2023.

[3] M. Hanan. On steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*, 14(2):255–265, 1966. doi:10.1137/0114025.

[4] Mark Allen Weiss. Data structures and algorithm analysis in c++. 2014.