Exploring A* Search For Single and Multi Layer Routing

Mohamed Shawky
Computer Engineering
Cairo University
Email: mohamed.sabae99@eng-st.cu.edu.eg

Mahmoud Adas
Computer Engineering
Cairo University
Email: mahmoud.ibrahim97@eng-st.cu.edu.eg

Remonda Talaat
Computer Engineering
Cairo University
Email: Remonda.Bastawres99@eng-st.cu.edu.eg

Evram Youssef
Computer Engineering
Cairo University
Email: evram.narouz00@eng-st.cu.edu.eg

***Abstract*—**

***Index Terms*—**

I. Introduction

II. Terminology

III. Related Work

A. Basic History

In 1959, Moore, Edward F. presented one of the first shortest path through a maze algorithm [1], after a couple of years in 1961 Lee, C. Y. presented the idea of simulating the board wiring on electronics board as a Maze [2]. Starting from there the idea of Lee Maze has been revisited many times, in 1983 Hightower, D. made more contribution to the idea such that using modern computers and virtual memories we can memic the routing problem precisely providing different techniques [3].

B. Router Anatomy

The Routing problem has many sections and subsections, in this paper we are mainly concerned with Detailed routing. Detailed routing is divided into many subsections, as you can see in fig.1 and we are exploring Maze and Line Search subsections.

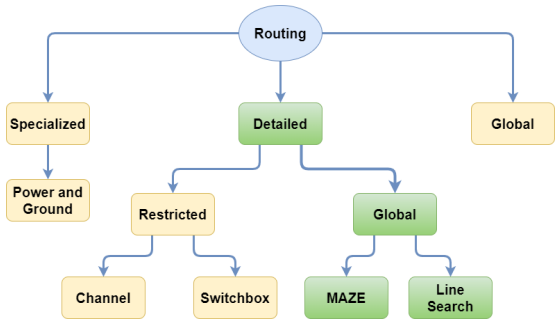

Fig. (1) Anatomy of various routing techniques.

In the next section we are showing how various algorithms work, and discussing three algorithms, first the Lee algorithm it's a Maze based algorithm, second the Mikami-Tabuchi algorithm it's a Line-Probe algorithm, and finally the Steiner algorithm it's a baseline algorithm.

C. Simulation environment

Rapidly we'll describe our Simulation's properties:

- Detailed routing (not global)
- System is presented as $3D$ Grid,$[D, W, H]$
  - D: number of layers
  - W: width of each grid
  - H: height of each grid
- Only one source exist as a start.
- Multiple targets exist (nested).
- Consistent Cost: no bending cost, no extra cost due to vias.
- Vias cost is 1.
- Within the same 2D grid path could be horizontally or/and vertically chosen.

D. Explored Techniques

Before diving into these algorithms make sure to read the Terminology section first.

1) Lee algorithm: This is one of the most common and origin routing algorithms [2].

If there's a path between source S and some target T, the algorithm will definitely find it, and in case of consistent cost (ie. no variable cost) the algorithm will not only find the a path but also the shortest one.

It uses BFS (breadth-first search) to connect targets with the source.

It works appropriately with multiple layers (ie, where vias exist).

The algorithm has main three stages:

- Expansion
- Back-tracking
- Clean up

The *Expansion* stage fig.**??** creates like a *halo* shape around the source, and it gets larger and the cost of each cell is incremented, unless there's an obstacle (block cell), until it hits a target and terminates, if the expansion reached it's limit with no target hit, this means that the target/s is/are not reachable.

The *Back-tracking* stage fig.3 gets the path from the target T to the source S. Since we are dealing with consistent cost, then the backtracking stage is not that much of an issue all we have to do is to decrement the cost by 1 from the target, until we reach the source. In other versions where inconsistent cost exist, and the cost of the path is the total length of it. **Data structure** such as **priority queue** is used to *pop-up* the cell with minimum cost.

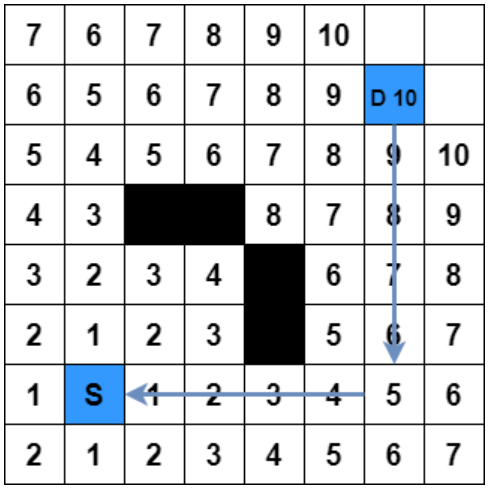

Fig. (3) Back-tracking Stage

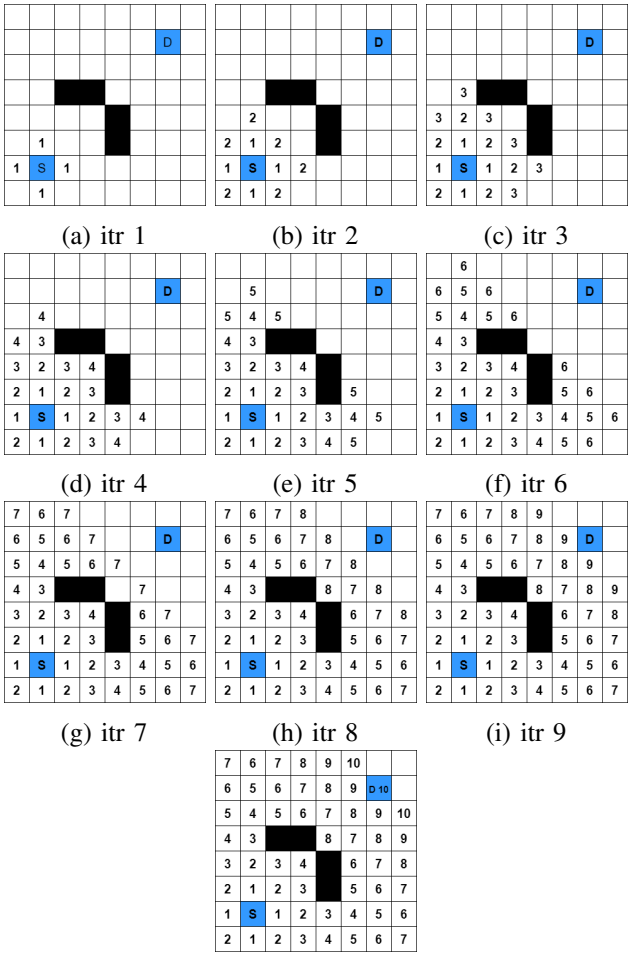

Fig. (2) Expansion Stage

The *Clean up* stage fig.4 converts the path from the source to the target into obstacles (blocks) so that no interference between pathes may exist. and then starts to connect another target to the source with that path of obstacles added.

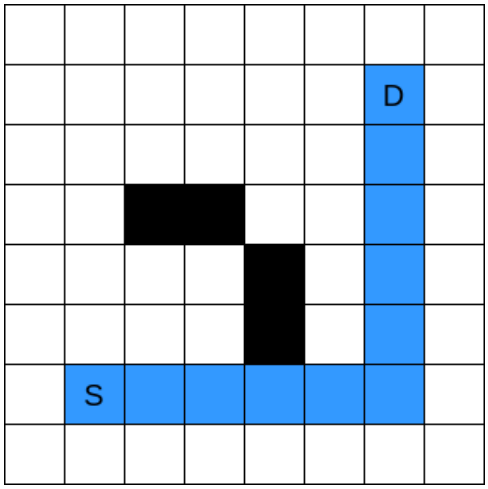

Fig. (4) Clean up Stage

2) Mikami-Tabuchi algorithm:

3) Steiner algorithm:

IV. Methodology

A. Motivation

Metal routing is a critical step in systems integration process, where each source is connected to its fan-outs using non-crossing metal. Routing in a modern chip, with millions or even billions of transistors, can be very complicated and cumbersome, so the manufacture process has moved to automatic routing.

Automatic routing is a very vast field, with several techniques being developed through time. Automatic routing involves lots of problems like finding shortest non-blocked path and VIAs in multi-layer routing.

Automatic routing techniques are always a trade-off between optimality and speed. Some techniques target execution time by reaching a sub-optimal solution. Others target optimal solution with very high execution time.

In this work, we seek a good balance between performance and optimality of solution by introducing the usage of A* search for automatic routing with a modified cost function.

B. Formulation

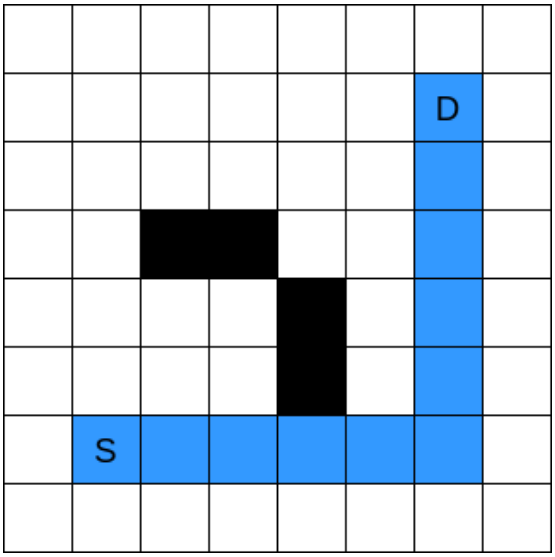

Fig. (5) Example of occupancy grid map (OGM), where S is the source, D is the destination, white cells aren't occupied, black cells are occupied and blue cells represent the path.

Routing is formulated as a grid search problem 5, where an occupancy grid map *(OGM)* is provided with some source and destination cells to be joined. The *OGM* cells can have the value of 1 for occupied cell or 0 for empty cell. The *OGM* can be of multiple levels, in case of multi-layer routing, where the cell can have a third value to indicate that the cell can be used as a VIA. Only 4-neighbor cells are considered, so the path can move up, down, left or right. The path can also move from one level to another through VIA cells.

Given this problem formulation, several grid search and shortest path algorithms can be used to find the optimal path between each source and its given destinations. These algorithms can vary based on optimality and performance.

The two main metrics considered in this work are path length and execution time. Total length of metal, covering the grid, is used as a measure of solution optimality. It's defined as the number of grid cells, on which the metal is placed for routing.

Execution time is used as a measure of algorithm performance. It's simply the total elapsed time by the algorithm to find all required paths.

C. Baseline

To measure the validity and performance of our proposed method, three main baselines are used. These baselines are well-established algorithms that are currently used in industry.

1) Maze Routing (Lee's Algorithm): The first baseline algorithm, to be considered, is *maze routing*. Maze routing is one of the most well-established and widely-used algorithms in metal routing. It consists of a wide range of algorithms and techniques. *Lee's algorithm* is the basic maze routing solution and it's considered as our first baseline for comparison. This algorithm is basically a grid search algorithm for routing.

2) Mikami-Tabuchi's Algorithm: The second considered baseline algorithm is *Mikami-Tabuchi's* algorithm. Unlike *Lee's algorithm*, this algorithm is a line search technique that adopts line-search operations. It can find a path between source and destination cells, but it doesn't guarantee the shortest possible path.

3) Steiner Tree Algorithm: Steiner tree spans though the given subset of vertices in a given graph. Steiner tree is suitable for any situation, where the task is minimize cost of connection among some important locations, like VLSI Design, Computer Networks, etc. So, Steiner tree is considered as our final baseline algorithm.

D. Modified A Search*

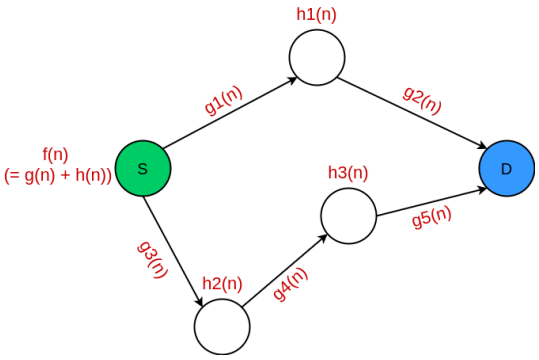

Fig. (6) An illustrative example for A* search process and cost function.

The main contribution of this work is the usage of *A* search* as a new routing technique. The main idea of *A* search*, as shown in fig.6, is the usage of some heuristic function to assign a cost for each node. To define a path from a source node to a destination node, the nodes that minimizes the estimated path cost are chosen at each step. The main objective is to minimize the following function:

$$f(n) = g(n) + h(n) \tag{1}$$

Where *f(n)* is the total cost of the path from a specific node, *g(n)* is the cost of the edge between current node and next chosen node and *h(n)* is the estimated cost of the next chosen node based on the used heuristic function.

Thus, the required information to solve an *A* search* problem is the cost of edge between each two nodes and the heuristic function to get an estimated cost for each node.

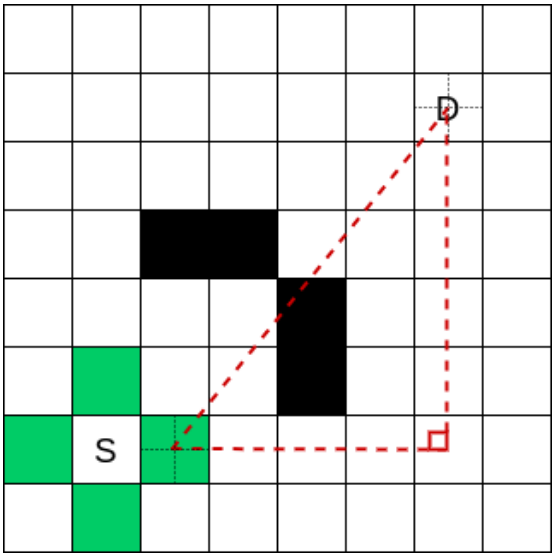

Fig. (7) An illustrative example for the new heuristic function based on Euclidean distance.

The *OGM* representation of the routing problem can be formulated to become an *A* search* problem. The edge cost *g(n)* is always 1, as the path can only move from one cell to an adjacent cell. The heuristic function, proposed in this work, is *Euclidean distance* between next chosen cell and destination cell, as shown in fig.7. So, the cost of taking any of the adjacent cells is *Euclidean distance* + 1, which can be simplified to *Euclidean distance* between an adjacent cell and destination cell.

The proposed algorithm can be summarized as follows:

V. Experimental Results

VI. Conclusion

References

[1] Edward F. Moore. The shortest path through a maze, 1959.

[2] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computers*, (EC-10 (2): 346–365, doi:10.1109/TEC.1961.), 1961.

[3] D. Hightower. The lee router revisited. *IEEE international Conference on computer design: 136-139*, 1983.

Algorithm 1: Modified A* Search For Automatic Routing

```
Result: Optimal path between source and
        destination cells.
append source cell to the path;
while destination not reached do
    get adjacent cell that minimizes euclidean
     distance;
    append cell to the path;
    if current cell is blocked then
        remove current cell from path;
        backtrack to the previous cell;
        continue;
    end
end
```