Obstacle-aware Synthesis of the Bus Topology Considering Wire Length Minimization

Meng Lian[1], Yushen Zhang[1], Mengchu Li[1], Tsun-Ming Tseng[1], Shejun Sun[2], and Ulf Schlichtmann[1]
[1]{m.lian, yushen.zhang, mengchu.li, tsun-ming.tseng, ulf.schlichtmann}@tum.de, [2]sunshejun@huawei.com
[1]Technical University of Munich, [2]Huawei Device Co., Ltd

***Abstract*—The bus topology, crucial in electronics for multi-device communication, faces challenges with an increasing number of devices and application-specific physical constraints. This work mathematically models bus topological features and obstacle-aware routing constraints in the rectilinear and octilinear routing planes to synthesize the bus topology with minimum total wire length. We implement our rectilinear and octilinear synthesis methods by constructing mixed-integer-linear programming (MILP) models and investigate their performance using eleven commercial inter-integrated circuit (I^2C) buses on a smartphone motherboard. Experimental results confirm that our methods can efficiently synthesize bus topologies with significantly shorter wire lengths, up to 24.3%, compared to two baseline methods.**

***Index Terms*—Bus topology, Rectilinear routing, Octilinear routing, Mixed-integer-linear programming**

I. Introduction

The bus topology is widely employed in the electronics industry due to its reliable and cost-effective data transmission capabilities. Fig. 1(a) illustrates a bus topology where multiple devices are sequentially connected to a shared communication medium. The bus topology forms the basis of multi-device communication within printed circuit boards (PCBs) and integrated circuits (ICs), enabling data exchange among various device categories, such as industrial sensors, displays, memory modules, audio codecs, and numerous other devices. A typical example of the bus topology is the inter-integrated circuit (I^2C) bus, as shown in Fig. 1(b), which requires two wires, i.e., the serial data line (SDA) and the serial clock line (SCL), as the shared communication medium to enable data exchange between the microcontroller (master device) and various peripherals (slave devices) [1].

During the synthesis of a bus based on the positions of the devices to be connected and the obstacles, as illustrated in Fig. 2(a), designers need to decide the order in which the devices are connected to the shared communication medium and route the wires to implement the shared and the branching wires without hitting obstacles. Notably, in environments like smartphone motherboards, where routing areas are extremely constrained and wire density is high, minimizing wire length is imperative for conserving space and optimizing area usage, thus enhancing subsequent wire placements' efficiency. In particular, significant obstacles, such as large electronic components like antennas and cameras, often result in central voids on the board, as shown in Fig. 2(a). In such cases, wire detours within the paths of bus routing relying on simple

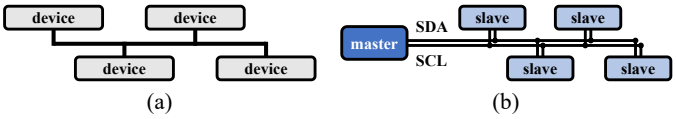


Fig. 1. (a) The bus topology. (b) The I^2C bus topology.

optimization criteria, such as connecting each device to its nearest neighbor, can result in substantial increases in the required wire length, leading to suboptimal design outcomes and considerable inefficiencies in resource use.

Historically, buses have been designed manually and employed in relatively small-scale applications for basic data exchange. However, with an increasing number of devices to be connected and application-specific physical constraints, the manual design of buses can be time-consuming, and the quality of the outcome often relies on the designers' experience. For example, the manual design on a smartphone motherboard shown in Fig. 2(a) can lead to a suboptimal result shown in Fig. 2(b). Specifically, the configuration of the wires originating from the master involves two potential detours to avoid obstacles: one route traverses above the obstacles

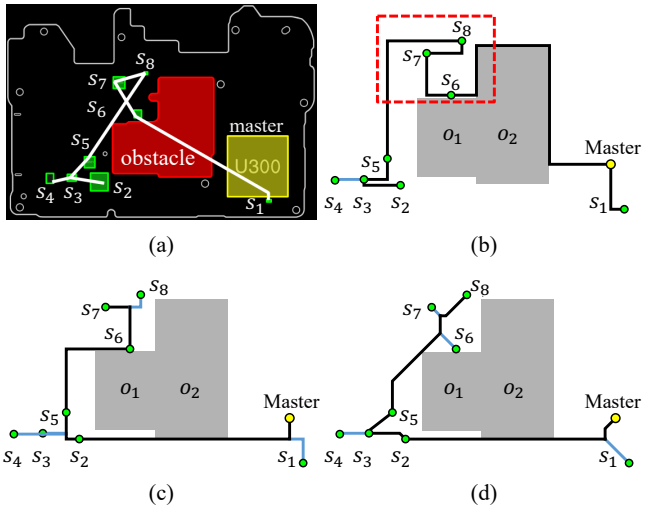


Fig. 2. Illustration of an I^2C bus, where green points denote slave devices (s_1–s_8), yellow points denote master devices, white lines denote manually designed logical connections between devices, black lines denote the shared communication medium, and blue lines denote the branching wires connecting devices. (a) Positions of devices and obstacles, with devices connected to their nearest neighbor while ignoring obstacles. (b) Non-optimized I^2C bus, where wires meander unnecessarily within the outlined area. Optimized I^2C buses in the (c) rectilinear and (d) octilinear routing planes.

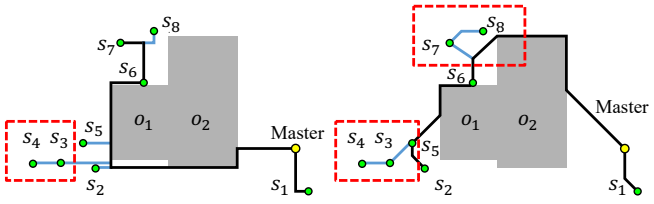


Fig. 3. Illustration of the routing results for the positions of devices and obstacles in Fig. 2 using (a) the OARSMT algorithm in [2] and (b) the OAOSMT algorithm in [3], where the connections of devices within the outlined areas violate the configuration of the bus topology.

to connect with s_6, s_7, or s_8, while an alternative route goes below to connect with s_2, s_3, s_4, or s_5. Compared to the optimal rectilinear routing outcome shown in Fig. 2(c), the master in the suboptimal I^{2}C bus is connected to s_6, chosen based on its minimal Manhattan distance to the master. This decision results in the upper routing path, significantly increasing wire lengths due to the large area occupied by obstacles. Meanwhile, wastage in routing areas is evident, as shown in Fig. 2(b) by the unnecessary meandering of wires within the area outlined in red. Further, routing using octilinear architecture, as depicted in Fig. 2(d), can reduce the wire length required for the I^{2}C bus instance even further.

When examining general obstacle-aware routing problems considering wire length minimization, the most widely acknowledged automatic rectilinear or octilinear routing method is the obstacle-avoiding Steiner minimum tree (OASMT) [2]–[10], which constructs a tree topology among a group of devices in the presence of obstacles. However, OASMT algorithms generate routing trees where devices are connected in a branching manner that directly contradicts the strict sequential connectivity required by bus topologies, where all devices must connect along a shared communication medium. For our exemplary I^{2}C bus, we illustrate the routing results using the obstacle-avoiding rectilinear Steiner minimal tree (OARSMT) algorithm in [2] and the obstacle-avoiding octilinear Steiner minimal tree (OAOSMT) algorithm in [3] in Figs. 3(a) and 3(b), respectively. Specifically, in Fig. 3(a), s_4 is not connected to the shared communication medium but rather links directly to s_3. Similarly, in Fig. 3(b), s_4 and s_8 are connected to s_3 and s_7, respectively. These deviations are not merely suboptimal but fundamentally incompatible with bus functionality, as the tree-based design disrupts the uniform signal propagation and timing consistency required in buses. While OASMT algorithms focus on minimizing wire lengths, they cannot produce or easily adapt to the strict sequential connectivity required by the bus topology. This fundamental conflict makes OASMT algorithms entirely unsuitable for applications requiring the bus topology.

This work proposes automatic *rectilinear* and *octilinear synthesis methods* of the bus topology with minimum wire length considering potential obstacles in rectilinear and octilinear routing planes, respectively. Both synthesis methods are implemented as mixed-integer-linear programming (MILP) models. The main contributions of our work are summarized as follows:

- It is the first work that mathematically models the bus topology.
- It proposes an obstacle-aware wire length calculation model that identifies detour requirements and determines the appropriate detour paths.
- It optimizes the connection order of devices and the routing of the wires simultaneously.

The rest of this paper is organized as follows. Section II formulates the problem. Sections III and IV detail the rectilinear and octilinear synthesis methods, respectively. Experimental results are shown in Section V, followed by our conclusion in Section VI.

II. Problem Formulation

This work aims to solve the following problem:
Input: The positions of (master/slave) devices and obstacles.
Output: Routed wires in a bus topology.
Constraints: Devices must connect to a shared communication medium in a strictly sequential manner, as defined by the bus topology. The wires must not overlap with the obstacles.
Objective: Minimize wire lengths with adjustable weight coefficients for the wire lengths of the shared and branching wires.

III. Rectilinear Synthesis Method

Our rectilinear method mathematically models bus topological features and obstacle-aware routing constraints to synthesize buses in the rectilinear plane with minimum wire length. The frequently used variables of our rectilinear method are outlined in Table I.

A. Bus topology

Every device is modeled as a point. Fig. 4 illustrates our model of the bus topology, where the shared communication

TABLE I
Model Variables

Binary variables	
q_{ij}	The i^{th} virtual point connects to the j^{th} device.
$q^l_{i,k}$, $q^r_{i,k}$, $q^t_{i,k}$, $q^b_{i,k}$	The i^{th} virtual point is located on the left, right, top, or bottom relative to the k^{th} obstacle.
$q^{s_L}_{i,k}$, $q^{s_R}_{i,k}$, $q^{s_T}_{i,k}$, $q^{s_B}_{i,k}$	The i^{th} virtual point is strictly located to the left, right, top, or bottom relative to the k^{th} obstacle.
$q^{r_h}_{(m,n)_k}$, $q^{r_v}_{(m,n)_k}$	Points p_m and p_n are horizontally or vertically opposite regarding the k^{th} obstacle.
$q_{(m,n)_k}$	Manhattan paths of points p_m and p_n intersect with the k^{th} obstacle.
$q^{tl}_{(m,n)_k}$, $q^{tr}_{(m,n)_k}$ $q^{br}_{(m,n)_k}$, $q^{bl}_{(m,n)_k}$	The path connecting points p_m and p_n bypasses via the top-left, top-right, bottom-right, or bottom-left corner of the k^{th} obstacle.
$q^{k,k'}_{m,n}$	The k^{th} obstacle's bypass corner connects to the k'^{th} obstacle's.
$q_{(m)_k,n}$, $q_{m,_k(n)}$	The k^{th} obstacle's selected bypass corner is connected by points p_m or p_n within the path connecting points p_m and p_n.
Continuous variables	
x_i, y_i	The x- and y-coordinates of the i^{th} virtual point.
$x_{(m,n)_k}$, $y_{(m,n)_k}$	The x- and y-coordinates of the k^{th} obstacle's bypass corner within the path connecting points p_m and p_n.
$d_M(p_m, p_n)$	Manhattan distance between points p_m and p_n.
$d^m_{m,n}$, $d^{k,k'}_{m,n}$, $d^n_{m,n}$	Distance between point p_m and its connecting bypass corner, between the k^{th} and k'^{th} obstacles, and between point p_n and its connecting bypass corner within the routing path connecting points p_m and p_n.
l_{trunk}, l_{branch}	Total wire lengths of the trunk and branch segments.

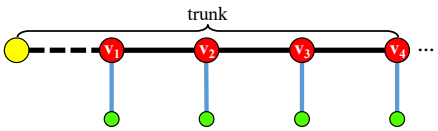


Fig. 4. The model of the bus topology features trunk segments in black and branch segments in blue, with yellow circles as the master device, green circles as slave devices, and red circles as virtual points.

medium is referred to as the *trunk*, and the branching wires connecting devices to the trunk are referred to as the *branch segments*. Meanwhile, to decide on the device connection order, we introduce *virtual points* that serve as the interfaces for the branching segments to access the trunk. Specifically, without loss generality, we define the connection order of virtual points as $v_1, v_2, \ldots, v_{n_d}$, where n_d denotes the number of devices. Thus, deciding the device connection order is equivalent to determining the device connected to each virtual point. Further, for buses using a *master-slave architecture*, such as I^{2}C buses, we connect v_1 to the master.

To establish the one-to-one correspondence between virtual points and devices, we define a binary variable q_{ij} that specifies whether the i^{th} virtual point connects to the j^{th} device and introduce the following constraints.

$$\forall_{1\le i\le n_d}: \sum_{1\le j\le n_d} q_{ij} = 1, \quad \forall_{1\le j\le n_d}: \sum_{1\le i\le n_d} q_{ij} = 1. \tag{1}$$

In multi-wire buses, e.g., four-wire serial buses (Serial Peripheral Interface) and two-wire I^{2}C buses, we abstract multiple wires into a single line to ensure uniform wire lengths within the bus. Ensuring this uniformity simplifies the overall routing architecture while also playing a crucial role in preserving consistent signal propagation delays, which are essential for maintaining precise timing and synchronization across devices. In time-sensitive systems, uniform wire lengths help mitigate timing discrepancies and reduce the risk of skew between signals.

B. Non-overlapping with obstacles

We represent each obstacle by a bounding box with its edges set at a distance of a minimum separation ϵ from the obstacle, as shown in Fig. 5(a), to ensure that the distances between virtual points and obstacles exceed ϵ. The position of any given obstacle, such as the k^{th} obstacle o_k, is represented with the coordinates of its bounding box's bottom-left and top-right corners, denoted as $(\min x_k, \min y_k)$ and $(\max x_k, \max y_k)$, respectively. To guarantee that the i^{th} virtual point v_i does not overlap with o_k, v_i should be located on the left, right, top, or bottom relative to o_k, i.e.,

$$q^l_{i,k} + q^r_{i,k} + q^t_{i,k} + q^b_{i,k} \ge 1, \tag{2}$$

where binary variables $q^l_{i,k}$, $q^r_{i,k}$, $q^t_{i,k}$, and $q^b_{i,k}$ indicate whether v_i is located on the *left*, *right*, *top*, or *bottom* relative to o_k, referring to the scenarios in Fig. 5(b), respectively. For example, v_i is located to the left of o_k, specifically to the left of the line $y = \min x_k$, implying that the binary variable $q^l_{i,k}$ is 1; otherwise, it equals 0. We characterize the relative

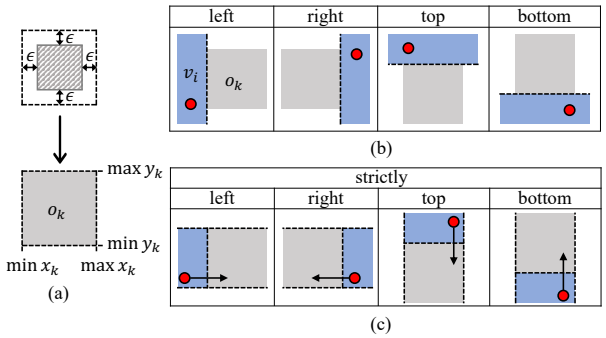


Fig. 5. (a) Each obstacle is represented by a bounding box with its edges set at a distance of a minimum separation ϵ from the obstacle. (b) Relative locations: v_i can be located on the left, right, top, or bottom relative to o_k, where blue areas indicate possible locations of v_i. (c) Strictly relative locations: v_i is exclusively located on the left, right, top, or bottom side of o_k.

location of v_i to the left of o_k using the *big M method* [11] as

$$x_i \ge \min x_k - q^l_{i,k} M, \tag{3a}$$
$$x_i \le \min x_k + \big(1 - q^l_{i,k}\big) M, \tag{3b}$$

where M is an extremely large constant. Specifically, when v_i is located to the left of o_k, (3a) ensures that $q^l_{i,k}$ equals 1, while (3b) sets it to 0 otherwise. For example, in the scenario where v_i is located exclusively to the left of o_k, binary variables $q^l_{i,k}$, $q^r_{i,k}$, $q^t_{i,k}$, and $q^b_{i,k}$ will take values of 1, 0, 0, and 0, respectively. When v_i is located on the top-left relative to o_k, indicating it is both on the left and top relative to o_k, the values of $q^l_{i,k}$, $q^r_{i,k}$, $q^t_{i,k}$, and $q^b_{i,k}$ will be 1, 0, 1, and 0, respectively.

C. Obstacle-aware wire length calculation

Based on the relative locations of virtual points to obstacles, we identify whether the connection between a virtual point and its assigned device or between two consecutive virtual points requires a detour to avoid obstacles. Once a detour is necessary, we apply a *detour mechanism* to navigate around those obstacles. After that, the lengths of the trunk and branch segments can be calculated considering obstacles, and we then minimize the total wire length to obtain the optimal locations of virtual points and device connection order.

Strictly relative locations: A virtual point is considered *strictly* on the left, right, top, or bottom relative to an obstacle if the rays originating from that virtual point in the respective opposite directions (right, left, bottom, or top) overlap with that obstacle. This implies that a virtual point is strictly relative to an obstacle if it is exclusively located on one of these four sides of that obstacle, as shown in Fig. 5(c). For example, v_i being not located to the right, top, or bottom of o_k implies that v_i is strictly placed to the left of o_k, represented by a binary variable $q^{s_L}_{i,k}$.

$$q^{s_L}_{i,k} \ge 1 - q^r_{i,k} - q^t_{i,k} - q^b_{i,k}. \tag{4}$$

The binary variables $q^{s_R}_{i,k}$, $q^{s_T}_{i,k}$, and $q^{s_B}_{i,k}$, describing v_i strictly to the right, top, and bottom of o_k, respectively, can be identified analogously.

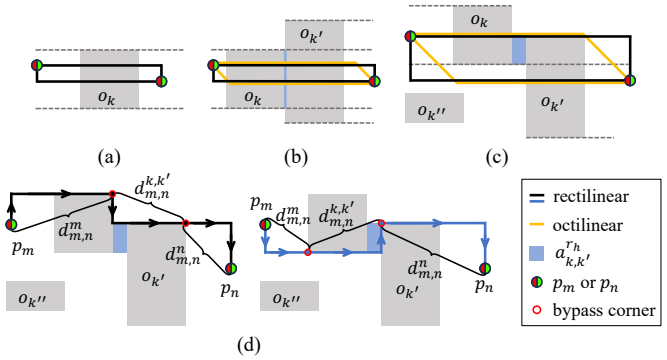


Fig. 6. Illustration of points p_m and p_n in (a)–(c) a horizontal opposite relation and (d) their routing paths.

Rectilinear opposite relations: We call the routing paths connecting two points with a length equal to their Manhattan distance as their *Manhattan paths*. When two points are on opposite sides of an obstacle, either the horizontal or the vertical segments of their Manhattan paths will inevitably overlap with that obstacle. In such cases, these points are identified as being *horizontally* or *vertically opposite* regarding that obstacle, respectively. As an example shown in Fig. 6(a), points p_m and p_n, representing virtual points or devices, are located strictly on the left and right relative to obstacle o_k, respectively. As a result, the horizontal segments of their Manhattan paths will overlap with o_k. The constraints characterizing this horizontal opposite relation can be formulated as

$$q^{r_h}_{(m,n)_k} \geq q^{s_L}_{m,k} + q^{s_R}_{n,k} - 1, \;\; q^{r_h}_{(m,n)_k} \geq q^{s_L}_{n,k} + q^{s_R}_{m,k} - 1. \quad (5)$$

Trivially, two points can be opposite regarding multiple obstacles, as shown in Fig. 6(b), where p_m and p_n are horizontally opposite regarding both o_1 and o_2. Further, two points may not be opposite regarding any *single* obstacle, yet they can still exhibit an opposite relation regarding a group of obstacles, as their Manhattan paths overlap with those obstacles, as shown in Fig. 6(c).

We denote the intersection of y-coordinate boundaries of o_k and $o_{k'}$ as a set of points $a^{r_h}_{k,k'}$, referring to the blue areas in Figs. 6(b)–6(d). If $a^{r_h}_{k,k'} = \emptyset$, the Manhattan paths between p_m and p_n can pass through the gap between these obstacles. In contrast, if $a^{r_h}_{k,k'} \neq \emptyset$, their Manhattan paths will cross o_k or $o_{k'}$, as shown in Fig. 6(c). To detect all obstacles intersecting their Manhattan paths in horizontal segments, we revise (5) accordingly. For all $o_{k'} \in O$ with $a^{r_h}_{k,k'} \neq \emptyset$:

$$q^{r_h}_{(m,n)_k}, \; q^{r_h}_{(m,n)_{k'}} \geq q^{s_L}_{m,k} + q^{s_R}_{n,k'} - 1, \; q^{s_L}_{n,k} + q^{s_R}_{m,k'} - 1, \quad (6)$$

where O contains all obstacles. Further, o_k is called *relevant* to the routing path between p_m and p_n, denoted by binary variable $q_{(m,n)_k}$, if the Manhattan paths of p_m and p_n intersect with o_k in their horizontal or vertical segments.

$$\forall_{o_k \in O}: \quad q^{r_h}_{(m,n)_k} + q^{r_v}_{(m,n)_k} \leq 2 \cdot q_{(m,n)_k}. \quad (7)$$

For example, in Fig. 6(c), o_k and $o_{k'}$ are both identified as relevant, necessitating a detour of the routing path between p_m and p_n. Conversely, $o_{k''}$ is identified as irrelevant, indicating that its presence does not obstruct the routing path connecting p_m and p_n.

Detour mechanism: Our work navigates detours via a single corner of relevant obstacles, referred to as *bypass corner*, and selects the bypass corner that minimizes the total wire length. Specifically, we construct the path connecting p_m and p_n as a path that originates at p_m and proceeds progressively via each selected bypass corner to arrive at p_n, as shown in Fig. 6(d). To decide the bypass corner of a relevant o_k, we define binary variables $q^{tl}_{(m,n)_k}$, $q^{tr}_{(m,n)_k}$, $q^{br}_{(m,n)_k}$, and $q^{bl}_{(m,n)_k}$, specifying the bypass via its top-left, top-right, bottom-right, or bottom-left corner, respectively. Then, if o_k is relevant, at least one of its corners will be selected.

$$q^{tl}_{(m,n)_k} + q^{tr}_{(m,n)_k} + q^{br}_{(m,n)_k} + q^{bl}_{(m,n)_k} \geq q_{(m,n)_k}. \quad (8)$$

For example, within the black path in Fig. 6(d), the top-right corners of o_k and $o_{k'}$ are selected as bypass corners. Also, $o_{k''}$ has no bypass corner as it is irrelevant to p_m and p_n. Meanwhile, o_k's selected bypass corner $c_{(m,n)_k}$ has the coordinates of

$$\begin{aligned} x_{(m,n)_k} &= \min x_k \cdot q^{bl}_{(m,n)_k} + \min x_k \cdot q^{tl}_{(m,n)_k} \\ &\quad + \max x_k \cdot q^{br}_{(m,n)_k} + \max x_k \cdot q^{tr}_{(m,n)_k}, \\ y_{(m,n)_k} &= \min y_k \cdot q^{bl}_{(m,n)_k} + \min y_k \cdot q^{br}_{(m,n)_k} \\ &\quad + \max y_k \cdot q^{tl}_{(m,n)_k} + \max y_k \cdot q^{tr}_{(m,n)_k}. \end{aligned} \quad (9)$$

A bypass corner of a relevant obstacle between p_m and p_n can be connected to p_m, p_n, or the bypass corner of another relevant obstacle between p_m and p_n. Let binary variable $q^{k,k'}_{m,n}$ indicate the connection of o_k's bypass corner to $o_{k'}$'s. Then, the following constraints detail these specific connecting principles.

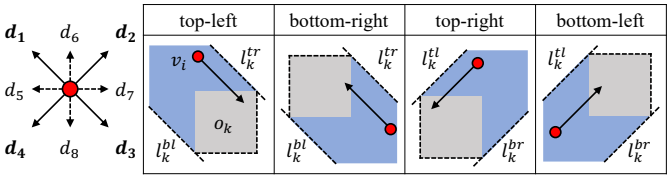
$$\begin{aligned} q_{(m)_k,n} + \sum_{k \neq k'} q^{k',k}_{m,n} \cdot q_{(m,n)_{k'}} &= q_{(m,n)_k} \\ q_{m,_k(n)} + \sum_{k \neq k'} q^{k,k'}_{m,n} \cdot q_{(m,n)_{k'}} &= q_{(m,n)_k}, \end{aligned} \quad (10)$$

where binary variables $q_{(m)_k,n}$ and $q_{m,_k(n)}$ indicate whether o_k's bypass corner is connected by p_m or p_n, respectively. The product terms ensure the bypass corner of a relevant obstacle cannot be connected to the bypass corner of an irrelevant obstacle. Here, the product of two binary variables, e.g., $C = A \cdot B$, is linearized as $C \leq A$, $C \leq B$, and $C \geq A + B - 1$.

Wire length calculation: The Manhattan distance of two points is calculated as the sum of the differences in their coordinates. Linearizing the absolute value function, continuous variable $d_M(p_m, p_n)$, denoting the Manhattan distance between p_m and p_n, is formulated as

$$\begin{aligned} d_M(p_m, p_n) \geq\; & x_m - x_n + y_m - y_n, \;\; x_m - x_n + y_n - y_m, \\ & x_n - x_m + y_m - y_n, \;\; x_n - x_m + y_n - y_m, \end{aligned} \quad (11)$$

where x_m, y_m, x_n, and y_n are continuous variables denoting coordinates of p_m and p_n. As minimizing the total wire length



(a) (b)

Fig. 7. (a) Wire segment directions in the octilinear routing plane. (b) Relative locations: v_i is on the top-left, bottom-right, top-right, and bottom-left relative to o_k, where the blue areas indicate possible locations of v_i.

is our optimization objective, $d_M(p_m, p_n)$ is constrained to the largest value among right-hand side terms.

For points in opposite relations, based on the connection principles introduced in (10), the length of the routing path connecting p_m and p_n, denoted by continuous variable $d(p_m, p_n)$, is formulated as

$$d(p_m, p_n) \geq d^m_{m,n} + \sum_{o_k, o_{k'} \in O} d^{k,k'}_{m,n} + d^n_{m,n}, \tag{12}$$

where continuous variables $d^m_{m,n}$, $d^{k,k'}_{m,n}$, and $d^n_{m,n}$, representing the distances between p_m and its connecting bypass corner, between o_k and $o_{k'}$, and between p_n and its connecting bypass corner, respectively, can be derived as

$$\begin{aligned} d^m_{m,n} &\geq d_M\big(p_m, c_{(m,n)_k}\big) - \big(1 - q_{(m)_k,n}\big)M, \\ d^{k,k'}_{m,n} &\geq d_M\big(c_{(m,n)_k}, c_{(m,n)_{k'}}\big) - \big(1 - q^{k,k'}_{m,n}\big)M, \\ d^n_{m,n} &\geq d_M\big(c_{(m,n)_k}, p_n\big) - \big(1 - q_{m,k(n)}\big)M, \end{aligned} \tag{13}$$

where $c_{(m,n)_k}$ denotes o_k's selected bypass corner.

Utilizing the length computation between two points, we establish constraints to define the total wire length of a bus topology. For a bus consisting of n_d devices, the trunk, which is considered a path that sequentially traverses each virtual point, has a length of

$$l_{trunk} = \sum_{1 \leq i \leq n_d - 1} d(v_i, v_{i+1}). \tag{14}$$

In a master-slave structured bus, we include the distance between the master and the first virtual point in l_{trunk}. Meanwhile, the total wire length of branch segments is

$$l_{branch} = \sum_{1 \leq i \leq n_d} d(v_i, \tilde{s}_i), \tag{15}$$

where $d(v_i, \tilde{s}_i)$, representing the distance between v_i and its connecting device $\tilde{s}_i$, is formulated as

$$d(v_i, \tilde{s}_i) \geq d(v_i, s_j) - (1 - q_{ij})M, \tag{16}$$

where s_j denotes the j^{th} device.

Finally, the objective function is formulated as

$$\begin{aligned} &\textit{minimize: } c_{trunk} \cdot l_{trunk} + c_{branch} \cdot l_{branch}, \\ &\textit{Subject to: (1)–(4), (6)–(16),} \end{aligned}$$

where c_{trunk} and c_{branch} are adjustable weight coefficients.

IV. Octilinear Synthesis Method

Considering the capability of octilinear architecture in reducing wire lengths, we expand our rectilinear synthesis method to octilinear routing. For computational efficiency, we retain the optimized device connection order achieved in the rectilinear synthesis method. In other words, the optimization starts with q_{ij} preset to the value that minimizes the wire length in rectilinear routing.

A. Octilinear opposite relations

When two points are in a rectilinear opposite relation, they are also in an opposite relation on an octilinear plane, as shown in Figs. 6(b) and 6(c). Further, we define two additional categories of opposite relations specific to the octilinear routing: *diagonal* and *semi-diagonal* opposite relations.

Relative locations: In addition to the horizontal and vertical directions, the octilinear routing plane also allows wire segments to be angled at 45° and 135°, as shown in Fig. 7(a). A virtual point v_i is considered on the *top-left*, *bottom-right*, *top-right*, or *bottom-left* relative to o_k if the rays originating from v_i in the respective opposite directions (d_3, d_1, d_4, or d_2) overlap with o_k, as shown in Fig. 7(b), where boundary lines l_k^{tr}, l_k^{bl}, l_k^{tl}, and l_k^{br} have a gradient of either 1 or -1, intersecting top-right, bottom-left, top-left, or bottom-right corners of o_k. The constraints characterizing these relative locations can be formulated analogous to (3).

Octilinear opposite relations: We refer to the shortest routing paths that connect two points in the octilinear routing plane, i.e., without consideration of detour, as their *octilinear paths*. When two points are on diagonally opposite sides of an obstacle, e.g., being on the top-left and bottom-right, respectively, as shown in Fig. 8(a), they are identified as being 45° or 135° diagonal opposite regarding that obstacle as either the 45° or 135° segments of their octilinear paths will inevitably overlap with the obstacle.

Further, the semi-diagonal opposite relation is a hybrid of rectilinear and diagonal opposite relations. For example, as shown in Fig. 8(b), when one of two to-be-connected points is located within the intersection of the strictly left and top-left areas relative to o_k, referring to the red area, and the other point is located on the corresponding opposite side of o_k, referring to the yellow area, their octilinear paths will overlap with o_k. The identification constraints for diagonal and semi-diagonal opposite relations can be formulated similarly to (6).

B. Obstacle-aware wire length calculation

Detour mechanism: In the octilinear plane, our detour mechanism builds on the principles used in the rectilinear plane, where detours are navigated via bypass corners of relevant obstacles. Considering the additional wire segment directions in the octilinear plane, we introduce the following bypass corner selection rules to prevent overlapping routing paths with obstacles:

1) For each segment along the routing path from the start to the endpoint, if the segment's start point is strictly on the left, right, top, or bottom relative to an obstacle,

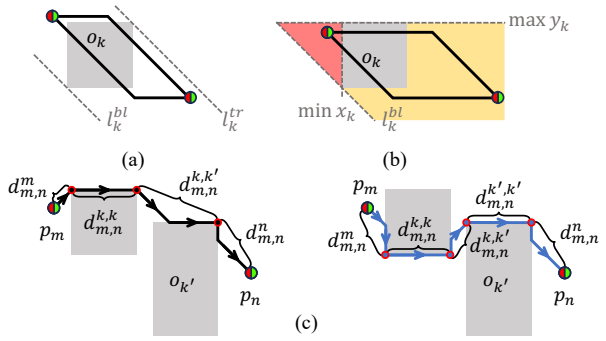


Fig. 8. Illustration of points p_m and p_n in (a) diagonal and (b) semi-diagonal opposite relations and (c) their routing paths.

at least one corner on that side of the obstacle must be chosen as a bypass corner.

2) For each segment along the routing path from the start to the endpoint, if the segment's start point is on the top-left, bottom-right, top-right, or bottom-left relative to an obstacle, at least one of the corners on the left or top, right or bottom, right or top, or left or bottom edges should be selected as bypass corners, respectively.

For example, in Fig. 8(c), with p_m being strictly left to o_k, at least one corner on the left edge of o_k should be selected if o_k is connected by p_m.

$$q^{tl}_{(m,n)_k} + q^{bl}_{(m,n)_k} \geq 1 - \left(2 - q_{(m)_k,n} - q^{s_L}_{m,k}\right) M. \tag{17}$$

Here, the black and blue paths indicate the selection of o_k's top-left and bottom-left corners, respectively. Further, since the top-right corner of o_k is on the top-left relative to $o_{k'}$, at least one corner on the left or top edges of $o_{k'}$ should be selected if $o_{k'}$'s bypass corner is connected by o_k's.

$$q^{tl}_{(m,n)_{k'}} + q^{tr}_{(m,n)_{k'}} + q^{bl}_{(m,n)_{k'}} \geq 1 - \left(2 - q^{k,k'}_{m,n} - q^{tr}_{(m,n)_k}\right) M. \tag{18}$$

The black path in Fig. 8(c) shows the selection of $o_{k'}$'s top-right corner. Similar constraints can be constructed to restrict bypass corner selections for other relative locations.

Wire length calculation: The length of the octilinear path between p_m and p_n is referred to as their *octilinear distance*, denoted by continuous variable $d_O(p_m, p_n)$ and given by

$$d_O(p_m, p_n) = \sqrt{2} \cdot \min\left(\Delta_x, \Delta_y\right) + \left|\Delta_x - \Delta_y\right|, \tag{19}$$

where Δ_x and Δ_y denote the coordinate differences between p_m and p_n. We linearize the minimum $\Delta_{\min} = \min(\Delta_x, \Delta_y)$ using big M method [11] as

$$\Delta_{\min} \geq \Delta_x - qM, \quad \Delta_{\min} \geq \Delta_y - (1-q)M, \tag{20}$$

where q is an auxiliary binary variable. As minimizing the total wire length is our optimization objective, $\Delta_{\min}$ is constrained to the lesser of Δ_x and Δ_y.

Applying the connecting principles introduced in Section III-C to the octilinear routing plane with distances $d^m_{m,n}$, $d^{k,k'}_{m,n}$, $d^n_{m,n}$, and $d(p_m, p_n)$ are calculated by modified (13) using the octilinear measurement in (19). Notably, the Manhattan measurement is employed to calculate the distance between two bypass corners of an obstacle to prevent wires from overlapping with obstacles. Finally, the objective function is formulated as in the rectilinear synthesis method, incorporating the constraints introduced in Section IV.

V. Experimental Results

We investigated the performance of our synthesis methods using eleven commercial I^{2}C buses in a smartphone motherboard as test cases. The details of these test cases are listed in Table II, where n_d and n_o are the numbers of devices and obstacles, respectively. For comparative analysis, we developed *Dijkstra* and *Dreyfus baseline methods* to synthesize buses using rectilinear and octilinear routings. We implemented the methods with Java and ran the experiments on a computer with an Apple M1 8-core CPU. The MILP models were solved using Gurobi [12].

In our experiments, we set the minimum separation ϵ to 3 mils and the unit length to 1 mil. Thus, the variables in Sections III and IV, describing coordinates and wire lengths, were considered integer variables. Accordingly, the equality in (19) was replaced with "$\geq$" to obtain $\lceil \sqrt{2} \min\left(\Delta_x, \Delta_y\right) + \left|\Delta_x - \Delta_y\right| \rceil$. Meanwhile, all weight coefficients used in our synthesis methods were set to 1.

A. Baseline methods

We develop *Dijkstra* and *Dreyfus baseline methods* to synthesize buses using rectilinear and octilinear routings. Unlike OASMT algorithms, which are designed for hierarchical tree structures, these baseline methods are tailored for bus topologies, ensuring correct device connections and reducing total wire length.

Dijkstra baseline method: The Dijkstra baseline method, a greedy approach, utilizes our synthesis methods with a device connection order predetermined by *Dijkstra's algorithm* [13]. Specifically, Dijkstra's algorithm is used to compute the shortest paths between device pairs considering obstacles in rectilinear and octilinear routings. After that, starting from the master, we generate a connection order by sequentially linking each device to its nearest neighbor, and our synthesis methods use this order to produce the final routing outcomes.

Dreyfus baseline method: We develop the Dreyfus baseline method based on the *Dreyfus-Wagner algorithm* [14], which systematically examines all possible configurations of device connections. This exhaustive exploration enables the identification of solutions that strictly satisfy the bus topology, thereby achieving optimality.

Given a graph $G = (V, E)$ and a set $Y \subset V$, the Dreyfus-Wagner algorithm exploits the *optimal decomposition property* [14] to determine the optimal Steiner tree $\mathcal{T}(Y)$.

TABLE II
Test cases used in the experiments, where n_d and n_o are the numbers of devices and obstacles, respectively.

case	1	2	3	4	5	6	7	8	9	10	11
n_d	3	3	3	5	5	7	8	8	9	9	12
n_o	1	2	4	1	1	1	2	4	1	2	1

TABLE III

SYNTHESIS COMPARISON, WHERE l_w IN MILS IS THE REQUIRED WIRE LENGTH, T IN SECONDS IS THE RUNTIME, AND "—" MEANS THAT THE RESULT IS UNAVAILABLE AFTER 24 HOURS OF RUNTIME.

	Rectilinear routing plane							Octilinear routing plane							
	RSM		Dijkstra	Dreyfus		Improvement (%)		OSM		Dijkstra	Dreyfus		Improvement (%)		
case	l_w	T	l_w	l_w	T	Dijkstra	Dreyfus	l_w	T	l_w	l_w	T	Dijkstra	Dreyfus	RSM
1	1261	0.14	1261	1517	0.33	0.00	16.88	1236	0.21	1236	1473	0.33	0.00	**16.09**	1.98
2	1121	0.08	1121	1240	0.34	0.00	9.60	977	0.15	977	1004	0.38	0.00	2.69	**12.85**
3	2565	0.13	2826	2686	0.30	9.24	4.50	2365	0.21	2416	2612	0.36	2.11	9.46	7.80
4	4076	0.24	4445	4076	1.12	8.30	0.00	3818	0.24	4127	3877	3.58	7.49	1.52	6.33
5	2489	0.26	2589	2489	1.20	3.86	0.00	2411	0.29	2439	2456	3.40	1.15	1.83	3.13
6	2580	0.93	2989	2580	52.99	13.68	0.00	2346	0.26	2630	2458	304.71	10.80	4.56	9.07
7	4132	9.73	4281	4132	253.63	3.48	0.00	3753	1.43	3877	4040	2169.96	3.20	7.10	9.17
8	4638	13.63	5326	5199	430.48	12.92	10.79	4465	1.26	4803	4876	1237.48	7.04	8.43	3.73
9	8392	54.82	9171	8869	3224.85	8.49	5.38	7494	1.13	7979	8431	6559.32	6.08	11.11	10.70
10	**5447**	**39.52**	**7198**	**5987**	**2659.16**	**24.33**	**9.02**	**5046**	**2.06**	**6223**	**5523**	**14925.86**	**18.91**	**8.64**	**7.36**
11	4891	380.51	6442	—	—	24.08	—	4640	1.25	5549	—	—	16.38	—	5.13
					Average	9.85	5.62					Average	6.65	7.14	7.02

Proposition (Optimal decomposition property). Let $\mathcal{T}(X)$ be any Steiner tree connecting a subset $X \subset V$ of the points in a graph $G = (V, E)$, and let x be any point in X. If X contains at least three members, then there exists a point $v \in V$ and a subset D of X such that: 1) D is a nonempty proper subset of $X - x$. 2) $\mathcal{T}(X)$ can be decomposed into three disjoint subtrees: T_1, T_2, and T_3, where T_1 connects $\{x, v\}$, T_2 connects $x \cup D$, while T_3 connects $x \cup (X - D - v)$.

The original problem can then be recursively solved by computing optimal subtrees $\mathcal{T}(X \cup v)$ for all $X \subseteq Y$ and $v \in V$. To iterate over each subtree following the I^{2}C bus topology, we solve Steiner problems connecting each subset $X \subset Y - m$, where m denotes the master device and Y denotes the set of devices to be connected. Specifically, for all $v_i \in V$, $S_i(X) = \min_{v_j \in V,\ e \in X} d(v_i, v_j) + S_j(X - e) + d(v_j, e)$, where $d(v_i, v_j)$ and $d(v_j, x)$ denote the weights of edges $\{v_i, v_j\}$ and $\{v_j, x\}$, respectively. These weights are accordingly calculated using the Manhattan and the octilinear measurements in the rectilinear and octilinear routing planes. Finally, we select one point $v_k \in V$ to connect the master device m such that $S_k(Y) = \min_{v_k \in V} d(m, v_k) + S_k(Y - m)$ is the required wire length.

We generate virtual point candidates, namely the nodes belonging to $V - Y$, based on the *escape graph* [15]. In the rectilinear routing plane, the optimal solution of an obstacle-avoiding rectilinear Steiner tree exclusively involves escape segments. Thus, we define the intersections of escape segments as virtual point candidates. Similarly, in the octilinear routing plane, virtual point candidates include, in addition to these intersections, the points where segments angled at $45°$ and $135°$ meet. When the results contain overlapping wire segments with obstacles, we choose corners that minimize the detour paths as the bypass corners.

B. Proposed vs. baseline methods

Table III presents the synthesis comparison of the proposed and baseline methods, where RSM and OSM represent the proposed rectilinear and octilinear synthesis methods, respectively, l_w in mils is the total wire length, T in seconds is the runtime, and "—" means that the result is unavailable after 24 hours of runtime.

Compared to the Dijkstra baseline method, the proposed synthesis methods require, on average, 9.85% and 6.65% shorter wire lengths in the rectilinear and octilinear routing planes, respectively. In small cases with few devices, like cases 1 and 2, the local optimal solution generated by the Dijkstra baseline method matches the global optimal solution generated by our rectilinear method. However, as the case size increases, the advantage of our methods becomes more significant, with the required wire lengths decreased by up to 24.33% in the rectilinear plane and 18.91% in the octilinear plane. In general, the comparison reveals that the device connection order greatly affects the resulting wire lengths.

Compared to the Dreyfus baseline method in the rectilinear routing plane, observations show no reduction in required wire lengths using the proposed method for cases 4–7. This occurs because we iterate over each subtree that forms an I^{2}C bus in the Dreyfus baseline method, and the final bus does not contain the wires that cross obstacles. Namely, these baseline results are optimal, indicating that the proposed rectilinear synthesis method achieves optimality. On average, the proposed rectilinear synthesis method can decrease the required wire length by 5.62% compared to the Dreyfus baseline results. Additionally, improvements in our octilinear routing results are observed in every case over the Dreyfus baseline results, with an average wire length reduction of 7.14%. This is because obstacles that do not intersect with the shortest rectilinear routing paths might still overlap with the shortest octilinear routing paths, leading to greater spatial wastage if suboptimal routing strategies are employed to avoid obstacles.

Fig. 9 displays the results for case 10 using the proposed and baseline methods, where the trunk, represented by the black line, originates at the master and sequentially traverses virtual points, represented by red points. Meanwhile, the branch segments, depicted as blue lines, connect virtual points to their corresponding slave devices, represented by green points. A comparison of the results in Figs. 9(b) with 9(a) reveals that our synthesis methods significantly reduce the occurrence of wires traveling back over neighboring areas by

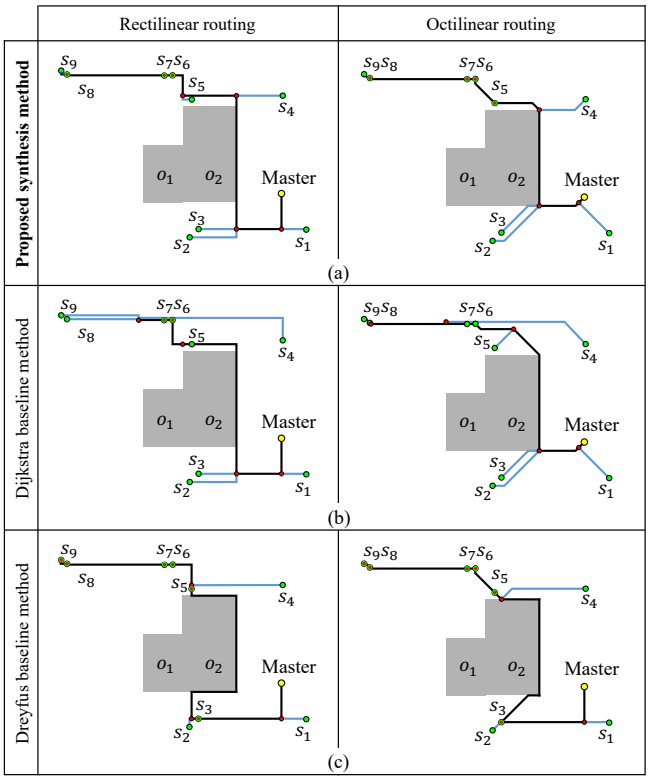


Fig. 9. Results of case 10 using different methods, where trunk segments are shown in black, branch segments in blue, with the master device represented by yellow circles, slave devices by green circles, and virtual points by red circles.

optimizing the device connection order with a global view on total wire length. Additionally, comparing Figs. 9(c) with 9(a) demonstrates that the detour mechanism in our synthesis methods effectively prevents wires from unnecessarily routing around obstacles. Consequently, the proposed rectilinear and octilinear synthesis methods achieve wire length reductions of 24.33%, 18.91%, 9.02%, and 8.64% compared to the Dijkstra and Dreyfus baseline methods in case 10, respectively.

C. Rectilinear vs. octilinear synthesis methods

We compare the wire lengths obtained by our two proposed synthesis methods. The results demonstrate that our octilinear synthesis method can significantly reduce the required wire length by up to 12.85%, compared to the rectilinear synthesis method. This improvement contributes to lower manufacturing costs. Notably, all test cases, except for case 11, were solved within a minute, reflecting the high efficiency of our methods. Generally, an increase in the number of devices leads to a substantial rise in the permutations of device connection orders, consequently extending the time needed to find the optimal solution. This also explains the unavailability of the result for case 11 using the Dreyfus baseline method even after 24 hours of runtime. Still, the proposed octilinear synthesis method consistently demonstrates a brief runtime across all cases, benefiting from the optimized device connection order established in the rectilinear synthesis method. Specifically, by retaining the predefined assignment of slave devices to each virtual point from the rectilinear synthesis, synthesizing the bus topology in the octilinear routing plane is significantly simplified. Thus, the octilinear synthesis method focuses on routing to achieve the minimum total wire length while considering obstacles, thus improving computational efficiency.

VI. Conclusions

This work proposed methods to synthesize the bus topology with minimum total wire length using rectilinear and octilinear routings on planes containing obstacles. Our rectilinear synthesis method decides the optimal device connection order and simultaneously routes the wires, minimizing the total wire length in the rectilinear routing plane. Our octilinear synthesis method builds on the rectilinear synthesis method to further shorten the wire length using octilinear architecture. Experimental results confirm that the proposed methods can efficiently synthesize bus topologies with significantly decreased wire lengths.

References

[1] F. Leens, "An introduction to i2c and spi protocols," *IEEE Instrumentation & Measurement Magazine*, vol. 12, no. 1, pp. 8–13, 2009.

[2] C.-W. Lin, S.-Y. Chen, C.-F. Li, Y.-W. Chang, and C.-L. Yang, "Obstacle-avoiding rectilinear steiner tree construction based on spanning graphs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 643–653, 2008.

[3] X. Huang, W. Guo, G. Liu, and G. Chen, "Fh-oaos: A fast four-step heuristic for obstacle-avoiding octilinear steiner tree construction," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 21, no. 3, apr 2016.

[4] X. Huang, W. Guo, and G. Chen, "Fast obstacle-avoiding octilinear steiner minimal tree construction algorithm for vlsi design," in *Sixteenth International Symposium on Quality Electronic Design*, 2015, pp. 46–50.

[5] G. Ajwani, C. Chu, and W.-K. Mak, "Foars: Flute based obstacle-avoiding rectilinear steiner tree construction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 194–204, 2011.

[6] C.-H. Liu, S.-Y. Yuan, S.-Y. Kuo, and S.-C. Wang, "High-performance obstacle-avoiding rectilinear steiner tree construction," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 14, no. 3, 2009.

[7] J. Long, H. Zhou, and S. O. Memik, "Eboarst: An efficient edge-based obstacle-avoiding rectilinear steiner tree construction algorithm," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2169–2182, 2008.

[8] T. Huang, L. Li, and E. F. Y. Young, "On the construction of optimal obstacle-avoiding rectilinear steiner minimum trees," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 5, pp. 718–731, 2011.

[9] T. Huang and E. F. Y. Young, "An exact algorithm for the construction of rectilinear steiner minimum trees among complex obstacles," in *Proceedings of the 48th Design Automation Conference*, ser. DAC '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 164–169.

[10] T. T. Jing, Z. Feng, Y. Hu, X. L. Hong, X. D. Hu, and G. Y. Yan, "λ-oat: λ-geometry obstacle-avoiding tree construction with $o(n \log n)$ complexity," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2073–2079, 2007.

[11] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization: Second Edition*. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[12] G. O. LLC, *Gurobi Optimizer Reference Manual*, 2022. [Online]. Available: https://www.gurobi.com

[13] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[14] S. E. Dreyfus and R. A. Wagner, "The steiner problem in graphs," *Networks*, vol. 1, no. 3, pp. 195–207, 1971.

[15] J. Ganley and J. Cohoon, "Routing a multi-terminal critical net: Steiner tree construction in the presence of obstacles," in *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, vol. 1, 1994, pp. 113–116 vol.1.