Obstacle-Aware Length-Matching Routing for Any-Direction Traces in Printed Circuit Board

Weijie Fang[1], Longkun Guo[1]*, Jiawei Lin[1], Silu Xiong[2], Huan He[2], Jiacen Xu[3] and Jianli Chen[4]
[1]Fuzhou University, [2]Hangzhou Huawei Enterprises Telecommunication Technologies Co., Ltd,
[3]Shanghai LEDA Technology Co., Ltd, [4]Fudan University
*Corresponding author: longkun.guo@gmail.com

ABSTRACT

Emerging applications in Printed Circuit Board (PCB) routing impose new challenges on automatic length matching, including adaptability for any-direction traces with their original routing preserved for interactiveness. The challenges can be addressed through two orthogonal stages: assign non-overlapping routing regions to each trace and meander the traces within their regions to reach the target length. In this paper, mainly focusing on the meandering stage, we propose an obstacle-aware detailed routing approach to optimize the utilization of available space and achieve length matching while maintaining the original routing of traces. Furthermore, our approach incorporating the proposed Multi-Scale Dynamic Time Warping (MSDTW) method can also handle differential pairs against common decoupled problems. Experimental results demonstrate that our approach has effective length-matching routing ability and compares favorably to previous approaches under more complicated constraints.

CCS CONCEPTS

• **Hardware → PCB design and layout**; • **Mathematics of computing → Combinatorial optimization**.

KEYWORDS

Length Matching, Any-Direction Trace, Obstacle-Aware Routing, Dynamic Programming, Differential Pair

ACM Reference Format:
Weijie Fang[1], Longkun Guo[1]*, Jiawei Lin[1], Silu Xiong[2], Huan He[2], Jiacen Xu[3] and Jianli Chen[4]. 2024. Obstacle-Aware Length-Matching Routing for Any-Direction Traces in Printed Circuit Board. In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655915

1 INTRODUCTION

Several protocols in Printed Circuit Board (PCB) designs demand some parallel signals in the same group to arrive at their destination simultaneously. Mismatching the arrival timing of these signals may result in a critical clock skew that harms the stability and functionality of designs, which derives length-matching techniques to minimize the difference between their propagation delay by matching the length of their traces. Many current length-matching tools still require manual assistance to resolve detail routing in obstacle-dense regions. Meanwhile, length-matching approaches for high-speed designs need to fit routing in any direction, including but not limited to the traditional routing in 90° or 135°, which is illustrated in Fig. 1.

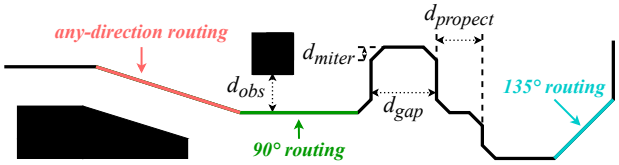


Figure 1: Illustration of routing in various directions and the primary distances restricted in DRC. Solid polygons in the figure denote obstacles.

Kubo et al. [9] approached length matching with a symmetric slant grid interconnect scheme. Ozdal et al. [14] and Yan [18] investigated the length matching of single-layer bus routing. Kohira et al. [7, 8] and Yan et al. [19] were concerned with length matching with obstacle environments. Ozdal et al. [11], Yan et al. [20], Tseng et al. [17], and Kito et al. [6] introduced Lagrangian Relaxation, Bounded-Sliceline Grid, Integer Linear Programming, and Simulated Annealing to length matching, respectively. Lee et al. [10] studied simultaneous escape routing considering length matching of differential pairs. Zhang et al. [22] addressed better wire shape resemblance for length matching. Nakatani et al. [13] attempted to reduce the total length of traces before length matching. Zhang et al. [21] proposed a length-matching method for disordered pins using R-flip and C-flip. Chang et al. [2] address a length-matching RDL routing problem for area-I/O flip-chip designs. Sato et al. [16] presented a pattern generator for set-pair routing by selecting and connecting pin-pairs. Cheng et al. [4] achieved length matching for obstacle-aware on-track bus routing [3, 5].

Most existing works on automatic length matching may override the original routing of traces or assume traces are routed in up to eight directions. However, many high-speed PCBs nowadays are designed with traces routed in any direction, and it is usually specified to route such any-direction traces. Leading industrial commercial tools, like Allegro PCB Designer [1], specially implement a route offset function to generate such traces. Hence, the routing of these traces is hoped to be preserved after length matching because users do not prefer a result that confuses their recognition and interaction, or seriously corrupts their previous specific routing during the Computer-Aided Design (CAD) process. Besides, a trace usually passes different Design Rule Areas (DRA), demanding the length matching approaches to consider multiple Design Rules Checking (DRC). These gaps motivate length-matching techniques to keep up with the industrial standard.

The length-matching process can be divided into two orthogonal stages: assigning non-overlapping regions for original traces and

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06…$15.00
https://doi.org/10.1145/3649329.3655915

meandering each trace within its own region. This paper mainly focuses on the second stage to achieve automatic length matching while preserving their original properties as much as possible. Meanwhile, applications of length matching frequently involve differential pairs. A differential pair is commonly regarded as a wide single-ended trace during length matching, but this scheme meets many difficulties in practice, especially when the differential pair is not strictly coupled. This paper proposed the Multi-Scale Dynamic Time Warping (MSDTW) method to help tackle these difficulties. Fig. 2 illustrates the algorithmic flow of our approach.

Our contributions are summarized as follows:

- To the best of our knowledge, this paper is the first length-matching work with respect to arbitrary routing directions, and it supports obstacle-aware routing and multiple DRCs.
- The presented length-matching method combines greedy, Dynamic Programming (DP), and computational geometry. Compared with existing approaches, *it routes more flexibly without following fixed tracks or pre-defined modes relying on space regularity*, achieving length matching of any-direction traces concerning original routing.
- We proposed the MSDTW method to facilitate the length matching among differential pairs. *It can convert a differential pair into a median trace against several issues in coupling*, and the median trace after length matching can be simply restored to the differential pair. The applications of MSDTW are not limited to our length-matching method.

The remainder of this paper is organized as follows. Section 2 introduces the preliminaries of our works. Section 3 briefly discusses our region assignment. Section 4 presents our detailed meandering and MSDTW method. Section 5 demonstrates the experimental results. Section 6 concludes this paper.

2 PROBLEM FORMULATION

Length matching is also known as delay tuning because it generally works on the traces already routed in a PCB. In this paper, we focus on length-matching routing on any-direction traces to meet the emerging industrial requirement from high-speed PCBs nowadays. The clarification of primary DRC about length matching shown in Fig. 1 is listed as follows:

d_{gap}: restricts the distance between traces to prevent interference between traveling signals.
d_{obs}: restricts the distance between a trace and an obstacle.
$d_{protect}$: restricts the minimum length to prevent the occurrence of extremely short trace segments.
d_{miter}: configures the corners mitered for convex patterns. In practice, any rotation of a right angle or an acute angle will be mitered by obtuse angles.

Some other important concepts in this paper are given as follows:

Trace: trace of a signal consisting of connected segments in PCB layout, also indicated by net or wire.
Any-direction: the traces that can be routed not only in 90° or 135° are called any-direction traces.
l_{target}**:** target length of a matching group that all traces in it need to match, no less than the length of the longest trace.
Routable Area: the union of non-overlapping routing regions assigned to a trace, represented as some irregular polygons.
Obstacle: a polygon that the trace cannot pass, converted into a part of the routable area in this paper.

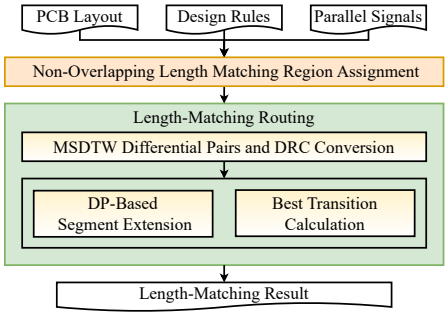


Figure 2: Overview of our length-matching approach.

Therefore, the problem in this paper can be formulated as follows:

Any-direction length-matching problem: Given a PCB layout, design rules, and matching groups. For each matching group with l_{target}, extend each trace in the group utilizing the space of its routable area to make its length equal l_{target}, while preserving its original specific routing as much as possible.

For digestibility, we use the convex pattern with corners at a right angle in the remaining discussion to omit tedious details of geometry computation.

3 REGION ASSIGNMENT

Based on the relation between length and space revealed in [20], we need only assign sufficient regions for each trace to hold feasible length-matching routing. Similar problems have been discussed in many works, such as [20] using Quadratic Programming and [11] using Lagrangian relaxation. For the sake of better fitting our specific requirement, we divide the design according to its layout to compose several regions and consider the following constraints:

(1) Neighbor Validity: A region can only be assigned to its neighbor traces:

$$x_{ij} = 0, \text{ region } i \text{ is not the neighbor of trace } j \tag{1}$$

where x_{ij} denotes the space that region i assign to trace j.

(2) Feasibility: Any assignment of a region space must be positive and bounded by its capacity:

$$\sum_j x_{ij} \leqslant Cap_i,\ x_{ij} \geqslant 0,\ \forall i, j \tag{2}$$

where Cap_i denotes the space capacity of region i.

(3) Sufficiency: A trace must receive sufficient space from its neighbor regions:

$$\sum_i x_{ij} \geqslant Req_j,\ \forall i, j \tag{3}$$

where Req_j denotes the required space for trace j.

Here, we employ an LP problem to solve this assignment:

$$\begin{array}{rl} \underline{\text{Assignment Problem:}} & \\ \text{find:} & \text{feasible } x_{ij} \\ \text{satisfying:} & \text{neighbor validity constraint (1)} \\ & \text{feasibility constraint (2)} \\ & \text{sufficiency constraint (3)} \end{array} \tag{4}$$

This assignment scheme ensures the preserved original routing is contained in the routable area for the following stages. Some techniques of existing works can help to figure out a better routing if the LP is infeasible [2]. Limited by the length of the article, we will not discuss them in detail.

4 LENGTH-MATCHING ROUTING

This section details our length-matching routing algorithm.

4.1 DP-Based Segment Extension

To increase the length of a trace l_{trace} to l_{target}, our routing method inserts convex patterns perpendicular to its segment, called the extension of segments. This extension is held by computational geometry so that it fits any-direction routing. Each segment is extended as much as possible, and a segment after extension is replaced by several new component segments for further extension if needed. The extension will be conducted iteratively until l_{trace} is within the error tolerance, resulting in patterns similar to the combination of Accordion and Trombone.

Our method optimizes the extension of a segment using a DP algorithm. First of all, we discretize the segment into points using a configurable step length l_{disc}. In this paper, we define $d(a, b)$ denotes the Euclidean distance between points a and b. Formally, a segment AB with node points A and B will be discretized to set $U = \{u \mid u \text{ is on segment } AB\}$ of n different points, where $u_1 = A$, $u_n = B$ and $\forall i \in [2, n-1], d(u_{i-1}, u_i) = l_{disc}$. We may slightly increase d_{gap} and $d_{protect}$ or adjust l_{disc} to make the former divisible by the latter.

Based on the discretization above, we define $dp[i][dir]$ to denote the best extension result, in which the patterns are inserted within the previous i points, and the last inserted pattern is in the dir direction of the segment. Without loss of generality, we define the clockwise direction as the positive direction and the counterclockwise direction as the negative direction, represented by "1" and "−1", respectively. We set the initial state as:

$$dp[1][-1] = dp[1][1] = 0 \tag{5}$$

During the transition, we initialize a new state by:

$$dp[i][dir] = dp[i-1][dir],\ i \in [2, n] \tag{6}$$

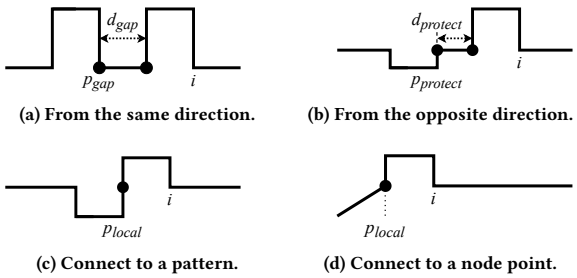
For each pattern with w-steps width, whose feet are located in u_{i-w} and u_i, respectively, we calculate its maximum valid height h and try to attach it with the best available previous states to obtain a better result of the current state. Thereby, we can obtain a simplified state transition equation as follows:

$$dp[i][dir] = \max(dp[i][dir], dp[i-w][\pm dir] + h),\ i \in [2, n] \tag{7}$$

According to the DRC introduced in Section 2, the actual available previous states rather than $dp[i-w][\pm dir]$ shall be detailed. If the current state is transited from the one with the same direction, it must keep at least d_{gap} away from the foot of any possibly existing pattern. And for the opposite direction, the distance is at least $d_{protect}$. Besides, a particular valid state is that the pattern connects to a previous one or a node of the extended segment. The above four cases are illustrated in Fig. 3. Hence, we have the meaningful states for transition as follows:

$$dp[i-w][\pm dir] = \max \begin{bmatrix} dp[p_{gap}][dir] \\ dp[p_{protect}][-dir] \\ dp[p_{local}][-dir] \end{bmatrix},\ i \in [2, n]$$

$$where \quad \begin{cases} p_{gap} & = i - w - \frac{d_{gap}}{l_{disc}} \\ p_{protect} & = i - w - \frac{d_{protect}}{l_{disc}} \\ p_{local} & = i - w, \text{ need extra condition} \end{cases} \tag{8}$$



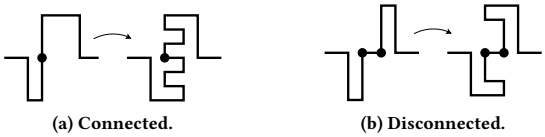
(a) From the same direction. (b) From the opposite direction.

(c) Connect to a pattern. (d) Connect to a node point.

Figure 3: Four kinds of valid state transitions.

(a) i meets the extra condition. (b) i does not meet the extra condition.

Figure 4: Illustration of different candidate states with the same value. (a) and (b) contribute the same value to $dp[i][dir]$.



(a) Connected. (b) Disconnected.

Figure 5: Illustration of why the patterns are hoped to be connected. In this figure, the former case can provide the capacity of an extra pattern.

Certainly, none of these positions can be less than 1. Otherwise, the corresponding state is invalid. After the DP, we choose the best state between $dp[n][1]$ and $dp[n][-1]$. Although $dp[i][dir]$ only maintains the value of best extension results, each state has only one transition edge, so we can easily backtrack the transition path and restore the patterns of the best solution.

If multiple previous states have the same value, choosing any of them will not change the final result of $dp[i][dir]$. However, they may affect the following states because the transition from p_{local} needs an extra condition illustrated in Fig. 4. Besides, without compromising the current result, the state in which two patterns are connecting will likely bring capacity for extra patterns, as illustrated in Fig. 5. We retain these states with higher priority during the predecessor state transitions.

Our DP-based extension is presented in Alg .1.

4.2 Maximum Transition Gain

To determine the maximum transition gain of the proposed DP, we need to calculate the maximum valid height h of pattern C built on u_{i-w}, u_i. Notably, that h is valid does not guarantee any height $h' \le h$ if the pattern routes around obstacles, so monotonicity-based methods like binary search cannot be adopted. Hence, we first create C with the height equal to the remaining extension requirement and then shrink h until all violations of DRC are eliminated. Multiple DRAs are separated into independent routable areas and handled independently.

Here, we give the concept of UnReachable Area (URA): The URA of a segment is a rectangle whose border is half of d_{gap} away from the segment, and the URA of a pattern is the union of three segments' URAs, as shown in Fig. 6a. Thereby, we convert DRC to checking the intersection between the polygons that stand for URAs or the routable area. Even though the URAs of the previous

Algorithm 1: DP-based segment extension.

Input: *trace* before length matching; target length l_{target}.
Output: *trace* after length matching.

maintaining unexpanded segments using queue Q
while $l_{trace} \neq l_{target}$ *and* $|Q| \neq 0$ **do**
  pop segment *seg* from Q, discretize *seg* into point set U
  $dp[1][1] = dp[1][-1] = 0$
  for $i \leftarrow 2$ *to* n, dir *in* $\{-1, 1\}$ **do**
    $dp[i][dir] = dp[i-1][dir]$
    if $i = n$ **or** $d(u_i, u_n) \geqslant d_{protect}$ **then**
      for $w \leftarrow 1$ *to* i **do**
        calculate the maximum h based on width w
        calculate $dp[i][dir]$ considering priority
      end
    end
  end
  $dir_{max} = \arg\max_{dir} dp[n][dir]$
  if $dp[n][dir_{max}] > 0$ **then**
    $l_{trace} = l_{trace} + dp[n][dir_{max}]$
    restore the patterns of the best result
    push the new segments replacing *seg* into Q
  end
end

patterns in DP are uncertain, they can be ignored since the valid transitions have considered DRC. For convenience, we call edges AB and CD the "sides" and BC the "hat" of URA, and the area below line AD need not be checked because there certainly only lies the URA of the original segment. During shrinking, we obtain h by updating the height of the outer border h_{ob} of URA:

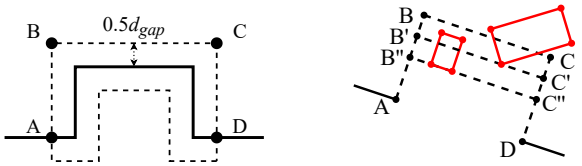
$$h = \max(0, h_{ob} - \frac{d_{gap}}{2}) \tag{9}$$

The shrinking of h begins with eliminating the violation of DRC for its outer border. In the first place, we shrink according to the intersection of "sides" with other polygons. In this paper, we define the distance between a point p and the extended segment *seg* as $d(seg, p)$, and the distance between a point set P and *seg* as $d(seg, P) = \min_{p \in P} d(seg, p)$. Then the shrinking is calculated as:

$$h_{ob}^0 = \min(d(A, B), d(seg, P_{inters})) \tag{10}$$

where P_{inters} is the set of all intersection points mentioned above.

This way, the violations of DRC caused by the outer border are reduced to the intersection with "hat". Shrinking according to "hat" may be iterative, as illustrated in Fig. 6b, because the shrunk border may lead to new intersections with polygons. Observing that there must be at least one node point of each intersected polygon inside the outer border since the intersection with "sides" has been done, we derive an efficient method to solve this problem by checking whether a polygon has node points both inside and outside the outer border. We classify all node points of polygon k into set $Poly_k$. Defining $P_{inside} = \{p \mid p \text{ is inside the outer border}\}$ and the subset of $Poly_k$ named $Poly_k^{in} = \{p \mid p \in Poly_k \cap P_{inside}\}$, then conditions $|Poly_k^{in}| = 0$ and $|Poly_k^{in}| = |Poly_k|$ can indicate the whole polygon k is outside or inside the outer border, respectively.



(a) URA. (b) "Hat" intersection.

Figure 6: Illustration of URA and "hat" intersection. (a) Dash segments represent the URA of a pattern. (b) For conciseness, we only draw the outer border and its shrinking.

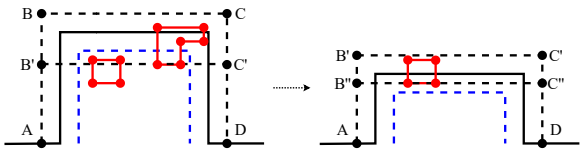


Figure 7: Illustration of the shrinking with the inner border. Black dash segments represent the outer border and its shrinking. Blue dash segments represent the inner border.

So h_{ob} is updated using:

$$h_{ob}^{i+1} = \min\left(h_{ob}^i, \min_{k:\ 0<|Poly_k^{in}|<|Poly_k|} d(seg, Poly_k^{in})\right) \tag{11}$$

We build a segment tree to reduce the checking range to $\{p \mid x_p \in [x_A, x_C], y_p \in [y_D, y_B]\}$, defining x_p and y_p as the coordinates of p, and figure out the initial P_{inside}. After each iteration, P_{inside} is reduced to $P_{inside} \setminus Poly_k$ for all k with $0 < |Poly_k^{in}| < |Poly_k|$, and the iterating ends when P_{inside} is no longer reduced.

For surrounded obstacles, the last work is to check whether polygons inside the outer border intersect the inner border. Similarly, we define the subset of $Poly_k$ named $Poly_k^{out} = \{p \mid p \text{ is outside the inner border, } p \in Poly_k\}$. When $Poly_k^{out} \cap P_{inside} \neq \emptyset$, h_{ob} must be shrunk to avoid the whole polygon k:

$$h_{ob}^{i+1} = \min\left(h_{ob}^i, \min_{k:\ |Poly_k^{out}|>0} d(seg, Poly_k)\right) \tag{12}$$

This shrinking is also iterative, as shown in Fig. 7, and P_{inside} is reduced to $P_{inside} \setminus Poly_k$ for all k with $|Poly_k^{out}| > 0$ here.

Compared to some meandering methods [2, 20], this shrinking is a switchable function to build patterns that route around obstacles if a better state transition is met. Compared to some re-route obstacle-aware methods [5, 18], our algorithm just slightly changes the topology to prevent overriding the original routing.

(a) An example of decoupled differential pairs in the real world.

(b) Original differential pair (white) and median trace (green). (c) Median trace (white) and restored differential pair (green).

Figure 8: Example of the functionality of MSDTW.

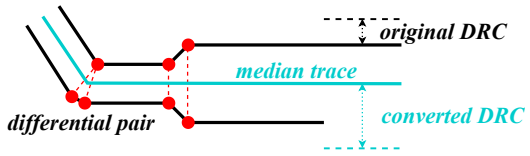
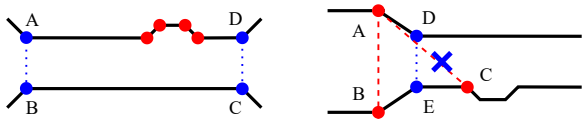


Figure 9: Matching of a differential pair using MSDTW. Red dash lines in the figure indicate matched pairs.



(a) Matching stops if no nodes remain in $trace_P$ or $trace_N$.

(b) Matching across split differential subpairs is forbidden.

Figure 10: Illustration of MSDTW handles multiple DRAs. Red dash lines and blue dot lines in the figure indicate matched pairs in this round and the last round, respectively.

4.3 Multi-Scale Dynamic Time Warping

A common method of length matching for differential pairs is to regard each differential pair as a wide single-ended trace bounded by its sub-traces. However, this trick still meets many problems in practice because the sub-traces may frequently not be coupled, caused by not strictly parallel, tiny patterns, or different passed DRAs. Fig. 8a provides a particular example from a real-world design. To tackle these issues, we proposed MSDTW based on the Dynamic Time Warping (DTW) algorithm [12, 15], which converts a differential pair and its DRC into a median single-ended trace.

We employ DTW to obtain the optimal node matching between sub-traces except the preserved breakout part by minimizing the total cost of all matched pairs. It allows multiple nodes to match the same node while promising every node is matched, as shown in Fig. 9. Let $trace_P$ and $trace_N$ respectively symbolize the sub-traces in a differential pair. Without loss of generality, we define $C[i][j]$ as the minimum cost of matching the previous i nodes of $trace_P$ and the previous j nodes of $trace_N$. Initializing state $C[0][0] = 0$, the state transition equation is:

$$C[i][j] = \min \begin{bmatrix} C[i-1][j] \\ C[i][j-1] \\ C[i-1][j-1] \end{bmatrix} + d(i,j), \begin{cases} i \in [1, I] \\ j \in [1, J] \end{cases} \tag{13}$$

where $d(i, j)$ is the distance between the ith node of $trace_P$ and the jth node of $trace_N$ that denotes the matching cost, and I and J denote the number of nodes in the two sub-traces, respectively. The matched pairs are restored by backtracking the state transition from $C[I][J]$ to $C[0][0]$. We connect every pair of matched nodes, thus making all nodes compose several connected components. Define V_C denotes the set of all nodes in a connected component, and then $V_C^P = \{v \mid v \in V_C, v \text{ is in } trace_P\}$, $V_C^N = \{v \mid v \in V_C, v \text{ is in } trace_N\}$. Each V_C will generate a median point p_m as:

$$p_m = \overline{\left\{\overline{V_C^P}, \overline{V_C^N}\right\}} \tag{14}$$

where $\overline{X}$ is the point with the average coordinate of all points in X. These median points compose the converted median trace.

Matched pairs involving tiny patterns usually cause an undesirable offset of median points. To avoid them, we define $cost_i$ as the matching cost of the matched pair $pair_i$ and r as the distance rule of the differential pair, and then drop $pair_i$ if $cost_i > \sqrt{2}r$ considering the rotation angle of a trace must be obtuse, i.e., $cost_i > \sqrt{2}r$ indicates $pair_i$ is a matched pair involving nodes of tiny patterns. To fit different passes DRAs, our MSDTW sequentially matches the nodes with increasing r, so-called multi-scale. Defining R as the set of all involved distance rules, the pairs are matched and dropped by enumerating $r \in R$ in increasing order.

To prevent the matching of a DRA with greater r from interfering with the nearby DRA with smaller r, we split the differential pair into differential subpairs consisting of the retained matched pairs of each round and forbid the nodes belonging to different subpairs to be matched with each other. For a subpair, if none of its remaining nodes are in $trace_P$ or $trace_N$, these nodes shall belong to tiny patterns and no longer participate in the successive matching. Fig. 10 helps to illustrate these cases. This strategy holds because tiny patterns are used for length matching between subpairs and shall only appear on either $trace_P$ or $trace_N$, or else two tiny patterns can be reduced by each other.

The MSDTW method is presented in Alg. 2. After length matching, we restore the differential pairs and compensate tiny patterns to sub-traces if needed. If the original traces do not violate DRC, the restored traces will not as well. Fig. 8b and Fig. 8c display the merged median trace and restored differential pair of a case from the design in Fig. 8a, respectively.

Algorithm 2: Multi-Scale Dynamic Time Warping.

Input: Differential pair $df_{original}$; Rule set R.
Output: Set of all matched pairs M.

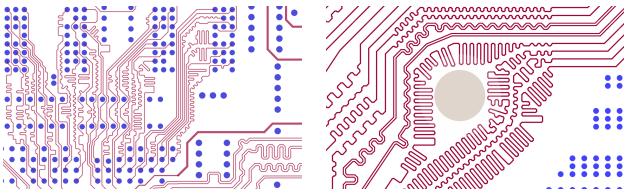
```
M = ∅, set of differential subpairs S = {df_original}
foreach r in R do
    foreach df_i in S do
        calculate the current matched pairs set M_i
        foreach pair_j in M_i do
            if cost_j > √2 r then
                M_i = M_i \ {pair_j}
            end
        end
        M = M ∪ M_i
        split df_i into S_split using matched pairs in M_i
        foreach df_split in S_split do
            if no nodes in trace_P or trace_N then
                S_split = S_split \ {df_split}
            end
        end
        S = S_split ∪ S \ {df_i}
    end
end
```

5 EXPERIMENTS

Our length-matching tool is developed using C++ programming language. All experiments were performed with an AMD Ryzen 7840H 3.80 GHz CPU and 16GB memory. The benchmark is derived from the sample design provided by Allegro PCB Designer [1]. We removed the tuning of preset matching groups and arranged 5 cases to evaluate the performance of automatic length matching. Since we knew no published works concerned with the same objective and constraints of this problem, we compared our approach with the SOTA Auto-interactive Delay Tune (AiDT) function of Allegro

Table 1: Length-matching performance Compared with Allegro [1] AiDT

case	$\frac{l_{target}}{d_{gap}}$	group size	trace type	spacing	Max. error (%)			Avg. error (%)			runtime (s)	
					Initial	Allegro	Ours	Initial	Allegro	Ours	Allegro	Ours
1	205.88	8	single-ended	dense	37.38	33.52	**3.02**	19.02	14.23	**1.30**	**0.92**	6.87
2	199.02	8	single-ended	dense	35.99	28.06	**3.93**	19.41	11.04	**1.39**	**0.78**	3.98
3	187.25	8	single-ended	dense	35.91	20.91	**3.51**	20.06	8.66	**1.37**	**0.81**	5.27
4	186.27	8	single-ended	dense	30.99	22.25	**5.46**	17.22	9.85	**1.83**	**0.72**	2.86
5	217.32	4	differential	sparse	26.55	**10.21**	10.3	15.18	5.14	**3.32**	5.07	**3.22**



(a) Our experimental result. **(b) Any direction functionality.**

Figure 11: Displays of our length-matching results.

PCB Designer, which is also applied in real-world industrial designs against the problem in this paper.

Table 1 gives case statistics and experimental results. The comparison is measured by the maximum and average matching error of each matching group and the runtime of the algorithm. As shown in the table, our approach addresses more precise length matching in the first 4 cases while compromising on runtime, which resulted from our DP-based extension having more flexible space utilization in spacing-dense environments and inevitable time complexity. Nevertheless, the presented runtime is still reasonable for human tolerance, and on the contrary, manually refining the delay tuning is usually pretty time-consuming. For case 5, our approach has similar precision to AiDT when spacing is sparse and an advantage in runtime. As the algorithm of AiDT is not public, we can only infer that it owes to the better efficiency of our MSDTW method.

Fig. 11a shows our experimental result, and Fig. 11b gives a dummy routing on a modified private commercial design that shows our functionality with any-direction traces.

6 CONCLUSION

In this paper, we present an automatic length-matching approach concerning any-direction traces in high-speed designs. Unlike previous works, we employ DP and computational geometry to meander the trace, which aims to preserve the specified original routing during length matching by maintaining direction and limiting overriding changes in topology. Meanwhile, it achieves more flexible obstacle-aware space utilization with reasonable runtime. Moreover, we proposed a method named MSDTW that converts differential pairs during length matching to tackle the issues against decoupling and multiple DRAs. The experimental illustration and the comparison with commercial tools demonstrate the effectiveness and functionality of our approach.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 12271098).

REFERENCES

[1] Cadence. [n. d.]. *Allegro PCB Designer*. https://www.cadence.com/en_US/home/tools/pcb-design-and-analysis/pcb-layout/allegro-pcb-designer.html

[2] Yu-Hsuan Chang et al. 2019. Obstacle-Aware Group-Based Length-Matching Routing for Pre-Assignment Area-I/O Flip-Chip Designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8.

[3] Jingsong Chen et al. 2019. MARCH: MAze Routing Under a Concurrent and Hierarchical Scheme for Buses *(DAC '19)*. 6 pages.

[4] Yi-Hao Cheng et al. 2020. Obstacle-Avoiding Length-Matching Bus Routing Considering Nonuniform Track Resources. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 8 (2020), 1881–1892.

[5] Chen-Hao Hsu et al. 2019. A DAG-Based Algorithm for Obstacle-Aware Topology-Matching On-Track Bus Routing. In *Proceedings of the 56th Annual Design Automation Conference 2019 (DAC '19)*. 6 pages.

[6] Nobutaka Kito et al. 2018. A Fast Wire-Routing Method and an Automatic Layout Tool for RSFQ Digital Circuits Considering Wire-Length Matching. *IEEE Transactions on Applied Superconductivity* 28, 4 (2018), 1–5.

[7] Yukihide Kohira, Suguru Suehiro, and Atsushi Takahashi. 2009. A fast longer path algorithm for routing grid with obstacles using biconnectivity based length upper bound. *IEICE transactions on fundamentals of electronics, communications and computer sciences* 92, 12 (2009), 2971–2978.

[8] Yukihide Kohira and Atsushi Takahashi. 2010. CAFE router: A fast connectivity aware multiple nets routing algorithm for routing grid with obstacles. *IEICE transactions on fundamentals of electronics, communications and computer sciences* 93, 12 (2010), 2380–2388.

[9] Yukiko Kubo, Hiroshi Miyashita, Yoji Kajitani, and Kazuyuki Tateishi. 2004. Equidistance Routing in High-Speed VLSI Layout Design. In *Proceedings of the 14th ACM Great Lakes Symposium on VLSI (GLSVLSI '04)*. 220–223.

[10] Yen-Jung Lee, Hung-Ming Chen, and Ching-Yu Chin. 2013. On simultaneous escape routing of length matching differential signalings. In *2013 IEEE Electrical Design of Advanced Packaging Systems Symposium (EDAPS)*. 177–180.

[11] Muhammet Mustafa Ozdal et al. 2006. A Length-Matching Routing Algorithm for High-Performance Printed Circuit Boards. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 12 (2006), 2784–2794.

[12] C. Myers, L. Rabiner, and A. Rosenberg. 1980. Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 28, 6 (1980), 623–635.

[13] Yuta Nakatani and Atsushi Takahashi. 2015. A length matching routing algorithm for set-pair routing problem. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 98, 12 (2015), 2565–2571.

[14] M.M. Ozdal and M.D.F. Wong. 2006. Algorithmic study of single-layer bus routing for high-speed boards. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25, 3 (2006), 490–503.

[15] H. Sakoe and S. Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49.

[16] Shimpei Sato, Kano Akagi, and Atsushi Takahashi. 2020. A Fast Length Matching Routing Pattern Generation Method for Set-Pair Routing Problem Using Selective Pin-Pair Connections. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 103, 9 (2020), 1037–1044.

[17] Tsun-Ming Tseng, Bing Li, Tsung-Yi Ho, and Ulf Schlichtmann. 2015. ILP-Based Alleviation of Dense Meander Segments With Prioritized Shifting and Progressive Fixing in PCB Routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 6 (2015), 1000–1013.

[18] Jin-Tai Yan. 2022. Single-Layer Obstacle-Aware Multiple-Bus Routing Considering Simultaneous Escape Length. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 12, 6 (2022), 902–915.

[19] Jin-Tai Yan et al. 2011. Obstacle-Aware Length-Matching Bus Routing. In *Proceedings of the 2011 International Symposium on Physical Design (ISPD '11)*. 61–68.

[20] Tan Yan and Martin D. F. Wong. 2009. BSG-Route: A Length-Constrained Routing Scheme for General Planar Topology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 11 (2009), 1679–1690.

[21] Ran Zhang, Tieyuan Pan, Li Zhu, and Takahiro Watanabe. 2015. A length matching routing method for disordered pins in PCB design. In *The 20th Asia and South Pacific Design Automation Conference*. 402–407.

[22] Ran Zhang and Takahiro Watanabe. 2013. A parallel routing method for fixed pins using virtual boundary. In *IEEE 2013 Tencon - Spring*. 99–103.