Wot the L: Analysis of Real versus Random Placed Nets, and Implications for Steiner Tree Heuristics *

Andrew B. Kahng[1,2], Christopher Moyes[1], Sriram Venkatesh[1] and Lutong Wang[2]
[1]CSE and [2]ECE Departments, UC San Diego, La Jolla, CA 92093
{abk, cmoyes, srvenkat, luw002}@ucsd.edu

ABSTRACT

The NP-hard Rectilinear Steiner Minimum Tree (RSMT) problem has been studied in the VLSI physical design literature for well over three decades. Fast estimators of RSMT cost (which reflects routed wirelength) are a required ingredient of modern physical planning and global placement methods. Constructive estimators build heuristic RSMTs whose costs are used as wirelength estimates; notably, these include FLUTE [8]. Analytic and lookup table-based estimators include the methods of Cheng [7] and Caldwell et al. [3]; the latter is based on both the number of points and the aspect ratio of the pointset in the RSMT instance. We observe that the physical design literature has numerous evaluations of RSMT heuristics and estimators on random pointsets, and that the relative merits of heuristics and estimators have been determined based on this use of random pointsets. In this paper, we show that a pointset attribute which we call *L-ness* highlights the difference between real placements and random placements of net pins. We explain why placements of netlists in practice result in pointsets with much higher L-ness than random pointsets, and we confirm this difference empirically for both academic and commercial placement tools. We further present an improved lookup table-based RSMT cost estimator that includes an L-ness parameter. Last, we illustrate how differences between Steiner tree heuristics can change depending on whether real or random pointsets are used in the evaluation.

ACM Reference Format:
Andrew B. Kahng[1,2], Christopher Moyes[1], Sriram Venkatesh[1] and Lutong Wang[2]. 2018. Wot the L: Analysis of Real versus Random Placed Nets, and Implications for Steiner Tree Heuristics . In *Proceedings of 2018 International Symposium on Physical Design (ISPD'18).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3177540.3178238

1 INTRODUCTION

VLSI global placement seeks to minimize routed wirelength (WL) along with timing path delays, dynamic power and other design metrics, subject to the constraint that placeable instances do not overlap. Because signal nets are routed as Steiner trees, their routed wirelengths are ideally modeled as the costs of respective *Rectilinear Steiner Minimum Trees* (RSMTs) over pin locations. Since the RSMT problem is NP-hard, placement tools typically minimize the sum over all nets of the bounding box half-perimeter of pin locations – i.e., the *half-perimeter wirelength* (HPWL) objective [13]. An important element of efficient placer implementation is the fast estimation of RSMT costs, e.g., by weighting HPWL according to a lookup table of scaling factors [3][7].

*Merriam-Webster https://www.merriam-webster.com/dictionary/wot defines "wot" as the old English verb meaning "know (of)".

ISPD'18, March 25–28, 2018, Monterey, CA, USA

ACM ISBN 978-1-4503-5626-8/18/03…$15.00
https://doi.org/10.1145/3177540.3178238

Our present work focuses on the qualitative difference between *real* pointsets corresponding to pin locations of placed nets, and *random* pointsets that have often been used to characterize the performance and relative merits of RSMT heuristics or RSMT cost estimators. As discussed below, placement tools will tend to move the pins of a net up against two adjacent edges of the net bounding box, as shown in Figure 4 below. This phenomenon is due to the HPWL objective in conjunction with each placeable instance having multiple incident nets. By contrast, with random pointsets, all point locations inside the pointset bounding box are equiprobable. We define the *L-ness* of a placed net's pin locations to capture how close they are to two adjacent edges of the net bounding box:

Definition: Given a pointset P, the *bounding box* of P is the minimum-area rectangle that contains all points of P; we use $B(P)$ to denote the bounding box area. The *L-ness* of P is measured as $R(P)/B(P)$, where $R(P)$ is the area of the *largest empty (isothetic) rectangle* that (i) is contained in the bounding box of P, (ii) contains one corner of the bounding box of P, and (iii) contains no points in P.

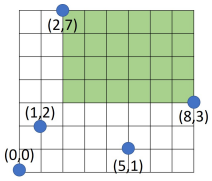

Figure 1: Illustration of largest empty (isothetic, i.e., with axis-parallel edges) rectangle. The L-ness of this 5-pin pointset is 24/56.

High $R(P)/B(P)$ ratio corresponds to large L-ness. If $B(P) = 0$, then we consider the L-ness of P to be 1. Figure 1 shows a pointset with $R(P)/B(P) = \frac{24}{56}$.

1.1 Motivation: Non-uniformity of Net Pin Placements

As a motivating study, we first confirm the non-uniform distribution of real placed pointsets (i.e., pins of signal nets). We use the leon3mp [15] and theia [20] design blocks mapped to a 28nm LP foundry enablement, with place-and-route performed using Cadence Innovus Implementation System version 15.2 [19]. Two types of placements are studied: *pseudo-1D* and *2D*. To obtain a pseudo-1D placement, we create a floorplan with width/height *aspect ratio* (AR) of 10:1 following the methodology described in [5]. We collect the point (pin) location distribution for each net along the x-axis, within a normalized range of 0 (left boundary of each given net) to 1 (right boundary of each given net). We categorize nets into three types – L, R, and O, defined as follows. A net n is of type L if, for each cell c of the net, no fanin/fanout net of c has a pin to the right of the rightmost pin of the net n, and at least one has a pin to the left of the leftmost pin of the net n. A net n is of Type R if for each cell c of the net, no fanin/fanout of c has a pin to the left of the bounding box (BBox) of net n, and at least one has a pin to the rightmost pin of net n. A net n is otherwise of type O. For example, in Figure 2, nets A and D are of type L; net B is of type R; and net C is of type O.

Figure 3 shows results of this empirical study on the designs mentioned above. We see that a "real" placement tool will clearly push cells (pins) of a type L net (respectively, a type R net) toward the left (respectively, right) boundary. There are virtually no cells in the middle, and only a few cells are pushed to the opposite boundary. From our study, we believe that there are two explanations for cells occurring at the opposite boundary: (i) we plot cell locations according to the center of the cell, which has error with respect to exact pin locations; and (ii) nets with short x-span can exhibit this behavior since the placer does not see a significant wirelength penalty for doing this. For a type O net, the cell distribution still shows preference to the bounding box boundary, indicating non-uniform distribution.

We have also performed the above experiment for 2D placements with floorplan height = width, i.e., aspect ratio = 1. In the y direction, "bottom" and "top" are respectively equivalenced to "left" and "right" in the x direction. Then, we sum up the pointset distribution in both directions. The results look similar to Figure 3. We see a very strong deviation from the uniform distribution that is seen with random pointsets.[1]

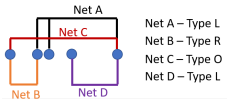

Figure 2: Illustration of L, R, and O types of nets.

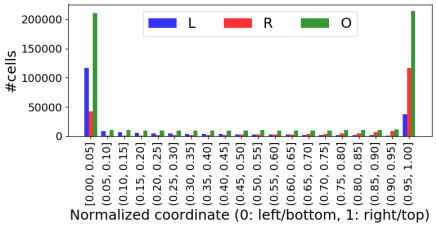

Figure 3: Empirical results from pseudo-1D placements.

Figure 4: Pins of a net from an industrial placer, clustered towards the left and bottom edges of the bounding box.

1.2 Related Works

Previous works have estimated RSMT cost based on characteristics of placed pin locations. Caldwell et al. [3] demonstrate how RSMT cost depends on both the cardinality and the aspect ratio of a pointset. This improves upon the earlier work of Cheng [7], which estimates RSMT cost based only on the pointset cardinality. Quite notably, Cheng [7] appears to point out the concept of L-ness in real placed pin locations when discussing the modeling of routing resource demand. However, this observation does not seem to have been followed up in the RSMT estimation or placement literatures.[2]

[1] While this motivating study uses the Cadence Innovus placer, Section III below shows similar non-uniformity across multiple academic and commercial placers' outputs.

[2] Section 3 of [7] states, "The high wiring probability at the top and bottom boundaries comes from the following two facts: (1) the probability of having two pins located at the same boundary is high because of bounding box. (2) when finding an optimal Steiner tree, either a left-L or a right-L is used to reduce the wire length of a minimum spanning tree."

In the computational geometry literature, Chazelle et al. [6] present an $O(n \log^3 n)$ divide-and-conquer algorithm which calculates the area of the largest empty (isothetic) rectangle in a set of n points. By using a semi-dynamic heap, Naamad et al. [14] calculate the largest empty rectangle in a set of n points in $O(s \log n)$ time where s is the number of possible empty rectangles.

With regard to RSMT heuristic constructions, Chu and Wong [8] give a well-known $O(n \log n)$ RSMT heuristic, FLUTE, which is the most accurate of the RSMT heuristics that we study in Section 4.1. The Prim-Dijkstra heuristic of Alpert et al. [1] "blends" classic minimum spanning tree and shortest-paths tree constructions using a weighting factor α to obtain a heuristic "shallow-light" spanning tree. Below, in our experimental studies, we augment the Prim-Dijkstra construction with the edge-overlapping method of Ho et al. [11] to obtain a heuristic RSMT from the Prim-Dijkstra spanning construction.

1.3 Contributions and Outline of This Paper

The main contributions of this paper are as follows.

- We propose a formal definition of *L-ness* of a pointset in the Manhattan plane.
- We empirically characterize a qualitatively significant difference in L-ness between real placed net pins and random pointsets. As seen in Section 3.1, real placed pointsets have significantly higher L-ness than random pointsets.
- We describe a pointset generator which can be used to generate more realistic pointsets with prescribed L-ness distribution. This can be used to assess RSMT heuristics and cost estimators with randomly generated pointsets that match AR and $R(P)/B(P)$ distributions (as well as RSMT costs - see Subsection 4.2) of real placed pointsets.
- We give a new lookup table-based RSMT cost estimator which improves over the method of [3] by adding L-ness as a parameter. Our implementation of this lookup table gives a non-dominated (speed, accuracy) option for RSMT cost estimation.

In the following, Section 2 presents notation and analyses of L-ness in planar pointsets. Section 3 describes empirical characterizations of real placed pointsets, contrasted with random pointsets. Section 4 discusses the impact of L-ness on the relative performance of various RSMT heuristics. Section 5 presents a new lookup table-based RSMT cost estimate that improves upon [3] by adding an L-ness dimension. Section 6 summarizes our results and concludes the paper.

2 PRELIMINARIES

In this section, we first give notations and facts used in this work. We then analyze different properties of pointsets, and discuss the relationship of L-ness to other pointset characteristics. Last, we provide methods to generate realistic pointsets, and an algorithm to compute $R(P)$ in $\Theta(n \log n)$ time.

2.1 Notations

Notations that we use in this paper are summarized in Table 1. The layout region is assumed to have lower-left corner $(0, 0)$ and upper-right corner (H, W). A *random* p-pin pointset consists of p points chosen randomly from a uniform distribution in the $H \times W$ layout region. As noted above, the *bounding box* of pointset P is the minimal isothetic (axis-parallel) rectangle that contains all points of P. The *half-perimeter* of a given bounding box is half the perimeter of the bounding box. For example, the half-perimeter of the bounding box in Figure 1 is 15, and its AR is $\frac{8}{7}$.

Our discussion furthermore assumes that points of a random pointset are in *general position*, i.e., all x-coordinates and all y-coordinates are distinct. To validate this assumption, we extract

Table 1: Notations.

Notation	Meaning
p	net degree (# pins of a signal net) ($p \geq 2$)
P	a net (pointset), $P = (x_1, y_1), \ldots, (x_p, y_p)$
$B(P)$	the area of the minimum bounding box of P
$R(P)$	the area of the largest empty rectangle of P
$RSMT(P)$	the rectilinear Steiner minimum tree over P
(H, W)	chip dimensions, i.e., height and width of the chip
AR	aspect ratio (W/H) of the bounding box
$R(P)/B(P)$	L-ness, the ratio of $R(P)$ divided by $B(P)$

Table 2: Probability that any two points in a pointset share the same x- or y-coordinate.

p	ICC/Innovus	Capo	ePlace
2	9.88%	7.48%	7.65%
3	10.98%	7.90%	7.46%
4	7.57%	6.84%	6.01%
5	8.03%	7.99%	6.32%
6	7.50%	7.69%	7.49%
7	7.45%	8.27%	5.28%
8	7.68%	4.86%	3.90%
9	8.48%	6.13%	4.37%
10	7.46%	4.35%	3.81%
11	6.78%	4.51%	3.59%
12	6.18%	4.27%	3.50%

placed pin coordinates from the placements of seven design blocks, including leon3mp and netcard from [15]; theia, jpeg, aes and mpeg from [20]; and an ARM Cortex A53 [18]. The placements are obtained using two leading commercial place-and-route tools, Cadence Innovus 15.2 [19] and Synopsys ICC L-2016.03-SP4 [22] with foundry enablements at 28nm and 16nm. We also extract the placements of the DAC-2012 benchmark suite [17] from two well-known academic placers, i.e., Capo [4] and ePlace [12]. These placements are collectively referred to as *real pointsets* in the rest of this paper. Table 2 shows that the percentage of any two points in a real pointset sharing the same x-coordinate or y-coordinate is less than 11%, supporting our assumption of distinct x- and y-coordinates.

We define *L-ness* of P as the ratio of $R(P)$ to $B(P)$, where $R(P)$ is the area of the *largest empty (isothetic) rectangle* that (i) is contained in the bounding box of P, (ii) contains one corner of the bounding box of P, and (iii) contains no points in P. High $R(P)/B(P)$ ratio corresponds to large L-ness.

2.2 Probability that k Points Define the Bounding Box

A bounding box can be represented by four extreme coordinate values, i.e., x_{min}, x_{max}, y_{min} and y_{max}. Given unique x- and y-coordinates, at most four points of a pointset can define the pointset's bounding box, where each of the points provides exactly one of the four extreme coordinates. Further, at least two points define the bounding box, where each of the points contains one extreme x-coordinate and one extreme y-coordinate. We use $Pr(p, k)$ to denote the probability that the bounding box of a pointset P (having cardinality p) is defined by k points ($k \in \{2, 3, 4\}$).

For $k = 2$, assume that points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ define the bounding box. Then, x_1 (resp. y_1) must be x_{min} or x_{max} (resp. y_{min} or y_{max}) out of the p x-coordinates (resp. y-coordinates), and x_2 (resp. y_2) can only be the other extreme x (resp. y)-coordinate out of $p - 1$ x-coordinates (resp. y-coordinates). Thus, Equation (1) gives the probability $Pr(p, 2)$.

For $k = 4$, each of four points can define only one extreme coordinate of the bounding box. Assume that these points are $p_1 = (x_{min}, \neg(y_{min} \vee y_{max}))$, $p_2 = (x_{max}, \neg(y_{min} \vee y_{max}))$, $p_3 = (\neg(x_{min} \vee x_{max}), y_{min})$, and $p_4 = (\neg(x_{min} \vee x_{max}), y_{max})$. Then, the probability that four points define the bounding box is as given in Equation (3). Supplemental equations using chain rules to derive probabilities are given in Equations (4)–(7). For example, $Pr(p_1)$ is computed by finding the probability that a point has the minimum x-coordinate and not an extreme y-coordinate. These probabilities are each computed separately and are then multiplied together since they are independent. The remaining probabilities in Equations (4)–(7) are computed in a similar fashion.

Table 3: $Pr(p, k)$ for $p \in [3, 10]$ and $k \in \{2, 3, 4\}$.

p	$Pr(p, k = 2)$	$Pr(p, k = 3)$	$Pr(p, k = 4)$
3	0.3333	0.6667	0.0000
4	0.1667	0.6667	0.1667
5	0.1000	0.6000	0.3000
6	0.0667	0.5333	0.4000
7	0.0476	0.4762	0.4762
8	0.0357	0.4286	0.5357
9	0.0278	0.3889	0.5833
10	0.0222	0.3556	0.6222

$$Pr(p, k = 2) = \binom{p}{2}\left(\frac{2}{p}\right)^2\left(\frac{1}{p-1}\right)^2 \tag{1}$$

$$Pr(p, k = 3) = 1 - Pr(p, k = 2) - Pr(p, k = 4) \tag{2}$$

$$\begin{aligned} Pr(p, k = 4) &= 4!\binom{p}{4} Pr(p_1\ p_2\ p_3\ p_4) \\ &= 4!\binom{p}{4} Pr(p_1) Pr(p_2|p_1) Pr(p_3|p_1 p_2) Pr(p_4|p_1 p_2 p_3) \\ &= \binom{p}{4}\left(\frac{4!}{(p^2)(p-1)^2}\right) \end{aligned} \tag{3}$$

For the remaining case of $k = 3$, we can calculate the probability $Pr(p, 3)$ using Equation (2). Table 3 provides a lookup table for $Pr(p, k)$ for $k \in \{2, 3, 4\}$ and $p \in [3, 10]$.

$$Pr(p_1) = \left(\frac{1}{p}\right)\left(\frac{p-2}{p}\right) \tag{4}$$

$$Pr(p_2|p_1) = \left(\frac{1}{p-1}\right)\left(\frac{p-3}{p-1}\right) \tag{5}$$

$$Pr(p_3|p_1 \cdot p_2) = \left(\frac{p-2}{p-2}\right)\left(\frac{1}{p-2}\right) \tag{6}$$

$$Pr(p_4|p_1 \cdot p_2 \cdot p_3) = \left(\frac{p-3}{p-3}\right)\left(\frac{1}{p-3}\right) \tag{7}$$

We use this probability in Algorithm 2 to determine the parameter k. Subsequently, we use Algorithm 2 to create the $real'$ pointsets, as described in Section 4.

2.3 Independence of AR and $R(P)/B(P)$

To justify the experimental methodology used below, we prove the intuitive claim that $R(P)/B(P)$ is preserved when a 2D pointset P is "stretched" (by scaling of x-coordinates and of y-coordinates) into a pointset P' that has a different aspect ratio. We refer to this property of pointsets as *independence* of AR and $R(P)/B(P)$. We show this independence of AR and $R(P)/B(P)$ by (i) exhibiting an appropriate 1-1 correspondence between pointsets P with bounding box area $B(P)$ and pointsets P' with bounding box area $B(P')$, then (ii) showing that the ratio $R(P)/B(P) = R(P')/B(P')$ is preserved by this correspondence. In Subsection 3.1, we measure $R(P)/B(P)$ without considering the effect of AR on L-ness of pointsets. Hence, we prove the independence of $R(P)/B(P)$ with AR below.

Fact 1. Scaling of x- and y-coordinates provides a (bidirectional) 1-1 mapping between pointsets P having unit square bounding box $B(P)$, and pointsets P', with $|P| = |P'|$ and bounding box B' having an arbitrary aspect ratio.

Fact 1 is established as follows. Denote the width and height of B' are w and h, respectively. We obtain pointset P' from P by scaling x- and y-coordinates of points in P by w and h, respectively. As a result, the x- and y-coordinates of the bounding box edges of P' are also scaled by w and h. The inverse scaling procedure can be applied to restore any such P' to the original P. The scaling of coordinates thus provides a 1-1 correspondence between pointsets having the same cardinality but different bounding box ARs.

Next, we say that the point (x_i, y_i) in P *corresponds* to a point (x'_i, y'_i) in P' if $(x'_i, y'_i) = (w \cdot x_i, h \cdot y_i)$. A bounding box-edge of P analogously *corresponds* to a scaled bounding box-edge of P'. For example, Figure 5(b) shows a pointset P' obtained by scaling P (in Figure 5(a)) by $(w, h) = (w, 1)$. From our definitions, we say that the point (x_i, y_i) in P corresponds to the point (x'_i, y'_i) in P', and the edge $x = x_{sp}$ of P corresponds to the edge $x = x'_{sp}$ of P'.

The following Fact 2 holds for pointsets (i) P with its largest empty rectangle defined by two edges of the bounding box, $x = x_{sp}$ and $y = y_{sp}$, and two points, (x_1, y_1) and (x_2, y_2); and (ii) P', which is created by scaling the x- and y-coordinates of points in P by w and h, respectively.

Fact 2. Given P and P', the edges and points that define $R(P)/B(P)$ correspond to edges and points that define $R(P')/B(P')$.

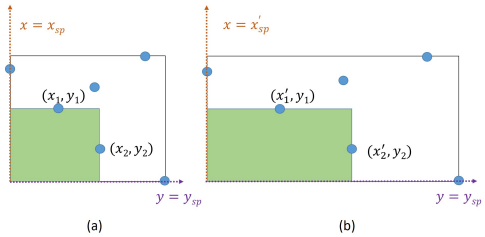

Figure 5: The pointsets (a) P and (b) P' with the largest empty rectangle colored green.

Fact 2 is established as follows. P contains p points, i.e., $P = \{(x_1, y_1), (x_2, y_2), \cdots, (x_p, y_p)\}$. Without loss of generality, we assume that the bounding box of P has $AR = 1$, and we only scale points in pointset P in the x-direction by w (i.e., $w > 0$, $h = 1$) to obtain P'. P' also contains p points, $P' = \{(x'_1, y_1), (x'_2, y_2), \cdots, (x'_p, y_p)\}$, where $(x'_j, y_j) = (w \cdot x_j, y_j)$ for $k \in [1, p]$. The following treats the case illustrated in Figure 5, namely, the case with the empty rectangle at the lower-left corner of the bounding box, i.e., $x_{sp} < x_1 < x_2$, and $y_{sp} < y_2 < y_1$. The other three cases are similarly analyzed. $R(P)$ is defined as

$$R(P) = (x_2 - x_{sp}) \cdot (y_1 - y_{sp}) \tag{8}$$

Assume toward a contradiction that the edges $x = w \cdot x_{sp}$, $y = y_{sp}$, and the points (x'_1, y_1) and (x'_2, y_2) do not form the largest empty rectangle of P'. Then, there must exist an empty rectangle of P' such that

$$R(P') > w \cdot R(P) \tag{9}$$

Suppose that the largest empty rectangle of P' is defined by the edges $w \cdot x_{sp}$ and y_{sp} and two points (x'_m, y_m) and (x'_n, y_n), with $\{n, m\} \neq \{1, 2\}$ and $x'_{sp} < x'_m < x'_n$ and $y_{sp} < y_n < y_m$. Then, $R(P')$ is calculated as

$$R(P') = (x'_n - x'_{sp}) \cdot (y_m - y_{sp}) \tag{10}$$

$$= (w \cdot x_n - w \cdot x_{sp}) \cdot (y_m - y_{sp}) \tag{11}$$

$$= w \cdot (x_n - x_{sp}) \cdot (y_m - y_{sp}) \tag{12}$$

According to the definition of $R(P)$, $(x_n - x_{sp}) \cdot (y_m - y_{sp}) \leq R(P)$. Therefore,

$$R(P') \leq w \cdot R(P) \tag{13}$$

which contradicts Equation (9). This establishes Fact 2.

2.4 Efficient Calculation of $R(P)$

We now describe an efficient method to obtain $R(P)$. Each of the four corners of the bounding box may be the intersection of the two edges that form $R(P)$. For simplicity, we only describe our algorithm for the corner (x_{min}, y_{min}). The final result can be obtained by invoking the algorithm on each corner of the bounding box of P with small modifications and then returning the largest value, at the cost of a constant-factor complexity increase.

Algorithm 1 describes the calculation of $R(P)$. The algorithm begins with pointset P sorted in ascending order of x-coordinates. Lines 1 and 2 perform initializations. In Lines 3 – 8, we check whether the current point has a smaller y-coordinate than the stored value of y_0. If so, the lower-left corner will form an empty rectangle, and we update the maximum known rectangle area. The same procedure is followed to compute the largest empty rectangle at the remaining corners. We step through the sorted list of points, check if each pair of points forms an empty rectangle, and if so, update the maximum known rectangle area. The time complexity of the algorithm is lower-bounded by the implied sorting step, which gives a $\Theta(p \log p)$ time complexity.

Algorithm 1 CalcRP (Assuming lower-left corner is selected).

Input: P with x-coordinates in ascending order
Output: $R(P)$
1: $R(P) = 0$
2: $y_0 = y_1$
3: **for** $i = 2$ to p **do**
4: **if** $y_i \leq y_0$ **then**
5: $R(P) \leftarrow \max(R(P), (x_i - x_{min}) \cdot (y_0 - y_{min}))$
6: $y_0 \leftarrow y_i$
7: **end if**
8: **end for**
9: **return** $R(P)$

3 REAL VS. RANDOM POINTSETS

In this section, we empirically demonstrate the significant difference in L-ness between real and random pointsets. We then present a method for generating pointsets with prescribed L-ness and aspect ratio.

3.1 L-ness of Real vs. Random Pointsets

We experimentally compare the $R(P)/B(P)$ distribution of 100K random pointsets with the $R(P)/B(P)$ distribution of real (placed net pins) pointsets. Figure 6 shows the distributions of $R(P)/B(P)$ in random and real pointsets. In each plot, the x-axis denotes the $R(P)/B(P)$ ratio and the y-axis denotes the fraction of nets for each $R(P)/B(P)$ value. From the figure, we see that the placements from commercial and academic placers result in pointsets with significantly larger L-ness (i.e., larger $R(P)/B(P)$ ratio) than random pointsets.[3] We also observe that the qualitative difference from random pointsets holds across academic and commercial placers.

We believe that this large L-ness arises due to the following reasons. Given a large chip area and a relatively small bounding box (b_0) area for any net n_0, it is intuitive that the other nets incident to the cells of net n_0 have their bounding boxes outside b_0. This causes the cells to get pulled towards the boundary, and extend the boundaries of net n_0 due to multiple inter-related nets (i.e., intersecting hyperedges of the netlist). Further, low net degrees usually result in a geometrically asymmetrical cell distribution, increasing the L-ness of a particular net.

To confirm the statistical difference for $p \in [3, 12]$, we perform two tests: (i) bootstrapping the mean with a 95% confidence interval [2], and (ii) Two-Sample Kolmogorov-Smirnov (KS) Test [16]. The bootstrap test provides a 0.95 confidence interval on the average of $R(P)/B(P)$ for 10000 random pointsets. We compare the means of $R(P)/B(P)$ values for real pointsets with the 0.95 confidence interval. Figure 7 shows that the means of real pointsets do not lie within the confidence intervals of random pointsets for

[3] We have separately extracted net pin locations from an advanced processor design from a leading semiconductor company, and confirmed that the $R(P)/B(P)$ distributions follow the same trend as in Figure 6.

Table 4: D_{nm} for $p \in [3, 12]$ using ICC/Innovus, Capo and ePlace placers.

p	ICC/Innovus	Capo	ePlace
3	3.363	6.160	3.256
4	3.788	6.204	3.926
5	5.159	6.913	4.240
6	4.641	5.605	3.341
7	3.658	5.150	2.884
8	3.219	4.500	2.481
9	1.737	2.953	2.747
10	4.754	3.413	3.777
11	5.790	3.987	3.162
12	7.106	4.028	4.708

any $p \in [3, 12]$. Hence, real pointsets have a statistically significant larger $R(P)/B(P)$ compared to random pointsets.

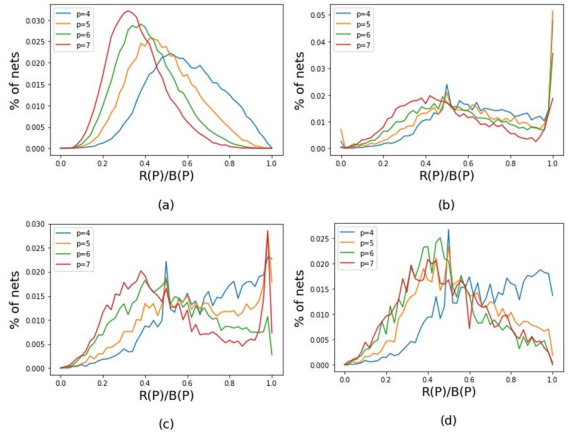

Figure 6: Distribution of $R(P)/B(P)$ from (a) random pointsets, (b) ICC [22] and Innovus [19] placements, (c) Capo [4] placements, and (d) ePlace [12] placements.

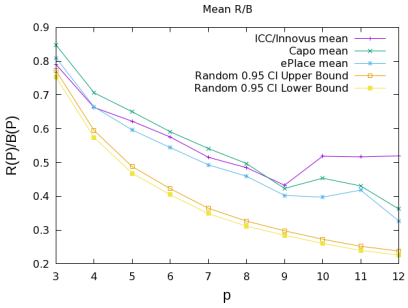

Figure 7: 95% confidence interval for $R(P)/B(P)$ in random pointsets and mean $R(P)/B(P)$ for real pointsets.

The Two-Sample Kolmogorov-Smirnov (KS) Test [2] states that for a confidence interval of 95%, we have a statistically significant difference if the KS statistic $D_{nm} > 1.36$. The KS statistic is computed as

$$D_{nm} = \sqrt{nm/(n+m)} \cdot sup|F(x) - G(x)| \quad (14)$$

where n and m are sample sizes of random and real pointsets respectively, F and G are cumulative distribution functions (CDFs) (with 100 bins of width 0.01) of $R(P)/B(P)$ values of random and real pointsets respectively. *sup* is the maximum distance between F and G for $0 \leq x \leq 1$. Table 4 shows the KS statistics. We see that $D_{nm} > 1.36$ for all random versus real pointsets, again confirming the statistically significant difference between $R(P)/B(P)$ distributions of random and real pointsets.

Algorithm 2 GenRandPointset.

Input: p, k, $RPBP$, Δ_{err}, AR
Output: P with $R(P)/B(P) \in [RPBP - \Delta_{err}, RPBP + \Delta_{err}]$
1: $P \leftarrow \emptyset$
2: $R(P)/B(P) \leftarrow 0$
3: $P \leftarrow getBBoxPts(P, k, RPBP, \Delta_{err}, AR)$
4: **while** $|P| < p$ **do**
5: $P \leftarrow AddPoint(P)$
6: **if** $calcRP(P) < RPBP - \Delta_{err}$ **then**
7: $RemovePoint(P)$
8: **end if**
9: **end while**
10: **if** $calcRP(P) \leq RPBP + \Delta_{err}$ **then**
11: **return** P
12: **else**
13: **return** -1
14: **end if**

3.2 Pointset Generation

Since random pointsets differ significantly from real placed pin locations, and since it is challenging to obtain real placement data, there is a need to generate pointsets with prescribed L-ness. Here, we present an algorithm (Algorithm 2) to generate a random pointset with prescribed $R(P)/B(P)$ (L-ness) and aspect ratio (AR). The inputs include #pins p, intended number of points k that define the bounding box (see Section 2.2), intended L-ness range $[RPBP - \Delta_{err}, RPBP + \Delta_{err}]$, and aspect ratio AR. The output is a pointset P that satisfies the L-ness range constraint.

Lines 1 and 2 perform initializations. In Line 3, we generate k points on the bounding box. Since $R(P)/B(P)$ will monotonically decrease as we add one more point to an existing pointset, the function $getBBoxPts$ comprehends the desired L-ness range and always gives k points with $R(P)/B(P) \geq RPBP - \Delta_{err}$. These k points form a bounding box with area 1M x 1M and aspect ratio AR. In Lines 5 – 9, we iteratively add one point with random location strictly inside the bounding box and check L-ness. If we do not meet the L-ness lower bound, the last added point is removed and reselected. The points are added with unique x- and y-coordinates, following the assumption of points in general position in Section 2.1. In Lines 11 and 12, we return the pointset satisfying the L-ness range constraint and discard the result otherwise. In our actual implementation, we can reuse discarded pointsets when generating for a different L-ness range – e.g., during the process of reproducing a distribution such as in Figure 6(b)-(d).

Algorithm 2 is qualitatively equivalent to randomly generating a pointset and checking if the pointset is valid, i.e., having $R(P)/B(P) \in [RPBP - \Delta_{err}, RPBP + \Delta_{err}]$. If we assume towards a contradiction that it does not, then at least one of the points we remove in Line 8 would contribute to a valid pointset. Since adding points within the bounding box cannot increase the $R(P)/B(P)$ value of a pointset, the points in this pointset cannot be part of a pointset with $R(P)/B(P)$ within the prescribed L-ness range. Hence, Algorithm 2 returns qualitatively the same pointsets as repeated generation of a pointset and checking whether the pointset has $R(P)/B(P)$ within the prescribed L-ness range. However, Algorithm 2 is much more efficient, e.g., we can produce 100K pointsets targeted to match the distribution of Figure 6(d) with $p = 7$ in 75.54 seconds with a 2.7 GHz Intel Xeon server.

4 IMPLICATIONS FOR RSMT HEURISTICS

In this section, we perform experiments to analyze the impact of L-ness on the performance (tree cost / wirelength estimation) of various RSMT heuristics. We first show how wirelength changes with different L-ness. Then, we show the RSMT cost difference between random and real pointsets.

4.1 Impact of L-ness on RSMT Heuristics

In this subsection, we study the change in wirelength with $R(P)/B(P)$ (L-ness). We generate 10K pointsets for each $R(P)/B(P)$ from 0.2

to 0.8, with a step of 0.1 and $\Delta_{err} = 0.02$. We use a fixed $B(P)$ size of 1M×1M. We evaluate the wirelength cost of four heuristics: (i) rectilinear MST implementation by Kahng et al. [21] using Prim's algorithm, (ii) Prim-Dijkstra (PD) [1] with $\alpha = 0.3$ (PD 0.3) and with $\alpha = 1.0$ (PD 1.0 constructs a shortest path tree, and is equivalent to Dijkstra's algorithm [9]), (iii) HVW [11] algorithm as a post-processing of PD 0.3 (HVW 0.3) and PD 1.0 (HVW 1.0), and (iv) FLUTE [8].

Figure 8 shows the wirelength values. The x-axis denotes the $R(P)/B(P)$ ratio and the y-axis represents the total wirelength for all 10K pointsets per each $R(P)/B(P)$. Each row of figures represents a particular value of $AR = \{1, 2, 4\}$; each column represents a value of $p = \{4, 5, 7\}$. We see that wirelength decreases as $R(P)/B(P)$ increases, indicating that we should expect lower wirelength for real pointsets which tend to have larger $R(P)/B(P)$ than random pointsets. Also, difference in wirelength among heuristics decreases with increase in $R(P)/B(P)$. Although PD 0.3 and HVW 0.3 have different optimization objectives (i.e., radius and wirelength balance) from FLUTE, wirelength follows the same trend with $R(P)/B(P)$ for all heuristics. These results suggest that assessments of cost or accuracy benefit versus runtime overhead when using these heuristics may have been misguided by the use of random pointsets in experimental studies, and that random pointsets might not give sufficient insight into the benefits of RSMT heuristics. We also observe that crossovers between heuristics tend to decline as AR increases.

4.2 RSMT Cost on Real Pointsets

Previous works [1][11] use random pointsets to verify the accuracy of RSMT heuristics. However, we reevaluate their accuracy and show their performance difference considering L-ness in real pointsets. We first generate *real' pointsets* with $R(P)/B(P)$ and AR distributions of *real pointsets* from academic and commercial placements, and show that our Algorithm 2 generates statistically similar pointsets to real placements. We then use *real' pointsets* to analyze the accuracy of heuristic WL estimation.

To generate real' pointsets, we extract the distributions of $R(P)/B(P)$ and AR from real pointsets for $p \in [3, 12]$ and use these distributions to create 10K real' pointsets for each p. We run FLUTE on all pointsets and perform the two-sample Kolmogorov-Smirnov test (KS) test on the wirelength distributions with a 95% confidence interval, using 50 bins to generate the CDFs. Table 5 shows that eight of nine values are smaller than the minimum D_{nm} value in Table 4. This shows that real' pointsets give a good representation of real pointsets for most cases. Figure 9 shows one case with a Kolmogorov-Smirnov failure. However, the probability distributions of wirelengths from real' and real pointset distributions are still similar in appearance.

Table 5: D_{nm} for wirelengths on real and real' pointsets.

p	4	5	6	7	8	9	10	11	12
D_{nm}	1.189	1.063	1.402	1.788	1.690	1.621	1.026	1.086	1.601

We use the above real' pointsets to evaluate the accuracy of each heuristics. Tables 6 and 7 report the errors of these heuristics versus FLUTE[4], and compare the differences in errors for real and random pointsets. A positive value in Table 6 means a larger wirelength is given compared to FLUTE.

Table 7 reports the percentage difference for each heuristic between real and random pointsets as $Error_{real} - Error_{random}$. A negative value means a smaller error when using real pointsets, and a positive value means a larger error, compared to using random pointsets. Hence, Tables 6 and 7 show that the errors of heuristics

[4]Errors are calculated relative to FLUTE, since FLUTE is optimal for $p \leq 9$ and introduces on average 0.16% RSMT error for $p \in [10, 17]$ [8].

Table 6: Percent error of heuristics vs. FLUTE for random and real pointsets.

p	Percent error of heuristics vs. FLUTE on random pointsets				
	HVW 0.3	RMST	PD 0.3	HVW 1.0	PD 1.0
4	1.93%	10.41%	12.35%	13.57%	44.05%
5	2.76%	11.15%	13.94%	16.14%	51.22%
6	3.38%	11.46%	14.96%	19.00%	56.94%
7	3.91%	11.52%	15.44%	21.08%	61.72%
8	4.47%	11.68%	16.02%	23.06%	65.29%
9	4.80%	11.77%	16.44%	24.72%	68.69%
10	5.07%	11.72%	16.71%	26.04%	71.06%
11	5.49%	11.80%	17.20%	27.34%	73.57%
12	5.57%	11.73%	17.24%	28.55%	75.81%
p	Percent error of heuristics vs. FLUTE on real pointsets				
	HVW 0.3	RMST	PD 0.3	HVW 1.0	PD 1.0
4	1.54%	8.96%	10.43%	15.29%	50.04%
5	1.92%	9.03%	10.90%	18.09%	58.06%
6	2.35%	9.31%	11.64%	20.37%	63.56%
7	2.99%	9.86%	12.77%	22.52%	68.10%
8	3.37%	10.19%	13.42%	24.42%	72.17%
9	4.01%	10.75%	14.58%	26.05%	74.38%
10	3.93%	10.38%	14.18%	28.00%	78.88%
11	4.19%	10.46%	14.44%	29.78%	82.00%
12	4.57%	10.60%	15.05%	30.89%	83.70%

Table 7: Difference in % error between heuristics and FLUTE for real and random pointsets.

p	Difference in % error between real and random pointsets				
	HVW 0.3	RMST	PD 0.3	HVW 1.0	PD 1.0
4	-0.39%	-1.45%	-1.92%	1.72%	5.99%
5	-0.84%	-2.12%	-3.04%	1.95%	6.84%
6	-1.03%	-2.15%	-3.32%	1.37%	6.62%
7	-0.92%	-1.66%	-2.67%	1.44%	6.38%
8	-1.10%	-1.49%	-2.60%	1.36%	6.88%
9	-0.79%	-1.02%	-1.86%	1.33%	5.69%
10	-1.14%	-1.34%	-2.53%	1.96%	7.82%
11	-1.30%	-1.34%	-2.76%	2.44%	8.43%
12	-1.00%	-1.13%	-2.19%	2.34%	7.89%

HVW 0.3, RMST and PD 0.3 are overestimated, whereas the errors of heuristics HVW 1.0 and PD 1.0 are underestimated. Since FLUTE is the most accurate of these heuristics and wirelength can only be overestimated when constructing spanning trees, all values in the tables are positive.

Table 7 can also be seen as a lookup table to improve the accuracy of existing RSMT cost estimators. For a given heuristic, more accurate wirelength values can be obtained by subtracting the errors reported in Table 7 from the wirelength of random pointsets.

5 AN IMPROVED WL ESTIMATION LOOKUP TABLE

In this section, we present a lookup table (LUT) for improved wirelength estimation. Previously, Caldwell et al. [3] constructed a lookup table indexed with p and AR. We build upon this table and add $R(P)/B(P)$ as a third parameter dimension for improved accuracy of wirelength estimation, as shown in Section 4. We use FLUTE to obtain the RSMT wirelength (see Footnote 4).

Table 8 shows a portion of our lookup table. [5] In the table, we report three sets of values for each p. The first row ($W1$) shows the FLUTE wirelength value by generating and averaging the wirelength over 1000 pointsets with $AR = \{1, 2, 4\}$. These values are equivalent to the wirelength values reported by Caldwell et al. [3]. The second row ($W2$) shows the FLUTE wirelength with specific $R(P)/B(P) = \{0.2, 0.4, 0.6, 0.8\}$ (generated using Algorithm 2), averaged over 1000 pointsets. The third row ($W3$) is the percent error between $W1$ and $W2$, i.e., $\frac{W2-W1}{W1} \cdot 100\%$. For example, with $p = 6$ and $AR = 1$, we see that the $W1$ row contains the value 2.39; this is the single value for estimated RSMT cost given by [3]. The $W2$ row contains four values, 2.71, 2.37, 2.22 and 2.10; these are our estimated RSMT costs with $R(P)/B(P)$ ratios of 0.2, 0.4, 0.6 and 0.8, respectively. The $W3$ row gives the four corresponding percentage differences between the L-ness dependent estimates and the single

[5]The entire lookup table is available at http://vlsicad.ucsd.edu/~sriram/Final_WL_estimate_LUT.htm

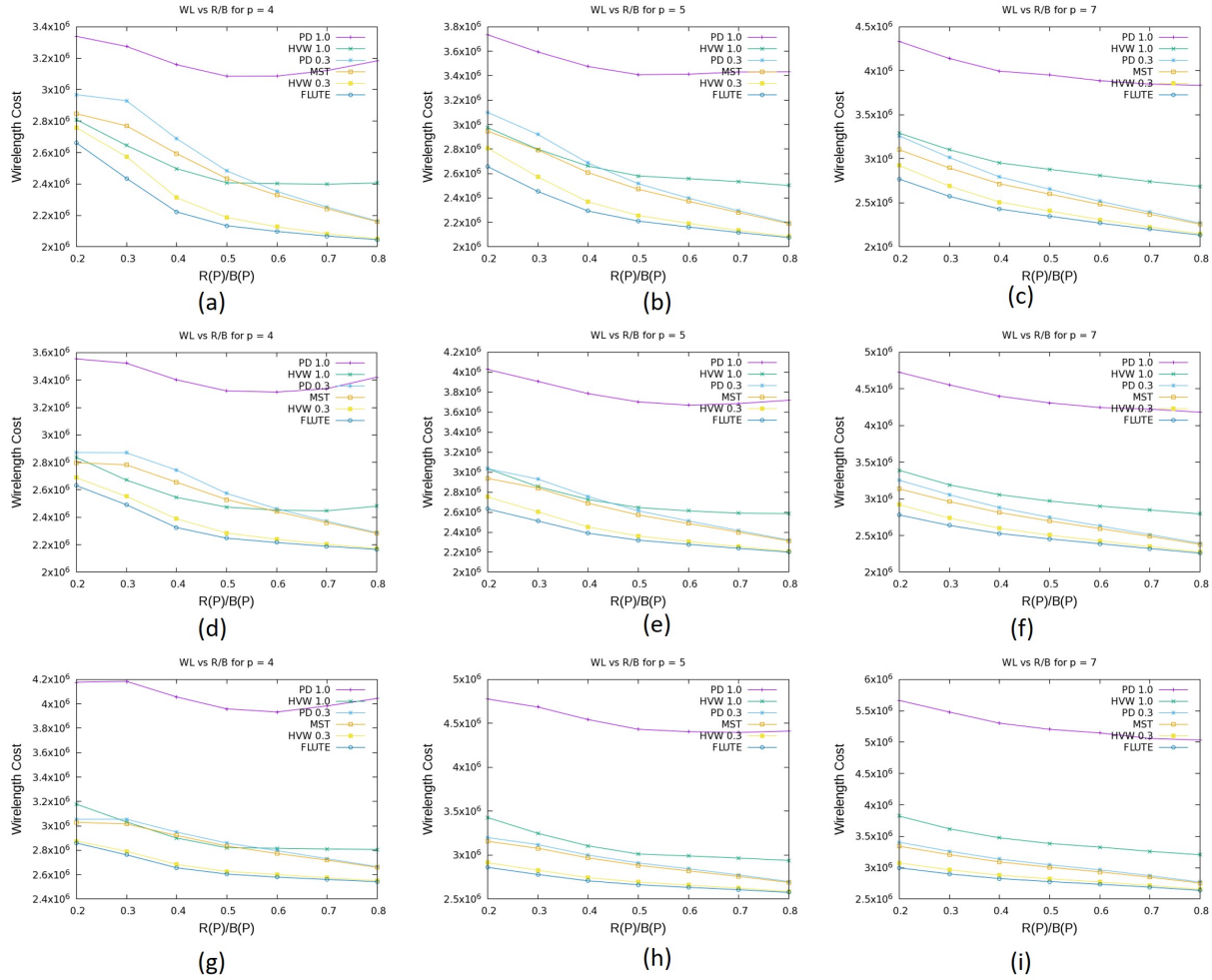

Figure 8: Change in wirelength with $R(P)/B(P)$ for nets with $AR = 1$ (a, b, c), $AR = 2$ (d, e, f) and $AR = 4$ (g, h, i) for $p \in \{4, 5, 7\}$.

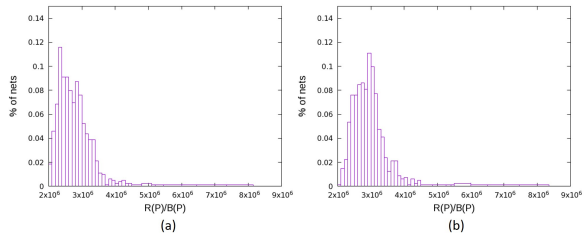

Figure 9: Wirelength distribution functions for (a) real pointsets and (b) real' pointsets for $p = 12$.

estimate of [3]. We omit estimates for $p \in [2, 3]$ since these RSMT costs are the half-perimeter wirelengths of the bounding boxes.

Runtime. We compare the runtime using different wirelength estimators: (i) FLUTE, (ii) our LUT, and (iii) rectilinear MST (RMST) implementation by Kahng et al. [21] using Prim's algorithm.[6] All algorithms are implemented using C and are executed on a 2.7 GHz Intel Xeon server with 8 threads. We evaluate using 500K real and random pointsets. Table 9 shows that our improved lookup table runs significantly faster than FLUTE for all values of $p \in [2, 12]$ and faster than RMST except for $p = 4$. We believe that this significant speedup (~10×), at the cost of small loss of accuracy of WL estimation, can be beneficial in modern-day contexts that involve very large designs, highly iterative methods, and a requirement for reduced tool turnaround times.

Accuracy. Table 10 reports the percent error, along with standard deviation and maximum error in wirelength estimates compared to FLUTE, using our lookup table (LUT), Caldwell LUT [3] and RMST implementation [21]. Percent error is calculated as $Error = \left(\frac{WL_{heur} - WL_{FLUTE}}{WL_{FLUTE}}\right) \cdot 100\%$. Our lookup table dominates that of [3] in all error metrics evaluated. However, our improved lookup table does give a higher standard deviation and maximum absolute error for higher values of p when compared to RMST. We note that the LUT errors reported in Table 10 are the averages of absolute errors, whereas RMST error is always positive. Figure 10 shows error distributions for the LUT and RMST estimators for $p = 9$.

WL estimation for pointsets with $p \in [2, 3]$ using our LUT has no error. (In our studies, 68% of the nets in a 16nm implementation of ARM Cortex A53 have $p = 2$ or $p = 3$.) The WL estimation error using our LUT for $p \in [4, 12]$ is $1 - 2\%$ lower than the error using

[6] We use an $O(n^2)$ implementation since it runs much faster than other $O(n \log n)$ algorithms for small p.

Table 8: Wirelength lookup table using aspect ratio and $R(P)/B(P)$.

AR		1				2				4			
$R(P)/B(P)$		0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8	0.2	0.4	0.6	0.8
p=4	W1	2.14				2.25				2.60			
	W2	2.66	2.23	2.10	2.04	2.63	2.32	2.22	2.17	2.86	2.66	2.58	2.54
	W3	24.37	4.24	-2.03	-4.51	16.89	3.31	-1.48	-3.77	9.93	2.20	-0.76	-2.24
p=5	W1	2.27				2.36				2.69			
	W2	2.66	2.30	2.16	2.08	2.63	2.39	2.27	2.20	2.86	2.71	2.63	2.58
	W3	17.02	1.22	-4.80	-8.57	11.41	1.25	-3.67	-6.84	6.38	0.66	-2.12	-4.19
p=6	W1	2.39				2.48				2.78			
	W2	2.71	2.37	2.22	2.10	2.70	2.45	2.34	2.23	2.93	2.77	2.69	2.61
	W3	13.58	-0.90	-7.21	-12.00	8.95	-1.13	-5.81	-10.28	5.39	-0.30	-3.17	-6.21
p=7	W1	2.52				2.59				2.87			
	W2	2.78	2.44	2.27	2.13	2.77	2.53	2.39	2.26	3.00	2.82	2.73	2.64
	W3	10.13	-3.32	-9.84	-15.42	7.12	-2.38	-7.76	-12.70	4.56	-1.58	-4.73	-8.13
p=8	W1	2.63				2.69				2.96			
	W2	2.83	2.50	2.32	2.16	2.86	2.59	2.44	2.29	3.06	2.89	2.79	2.67
	W3	7.57	-4.81	-11.62	-17.96	6.18	-3.72	-9.30	-15.01	3.50	-2.33	-5.86	-9.96
p=9	W1	2.73				2.81				3.03			
	W2	2.90	2.57	2.37	2.18	2.92	2.67	2.49	2.32	3.13	2.95	2.83	2.69
	W3	6.35	-5.72	-13.07	-20.02	3.99	-5.14	-11.30	-17.51	3.34	-2.72	-6.49	-11.14
p=10	W1	2.84				2.91				3.13			
	W2	2.98	2.65	2.42	2.21	2.99	2.73	2.54	2.34	3.19	3.01	2.88	2.72
	W3	4.77	-6.80	-14.68	-22.07	2.79	-6.24	-12.71	-19.53	1.99	-3.70	-8.10	-13.16
p=11	W1	2.95				3.00				3.22			
	W2	3.03	2.71	2.47	2.24	3.07	2.79	2.59	2.37	3.27	3.07	2.91	2.75
	W3	2.65	-8.22	-16.15	-24.15	2.38	-6.85	-13.83	-21.04	1.64	-4.52	-9.48	-14.69
p=12	W1	3.04				3.09				3.30			
	W2	3.11	2.77	2.52	2.27	3.15	2.85	2.63	2.40	3.33	3.13	2.97	2.77
	W3	2.43	-8.85	-16.97	-25.45	1.80	-7.68	-14.89	-22.47	1.04	-5.14	-10.07	-15.97
p=13	W1	3.14				3.20				3.39			
	W2	3.18	2.84	2.56	2.29	3.21	2.92	2.68	2.42	3.40	3.19	3.02	2.80
	W3	1.14	-9.56	-18.48	-26.94	0.17	-8.70	-16.25	-24.27	0.44	-5.80	-11.02	-17.28
p=14	W1	3.23				3.27				3.48			
	W2	3.24	2.90	2.62	2.32	3.28	2.98	2.73	2.44	3.47	3.25	3.05	2.83
	W3	0.45	-10.08	-19.00	-28.17	0.27	-8.82	-16.49	-25.32	-0.43	-6.64	-12.22	-18.76
p=15	W1	3.34				3.38				3.55			
	W2	3.32	2.97	2.66	2.35	3.35	3.04	2.77	2.48	3.54	3.29	3.10	2.86
	W3	-0.60	-11.01	-20.45	-29.62	-0.86	-10.06	-18.19	-26.71	-0.31	-7.20	-12.75	-19.51

Table 9: Execution time (seconds) for 0.5M pointsets with $p \in [2, 12]$.

p	Random pointsets			Real pointsets		
	FLUTE	Impr. LUT	RMST	FLUTE	Impr. LUT	RMST
2	0.051	0.003	0.012	0.050	0.003	0.012
3	0.185	0.004	0.023	0.254	0.006	0.040
4	0.229	0.047	0.045	0.295	0.065	0.050
5	0.262	0.060	0.061	0.240	0.061	0.063
6	0.299	0.077	0.095	0.328	0.116	0.109
7	0.352	0.093	0.135	0.318	0.089	0.130
8	0.431	0.111	0.173	0.368	0.104	0.171
9	0.576	0.127	0.216	0.492	0.119	0.223
10	1.192	0.146	0.258	1.248	0.134	0.259
11	1.761	0.164	0.303	1.241	0.152	0.314
12	1.804	0.184	0.378	1.607	0.166	0.360

Table 10: Error for $p \in [2, 12]$ with real pointsets.

p	Absolute Error			Std. Dev. of Abs. Error			Max. Absolute Error		
	Impr. LUT	Cald-well	RMST	Impr. LUT	Cald-well	RMST	Impr. LUT	Cald-well	RMST
3	0.00%	0.00%	6.13%	0.00%	0.00%	7.81%	0.00%	0.00%	33.31%
4	4.06%	5.61%	6.01%	3.62%	3.67%	6.49%	24.51%	28.19%	46.17%
5	4.47%	7.14%	6.20%	3.76%	4.48%	6.01%	24.94%	23.44%	42.73%
6	4.70%	8.07%	6.48%	3.95%	5.44%	5.66%	25.53%	25.24%	36.02%
7	4.93%	8.75%	6.82%	4.04%	6.41%	5.36%	28.20%	25.71%	34.60%
8	5.17%	9.85%	7.15%	4.21%	7.66%	5.14%	27.56%	31.46%	32.25%
9	5.28%	9.81%	7.73%	4.21%	7.88%	4.90%	30.95%	37.03%	32.13%
10	5.75%	11.38%	7.35%	4.69%	9.39%	4.76%	32.94%	42.16%	28.06%
11	6.00%	12.47%	7.14%	4.85%	10.37%	4.59%	37.01%	46.94%	27.46%
12	6.46%	12.62%	7.18%	5.32%	10.82%	4.58%	40.35%	52.94%	28.25%

RMST as an estimate. Thus, in terms of speed and accuracy, the new LUT provides a non-dominated wirelength estimate.[7]

6 CONCLUSION

In this paper, we have given a formal definition of the concept of *L-ness*, that is, the phenomenon that a net's pin locations within a real placement tend to be clustered towards two adjacent edges of the net's bounding box. We have provided empirical data showing the extent to which real pointsets have larger L-ness values than random pointsets. This data suggests at least the possibility that previous usage of random pointsets may have led to inaccurate assessments of RSMT heuristics and RSMT cost estimators. With this in mind, we describe a pointset generation function which can produce artificial pointsets that are similar to real placed pointsets. We furthermore present an improved lookup table for RSMT cost estimation that is sensitive to L-ness of a pointset; its implementation gives a speed-accuracy tradeoff point between FLUTE [8] and a fast rectilinear MST implementation [21].

Our ongoing and future works seek ways to exploit the L-ness attribute to achieve better estimates of routed WL or FLUTE heuristic RSMT costs – e.g., after placement and without any running of global/detailed routers. We are also exploring the direct optimization of an L-ness-aware wirelength estimate during placement. A high-fidelity wirelength predictor, congestion- and DRC-aware wirelength predictor, as well as hierarchical placement-based predictors are also of interest. Other future directions include tree topology generation considering L-ness and objectives such as timing or power, as well as comprehension of driver vs. sink pin locations.

[7] For $p \in [10, 12]$ our LUT is approximately 10 times faster than FLUTE and twice as fast as RMST.

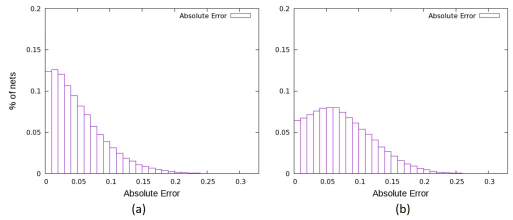

Figure 10: Error distributions with (a) lookup table and (b) RMST estimators for $p = 9$.

7 ACKNOWLEDGMENTS

We thank the authors of ePlace [12] for providing the executable that we used in our experiments. ABKGroup research is supported by NSF, Samsung, Qualcomm, NXP, Mentor Graphics and C-DEN.

REFERENCES

[1] C. J. Alpert, T. C. Hu, J. H. Huang, A. B. Kahng and D. Karger, "Prim-Dijkstra Tradeoffs for Improved Performance-driven Routing Tree Design", *IEEE TCAD* 14(7) (1995), pp. 890-896.
[2] J. Bloom and J. Orloff, *18.05 Introduction to Probability and Statistics*, Cambridge, Massachusetts Institute of Technology: MIT OpenCourseWare, 2014. https://ocw.mit.edu
[3] A. E. Caldwell, A. B. Kahng, S. Mantik, I. L. Markov and A. Zelikovsky, "On Wirelength Estimations for Row-Based Placement", *IEEE TCAD* 18(9) (1999), pp. 1265-1278.
[4] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Can Recursive Bisection Alone Produce Routable Placements", *Proc. DAC*, 2000, pp. 477-482.
[5] W.-T. J. Chan, A. B. Kahng and J. Li, "Revisiting 3DIC Benefit with Multiple Tiers", *Proc. SLIP*, 2016, pp. 6:1-6:8.
[6] B. Chazelle, R. L. Drysdale and D. T. Lee, "Computing the Largest Empty Rectangle", *SIAM J. Computing* 15(1) (1986), pp. 300-315.
[7] C. L. Cheng, "RISA: Accurate and Efficient Placement Routability Modeling", *Proc. ICCAD*, 1994, pp. 690-695.
[8] C. Chu and Y. Wong, "FLUTE: Fast Lookup Table Based Rectilinear Steiner Minimal Tree Algorithm for VLSI Design", *IEEE TCAD* 27(1) (2008), pp. 70-83.
[9] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik* 1 (1959), pp. 269-271.
[10] A. E. Dunlop and B. W. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", *IEEE TCAD* 4(1) (1985), pp. 92-98.
[11] J. Ho, G. Vijayan and C. K. Wong, "New Algorithms for the Rectilinear Steiner Tree Problem", *IEEE TCAD* 9(2) (1990), pp. 185-193.
[12] J. Lu, H. Zhuang, P. Chen, H. Chang, C.-C. Chang, Y.-C. Wong, L. Sha, D. Huang, Y. Luo, C.-C. Teng and C.-K. Cheng, "ePlace-MS: Electrostatics based Placement for Mixed-Size Circuits", *IEEE TCAD* 34(5) (2015), pp. 685-698.
[13] I. L. Markov, J. Hu and M.-C. Kim, "Progress and Challenges in VLSI Placement Research", *Proc. ICCAD*, 2015, pp. 1985-2003.
[14] A. Naamad, D. T. Lee and W. L. Hsu, "On the Maximum Empty Rectangle Problem", *Discrete Applied Mathematics* 8(3) (1984), pp. 267-277.
[15] M. M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke and C. Zhuo, "The ISPD-2012 Discrete Cell Sizing Contest and Benchmark Suite", *Proc. ISPD*, 2012, pp. 161-164.
[16] D. Panchenko, *18.650 Statistics for Applications*, Cambridge, Massachusetts Institute of Technology: MIT OpenCourseWare, 2004. https://ocw.mit.edu
[17] N. Viswanathan, C. J. Alpert, C. C. N. Sze, Z. Li and Y. Wei, "The DAC 2012 Routability-driven Placement Contest and Benchmark Suite", *Proc. DAC*, 2012, pp. 774-782.
[18] ARM Cortex A53 Processor. https://developer.arm.com/products/processors/cortex-a/cortex-a53
[19] Cadence Innovus User Guide.
[20] OpenCores: Open Source IP-Cores. http://www.opencores.org
[21] RMST-Pack: Rectilinear Minimum Spanning Tree Algorithms [Source code]. http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/RSMT/RMST
[22] Synopsys IC Compiler User Guide.