New Placement Prediction and Mitigation Techniques for Local Routing Congestion

Taraneh Taghavi
IBM, San Diego, CA
asadatt@us.ibm.com

Charles Alpert
IBM, Austin, TX
alpert@us.ibm.com

Andrew Huber
IBM,Hopewell Junction, NY
adhuber@us.ibm.com

Zhuo Li
IBM, Austin, TX
lizhuo@us.ibm.com

Gi-Joon Nam
IBM, Austin, TX
gnam@us.ibm.com

Shyam Ramji
IBM, Hopewell Junction, NY
ramji@us.ibm.com

ABSTRACT

Local routing congestion is becoming increasingly important as complex design rules make local pin access the bottleneck for modern designs and routers. Since congestion analysis based on global routing does not model these effects, routability-driven placement and physical synthesis fail to alleviate local congestion. This work models routing congestion at the placement level in order to apply local congestion mitigation. We propose a local congestion metric that computes a "routing-difficulty" score for every cell in the design library. To disperse local congestion, we apply a suite of detailed placement techniques called MILOR (**M**ovement, cell **I**nflation and **L**egalization, and **O**ptimization within a **R**ow). Experimental results show that our techniques can significantly improve routing quality on real industry designs from 65, 45, and 32 nanometer technologies.

1. INTRODUCTION

Sub-wavelength lithography and design-for-manufacturability requirements have caused the number of design rules to increase exponentially with each technology generation. At the 32nm node, sophisticated lithographic techniques require wider wire tips, more space between long running adjacent metal shapes, etc. [6, 8]. One industry routing expert [13] estimates that the number of design rules has more than tripled between the 45nm and 32nm nodes. These trends, in addition to growth in the number and variation in metal layers [1], has made the routing task much more difficult with each technology node.

A router's first phase is *global routing*. A *global router* divides the routing region into small tiles and attempts to route nets through the tiles such that no tile overflows its capacity. Recent years have seen several new innovations in global routing [3, 5, 10], motivated by the ISPD routing contests [7]. After global routing, wires must be assigned to tracks within each tile, followed by detailed routing which must connect each global route to the actual pin on the cell. To date, the design rule explosion has pushed all the routing complexity directly onto detailed routing.

Not only is global routing the first routing phase, it also serves to estimate congestion [1] for use by routability-driven optimizations like floorplanning, placement, and physical synthesis. However, basing these optimizations on feedback from global routers is problematic since they do not capture local congestion effects. By the time timing is closed and detailed routing is run, it is too late to easily mitigate local congestion. The purpose of this work is to show that local routing congestion can be modeled at the placement level.

Complex design rules require sufficient space porosity in the local metal layers above the transistor layer for the router to access the pins. However, technology scaling and highly optimized, area-efficient cell layouts have resulted in an increacing amount of resource contention for pin-access routing. In other words, making pins too small and packing "hard-to-route" cells too close together can create an unresolvable local routing hot-spot. These local congestion hot-spots are not adequately modeled by today's global routers.

A fundamental contribution of this work is a new cost function that derives a routing-difficulty score for every cell in the design library. We propose several new detailed placement techniques, designed specifically to alleviate local routing congestion. We call our algorithm MILOR, an acronym combining its three primary phases: cell **M**ovement, cell **I**nflation and **L**egalization, and **O**ptimization within a **R**ow. We ran a router from an EDA vendor on designs from industry standard IBM 65, 45, and 32 nm cell libraries. Our experiments show that on average MILOR reduces detailed routing errors by 89% and router runtime by 58%.

2. IDENTIFYING HARD-TO-ROUTE CELLS

Consider the two different cell layouts from the 45nm library in Figures 1(a) and 1(b) for the same four-input logic function. Both have a large output pin on the right side of the cell, but the layout of the four input pins is remarkably different. Cell C_1 will likely be more difficult to route than C_2 because its pins are smaller and the alignment of pins makes it more difficult to access vias. Staggered alignments of the pins in C_2 give the router more pin access choices.

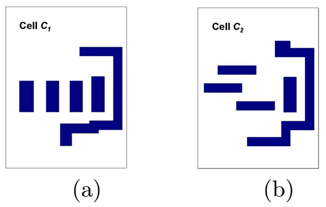

Figure 1: Pin shapes for two standard cells implementing the same logic function.

This example illustrates that pin count and area alone are insufficient to identify bad cells. We now present our cost function which consists of three components: (i) a pin-existence cost PEC, (ii) a pin-area cost PAC and (iii) a pin-resolution (or spacing) cost PRC. Let $P(C)$ be the number of pins of cell C, $p_i(C)$ be the i^{th} pin of C, and $A(p)$ be the area of pin p. The pin existence cost is simply

$$PEC(C) = P(C) \tag{1}$$

Clearly, more pins on a cell require more local routing resources. In Figure 1, $PEC(C_1) = PEC(C_2) = 5$.

Larger pins enable more potential via insertion points for pin-access. Therefore, PAC imposes a penalty for smaller pins. Let θ be the technology defined minimum cell pin width. We define the pin area cost as

$$PAC(C) = \sum_{i=1}^{P(C)} 2^{\left(2-\frac{A(p_i(C))}{\theta}\right)} \tag{2}$$

Observe that the cost goes down by a factor of two with each unit increase in pin area. This exponential decrease in penalty was chosen to reflect the routing difficulty imposed by especially small pin shapes. In practice, when a pin's cost contribution is sufficiently small because the pin is large (e.g., $\geq 4\theta$), its contribution to PAC can be ignored. In Figure 1, $PAC(C_1) = 3.058$ while $PAC(C_2) = 3.722$, so the cell on the right has higher area cost.

The final component of the cost function is the most complex. The general principle is that pins packed together limit the number of via configurations, and this is somewhat design-rule dependent. As a first order approximation, we consider each pair of pins and if their collective bounding box is small, this suggests that they have difficult pin access. We define pin resolution cost as

$$PRC(C) = \sum_{i=1}^{P(C)-1} \sum_{j=i+1}^{P(C)} 2^{\left(2-\frac{B(P_i(C),P_j(C))}{3\theta}\right)} \tag{3}$$

where $B(P_i(C), P_j(C))$ is the area of the smallest bounding box containing $P_i(C)$ and $P_j(C)$. Similar to PAC, the cost decreases exponentially as the bounding box size increases. The purpose is to have a sharp penalty for pairs of pins that are tightly packed, as in Figure 1(a). When the bounding box is large (e.g., $\geq 12\theta$), the contribution to PRC can be ignored so that only pairs of pins that limit each others pin access are counted. In Figure 1, $PRC(C_1) = 3.062$ while $PRC(C_2) = 0.935$ which demonstrates PRC can capture the effect of staggered spacing of the rightmost cell. Using the above components, we define the cell cost $K(C)$ for cell C as

$$K(C) = \alpha \cdot PEC(C) + \beta \cdot PAC(C) + \gamma \cdot PRC(C) \tag{4}$$

where α, β and γ are relative weights that can be adjusted for each technology. In our experiments, we simply use $\alpha = \beta = \gamma = 1$. Since each standard cell has a small number of pins (typically less than 10), $K(C)$ can be computed in constant time for a given cell C. We precompute $K(C)$ for each library cell and then store the result, making it cheap to score the local routability for the entire design, as explained in Section 3. Returning to our first example, $K(C_1) = 11.12$ while $K(C_2) = 9.66$ which matches our observation that C_1 should be harder to route.

3. PROBLEM FORMULATION

Given a routability cost for each cell in the library, one can aggregate the $K(C)$ scores to quantify local congestion across a design. If too many high-cost, hard-to-route cells are packed too closely together, this will cause local routing congestion. To capture this notion, we borrow the concept from global routing of dividing the routing region into equal-sized routing tiles. We generalize the cost of a tile T

$$K(T) = \sum_{\forall C_i \in T} \frac{K(C_i)}{A(T)} \tag{5}$$

where $A(T)$ is the area (in routing tracks) of a tile. Note that this cost models cell area implicitly. Larger cells are generally easier to route than smaller cells because they span more routing tracks. Normalizing by the area of the tile gives a value that is independent of tile size.

We compute local congestion metrics by sorting all the tiles from highest to lowest cost. The *max* cost tile is the $\max_{\forall T} K(T)$. One can similarly define the top 1%, 2% and 5% of tiles. The objective of the tile algorithms discussed in Section 4 is to reduce all these metrics while minimizing the wirelength degradation.

Two hard-to-route cells that are close together are potentially problematic, so we seek to separate them. The degree of separation increases the ability to route dramatically. If S is the space between two adjacent cells C_1 and C_2 we wish the cost to decrease exponentially with S so we propose

$$K(C_1, S, C_2) = \left(2^{-\frac{S}{2}}\right) \frac{K(C_1) + K(C_2)}{W(C_1) + W(C_2)} \tag{6}$$

where $W(C_i)$ is the width of cell C_i. The techniques described in Section 6 seek to shift cells within a row to optimize this metric.

4. TILE-BASED DETAILED PLACEMENT

An obvious way to reduce local congestion is to simply pick a cell in a high-cost tile and move it to a low-cost one. The challenge is to make these moves so that bins do not get overfull, wirelength does not degrade significantly, and cells do not move too far away from their original location (since that can potentially distort timing closure). The cell movement approach described in Subsection 4.1 borrows concepts from the detailed placement literature [2, 4] to find good moves and adapt them to our local congestion metrics.

At some point though, cell movement runs out of "good moves" and can only alleviate local congestion at the expense of severe wirelength degradation (which could then hurt global congestion). Thus, Subsection 4.2 applies a cell inflation and legalization technique inspired by the works of [9, 11, 12] that incrementally and locally spreads cells to relieve global congestion in problematic regions.

4.1 Single-cell Movement (M) Phase

The basic idea for the **M** cell movement phase of MILOR is to iteratively

1. Select a source tile T_s with high cost
2. Identify a cell $C \in T_s$ to move
3. Move cell C to a low-cost target tile T_t

Since $K(T_s)$ is the sum of the costs of the cells inside T_s, $K(T_s)$ is guaranteed to decrease since one of its cells is moved to a different tile. If no good move for C can be determined, another cell in T_s can be moved instead. A good move is one where (a) sufficient space in T_t exists for cell C, (b) that improves the maximum of $K(T_s)$ and $K(T_t)$, and (c) does not significantly degrade wirelength.

The challenge is to find a cell move that does not degrade, and perhaps even improves wirelength. It seems reasonable to only consider moving the highest-cost cells, but those often are anchored by neighboring cells. For that purpose, the bounding box $B(C)$ which produces the minimum wirelength solution for C is determined. Since $B(C)$ often contains the current location of C, we expand $B(C)$ by Δ on each side to increase the search space for a new location for C. A larger value of Δ corresponds to exploring a larger solution space. For each of N random locations (x, y) in the expanded $B(C)$, we compute the change in wirelength $d(C, x, y)$ of moving C to (x, y) and choose the best move.

The top 1% of highest-cost tiles are chosen to give the algorithm a chance to make several moves before calling legalization. Legalization never needs to make too many moves to resolve overlaps since only moves to tiles with sufficient space are permitted. We run detailed placement between passes to recover wirelength increased by displacing a cell C from its current location.

4.2 Inflation / Legalization (IL) Phase

When no easily-found, good cell moves remain, MILOR deploys its **IL** inflation and legalization phase. Cell inflation and legalization has been applied as an incremental technique to mitigate global routing congestion in CRISP [11] and CROP [12]. These approaches first run global routing estimation to identify tiles that are problematic, inflate cells in these tiles to produce overlaps, then remove overlaps through legalization and detailed placement techniques. A similar concept was also deployed by diffusion [9] which seeks to dissolve over congested bins through cell spreading.

In this paper, with each iteration, a subset of cells in high-cost tiles are inflated by at most 1% of the total cell area, following the convention of [11]. The cells are selected from the tiles with the highest 5% of $K(T)$ values to force spreading only for bad tiles. The cell C with the highest $K(C)$ to $W(C)$ ratio is selected to bias inflation for smaller cells.

After cells are inflated, some overlaps will be created. Like diffusion, the algorithm tries to spread cells by pushing them apart as opposed to making big cell moves, until one can legalize the design. Running a pass of wirelength-driven detailed placement is an important step since the spreading phase can severely degrade wirelength. However, this phase can hurt local congestion since it may pack cells back, but since cells remain inflated, iterations of the inflation and spreading phase improve local congestion more than detailed placement degrades it. When the design starts to get too full, spreading and legalization have a difficult time finding a solution that does not severely degrade wirelength, and this causes the algorithm to exit. A major advantage to this approach is that while it reduces $K(T)$ for the highest cost tiles, it also locally creates space around individual high-cost cells within high-cost tiles. Experiments in Section 6 demonstrate that the IL phase alone significantly improves router performance.

5. OPTIMIZATION WITHIN A ROW (OR)

Optimization within a row is the final phase of MILOR and is designed to attack local congestion at the cell-pair level. The problem we solve is finding new locations for cells within a row that preserve their relative order, while minimizing the maximum cell-pair cost. Assume we are given a row with cell coordinates 0 to L consisting of already placed and ordered cells $C_1, C_2, \cdots, C_n$. For each C_i, let $X(C_i)$ be the coordinate of the leftmost cell boundary. All widths $W(C_i)$ and locations $X(C_i)$ values are integers, a reasonable assumption since the widths of all cells in the available libraries are actually multiples of the routing track width.

Let d_i denote the displacement of cell C_i from its original location X_i (d_i can be positive, zero, or negative), making the $X(C_i) + d_i$ the new location. For two adjacent cells C_i and C_{i+1}, let $s_i(d_i, d_{i+1})$ be the space between the cells, so

$$s_i(d_i, d_{i+1}) = X(C_{i+1}) + d_{i+1} - X(C_i) - d_i - W(C_i)$$

Recall Equation 6 defines the cost between adjacent cell pairs. Extending this definition, let $K_i(d_i, d_{i+1})$ denote the local congestion cost of C_i and C_{i+1} as a function of their displacements. We have

$$K_i(d_i, d_{i+1}) = \begin{cases} K(C_i, s_i(d_i, d_{i+1}), C_{i+1}), & \text{if } s_i(d_i, d_{i+1}) \geq 0 \\ \infty, & \text{if } s_i(d_i, d_{i+1}) < 0 \end{cases} \tag{7}$$

Since K_i does not consider wirelength, we apply two techniques to bound cell movement. First, each cell is forbidden from moving more than distance M (we use $M = 10$ in practice) from its original location. Also, we seek to penalize large cell moves, so we define a new function:

$$F_i(d_i, d_{i+1}) = K_i(d_i, d_{i+1}) + \beta \cdot (d_{i+1}^2) \tag{8}$$

where β is a constant (we use $\beta = 0.01$) that reflects the degree of penalty for cell movement. The cost F_{max} for the entire row is the maximum of each pair of cells in the row:

$$F_{max} = \max_{i=1,\cdots,n-1} F_i(d_i, d_{i+1}) \tag{9}$$

A legal placement for the first cell requires (1) $X(C_1) + d_1 \geq 0$, and similarly, a legal placement for the last cell requires (2) $X(C_n) + d_n + W(C_n) \leq L$. Legally placing cells $C_2, \ldots, C_{n-1}$ requires (3) $X(C_i) + d_i \geq X(C_{i-1}) + d_{i-1} + W(C_{i-1})$. Note that one can also consider C_1 and/or C_n to be fixed cells, in which case we also require $d_1 = 0$ and $d_n = 0$.

Row Placement for Local Congestion Problem: Given a legal placement of cells $C_1, \ldots, C_n$ with widths $W(C_i)$, locations $X(C_i)$, cost function $F_i(d_i, d_{i+1})$, and a max-displacement constraint M, find cell displacement values d_i with $|d_i| \leq M$ such that the new cell placement obeys constraints (1), (2), and (3) and F_{max} is minimized.

In the next section, we use a dynamic programming formulation to derive an optimal solution to this problem.

5.1 Dynamic Programming

Each cell C_i has $2M+1$ possible locations between $X(C_i) - M$ and $X(C_i) + M$. Our approach processes cells starting from C_1 and explores cell pair locations for (C_1, C_2), followed by (C_2, C_3), etc. Once the optimal placements for $C_1, \ldots, C_{i-1}$ are computed for the $2M + 1$ possible values of d_{i-1}, we compute the costs for each combination of d_{i-1} and d_i to obtain the $2M + 1$ optimal placements for $C_1, \ldots, C_i$. Let $t_i(d_i)$ denote the cost (F_{max}) of the best placement solution for $C_1, \ldots, C_i$ for each possible d_i value. This value is computed recursively. For bookkeeping purposes, define $\gamma_i(d_i)$ to be the displacement for cell C_{i-1} in the optimal sub-solution of $C_1, \ldots, C_i$ with displacement d_i for C_i. Pseudocode for this algorithm is given in Algorithm 1. Steps 1 and 2 initialize the solution costs. The main algorithmic computation takes place between Steps 5 and 7, where y iteratively stores the lowest cost solution of placing C_i at d_i and trying all possible locations for C_{i-1}, and $\gamma_i(d_i)$ stores the corresponding best location for C_{i-1}. Steps 8-11 compute the end case of the last cell in the row, and the solution is recovered in Steps 12-13.

Algorithm 1 Optimization within a Row (OR)

```
▷ Input: Cells $C_1$ to $C_n$ in a row from 0 to $L$
▷ Output: Displacement $d_i$ for each cell $C_i$
$t_1(d_1) = 0$, $d_1 = -M$ to $M$
$t_i(d_i) = \infty$, $i = 2$ to $n$, $d_i = -M$ to $M$
for Cell $C_i$, $i = 2$ to $n$
    for $d_i = -M$ to $M$
        for $d_{i-1} = -M$ to $M$
            $y = \max(t_{i-1}(d_{i-1}), F_{i-1}(d_{i-1}, d_i))$
            if($y < t_i(d_i)$) then $t_i(d_i) = y$, $\gamma_i(d_i) = d_{i-1}$
$F_{max} = \infty$, $q = 0$
for $d_n = -M$ to $M$
    if($t_n(d_n) < F_{max}$) then $F_{max} = t_n(d_n)$, $q = d_n$
$d_n = q$
for $i = n$ downto 2
    $d_{i-1} = \gamma_i(d_i)$
```

Table 1: Design Characteristics

Tech	Design	# cells	# nets	Density (%)	Area (mm^2)
65nm	CKT1	162K	157K	67	2.62
	CKT2	1502K	1045K	21	42.14
45nm	CKT3	357K	322K	57	12.81
	CKT4	960K	1039K	53	31.55
32nm	CKT5	20K	12K	66	0.16
	CKT6	6K	5K	51	0.08

6. EXPERIMENTAL RESULTS

To validate MILOR, we chose six placed designs from logic blocks synthesized from 65, 45, and 32nm production libraries, ranging from 6000 to over 1.5 million cells. All these designs have been through detail placement, legalization, circuit optimization (e.g. buffering) and timing optimization steps. Design characteristics are shown in Table 1. Since 32nm is a fairly new technology node, only small technology-flushing testcases were available.

To measure the effectiveness of local congestion mitigation techniques we routed the results with a commercial router.[1] The key router statistics we extract are as follows: **Global Overflow** tells how many nets violate the global routing constraints, which is a typical measure of global routing success, but ignores local congestio. **Routing Errors** gives all the violations (shorts, opens, spacing, same-net, . . .) after routing and is the best indicator at how close the design was to achieving a complete routing solution. **Fast Errors** is the number of errors reported by a "light" version of detailed routing which gives an indication of how many hard detailed routing problems exist. **Run Time** is the total router runtime on a 2.6 GHz Linux machine, while restricting the router to a single thread for consistency in comparison.

To see the contribution of each component of MILOR, we ran routing on four different placements, the first being the initial placement of the design. Second, we ran just the inflation/legalization phase (IL). Third, we combine IL with optimization within a row (OR) to obtain the ILOR algorithm. Finally, we add the movement component to obtain the entire MILOR algorithm. Results are shown in Table 2. As shown in this table, MILOR runs in a few hours or less, even on the largest designs. On all designs MILOR runtime is overshadowed by router runtime. On some designs, like CKT1 and CKT2, the MILOR phases improve the global overflow statistics, while on CKT3 they make them worse. It is shown that all components of MILOR consistently reduce routing errors, again supporting the notion that local congestion is not well captured by overflows from global routing. It can be seen that IL, ILOR, and MILOR are able to reduce

[1]Unfortunately, we cannot utilize academic routers for this experimental methodology since they lack design rule knowledge and model pins as points.

Table 2: IL, ILOR and MILOR routing results

Design	Flow	CPU (min)	Global OV	Fast Errors	Route Errors	Router CPU (h)
CKT1	None	0	161	44620	43	15.7
	IL	24	84	4657	22	3.5
	ILOR	30	85	3793	0	3.4
	MILOR	66	30	3715	0	2.3
CKT2	None	0	1810	53299	3703	47.5
	IL	116	484	16669	319	14.3
	ILOR	117	91	15226	144	9.5
	MILOR	259	83	15659	164	12.2
CKT3	None	0	476	24761	3583	58.2
	IL	54	677	7375	1726	49.5
	ILOR	60	735	6552	1070	45.1
	MILOR	120	684	5723	190	45.3
CKT4	None	0	429	167724	544	68.1
	IL	126	411	41237	238	15.2
	ILOR	129	420	38403	199	13.5
	MILOR	294	430	28507	137	13.8
CKT5	None	0	0	40658	967	6.1
	IL	0.6	0	33965	720	5.9
	ILOR	1.2	0	33547	648	5.3
	MILOR	1.8	0	29667	212	3.9
CKT6	None	0	0	12999	87	2.0
	IL	0.6	0	6760	57	1.3
	ILOR	0.6	0	6029	35	1.3
	MILOR	0.6	0	5548	5	0.9

fast errors. Similarly, IL, ILOR, and MILOR all significantly reduce total routing errors on average. MILOR produces an average of 89.5% fewer router errors. All three approaches drastically improve router runtime. Running MILOR before routing reduces runtime by an average of 58.7%, a greater than $2X$ speedup. A faster router runtime strongly indicates that the router had an easier-to-solve instance to optimize.

7. CONCLUSION

In this paper, we proposed new metrics to measure local routing congestion, demonstrating shortcomings with the global routing model. We also presented a suite of detailed placement algorithms, MILOR, that significantly improve router errors. In future work, we seek to further improve the local congestion metrics by closely studying the causes of actual routing failures.

8. ADDITIONAL AUTHORS

Lakshmi Reddy (IBM, email: reddyl@us.ibm.com), Jarrod Roy (IBM, email: royj@us.ibm.com), Gustavo Tellez (IBM, email: tellez@us.ibm.com), Paul Villarrubia (IBM, email: pgvillar@us.ibm.com) and Natarajan Viswanathan (IBM, email: nviswan@us.ibm.com),

9. REFERENCES

[1] C. Alpert, Z. Li, M. Mo¡tt, G.-J. Nam, J. Roy, and G. Tellez, "What Makes a Design Di¡cult to Route," *ACM ISPD*, pp. 7–12, March 2010.
[2] A.E. Caldwell, A.B. Kahng adn I.L. Markov, "Optimal Partitioners and End-Case Placers for Sandard-Cell Layout," *IEEE TCAD*, vol. 19, 2000.
[3] Y.-J. Chang, Y.-T. Lee and T.-C. Wang, "NTHU-Route 2.0: A Fast and Stable Global Router," *IEEE/ACM ICCAD*, pp. 338–343, November 2008.
[4] D. Hill, "Method and System for High Speed Detailed Placement of Cells within an Integrated Circuit Design," *US Patent 6370673*, April 2002.
[5] M. Mo¡tt, "MaizeRouter: Engineering an E↑ective Global Router," *IEEE TCAD*, vol. 27, no. 11, pp. 2017–2026, 2008.
[6] S. Nassif and K. Nowka, "Physical Design Challenges beyond the 22nm Node," *ACM ISPD*, March, 2010.
[7] G.-J. Nam, C. Sze and M. Yildiz, "The ISPD Global Routing Benchmark Suite," *ACM ISPD*, pp. 156–159, 2008.
[8] K. Nowka, S. Nassif, and K. Agarwal, "Characterization and Design for Variability and Reliability," *IEEE CICC*, pp. 341–346, September 2008.
[9] H. Ren, D. Pan, C.J. Alpert, P.G. Villarrubia and G.-J. Nam, "Di↑usion-based Placement Mitigration with Application on Legalization," *IEEE TCAD*, vol. 26, no. 12, pp. 2158–2172, 2007.
[10] J. Roy and I. Markov, "High Performance Routing at the Nanometer Scale," *IEEE/ACM ICCAD*, pp. 496–502, 2007.
[11] J. Roy, N. Viswanathan, G.-J. Nam, C. Alpert and I. Markov, "CRISP: Congestion Reduction by Iterated Spreading During Placement," *IEEE/ACM ICCAD*, pp. 357–362, November 2009.
[12] Y. Zhang and C. Chu, "CROP: Fast and E↑ective Congestion Re↓nement of Placement," *IEEE/ACM ICCAD*, pp. 344–350, 2009.
[13] Personal communication with an industrial routing expert