Submodular Maximization-inspired Adaptive Routing Bend Space Planning

Siting Liu[1,2], Peng Xu[2], Peiyu Liao[2], Keren Zhu[3], Yibo Lin[4], Bei Yu[2]
teng

***Abstract*—Routing can greatly impact tape-out chip performance by determining the physical layout of metal wire segments. As designs grow in complexity and size, modern routing frameworks struggle to manage limited routing resources among numerous nets efficiently. In this paper, we introduce a novel adaptive routing bend space planning framework, ARSP, that adaptively adjusts the routing bend space for each net based on the availability of routing resources throughout the routing flow. ARSP is built on a well-defined submodular maximization problem and uses an efficient approximation algorithm to ensure sub-optimal performance. Integrating ARSP with state-of-the-art routing flows shows an average improvement of 6.34% and 5.11% in reducing shorts and spacing violations, respectively. Additionally, our adaptive planning framework outperforms all static routing space planning strategies in both effectiveness and efficiency, showing the necessity of adaptive planning.**

I. Introduction

Routing [1] is a crucial component in the very large-scale integration (VLSI) design flow, responsible for assigning metal wires to ensure shorter signal propagation delays and improved yield [2], [3]. Due to the exponential increase in design scales and the complexity of design rules, routing has become an extremely time-intensive stage, often consuming more than half of the physical design cycle [4]. Modern routing frameworks incorporate various algorithms, such as pattern routing and maze routing, on a rectilinear Manhattan routing grid [5]. To improve the quality of routing solutions, it is generally necessary to expand the routing space for additional candidate paths, which can, however, negatively impact efficiency. We illustrate the routing spaces of three widely-used routing patterns [6], [7] in Fig. 1, where the candidate spaces to insert a bend point are extracted for different patterns, called routing bend spaces. It is important to note that a unique routing path can be determined once all intermediate bend positions are specified [8]. As shown in Fig. 1, the complexity of different routing patterns is demonstrated by their routing bend spaces, with more complex patterns operating within more extensive routing bend spaces. This complexity highlights the challenge of simultaneously achieving both efficiency and effectiveness.

The typical routing framework always employs an easy and efficient routing pattern first as the initial routing and then invokes more complex routing algorithms iteratively to route the nets failed during the initial routing. Most nets would complete routing during the initial routing and only minor nets passed to the rip-up and reroute iterations. Therefore, the effectiveness of the initial routing is important. However, current initial routing patterns are heuristically selected relying on the experience without performance guarantee [9]–[11]. On the other hand, as depicted by Fig. 2, the available routing resource keeps changing and is not uniformly utilized as more nets have been routed. A static initial pattern choice cannot adapt to the changing resource situation and net structure, which may lead unnecessary detours in the subsequent stages. Proactively managing the routing resource in the early stage can better mitigate potential severe detours and unsolvable violations later on. However, finding optimal routing pattern/space planning solution for all the millions of nets is highly challenging. Therefore, developing an efficient adaptive routing space planning framework is vital to utilizing routing resources more rationally, enhancing routing quality, and reducing design violations.

This work is partially supported by Research Grants Council of Hong Kong SAR (No. CUHK14210723).

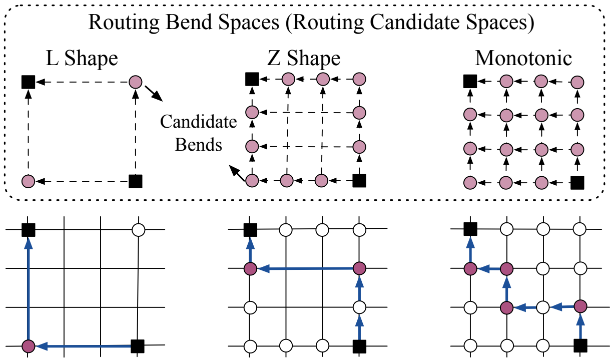


Fig. 1 Illustration of routing bend spaces for different routing patterns. Black squares represent the pins to connect, and pink circles indicate candidate bend points. Sample routing paths with different bend spaces are shown below.

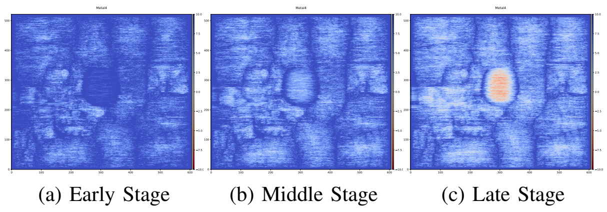


Fig. 2 Heatmap of varying routing resources as routing progresses. Dark blue indicates ample resources, while red shows scarcity.

In this work, we introduce a novel adaptive routing space planning framework, ARSP, to efficiently allocate appropriate routing bend spaces to each net based on the current availability of routing resources. The adaptive routing space planning aims to maximize the average available routing resources across all nets, which makes it as a submodular maximization problem. By leveraging the submodular nature of the problem, ARSP can use approximated algorithms that guarantee near-optimal solutions, ensuring high performance and efficiency. The key contributions of this paper are listed,

- We are the first to model the adaptive routing space planning as a submodular maximization problem.
- We introduce a novel routing space planning framework, **ARSP**, which provides performance guarantees and linear runtime complexity. ARSP also incorporates lookahead routing demands to mitigate subsequent detours.
- Integrating our ARSP framework with state-of-the-art routing flows results in a higher quality of solutions, with 6.34% fewer

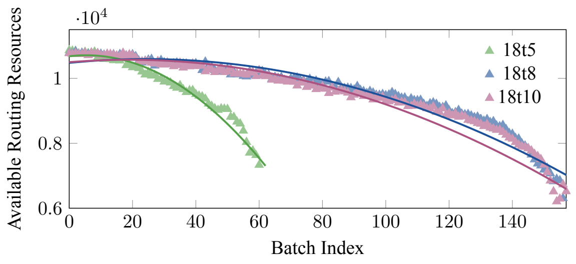
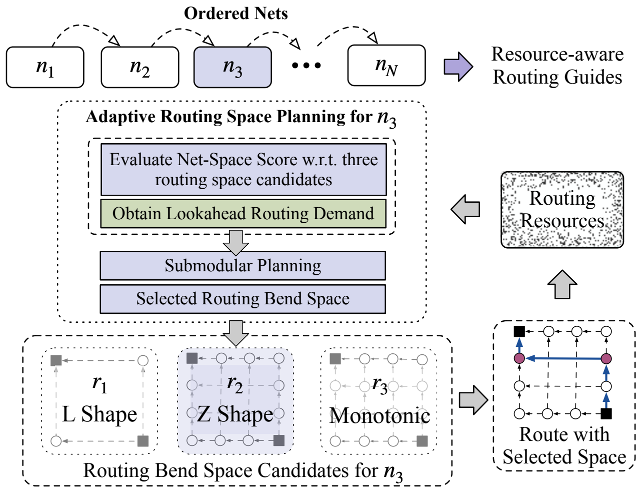


Fig. 3 Normalized batch routing resource during sequential routing shows a positive, declining trend.



Ordered Nets
n_1
n_2
n_3
n_N
Resource-aware Routing Guides
Adaptive Routing Space Planning for n_3
Evaluate Net-Space Score w.r.t. three routing space candidates
Obtain Lookahead Routing Demand
Submodular Planning
Selected Routing Bend Space
Routing Resources
r_1 L Shape
r_2 Z Shape
r_3 Monotonic
Routing Bend Space Candidates for n_3
Route with Selected Space

Fig. 4 Overall ARSP flow to process a set of ordered nets sequentially. The purple blocks represent the parts with submodularity and the green part is an additional optimization for resource awareness.

shorts and 5.11% fewer spacing violations. on the ICCAD2019 contest benchmarks. Its robustness is further demonstrated by the integration into various routing flows.

II. Preliminaries

A. Routing

Modern routing frameworks can be broadly categorized as concurrent or sequential. Concurrent frameworks, such as those based on integer linear programming (ILP) and multi-commodity flow, aim to address the routing problem efficiently. However, directly applying ILP to millions of nets is not practical. BoxRouter [12], [13] utilizes ILP in constrained regions but sacrifices optimality for quick solutions. On the other hand, the min-max algorithm [14] approaches the global routing problem as a multi-commodity flow, easing some constraints. However, most mathematical models proposed for the routing problem are proven to be NP-complete [15], [16], rendering them impractical in the actual design flow. Recently, researchers have focused on the heuristic net-by-net routing framework to address runtime issues. While frameworks like FastRoute [7], [17]–[19] and CUGR [9], [11] can balance multiple objectives efficiently, net-by-net processing may lead to detours in the late nets. Considering global routing resource management early can help optimize detours during the sequential framework.

B. Varying Available Routing Resources in Sequential Routing

The previous analysis of available routing resources suggests that as more nets are routed, the available routing resources should decrease. To validate this hypothesis, we depict the changing trend of the average available routing resource in Fig. 3. Note that we have grouped every 1000 nets into a single batch and plotted the average routing resource after processing each batch. For more comprehensive verification, three different designs are used here. Irrespective of the design complexity, the overall trend of the available routing resource consistently decreases. This trend underscores the need for adaptive routing space planning to accommodate the varying situations of available routing resources.

C. Adaptive Routing Space Planning

Nets have varying resource requirements based on bounding box size and available routing resources. Therefore, it is essential to address each net individually rather than adopting a uniform routing space. Our goal is to optimize routing resources for each net, which leads us to introduce adaptive routing space planning.

Problem 1 (Adaptive Routing Space Planning)**.** *Given a fixed ordered set of nets $\{n_1, \cdots, n_N\}$ and potential routing spaces $\{r_1, r_2, r_3\}$, our objective is to identify an optimal net-space assignment set $\mathcal{A}_N$, comprising N elements, to maximize the average (or total) available routing resources during the sequential routing process.*

D. Submodular Optimization for Total Available Routing Resources

Submodularity is a property on the set function, $F(\mathcal{S}) : 2^{\mathcal{V}} \to \mathbb{R}$, that captures the decreasing marginal gains Δ_F associated with adding elements to an expanding set. Mathematically, the formal definition of submodularity can be expressed as follows,

Definition 1 (Submodularity)**.** *A function $F : 2^{\mathcal{V}} \to \mathbb{R}$, where $\mathcal{V}$ is a finite set and $2^{\mathcal{V}}$ is the power set of $\mathcal{V}$, is submodular if for every $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $v \in \mathcal{V} \setminus \mathcal{B}$, it holds that*

$$F(\mathcal{A} \cup \{v\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{v\}) - F(\mathcal{B}).$$

Submodularity implies that the marginal gain of the objective function F decreases as the set $\mathcal{S}$ increases. This feature presents new opportunities for modeling the total available routing resources as a submodular function, as these resources continuously decrease throughout the entire routing flow, as illustrated in Section II-B. Moreover, a significant subclass of submodular functions is those with monotonicity, where expanding the set $\mathcal{S}$ does not decrease $F(\mathcal{S})$.

Definition 2 (Monotonicity)**.** *A set function $F : 2^{\mathcal{V}} \to \mathbb{R}$ is monotone if $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}, F(\mathcal{A}) \leq F(\mathcal{B})$.*

Utilizing monotonicity and submodularity, various efficient algorithms [20]–[23] can obtain an approximated solutions to maximize the submodular objective function. It is noted that [20] has proved that the greedy algorithm can provide an $(1 - \frac{1}{e})$ approximation to the optimal value. This property could provide a new opportunity for adaptive routing space planning to maximize the averaged (or the total) available routing resource for all nets.

III. Sequential Routing Space Planning

A. Overview

We begin by outlining the ARSP framework's overall flow, illustrated in Fig. 4. Using net n_3 as an example, our approach first evaluates its net-space scores across three candidate routing spaces, quantifying the average available routing resources. This planning problem naturally formulates as a submodular maximization problem

(Section III-B), solvable with approximation-guaranteed algorithms (Section III-C).

To enhance planning quality, we incorporate lookahead routing demand estimation (Section III-D) to guide optimal space selection for n_3. In Fig. 4, purple blocks denote core submodular optimization components, while the green block represents our additional lookahead optimization. Section IV-E quantitatively validates each component's contribution.

B. Modeling by Submodular Function

Given constrained routing resources, effective resource management becomes crucial during routing. As formalized in Problem 1, we aim to maximize the average/total available routing resources across all nets. We first define a net-space score metric $R_r(n_i)$ to quantify the resource availability when net n_i utilizes routing space $r \in \{r_1, r_2, r_3\}$.

The net-space score $R_r(n_i)$ represents the expected available routing resources across all feasible paths when net n_i select the routing bend space r:

$$R_r(n_i) = \mathbb{E}[R_e] = \sum_{e \in E_r^i} R_e \cdot p_e(n_i, r), \tag{1}$$

where E_r^i denotes the set of routing edges covered by space r for net n_i (as shown in Fig. 1), R_e is the available resource on edge e, and $p_e(n_i, r)$ is the edge occupation probability under uniform path distribution.

As for the probability calculation, take a net with $M \times N$ bounding box as the example. The edge occupation probabilities differ by routing space type. In L-shaped space (r_1), there are exactly 2 candidate paths with $p_e(n_i, r_1) = 1/2$ for edges on either path. Z-shaped space (r_2) contains $M + N$ paths, where boundary edges have occupation probabilities $\frac{M-m}{M+N}$ for bottom (or top) horizontal edges at column m (or $M - m$) and $\frac{N-n}{M+N}$ for right (or left) vertical edges at row n (or $N - n$), while internal edges have uniform probability $\frac{1}{M+N}$. Monotonic space (r_3) offers $\binom{M+N}{M}$ paths, with edge-specific probabilities given by $p_e(n_i, r_3) = \binom{m+n}{m}\binom{(M+N)-(m'+n')}{M-m'}/\binom{M+N}{M}$ for each edge $e = (m, n) \to (m', n')$.

Figure 5 demonstrates the net-space score computation (Eq. 1) for a 2×2 routing grid, displaying available resources R_e on each edge in the leftmost. The analysis compares three routing bend spaces with bend points marked by pink circles: (1) L-shape (r_1) with exactly 2 candidate paths, (2) Z-shape (r_2) offering 4 potential paths, and (3) monotonic (r_3) with 6 possible paths. The net-space score for each configuration calculates the expected available resources along its utilized edges, weighted by their respective path distribution probabilities as marked on the corresponding edges.

The proposed net-space score provides dual benefits: it quantifies the expected available routing resources across all paths in a given routing space, while the edge probability term p_e explicitly models the utilization likelihood of individual routing edges. Building on Eq. 1, we formulate the objective from Problem 1 as:

$$F(\mathcal{A}_i) = \sum_{j=1}^{i} R_{r_j}(n_j), \tag{2}$$

where $r_j \in \{r_1, r_2, r_3\}$ denotes the routing space assigned to net n_j in the partial assignment set $\mathcal{A}_i$, containing the first i assigned nets while leaving $N - i$ nets remaining in the routing sequence.

The function $F(\cdot)$ serves as an evaluation metric for net-space assignment sets, where each assignment's contribution is quantified through the net-space score $R_{r_i}(n_i)$. Specifically, $F(\mathcal{A}_N)$ represents the total available routing resources when applying the complete assignment solution $\mathcal{A}_N$ to an ordered set of nets.

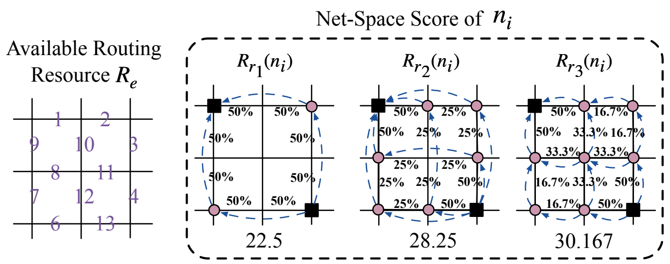


Fig. 5 Net-space score computation sample. The left graph shows the grids within net n_i's bounding box with available routing resources R_e. The right figures calculate the net-space scores for three different routing bend spaces (pink circles).

The marginal gain $\Delta_F(n_i^{r_i}|\mathcal{A}_{i-1})$, which captures the incremental improvement from assigning routing space $r \in \{r_1, r_2, r_3\}$ to net n_i, is defined as:

$$\Delta_F(n_i^r|\mathcal{A}_{i-1}) = R_r(n_i). \tag{3}$$

This allows us to express the set evaluation function recursively:

$$F(\mathcal{A}_i) = \begin{cases} 0, & \mathcal{A}_i = \emptyset \\ F(\mathcal{A}_{i-1}) + \Delta_F(n_i^r|\mathcal{A}_{i-1}), & \mathcal{A}_i \neq \emptyset \end{cases} \tag{4}$$

Based on our analysis of routing resource allocation, we maintain two key properties: (1) non-negativity ($F(\mathcal{A}_i) \geq 0$ since $\Delta_F > 0$), ensuring sufficient resources for manufacturability; and (2) monotonically decreasing available resources, as demonstrated in Fig. 2, which implies that the net-space score $R_{r_i}(n_i)$ diminishes as more nets are sequentially assigned and routed.

Following the above analysis, we can extend them to the following lemma:

Lemma 1. *The non-negative net-space assignment set evaluation function: F is monotone and submodular during the sequential routing process.*

Proof. The monotonicity of F has been established through the above non-negative assumption for the net-space score and the recursive calculation in Equation (4). Referring to the definition of submodularity in Section II-D, the decreasing feature of the marginal gain Δ_F, in the other word, net-space score $R_{r_i}(n_i)$, makes F to be a submodular function. □

C. Planning with Submodular Optimization

The above analysis points out two imortant features, submodularity and monotonicity, of the objective function F for the adaptive routing space planning problem. This greatly aids us in transforming the adaptive routing space planning problem into a submodular function maximization problem, subject to a cardinality constraint, as follows:

$$\mathcal{A}_N = \underset{\mathcal{A}_N : |\mathcal{A}_N| = N}{\operatorname{argmax}} \; F(\mathcal{A}_N). \tag{5}$$

Maximizing a submodular function with a cardinality constraint is a known NP-hard problem. Luckily, several efforts have been made to provide efficient approximation solutions, as previously mentioned. We employ the submodular planning framework outlined in [20] to solve the routing space planning problem defined in Equation (5). This straightforward greedy algorithm can deliver a solution with an $1 - \frac{1}{e}$ approximation ratio as mentioned in [20].

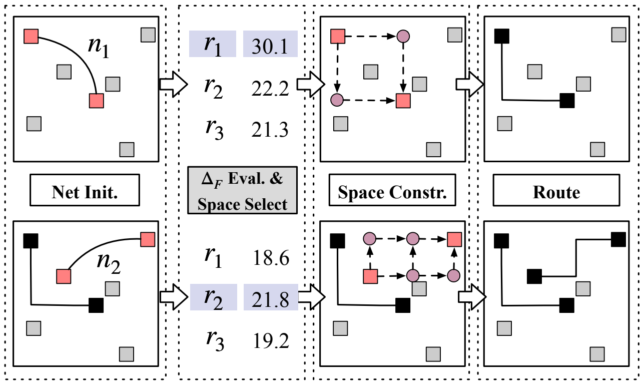


Fig. 6 Approximated submodular optimization illustration. Each time, only one net n_i is processed. The marginal gains for n_i across three routing spaces are evaluated, and the best net-space assignment with the highest marginal gain is selected.

The greedy submodular optimization begins with the empty set $\mathcal{A}_0$ and then repeats the subsequent steps for $i = 1, \cdots, N$.

$$\mathcal{A}_i = \mathcal{A}_{i-1} \cup \left\{ \underset{r \in \{r_1, r_2, r_3\}}{\operatorname{argmax}} \; F\left(\mathcal{A}_{i-1} \cup \{n_i^r\}\right) \right\}. \tag{6}$$

Note that,

$$\underset{r \in \{r_1, r_2, r_3\}}{\operatorname{argmax}} \; F\left(\mathcal{A}_{i-1} \cup \{n_i^r\}\right) = \underset{r \in \{r_1, r_2, r_3\}}{\operatorname{argmax}} \; \Delta_F\left(n_i^r | \mathcal{A}_{i-1}\right) \tag{7}$$

The size of the final optimal solution $\mathcal{A}_N$ is always N because the gain value $\Delta_F(n_i^{r_i}|\mathcal{A}_{i-1})$ is non-negative, and the maximum set evaluation value is achieved when the set is at its maximum size.

Fig. 6 demonstrates the detailed greedy submodular optimization flow. Given a list of ordered nets $\{n_1, n_2, \cdots\}$ and three candidate routing spaces r_1, r_2, r_3, the process initiates with net n_1. The gain values $\Delta(n_1^{r_1}|\emptyset)$, $\Delta(n_1^{r_2}|\emptyset)$, and $\Delta(n_1^{r_3}|\emptyset)$ are evaluated independently, considering that $\mathcal{A}_0 = \emptyset$. As depicted in Fig. 6, the respective gain values are 30.1, 22.2, and 21.3. Consequently, r_1 is selected for n_1, resulting in $\mathcal{A}_1 = \{n_1^{r_1}\}$. Then, it proceeds to process n_2 by assessing $\Delta(n_2^{r_1}|\mathcal{A}_1)$, $\Delta(n_2^{r_2}|\mathcal{A}_1)$, and $\Delta(n_2^{r_3}|\mathcal{A}_1)$, with updated routing resources yielding values of 18.6, 21.8, and 19.2, respectively. As dictated by Equation (7), r_2 is selected for net n_2. Consequently, $\mathcal{A}_2 = \{n_1^{r_1}, n_2^{r_2}\}$. As outlined in Equation (5), this process continues until all net have been assigned an appropriate routing space.

Meanwhile, we can establish the following theorem by capitalizing on the submodularity and monotonicity discussed in the previous section. This theorem provides a $1 - \frac{1}{e}$ approximation factor with our submodular planning framework (Equation (7)). The proof is refer to [20].

Theorem 1. *Let $\mathcal{A}_i = (a_1, a_2, \ldots, a_i)$ be the assignment sequence formed by our submodular planning framework, for i times, as defined in Equation (7). $\mathcal{A}^* = (a_1^*, a_2^*, \ldots, a_N^*)$ denotes the optimal solution with maximal total available routing resources across all the nets. When F is the objective function defined in Equation (4), we can conclude that $F(\mathcal{A}_N) \geq \left(1 - \frac{1}{e}\right) F(\mathcal{A}^*)$.*

Proof. For all $i \leq N$, we have:

$$\begin{aligned} F\left(\mathcal{A}^*\right) & \leq F\left(\mathcal{A}^* \cup \mathcal{A}_i\right) \\ & = F\left(\mathcal{A}_i\right) + \textstyle\sum_{j=1}^N \Delta_F\left(a_j^* \mid \mathcal{A}_i \cup \left\{a_1^*, a_2^*, \ldots, a_{j-1}^*\right\}\right) \\ & \leq F\left(\mathcal{A}_i\right) + \textstyle\sum_{a^* \in \mathcal{A}^*} \Delta_F\left(a^* \mid \mathcal{A}_i\right) \end{aligned}$$

With $a_{i+1} = \operatorname{argmax}_{a \in \mathcal{V} \setminus \mathcal{A}_i} \Delta_F(a|\mathcal{A}_i)$ in Equation (7),

$$\begin{aligned} F(\mathcal{A}^*) & \leq F\left(\mathcal{A}_i\right) + \textstyle\sum_{a* \in \mathcal{A}^*} \Delta_F\left(a_{i+1} \mid \mathcal{A}_i\right) \\ & = F\left(\mathcal{A}_i\right) + N \Delta_F\left(a_{i+1} \mid \mathcal{A}_i\right) \\ & = F\left(\mathcal{A}_i\right) + N(F(\mathcal{A}_{i+1}) - F(\mathcal{A}_i)). \end{aligned}$$

Rearranging the terms, we have proved that:

$$\begin{aligned} F(\mathcal{A}^*) - F(\mathcal{A}_{i+1}) & \leq \left(1 - \tfrac{1}{N}\right)\left(F(\mathcal{A}^*) - F(\mathcal{A}_i)\right) \\ & \leq \left(1 - \tfrac{1}{N}\right)^{i+1}\left(F(\mathcal{A}^*) - F(\mathcal{A}_0)\right). \end{aligned}$$

Then, we have the following inequality when $i = N - 1$,

$$F(\mathcal{A}^*) - F(\mathcal{A}_N) \leq (1 - \frac{1}{N})^N F(\mathcal{A}^*) \leq \frac{1}{e} F(\mathcal{A}^*).$$

Therefore, $F(\mathcal{A}_N) \geq \left(1 - \frac{1}{e}\right) F(\mathcal{A}^*)$, which concludes our proof. □

Besides the performance guarantee, our proposed submodular planning framework is efficient with polynomial runtime complexity. For each net, we need to evaluate the marginal gain $\Delta_F(\cdot)$ for M candidate routing spaces, which only takes $\mathcal{O}(M)$ runtime complexity. Therefore, the overall time complexity is $\mathcal{O}(NM)$.

D. Lookahead Routing Demand

Besides the available resources, we find that it is also necessary to recognize the importance of the future resource demand from the unrouted nets to improve the early-stage routing resource management. Inspired by the rectangular uniform wire density (RUDY) [24], we introduce a lookahead routing demand to estimate the possible routing demand distribution from the nets that are still not routed. RUDY can model the wire distribution over the die area by calculating the uniform wire density within the enclosing rectangle of each single net and then accumulating the demands of all the nets over all the routing grids. It assumes the routing demand of net n_i is uniformly distributed to the bounding box of n_i.

Then, our lookahead routing demand D_e on a single grid graph edge e is calculated as the estimated RUDY value on e.

$$D_e = \text{RUDY}(x_e, y_e), \tag{8}$$

where x_e, y_e is the location of the edge e and RUDY reflect the accumulated demands on e of all the unrouted nets as introduced in [24]. Having the lookahead routing demand on each grid graph edge, we incorporate the normalized demand into the preceding objective function (Equation (2)) to consider both available resources and the lookahead demands.

$$\Delta_F(n_i^{r_i}|\mathcal{A}_{i-1}) = R_{r_i}(n_i) - \sum_{e \in E_{r_i}^i} D_e \times p_e(n_i, r_i). \tag{9}$$

The net-space score $R_{r_i}(n_i)$ and the lookahead routing demand D_e keep updating across the sequential routing process. Note that we use the prefix matrix calculation [25] to make it more efficient in our implementation. Incorporating the available routing resources (net-space score) and the lookahead routing demand in Equation (9), our overall adaptive routing space planning framework, ARSP, is described in Algorithm 1.

IV. Experimental Results

A. Experimental Setting

Our adaptive routing space planning (ARSP) framework was implemented in C++ and evaluated through integration with the CUGR 2.0 as the global router [11] and Dr.CU 2.0 as the detailed router [26]. To demonstrate robustness, we additionally validated

Algorithm 1 Adpative Routing Space Planning

```
Input: Nets $\{n_1, \cdots, n_N\}$, routing spaces $\{r_1, r_2, r_3\}$.
Output: Net-space assignment $\mathcal{A}_N$, routed nets $\{n_1, \cdots, n_N\}$.
 1: Resource ← initial available routing resources;
 2: Demand ← lookahead routing demand of all unrouted nets;
 3: $\mathcal{A}_0 \leftarrow \emptyset$;
 4: for all $i \in 1, \cdots, N$ do
 5:     $c_1 \leftarrow \Delta_F(n_i^{r_1} | \mathcal{A}_{i-1})$;                ▷ Equation (9)
 6:     $c_2 \leftarrow \Delta_F(n_i^{r_2} | \mathcal{A}_{i-1})$;                ▷ Equation (9)
 7:     $c_3 \leftarrow \Delta_F(n_i^{r_3} | \mathcal{A}_{i-1})$;                ▷ Equation (9)
 8:     $r_i \leftarrow \text{argmax}_{r \in \{r_1, r_2, r_3\}}(\{c_1, c_2, c_3\})$;
 9:     $\mathcal{A}_i \leftarrow \mathcal{A}_{i-1} \cup \{n_i^r\}$;                ▷ Equation (7)
10:     Route($n_i$, $r$);        ▷ Route $n_i$ with the routing space $r$
11:     Resource ← subtracting the resource usage of $n_i$;
12:     Demand ← removing $n_i$ from the unrouted net set;
13: end for
```

TABLE I ICCAD2019 Benchmarks.

Design	# nets	# G-cells
18t5 & 18t5m	72394	619×613
18t8 & 18t8m	179863	905×883
18t10 & 18t10m	182000	606×522
19t7 & 19t7m	358720	1053×1011
19t8 & 19t8m	537577	1202×1138
19t9 & 19t9m	895252	1337×1433

ARSP with TritonRoute [27] as discussed in Section IV-F. All experiments were conducted on a 64-bit Linux server, using single-thread execution for ARSP and global routing stages, while leveraging eight threads for detailed routing. Solution quality metrics were evaluated using an industry-leading commercial tool.

For comparative analysis, we maintained CUGR 2.0's original rip-up and reroute stage while integrating our ARSP framework into the initial routing stage. This controlled methodology enables direct comparison against three static routing space selection strategies: the default all-L (exclusive L-shape routing), all-Z (exclusive Z-shape routing), and all-mono (exclusive monotonic routing) configurations. The net-space score calculation for monotonic routing spaces follows the efficient implementation from [28]. This experimental design precisely isolates the impact of adaptive routing space planning while maintaining all other optimization factors constant. We employed the ICCAD 2019 contest benchmarks [29] for comprehensive evaluation, which include designs ranging from 70k to 900k nets. The benchmark suite features both standard configurations (nine metal layers) and reduced-resource variants (five metal layers, denoted by 'm' suffix), providing an effective platform to assess our framework's scalability. Complete benchmark specifications are provided in Table I.

B. Detailed Routing Solution Quality

To evaluate the effectiveness of our proposed adaptive routing bend space planning framework, ARSP, we integrate it into a modern routing flow, CUGR 2.0 [11] and Dr.CU 2.0 [26], and evaluate the detailed routing solution performance with the evaluator in [29]. Compared to the routing flow without the adaptive planning framework, the experimental results are illustrated in TABLE II. To begin with, as for the design rule violations, we list the score of shorts violations and spacing violations separately since the shorts violation is more related to the routing resource management. The spacing design rules here include the min-area rule, PRL rule, EOL spacing rule, cut spacing rule, and corner spacing rule. As listed in TABLE II, with the help of ARSP, the state-of-the-art global router could obtain up to 52.3% and 28.4% improvement and an averaged 6.34% and 5.11% reduction on the score of shorts and spacing violations, respectively. Besides the direct effect on the design rule violations, we also estimate the other metrics for detailed routing solution performance. Overall, integrating ARSP could bring slightly improved wirelength and the number of vias by 0.4% and 0.2%. Further, ARSP reduces the non-preferred usage by 0.91% on average. All in all, the significant improvement in the number of design rule violations proves the efficacy of our ARSP framework in routing resource management by modeling as a submodular maximization problem and obtain a performance guarantee with the approximated algorithm.

C. Performance with Planning

The adequacy of directly utilizing a complex routing space without prior planning is a significant concern. To address this, we conducted experiments where we directly replaced the default L-shaped routing space (all-L) in the initial stage of CUGR2.0 [11] with more complex configurations, referred to as 'all-Z' and 'all-mono'. As shown in the comparison on the left side of TABLE III, our proposed solution, ARSP, increases runtime by only 15.8% but significantly reduces shorts violation scores and requires substantially less runtime compared to other static complex routing space configurations. The overall routing runtime increases because ARSP plans for more complex routing spaces, such as Z-shape and monotonic shape, for certain nets to optimize the available routing resources. A detailed analysis of the runtime is provided in the following section, demonstrating our framework's efficiency with ARSP.

D. Runtime Analysis

The runtime analysis of modern sequential routing frameworks with different routing space configurations is presented in Table III. We distinguish between the adaptive routing space planning (ARSP) time and the global routing (GR) time to emphasize our framework's efficiency. The results reveal significant runtime differences: using Z-shape routing spaces universally increases runtime by 1.893×, while monotonic spaces require 28.155× running time compared to the baseline L-shape initial routing strategy. Our ARSP-guided sequential routing process takes less than twice the running time compared to the original sequential process, CUGR 2.0 [11], as some nets are assigned more complex routing spaces considering the dynamically available routing resource. Notably, ARSP accounts for only 15.8% to the total runtime while substantially improving global routing performance over direct replacement strategies. It is noted that integrating ARSP into the modern sequential routing framework results in significantly less runtime than flows with a direct replacement to the complex routing bend space. This comparison underscores ARSP's ability to balance routing resource usage and runtime cost dynamically. Table III thus confirms both the efficiency of our planning framework and its effective resource management capabilities.

E. Ablation on Lookahead Routing Demand

Integrating the submodular objective with lookahead routing demand modifies the performance approximation achieved by submodular optimization. However, lookahead routing demand allows the current planning step to account for future needs, potentially enhancing the overall routing space planning process. The final design rule checking results, shown in Fig. 7, validate the importance of lookahead routing demand in improving overall performance. Here, ARSP (✘ Lookahead) denotes the ARSP flow without lookahead routing demand, as described in Section III-B, while ARSP (✔

TABLE II Detailed routing solution quality comparison with the state-of-the-art routing flow, CUGR 2.0 [11] and Dr.CU 2.0 [26].

Design	Wirelength		# Vias		Non-prefer Usage		Shorts Vio. Score		Spacing Vio. Score	
	w/o ARSP	w/ ARSP	w/o ARSP	w/ ARSP	w/o ARSP	w/ ARSP	w/o ARSP	w/ ARSP	w/o ARSP	w/ ARSP
18t5	13699907	**13695406**	**1849918**	1850042	**219273**	237294	55109	**54500**	344000	**313000**
18t5m	13636161	**13601788**	**1828950**	1829644	96187	**94841**	31850	**31025**	**221500**	246000
18t8	32422365	**32412342**	**4648996**	4650840	261796	**233339**	**62613**	65757	156500	**130500**
18t8m	31767069	**31748442**	4568180	**4567856**	299218	**296329**	67600	**67515**	162500	**126500**
18t10	**33826371**	33827319	4988012	**4986012**	**767729**	775973	142740	**93708**	**459500**	475500
18t10m	**38167335**	38236602	5034728	**5034706**	1250505	**1213862**	1481905	**1300400**	620000	**556000**
19t7	60663314	**60578544**	8101456	**8096040**	875068	**873781**	2380460	**2360045**	5810500	**5791500**
19t7m	54368026	**54350639**	8119810	**8119784**	868979	**862940**	2132883	**2094895**	6109000	**6018000**
19t8	93172400	**93100471**	12657942	**12653226**	970780	**965609**	1658028	**1560165**	5969000	**5784000**
19t8m	90641938	**90596648**	12646890	**12644508**	1039826	**1034758**	2044363	**1988013**	**6010500**	6031000
19t9	140895973	**140877293**	21074040	**21061726**	**1587265**	1593804	**3643748**	3743920	**10407500**	10430000
19t9m	**135702014**	135709020	21015216	**21008288**	1707903	**1698419**	4634020	**4564885**	10764500	**10722000**
Ratio	1.0004	**1.0000**	1.0002	**1.0000**	1.0091	**1.0000**	1.0634	**1.0000**	1.0511	**1.0000**

TABLE III Comparison with static routing space planning strategies to show the importance of adaptive planning.

Design	Shorts Vio. Score				Global Routing Runtime(s)			
	all-L [11]	all-Z	all-mono	w/ ARSP	all-L [11]	all-Z	all-mono	ARSP / GR
18t5	55109	60996	43019	54500	27.951	111.738	650.811	6.055 / 64.965
18t5m	31850	31899	31789	31025	38.194	73.114	262.961	4.535 / 79.627
18t8	62613	64026	64563	65757	75.461	294.036	2214.748	16.899 / 177.852
18t8m	67600	65487	65184	67515	113.630	188.923	1024.927	9.778 / 163.927
18t10	142740	121765	126800	93708	105.189	260.682	1127.414	10.310 / 193.029
18t10m	1481905	1289470	1490305	1300400	190.094	258.627	522.388	7.388 / 238.763
19t7	2380460	2185983	2335888	2360045	158.421	751.069	15224.519	56.435 / 471.380
19t7m	2132883	2113043	2126588	2094895	227.169	357.820	1307.419	14.208 / 319.639
19t8	1658028	1554385	1541443	1560165	193.891	991.567	14628.345	57.010 / 497.325
19t8m	2044363	2029460	1868595	1988013	329.019	563.184	2767.920	26.273 / 462.034
19t9	3643748	3804293	3768655	3743920	311.318	1408.618	18974.665	75.649 / 700.663
19t9m	4634020	4349130	4403657	4564885	435.434	759.234	4001.113	36.421 / 626.298
Ratio	1.000	0.970	0.956	0.954	1.000	2.893	28.155	0.158 / 1.945

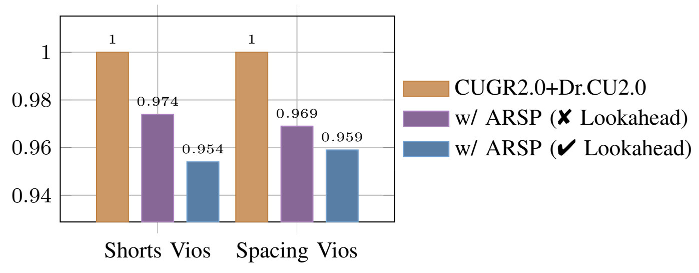


Fig. 7 Comparison to show the necessity of the lookahead routing demand for global routing resource management.

Lookahead) represents the complete ARSP framework with lookahead routing demand. Fig. 7 reveals that our lookahead routing demand contributes to a 2.0% improvement in shorts and a 1.0% improvement in spacing violations. These enhancements are in addition to the overall gains provided by the full ARSP framework. These results highlight the significance of lookahead routing demand in global routing resource management.

F. Robustness on Different Routing Flows

To further validate the robustness of our proposed adaptive routing space planning framework, besides the 'flow 1' used above, we integrated it into another popular state-of-the-art routing flow, 'flow 2', which includes CUGR2.0 [11] and TritonRoute [27]. We compare the overall quality score under two different flows in TABLE IV. The results indicate that the ARSP framework improves solution quality for both widely-used routing flows across most designs, leveraging the $(1 - \frac{1}{e})$ approximation guarantee of the ARSP framework to achieve near-optimal solutions. Overall, the findings in TABLE IV demonstrate the robustness of our ARSP framework in enhancing the performance of existing state-of-the-art sequential routing flows.

TABLE IV Quality score comparison with two state-of-the-art routing flows, where 'Flow 1' represents CUGR2.0 [11]+Dr.CU2.0 [26] and 'Flow 2' is the flow with CUGR2.0 [11]+TritonRoute [27].

Design	Flow 1	Flow 1-ARSP	Flow 2	Flow 2-ARSP
18t5	16168206	**16150243**	15675469	**15622775**
18t5m	15814649	**15803299**	15672948	**15640007**
18t8	37552270	**37492777**	37526986	**37379797**
18t8m	36864567	**36806643**	36692008	**36658519**
18t10	40184353	**40158512**	39595790	**39583744**
18t10m	46554472	**46341569**	**44634666**	44642031
19t7	77830797	**77699910**	78236619	**78078160**
19t7m	71598698	**71446258**	71953520	**71922023**
19t8	114428150	**114063471**	120155443	**120023451**
19t8m	112383517	**112294927**	118177223	**118096122**
19t9	**177608525**	177706743	185305495	**185235959**
19t9m	173823653	**173702612**	180632034	**180577186**
Average	77018738	**76893580**	78688184	**78621648**

V. Conclusion

We presented ARSP, an adaptive routing framework that dynamically optimizes routing resources using submodular optimization. By formulating routing space selection as a submodular maximization problem, ARSP achieves intelligent resource allocation while maintaining efficiency. Comprehensive experimental results demonstrate that our ARSP framework significantly improve the design rule violation performance of existing state-of-the-art sequential routers. The framework's lightweight design enables seamless integration with existing routers, offering practical improvements without significant overhead. ARSP demonstrates how theoretical optimization principles can effectively address key challenges in physical design automation.

References

[1] N. A. Sherwani, *Algorithms for VLSI physical design automation*. Springer Science & Business Media, 2012.

[2] G.-J. Nam, C. Sze, and M. Yildiz, "The ISPD global routing benchmark suite," in *ACM International Symposium on Physical Design (ISPD)*, 2008, pp. 156–159.

[3] W.-H. Liu, S. Mantik, W.-K. Chow, Y. Ding, A. Farshidi, and G. Posser, "ISPD 2019 initial detailed routing contest and benchmark with advanced routing rules," in *ACM International Symposium on Physical Design (ISPD)*, 2019, pp. 147–151.

[4] X. Jiang, Z. Guo, Z. Chai, Y. Zhao, Y. Lin, R. Wang, and R. Huang, "Accelerating routability and timing optimization with open-source ai4eda dataset circuitnet and heterogeneous platforms," in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. IEEE, 2023, pp. 1–9.

[5] C.-K. Koh and P. H. Madden, "Manhattan or non-manhattan? a study of alternative vlsi routing architectures," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2000, pp. 47–52.

[6] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Pattern routing: Use and theory for increasing predictability and avoiding coupling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 21, no. 7, pp. 777–790, 2002.

[7] M. Pan and C. Chu, "FastRoute 2.0: A high-quality and efficient global router," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2007, pp. 250–255.

[8] S. Liu, Y. Pu, P. Liao, H. Wu, R. Zhang, Z. Chen, W. Lv, Y. Lin, and B. Yu, "Fastgr: Global routing on cpu-gpu with heterogeneous task graph scheduler," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[9] J. Liu, C.-W. Pui, F. Wang, and E. F. Young, "Cugr: Detailed-routability-driven 3d global routing with probabilistic resource model," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[10] J. He, U. Agarwal, Y. Yang, R. Manohar, and K. Pingali, "Sproute 2.0: A detailed-routability-driven deterministic parallel global router with soft capacity," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 586–591.

[11] J. Liu and E. F. Young, "EDGE: Efficient DAG-based Global Routing Engine," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[12] M. Cho and D. Z. Pan, "BoxRouter: A new global router based on box expansion and progressive ILP," in *ACM/IEEE Design Automation Conference (DAC)*, 2006, pp. 373–378.

[13] M. Cho, K. Lu, K. Yuan, and D. Z. Pan, "BoxRouter 2.0: A hybrid and robust global router with layer assignment for routability," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 14, no. 2, pp. 1–21, 2009.

[14] E. Shragowitz and S. Keel, "A global router based on a multicommodity flow model," *Integration*, vol. 5, no. 1, pp. 3–16, 1987.

[15] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[16] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1975, pp. 184–193.

[17] M. Pan and C. Chu, "FastRoute: A step to integrate global routing into placement," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2006, pp. 464–471.

[18] Y. Xu, Y. Zhang, and C. Chu, "FastRoute 4.0: Global router with efficient via minimization," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*. IEEE, 2009, pp. 576–581.

[19] M. Pan, Y. Xu, Y. Zhang, and C. Chu, "FastRoute: An efficient and high-quality global router," *VLSI Design*, vol. 2012, pp. 14–14, 2012.

[20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—I," *Mathematical programming*, vol. 14, pp. 265–294, 1978.

[21] S. Iwata, S.-i. Tanigawa, and Y. Yoshida, "Improved approximation algorithms for k-submodular function maximization," in *SIAM International Conference on Data Mining (SDM)*. SIAM, 2016, pp. 404–413.

[22] T. Horel and Y. Singer, "Maximization of approximately submodular functions," *Annual Conference on Neural Information Processing Systems (NIPS)*, vol. 29, 2016.

[23] L. Zheng, H. Chan, G. Loukides, and M. Li, "Maximizing approximately k-submodular functions," in *SIAM International Conference on Data Mining (SDM)*. SIAM, 2021, pp. 414–422.

[24] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*. IEEE, 2007, pp. 1–6.

[25] G. E. Blelloch, "Prefix sums and their applications," 1990.

[26] H. Li, G. Chen, B. Jiang, J. Chen, and E. F. Young, "Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–7.

[27] A. B. Kahng, L. Wang, and B. Xu, "Tritonroute: The open-source detailed router," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 3, pp. 547–559, 2020.

[28] P. Spindler and F. M. Johannes, "Fast and accurate routing demand estimation for efficient routability-driven placement," in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2007.

[29] S. Dolgov, A. Volkov, L. Wang, and B. Xu, "2019 cad contest: Lef/def based global routing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2019, pp. 1–4.