Feature Learning and Optimization in VLSI CAD

GENG, Hao

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
July 2021

Thesis Assessment Committee

Professor KING Kuo Chin Irwin (Chair)
Professor YU Bei (Thesis Supervisor)
Professor YOUNG Fung Yu (Committee Member)
Professor YANG Fan (External Examiner)

Abstract of thesis entitled:

Feature Learning and Optimization in VLSI CAD

Submitted by GENG, Hao

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in July 2021

As the technology node of integrated circuits rapidly scales down to 7nm and beyond, the electronic design automation (EDA) in very large-scale integration (VLSI), which has been developed over the last few decades, is challenged by the ever-increasing design complexity of integrated circuits (ICs). To combat progressively large IC design space more effectively and efficiently is in demand.

Recently, with the surge of artificial intelligence, increasingly sophisticated machine learning (ML) including deep learning (DL) techniques have been heavily introduced into our EDA community. These technologies innovate and accelerate the chip design from the front-end stage to the back-end stage. However, several critical issues emerge during the evolutional process, among which are extracting features and optimizing the associated attribute values of features in some applications in the EDA domain. Particularly, exceedingly relying on the domain knowledge, many of the ML-based EDA solutions still manually craft feature and corresponding attribute values, and the feature extraction is frequently isolated from the model training. On this account, the quality of the ML-based solutions is jeopardized.

This thesis outlines several methodologies alleviating the aforementioned issues in different applications at distinct stages of VLSI design. It is mainly composed of the following four themes.

- Revising the feature extraction method of ML techniques-based sub-resolution assist feature (SRAF) insertion, and developing the integer linear programming framework for insertion process at mask synthesis stage.
- Proposing a new deep learning-accelerated layout hotspot detector at the early phase of mask synthesis.
- Offering a sequential optimization-based framework with an elaborated feature extractor for high-speed adder design space exploration (DSE).
- Presenting a new multi-objective optimization framework for parameter tuning of a certain physical design tool.

We expect and believe that the thesis will contribute to feature learning and optimization in the EDA domain, and our ideas will enlighten future studies.

摘要

超大規模集成電路（VLSI）中的電子設計自動化（EDA）技術已歷經數十年的發展。而伴隨著集成電路工藝節點快速演變到 7nm 以下，與日俱增的電路設計複雜性向 EDA 技術發起了挑戰。因此，我們需要更有效的技術去應對超大規模的集成電路（ICs）設計空間。

近年來，隨著人工智能浪潮的興起，包括深度學習（DL）在內的機器學習 (ML) 技術愈發成熟，並且已被大量引入 EDA 領域。這些技術革新並加速了從前端到後端的整個芯片設計流程。然而，一些關鍵問題逐步凸顯在人工智能技術引進的過程中。其中就包括在 EDA 的相關應用中，如何提取特徵以及優化特徵的屬性值。尤其是，很多 EDA 領域的工作雖然採用了機器學習算法，但嚴重依賴於專家知識去人為的設計特徵和屬性值。並且，還孤立地看待特徵的提取和模型的學習過程。這大大降低了這些工作的性能。

本論文提出了在 VLSI 設計的不同階段及不同應用中緩解上述問題的幾種方法論。主要由以下四個主題組成。

- 在掩膜版合成階段，改進了基於機器學習技術的亞分辨率輔助特徵（SRAF）插入的特徵提取方法，並開發了基於整數線性規劃的 SRAF 插入流程。
- 在掩膜版合成早期階段，提出一種新的基於深度學習技術加速的版圖熱點檢測器。
- 提出了一個基於連續優化技術的加法器設計空間探索架構並設計了相應地特徵提取技術。
- 面對特定物理設計工具的參數調優問題，提出了一種新的多

目標優化框架。

我們期望並且也相信該論文將有助於 EDA 領域的特徵學習和優化，同時會對未來的研究有所啟迪。

Acknowledgement

I would like to express my forever and sincere appreciation to my supervisor, Prof. Bei YU, for his sophisticated guidance, sparkling enlightenment, sustained support. Without his supervision, patience and encouragement, the completion of this thesis and my Ph.D. studies seems to be impossible.

Taking this opportunity, I also would like to greatly thank my committee members, Professor Irwin KING, Professor Evangeline F.Y. YOUNG and Prof. Fan YANG, who have offered useful feedback and thoughtful suggestions throughout the process.

Highly thanks to Prof. Lizhe WANG for introducing me to scientific research about eight years ago and for continuously inspiring and helping me to pursue research.

It has been a great honor to work with many great collaborators during my Ph.D. studies. I highly appreciate Prof. Xuan ZENG and Prof. Fan YANG for plenty of cooperation on DFM topics. Thanks to Dr. Haoyu YANG and Dr. Yuzhe MA for fascinating collaborations on the topics of mask synthesis and design space exploration. Thanks to Tinghuan CHEN, Qi SUN, Lu ZHANG and Ran CHEN for heaps of constructive discussions and cooperation. I would like to thank other intelligent and accomplished collaborators, Dr. Qi XU, Dr. Jin MIAO, Dr. Subhendu ROY, Dr. Joydeep MITRA, Dr. Xiaoqing Xu, Prof. Wei ZHONG, Dr. Xingquan LI, Ziyang YU, Binwu ZHU, Peiyu LIAO and all other CUDA members.

I would like to extend my appreciation to the staff in the CSE department for generously helping me.

I owe the deepest gratitude to my parents and my beloved wife for their ever-encouraging and -supporting me to pursue what I am interested in.

This work is dedicated to my ever supportive and loving family.
Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Challenges and Motivations

With the ever-shrinking of advanced technology nodes, the community of electronic design automation (EDA) in very large-scale integration (VLSI) has witnessed a dramatic leap in design complexity recently. The challenge from the scalability, reliability of the chip design flow (shown in Fig. 1.1) drives EDA techniques to be upgraded rapidly. The recent advances of machine learning (ML) methods have shown the transcendent potential to be incorporated into techniques to tackle EDA tasks. Essentially, ML techniques including DL methods have already been heavily introduced into the EDA domain. These technologies innovate and accelerate the chip design from the front-end stage to the back-end stage. Nevertheless, a few crucial challenges appear during the accelerating process, which are feature extracting and optimizing the associated attribute values of features in design space exploration (DSE) applications. Specifically, excessively relying on the domain knowledge, many of the ML-based EDA arts still exploit hand-crafted feature design and attribute values. What's worse, the feature extraction is frequently independent of the model training. It is conspicuous that effective, automatic feature learning or extraction will enable the downstream learning tasks, and jointly handling the extraction and

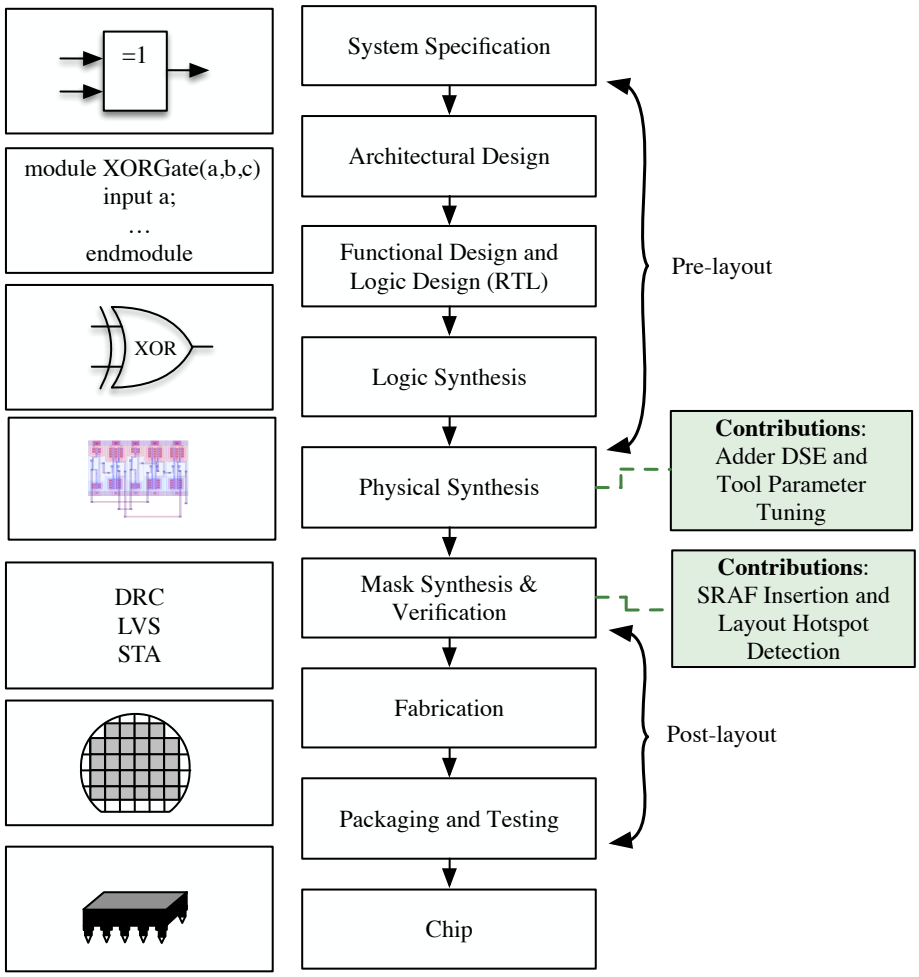


Figure 1.1 The chip design flow (partially referencing [70]).

model calibration will benefit the whole training step. Besides, developing an appropriate optimization method for some applications like design space exploration (DSE) for CAD tool parameter tuning will increase the quality as well as the yield of chips and further speed up the whole chip design process.

In the following, we will present the current research status of some post-layout applications to exemplify how the ML techniques aid academia and industry to address the specific EDA issues and the importance of feature learning.

Under current advanced technology nodes, lithographic process

variations emerge more commonly in the manufacturing process. It will lead to manufacturing defects and decrease yield. Although lithographic simulation is able to generate fabrication results accurately, it suffers from great runtime consumption. To address these problems, different kinds of approaches are proposed. One is mask optimization through various resolution enhancement techniques (RETs) [8, 23, 45, 54, 55, 71, 93, 153, 157, 160]. This strategy is of importance in patterning and litho-friendly layout design [121], which can improve the yield of semiconductors. Sub-resolution assist feature (SRAF) [142] is a representative strategy of numerous RET techniques. There are plenty of algorithms to generate SRAFs such as rule-based [76], model-based [120], traditional machine learning model-based approach [54,55,152] and deep learning-based solutions like [8]. Rule-based SRAF generation method is very fast, but it is hard to define and extract rules from model-based SRAFs. Consequently, the performance can not be guaranteed. Although the model-based method is more accurate, it is time-intensive. Furthermore, it is hard for conventional algorithms to generate consistent SRAF patterns and may require too much engineering effort. By contrast, the machine learning model-based scheme can faster and more precisely obtain consistent SRAF patterns.

Another way to lighten lithographic process variations, especially for some sensitive layout patterns, is so-called hotspot detection. Many methods such as pattern matching-based [20, 137, 145, 163], machine learning model-based [102, 113, 130, 159, 168] and recently convolutional neural network (CNN) model-based [25, 25, 28, 33, 52, 72, 155] hotspot detection algorithms are proposed. Pattern matching provides speedup in comparison with lithographic simulation. However, it is merely applicable to detect already known or similar patterns and has a poor hotspot recognition rate on unknown patterns. The approaches based on machine learning, especially deep learning techniques have been able to achieve reasonably good results for hotspot detection with less time consumption.

In these ML-based post-layout tasks, measuring the similarity among different layout designs is extremely imperative and meanwhile involved in almost all applications in the field [22, 51, 70, 151]. Capturing and representing the intrinsic characteristics such as topological information of a layout is the kernel to resolve the problem. Since pattern intuitively describes and summarizes two-dimensional polygon configurations in a layout design, pattern-based schemes are widely used in layout design. For instance, design rule check (DRC) plus exploiting a library of patterns to identify problematic 2D patterns, has been proven to be effective [131]. Nevertheless, as integrated circuit feature sizes continue to decrease, patterning technology may have a poor process margin [21]. Additionally, the number of patterns increases dramatically, which brings about challenges in identifying, organizing, and carrying forward the learning of each pattern from test layout designs to mature products. In one aspect, to a machine learning model, the features directly affect the quality of regression and prediction. Namely, with more representative and generalized features, the learning model has better performance on approximation and prediction. Besides, the better-extracted features can avoid overfitting to some extent. Therefore, the problem of how to extract characteristics from numerous layout patterns properly demands prompt solutions.

1.2 Thesis Structure and Contributions

This thesis attempts to investigate several methodologies lightening the aforementioned issues mainly at physical synthesis (pre-layout) and mask synthesis (post-layout) stages. The corresponding contributions are illustrated in Fig. 1.1.

The first contribution of the thesis is revising the feature extraction method in ML techniques-based sub-resolution assist feature (SRAF) insertion, and developing the integer linear programming framework for the insertion process [54, 55]. In modern VLSI design

flow, SRAF insertion as part of mask synthesis is one of the RETs to improve chip manufacturing yield. With aggressive feature size continuously scaling down, layout feature learning becomes extremely critical. In this thesis, for the first time, we enhance conventional manual feature construction, by proposing a supervised online dictionary learning algorithm for simultaneous feature extraction and dimensionality reduction. By taking advantage of label information, the proposed dictionary learning framework can discriminatively and accurately represent the input data. We further consider SRAF design rules in a global view, and design two integer linear programming models in the post-processing stage of the SRAF insertion framework. Experimental results demonstrate that, compared with a state-of-the-art SRAF insertion tool, our insertion design not only boosts the performance of the machine learning model, but also improves the mask optimization quality in terms of edge placement error (EPE) and process variation (PV) band area.

The second contribution of the thesis is proposing a new DL-accelerated hotspot detection framework at the early stage of mask synthesis [52, 53]. With the feature size of circuits entering the nanometer era, hotspot detection has become a crucial and challenging problem in the generation of optimized mask design for better printability. ML techniques, especially DL, have attained notable success on hotspot detection tasks as well. However, most existing hotspot detectors suffer from suboptimal performance due to two-stage flow and less efficient representations of layout features. What is more, most works can only solve simple benchmarks with apparent hotspot patterns like ICCAD 2012 Contest benchmarks. In this thesis, we firstly develop a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are jointly performed. An attention mechanism-based deep convolutional neural network is exploited as the backbone to learn embeddings for layout features and classify the hotspots simultaneously. Experimental results demonstrate that our detection framework achieves

accuracy improvement over prior arts with fewer false alarms and faster inference speed on much more challenging benchmarks.

The third contribution is offering a sequential optimization framework with an elaborated feature extractor for 64-bit high-speed adders in the application of adder DSE [49]. As we know, adders are the primary components in the data-path logic of a microprocessor, and thus adder design has been always a critical issue in the VLSI industry. However, it is infeasible for designers to obtain optimal adder architecture by exhaustively running EDA flow due to the extremely large design space. Previous arts have proposed the machine learning-based framework to explore the design space. Nevertheless, they fall into sub-optimality due to a two-stage flow of the learning process and less efficient nor effective feature representations of prefix adder structures. In this thesis, we first integrate a variational graph auto-encoder and a neural process into an end-to-end, multi-branch framework which is termed as a *graph neural process*. The former performs automatic feature learning of prefix adder structures, whilst the latter one is designed as an alternative to the Gaussian process. Then, we propose a sequential optimization framework with the graph neural process as the surrogate model to explore the Pareto-optimal prefix adder structures with trade-offs among quality-of-result (QoR) metrics like power, area, and delay. Experimental results show that, compared with state-of-the-art methodologies, our framework can achieve a much better Pareto frontier in multiple QoR metric spaces with fewer design-flow evaluations.

Our fourth contribution is investigating a new optimization framework for parameter tuning of a certain physical design tool. Physical design flow through physical design tools plays a critical role in advanced integrated circuit design. Mostly, the parameters fed into physical design tools are mainly manually determined based on the domain knowledge of the experts. Nevertheless, owing to the ever-shrinking scaling down of technology nodes and the complexity of

the design space spanned by combinations of the parameters, even coupled with the time-consuming simulation process, such manual explorations for parameter configurations of physical design tools have become extremely laborious. A few works are proposed in the field of design flow parameter tuning. However, very limited prior arts explore the complex correlations among multiple quality-of-result (QoR) metrics of interest (e.g., delay, power and area) and explicitly optimize these goals simultaneously. To overcome these weaknesses and seek effective parameter settings (or values) of physical design tools, in this thesis, we introduce a multi-objective Bayesian optimization framework with a multi-task Gaussian model as the surrogate model. An information gain-based acquisition function is adopted to sequentially choose candidates for tool simulation to efficiently approximate the Pareto-optimal parameter configurations. The experimental results on two industrial benchmarks under the 7nm technology node prove the superiority of the proposed framework compared to the cutting-edge works.

The structure of the thesis is organized as follows. Chapter 2 provides related backgrounds about some common features designed in different ML-aided design flow stages. Chapter 3 covers the first contribution with corresponding technique details, while the second contribution is introduced in Chapter 4. Chapter 5 describes the the third contribution, and the last contribution is shown in Chapter 6. Chapter 7 summarizes this thesis and delivers the possible future study directions with Appendix A supplementing the associated derivation and proof of equations.

□ **End of chapter.**

Chapter 2

Literature Review

In this chapter, the features learning methods in machine learning techniques-based solutions for different EDA tasks will be introduced. Besides, the developing status of four specific EDA applications covered by the thesis (i.e. SRAF insertion, layout hotspot detection, high speed adder design exploration and parameter tuning of physical design tool) will be discussed as well.

2.1 Feature Learning in ML-based Post-Layout Applications

In a large amount of ML-based post-layout applications, layout feature extraction is indispensable. To describe layout efficiently and effectively, numerous feature extraction methods have been proposed in previous works.

2.1.1 Density-based Feature

The density-based feature [104, 145] summarizes the local pattern density of a layout grid-wisely. It is plausible to measure mask printability since layouts with high pattern density have a higher risk of suffering defects. The essence behind this feature is first placing a layout onto a canvas full of few squared grids, and then

calculate the ratio between the area occupied by the pattern in one grid and the associated grid area. Compared with other features, the density-based feature vectors are prone to be separated in low-dimension space. It benefits the training and inference of a machine learning model. However, this rough idea will lead to global information loss and degeneration of a machine learning model in high-dimension space. In some extreme cases, the extraction method even extracts the same feature vectors from different patterns. Hence, a modified version of density-based feature extracted by the local grid density differential (LGDD) method is proposed in [167]. The traditional density-based feature just calculates the density value of a gird. However, the new scheme sets triangles locating at 4 corners of a grid as sampling area and concatenates density values from different sampling areas to form a feature vector. But the longer feature dimensionality may increase the risk of overfitting. To relieve the overfitting problem, some extended approaches are investigated. In [102], the parameters like grid size and window size of a density-based feature are optimized by maximizing the Mahalanobis distance [97] between non-hotspot and hotspot features. The principal component analysis (PCA) method [75] is exploited to reduce the dimensionality of feature vectors. However, information loss is inevitable since non-principal components are ignored. In addition, there also exists a problem that how many principal components need to be kept. To give a direct understanding, the density feature is shown in Fig. 2.1(b).

2.1.2 Spectrum-based Feature

As illustrated in Fig. 2.1(c), spectrum-based feature applies frequency domain transforms such as the discrete Fourier transform (DFT), discrete cosine transform (DCT). In [104, 123], a feature vector consists of the coefficients of Fourier transform of a layout pattern. Since the feature reflects an effect due to projection op-

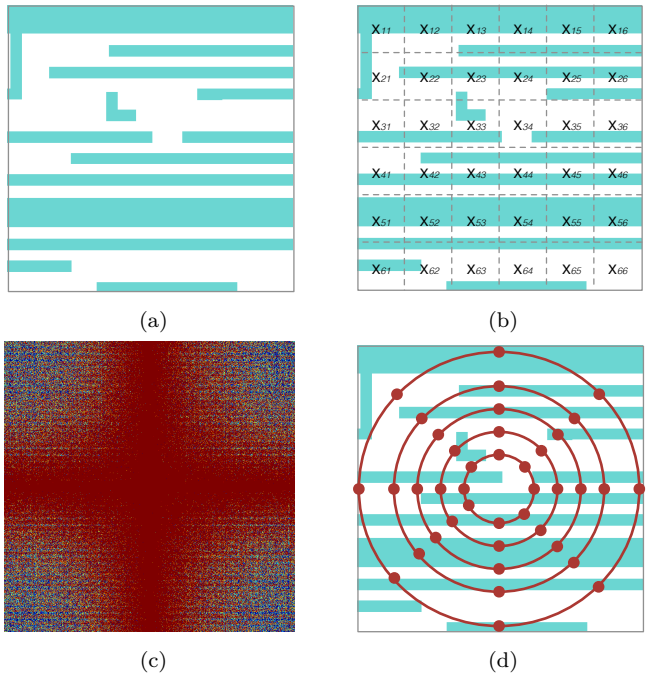


Figure 2.1 The visualizations of layout features: (a) Original layout; (b) Density-based feature; (c) Spectrum-based feature; (d) CCAS feature.

tics, it is expected to benefit a machine learning model with highly accurate prediction. In addition, in [123], the feature has made a remarkable success on reflected and shifted patterns.

After achieving success on wafer clustering tasks [122, 172], DCT coefficients are also exploited as the input of convolutional neural network (CNN) [25, 33, 155, 156]. Compared with raw layouts as inputs, CNN with DCT as inputs can achieve higher accuracy. The success of DCT is that the extracted features will be easier to obtain high sparsity and global representations than raw images. The defect is that the feature extraction process lacks guided information from the follow-up network training owing to the independence between them.

2.1.3 Concentric Circle Area Sampling Feature

Recently proposed concentric circle area sampling (CCAS) [103] is developed from concentric square sampling (CSS) [60]. It takes advantage of layout properties and lithography process, thus has made considerable improvements on hotspot detection accuracy. Meanwhile, because of reflecting light diffraction effects, CCAS layout features are also exerted to generate SRAFs in [152]. With considering concentric propagation of diffracted light from mask patterns, the core method of CCAS is sub-sampling on concentric circles. However, since adjacent circles contain similar information, the CCAS features have much redundant information. The redundancy will result in several problems like hindering the fitting of a machine learning model. As a result, the concentric circle sampling (CCS) method which exploits the mutual information to select import circles of CCAS is proposed in [168]. The objective of circle selection is to maximize the dependency of selected circles on the corresponding classification variable. Because of reducing redundancy of CCAS, CCS is benefit for calibrating machine learning models. Fig. 2.1(d) displays the CCAS feature extraction method.

2.1.4 Other Successful Features

Besides the above features, there are many kinds of other successful features such as modified transitive closure graph representation [163], fragmentation-based context characterization feature [40,161], histogram of oriented light propagation (HOLP) [134], improved tangent space-based characterization [61] and so on. In [163], the authors modified the transitive closure graph (TCG) [88] method to extract critical topological features within a pattern. Meanwhile, in [61], the improved tangent space representation which reflects the radius and angle of a polygon in a layout clip has been proposed. Considering the geometric shape of a layout pattern and the combined impact of its neighboring patterns, Yu *et al.* inves-

tigated a fragmentation-based feature [161] consisting of important information such as pattern shapes, the distance between patterns and corner information (convex or concave). Recently, inspired by the histogram of oriented gradient (HOG) [37], HOLP [134] feature which reflects light propagation in the exposure process has been presented. This feature is robust to small shifts of layout patterns.

Currently, a large quantity of state-of-the-art works based on deep neural networks (DNNs) [8, 23, 28, 52, 72, 153] treat the post-layout tasks as image-related issues. After translating the layouts into images, the networks are directly fed with the images. Very recently, Li *et al.* propose point-wise layout features in a two-stage hotspot detector, where each layout feature is decomposed to discrete points [85].

2.2 Feature Learning in ML-based Pre-Layout Applications

In oceans of ML-based pre-layout applications, features can be coarsely categorized according to the extraction paradigms: manual design and automatic design.

First, let us exemplify manual feature design in EDA applications. In high-level synthesis (HLS, which translates C/C++/System C-based specifications to hardware description languages such as VHDL or Verilog) applications, features (e.g. clock periods, resources, memory, etc.) are usually manually selected and extract from the HLS reports [36, 99]. Xie *et al.* first propose a RouteNet [149] to apply CNN for design rule checking (DRC) hotspot detection at the placement synthesis phase. The input features are elaborately designed by hand, which contain the information about density distribution of macros, cells, DRC violations, and so forth. Another example is the work [94] in adder design space exploration (covering logic synthesis and physical synthesis stages). It exploits the two-stage framework where the feature extractor for adders and sub-

sequent machine learning models are separated. The feature representations (e.g. sum-path-fan-out, maximum-fan-out, node size) are determined manually to characterize the adder structure.

Regarding the examples of automatic feature extraction, the very recent work for floorplanning [107] exploits the graph neural network to extract the features from netlists. In [87] for DRC hotspot prediction, the input features are images translated from pins and macros density. [7] first casts the routing congestion prediction for large-scale FPGAs to an image translating issue, and then exploits conditional GAN to combat it. Chen *et al.* investigate a practical plug-in for routability optimization, where the placement results are converted into images like horizontal/vertical track capacity map, cell density map, and cell pin density map, etc. Then these features in image format are sent into fully convolution networks to predict routing congestion. For the work on the clock tree prediction and optimization, [91] regards flip flop distribution, clock net distribution, and trial routing results as input images. In [86] to construct routing tree, Li *et al.* consider the pins of a net as point cloud and develop a deep neural network termed as TreeNet to acquire the feature embedding of the point cloud. In addition, several arts [5, 82, 138, 148], which regard the parameters as features to be encoded and automatic determine the attribute values, are proposed to concentrate on design flow parameter tuning.

To sum up, the feature learning techniques exploited in ML-based EDA applications remain at the coarse level (e.g. mainly handcrafted without automatic learning).

2.3 Survey on Four Specific EDA Applications

2.3.1 Case I: SRAF insertion

As feature size of semiconductors enters nanometer era, lithographic process variations are emerging as more severe issues in chip man-

ufacturing process. That is, these process variations may result in manufacturing defects and a decrease of yield. Besides some design for manufacturability (DFM) approaches such as multiple patterning and litho-friendly layout design [109, 121], a de facto solution alleviating variations is mask optimization through various resolution enhancement techniques (RETs) (e.g. [8, 12, 23, 24, 45, 54, 55, 93, 153, 157, 164]). Optical proximity correction (OPC) is the most successful representative strategy among numerous RETs, which aims at compensating lithography proximity effects by correcting mask pattern shapes and inserting assist features.

Although conventional OPC can size the mask to give the correct critical dimension (CD) on the wafer, it cannot make the isolated target pattern become dense [96]. As a result, sub-resolution assist feature (SRAF) [142] insertion was proposed. Without printing SRAF patterns themselves, the small SRAF patterns can transfer light to the positions of target patterns, and therefore SRAFs are able to improve the robustness of the target patterns under different lithographic variations. A lithographic simulation example demonstrating the benefit of SRAF insertion is illustrated in Fig. 2.2. Here process variation (PV) band [152] (i.e. yellow circuit) area is applied to measure the performance of lithographic process window. As a matter of fact, the smaller area of the PV band, the better printing performance. In Fig. 2.2(a), only OPC is conducted to improve the printability of the target pattern, while in Fig. 2.2(b) both SRAF insertion and OPC are applied. We can see that, through SRAF insertion, the PV band area of the printed target pattern is effectively reduced from 2688 nm^2 as in Fig. 2.2(a) to 2318 nm^2 as in Fig. 2.2(b).

There is a wealth of literature on the topic of SRAF insertion for mask optimization, which can be roughly divided into three categories: rule-based approach [76], model-based approach [120], and machine learning-based approach [152]. Rule-based approach is able to achieve high performance on simple designs, but it cannot handle

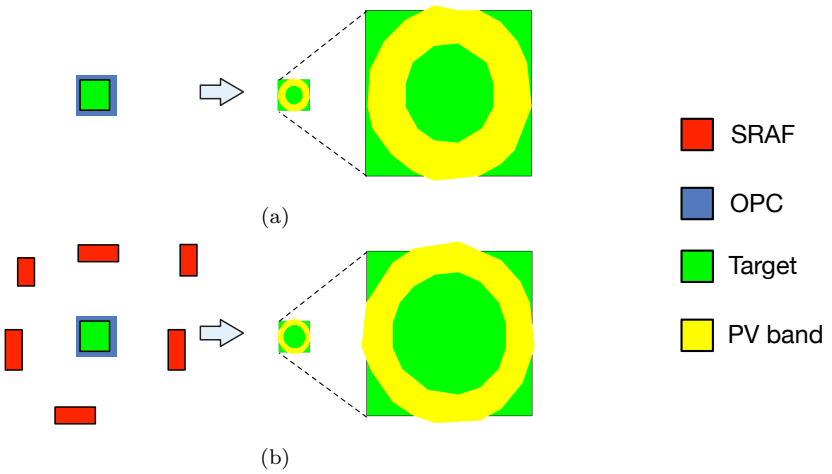


Figure 2.2 The impact of SRAF insertion on lithography: (a) Printing with OPC only (2688 nm^2 PV band area); (b) Printing with both OPC and SRAF (2318 nm^2 PV band area).

complicated target patterns. Although model-based approach has a better performance, it is unfortunately very time-consuming. Recently, Xu *et al.* investigated an SRAF insertion framework based on machine learning techniques [152]. After trained on the training data set, the machine learning model draws inferences that can guide SRAF insertion on testing data. However, on account of coarse feature extraction techniques and lack of a global view of SRAF designs, the lithographic simulation results may not be good enough.

2.3.2 Case II: Layout Hotspot Detection

As the technology node of integrated circuits scales down to $7nm$ and beyond, the techniques for lithographic processes are supposed to manage the ever-shrinking feature size. However, owing to the delayed progress of lithography techniques, lithographic processes variations emerge during the manufacturing, and thus lead to yield loss. For example, lower fidelity patterns on a wafer (a.k.a. lithography hotspot) is one of the crucial issues. Many techniques of early detection for hotspots have been proposed to ensure manufacturabil-

ity. The prevalently applied lithography simulation technique could attain high accuracy, nevertheless, it is also known to be pretty time-consuming. Therefore, two other types of quick detection methods are proposed as alternatives to lithography simulation. One is based on pattern matching [89,137,145], and the other is machine learning-driven. The pattern matching approaches take input pre-defined pattern library of known hotspots, but they cannot detect unknown hotspots. Fortunately, detection methods which are built upon machine learning methodologies [40,51,102,130,154,159,161,168], and particularly deep learning techniques [25–28,33,52,72,155,173], are able to offer fast and accurate solutions to both known and unknown hotspot patterns.

Motivated by the recent advances of deep learning in other domains like computer vision, Yang *et al.* firstly proposed the hotspot detector based on a shallow convolutional neural network (CNN), where layout clips were firstly converted into the frequency domain via discrete cosine transform (DCT) before fed into networks [155]. To extract DCT features in different scales, the inception modules are introduced in the work [25] to modify the structure of neural networks. In [33], to overcome the limitation imposed by the number of labelled hotspots or non-hotspots, the concept of semi-supervised learning was introduced so that unlabeled data can be harnessed as well. By considering input layout clips as binary images, Jiang *et al.* employed a binarized neural network to further speed up the detection process [72].

2.3.3 Case III: Adder Design Space Exploration

Very large scale integration (VLSI) design methodologies have developed for about 50 years, from manually-crafted design to computer-aided design (CAD) with increasingly higher levels of design specifications. Unfortunately, with the aggressive and amazing scaling down of semiconductor technology nodes, design complexity in-

creases dramatically. As a result, efficient design space exploration (DSE) [11, 62, 84, 90, 92, 94, 111, 116, 126, 127, 170, 171] has emerged as a promising solution due to the exponentially increasing size of design space of microprocessors and the consequent time-consuming synthesis runs.

Quintessential DSE optimizes a single objective when other objectives are considered as constraints. But, for VLSI design, exploration in multiple quality-of-result (QoR) metrics space such as performance, power and area is required, which naturally involves trade-offs. In this case, multi-objective DSE seeks optimal solutions [94, 116] which trade off multiple objectives rather than finding one single optimal point. In other words, without any further information, none of these Pareto-optimal points can be regarded as being better than the others in all the objectives simultaneously. Nevertheless, obtaining a set of Pareto-optimal points with an acceptable cost is a big challenge. One main reason is that the objective functions are often unknown, and can only be determined through a point-wise evaluation which is expensive and necessitates large computational and time resources. Existing EDA design flow, including synthesis tool and physical design tool, can only return one implementation per call and even worse, the implementation is not guaranteed to be in a Pareto set. Therefore, it is not difficult to imagine that searching for the Pareto-optimal set is time-consuming since massive evaluations are required. In summary, the goal of multi-objective DSE is to find a set of Pareto-optimal points in as few evaluations of the multiple objective functions as possible so that the total expense is minimized.

In the VLSI domain, DSE is widely exploited but not limited to handling analog circuit synthesis [92, 171], FPGA CAD flow [90], FPGA HLS directives design optimization [127], DNN hardware deployment [126], processor architecture design [11], adder design [94, 116]. To search for Pareto-optimal solutions as efficiently and effectively as possible, most of them are based on machine learning

techniques. Although the above DSE technologies have achieved great success in different scenarios, most of them utilize the Gaussian process which has a high computational complexity as the regressor for performance estimation.

In this thesis, we focus on DSE techniques for adder design. Adder design is one of the fundamental problems in VLSI, where designing carry-propagation units plays the most critical role. Although the unit can be implemented by enormous parallel prefix structures, real synthesis and physical design running are still needed. Regular adder structures proposed in [80, 124] and some recently developed works [101, 115] which try to generate a single prefix adder network under a set of structural constraints cannot cover a large design space yet. By incorporating two pruning techniques in the prior, Ma *et al.* [94] proposed a state-of-the-art algorithm to generate the prefix graph structures, and exploit an active learning-based optimization model to explore Pareto-optimal adder designs.

2.3.4 Case IV: Parameter Tuning of Physical Design Tool

Over the past decades, both academia and industry have made great efforts on studying physical design flow which is the core of modern VLSI design. As a result, several physical design tools with dozens of sophisticated algorithms incorporated are developed to improve chip productivity, design quality and reduce time-to-market. Manual configuring the input parameters of physical design tools relies on expertise and domain knowledge, which may be time-intensive and un-robust. Therefore, automatic parameter tuning is highly desirable.

Recently, several works [5, 82, 138, 148] focus on design flow parameter tuning. For example, Ustun *et al.* [138] accelerate FPGA design closure by utilizing XGBoost [132] regressor-based machine learning tuning framework with design-specific features extracted

from early stages of the design flow as guidance. [82] uses a tensor decomposition-based recommender system to tune parameters for macros. Agnesina *et al.* [5] optimize the placement parameters via a deep reinforcement learning framework fed with a mixture of handcrafted features covering graph topology theory along and graph embeddings generated using Graph Neural Networks. In [148], the active learning-based approach leveraging the feature importance sampling and XGBoost regressor is proposed for automatic parameter tuning. Although these pioneered parameter tuning frameworks have started the research explorations in this field and achieved some progress, they have their own limitations. [138] only considers the single QoR metric, while deep reinforcement learning framework utilized in [5] is hard to train and requires a large amount of training data. Neither [82] nor [148] explicitly explores the correlations among QoR metrics to be optimized, which may lead to performance degradation.

It is worth mentioning that the parameter tuning issue often involves optimizing multiple QoR metrics simultaneously. However, these QoR metrics (e.g. delay, power and area) are usually in conflict or coupled. The trade-offs naturally needs to be considered. What is more, "white-box" and "Black-box" optimization problems emerge. For "white-box" optimization problems, the objective functions have explicit formulations, which can be directly addressed by meta-heuristics like evolutionary strategies [38] and gradient-based optimization methods [39]. Nevertheless, there are no explicit functions for optimized objectives in "Black-box" problems. Parameter tuning of physical design tools is required to combat the "Black-box" optimization. To a certain extent, the physical design tool parameter tuning issue, which can be treated as parameter space exploration, is analogous to design space exploration (DSE). Thereby, some QoR metrics optimization frameworks developed in DSE may have the potential to be harnessed for reference. Some arts like [95] exploit single-objective Bayesian optimization (BO) framework with

a scalarization trick to handle multiple objectives. In these works, scalar weights of multi-objective functions are commonly sampled from a uniform distribution to construct a single-objective-like function. However, they are not really designed to seek near Pareto-optimal solutions to real QoR metrics (e.g. area vs. delay, power vs. delay and area vs. delay vs. power). Pareto active learning (PAL) based works [94] try to classify the input points based on the learned models into three classes: Pareto-optimal, non-Pareto-optimal, and uncertain. In each iteration, they select the candidate input for evaluation towards the goal of minimizing the size of uncertain set. Although PAL flow provides theoretical guarantees, it is only applicable to input space with finite dataset of discrete points.

□ **End of chapter.**

Chapter 3

SRAF Insertion

3.1 Introduction and Motivation

In a learning-based SRAF insertion flow, firstly, features are extracted from raw layout clips. One of the key takeaways of previous arts is the importance of features extracted from clips that leverage prior gained knowledge to achieve expected results. Namely, with more representative, generalized and discriminative layout features, the calibrated model can perform better. In this paper, we argue that the label information utilized in learning stage can be further imposed in feature extraction stage, which in turn will benefit the learning counterpart. In accordance with this argument, we propose a supervised online dictionary learning algorithm, which converts features from a high-dimension space into a low-dimension space with label information integrated into the feature representations at the same time. To the best of our knowledge, this is the first layout feature extraction work seamlessly combining with label information. There is no prior art in applying the dictionary learning techniques or further supervised dictionary approaches into SRAF insertion issue. Our main contributions are listed as follows.

- Leverage supervised online dictionary learning algorithm to handle a large amount of layout patterns.

- Our proposed feature is more discriminative and representative, and is embedded into SRAF insertion framework.
- The SRAF insertion with design rules is modeled as an integer linear programming (ILP) problem, and two ILP models are proposed.
- Experimental results show that our method not only boosts the performance of the machine learning model, but also improves the mask optimization quality.

The rest of this chapter is organized as following. Section 5.2.1 introduces some preliminaries on metrics, problem formulation in the paper and illustrates the whole working flow of our framework to insert SRAFs. Section 3.3 describes the specific feature extraction method, and our supervised online dictionary learning algorithm. Section 3.4 reveals two ILP models in post-processing stage. Section 6.5 presents the experiment results, followed by the summary in Section 6.6.

3.2 Preliminaries

3.2.1 Problem Formulation

Given a machine learning model, F_1 score is used to measure its accuracy. Specifically, the higher, the better. In addition, we employ PV band area and edge placement error (EPE) to quantify lithographic simulation results.

Definition 1 (F_1 Score [58]). *F_1 score is the harmonic mean of two metrics, precision and recall. Precision is the number of true positive results divided by the number of all positive results, while recall is the ratio between number of true positive results and the number of positive results should be returned.*

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}. \tag{3.1}$$

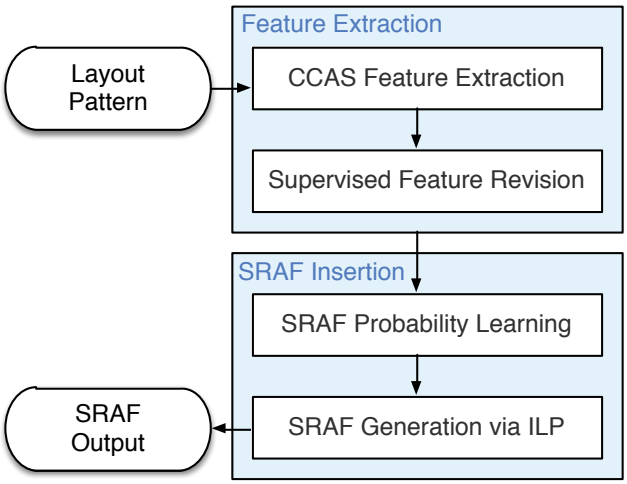


Figure 3.1 The proposed SRAF generation flow.

Definition 2 (PV Band)**.** *Given a lithography simulation contours at a set of process conditions, the process variation (PV) band is the area between the outer contour and inner contour.*

Definition 3 (EPE [152])**.** *Given a lithography simulation contour at the nominal condition, the edge placement error (EPE) is defined as the displacement between the target pattern contour and the nominal contour.*

Based on the above metrics, we define the SRAF insertion problem as follows.

Problem 1 (SRAF Insertion)**.** *Given a training set of layout clips and specific SRAF design rules, the objective of SRAF insertion is to place SRAFs in the testing set of layout clips such that the corresponding PV band and the EPE are minimized.*

3.2.2 Overall Flow

The overall flow of our proposed SRAF insertion is shown in Fig. 3.1, which consists of two stages: feature extraction and SRAF insertion. In the feature extraction stage, after feature extraction via concentric circle area sampling (CCAS), we propose supervised fea-

ture revision, namely, mapping features into a discriminative low-dimension space. A dictionary consists of atoms which are the representatives of the original features. Sparsely encoded over a well-trained dictionary, the original features are described as combinations of atoms. Due to space transformation, the new features (i.e. sparse codes) are more abstract and discriminative with negligible information loss for classification. Therefore, proposed supervised feature revision is expected to avoid over-fitting of a machine learning model. In the second stage, based on the predictions inferred by learning model, SRAF insertion can be treated as a mathematical optimization problem accompanied by taking design rules into consideration.

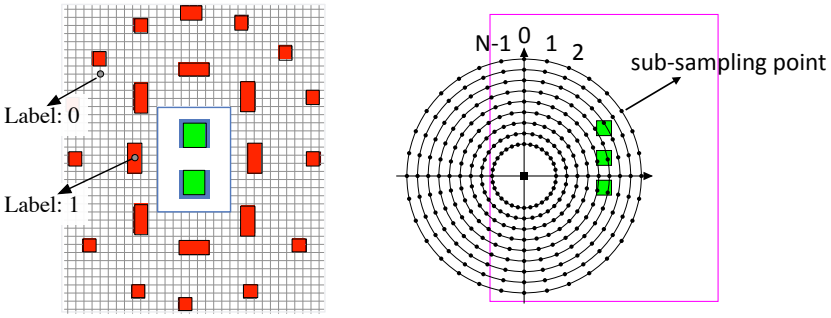
3.3 Feature Extraction

In this section, we firstly introduce the CCAS feature extraction method, and supervised feature revision. By the end, we give the details about our supervised online dictionary learning algorithm and corresponding analysis.

3.3.1 CCAS Feature Extraction

With considering concentric propagation of diffracted light from mask patterns, recently proposed CCAS [103] layout feature is used in SRAF generation domain.

In SRAF insertion, the raw training data set is made up of a set of layout clips which include a set of target patterns and model-based SRAFs. Each layout clip is put on a 2-D grid plane with a specific grid size so that real training samples can be extracted via CCAS method at each grid. For every sample, according to the model-based SRAFs, the corresponding label is either “1” or “0”. As Fig. 3.2(a) illustrates, “1” means inserting an SRAF at this grid, while “0” indicates there is no assist feature. Fig. 3.2(b) shows the CCAS feature extraction method.



(a) (b)

Figure 3.2 The CCAS feature extraction method: (a) SRAF label; (b) CCAS for feature extraction in machine learning model-based SRAF generation.

However, since adjacent circles contain similar information, the CCAS feature has much redundancy. In fact, the redundancy will hinder the fitting of a machine learning model.

3.3.2 Supervised Feature Revision

With CCAS features as inputs, the dictionary learning model is expected to output the discriminative feature of low-dimension. In the topic of data representation [43], a self-adaptive dictionary learning model can sparsely and accurately represent data as linear combinations of atoms (i.e., columns) from a dictionary matrix. This model reveals the intrinsic characteristics of raw data.

In recent arts, sparse decomposition and dictionary construction are coupled in a self-adaptive dictionary learning framework. As a result, the framework can be modeled as an unconstrained optimization problem. The joint objective function of a self-adaptive dictionary model for feature revision problem is proposed as Equation (3.2):

$$\min_{\boldsymbol{x},\boldsymbol{D}} \frac{1}{N} \sum_{t=1}^{N} \{ \frac{1}{2} \left\| \boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t \right\|_2^2 + \lambda \left\| \boldsymbol{x}_t \right\|_p \}, \tag{3.2}$$

where $\boldsymbol{y}_t \in \mathbb{R}^n$ is an input CCAS feature vector, and $\boldsymbol{D} = \{\boldsymbol{d}_j\}_{j=1}^{s}, \boldsymbol{d}_j \in \mathbb{R}^n$ represents the dictionary made up of atoms to encode input

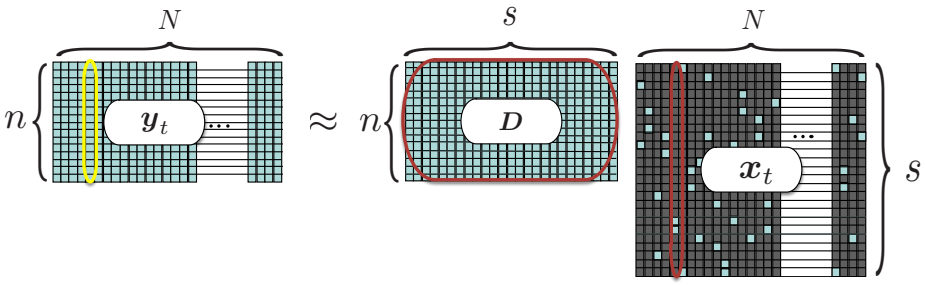


Figure 3.3 The overview of dictionary learning.

features. $\boldsymbol{x}_t \in \mathbb{R}^s$ denotes sparse codes (i.e. sparse decomposition coefficients) with p referring to the type of norm. Meanwhile, N is the total number of training data vectors in memory. The above equation, illustrated in Fig. 3.3, consists of a series of reconstruction error, $\|\boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t\|_2^2$, and a regularization term $\|\boldsymbol{x}_t\|_p$. In Fig. 3.3, every grid represents a numerical value, and dark grids of $\boldsymbol{x}_t$ indicate zero. It can be seen that the motivation of dictionary learning is to sparsely encode input CCAS features over a well-trained dictionary.

Latent Supervised Information

However, from Equation (3.2), it is easy to discover that the main optimization goal is minimizing the reconstruction error in a mean squared sense, which may not be compatible with the goal of classification. Therefore, we try to explore the latent label information, and then propose our joint objective function as Equation (3.3). As aforementioned, a label indicates whether a grid is occupied with an SRAF or not. Here, an assumption has been made in advance that every atom is associated with a particular label. It is true because in initialization, atoms are trained by CCAS features for each class.

$$\min_{\boldsymbol{x},\boldsymbol{D},\boldsymbol{A}} \frac{1}{N}\sum_{t=1}^{N}\{\frac{1}{2}\left\|\left(\boldsymbol{y}_t^\top, \sqrt{\alpha}\boldsymbol{q}_t^\top\right)^\top - \begin{pmatrix}\boldsymbol{D}\\ \sqrt{\alpha}\boldsymbol{A}\end{pmatrix}\boldsymbol{x}_t\right\|_2^2 + \lambda\|\boldsymbol{x}_t\|_p\}. \quad (3.3)$$

In Equation (3.3), α is a hyper-parameter balancing the contribution of each part to reconstruction error. $\boldsymbol{q}_t \in \mathbb{R}^s$ is defined as discriminative sparse code of t-th input feature vector. Thus, $\boldsymbol{A} \in \mathbb{R}^{s \times s}$ transforms original sparse code $\boldsymbol{x}_t$ into discriminative sparse code. Filled with constants ("0.0" or "1.0"), $\boldsymbol{q}_t$ reflects the categorical relationship between corresponding atoms and t-th input. For example, "1.0" means the input shares the same label with the corresponding atom, while "0.0" vice versa. Given the input and the dictionary $\boldsymbol{D}$, it is obvious that $\boldsymbol{q}_t$ is not undetermined. On the contrary, it has some fixed types and once an input is given, one type is selected.

For example, assume $\boldsymbol{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \boldsymbol{d}_4\}$ with $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ from class 1, $\boldsymbol{d}_3$ and $\boldsymbol{d}_4$ from class 2, then $\boldsymbol{q}_t$ for the corresponding input $\boldsymbol{y}_t$ is supposed to be either $(1.0, 1.0, 0.0, 0.0)^\top$ or $(0.0, 0.0, 1.0, 1.0)^\top$. For further explanation, we merge different types of $\boldsymbol{q}_t$ as a $\boldsymbol{Q}$ matrix:

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{q}_1, & \boldsymbol{q}_2 \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix}, \tag{3.4}$$

where $(1.0, 1.0, 0.0, 0.0)^\top$ means input sample shares the same label with $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, and $(0.0, 0.0, 1.0, 1.0)^\top$ indicates that the input, $\boldsymbol{d}_3$ and $\boldsymbol{d}_4$ are from the same class.

To illustrate physical meaning of Equation (3.3) clearly, we can also rewrite it via splitting the reconstruction term into two terms within l_2-norm as Equation (3.5):

$$\min_{\boldsymbol{x},\boldsymbol{D},\boldsymbol{A}} \frac{1}{N} \sum_{t=1}^{N} \{ \frac{1}{2} \left\| \boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t \right\|_2^2 + \frac{\alpha}{2} \left\| \boldsymbol{q}_t - \boldsymbol{A}\boldsymbol{x}_t \right\|_2^2 + \lambda \| \boldsymbol{x}_t \|_p \}. \tag{3.5}$$

The first term $\left\| \boldsymbol{y}_t - \boldsymbol{D}\boldsymbol{x}_t \right\|_2^2$ is still the reconstruction error term. The second term $\left\| \boldsymbol{q}_t - \boldsymbol{A}\boldsymbol{x}_t \right\|_2^2$ represents discriminative error, which imposes a constraint on the approximation of $\boldsymbol{q}_t$. As a result, the input CCAS features from same class share quite similar representations.

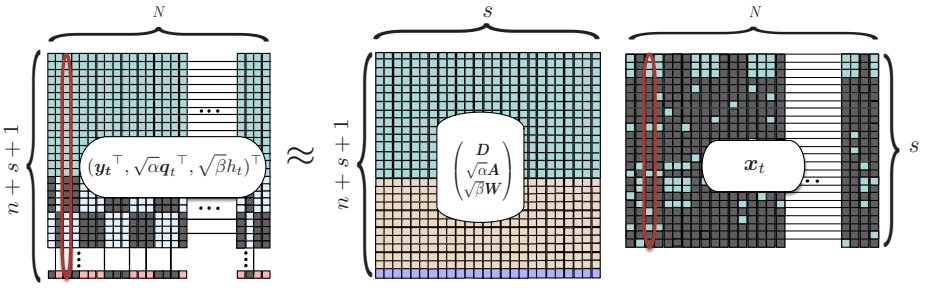


Figure 3.4 The illustration of supervised dictionary learning.

Direct Supervised Information

Since the latent class information has been exploited, the label information can also be directly harnessed. After adding the prediction error term into initial objective function Equation (3.3), we propose our final joint objective function as Equation (3.6):

$$\min_{\boldsymbol{x},\boldsymbol{D},\boldsymbol{A},\boldsymbol{W}} \frac{1}{N} \sum_{t=1}^{N} \{ \frac{1}{2} \left\| \left(\boldsymbol{y}_t^\top, \sqrt{\alpha} \boldsymbol{q}_t^\top, \sqrt{\beta} h_t \right)^\top - \begin{pmatrix} \boldsymbol{D} \\ \sqrt{\alpha} \boldsymbol{A} \\ \sqrt{\beta} \boldsymbol{W} \end{pmatrix} \boldsymbol{x}_t \right\|_2^2 + \lambda \| \boldsymbol{x}_t \|_p \}, \tag{3.6}$$

where $h_t \in \mathbb{R}$ is the label with $\boldsymbol{W} \in \mathbb{R}^{1 \times s}$ the related weight vector, and therefore $\| h_t - \boldsymbol{W} \boldsymbol{x}_t \|_2^2$ refers to the classification error. α and β are hyper-parameters which control the contribution of each term to reconstruction error and balance the trade-off. It can be seen that Equation (3.6) restricts the representation of original data. Furthermore, it is expected to let the original data vectors from the same class share similar representations, thus benefits the calibration of a machine learning model. The illustration for supervised dictionary learning is shown in Fig. 3.4, where the supervised information is appended to the end of the unsupervised data vector and new feature vectors from same class are similar in terms of structure.

The discriminative property in low-dimension of the new feature is demonstrated in Fig. 3.5, where blue color refers to the non-zero number. More than 1000 CCAS features and their corresponding

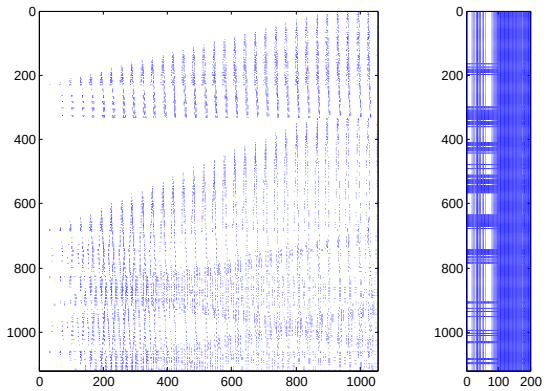


Figure 3.5 Feature map comparison: (a) CCAS based; (b) The same features after supervised feature revision.

new features are selected as exemplars. Fig. 3.5(a) shows the original CCAS feature which is sparse due to benchmark layouts. The dimensionality of the CCAS features is over 1000. On the other hand, as shown in Fig. 3.5(b), the new features are in low-dimension space (dimensionality only 200) meanwhile they can be roughly divided into two classes.

3.3.3 Online Algorithm

Recently, some attempts which explore label information are proposed in succession such as Discriminative K-SVD [169], kernelized supervised dictionary learning [44], label consistent K-SVD [73] (LCK-SVD) dictionary learning and supervised K-SVD with dual-graph constraints [158].

However, most of them are based on K-SVD [6] which belongs to batch-learning method. They are not suitable for dealing with large dataset since the computation overhead (e.g. computing the inverse of a very large matrix) may be high. Online learning method applied in dictionary learning model [98,125] is a good idea, yet these algorithms are unsupervised.

Therefore, we develop an efficient online learning method, which

seamlessly combines aforementioned supervised dictionary learning. Unlike the batch approaches, online approaches process training samples incrementally, one training sample (or a small batch of training samples) at a time, similarly to stochastic gradient descent.

According to our proposed formulation (i.e. Equation (3.6)), the joint optimization of both dictionary and sparse codes is non-convex, but sub-problem with one variable fixed is convex. Hence, Equation (3.6) can be divided into two convex sub-problems. Note that, in a taste of linear algebra, our new input with label information, i.e. $\left(\boldsymbol{y}_t^\top, \sqrt{\alpha}\boldsymbol{q}_t^\top, \sqrt{\beta}h_t\right)^\top$ in Equation (3.6), can be still regarded as the original $\boldsymbol{y}_t$ in Equation (3.2). So is the new merged dictionary consisting of $\boldsymbol{D}$, $\boldsymbol{A}$ and $\boldsymbol{W}$. For simplicity of description and derivation, in following analysis, we will use $\boldsymbol{y}_t$ referring to $\left(\boldsymbol{y}_t^\top, \sqrt{\alpha}\boldsymbol{q}_t^\top, \sqrt{\beta}h_t\right)^\top$ and $\boldsymbol{D}$ standing for merged dictionary with $\boldsymbol{x}$ as the sparse codes.

The two stages, i.e. sparse coding and dictionary constructing, will be alternatively performed in iterations. Thus, in t-th iteration, the algorithm firstly draws the input sample $\boldsymbol{y}_t$ or a mini-batch over the current dictionary $\boldsymbol{D}_{t-1}$ and obtains the corresponding sparse codes $\boldsymbol{x}_t$. Then use two updated auxiliary matrices, $\boldsymbol{B}_t$ and $\boldsymbol{C}_t$ to help compute $\boldsymbol{D}_t$.

The objective function for sparse coding is showed in Equation (3.7):

$$\boldsymbol{x}_t \triangleq \arg\min_{\boldsymbol{x}} \frac{1}{2}\left\|\boldsymbol{y}_t - \boldsymbol{D}_{t-1}\boldsymbol{x}\right\|_2^2 + \lambda\|\boldsymbol{x}\|_1. \tag{3.7}$$

If the regularizer adopts l_0-norm, solving Equation (3.7) is NP-hard. Although there exist some classical algorithms like matching pursuit (MP) [100] and orthogonal matching pursuit (OMP) [114] to address problem with l_0-norm regularizer, nowadays, adopting l_1-norm is much more popular. The reason is that l_1-norm is a convex replacement of l_0-norm, and thus many more fancy optimization algorithms can be exploited. In statistical community, Equation (3.7) is a model called Lasso [133], which is short for the least absolute shrinkage and selection operator. As we can see in Equation (3.7), Lasso is a penalized least squares technique that puts l_1 constraint on the estimated

regression coefficients. It is mainly based on the following two concepts. One is that l_1 regularization approximates l_0 regularization. The other is that shrinkage operation improves prediction performance. Lasso can be solved by coordinate descent [41, 136] and some other first-order algorithms which are critical in large-scale machine learning and deep learning [117]. For example, fast iterative shrinkage-threshold algorithm (FISTA) [13] is a fancy Lasso solver, which achieves the optimal convergence rate of first-order algorithms. Although various variants exist, gradient descent and mirror descent are the foundations of first-order methods [9].

Two auxiliary matrices $\boldsymbol{B}_t \in \mathbb{R}^{(n+s+1)\times s}$ and $\boldsymbol{C}_t \in \mathbb{R}^{s\times s}$ are defined respectively in Equation (3.8) and Equation (3.9):

$$\boldsymbol{B}_t \leftarrow \frac{t-1}{t}\boldsymbol{B}_{t-1} + \frac{1}{t}\boldsymbol{y}_t\boldsymbol{x}_t^\top, \tag{3.8}$$

$$\boldsymbol{C}_t \leftarrow \frac{t-1}{t}\boldsymbol{C}_{t-1} + \frac{1}{t}\boldsymbol{x}_t\boldsymbol{x}_t^\top. \tag{3.9}$$

The objective function for dictionary construction is:

$$\boldsymbol{D}_t \triangleq \arg\min_{\boldsymbol{D}} \frac{1}{t}\sum_{i=1}^{t}\{\frac{1}{2}\left\|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i\right\|_2^2 + \lambda\|\boldsymbol{x}_i\|_1\}. \tag{3.10}$$

Algorithm 1 summarizes the algorithm details of the proposed supervised online dictionary learning (SODL) algorithm. First, in initialization stage, we randomly pick some data of two categories from training dataset as inputs $\boldsymbol{Y}$. To initialize $\boldsymbol{D}$, several iterations of K-SVD [6] for each class are computed, and then all the outputs of each K-SVD are combined. As a result, atoms are naturally allocated to corresponding classes. After initialization of $\boldsymbol{D}$, initial $\boldsymbol{Q}$ is ready. For initialization of $\boldsymbol{A}$, we adopt ridge regression model [56], as showed in Equation (3.11):

$$\arg\min_{\boldsymbol{A}} \left\|\boldsymbol{Q} - \boldsymbol{A}\boldsymbol{X}\right\| + \lambda_2 \left\|\boldsymbol{A}\right\|_2^2. \tag{3.11}$$

Similarly, for initial $\boldsymbol{W}$, the ridge regression model is again exploited. Then, we use coordinate descent algorithm [41, 136] as

Algorithm 1 Supervised Online Dictionary Learning (SODL)

Require: Input merged features $\boldsymbol{Y} \leftarrow \{\boldsymbol{y}_t\}_{t=1}^{N}, \boldsymbol{y}_t \in \mathbb{R}^{(n+s+1)}$ (including original CCAS features, discriminative sparse code $\boldsymbol{Q} \leftarrow \{\boldsymbol{q}_t\}_{t=1}^{N}, \boldsymbol{q}_t \in \mathbb{R}^{s}$ and label information $\boldsymbol{H} \leftarrow \{h_t\}_{t=1}^{N}, h_t \in \mathbb{R}$).

Ensure: New features $\boldsymbol{X} \leftarrow \{\boldsymbol{x}_t\}_{t=1}^{N}, \boldsymbol{x}_t \in \mathbb{R}^{s}$, dictionary $\boldsymbol{D} \leftarrow \{\boldsymbol{d}_j\}_{j=1}^{s}, \boldsymbol{d}_j \in \mathbb{R}^{(n+s+1)}$.

1: **Initialization**: Initial merged dictionary $\boldsymbol{D}_0, \boldsymbol{d}_j \in \mathbb{R}^{(n+s+1)}$ (including initial transformation matrix $\boldsymbol{A}_0 \in \mathbb{R}^{s\times s}$ and initial label weight matrix $\boldsymbol{W}_0 \in \mathbb{R}^{1\times s}$), $\boldsymbol{C}_0 \in \mathbb{R}^{s\times s} \leftarrow \boldsymbol{0}$, $\boldsymbol{B}_0 \in \mathbb{R}^{(n+s+1)\times s} \leftarrow \boldsymbol{0}$;
2: **for** $t \leftarrow 1$ to N **do**
3: Sparse coding $\boldsymbol{y}_t$ and obtaining $\boldsymbol{x}_t$; ▷ Equation (3.7)
4: Update auxiliary variable $\boldsymbol{B}_t$; ▷ Equation (3.8)
5: Update auxiliary variable $\boldsymbol{C}_t$; ▷ Equation (3.9)
6: Update dictionary $\boldsymbol{D}_t$; ▷ Algorithm 2;
7: **end for**

the solving scheme to Equation (3.7) (line 3). To accelerate the convergence speed, Equation (3.10) involves the computations of past signals $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ and the sparse codes $\boldsymbol{x}_1, ..., \boldsymbol{x}_t$. One way to efficiently update dictionary is that introduce some sufficient statistics, i.e. $\boldsymbol{B}_t \in \mathbb{R}^{(n+s+1)\times s}$ (line 4) and $\boldsymbol{C}_t \in \mathbb{R}^{s\times s}$ (line 5), into Equation (3.10) without directly storing the past information, namely input data sample $\boldsymbol{y}_i$ and corresponding sparse codes $\boldsymbol{x}_i$ for $i \leq t$. These two auxiliary variables play important roles in updating atoms, which aggregates the past information during computations. We further exploit block coordinate method with warm start [41] to resolve Equation (3.10) (line 6). As a result, through some gradient calculations, we bridge the gap between Equation (3.10) and sequentially updating atoms based on Equations (3.12) and (3.13).

$$\boldsymbol{u}_j \leftarrow \frac{1}{\boldsymbol{C}\left[j,j\right]}\left(\boldsymbol{b}_j - \boldsymbol{D}\boldsymbol{c}_j\right) + \boldsymbol{d}_j. \tag{3.12}$$

$$\boldsymbol{d}_j \leftarrow \frac{1}{\max\left(\left\|\boldsymbol{u}_j\right\|_2, 1\right)}\boldsymbol{u}_j. \tag{3.13}$$

For each atom $\boldsymbol{d}_j$, the updating rule is illustrated in Algorithm 2. In Equation (3.12), $\boldsymbol{D}_{t-1}$ is selected as the warm start of $\boldsymbol{D}$. $\boldsymbol{b}_j$ in-

Algorithm 2 Rules for Updating Atoms

Require: $\boldsymbol{D}_{t-1} \leftarrow \{\boldsymbol{d}_j\}_{j=1}^{s}, \boldsymbol{d}_j \in \mathbb{R}^{(n+s+1)}$,
$\boldsymbol{B}_t \leftarrow \{\boldsymbol{b}_j\}_{j=1}^{s}, \boldsymbol{b}_j \in \mathbb{R}^{(n+s+1)}$,
$\boldsymbol{C}_t \leftarrow \{\boldsymbol{c}_j\}_{j=1}^{s}, \boldsymbol{c}_j \in \mathbb{R}^{s}$.
Ensure: dictionary $\boldsymbol{D}_t \leftarrow \{\boldsymbol{d}_j\}_{j=1}^{s}, \boldsymbol{d}_j \in \mathbb{R}^{(n+s+1)}$.
1: **for** $j \leftarrow 1$ to s **do**
2: Update the j-th atom $\boldsymbol{d}_j$; ▷ Equations (3.12) and (3.13)
3: **end for**

dicates the j-th column of $\boldsymbol{B}_t$, while $\boldsymbol{c}_j$ is the j-th column of $\boldsymbol{C}_t$. $\boldsymbol{C}[j,j]$ denotes the j-th element on diagonal of $\boldsymbol{C}_t$. Equation (3.13) is an l_2-norm constraint on atoms to prevent atoms becoming arbitrarily large (which may lead to arbitrarily small sparse codes). [15] proves that in the stage of constructing dictionary, the convex optimization problem allowing separable constraints in the updated blocks (columns) will guarantee the convergence to a global optimum.

3.3.4 Convergence Analysis

Proposed algorithm deals the non-convex optimization problem in an alternative framework. As a result, our algorithm converges to a stationary point of the objective function, while finding the global optimum is not guaranteed [59]. In fact, for practical applications, stationary points are enough empirically.

Before our analysis, some definitions should be made in advance. The optimal value of the sparse coding problem is defined as Equation (3.14).

$$l(\boldsymbol{y}_i, \boldsymbol{D}) \triangleq \min_{\boldsymbol{x}} \{\frac{1}{2} \|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1\}. \tag{3.14}$$

As illustrated in Equation (3.15), we denote the empirical cost function (whose value is supposed to be small) to check whether D is good at representing the input data.

$$f_t(\boldsymbol{D}) \triangleq \frac{1}{t} \sum_{i=1}^{t} l(\boldsymbol{y}_i, \boldsymbol{D}). \tag{3.15}$$

Actually, according to [18], the minimization of empirical cost $f_t(\boldsymbol{D})$ with high accuracy is not the real spotlight. By contrast, the minimization of expected cost $f(\boldsymbol{D})$, i.e. Equation (3.16) , is what we want to explore.

$$f(\boldsymbol{D}) \triangleq \mathbb{E}_{\boldsymbol{y}_t}[l(\boldsymbol{y}_t, \boldsymbol{D})] = \lim_{t\to\infty} f_t(\boldsymbol{D}). \tag{3.16}$$

We define the surrogate function, $\hat{f}_t$ in Equation (3.17), for f_t. It can be seen that it upperbounds the empirical cost $f_t(\boldsymbol{D}_t)$.

$$\hat{f}_t(\boldsymbol{D}) \triangleq \frac{1}{t}\sum_{i=1}^{t}\{\frac{1}{2}\|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i\|_2^2 + \lambda\|\boldsymbol{x}_i\|_1\}. \tag{3.17}$$

The $\hat{f}_t(\boldsymbol{D}_t)$ and $f_t(\boldsymbol{D}_t)$ converge almost to the same limit and hence that $\hat{f}_t$ plays a role as the surrogate function for f_t. As a result, $\boldsymbol{D}_t$ is close to $\boldsymbol{D}_{t-1}$ for large value of t, which motivates us to use $\boldsymbol{D}_{t-1}$ as a warm start when computing $\boldsymbol{D}_t$.

Theorem 1. *Assume the surrogate function $\hat{f}_t$ are strictly convex with lower-bounded Hessians, when t goes infinity, the distance between the stationary points of proposed dictionary learning problem and dictionary $\boldsymbol{D}_t$ will converge to* 0.

The detailed proof is provided in Appendix A.1.2.

3.4 SRAF Insertion

3.4.1 SRAF Probability Learning

After feature extraction via CCAS and revision through SODL framework, the discriminative, low-dimension features of grids on a layout are obtained. Therefore, a machine learning model is calibrated by new features of training set, and predicts the probabilities of inserting SRAF patterns into gridded testing layout. This procedure is named as SRAF Probability Learning, which not only guides the SRAF insertion, but also bridges the gap between the previous

continuous optimization problem and the consequent discrete optimization issue. For fair comparison, we exploit the same classifier, logistic regression, as used in [152]. This widely used model harnesses the logistic function as the core to predict the probabilities for classes.

3.4.2 SRAF Insertion via ILP

Through SODL model and classifier, the probabilities of 2-D grids can be obtained. Based on design rules for the machine learning model, the label for a grid with probability less than the threshold is "0". It means that the grid will be ignored when doing SRAF insertion. However, in [152], the scheme to insert SRAFs is a little naive and greedy. Actually, combined with some SRAF design rules such as maximum length and width, minimum spacing, the SRAF insertion can be modeled as an integer linear programming problem. With ILP model to formulate SRAF insertion, we will obtain a global view of SRAF generation.

In the objective of the ILP approach, we only consider valid grids whose probabilities are larger than the threshold. The probability of each grid is denoted as $p(i, j)$, where i and j indicate the index of a grid. For simplicity, we merge the current small grids into new bigger grids, as shown in Fig. 3.6. Then we define $c(x, y)$ as the value of each merged grid, where x, y denote the index of a merged grid. The rule to compute $c(x, y)$ follows Equation (3.18).

$$c(x,y) = \begin{cases} \sum_{(i,j)\in(x,y)} p(i,j), & \text{if } \exists\ p(i,j) \geq \text{threshold}, \\ -1, & \text{if all } p(i,j) < \text{threshold}. \end{cases} \tag{3.18}$$

The motivation behind this approach is twofold. One is to speed up the ILP via reducing the problem size. Because we can predetermine some decision variables whose values are negative. The other is to keep the consistency of predictions in SRAF probability learning.

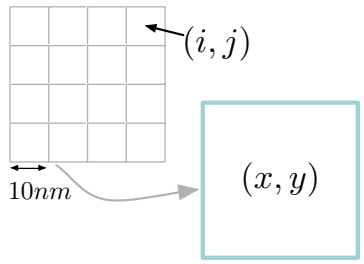


Figure 3.6 SRAF grid model construction.

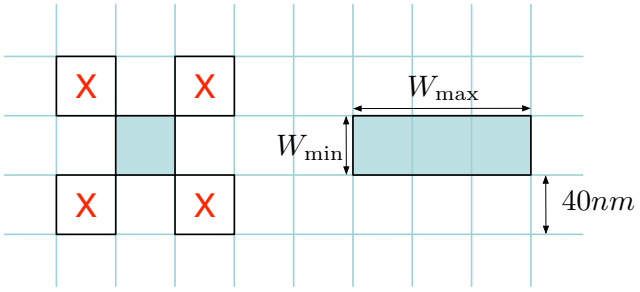


Figure 3.7 Relaxed design rules for SRAF insertion under the grid model.

Although the grid mergence may lead to some quality degradation, it is acceptable compared to the magnificent efficiency increase. The quality degradation is caused by the a few location deviations of SRAF patterns under grid models with different sizes. However, in SRAF insertion, the lithographic performance mainly depends on the number of SRAF patterns and SRAF pattern shapes, while few location deviations affects the lithographic performance minorly. On the other hand, according to the design rules, the minimum size of an SRAF pattern is $40nm \times 40nm$, and the size of a merged grid is also $40nm \times 40nm$. It is more natural to exploit grid mergence, and thus boosts the ILP. Otherwise, it would be time-consuming to find solutions for ILP since constraints are much more complicated.

ILP with Relaxed Design Rules

In ILP for SRAF insertion, our real target is to maximize the total probability of valid grids with feasible SRAF insertion. Accordingly,

it is manifest to put up with the objective function, which is to maximize the total value of merged grids. With considering relaxed design rules (e.g. relaxing maximum length from 90nm to 120nm), the ILP formulation is shown in Formula (3.19).

$$\max_{a(x,y)} \quad \sum_{x,y} c(x,y) \cdot a(x,y) \tag{3.19a}$$

$$\text{s.t.} \quad a(x,y) + a(x-1,y-1) \leq 1, \qquad \forall (x,y), \tag{3.19b}$$

$$a(x,y) + a(x-1,y+1) \leq 1, \qquad \forall (x,y), \tag{3.19c}$$

$$a(x,y) + a(x+1,y-1) \leq 1, \qquad \forall (x,y), \tag{3.19d}$$

$$a(x,y) + a(x+1,y+1) \leq 1, \qquad \forall (x,y), \tag{3.19e}$$

$$a(x,y) + a(x,y+1) + a(x,y+2) + a(x,y+3) \leq 3, \qquad \forall (x,y), \tag{3.19f}$$

$$a(x,y) + a(x+1,y) + a(x+2,y) + a(x+3,y) \leq 3, \qquad \forall (x,y), \tag{3.19g}$$

$$a(x,y) \in \{0,1\}, \qquad \forall (x,y). \tag{3.19h}$$

Here $a(x,y)$ refers to the insertion situation at the merged grid (x,y). According to the rectangular shape of an SRAF and the spacing rule, the situation of two adjacent SRAFs on the diagonal is forbidden by Constraints (3.19b) to (3.19e); e.g. Constraint (3.19b) requires the $a(x,y)$ and the left upper neighbor $a(x-1,y-1)$ cannot be 1 at the same time, otherwise which will lead to the violation against design rules. Constraints (3.19f) to (3.19g) restrict the maximum length of SRAFs. The Fig. 3.7 actively illustrates these linear constraints coming from design rules.

ILP with General Design Rules

Previous case is a relaxed version. However, in most scenarios, more general design rules for inserting SRAFs are considered. Especially, the off-grid pattern is allowed. Now 90nm as maximum length for SRAFs and $40nm$ still the minimum length is taken into account,

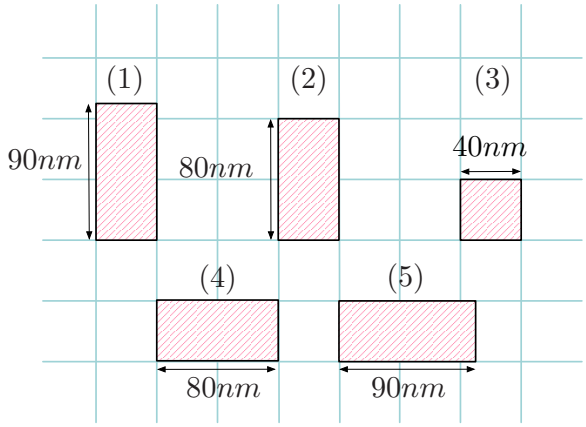


Figure 3.8 Five basic types of SRAF patterns under general design rules.

and thus we choose 5 most representative types for SRAF patterns which are shown in Fig. 3.8.

After considering the flexibility for more general rules, the new ILP model is proposed in Formula (3.20).

$$\max_{a(x,y,i)} \sum_{x,y,i} w_i \cdot a(x,y,i) \cdot v(x,y,i) \tag{3.20a}$$

$$\text{s.t. } v(x,y,1) = c(x,y) + c(x-1,y) + 0.25 \cdot c(x-2,y), \quad \forall (x,y), \tag{3.20b}$$

$$v(x,y,2) = c(x,y) + c(x-1,y), \quad \forall (x,y), \tag{3.20c}$$

$$v(x,y,3) = c(x,y), \quad \forall (x,y), \tag{3.20d}$$

$$v(x,y,4) = c(x,y) + c(x,y+1), \quad \forall (x,y), \tag{3.20e}$$

$$v(x,y,5) = c(x,y) + c(x,y+1) + 0.25 \cdot c(x,y+2), \quad \forall (x,y), \tag{3.20f}$$

$$\sum_{i} a(x,y,i) \leq 1, \quad \forall (x,y), \tag{3.20g}$$

$$a(x_1,y_1,i) + a(x_2,y_2,j) \leq 1, \quad \forall (x_1,y_1,i),(x_2,y_2,j) \in \mathbb{C}, \tag{3.20h}$$

$$a(x,y,i) \in \{0,1\}, \quad \forall (x,y,i). \tag{3.20i}$$

The decision variable $a(x,y,i)$ is extended to three dimensions, which indicates whether the i-th type of SRAF pattern should be

inserted at the merged grid (x, y). $v(x, y, i)$ means the contribution of inserting the i-th type into the merged grid (x, y) to the objective function, while w_i is the corresponding weight parameter. The values for w_1, w_2, w_3, w_4, w_5 are set to be 1.4, 0.1, 1.0, 0.1, 1.4 respectively, which indicates that SRAF patterns of 90nm in length are more encouraged to insert over patterns of 80nm. Since the regular SRAFs around the target pattern benefit lithography, we set the same value for w_1 and w_5, and the same principle of assignment works on w_2 and w_4. The different values for different SRAF patterns are defined in Constraints (3.20b) to (3.20f). Constraint (3.20g) sets the restriction that at most only one SRAF pattern can be inserted in one grid. We introduce a conflict set $\mathbb{C}$ in the formulation (see Constraint (3.20h)), and it contains the conflict pairs of SRAF patterns between one grid and its neighbors. Here, the "conflict" means the distance between two patterns violates the spacing design rule. For instance, $(x, y, 1)$ and $(x-3, y+1, 5)$ form one conflict pair, which is exemplified in Fig. 3.9. To address the problem efficiently, we can assume the main directions for inserting an SRAF pattern are up and right. Hence, under current spacing design rule, we only consider the partial neighborhood of the merged grid (x, y) (i.e. the grid labeled with "1" in Fig. 3.9), which is the area filled with small green dots and the grids labeled with "2" and "3". It can be seen that by introducing the concept of conflict set, the new ILP model is more flexible than previous one and able to to handle new spacing design rule.

3.5 Experimental Results

We implement the framework using `python` on an 8-core 3.7GHz Intel platform. To verify the effectiveness and the efficiency of our SODL algorithm, we employ the same benchmark set as applied in [152], which consists of 8 dense layouts and 10 sparse layouts with contacts sized $70nm$. The spacing for dense and sparse layouts

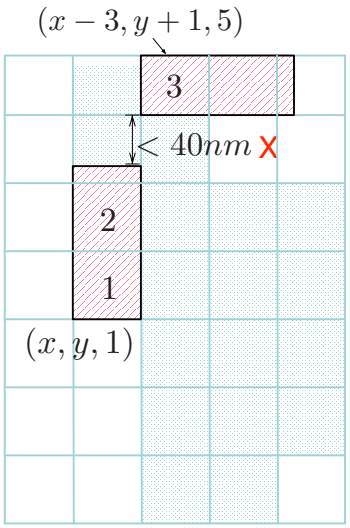


Figure 3.9 The illustration of new ILP model.

are set to $70nm$ and $\geq 70nm$ respectively.

TABLE 3.1 and TABLE 3.2 compare our results with a state-of-the-art machine learning based SRAF insertion tool [152]. Column "Benchmark" lists all the test layouts in both tables. In TABLE 3.1, Column "ISPD'16" refers to the performance of the machine learning framework in [152], while Column "SODL" denotes the results of our model. Within TABLE 3.2, Columns "SODL+Greedy" corresponds to the results of our supervised online dictionary learning framework without ILP model in post-processing, while "SODL+ILP" and "SODL+NewILP" indicate the results from different ILP models with and without relaxed SRAF rules. Columns "F_1 score", "PV band", "EPE" and "CPU" are the evaluation metrics in terms of the learning model performance, the PV band area, the EPE, and the total runtime. Note that in "SODL+Greedy", the same greedy SRAF generation approach as in [152] is utilized.

It can be seen from TABLE 3.1 that the SODL algorithm outperforms [152] in terms of F_1 score by 5.5%. This indicates the predicted SRAFs by our model match the reference results better than [152]. In other words, the proposed SODL based feature revi-

Table 3.1 F_1 Score (%) Comparison with [152].

Benchmark	ISPD'16 [152]	SODL
Dense1	95.37	96.69
Dense2	94.77	97.00
Dense3	93.70	96.87
Dense4	93.89	96.49
Dense5	93.54	96.16
Dense6	93.02	96.86
Dense7	94.22	97.13
Dense8	93.24	96.85
Sparse1	90.51	93.62
Sparse2	87.65	94.03
Sparse3	85.75	91.68
Sparse4	85.56	93.35
Sparse5	85.69	90.48
Sparse6	84.65	91.57
Sparse7	85.00	93.01
Sparse8	84.05	92.72
Sparse9	84.71	90.21
Sparse10	84.03	92.60
Average	89.41	94.30
Ratio	1.000	**1.055**

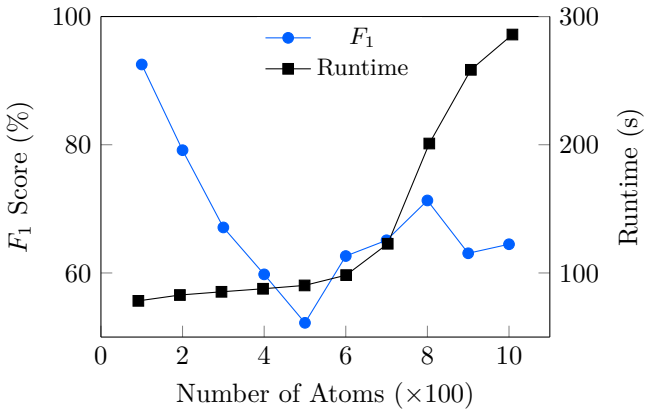


Figure 3.10 The trend of changing number of atoms.

sion can efficiently improve machine learning model generality.

Table 3.2 Lithographic Performance Comparison with [152].

Benchmark	ISPD'16 [152]			SODL+Greedy			SODL+ILP			SODL+NewILP		
	PV band ($.001\mu m^2$)	EPE (nm)	CPU (s)	PV band ($.001\mu m^2$)	EPE (nm)	CPU (s)	PV band ($.001\mu m^2$)	EPE (nm)	CPU (s)	PV band ($.001\mu m^2$)	EPE (nm)	CPU (s)
Dense1	1.891	0.625	1.46	1.840	0.625	1.12	1.823	0.750	1.26	1.850	0.667	2.22
Dense2	1.960	0.313	1.35	2.033	0.500	1.06	2.003	0.438	1.20	1.987	0.438	2.03
Dense3	2.677	1.625	1.25	2.718	1.375	0.97	2.445	1.375	1.07	2.545	1.750	1.88
Dense4	2.426	1.313	1.47	2.288	1.125	1.13	2.459	1.625	1.29	2.363	1.125	2.21
Dense5	2.445	1.250	1.47	2.428	1.375	1.18	2.336	1.375	1.28	2.413	0.938	2.26
Dense6	2.933	0.750	1.17	2.871	1.000	0.91	2.886	0.250	1.00	2.538	0.000	1.78
Dense7	2.426	1.500	1.42	2.409	1.333	1.09	2.318	1.500	1.21	2.277	1.083	2.10
Dense8	2.354	1.417	1.40	2.436	1.417	1.10	2.366	1.167	1.20	2.445	1.000	2.14
Sparse1	2.937	0.438	2.61	2.866	0.563	2.04	2.803	0.375	2.18	2.813	0.500	4.01
Sparse2	2.870	0.625	7.02	2.872	0.516	5.25	2.873	0.594	5.63	2.803	0.625	10.44
Sparse3	2.882	0.556	14.07	2.911	0.535	10.74	2.829	0.528	11.50	2.764	0.563	21.77
Sparse4	2.896	0.566	23.81	2.891	0.496	18.44	2.830	0.547	19.66	2.785	0.547	37.00
Sparse5	2.889	0.565	28.96	2.931	0.571	23.18	2.850	0.580	24.80	2.799	0.633	45.86
Sparse6	2.875	0.558	41.87	2.852	0.630	32.43	2.787	0.572	34.29	2.789	0.552	65.28
Sparse7	2.881	0.540	56.95	2.921	0.611	44.87	2.841	0.575	47.58	2.786	0.536	90.51
Sparse8	2.899	0.564	74.56	2.860	0.573	59.70	2.835	0.560	63.08	2.780	0.610	117.52
Sparse9	2.885	0.586	94.93	2.940	0.549	75.34	2.833	0.568	79.58	2.801	0.573	151.87
Sparse10	2.884	0.599	106.33	2.915	0.512	82.90	2.836	0.560	88.00	2.790	0.555	165.34
Average	2.667	0.799	25.67	2.666	0.795	20.19	2.609	0.774	21.43	2.574	0.705	40.35
Ratio	1.000	1.000	1.000	0.999	0.994	**0.787**	0.978	0.969	0.835	**0.965**	**0.882**	1.572

We exemplify the trends of runtime and F_1 score with respect to the changing of number of atoms, which is depicted in Fig. 3.10. With an increment in number of atoms, runtime ascends. Meanwhile, F_1 score goes down until number of atoms reaches a threshold. According to the theory of dictionary learning, the number of atoms is the dimension of the new feature. Dimension of features impacts the fitting of a machine learning model. Inappropriate dimensionality may cause under-fitting or over-fitting issues. That is the main reason why F_1 scores fluctuate. On the other hand, with the dimension increase of new features, the machine learning model suffers the running time issue. The fitting time of some machine learning models like SVM, increases non-linearly. One reason is the time complexities of algorithms. More importantly, the non-converge problem affects the running time.

We also feed the SRAFed layouts into Calibre [106] to go through a simulation flow that includes OPC and lithography simulation, which will generate printed contours under a given process window. The simulation results summarized in TABLE 3.2 show that we get better PV band and EPE results. In particular, after incorpo-

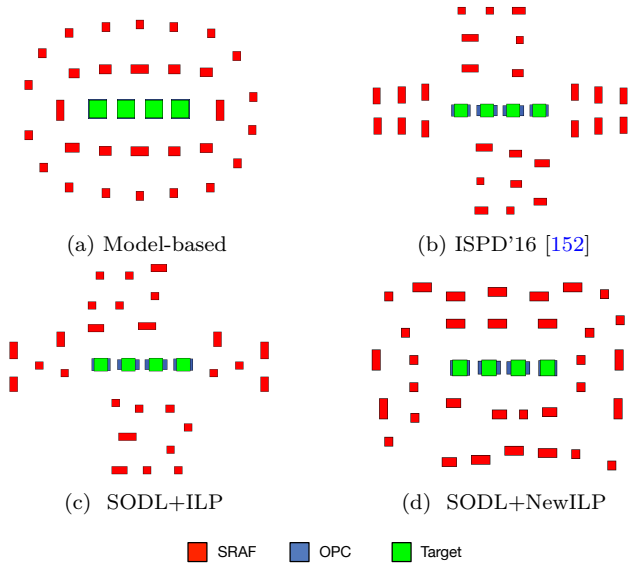


Figure 3.11 Different SRAF insertion methods on a dense layout example.

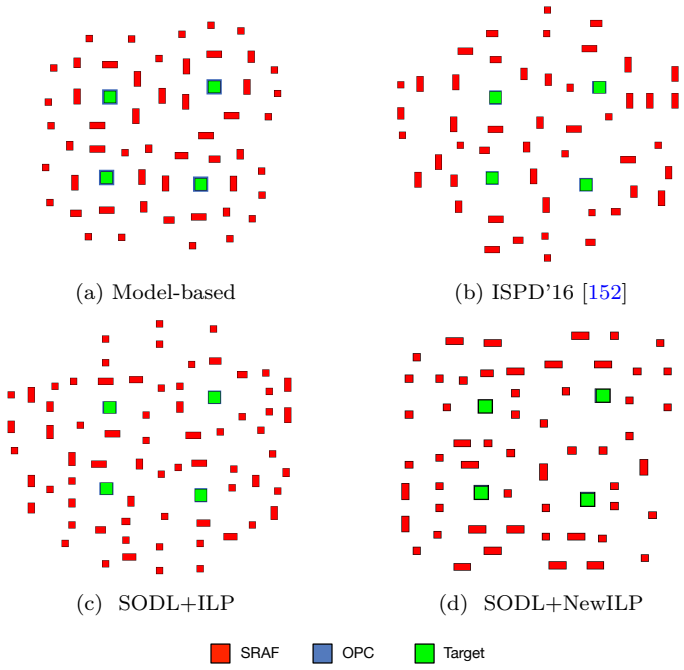


Figure 3.12 Different SRAF insertion methods on a sparse layout example.

rating relaxed SRAF design rules with an ILP solution, proposed algorithm behaves even much better with an average PV band of $2.609\times10^{-3}\mu m^2$ and an average EPE of $0.774nm$ that surpass [152]

with 2% less PV band and 3% less EPE. With considering general SRAF design rules in a new ILP model, our method also performs better in terms of 3.5% less PV band area and 11.8% less EPE within an acceptable runtime. The reason why the running time increases is that general design rules are much more complicated. For example, we need to consider the off-grid patterns. On the other hand, the neighborhood considered in the new ILP model also affects the running time. For one grid, the neighborhood area where SRAF patterns may conflict with others becomes much larger than in the original ILP (see Fig. 3.9). Although the runtime of new ILP model is almost as twice as the original one's, it is worth adopting the model considering the increase of lithographic performance and size of benchmarks.

To further demonstrate the effectiveness and efficiency of proposed ILP models, scalability is explored. In TABLE 3.2, the area of different benchmarks ranges from 1070×1070 nm^2 to 10630×10670 nm^2. We drew the relationship curve between the layout area and the runtime, which is visualized in Fig. 3.13 to clarify the scalability issue further. Note that the benchmarks of same area are removed in the illustration. For example, the benchmarks "`Dense4`" and "`Dense5`" are of the same size, and we only keep "`Dense4`".

The visualizations of simulation results for dense and sparse patterns are exemplified in Fig. 3.11 and Fig. 3.12 respectively. The differences between Figs. 3.11(c) and 3.11(d), Figs. 3.12(c) and 3.12(d) are mainly due to different ILP models following different design rules. Although the predictions from the machine learning model guide the inserting, however, they do not dominate the inserting process. The learning model only filters the invalid small grids. Hence, after solving, the feasible solutions of two ILPs to insert SRAFs can differ a lot.

In addition, the comparisons of total area of inserted SRAFs among different methods are also summarized in TABLE 3.3. According to the results in TABLE 3.2 and TABLE 3.3, we can infer

Table 3.3 The Comparisons of total Area (μm^2) of Inserted SRAFs.

Benchmark	ISPD'16 [152]	SODL+Greedy	SODL+ILP	SODL+NewILP
Dense1	0.072	0.078	0.090	0.122
Dense2	0.099	0.087	0.118	0.149
Dense3	0.123	0.127	0.142	0.120
Dense4	0.138	0.131	0.157	0.157
Dense5	0.107	0.107	0.146	0.148
Dense6	0.146	0.142	0.133	0.140
Dense7	0.138	0.153	0.157	0.136
Dense8	0.116	0.117	0.142	0.145
Sparse1	0.387	0.336	0.408	0.311
Sparse2	1.213	1.299	1.150	0.958
Sparse3	2.350	1.977	2.502	1.943
Sparse4	4.004	4.028	4.027	3.123
Sparse5	4.832	4.909	4.950	3.875
Sparse6	6.922	6.937	7.136	5.510
Sparse7	9.688	7.602	9.658	7.776
Sparse8	12.61	11.73	12.64	9.925
Sparse9	16.18	10.46	16.37	12.54
Sparse10	17.97	14.00	17.68	13.94
Average	4.283	3.568	4.311	3.389
Ratio	1.000	0.833	1.007	0.791

that the total area of inserted SRAF is not the critical factor affects the lithography performance.

3.6 Summary

In this chapter, for the first time, we have introduced the concept of dictionary learning into the layout feature extraction stage and further proposed a supervised online algorithm constructing dictionary. This algorithm has been exploited into a machine learning-based SRAF insertion framework. To get a global view of SRAF generation, combined with design rules, two ILP models have been built to generate SRAFs. The experimental results show that the F_1 score of machine learning model in SRAF insertion has been

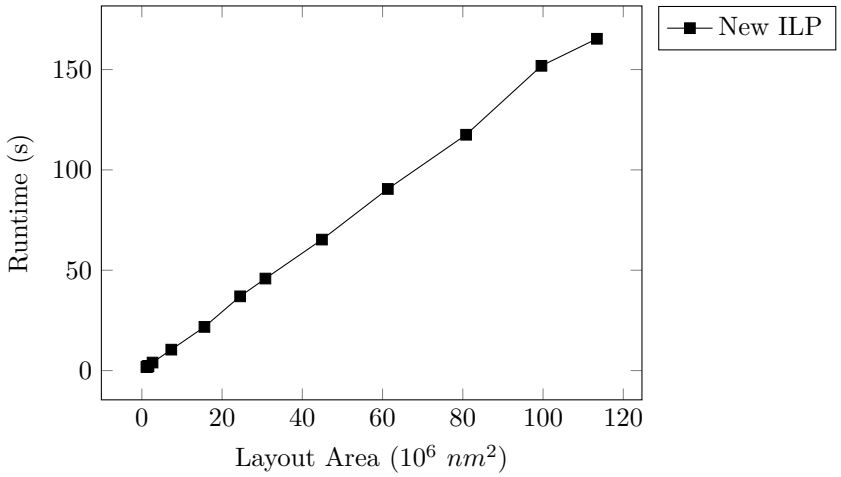


Figure 3.13 The scalability of proposed New ILP model.

boosted and runtime overhead is also acceptable compared with a state-of-the-art SRAF insertion tool. More importantly, the results of lithography simulations demonstrate the promising lithography performance in terms of PV band area and EPE. With the transistor size shrinking rapidly and the layouts becoming more and more complicated, we expect to apply our ideas into general VLSI layout feature learning and encoding.

□ **End of chapter.**

Chapter 4

Layout Hotspot Detection

4.1 Introduction and Motivation

Layout hotspot detection is imperative at the early stage of the mask synthesis. It is usually employed to find and revise problematic designs at an early stage of the whole layout verification flow. As described in Chapter 2, many machine learning and deep learning-based works are proposed and have achieved notable success. Nevertheless, most of them fall into a two-stage flow, where layout feature extraction and detection process are separable. Moreover, existing techniques for layout feature extraction lack both good understandings of similarity among different layout clips and guidances from supervised information. Only a few hotspot detectors [72] ensemble the feature extraction and detection into a one-stage framework. Unfortunately, the features which are automatically learned by convolutional kernels may be ill-separated owing to the lack of discrimination-oriented similarity metrics. Without an effective weighing procedure on the features per their informative significance, those redundant or even misleading features may very well degrade the overall prediction accuracy and performance. In light of these facts, those existing works can only solve simple benchmarks with apparent hotspot patterns like ICCAD 2012 Contest benchmarks, whilst in the case where non-hotspot patterns and

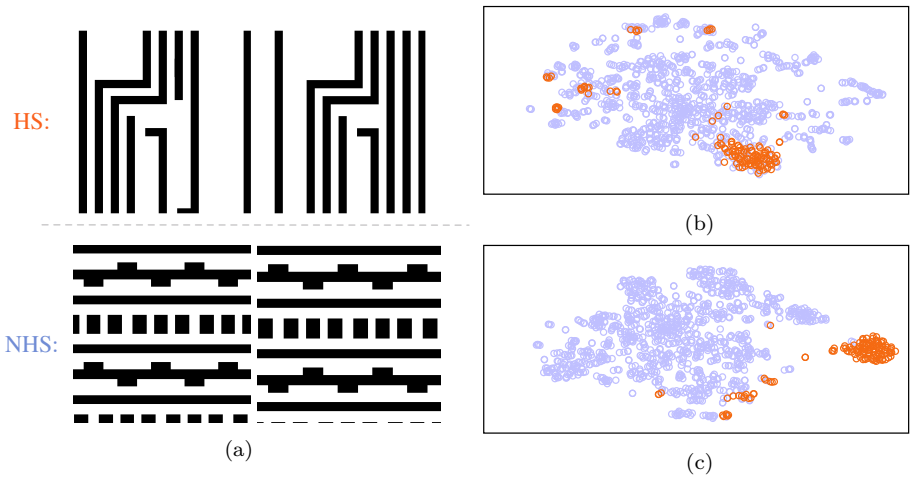


Figure 4.1 The snapshots of layout clips from ICCAD12 benchmarks [135] and Barnes-Hut t-SNE [139] visualizations of feature embeddings on the same benchmarks: (a) The examples of hotspots and non-hotspots; (b) The DCT feature embeddings of TCAD'19 [155]; (c) The feature embeddings of our proposed framework. "HS" is used to represent hotspot clips, while "NHS" refers to non-hotspot clips.

hotspot patterns become increasing similar brings a big challenge to them. To visualize our concern, we sample some hotspots and non-hotspots from ICCAD12 [135] and new via layer benchmarks as exemplars in Figs. 4.1(a) and 4.2(a). Besides, the embeddings learned from TCAD'19 and ours on both benchmark suites are projected into a two-dimension space (shown in Figs. 4.1(b), 4.1(c), 4.2(b) and 4.2(c)). It can be seen that, in easy benchmarks like ICCAD12, the layout clips sharing the same label are usually similar, while the layout clips from different classes have prominent differences. Opposite to ICCAD12 benchmarks, layout clips in the via benchmarks from different classes may look very similar. What's worse, the lack of diversity of via layer patterns introduces an additional challenge to existing hotspot detectors. According to the distributions of layout feature embeddings displayed in Figs. 4.1(b) and 4.2(b), we can infer that although existing works like TCAD'19 work well on ICCAD12 benchmarks, they may have poor performance on more

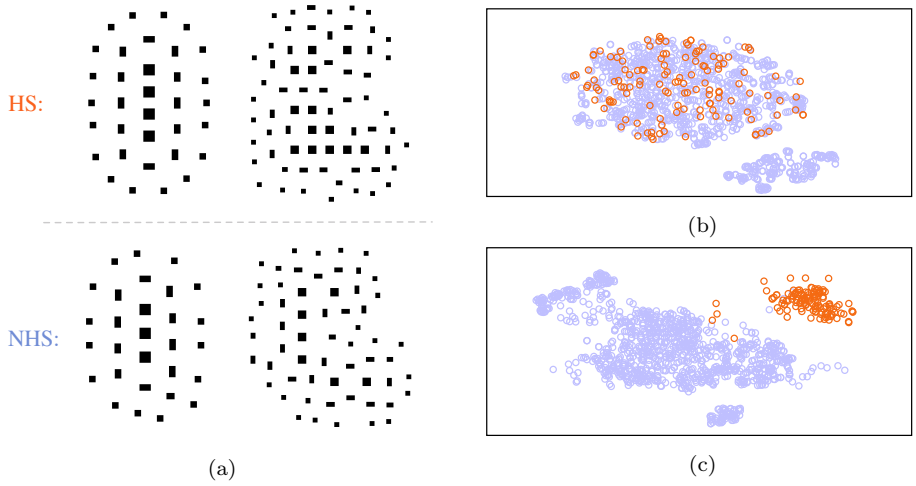


Figure 4.2 The snapshots of layout clips from the via layer benchmarks and Barnes-Hut t-SNE visualizations of feature embeddings on the same benchmarks: (a) The examples of hotspots and non-hotspots; (b) The DCT feature embeddings of TCAD'19; (c) The feature embeddings of our proposed framework. The via layer benchmarks consist of via patterns that contain vias and model-based sub-resolution assist features (SRAFs). Here "via" refers to the via connecting multiple metal layers.

challenging benchmarks. As demonstrated above, layout patterns in such easy benchmarks like ICCAD12 contain too much discriminative information to determine the true states of hotspot detectors.

In this work, we argue that it is of great importance to learn good embeddings of layout clips, which can be assembled into one-stage hotspot detection flow. In accordance with this argument, we propose an end-to-end detection framework where the two tasks, learning embeddings and classification, are jointly performed and mutually benefited. Furthermore, in order to focus on important features and suppressing unnecessary ones, we introduce an attention module and incorporate it into the proposed hotspot detector. To the best of our knowledge, this is the first work for layout embedding learning seamlessly combining with hotspot detection task, and there is no prior work applying attention mechanism based deep metric learning into hotspot detection. To evaluate the true efficacy

of existing works and the proposed detector, we adopt a much more challenging benchmark suite. Figs. 4.1(c) and 4.2(c) prove the performance of the proposed framework on both easy and more challenging benchmarks to some extent. Our main contributions are listed as follows.

- Leverage deep layout metric learning model to learn the good layout feature embeddings.
- Analyze the generalized representation ability of triplet-based deep metric learning algorithm.
- Incorporate feature embedding and hotspot detection into a one-stage multi-branch flow.
- Apply attention mechanism to learn better feature embeddings.
- A more challenging benchmark suite that fills the void created by previous easy benchmarks will be released.
- The proposed detector improves the performance on accuracy, false alarm, and runtime of inference compared to the state-of-the-art frameworks.

The rest of the chapter is organized as follows. Section 5.2.1 introduces some preliminaries on metrics and problem formulation. Section 4.3 first gives a whole view of the proposed framework, then illustrates the backbone network with the applied inception and attention modules, as well as multi-branch design. Section 4.4 describes the loss functions for each branch and some training strategies. Section 6.5 depicts the new via benchmarks and then provides experimental results, followed by the summary in Section 6.6.

4.2 Problem Formulation

In general, our task can be defined as an image classification problem. Our proposed framework treats input layout clips as images.

The label of a layout clip is given according to the information that whether the core region [135] of the clip contains hotspots or not.

We adopt the same metrics exploited in previous work to evaluate the performance of our proposed hotspot detector. The following show definitions of these metrics.

Definition 4 (Accuracy)**.** *The ratio between the number of correctly categorized hotspot clips and the number of real hotspot clips.*

Definition 5 (False Alarm)**.** *The number of non-hotspot clips that are classified as hotspots by the classifier.*

With the evaluation metrics above, our problem is formulated as follows.

Problem 2 (Hotspot Detection)**.** *Given a collection of clips containing hotspot and non-hotspot layout patterns, the objective of deep layout metric learning-based hotspot detection is training a model to learn optimal feature embeddings and classify all the clips so that the detection accuracy is maximized whilst the false alarm is minimized with a less runtime overhead.*

4.3 Hotspot Detection Architecture

4.3.1 Overall Framework

The prior arts are either in a two-stage framework or lacking discriminative feature extractor. By contrast, our proposed algorithm adopts a well-designed multi-branch flow which works in an end-to-end manner. During the training, the proposed multi-branch flow simultaneously works on two branches: one is feature embedding and the other is classification. When the training process finishes, our hotspot detector identifies not only layout feature embeddings, but also a pretty good boundary to divide the hotspots and non-hotspots in embedding space. By the merit of the end-to-end nature, our detector is fast and flexible for both training and inference phases.

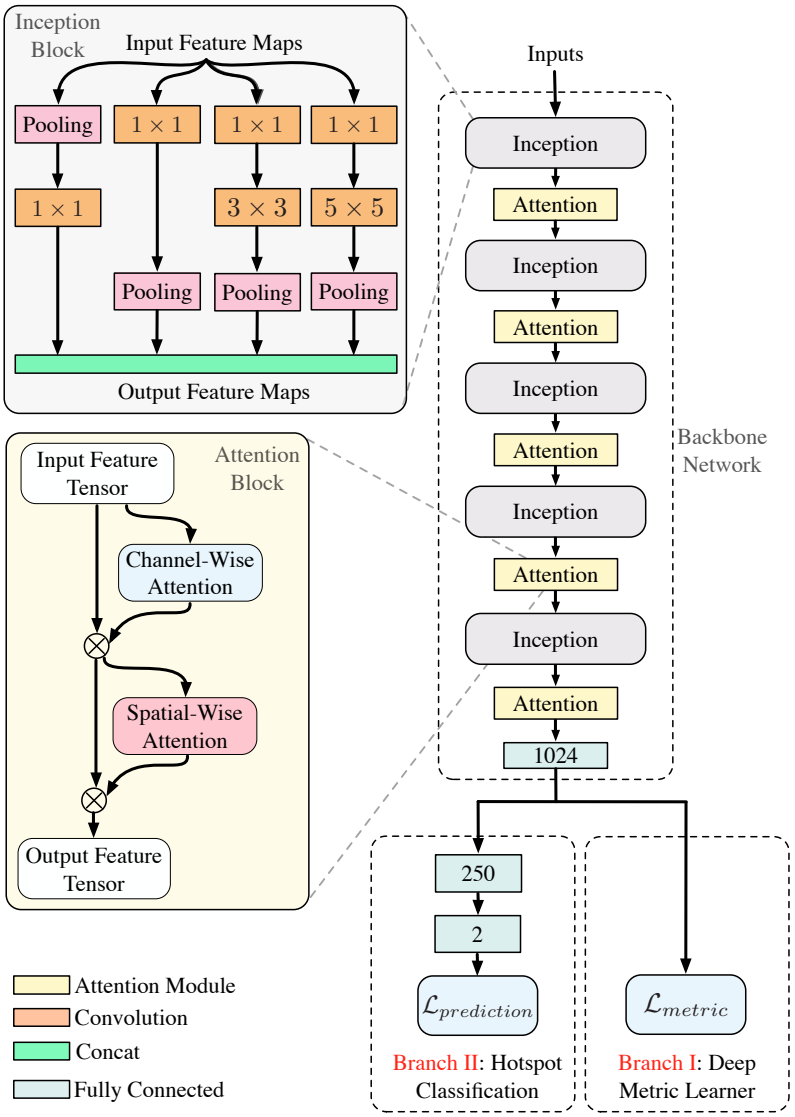


Figure 4.3 The architecture of the proposed hotspot detector.

Fig. 4.3 shows the architecture of the proposed framework, where "⊗" stands for the element-wise multiplication. The proposed framework is composed of three main components: (1) Backbone network which is based on inception structures and attention modules. It is shared by two jointly learned tasks and then is split into two branches. The whole backbone includes 5 inception modules, 5 attention modules and 1 fully connected layer; (2) For deep layout

metric learner branch, it is guided by a triplet loss function to strive for good embeddings of layout clips; (3) For hotspot classification branch, it behaves as an ordinary learning model-based hotspot detector as in other works.

4.3.2 Backbone: Inception Block

Recent progress of deep learning techniques in computer vision reveals that by virtue of the increasing model complexity, a deeper neural network achieves a more robust feature expression and a higher accuracy comparing to a shallow one. However, deeper networks are susceptible to be overfitted, and gradient vanish emerges. What is worse, the turn-around-time at inference stage and training stage are greatly affected, too.

In our context, more cases are needed to concern. For instance, a layout pattern which usually contains several rectangles is monotonous. Another example is in our via pattern, the distances between vias and surrounding SRAFs, the distances among vias have pretty large variations. A single-sized kernel cannot capture multi-scale information. To tackle these issues, we employ an Inception-based structure [129] which consists of several convolutional kernels with multiple sizes. At the same level, these convolutional kernels perform convolution operations with different kernel scales on layout patterns, and the outputs are concatenated in the channel dimension. After going through pooling layers, the fused feature maps are scaled down in height and width dimensions. As a result, the feature maps are comprehensive and rich. The backbone network based on inception structure is illustrated in Fig. 4.3.

4.3.3 Backbone: Attention Block

Recently presented attention mechanism which mimics human perception has achieved much success in the computer vision domain [42, 144, 147]. With attention techniques, neural networks focus on

the salient parts of input features and generate attention-aware feature maps. In order to capture structures of feature maps better, we exploit this mechanism and embed the corresponding modules into our backbone network.

As we know, the features include both cross-channel and spatial information on the back of convolution computations. Based on that fact, the embedded attention modules can emphasize informative parts and suppress unimportant ones along the channel and spatial axes. Channel-wise attention focuses on the informative part itself, while spatial attention concentrates on its spatial location. They are complementary to each other. To fit the motivation, the structure of one attention module consists of two sub-parts: one is channel-wise, and the other is spatial-wise. For a better understanding, we visualize an attention module and zoom in on its intra-structure in Fig. 4.3. The whole attention module sequentially infers a 1D channel attention map, and then a 2D spatial attention map. Through the broadcasting operation, each attention map will perform element-wise multiplication with input feature maps.

The whole process of channel-wise attention is concluded in Equations (4.1) and (4.2), and visualized in Fig. 4.4. First, given an input feature tensor $\boldsymbol{T}$, spatial information of $\boldsymbol{T}$ is firstly aggregated by average-pooling and max-pooling operations, and then two different spatial context descriptors are produced. Next, two descriptors go through a shared encoder-decoder network (i.e. a single hidden layer perceptron named ED() in Equation (4.2)), and output feature vectors are merged using element-wise summation. After activation operation (i.e. defined by $\sigma()$ in Equation (4.2)), the elements of the channel-wise attention masks $A_c(\boldsymbol{T})$ are firstly broadcasted along the spatial dimension, then multiplied with corresponding elements in feature maps. Finally, the module outputs the feature tensor $\boldsymbol{T}'$. In a nutshell, the channel-wise attention module infers the channel-wise attention masks, and each mask will be multiplied element-wisely with input feature maps.

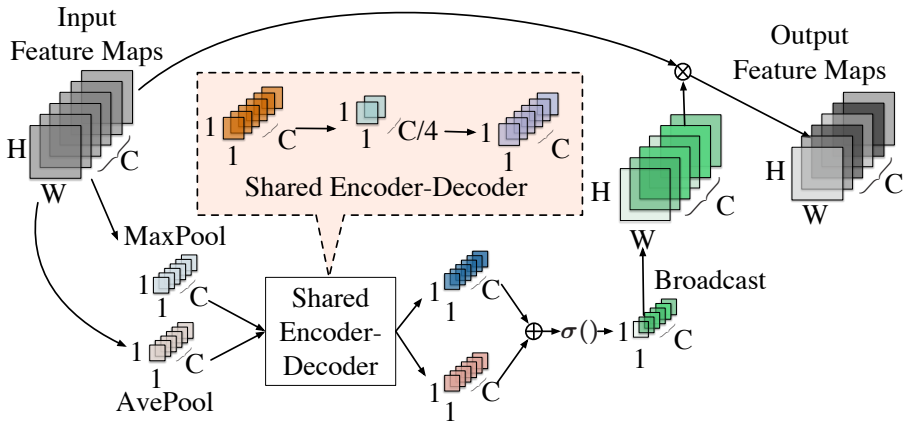


Figure 4.4 The channel-wise attention module.

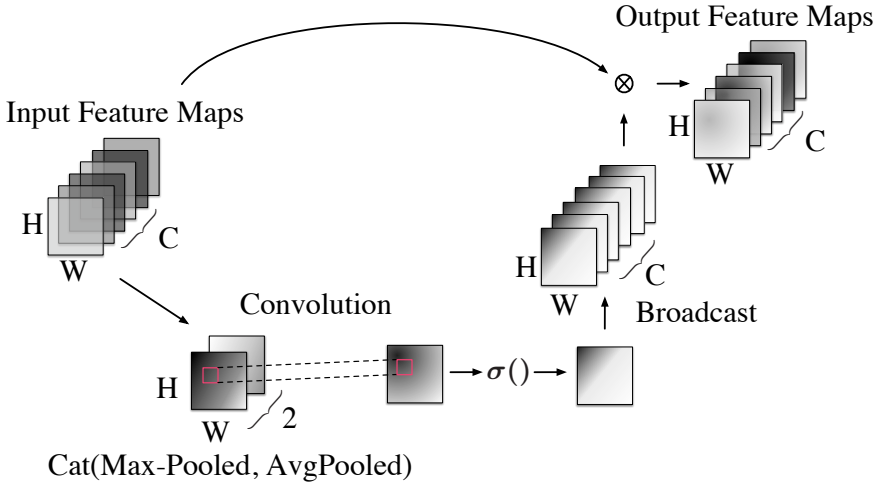


Figure 4.5 The spatial-wise attention module.

$$\boldsymbol{T}' = A_c(\boldsymbol{T}) \otimes \boldsymbol{T}, \tag{4.1}$$

$$A_c(\boldsymbol{T}) = \sigma(\mathrm{ED}(\mathrm{AvgPool}(\boldsymbol{T})) + \mathrm{ED}(\mathrm{MaxPool}(\boldsymbol{T}))). \tag{4.2}$$

We mathematically summarize the computation process for spatial-wise attention module in Equations (4.3) and (4.4), and visualize it in Fig. 4.5. First, through pooling operations, we aggregate channel information of the input feature tensor $\boldsymbol{T}'$, which is also the output of channel-wise attention module. Then, concatenate (denoted by Cat() in Equation (4.4)) two aggregated features and perform

convolution computation (i.e. Conv()) on the concatenation. After activated by sigmoid function, the spatial attention mask $A_s(\boldsymbol{T}')$ is generated. At last, the mask will be multiplied with $\boldsymbol{T}'$ element-wisely, which is shown in Equation (4.3).

$$\boldsymbol{T}'' = A_s(\boldsymbol{T}') \otimes \boldsymbol{T}', \tag{4.3}$$

$$A_s(\boldsymbol{T}') = \sigma(\text{Conv}(\text{Cat}(\text{AvgPool}(\boldsymbol{T}'), \text{MaxPool}(\boldsymbol{T}')))). \tag{4.4}$$

4.3.4 Multi-branch Design

Our end-to-end framework has two jointly-performed tasks: hotspot classification and deep layout metric learning. The two tasks share the backbone network for feature extraction, but are guided by different loss functions. For the deep layout metric learner, the proposed triplet loss is directly calculated on the high dimensional vector which is the output of the fully connected layer in the backbone network. The learner searches for a good feature embedding which non-linearly maps the image representations of layout patterns (i.e. grey images) into a new space. Therefore, pairs of hotspot patterns and non-hotspot patterns can be effectively measured and separated by Euclidean distance metric. For hotspot detection, the main task is to find an appropriate boundary that well divides hotspots and non-hotspots. Via back-propagation, the guide information (i.e. gradients) from two branches update the backbone collectively.

4.4 Loss Functions and Training

4.4.1 Metric Learning Loss in Branch I

Metric learning is typically referred to learning a distance metric or pursuing an embedding function to map images onto a new manifold. Given a similarity metric function, similar images are projected into a neighbourhood, while dissimilar images are mapped apart from each other. Note that the term “similar images” means the images share the same label. Over past decades, the machine learning

community has witnessed a remarkable growth of metric learning algorithms used in a variety of tasks such as classification [35], clustering [150], image retrieval [46] and etc. However, many metric learning approaches explore only linear correlations, thus may suffer from nonlinearities among samples. Although kernel tricks have been developed, it is resource-consuming to find an appropriate one. With a notable success achieved by deep learning, deep metric learning methods have been proposed to find the non-linear embeddings. By taking advantage of deep neural networks to learn a non-linear mapping from the original data space to the embedding space, deep metric learning methods measuring Euclidean distance in the embedding space can reflect the actual semantic distance between data points.

Contrastive loss [68] and triplet loss [34] are two conventional measures which are widely utilized in most existing deep metric learning methods. The contrastive loss is designed to separate samples of different classes with a fixed margin and pull closer samples of the same category as near as possible. The triplet loss is more effective and more complicated, which contains the triplets of anchors, positive and negative samples. Since the triplet loss takes into consideration of higher-order relationships in embedding space and thus can achieve better performance than the contrastive loss. Therefore, in the proposed deep layout metric learning, we adopt the triplet loss.

As aforementioned, the goal of deep metric learning aims at finding a good embedding which is denoted by $f_{\boldsymbol{w}}(\boldsymbol{x}) \in \mathbb{R}^d$ with $\boldsymbol{w}$ as the parameter of f in our assumption. The embedding function $f_{\boldsymbol{w}}(\cdot)$ projects the layout pattern $\boldsymbol{x}$ onto a hypersphere located in a $d-$dimensional compact Euclidean space, where distance directly corresponds to a measure of layout similarity. In other words, the normalization constraint $||f_{\boldsymbol{w}}(\boldsymbol{x})||_2^2 = 1$ is attached to all embeddings. The mechanism behind the triplet loss is based on the following rules:

- During training, the layout clips constitute several triplet instances, where $f_{\boldsymbol{w}}(\boldsymbol{x}_i)$, $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)$, $f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)$ denote an anchor layout clip, a layout clip sharing the same label with the anchor and a layout clip which has the opposite label, respectively;
- Each triplet instance indicates the triplet relationship among three layout clips as:

$$S(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) + M < S(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)), \\ \forall (f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) \in \mathcal{T}. \tag{4.5}$$

  In Equation (4.5), S is the similarity measurement metric and can produce the similarity values always satisfying the triplet relationship. M refers to a margin between positive and negative pairs, and $\mathcal{T}$ collects all valid triplets in training set with $|\mathcal{T}| = n$.
- The Euclidean distance is employed as the metric to measure the similarity between the embedding pairs in new feature space.

To explore the relationship among triplet layout clips, the objective function of deep layout metric learning can be formulated as a loss function $\mathcal{L}_{metric}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$, which is based on the hinge loss (displayed in Equation (4.6a)) with Constraint (4.6b). $\mathcal{L}_{metric}((f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)))$ is also called empirical loss. Minimizing the empirical loss is equivalent to minimizing the violations on the relationship defined in Equation (4.5).

$$\min_{\boldsymbol{w}} \frac{1}{n} \sum_{i=1}^{n} \max(0, M + ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2^2 \\ - ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)||_2^2) \tag{4.6a}$$

$$\text{s.t. } ||f_{\boldsymbol{w}}(\boldsymbol{x}_i)||_2^2 = 1, \ \forall (f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) \in \mathcal{T}. \tag{4.6b}$$

The illustration for the proposed loss is shown in Fig. 4.6. It can be seen that after training, the layout clips of the same category

will be kept apart from the one which is from the other class. The proposed layout triplet loss attempts to enforce a margin between each pair of layout clips from hotspot to non-hotspot. To prove the property, we calculate the gradients of $\mathcal{L}_{metric}$ with respect to the embedding vectors by the following Equations (4.7a) to (4.7c).

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+) - f_{\boldsymbol{w}}(\boldsymbol{x}_i)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{4.7a}$$

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{4.7b}$$

$$\frac{\partial \mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-))}{\partial f_{\boldsymbol{w}}(\boldsymbol{x}_i)} = \frac{2}{n}\left(f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)\right) \cdot \mathbf{1}\left(\mathcal{L}_{metric}(f_{\boldsymbol{w}}(\boldsymbol{x}_i), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+), f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)) > 0\right), \tag{4.7c}$$

where $\mathbf{1}$ is the indicator function which is defined as:

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if x is true,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

When the distance constraint is satisfied, i.e. there are no violations on the triplet relationship, gradients become zeros. As a result, triplet loss $\mathcal{L}_{metric}$ allows the layouts from one category to live on a manifold, while still keep the distance and hence discriminative to another category.

We further analyze the effectiveness of triplet loss by offering its bias between the generalization error and the empirical error. In other words, we evaluate the upper-bound of the network representation ability in a more mathematical way. In the following descriptions, for a better explanation, we use $(\boldsymbol{x}_i, \boldsymbol{x}_j, \boldsymbol{x}_k)$ expressing a valid triplet. Before showing the bias, we readily extend the pair-based definitions in [74] to the triplet scenarios, as in Lemma 1.

Lemma 1. *A triplet-based metric learning algorithm has β-uniform*

stability ($\beta \geq 0$), if

$$\sup_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}} \left|\ell\left(f_{\boldsymbol{w}_\mathcal{T}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right| \leq \beta, \quad \forall \mathcal{T}, \mathcal{T}_i, \tag{4.9}$$

where ℓ indicates a loss function, $\mathcal{T}_i$ is the training set $\mathcal{T}$ with sample $\boldsymbol{x}_i$ replaced by an independent and identically distributed exemplar $\boldsymbol{x}_i'$, and $\mathcal{D}$ is some kind of distribution. $f_{\boldsymbol{w}_\mathcal{T}}(\cdot)$ and $f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)$ are mapping functions learned over the training set $\mathcal{T}$ and $\mathcal{T}_i$, respectively.

Theorem 2. *Assume ℓ be a loss function upper-bounded by $\mathcal{B} \geq 0$, and let $\mathcal{T}$ be a training set consisting of n valid triplets drawn from distribution $\mathcal{D}$, and $f_{\boldsymbol{w}_\mathcal{T}}(\cdot)$ the mapping function parameterized by $\boldsymbol{w}$ which is learned over the training set $\mathcal{T}$ by a β-uniformly stable deep metric learner. The empirical loss over $\mathcal{T}$ is $\mathcal{L}(f_{\boldsymbol{w}_\mathcal{T}}(\cdot))$, while the expected loss of learned mapping function $f_{\boldsymbol{w}_\mathcal{T}}(\cdot)$ over distribution $\mathcal{D}$ is $\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_\mathcal{T}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]$. Then, for $0 < \delta < 1$, with confidence $1-\delta$ approaching to 1, the following inequality exists:*

$$\begin{aligned} &\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_\mathcal{T}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)] - \mathcal{L}(f_{\boldsymbol{w}_\mathcal{T}}(\cdot)) \\ &\leq 3\beta + (2n\beta + 3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \end{aligned} \tag{4.10}$$

The proof of Theorem 2 is detailed in Appendix A.2.1. It can be observed that with the increasing of the number of training triplets, the bias converges. Our analyses for generalized representation ability of triplet-based deep metric learning demonstrate the performance of the proposed layout metric learner.

4.4.2 Classification Loss in Branch II

Except for the same backbone network, there are two fully-connected layers in our classification branch. Different from those hotspot detectors in previous deep learning-based work, the proposed classifier predicts labels based on the features learned by backbone network and deep layout metric learner. Like prior art [155], the loss function

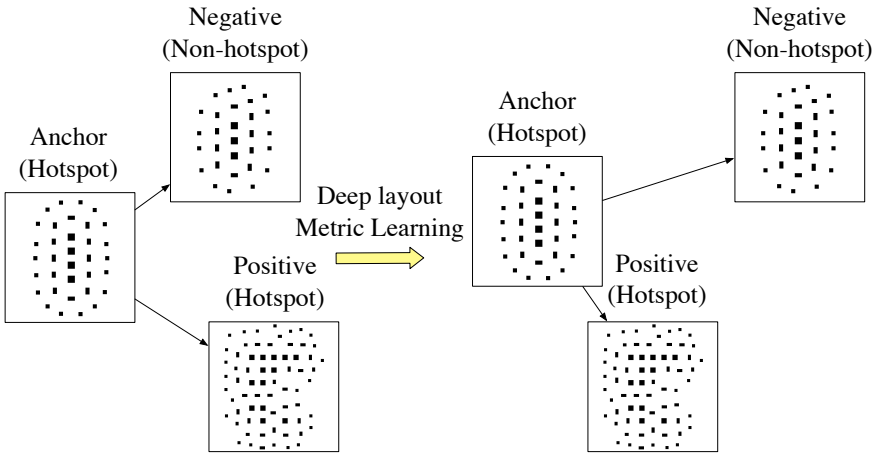


Figure 4.6 The visualization of the proposed deep layout metric learning. In the worst case, the anchor is much similar to the negative than to the positive. In other words, in original space, the distance between the anchor and the negative is shorter than the one between the anchor and the positive. But after deep layout metric learning, in a new manifold, the two hotspot layout clips are kept apart from the non-hotspot clip.

for classification, defined by $\mathcal{L}_{prediction}$, is cross-entropy loss:

$$-\left(y\log(y^*)+(1-y)\log(1-y^*)\right), \tag{4.11}$$

where y^* is the predicted probability of a layout clip, while y is the relevant ground truth (binary indicator).

4.4.3 Training Strategies

Hotspot detection is haunted with a crucial issue that relative datasets (e.g. ICCAD12 benchmark suite [135]), no matter in academia or industry, are quite unbalanced. That is, the number of hotspots is much less than that of non-hotspots. This property results in a biased classification towards the non-hotspots. Additionally, limited by the bottleneck of hardware, it is infeasible to compute the $\arg\min$ or $\arg\max$ across the whole training set. Hence, sampling matters in our framework.

What we do is to generate triplets from a balanced mini-batch in an online fashion. This can be done by constructing a balanced mini-

batch (i.e. the numbers of hotspots and non-hotspots are equal) in one iteration and then selecting the hard negative exemplars from within the batch. Here we divide the negative samples based on a simple rule that easy negatives will lead the loss to become zero, whilst the hard negatives make the loss valid. Since only picking the hardest negatives leads to bad local minima early during training, we keep all anchor-positive pairs in a mini-batch while selectivity sample some hard negatives. Besides hardest negatives, we also consider some negative exemplars which are further away from the anchor than the positive samples but still hard. Those so-called "semi-hard" negatives which lie inside the margin obey the following inequality:

$$\begin{aligned} ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2{}^2 &\leq ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^-)||_2^2 \\ &\leq M + ||f_{\boldsymbol{w}}(\boldsymbol{x}_i) - f_{\boldsymbol{w}}(\boldsymbol{x}_i{}^+)||_2^2. \end{aligned} \tag{4.12}$$

Aiming at progressively selecting false-positive samples that will benefit training, this kind of sampling strategy is widely used in deep metric learning methodologies. Many visual tasks [64, 69, 119, 162, 165] have proven its effectiveness. One reason is that it reduces the number of layout tuples that can be formed for training, and thus enhances the training efficiency.

Nonetheless, the problem that the hotspot clip number is much less than the non-hotspot clip number still exists. We perform top-bottom and left-right flipping on hotspot clips, as flipping a clip does not affect its property during lithography process [155]. This data augmentation will introduce diversity to the dataset and further increase the generalization of our model.

4.5 Experiment Results

The implementation of our framework is in `Python` with the `TensorFlow` library [4], and we test it on a platform with the Xeon Silver 4114 CPU processor and Nvidia TITAN Xp Graphic card. To verify the

Table 4.1 Benchmark Statistics

Benchmarks	Training Set		Testing Set		Size/Clip
	HS#	NHS#	HS#	NHS#	(μm^2)
ICCAD12	1204	17096	2524	13503	3.6×3.6
Via-1	3418	10302	2267	6878	2.0×2.0
Via-2	1029	11319	724	7489	2.0×2.0
Via-3	614	19034	432	12614	2.0×2.0
Via-4	39	23010	26	15313	2.0×2.0
Via-Merge	5100	63665	3449	42294	2.0×2.0

effectiveness and efficiency of our detector, two benchmarks are employed. One is ICCAD12 benchmarks [135], and the other is a more challenging via layer benchmark suite which is under $45nm$ technology node. For fair comparisons against previous works, following these arts, all $28nm$ designs in ICCAD12 benchmarks [135] are merged into a unified case named `ICCAD12`. The details for `ICCAD12` and the via benchmarks are listed in TABLE 4.1.

Columns "Train HS#" and "Train NHS#" list the total number of hotspots and the total number of non-hotspots in the training set, whilst columns "Test HS#" and "Test NHS#" refer to the total number of hotspots and the total number of non-hotspots in the testing set. "Size/Clip (μm^2)" shows the resolution for each benchmark. It is manifest that the via benchmark suite has four individual cases, which are arranged in order of the density of target designs. For example, `Via-1` has the highest density of target designs in its layout clips among other cases. As a result, it contains the most hotspot patterns. In a low-density case, like `Via-4`, the number of hotspot clips are reduced accordingly. We also merge all four small cases into a big one named "`Via-Merge`". Note that, as listed in TABLE 4.1, images in the testing set of `ICCAD12` have a resolution of 3600×3600 which is larger than the images in the via benchmarks of 2048×2048, and in all benchmarks, the pixel size is 1 nm.

Mask images in our employed benchmarks are from distinct stages of optical proximity correction (OPC), and have different targets. For the ICCAD12 benchmark, layout patterns are before OPC, while regarding the rest via layer benchmarks, they are from intermediate OPC results (i.e. inserting SRAFs without OPC). We expect to mimic two scenarios. One is to target at finding and revising problematic designs at an early stage of the whole layout verification flow which is associated with ICCAD12 benchmark. This benchmark is labeled according to the results of the entire layout verification flow containing OPC and lithography simulation. The other is to demonstrate the potential of incorporating the hotspot detectors into OPC engines and facilitate the procedure, which corresponds to the via layer benchmarks (each layout has model-based SRAFs and un-OPCed vias). The labelling on hotspot or non-hotspot patterns in via layer benchmarks is based on the lithography simulation results of current OPC step.

TABLE 4.2 summarizes the comparing results between the proposed framework and several state-of-the-art hotspot detectors. Column "Bench" lists 6 benchmarks used in our experiments. Columns "Accu", "FA", "Time" are hotspot detection accuracy, false alarm count and detection runtime, respectively. Columns "TCAD'19 [155]", "DAC'19 [72]", "ASPDAC'19 [33]" and "JM3'19 [25]" denote the results of selected baseline frameworks respectively. We can see that our framework outperforms TCAD'19 averagely with 15.22% improvement on detection accuracy and 2% less false alarm penalty. Especially, our framework behaves much better on average with 89.89% detection accuracy compared to 77.78%, 83.06%, 58.93% and 62.9% for TCAD'19, DAC'19, ASPDAC'19, JM3'19, respectively. Moreover, the advantage of the proposed one-stage multi-branch flow can also be noticed that it achieves over 3× speedup compared to previous two-stage flows. Note that DAC'19 exhibits a slightly better accuracy on ICCAD12 case, it suffers from high false alarm penalties over almost all cases due to the nature of the bina-

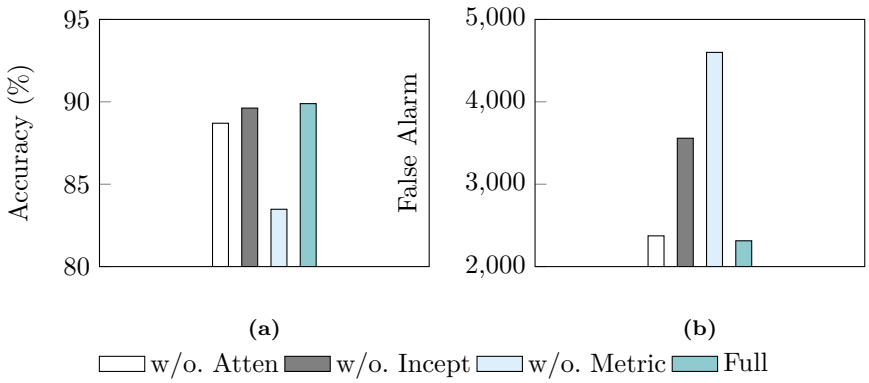


Figure 4.7 Comparison among different configurations on (a) average accuracy and (b) average false alarm.

rized neural network.

An ablation study is also performed on the proposed flow to investigate how different configurations affect the performance. Fig. 4.7 illustrates the contribution of attention module, inception block and layout metric learning loss to our flow. "w/o. Atten" refers to the detector without attention modules, "w/o. Incept" stands for the detector with inception blocks replaced by vanilla convolutional layers, "w/o. Metric" denotes the detector trained without layout metric learning loss, whilst "Full" is our detector with entire techniques. The histogram shows that with attention modules, 1.29% accuracy improvement without additional false alarms on average is achieved, which confirms that the attention modules help the backbone network extract feature more efficiently. With inception blocks, we get a notable reduction on false alarm penalties, i.e. 1245 less on average, which means under the same experiment settings, the inception blocks capture richer information on layout patterns than vanilla convolutional layers. Comparing the whole framework with the model trained without layout metric learning, the model trained with layout metric learning loss reduces 49.72% of the false alarm and get 6.41% further improvement on accuracy.

4.6 Summary

In this chapter, for the first time, we have proposed a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are simultaneously performed. The deep layout metric learning mechanism offers a new solution to extract features from via layer patterns that contain much less discriminative information than metal layer patterns. We further exploit attention techniques to make backbone network self-adaptively emphasize on more informative parts. Additionally, to test the true performance of hotspot detectors, a new via layer benchmark suite has been used for comprehensive verification. The experimental results demonstrate the superiority of our framework over current deep learning-based detectors. The corresponding theoretical analyses are also provided as the pillars. With the transistor size shrinking rapidly and the layouts becoming more and more complicated, we hope to apply our ideas into more general VLSI layout feature learning and encoding.

□ **End of chapter.**

Table 4.2 Comparison with state of the arts

Bench	TCAD'19 [155]			DAC'19 [72]			ASPDAC'19 [33]			JM3'19 [25]			Ours		
	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)	Accu (%)	FA	Time (s)
ICCAD12	98.40	3535	502.70	98.54	3260	561.28	97.66	2825	441.96	97.82	2651	505.67	98.42	2481	143.79
Via-1	71.50	773	43.36	89.85	1886	57.76	64.18	1077	52.95	89.19	2624	47.87	93.42	1589	19.83
Via-2	65.06	1290	40.02	73.00	1222	21.66	30.52	372	43.21	38.81	454	43.06	86.32	1100	13.22
Via-3	48.15	760	60.23	73.38	3406	43.15	26.92	148	77.09	21.06	42	67.13	88.20	2105	20.69
Via-4	76.92	155	67.44	73.08	15288	51.98	61.54	74	87.24	46.15	21	77.15	80.77	152	20.70
Via-Merge	88.01	7633	165.85	90.42	9295	105.30	72.77	3859	228.57	84.34	6759	170.91	92.20	6453	59.74
Average	74.67	2357.67	146.60	83.06	5726.17	140.19	58.93	1392.50	155.17	62.90	2091.83	151.97	89.89	2313.33	46.33
Ratio	0.83	1.02	3.16	0.92	2.48	3.03	0.66	**0.60**	3.35	0.70	0.90	3.28	**1.00**	1.00	**1.00**

Chapter 5

Adder Design Space Exploration

5.1 Introduction and Motivation

Recently, to handle the increasingly complicated design complexity, obtaining high-quality IC design becomes a design space exploration problem. Machine learning techniques are started to be applied into the huge IC design space searching. High-speed adder design space exploration is not an exception. However, there exist two main issues in ML-based adder DSE works like [94]. One is that [94] exploits the two-stage framework where the feature extractor for prefix adders and subsequent machine learning models are separated. Namely, the feature extraction process lacks guided information from the learning models. The other is feature representations are still manually crafted based on domain knowledge, which would be likely to lose useful latent information. More specifically, the manually-crafted attributes like sum-path-fan-out, maximum-fan-out are calculated to characterize the prefix adder structures. However, these attributes only partially represent the prefix adder structure. Therefore, even with a lot of hand-engineered efforts, information loss is still inevitable. We argue that learning representations of prefix adder networks plays a critical role in our framework

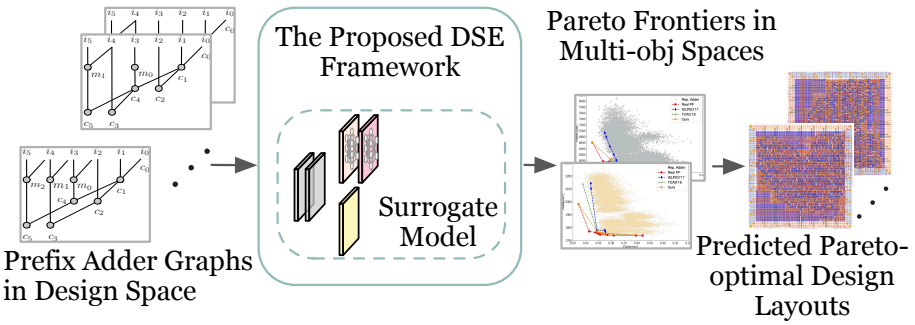


Figure 5.1 The objective spaces include area vs. delay space, power vs. delay space and area vs. power vs. delay space.

because they enable many downstream learning tasks, and a unified feature learning and model training paradigm could boost the DSE process.

In this chapter, we propose an end-to-end deep learning model, graph neural process (GNP), which outputs predictions and uncertainties based on the feature representations automatically learned from adder structures. With GNP as the surrogate model, in this paper, we harness a sequential optimization algorithm [14,175] to perform the design space exploration in physical solution space. The visualization of adder design space exploration is displayed in Fig. 5.1. Our main contributions are summarized as follows:

- A variational graph autoencoder is built to extract features from prefix adder structures automatically.
- A neural process is exploited as an alternative to the Gaussian process to reduce computational complexity.
- A multi-branch, end-to-end surrogate model (i.e. GNP) which incorporates a variational graph autoencoder and a neural process is proposed.
- A GNP-based sequential optimization algorithm to explore Pareto-optimal solutions is investigated.

- The proposed optimization framework with the developed surrogate model uses less labeled data and achieves better Pareto frontiers.

The rest of the chapter is organized as follows. Section 6.2 introduces some prior knowledge about prefix adder synthesis, multi-objective optimization, and then gives the problem formulation. Section 5.3 proposes our graph neural process, while Section 5.4 discusses the sequential optimization framework for design space exploration. Section 6.5 presents the experimental results, followed by the summary in Section 6.6.

5.2 Preliminaries

In this section, the backgrounds of the prefix adder synthesis and multi-objective optimization are offered, and then we give the problem formulation.

5.2.1 Prefix Adder Network

An l bit binary adder eats two l bit inputs $\boldsymbol{A} = \{a_{l-1}..a_1, a_0\}$ and $\boldsymbol{B} = \{b_{l-1}..b_1, b_0\}$, and outputs the sum $\boldsymbol{S} = \{s_{l-1}..s_1, s_0\}$ with a carry out $C_{out} = c_{l-1}$, where $s_i = a_i \oplus b_i \oplus c_{i-1}$ and $c_i = a_i b_i + a_i c_{i-1} + b_i c_{i-1}$. According to the recent study in [166], given the bitwise generation function and propagate function, l-bit binary addition can be represented as a prefix computation process. We visualize the computation process in Fig. 5.2. It can be seen that the process consists of three main steps. The first one is pre-processing where inputs for prefix processing (i.e. $\boldsymbol{g}$ and $\boldsymbol{p}$) are generated bit-wisely. The second is prefix processing which is the main carry-propagation step, and the last is post-processing or carry-out generation.

The prefix processing or carry propagation network can be mapped to a prefix graph problem by taking the outputs of the pre-processing part as inputs and generates c_{out}. Each grey node in Fig. 5.2 is called

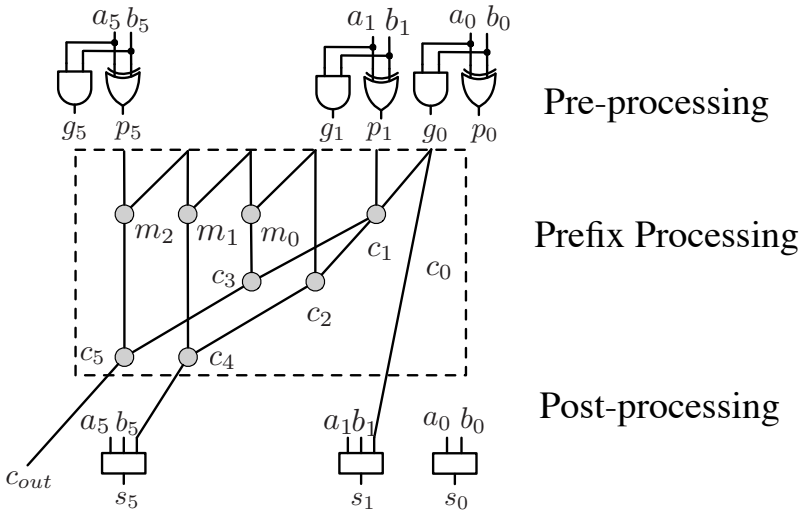


Figure 5.2 6-bit prefix computation process.

a prefix node which represents a certain logic operation. The logic operation of each prefix node in any prefix adder graph is the same except for the different inputs. In the prefix processing part showed in Fig. 5.2, we first extend g and p to multiple bits and define $G_{[i:j]}$, $P_{[i:j]}$ $(i \geq j)$ as

$$P_{[i:j]} = \begin{cases} p_i, & \text{if } i = j \\ P_{[i:k]} \cdot P_{[k-1:j]}, & \text{otherwise} \end{cases} \tag{5.1}$$

$$G_{[i:j]} = \begin{cases} g_i, & \text{if } i = j \\ G_{[i:k]} + P_{[i:k]} \cdot G_{[k-1:j]}, & \text{otherwise} \end{cases} \tag{5.2}$$

The associative logic operation $\circ$ (i.e. any grey node in Fig. 5.2) is defined for $(G,\ P)$ as:

$$\begin{aligned} (G, P)_{[i:j]} &= (G, P)_{[i:k]} \circ (G, P)_{[k-1:j]} \\ &= (G_{[i:k]} + P_{[i:k]} \cdot G_{[k-1:j]}, P_{[i:k]} \cdot P_{[k-1:j]}). \end{aligned} \tag{5.3}$$

Specially, we use c_i to indicate the logic operation whose output involves in the s_i computation.

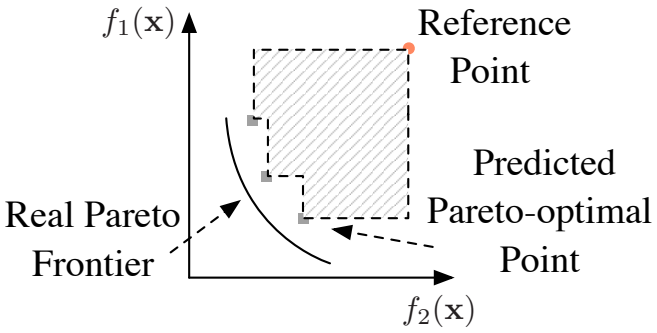


Figure 5.3 An example of hypervolume in a bi-objective space.

5.2.2 Multi-objective Optimization

Assume a multi-objective optimization problem has a set of feasible solutions $\mathcal{X}$. There are n objective functions, $f_1(\cdot), f_2(\cdot), \ldots, f_n(\cdot)$, which map an input $\boldsymbol{x}$ to corresponding results $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})$ and form an n-d result vector $\boldsymbol{f}(\boldsymbol{x})$. A result vector $\boldsymbol{f}(\boldsymbol{u})$ is said to dominate another result vector $\boldsymbol{f}(\boldsymbol{v})$ if $\boldsymbol{f}(\boldsymbol{u})$ is at least as good as $\boldsymbol{f}(\boldsymbol{v})$ in all the objectives, namely, $f_i(\boldsymbol{u}) \geq f_i(\boldsymbol{v}), \forall i \in [1, n]$ if the objectives are to be maximized, and $f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}), \forall i \in [1, n]$ if the objectives are to be minimized. Hence, we say that a solution $\boldsymbol{x}$ is Pareto-optimal if it is not dominated by other solutions in the feasible solution set $\mathcal{X}$. Some prior arts [14, 175] in the machine learning field are proposed to handle the practical multi-objective problems where the evaluation of the multi-objective functions is expensive

In our context for high-speed adder design, a feasible solution $\boldsymbol{x}$ is the feature representation of an adder design implementation which satisfies the pre-determined constraints, while a Pareto-optimal design is where none of the QoR metrics (or objectives) like area, power and delay, can be minimized without worsening at least one of the others. The Pareto set is the set consisting of all the Pareto-optimal solutions, and the Pareto frontier contains objective space values (i.e. the values of QoR metrics) associated with the Pareto set.

5.2.3 Problem Formulation

Definition 6 (Hypervolume [174]). *The metric hypervolume is a Pareto-compliant evaluation, which refers to the volume fenced by the Pareto frontier and a reference point in the objective space. It measures how well distributed the points are on the Pareto frontier approximation.*

In Fig. 6.2, the area filled with dash lines is an example of the hypervolume of a predicted Pareto-optimal set in a bi-objective space. The hypervolume error for a predicted Pareto-optimal set $\hat{\mathcal{P}}$ is defined:

$$\eta = \frac{V(\mathcal{P}) - V(\hat{\mathcal{P}})}{V(\mathcal{P})}, \tag{5.4}$$

where $\mathcal{P}$ is the golden Pareto-optimal set, and $V(\mathcal{P})$ is the ground-truth of hypervolume. If a solution set $\mathcal{P}'$ is better than another set $\mathcal{P}''$, $V(\mathcal{P}')$ is greater than $V(\mathcal{P}'')$. It can be observed that a prediction $\hat{P}$ which contains the whole design space has an error of 0.

Definition 7 (Average Distance from Reference Set (ADRS) [10]). *Given a reference Pareto-optimal set* $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots | \boldsymbol{a} = ({m_1}^a, {m_2}^a, \ldots, {m_n}^a)\}$ *and an approximated Pareto-optimal set* $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots | \boldsymbol{p} = ({m_1}^p, {m_2}^p, \ldots, {m_n}^p)\}$ *in n-objective DSE problem:*

$$ADRS(\mathcal{A}, \mathcal{P}) = \frac{1}{(|\mathcal{A}|)} \sum_{\boldsymbol{a} \in \mathcal{A}} \min_{\boldsymbol{p} \in \mathcal{P}} \delta(\boldsymbol{a}, \boldsymbol{p}), \tag{5.5}$$

where $\delta(\boldsymbol{a}, \boldsymbol{p}) = \max\left\{\left|\frac{{m_1}^p - {m_1}^a}{{m_1}^a}\right|, \ldots, \left|\frac{{m_n}^p - {m_n}^a}{{m_n}^a}\right|\right\}$.

ADRS is used to quantify how close a set of non-dominated points is from the Pareto frontier in the objective space. The smaller ADRS value is, the closer the approximate set P is to the reference set A.

With the aforementioned knowledge, our problem can be formulated.

Problem 3 (Adder Design Space Exploration)**.** *Given a collection of prefix adder networks, the aim of adder design space exploration is to search for the Pareto-optimal adder designs with trade-off among multiple objectives like power, area, and delay over a wide design space.*

5.3 Adder Feature Extraction and Regression

In traditional machine learning techniques, most of the features are manually crafted and based on domain knowledge [57,83]. Followed-up on the idea, previous machine learning-based adder DSE works like [94, 116] exploit hand-engineered features, and the feature extractor and the consequent learning model are separated into two independent components. Isolating the two parts makes the whole framework be susceptible to converge to sub-optimal performance.

Fortunately, deep Learning algorithms attempt to use a general-purpose learning procedure to automatically learn high-level features from data, which eliminates the need for domain expertise and formidable feature extraction [57, 83]. Many of deep learning models (e.g. AlexNet [81], GoogleNet [128], ResNet [65]) perform feature extraction and classification in an end-to-end fashion or in a unified network structure. Recently, the end-to-end learning structure has also achieved notable success in oceans of EDA applications [28,31,32,50,52,53]. Based on the observations and arguments, we design an end-to-end, deep learning-based model, GNP, which incorporates the customized automatic feature extractor for prefix adder networks and the regressor.

In this section, the proposed GNP which adopts a well-designed multi-branch flow is introduced. The proposed multi-branch flow, shown in Fig. 5.4, has a backbone (i.e. the encoder part of a graph auto-encoder) and simultaneously works on two branches: one is the decoder part of graph auto-encoder and the other is the neural process working as an alternative to the traditional Gaussian

process. For the backbone and stream I, a graph autoencoder that aims at summarizing the graph structure and attaining the latent representation of an input prefix adder is adopted. For stream II, a neural process built upon the encoder-decoder structure outputs the regression values with corresponding uncertainties.

5.3.1 Graph Construction of a Prefix Adder Network

Because the prefix adder structure is graph-like, the adjacency matrix and node feature matrix can describe it without any information loss. In the following descriptions, an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n vertices and m edges refers to a prefix adder network. $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is the adjusted adjacency matrix of $\mathcal{G}$, whilst $\boldsymbol{X} \in \mathbb{R}^{n \times s}$ indicates the attribute matrix of nodes. Note that it is customary to set diagonal elements in an adjacency matrix be zeros, however, it is added by an identity matrix $\boldsymbol{I}$ in the operation of a graph convolutional network (GCN). Accordingly, we term $\boldsymbol{A}$ as the adjusted adjacency matrix.

As aforementioned in Section 6.2, the prefix adder network can be mapped to a prefix graph structure. For a better understanding, we list an example of the mapping in Fig. 5.4. According to our design, we regard inputs, outputs and prefix nodes in a prefix graph as vanilla vertices in graph $\mathcal{G}$, while the logic relationship between two nodes is equivalent to an edge. Until now, the basic structure of $\mathcal{G}$ is constructed, and then the adjusted adjacency matrix $\boldsymbol{A}$ is built. Each row in $\boldsymbol{X}$ represents corresponding node attributes, which is composed of the logic level, in-degree, the out-degree and EDA tool settings. For each vertex, the in-degree means the number of vertices taking part in one logic operation, while the out-degree counts the number of vertices in higher logic levels connecting to the current vertex. Apart from the prefix graph structural features, tool settings from logical synthesis stage and physical design stage as other features are also considered as part of node features. We syn-

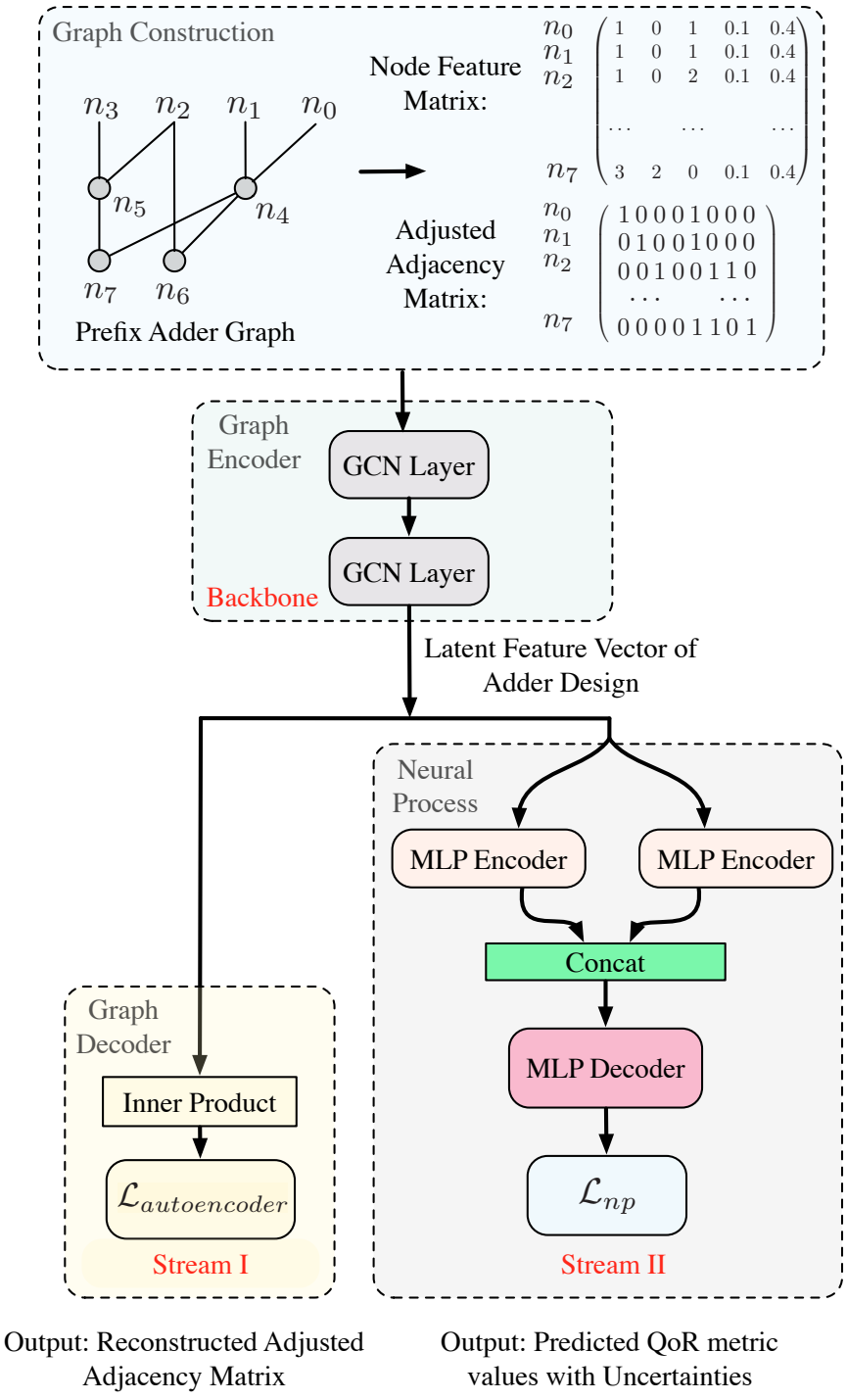


Figure 5.4 The diagram of the proposed graph neural process.

thesize the adder structures by industry-standard EDA synthesis tool [1], where we can configure the synthesis parameters for the adder. Different values of the synthesis parameters can result in different QoR metric (e.g. power/timing/area) values. More specifically, target-delay and utilization are crucial parameters in synthesis flows, which define timing and area constraints. The tool adopts different strategies internally to satisfy that target-delay which we can hardly consider during prefix graph synthesis. On the other hand, changing utilization values can lead to significantly distinct layouts. So we consider these two synthesis parameters as attributes of a node. In fact, besides the target delay and utilization, we also attempt other tool settings. The optimization level setting in logical synthesis potentially impacts on the performance of adders, which can be configured by `compile` and `compile_ultra` commands with different options. However, after synthesizing, it is observed that the solutions generated with `compile_ultra` can significantly dominate the solutions generated by `compile`. Hence, this setting is fixed to `compile_ultra` level as we are aiming at superior designs.

For a better understanding, we have drawn the 4-bit prefix adder graph with its adjusted adjacency matrix and node feature matrix as an explicit example of mapping in the top part of Fig. 5.4. It can be seen that the 4-bit prefix adder graph has 8 nodes, and the shape of the adjusted adjacency matrix is $8*8$. "1" in the adjusted adjacency matrix refers to the existence of a logical relationship between two nodes, and "0" vice versa. The first row in $\boldsymbol{X}$ in Fig. 5.4 means that the n_0 node is in the logic level 1, and it is an input node connecting one node in the next logic level, while the EDA tool settings (target delay and utilization) are 0.1 and 0.4 respectively.

5.3.2 Backbone: The Encoder of Graph Auto-encoder

It is widely known that in graph theory, a tree is an undirected graph in which any two vertices are connected by exactly one path,

or equivalently a connected acyclic undirected graph. Hence, the structure of a prefix adder network can be recognized as a tree-like graph. Recently, the graph autoencoders (GAEs) [79] and variational graph autoencoders (VGAEs) [79] are proposed as powerful node embedding methods to describe graphs. In view of this, we adopt a graph autoencoder structure to comprehensively capture the latent information of prefix adder. The encoder-decoder architecture is made up of two components. For the encoder part, it encodes adder graphs into the latent feature representations in a low dimensional vector space, while the decoder component tries to reconstruct the original graph structure based on the information passing through the encoding network.

Our design for Stream I is based on a two-layer GCN [63] encoder. The intuitive idea of the design is mapping each node $i \in \mathcal{V}$ to a latent vector $\boldsymbol{z}_i \in \mathbb{R}^d$ ($d \ll n$). More precisely, the $n \times d$ matrix $\boldsymbol{Z}$ with all $\boldsymbol{z}_i$ vectors as rows is generated by the two-layer GCN processing $\boldsymbol{A}$.

We build a probabilistic model inferring latent variable $\boldsymbol{z}_i \in \mathbb{R}^d$ ($d \ll n$) that is the latent representation in an embedding space for each node i. The encoder parameterized by $\boldsymbol{\phi}$ is defined in Equation (5.6), which is also called the inference model of the VGAE.

$$q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A}) = \prod_{i=1}^{n} q_{\boldsymbol{\phi}}(\boldsymbol{z}_i|\boldsymbol{X},\boldsymbol{A}) = \prod_{i=1}^{n} \mathcal{N}(\boldsymbol{z}_i|\boldsymbol{\mu}_i, \mathrm{diag}({\boldsymbol{\sigma}_i}^2)). \quad (5.6)$$

The parameters of approximated Gaussian distribution, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, are learnt by a two-layer GCN. It takes the adjusted adjacency matrix $\boldsymbol{A}$ and the feature matrix $\boldsymbol{X}$ as inputs and generates the distribution parameters for the latent variable $\boldsymbol{Z}$. The first layer of the GCN outputs a lower-dimensional feature matrix $\bar{\boldsymbol{X}}$ shown in Equation (5.7):

$$\bar{\boldsymbol{X}} = \mathrm{ReLU}(\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_0), \quad (5.7)$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}(\boldsymbol{A})\boldsymbol{D}^{-1/2}$ with the degree matrix $\boldsymbol{D}$ is the symmetrically normalized adjacency matrix, and $\mathrm{ReLU} = \max(0, \cdot)$.

The second layer of the GCN generates $\boldsymbol{\mu}$ and $\log\boldsymbol{\sigma}$, respectively:

$$\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(\tilde{\boldsymbol{A}}, \bar{\boldsymbol{X}}) = \tilde{\boldsymbol{A}}\bar{\boldsymbol{X}}\boldsymbol{W}_1, \tag{5.8}$$

$$\log\boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{A}}, \bar{\boldsymbol{X}}) = \tilde{\boldsymbol{A}}\bar{\boldsymbol{X}}\boldsymbol{W}_1'. \tag{5.9}$$

Note that $\mathrm{GCN}_{\boldsymbol{\sigma}}(\tilde{\boldsymbol{A}}, \bar{X})$ and $\mathrm{GCN}_{\boldsymbol{\mu}}(\tilde{\boldsymbol{A}}, \bar{\boldsymbol{X}})$ only share the first layer parameter $\boldsymbol{W}_0$. Two-layer GCN generates the parameters for a distribution where the latent variable $\boldsymbol{Z}$ can be sampled. Since sampling is unexpected to be involved in the back-propagation while $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ need to be kept in the computational graph to update the GCN layers, the re-parameterization trick is exploited. So $\boldsymbol{Z}$ is not directly sampled from a normal distribution that is approximated by the encoder, instead, an auxiliary variable $\boldsymbol{\epsilon}_i$ is sampled from the standard normal distribution. By making use of the re-parameterization trick, each row in the latent variable $\boldsymbol{Z}$ can be obtained according to Equation (5.10). Through taking the mean of rows of $\boldsymbol{Z}$, the latent representation for an adder is acquired.

$$\boldsymbol{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\epsilon}_i \odot \boldsymbol{\sigma}_i, \tag{5.10}$$

where $\boldsymbol{z}_i$, $\boldsymbol{\mu}_i$ and $\boldsymbol{\epsilon}_i$ are the rows of matrix $\boldsymbol{z}$, $\boldsymbol{\mu}$ and $\boldsymbol{\epsilon}$ respectively, and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ with $\odot$ the Hadamard product. Here is the proof in 1-D case for re-parameterization trick applied in Equation (5.10), which can be readily extended to multi-dimensional case.

$$\begin{aligned} &\int \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right) dz \\ =&\int \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{z-\mu}{\sigma}\right)^2\right] d\left(\frac{z-\mu}{\sigma}\right) \\ =&\int \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}(\epsilon)^2\right] d(\epsilon). \end{aligned} \tag{5.11}$$

Therefore, $z = \mu + \epsilon\sigma$.

5.3.3 Stream I: The Decoder of Graph Auto-encoder

To reconstruct the graph, a simple inner product decoder parameterized by $\boldsymbol{\theta}$ is stacked to the aforementioned GCN layers. This

module leverages the inner product between latent variable pairs $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$, shown in Equation (5.12), to output the reconstructed adjusted adjacency matrix $\hat{\boldsymbol{A}}$. The reason for performing the inner product is that it could calculate the cosine similarity between rows in the latent variable $\boldsymbol{Z}$ with being invariant to the magnitude of the vectors. By applying the inner product on the latent variables $\boldsymbol{Z}$ and $\boldsymbol{Z}^\top$, the similarities among nodes in latent vector space can be learned to predict $\boldsymbol{A}$.

$$p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z}) = \prod_{i=1}^{n}\prod_{j=1}^{n} p_{\boldsymbol{\theta}}\left(A_{ij}|\boldsymbol{z}_i, \boldsymbol{z}_j\right), \tag{5.12}$$

in which $p_{\boldsymbol{\theta}}(A_{ij} = 1|\boldsymbol{z}_i, \boldsymbol{z}_j) = \sigma(\boldsymbol{z}_i^\top \boldsymbol{z}_j)$, $\sigma(\cdot)$ is the sigmoid activation function with $\sigma(x) = 1/\left(1 + e^{-x}\right)$. Obviously, the larger the inner product $\boldsymbol{z}_i^\top \boldsymbol{z}_j$ is, the more likely nodes i and j will be connected.

As minimizing a reconstruction loss in GAE, the weights of neural networks in our model of Stream I are updated by maximizing a tractable variational evidence lower bound (ELBO) of the model's marginal likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{A})$ through gradients. This kind of trick is employed in [29, 30] as well. Hence, the loss function $\mathcal{L}_{autoencoder}$ can be rewritten as minimizing the opposite of the ELBO, which is shown as follows:

$$\min_{\boldsymbol{\phi},\boldsymbol{Z},\boldsymbol{\theta}} \mathrm{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{A})\|p(\boldsymbol{Z})] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z})]. \tag{5.13}$$

The complete derivation of Equation (5.13) is based on Jensen's inequality. Bear in mind that for a convex function f with a random variable $\boldsymbol{x}$ as input, $\mathbb{E}[f(\boldsymbol{x})] \geq f(\mathbb{E}[\boldsymbol{x}])$ holds. On the contrary, if f

is concave, $\mathbb{E}[f(\boldsymbol{x})] \leq f(\mathbb{E}[\boldsymbol{x}])$ holds.

$$
\begin{aligned}
& \log p_{\boldsymbol{\theta}}(\boldsymbol{A}) \\
= & \log \int p(\boldsymbol{Z}) p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z}) d\boldsymbol{Z} \\
= & \log \int q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X}) \frac{p(\boldsymbol{Z})}{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})} p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z}) d\boldsymbol{Z} \\
= & \log(\mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})}[\frac{p(\boldsymbol{Z})}{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})} p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z})]) \\
\geq & \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})}[\log(\frac{p(\boldsymbol{Z})}{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})} p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z}))] \\
= & \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})}[\log \frac{p(\boldsymbol{Z})}{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})}] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{A},\boldsymbol{X})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z})] \\
= & -\mathrm{KL}[q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A}) \| p(\boldsymbol{Z})] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{A}|\boldsymbol{Z})].
\end{aligned}
\tag{5.14}
$$

In Equation (5.13), the first part, $\mathrm{KL}(q(\cdot)\|p(\cdot))$, is the Kullback-Leibler divergence between $q(\cdot)$ and $p(\cdot)$. It measures the "distance" between $q_{\boldsymbol{\phi}}(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})$ and $p(\boldsymbol{Z})$, where $p(\boldsymbol{Z}) = \prod_i p(\boldsymbol{z}_i) = \prod_i \mathcal{N}(\boldsymbol{z}_i|0,\boldsymbol{I})$. The computational rule for KL divergence term is shown as follows. Since the loss can be calculated by each dimension of the latent variable $\boldsymbol{Z}$, we exemplified the calculation in 1-D dimension. Note that, from Equation (5.17) to Equation (5.18), the integral of the probability density over the whole space equalling to one, the mathematical definitions of the second moment and the variance are used. Since our decoder is a Bernoulli-based model, the second term can be readily transformed into the binary cross-entropy loss function. Generally, by gradually minimizing the reconstruction loss (i.e. Equation (5.13)), we obtain increasing better latent representations of adders. It is worth mentioning that the purpose of the decoder is not perfectly reconstructing the adjacency matrix. More importantly, it plays the role of calibration the encoder part to find

good mapping.

$$
\begin{aligned}
&\mathrm{KL}(\mathcal{N}(\mu,\sigma^2)||\mathcal{N}(0,1)) \\
=&\int \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \left(\log \frac{e^{-(x-\mu)^2/2\sigma^2}/\sqrt{2\pi\sigma^2}}{e^{-x^2/2}/\sqrt{2\pi}} \right) dx && (5.15) \\
=&\int \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \times \\
&\log \left\{ \frac{1}{\sqrt{\sigma^2}} \exp \left\{ \frac{1}{2} \big[x^2 - (x-\mu)^2/\sigma^2 \big] \right\} \right\} dx && (5.16) \\
=&\frac{1}{2} \int \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2} \Big[-\log\sigma^2 + x^2 - (x-\mu)^2/\sigma^2 \Big] dx && (5.17) \\
=&-\frac{1}{2} \big[\log\sigma^2 - (\mu^2+\sigma^2) + 1 \big]. && (5.18)
\end{aligned}
$$

5.3.4 Stream II: Neural Processes

Function approximation plays a core role in numerous machine learning problems. One representative approach is the Gaussian process (GP) [146]. GPs do not require an expensive training stage and can perform inference about the ground truth function conditioned on some observations, which makes them very flexible at testing. Nevertheless, traditional GPs are computationally costly due to cubical scaling concerning the number of data-points. Thus, they are computationally expensive and their applicabilities are limited. On the other hand, a neural network (NN) can be regarded as a parameterized function. Recent works reveal that combining desirable properties of GPs and NNs, a collection of latent variables which are modeled as neural networks can learn an approximation of a stochastic process [47, 48, 78]. Like GPs, this kind of formulation termed as Neural Process (NP) can provide the predictions as well as uncertainties on the QoR metric value of an adder design. Inheriting from NNs, NPs are more computationally efficient than GPs. [118] mathematically demonstrates that under certain conditions, NPs are mathematically equivalent to GPs with deep kernels, while [47] experimentally demonstrate that when the number of context points

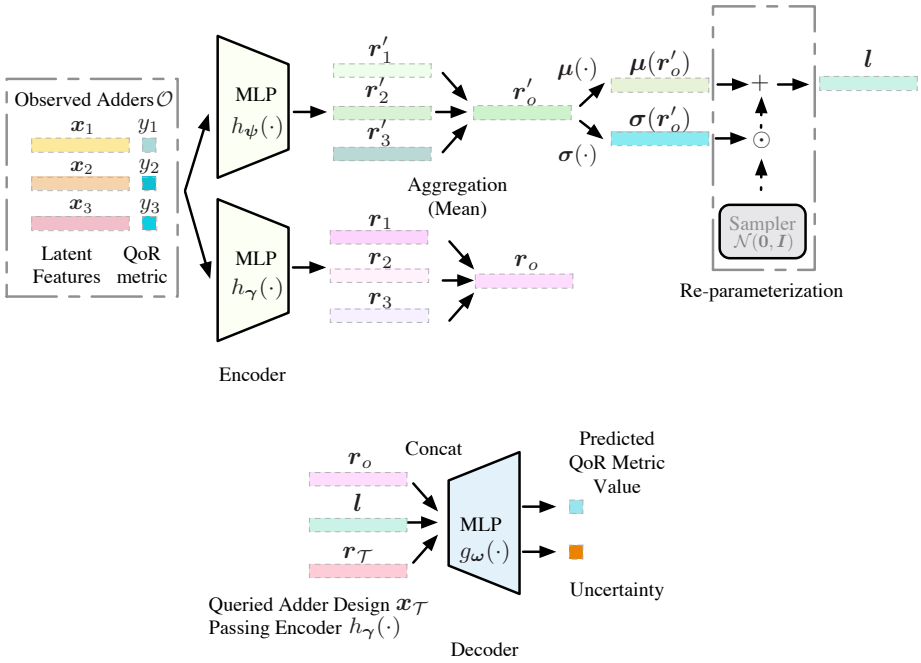


Figure 5.5 The detailed regime of the neural process.

is small, the neural process outperforms the GP in terms of mean squared error metric in the image completion task. The conclusion is essential to our task since the adder DSE process works in a circumstance where the tool evaluation is expensive. Therefore, we adopt the neural process in lieu of the Gaussian process.

The Structure

In Stream II, we harness the NP as a good replacement to the traditional Gaussian process, which takes the latent representations of adders as input $\boldsymbol{x}$ and the associated QoR metric values as input $\boldsymbol{y}$ during training. NP is implemented in an encoder-decoder framework, which is given in Fig. 5.4. More specifically, the architecture includes two multi-layer perceptrons (MLPs) based encoders and a MLP based decoder. One encoder $h_{\boldsymbol{\gamma}}(\cdot)$ maps the adder representations from the original space into the representation space to produce a representation $\boldsymbol{r}_i = h_{\boldsymbol{\gamma}}\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right), \forall\left(\boldsymbol{x}_i, \boldsymbol{y}_i\right) \in \mathcal{O}$ for each of

the pairs, where $\mathcal{O}$ represents a dataset consisting of observations in pairs $\{\boldsymbol{x}_{\mathcal{O}}, \boldsymbol{y}_{\mathcal{O}}\}$ and $h_{\boldsymbol{\gamma}}(\cdot)$ stands for the encoder parametrized by $\boldsymbol{\gamma}$. Then, by taking the mean of these representations, $\boldsymbol{r}_o$ is generated. Another encoder $h_{\boldsymbol{\psi}}(\cdot)$ is built for parameterizing the distribution of a latent variable $\boldsymbol{l}$. Conditioned on observations, NPs define conditional distributions of the latent variable and thus model a dataset $\mathcal{T}$ which is made up of a collection of target pairs defined as $\{\boldsymbol{x}_{\mathcal{T}}, \boldsymbol{y}_{\mathcal{T}}\}$. By virtue of the idea, the conditional decoder $g_{\boldsymbol{\omega}}(\cdot)$ treats the sampled global latent variable $\boldsymbol{l}$ and the input representation as well as target data $\boldsymbol{x}_{\mathcal{T}}$ (new queried data) as input and outputs the corresponding predictions with their uncertainties. The details of the neural process are visualized in Fig. 5.5. To describe an NP, a Gaussian likelihood is given in Equation (5.19):

$$p(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{l}, \boldsymbol{x}_{\mathcal{T}}, \boldsymbol{x}_{\mathcal{O}}, \boldsymbol{y}_{\mathcal{O}}) = p\left(\boldsymbol{l}\right) \prod_{j=1}^{|\mathcal{T}|} \mathcal{N}\left(\boldsymbol{y}|g_{\boldsymbol{\omega}}(\boldsymbol{l}, \boldsymbol{r}_{\mathcal{O}}), \tau^{-1}\boldsymbol{I}\right), \tag{5.19}$$

where the prior $p\left(\boldsymbol{l}\right)$ is assumed as a multivariate standard normal distribution function. Following the same variational idea in VGAEs [79], to perform approximate inference in the NP, a variational posterior is defined in Equation (5.20).

$$q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{O}}, \boldsymbol{y}_{\mathcal{O}}\right) = \mathcal{N}\left(\boldsymbol{l}|\boldsymbol{\mu}\left(h_{\boldsymbol{\psi}}\left(\boldsymbol{x}_{\mathcal{O}}, \boldsymbol{y}_{\mathcal{O}}\right)\right), \boldsymbol{\sigma}\left(h_{\boldsymbol{\psi}}\left(\boldsymbol{x}_{\mathcal{O}}, \boldsymbol{y}_{\mathcal{O}}\right)\right)\right), \tag{5.20}$$

where $\boldsymbol{\mu}(\cdot)$ (i.e. mean or location) and $\boldsymbol{\sigma}(\cdot)$ (i.e. the diagonal of the covariance matrix) take aggregated and encoded input-output pairs as inputs and parameterize a normal distribution from which $\boldsymbol{l}$ is sampled [48]. The dense layer is often applied to mimic $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\sigma}(\cdot)$. Intuitively, the latent variable $\boldsymbol{l}$ is designed to capture all information about the data-generating process needed to make predictions on the target inputs. Besides, introducing such a latent variable gives the most expressive model. Similarly, the re-parameterization trick is performed for latent variable $\boldsymbol{l}$ due to exploiting the variational idea.

Loss function

By using the variational distribution in Equation (5.20), an evidence lower bound on the log marginal likelihood is given as follows. Considering the observations and targets, Equation (5.21) reflects the desired model behavior of an NP.

$$\begin{aligned}\log p\left(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right) \geq \mathbb{E}_{q(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}})}\left[\log p\left(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{l},\boldsymbol{x}_{\mathcal{T}}\right)\right] - \\ \mathrm{KL}\left(q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}}\right)\|p\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right)\right).\end{aligned} \tag{5.21}$$

In Equation (5.21), the intractable conditional prior $p\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right)$ replaces the prior $p(\boldsymbol{l}) = \mathcal{N}(\boldsymbol{l};\boldsymbol{0},\boldsymbol{I})$. Through approximating the intractable conditional prior, Equation (5.21) can be reformulated as:

$$\begin{aligned}\log p\left(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right) \geq \mathbb{E}_{q(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}})}\left[\log p\left(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{l},\boldsymbol{x}_{\mathcal{T}}\right)\right] - \\ \mathrm{KL}\left(q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}}\right)\|q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right)\right).\end{aligned} \tag{5.22}$$

Meanwhile, we acquire the loss function $\mathcal{L}_{np}$ (the opposite of the RHS of Equation (5.22)):

$$\begin{aligned}\min_{\boldsymbol{\psi},\boldsymbol{\gamma},\boldsymbol{\omega}} \mathrm{KL}\left(q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}}\right)\|q\left(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{O}},\boldsymbol{y}_{\mathcal{O}}\right)\right) - \\ \mathbb{E}_{q(\boldsymbol{l}|\boldsymbol{x}_{\mathcal{T}},\boldsymbol{y}_{\mathcal{T}})}\left[\log p\left(\boldsymbol{y}_{\mathcal{T}}|\boldsymbol{l},\boldsymbol{x}_{\mathcal{T}}\right)\right].\end{aligned} \tag{5.23}$$

According to Equation (5.23), the NP learns to reconstruct targets, regularized by a KL term that encourages the summary of the observations to be not too far from the summary of the targets.

5.3.5 The Graph Neural Process

The loss function for the whole framework is the simple addition of losses of two branches (i.e. $\mathcal{L}_{autoencoder}$ in Equation (5.13) and $\mathcal{L}_{np}$ in Equation (5.23)). During training, via back-propagation, the guide information (i.e. gradients) from the graph decoder and the neural process updates each part respectively and then calibrates the encoder in graph autoencoder collectively. The two branches are jointly performed and mutually benefited. When conducting

inference, the prefix adder first goes through the graph encoder to convert into the latent feature embedding, and then the embedding will be fed into the neural process to acquire the prediction and uncertainty of the associated QoR metric value. Our graph neural process not only finds good graph feature embeddings of prefix adders but also builds a pretty good regressor for adder performance value. By the merit of the end-to-end nature, GNP is fast and flexible for both training and inference phases.

5.4 The proposed DSE framework

In the adder DSE problem, exhaustively determining golden QoR metric (e.g. area/power/delay) values of each adder design by running EDA flow is time-intensive. The sequential optimization-based algorithm approximates the tool evaluations with a surrogate which is cheaper to evaluate. Therefore, it is the cure to combat our problem. On the other hand, considering that sequential-optimization model [14, 175] iteratively and incrementally samples the data to calibrate the surrogate model, the number of data-points to train a machine learning-based surrogate model can be much less than the one in conventional DSE flow. By virtue of the incremental sampling stage, a method for simultaneous prediction and uncertainty estimation is in demand. Prior art [94] utilizes a Gaussian process as the surrogate model to offer predictions with uncertainties, which has high computational complexity.

As aforementioned in Section 5.3, we incorporate the feature extractor into the sequential optimization framework so that the update of the learning model in the framework also benefits the feature extraction process. To reduce the high computation cost brought by a Gaussian process model, we exploit a neural process which is based on the neural network structure. It can provide predictions as well as uncertainty estimations within a lower computational complexity [47, 48]. More importantly, the feature extractor is com-

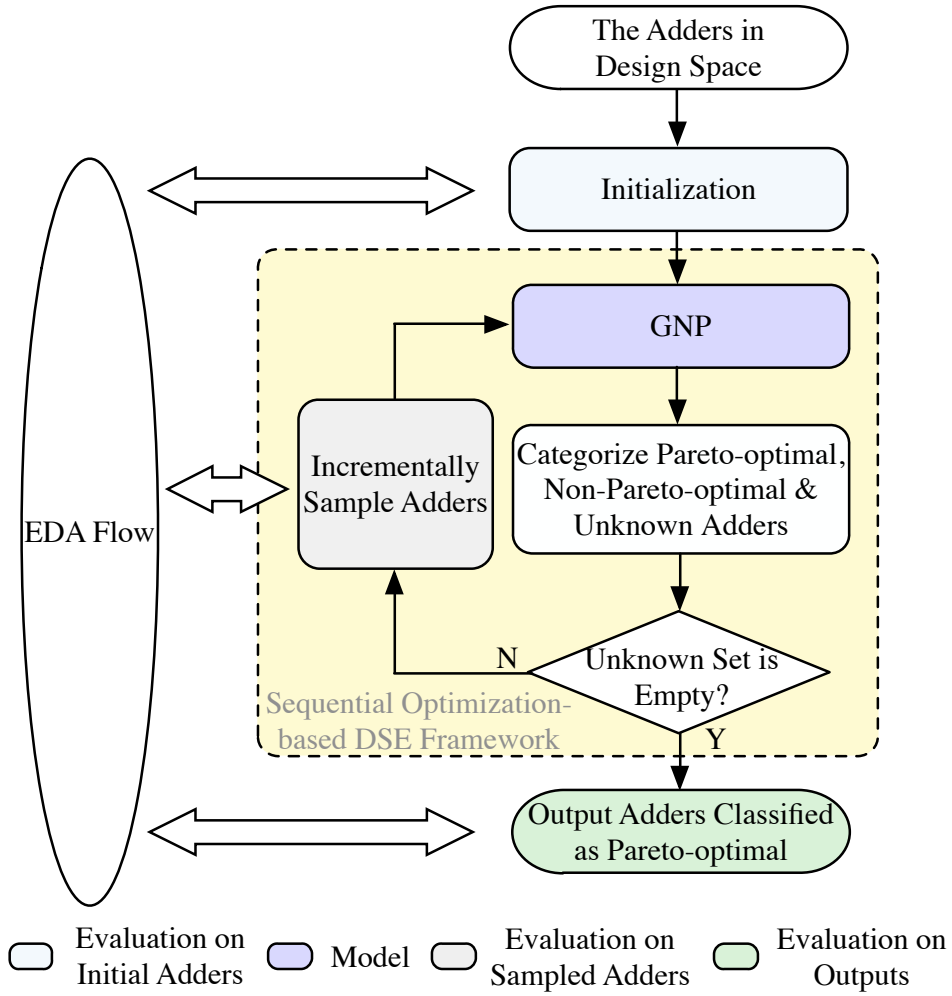


Figure 5.6 The workflow of sequential optimization-based DSE framework.

bined with the neural process to form an end-to-end, multi-branch model. Based on the above, we propose a sequential optimization-based DSE framework with the graph neural process as the surrogate model, which is shown in Fig. 5.6.

During the initialization, the proposed sequential optimization-based DSE framework interacts with EDA tools to obtain the golden QoR metric (area/power/delay) values of a small number of prefix adders which are randomly sampled from the entire adder design space $\mathbb{E}$. Afterwards, the DSE framework start working iteratively. Our graph neural process is first calibrated with the initial data.

The trained GNP outputs the QoR metric values of the adder designs in design space $\mathbb{E}$ with prediction uncertainties. The proposed DSE framework paradigm tries to classify the input adder designs based on the GNP's outputs into three classes: Pareto-optimal, non-Pareto-optimal, and unknown. During iterations, it incrementally selects the most representative adder designs as candidates for EDA flow (including synthesis, placement and routing tools) evaluation towards a goal of minimizing the size of the unknown set. By harnessing the representative data along with their ground-truth QoR metric values, the GNP is updated. As more and more adder designs being selected, the GNP gets more and more accurate. The whole DSE process is terminated when the number of maximum iterations is reached or the unknown set is empty.

For the proposed DSE framework, the classification rules and selecting rules are based on the predictions and uncertainties offered by the GNP. More specifically, in each iteration, the GNP is invoked to infer the predictions as well as the uncertainties on QoR metrics over all un-sampled adder $\boldsymbol{x}$ in the adder design space $\mathbb{E}$. A vector $\boldsymbol{m}(\boldsymbol{x})$ which includes concatenated QoR metric predictions (e.g. (area, power, delay)) of $\boldsymbol{x}$ and a vector of corresponding standard deviations $\boldsymbol{\sigma}(\boldsymbol{x})$ are acquired. Consequently, a hyper-rectangle $\mathcal{H}(\boldsymbol{x})$ is built to represent the prediction uncertainty in QoR metric space for a prefix adder design $\boldsymbol{x}$, which is defined as follows.

$$\mathcal{H}(\boldsymbol{x}) := \{\boldsymbol{y} \mid \boldsymbol{m}(\boldsymbol{x}) - \beta^{\frac{1}{2}}\boldsymbol{\sigma}(\boldsymbol{x}) \preceq \boldsymbol{y} \preceq \boldsymbol{m}(\boldsymbol{x}) + \beta^{\frac{1}{2}}\boldsymbol{\sigma}(\boldsymbol{x})\}, \quad (5.24)$$

where one element y_i in $\boldsymbol{y}$ (i.e. a point in the QoR metric space) refers to the coordinate of associated axes with $i \in \{1, 2, 3\}$ indicating different QoR metrics (area/power/delay), and β is a scaling parameter that controls the contribution of each element of $\boldsymbol{\sigma}$ to the hyper-rectangle. The uncertainty information is exploited to guide the consequent sampling and to make a probabilistic assumption on the Pareto-optimality of every adder design $\boldsymbol{x}$. With the continuously updating GNP with new evaluated designs, the confidence of

prediction increases. In light of ever-shrinking uncertainty, the uncertainty region of an adder design $\boldsymbol{x}$ in the t-th iteration is written as

$$\mathcal{R}_t(\boldsymbol{x}) := \mathcal{R}_{t-1}(\boldsymbol{x}) \cap \mathcal{H}(\boldsymbol{x}). \tag{5.25}$$

The initial $\mathcal{R}_{-1}$ is the entire objective space $\mathbb{R}^n$ with n the number of QoR metrics. The iterative intersection guarantees the uncertainty regions are non-increasing. In the uncertainty region $\mathcal{R}_t(\boldsymbol{x})$, the QoR metric performance upper-bound comes from the optimistic prediction $\min(\mathcal{R}_t(\boldsymbol{x}))$, while the lower-bound is associated with the pessimistic prediction $\max(\mathcal{R}_t(\boldsymbol{x}))$ of one certain adder design $\boldsymbol{x}$.

On account of the fact that the sequential optimization process in the DSE framework is monotonic, the numbers of designs in Pareto-optimal set $\mathcal{P}_t$ and non-Pareto-optimal set $\mathcal{N}_t$ are non decreasing regarding the iteration number t. In other words, at iteration t, the previous predicted design points in $\mathcal{P}_{t-1}$ and $\mathcal{N}_{t-1}$ remain their classification. Only designs in unknown set $\mathcal{U}_t$ need to be classified. With relaxing by a tolerance parameter $\boldsymbol{\epsilon}$ on both sides, the classification of a prefix adder design $\boldsymbol{x}$ abides by the following rules based on inequalities. If the pessimistic QoR metric prediction of $\boldsymbol{x}$, $\max(\mathcal{R}_t(\boldsymbol{x}))$, is not dominated by optimistic results of any other design $\boldsymbol{x}'$ in $\mathbb{E}$ (i.e. $\min(\mathcal{R}_t(\boldsymbol{x}'))$), then $\boldsymbol{x}$ is classified as Pareto-optimal:

$$\max(\mathcal{R}_t(\boldsymbol{x})) - \boldsymbol{\epsilon} \preceq \min(\mathcal{R}_t(\boldsymbol{x}')) + \boldsymbol{\epsilon}. \tag{5.26}$$

If the optimistic outcomes of $\boldsymbol{x}$, $\min(\mathcal{R}_t(\boldsymbol{x}))$, is dominated by the pessimistic QoR metric prediction of any other design $\boldsymbol{x}'$ in $\mathbb{E}$ (i.e. $\max(\mathcal{R}_t(\boldsymbol{x}'))$), then $\boldsymbol{x}$ is classified as non-Pareto-optimal:

$$\max(\mathcal{R}_t(\boldsymbol{x}')) - \boldsymbol{\epsilon} \preceq \min(\mathcal{R}_t(\boldsymbol{x})) + \boldsymbol{\epsilon}. \tag{5.27}$$

If the above rules do not exist regarding $\boldsymbol{x}$, then $\boldsymbol{x}$ is still in the unknown set.

When the classification finishes, a prefix adder design $\boldsymbol{x}_t^s$ with the longest diagonal of its uncertainty region $\mathcal{R}_t(\boldsymbol{x})$ is sampled from Pareto-optimal and unknown categories for tool evaluation. The

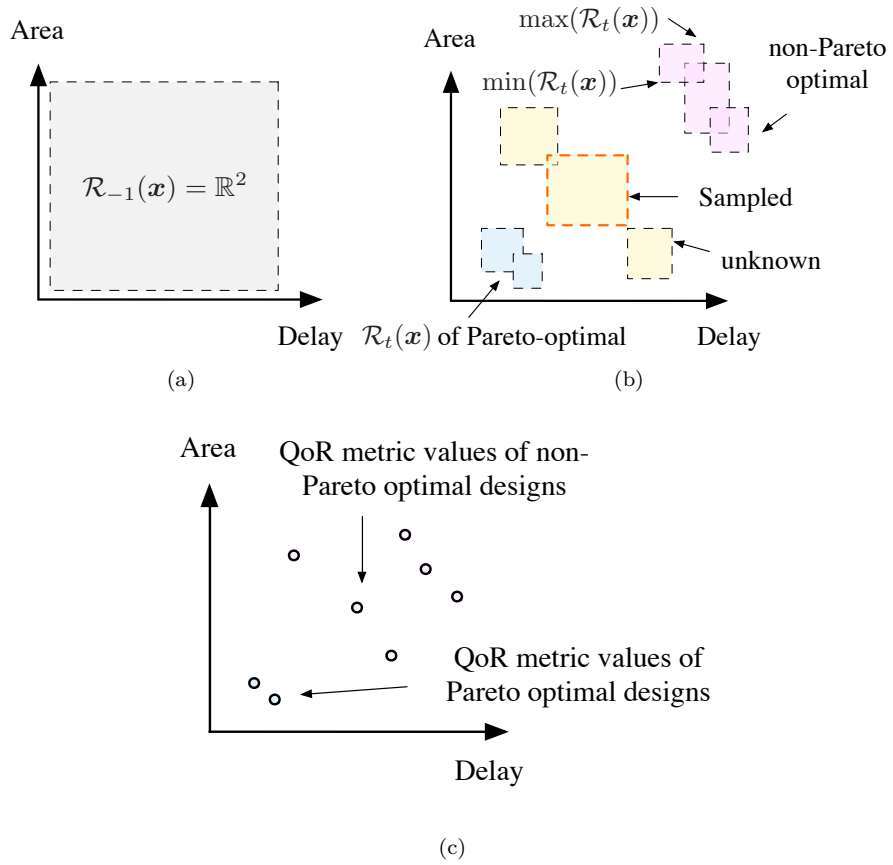


Figure 5.7 An example of mathematical principles of the sequential optimization-based DSE framework. (a) Before starting; (b) Classification and sampling in one iteration; (c) After the termination of the stop criteria.

sampling rule can be written in Equation (5.28).

$$\boldsymbol{x}_t^s := \arg\max_{\boldsymbol{y},\boldsymbol{y}' \in \mathcal{R}_t(\boldsymbol{x})} \|\boldsymbol{y} - \boldsymbol{y}'\|_2 . \tag{5.28}$$

The GNP model calibration and prediction, the classification of prefix adder designs, and adder design incremental sampling perform alternatively in iterations until the stopping criteria (exceeds the maximum iterations or the unknown set is empty) meets. Eventually, the predicted Pareto-optimal adder designs are evaluated by running EDA flow. For a better understanding, we visualize the mathematical principles of the DSE framework in Fig. 5.7.

In a nutshell, the hyper-parameters used in the DSE framework

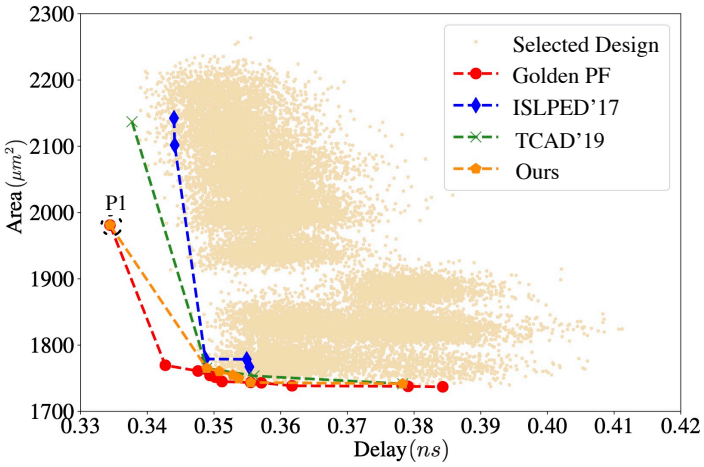
Table 5.1 The quantitive comparison of the Pareto frontiers.

Multi-objective	ISLPED'17 [116]			TCAD'19 [94]			Ours		
	HV	ADRS	Flows	HV	ADRS	Flows	HV	ADRS	Flows
`Area-Delay`	0.150	0.029	598	0.144	0.028	434	0.126	0.026	413
`Power-Delay`	0.155	0.017	615	0.168	0.021	482	0.128	0.010	422
`Area-Power-Delay`	0.110	0.019	1995	0.098	0.012	1636	0.073	0.008	1470
Average	0.138	0.022	1069	0.136	0.020	851	**0.109**	**0.015**	**768**
Ratio	1.266	1.467	1.392	1.248	1.333	1.108	**1.000**	**1.000**	**1.000**

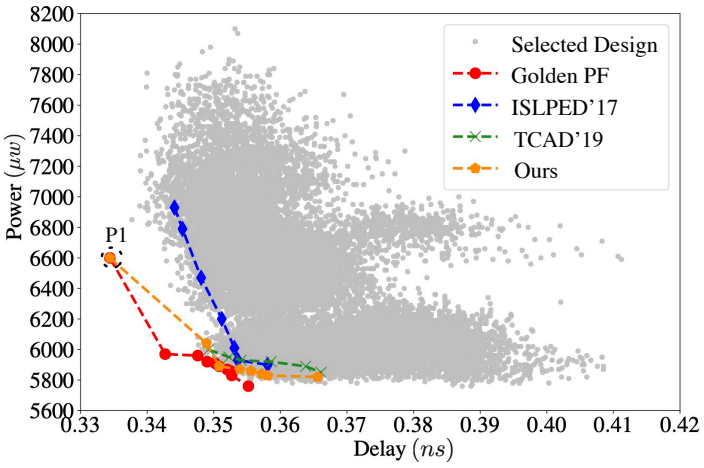
embrace the scaling parameter β controlling the hyper-volume of a hyper-rectangle when calculating the uncertainty region, the tolerance parameter ϵ in classification rules, the max-iterations T_{max} in the stop condition. For the β, if we follow the value setting rule like $\beta_t = 2\log\left(n|\mathbb{E}|\pi^2 t^2/(6\delta)\right)$, a maximum hypervolume error can be achieved with a high probability $1-\delta$ where $\delta \in (0,1)$. This conclusion is proved in [175]. With regarding the ϵ, we adopt the cross-validation method on our initial dataset to determine the value. When it comes to T_{max}, theoretically, it is the smallest number satisfying

$$\frac{\sqrt{T\log\left(1-\sigma^{-2}\right)}}{4K\sqrt{\log\left(n|\mathbb{E}|\pi^2 T^2/(6\delta)\right)\log^{d+1}T}} \geq \frac{na^{n-1}}{\eta(n-1)!}, \tag{5.29}$$

where η refers to the hyper-volume error, a indicates the maximum embedding distance between two adder designs after one iteration, d is the dimensionality of the latent feature vector of a prefix adder design, and n denotes the dimension of QoR metric space with σ the standard deviation of the Gaussian distribution characterizing one QoR metric. In the practical experiment, we have found the sequential optimization-based DSE process finishes classification for the entire design space $\mathbb{E}$ after 10 iterations. We relax the T_{max} to 20 so that the proposed whole framework can finish the optimization process within an acceptable sampling budget.



(a)



(b)

Figure 5.8 (a) Pareto Frontier: area vs. delay; (b) Pareto Frontier: power vs. delay.

5.5 Experimental Results

5.5.1 Experimental Settings

The implementation of our framework is in `Python` with the `Pytorch` library [112], and we test it on a platform with an Xeon Silver 4114 CPU processor and an nVIDIA TITAN Xp Graphic card. To verify the effectiveness and the efficiency of our framework, we compare our framework with the best-of-breed methods [94, 116] by exploring the 64-bit adder design space. First, we visually compare

the quality of the Pareto frontiers found by the proposed framework and [94, 116] in Fig. 6.6, and then quantitatively evaluate associated performance of these methods by the hypervolume error, the ADRS and the number of design flow runs in TABLE 5.1. The former two metrics measure the quality of Pareto frontiers and the rest estimates time expense. Next, we exemplify one predicted Pareto-optimal prefix adder design found by the proposed algorithm in Fig. 5.9 to show the corresponding architecture and the layout. The corresponding time analyses of the exploration are provided by Figs. 5.10 and 5.11. Ultimately, the comparison between explored adders against some classical adders and a state-of-the-art adder synthesis algorithm [115] is depicted in TABLE 5.2.

It is impossible to exhaustively enumerate all adder designs in the infinite design space for tool evaluation. A set of 22000 adders, which is reasonable and comparatively large (almost 80 days are needed to run the flow), is generated and selected to represent the entire design solution space. This dataset is sampled in a Quasi-random manner, which is based on a two-level (max-fan-out constraint and size) binning scheme. The approach evenly samples the prefix adders covering different architectural bins. The first level of binning is based on the max-fan-out constraint. As many architectures may exist under the same max-fan-out constraint, another level relies on the size of the prefix adder. Note that, we follow the prefix adder generation method proposed in [94], but the obtained design space is much larger than that in [94]. All the designs need to go through the whole design flow including front-end stages like logical synthesis and back-end steps such as placement and routing. The tools in the design flow include Design Compiler [1] (version F-2011.09-SP3) for logical synthesis and IC Compiler [2] (version J-2014.09-SP5-3) for the placement and routing. `"tt1p05v125c"` corner and Non-Linear Delay Model (NLDM) in $32nm$ SAED cell-library for LVT class [3] (available by University Program) is used for technology mapping. Primary input activity of 0.1 is used along

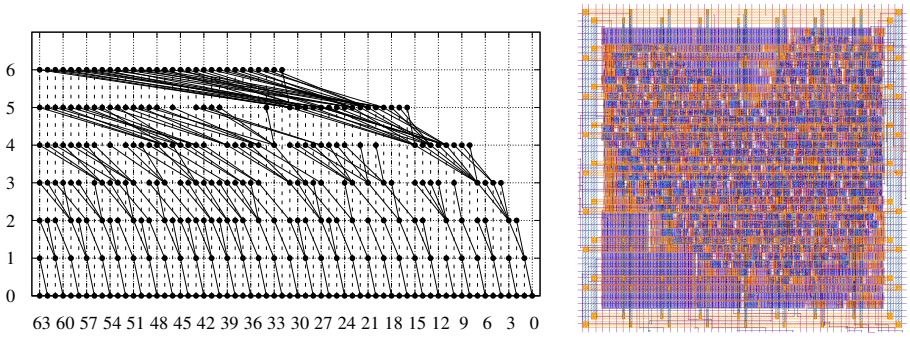
with 1GHz operating frequency for power estimation. Concerning the tool settings, $0.1ns$, $0.2ns$, $0.3ns$ and $0.4ns$ are chosen as target delays. Utilization values are configured with 0.4, 0.5, 0.6, 0.7 and 0.8. Per the design flow, it costs about 5.5 minutes. Although the involved generation overheads seem high, we utilize the dataset to demonstrate the superiority of the proposed methodology against SOTA DSE works and manual design processes for some classical adders. The proposed methodology can effectively aid and accelerate the adder design process under the current technology node. On the other hand, the dataset may lay the foundation of our future work like transfer learning among different bit-width adders or distinct technology nodes.

Our framework, as well as the previous works [94, 116], explores Pareto frontiers in area vs. delay space, power vs. delay space, and area vs. power vs. delay space. Fig. 5.8(a) and Fig. 5.8(b) visualize the corresponding Pareto frontiers discovered by the prior arts [94, 116] and our framework in two kinds of 2-d objective spaces, respectively. In Fig. 6.6, each dot in the delay vs. area or delay vs. power space indicates the selected representative adder design after going through the design flow. Except the red dots indicate the associated positions of Pareto-optimal designs in QoR metric spaces with a straight line connecting these dots standing for the frontier, the other dash lines of different colors and with distinct markers refer to the Pareto frontiers found by the listed DSE algorithms. It is interesting to note that there is one point ("P1" outlined at the extreme left in Figs. 5.8(a) and 5.8(b)) which has 5ps better delay than those of other points, causing this point to be at some distance than the data cloud. Since we have connected the Pareto points by straight lines, the shift of this single point resulted in the "noticeable" shift from the cloud of data points. According to the observations of Fig. 5.8(a) and Fig. 5.8(b), both Pareto frontiers searched by our approach (orange lines) are much closer to the corresponding golden frontiers (red lines) than the frontiers explored by

other methods. In other words, Fig. 6.6 qualitatively demonstrates our method finds the Pareto frontier of better quality.

5.5.2 The comparisons against DSE-based SOTA works

The supplied TABLE 5.1 summarizes the qualities of Pareto frontiers including the Pareto frontiers of 2-D QoR metrics spaces presented in Figs. 5.8(a) and 5.8(b) and a 3-D case in quantification. Note that we utilize the golden Pareto set as the reference set. Since the learning model establishment and calibration require far less time than design flows, we omit the model training and testing time. Our time metric mainly counts the cardinality of the dataset to train (or initialize) the machine learning-based surrogate model, and the final Pareto-optimal predictions for tool evaluation. Column "Multi-objective" lists three QoR metric spaces: area vs. delay, power vs. delay, area vs. power vs. delay, whilst columns "HV", "ADRS" and "Flows" are the evaluation metrics in terms of hypervolume error, the average distance from reference set and the number of design flow runs. Columns "ISLPED'17", "TCAD'19" and "Ours" correspond to the results searched by [116], [94] and our proposed sequential optimization-based DSE method with graph neural process as the surrogate. According to the results recorded in TABLE 5.1, our algorithm averagely surpasses [116] by 21.0% less hypervolume error and 31.8% smaller ADRS value, and reduces 19.9% less hypervolume error and 25.0% ADRS value comparing to [94]. Moreover, the time expense is still less than those of [94, 116]. In brief, our method performs better than the previous works [94, 116] with less hypervolume error, shorter average distance from reference set and fewer design flow calls. Here we just illustrate one golden Pareto-optimal design ("P1") which is not searched by [94, 116] but found by ours. The architecture and layout are visualized in Fig. 5.9.



(a) (b)

Figure 5.9 The golden Pareto-optimal solution ("P1") that is not found by previous works. (a) The architecture overview: Bit-width = 64, size = 234, Max. Level = 6, Max. fanout = 6. The associated QoR metric values are 1981.13μm^2 for the area, 6600μw for the power, and 0.334ns for the delay; (b) The corresponding layout snapshot.

5.5.3 Time Analyses

The "Flows" metric in TABLE 5.1 indicates the number of design flow runs, coarsely consists of two parts:

1. the number of evaluated adder designs for training;
2. the number of predicted Pareto-optimal adder designs to go through the EDA flow for evaluation.

Different from the pure SVR regressor-based work [116] which does not involve an incremental sampling process, the training set in the active learning-based DSE flow [94] includes an initial dataset and incrementally sampled dataset. To extract the Pareto-frontier, EDA tools are invoked to acquire the real QoR metric values of the predicted Pareto-optimal adder designs of the three methods. We draw a stacked bar plot, Fig. 5.10, to show the contribution of each part to total flow runs in three QoR metric spaces. In Fig. 5.10, "Evaluation on Initial Adders" refers to the number of adder designs for evaluation during initialization, and "Evaluation on Sampled Adders" indicates the number of incrementally sampled adder designs during iterations in [94] and our work (both are 4 samples per iteration),

while "Evaluation on Outputs" stands for the number of predicted Pareto-optimal designs by each methodology. As Fig. 5.10 suggests, the SVR regressor-based framework in [116] is trained with 500 adder designs when exploring Pareto frontier in area vs. delay or power vs. delay QoR metric space, and separately searches 98 and 115 Pareto-optimal adder designs for EDA flow evaluation in two metric spaces. The active learning-based work [94] initializes the Gaussian process with 400 adder designs in the area vs. delay and power vs. delay cases. After actively samples 20 (5 iterations) and 28 (7 iterations) adder designs, it ultimately seeks 14 and 54 adder designs for evaluation in two metric spaces, respectively. The given bar charts in Fig. 5.10 represent only 300 adder samples are utilized for initialization in our proposed flow in 2-dimensional QoR metric spaces, and both 48 (12 iterations) adder designs are incrementally selected to fine-tune the GNP model. Our flow predicts 65 and 74 Pareto-optimal samples to be evaluated by tools in area vs. delay and power vs. delay cases, respectively. For a more complicated task (i.e. optimization for area vs. power vs. delay QoR metrics), both the numbers of adders for training and EDA flow evaluation required by three methods surges upwards. [116] calibrates the SVR regressor via 1800 samples and finds 195 samples, and [94] trains the surrogate model, GP regressor, by 1300 initial adder designs and 40 (10 iterations) additional selected samples and invokes EDA flow 296 times to evaluate the predictions, while our framework only harnesses 1100 adders to initialize the GNP model and 56 (14 iterations) adders for follow-up tuning, and 314 adders are regarded as Pareto-optimal solutions. It is conspicuous that compared with the active learning-based DSE framework [94] and ours, [116] needs more adders for initialization due to a lack of sampling process to update the model. On the other hand, despite the similar sampling process for updating surrogate models, the proposed framework reduces approximately 4.8%, 12.4% and 10.1% time cost in area vs. delay, power vs. delay, and area vs. power vs. delay cases compared

to [94]. Besides, the proposed framework requires the least adder designs for initialization, which alleviates the pressure for starting the high-quality design searching.

For a better demonstration, illustrations in Fig. 5.11 depict the time (in minute) spent on statistical models in area vs. power vs. delay case. A glance at the pie charts reveals that the time (denoted as "Model" in Fig. 5.11) including model training time and prediction time is far less than the time spent on running EDA flows to generate golden QoR metric values of adder designs during initialization and outputs evaluation. As aforementioned, each EDA synthesis run takes about 5.5 minutes. The searching process totally costs [116] 10982.4 minutes among which 3.8 and 6.1 minutes are used for building the machine learning model and performing predictions. [94] spends about 9000.2 minutes to search for Pareto-optimal adder designs, and it takes only 1.7 and 0.5 minutes for training and inference. In our work with GPU acceleration, 7538.8 minutes are completely needed for design space exploration, where 1.26 and 2.53 minutes are consumed for training and testing. If we train and test the network only on CPU, the training and testing time climb to 8.5 and 9.4 minutes, respectively. Yet they still occupy extremely little proportion of whole runtime expenditure. It is worth mentioning that our framework directly extracts the features from adder designs without too much manual labour. On the contrary, both [116] and [94] feed 32-dimension, hand-crafted features to their machine learning model. The proposed GNP processes adjusted adjacency matrixes of the size up to 448×448 (64 bit and 7 logic levels). Due to the different feature extraction processes and structures of features, our work behaves not as well as [94] when only considering model time. But the time cost on the model is still acceptable regarding the whole DSE process. Nevertheless, via the GPU acceleration, it outperforms [116] on model time.

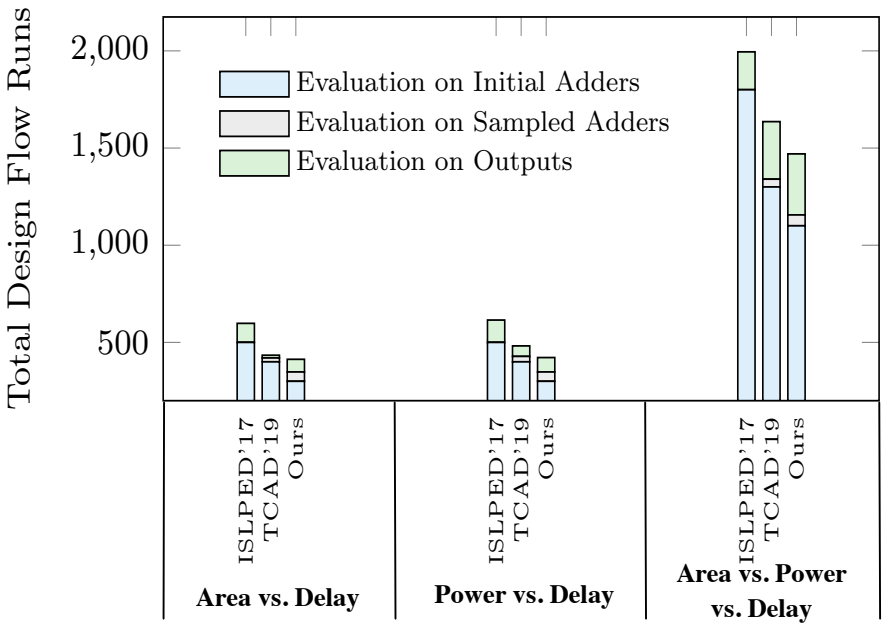


Figure 5.10 The contributions of sub-processes of DSE works to total flow runs.

5.5.4 The comparisons against classical adders

TABLE 5.2 records the comparison between adder designs searched by aforementioned DSE methods against Design-Ware adders (best delay), an adder design generated by a highly sophisticated adder synthesis algorithm [115] as well as some classical adders like Kogge-Stone and Sklansky. These legacy adders are manually designed, which involve much human assiduous and masses of trials. When it turns to [115], a polynomial-time algorithm has been developed to generate prefix graph structures. The main idea behind [115] is to generate a single prefix graph network for a set of structural constraints, such as the logic level, fan-out, etc. Theoretically, the legacy adders and the traditional adder synthesis algorithm in [115] are not capable of handling the exploring task in a huge adder design space. To fairly compare with classical adders and conventional synthesis method [115], we select a slice of Pareto-optimal designs predicted by previous adder DSE methods [94, 116] and ours. For example, "ISLPED'17-P1" means one predicted Pareto-optimal de-

Figure 5.11 The runtime breakdown of previous arts and our framework.

sign by [116], and "ISLPED'17-P2" stands for another solution in the explored Pareto-optimal set. It can be observed the designs searched by DSE methods behave better in all three targets (delay, area and energy) than Design-Ware adders, Kogge-Stone adders, and the solution offered by [115]. Our solutions ("P1", "P2" and "P3") dominate the corresponding solutions explored by [94, 116]. Notice that "P1" just happens to be the same solution point in the delay vs. area and delay vs. power Pareto-optimal curves, while other points ("P2" and "P3") are not outlined in Fig. 6.6 since they are just found in three QoR metric space case (area vs. power vs. delay). TABLE 5.2 implies that DSE methods are more effective than conventional adder solution providers, and more importantly, compared with the existing adder DSE methodologies, the proposed methodology can discover better adder designs.

In a nutshell, the proposed DSE method surrogated by graph

Table 5.2 Comparison with other approaches for 64 bit adder.

Method	Delay (ps)	Area (μm^2)	Energy (fJ/op)
DesignWare	346.5	2531.3	8160
Kogge-Stone	347.9	2563.7	8780
ISLPED'17-P1 [116]	344.1	2101.9	6930
TCAD'19-P1 [94]	339.0	2180.8	6930
Ours-P1	**334.4**	**1981.12**	**6600**
Sklansky	356.1	1792.5	6100
ISLPED'17-P2 [116]	353.5	1753.3	5980
TCAD'19-P2 [94]	353.0	1753.0	5900
Ours-P2	**353.0**	**1750.1**	**5830**
ASPDAC'15 [115]	348.7	1971.4	6980
ISLPED'17-P3 [116]	348.2	1969.8	6450
TCAD'19-P3 [94]	343.0	1912.6	6390
Ours-P3	**342.7**	**1769.6**	**5970**

neural process behaves better than previous arts. Additionally, our DSE framework has the potential to be generalized to different bit-width adder designs in theory. We utilize the 32-bit adder design as an exemplar for the following descriptions. There is no limitation on the size of data input of our DSE model. Besides, in terms of the graph structure and impact of tool settings, 32-bit adder designs are analogous to 64-bit adder designs to some extent. By harnessing certain transfer learning techniques [77,110,141] to achieve the domain adaptation of different bit-width adder designs, we can fine-tune a pre-trained model (on 64-bit adder dataset) on a small amount of 32-bit adder designs.

5.6 Summary

In this paper, for the first time, we have proposed a new end-to-end learning model, graph neural process, where a graph autoencoder for prefix adder structures and a neural process are simultaneously performed. The graph neural process provides a new solution to

automatically extracting features from prefix adder structures. Besides, we have proposed a sequential optimization model-based DSE methodology with the graph neural process as its surrogate model to guide design space exploration for power-efficient, high-speed prefix adders. Our methodology is almost automatic from feature extraction to high-quality adder design space exploration. The experimental results have demonstrated the superiority of the proposed framework over the prior arts. With the VLSI designs becoming increasingly complicated, we expect to generalize our idea to settle more VLSI design space exploration problems (e.g. multiplier DSE problem, adder DSE issues among different bit width and technology nodes.).

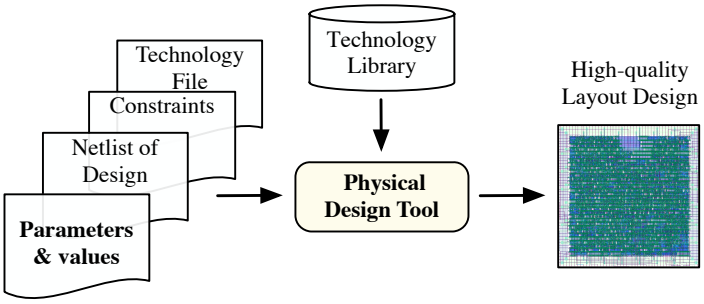
□ **End of chapter.**

Chapter 6

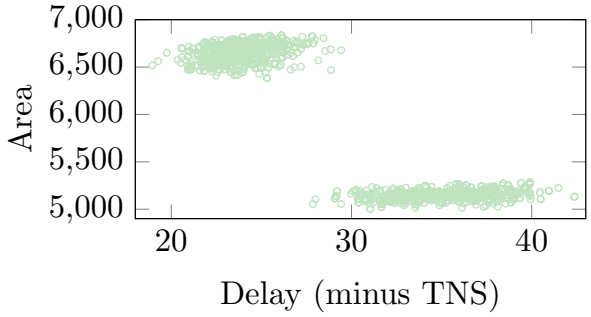
Parameter Tuning of Physical Design Tool

6.1 Introduction and Motivation

Given a design (e.g. a multiply accumulate (MAC) design under the advanced 7nm technology node in our scenario) and other PDK files (seen in Fig. 6.1(a)), how to automatically and efficiently acquire high-quality (near Pareto-optimal) parameter configurations of the physical design tool is a critical issue. We try to visualize the challenges existing in parameter tuning in Fig. 6.1(b). The first problem is that the final performance after physical synthesis could vary significantly under delicate change of sensitive parameters. We can see in Fig. 6.1(b), the design quality variance in QoR metric space with changing 7 physical synthesis tool parameters based on human intervention. Especially, manually tuning the parameter noted as "maxDensity" leads to the two clusters in QoR metric space. Even worse, QoR metrics are realistically correlated, which affects the tuning process. In Fig. 6.1(b), most of the dots in QoR metrics show either a unsatisfied area quality with low delay or a small area value with a high delay. Additionally, the parameter design space is huge (thousands of dots in the QoR metric space in Fig. 6.1(b)) and will expand exponentially with more constraints or parameters are taken



(a) A physical design tool



(b) The QoR metric space

Figure 6.1 The visualizations of the working flow of a physical design tool and a QoR metric space (area vs. minus total negative slack) generated by invoking physical design tool 1440 times in a manual parameter tuning manner. In the QoR metric space, each light green dot represents an output QoR metric value tuple of a physical design tool, which is associated with a specific parameter configuration. The units for area and delay are ns and um^2, respectively.

into account, which makes an efficient flow with as few evaluations as possible in high demand. To address aforementioned issues, we develop a multi-task Gaussian process-based multi-objective Bayesian optimization flow for efficient physical design tool parameter tuning. Multi-task Gaussian process regressor is leveraged as the surrogate model in the proposed Bayesian optimization framework, which captures the correlations among the tool parameters and QoR metrics. Compatible with the multi-objective optimization problem, the information gain-based acquisition function is developed to guide the exploration process to find the superior parameter settings. Our main contributions are summarized as follows:

- A multi-task Gaussian process is built to learn inter-objective dependencies.
- A multi-objective Bayesian optimization framework with a multi-task Gaussian process as its surrogate model is investigated to attempt to tune physical design tool parameters.
- The proposed optimization framework finds better Pareto frontiers on two benchmarks under the advanced 7nm technology node with less expense on physical design tool runs.

The rest of the chapter is organized as follows. Section 6.2 introduces some prior knowledge about multi-objective optimization, and then gives the problem formulation. Section 6.3 describes the proposed multi-objective Bayesian optimization framework, while Section 6.4 discusses the developed parameter tuning flow. Section 6.5 presents the experimental results, followed by summary in Section 6.6.

6.2 Preliminaries

In this section, the background of the multi-objective optimization is depicted, and then with descriptions of two evaluation metrics, we give the problem formulation.

6.2.1 Multi-objective Optimization

Assume a multi-objective optimization problem has a set of feasible solutions $\mathcal{X} \in \mathbb{R}^P$. There are N objective functions, $f_1(\cdot), \ldots, f_N(\cdot)$, which map an input $\boldsymbol{x}$ to corresponding results $f_1(\boldsymbol{x}), \ldots, f_N(\boldsymbol{x})$ forming an N-dimensional result vector $\boldsymbol{f}(\boldsymbol{x})$. A result vector $\boldsymbol{f}(\boldsymbol{u})$ is said to dominate another result vector $\boldsymbol{f}(\boldsymbol{v})$ if $\boldsymbol{f}(\boldsymbol{u})$ is at least as good as $\boldsymbol{f}(\boldsymbol{v})$ in all the objectives, namely, $f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}), \forall i \in [1, N]$ if all the objectives are to be minimized. Hence, we say that a solution $\boldsymbol{x}$ is Pareto-optimal if it is not dominated by other solutions in

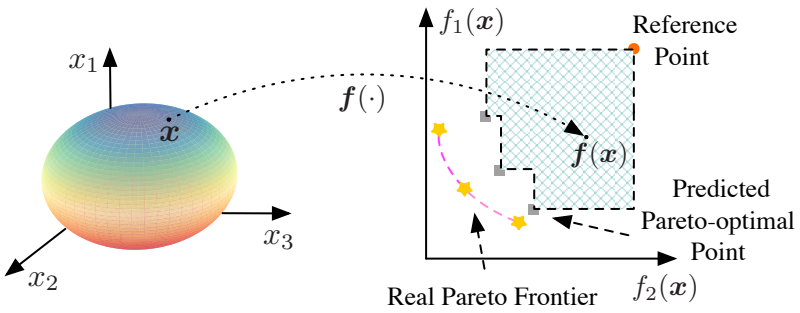


Figure 6.2 An example of a bi-objective (two correlated QoR metrics) optimization problem with a tridimensional parameter configuration space. The left sub-figure is the tridimensional parameter space, whilst the right sub-figure displays the associated bi-objective space. The middle curve connecting the two parts refers to a parameter-to-QoR mapping $\boldsymbol{f}(\cdot)$. Suppose that the parameter configuration features are the physical design parameters like frequency, max fanout and max density, while QoR metrics are area and delay. Each data point in the parameter space is a simply one-hot encoded, normalized and concatenated feature vector of above three design parameters. The Pareto frontier, golden star with dash carnation lines in right sketch, features delay and area.

the feasible solution set $\mathcal{X}$. A sketch of multi-objective optimization is exemplified in Fig. 6.2.

In our context of physical design tool parameter tuning, a feasible solution vector $\boldsymbol{x}$ is the feature representation of one-hot encoded, normalized and concatenated physical tool parameters which satisfy the pre-determined constraints, while a Pareto-optimal design is where none of the QoR metrics like area, power and delay, can be optimized without worsening at least one of the rest. The Pareto set contains all the Pareto-optimal solutions.

6.2.2 Problem Formulation

Definition 8 (Hypervolume)**.** *This metric reflects the volume fenced by the Pareto frontier and a reference point in the objective space. It measures how well distributed the points are on the Pareto frontier approximation.*

In the right subfigure of Fig. 6.2, the area filled with grids is an example of the hypervolume of a predicted Pareto-optimal set in a bi-objective space. The hypervolume error for a Pareto set approximation $\hat{\mathcal{P}}$ is defined:

$$e = \frac{H(\mathcal{P}) - H(\hat{\mathcal{P}})}{H(\mathcal{P})}, \tag{6.1}$$

where $\mathcal{P}$ is the golden Pareto-optimal set, and $H(\mathcal{P})$ is the ground-truth of hypervolume. If a solution set $\mathcal{P}'$ is better than another set $\mathcal{P}''$, $H(\mathcal{P}')$ is greater than $H(\mathcal{P}'')$ in our scenario.

Definition 9 (Average Distance from Reference Set (ADRS))**.** *Given a reference Pareto-optimal set* $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots | \boldsymbol{a} = ({m_1}^a, {m_2}^a, \ldots, {m_N}^a)\}$ *and an approximated Pareto-optimal set* $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots | \boldsymbol{p} = ({m_1}^p, {m_2}^p, \ldots, {m_N}^p)\}$ *in N-objective optimization problem:*

$$\mathrm{ADRS}(\mathcal{A}, \mathcal{P}) = \frac{1}{(|\mathcal{A}|)} \sum_{\boldsymbol{a} \in \mathcal{A}} \min_{\boldsymbol{p} \in \mathcal{P}} \delta(\boldsymbol{a}, \boldsymbol{p}), \tag{6.2}$$

where $\delta(\boldsymbol{a}, \boldsymbol{p}) = \max\left\{\left|\frac{{m_1}^p - {m_1}^a}{{m_1}^a}\right|, \ldots, \left|\frac{{m_N}^p - {m_N}^a}{{m_N}^a}\right|\right\}$.

ADRS is used to quantify how close a set of non-dominated points is from the Pareto frontier in the objective space. The smaller ADRS value is, the closer the approximate set $\mathcal{P}$ is to the reference set $\mathcal{A}$.

With aforementioned knowledge, our problem can be formulated as follows.

Problem 4 (Automatic Parameter Tuning for Physical Design Tool)**.** *Given the boundary of parameters of a physical design tool, the objective of physical design tool parameter tuning is to automatically search the Pareto-optimal parameter configurations which bring about the high design quality concerning multiple QoR metrics like delay versus power/area and delay versus power versus area.*

6.3 The Multi-Objective Bayesian Optimization Method

As aforementioned, the ultimate goal of physical design parameter tuning is to simultaneously achieve timing closure and a smallest reachable area and lowest acceptable power. Quite often the QoR metrics are coupled or conflicted, which is not ideal for traditional tuning flows. For example, reducing the supply voltage can effectively cut down the dynamic power consumption. However, it leads to an increase in the gate delays. To tackle such negatively correlated issue, we propose a tuning flow which is based on a multi-task Gaussian process surrogated multi-objective Bayesian optimization method. The optimization method explores the correlations among QoR metrics and attempts to explores the best trade-offs among them. The two main components of the proposed optimization method, i.e., multi-task Gaussian process surrogate model and information gain-based acquisition function, are well-customized for our case. In the following, we will first introduce the two vital keys and then the whole optimization method.

6.3.1 The surrogate model: multi-task Gaussian process

Multi-task Gaussian process (MGP) models are prevalently harnessed to couple related objectives or functions for a joint regression [16, 140]. This coupling is achieved by designing a structured covariance function, yielding a prior on objectives to be regressed. More importantly, MGP tries to learn a kernel which involves inter-task dependencies based solely on the task identities and the observed data for each task. This kind of property is naturally compatible with our optimization methodology. We can utilize MGP as the surrogate model to jointly predict the multiple QoR metric (e.g. power-performance-area, PPA) values with respect to tool parameters as inputs. Note that the "task" means regression on one QoR metric value in our case.

In the same way as single-task GPs, multi-task GPs are mainly specified by their kernels which are usually assumed to be zero mean functions. For developing valid covariance functions, we adopt a linear model of coregionalization (LMC) where the outputs are expressed as linear combinations of independent random processes. As aforementioned, MGP attempts to build a multi-output function $\boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^p \rightarrow \mathbb{R}^N$ with a feature vector $\boldsymbol{x}$ as input. Note that in our work, by one-hot encoding and normalization tricks, the physical design tool parameters are simply encoded and concatenated as $\boldsymbol{x}$. In the LMC assumption, the d-th element of output vector (in our case, the output vector contains estimated QoR metric values such as area, delay, power given the input tool parameter configuration $\boldsymbol{x}$), i.e. $f_d(\boldsymbol{x})$, can be represented as

$$f_d(\boldsymbol{x}) = \sum_{q=1}^{Q} a_{d,q} g_q(\boldsymbol{x}), \tag{6.3}$$

where each latent function $g_q(\boldsymbol{x})$ is considered to follow an independent Gaussian prior with zero mean and covariance $\operatorname{cov}\left[g_q(\boldsymbol{x}), g_{q'}\left(\boldsymbol{x}'\right)\right] = k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ if $q = q'$, and $a_{d,q}$ is a scalar coefficient. Specifically, when we set Q equal to 1, the Equation (6.3) is degraded to $f_d(\boldsymbol{x}) = g_d(\boldsymbol{x})$ with a little abuse of notations. For a better understanding, a visualization for MGP is exemplified in Fig. 6.3.

Obviously, the process $\{g_q(\boldsymbol{x})\}_{q=1}^{Q}$ are independent if $q \neq q'$. We can derive the cross covariance between any two functions $f_d(\boldsymbol{x})$ and $f_{d'}(\boldsymbol{x})$ like:

$$\begin{aligned} \operatorname{cov}\left[f_d(\boldsymbol{x}), f_{d'}\left(\boldsymbol{x}'\right)\right] &= \sum_{q=1}^{Q} a_{d,q} a_{d',q} k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right) \\ &= \sum_{q=1}^{Q} b_{d,d'}^{q} k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right). \end{aligned} \tag{6.4}$$

$\left(\boldsymbol{K}\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right)_{d,d'}$ is exploited to denote $\operatorname{cov}\left[f_d(\boldsymbol{x}), f_{d'}\left(\boldsymbol{x}'\right)\right]$ which indicates the similarity or covariance across d- and d'-th task at $\boldsymbol{x}$ and $\boldsymbol{x}'$ respectively. According to the expression in Equation (6.4), the

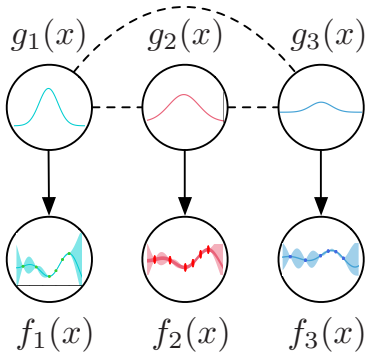


Figure 6.3 The visualization for a tri-task Gaussian process model with a scalar input and $Q = 1$. The dash lines indicate the correlations. We can image this model is built for an area vs. power vs. delay joint regression problem with a parameter, `max_transition`, as input x.

kernel for multi-task Gaussian process is shown as follows.

$$\boldsymbol{K}\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \sum_{q=1}^{Q} \boldsymbol{B}_q(d, d') k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right), \tag{6.5}$$

where $\boldsymbol{B}_q \in \mathbb{R}^{D \times D}$ is a so-called coregionalization matrix, and the coefficient $b^q_{d,d'}$ is the element of $\boldsymbol{B}_q$. $\boldsymbol{B}_q$ is usually set to be positive semi-definite matrix which specifies the inter-task similarities. Above linear expression of outputs represents the covariance function as the sum of the products of two covariance functions. More specifically, $\boldsymbol{B}_q$ models the dependence between the outputs, which is independent of the input parameter vector $\boldsymbol{x}$, while $k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right)$ discovers the parameter dependence, independently of the ultimate output QoR metrics $\boldsymbol{f}(\boldsymbol{x})$.

For further reduction, a reasonable assumption that $b^q_{d,d'} = k^g_{d,d'} b_q$ for a suitable scalar coefficient $k^g_{d,d'}$ can be made. With substituting this assumption for $b^q_{d,d'}$, Equation (6.4) can be re-written as Equation (6.6):

$$\operatorname{cov}\left[f_d(\boldsymbol{x}), f_{d'}\left(\boldsymbol{x}'\right)\right] = k^g_{d,d'} k^x\left(\boldsymbol{x}, \boldsymbol{x}'\right), \tag{6.6}$$

where $k^x\left(\boldsymbol{x}, \boldsymbol{x}'\right) = \sum_{q=1}^{Q} b_q k_q\left(\boldsymbol{x}, \boldsymbol{x}'\right)$. Finally, on the back of independent GP priors over the latent functions $g_q(\boldsymbol{x})$, our kernel matrix

corresponding to a dataset $\boldsymbol{X}$ (stacking the tool parameter configurations as rows) takes the form in Equation (6.7).

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) &= \boldsymbol{K}^g \otimes \boldsymbol{K}^x(\boldsymbol{X},\boldsymbol{X}), \\ f_d &\sim \mathcal{N}\left(\sum_{q=1}^{Q} a_{d,q} g_q(\boldsymbol{x}), \sigma_d^2\right), \end{aligned} \tag{6.7}$$

where $\otimes$ refers to Kronecker product. Equation (6.7) decouples the correlation between task (estimating the QoR metric values) similarities and input (tool parameter configurations) similarities.

In the same spirit of standard GP for the mean and variance predictions, inference in the MGP model can be calculated. Given M training points (tool parameter configurations) with concatenated golden QoR metric values $\boldsymbol{y} \in \mathbb{R}^{MN}$ and a new parameter configuration $\boldsymbol{x}_*$, The closed-form expressions of mean and uncertainty for task d at $\boldsymbol{x}_*$ shown in following Equations (6.9) and (6.10) are acquired by using the first-order optimality condition on the marginal likelihood function (see below Equation (6.8)) of the multi-task Gaussian process.

$$\begin{aligned} L = &-\frac{M}{2}\log|\boldsymbol{K}^g| - \frac{N}{2}\log|\boldsymbol{K}^x| \\ &- \frac{1}{2}\operatorname{tr}\left[(\boldsymbol{K}^g)^{-1}\boldsymbol{F}^\top(\boldsymbol{K}^x)^{-1}\boldsymbol{F}\right] - \frac{M}{2}\sum_{l-1}^{N}\log\sigma_l^2 \\ &- \frac{1}{2}\operatorname{tr}\left[(\boldsymbol{Y}-\boldsymbol{F})\boldsymbol{D}^{-1}(\boldsymbol{Y}-\boldsymbol{F})^\top\right] - \frac{MN}{2}\log 2\pi, \end{aligned} \tag{6.8}$$

$$\bar{\boldsymbol{f}}_d(\boldsymbol{x}_*) = (\boldsymbol{k}_d^g \otimes \boldsymbol{k}_*^x)^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{y}, \tag{6.9}$$

$$\begin{aligned} \bar{\text{var}}\left[\bar{\boldsymbol{f}}_d(\boldsymbol{x}_*)\right] = &\, k_{d,d}^g k^x(\boldsymbol{x}_*,\boldsymbol{x}_*) \\ &- (\boldsymbol{k}_d^g \otimes \boldsymbol{k}_*^x)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{k}_d^g \otimes \boldsymbol{k}_*^x), \end{aligned} \tag{6.10}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{MN\times MN} := (\boldsymbol{K}^g \otimes \boldsymbol{K}^x + \boldsymbol{D} \otimes \boldsymbol{I})$ with $\boldsymbol{D}$ an $N \times N$ diagonal matrix in which the element in (d,d) is σ_d^2. $\boldsymbol{K}^x$ indicates the matrix of covariances between all pairs of training points. $\boldsymbol{K}^g$ which is treated as a kind of hyperparameter of the MGP describes the task similarities, and it can be obtained by maximizing likelihood

estimation. Further, $\boldsymbol{k}_d^g$ denotes the d-th column of $\boldsymbol{K}^g$, and $\boldsymbol{k}_*^x$ refers to the vector of covariances between the test point $\boldsymbol{x}_*$ and the other training points. Until now, we have built the surrogate model and prepared the predictions on QoR metrics like area, delay and power for further calculations of the acquisition function.

6.3.2 The information gain-based acquisition function

Given the parameter configuration space $\mathcal{X}$ (i.e. the boundaries of parameters) of a physical design tool and current training dataset $\mathcal{D}$, we attempt to maximize the information gain about the predicted Pareto frontier $\mathcal{Y}^*$, namely expected reduction in entropy over $\mathcal{Y}^*$. Consequently, an information gain-based acquisition function $\mathrm{I}(\boldsymbol{x})$ is emerged in Equation (6.11).

$$\begin{aligned}\mathrm{I}(\boldsymbol{x}) &= \mathrm{I}\left(\{\boldsymbol{x}, \boldsymbol{y}\}, \mathcal{Y}^* \mid \mathcal{D}\right) \\ &= \mathrm{H}\left(\mathcal{Y}^* \mid \mathcal{D}\right) - \mathbb{E}_{\boldsymbol{y}}\left[\mathrm{H}\left(\mathcal{Y}^* \mid \mathcal{D} \cup \{\boldsymbol{x}, \boldsymbol{y}\}\right)\right], \end{aligned} \tag{6.11}$$

where $\boldsymbol{x}$ is the parameter configuration to be selected and $\boldsymbol{y}$ is the corresponding predicted the QoR values. By using the symmetric property of mutual information, we can rewrite Equation (6.11) as Equation (6.12).

$$\mathrm{I}(\boldsymbol{x}) = \mathrm{H}\left(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}\right) - \mathbb{E}_{\mathcal{Y}^*}\left[\mathrm{H}\left(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}, \mathcal{Y}^*\right)\right]. \tag{6.12}$$

Equation (6.12) tries to minimize the uncertainty estimation of Pareto frontier approximation $\mathcal{Y}^*$ after searching the next candidate $\boldsymbol{x}$ for design tool evaluation. The first term of Equation (6.12) is straightforward to compute. In fact, it is simply the entropy of the predictive distribution $\mathrm{p}\left(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}\right)$ which is a N-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$. Applying the substitution trick in a normalization fashion, we can obtain a simplified expression of the first term as Equation (6.13).

$$\mathrm{H}\left(\boldsymbol{y} \mid \mathcal{D}, \boldsymbol{x}\right) = \frac{N}{2}(\ln 2\pi + 1) + \frac{1}{2}\ln|\boldsymbol{\Sigma}|. \tag{6.13}$$

The main part of second term in Equation (6.12) is the entropy of the predictive distribution conditioned on the Pareto frontier $\mathcal{Y}^*$. This expectation term can be approximated fast and effectively by:

$$\mathbb{E}_{\mathcal{Y}*}\left[H\left(\boldsymbol{y} \mid D, \boldsymbol{x}, \mathcal{Y}^*\right)\right] \simeq H\left(\boldsymbol{y} \mid D, \boldsymbol{x}, \mathcal{Y}^*\right). \tag{6.14}$$

In Equation (6.14), we can observe that the computation involves the predicted Pareto frontier $\mathcal{Y}^*$, which means the Pareto frontier needs to be estimated. To calculate the Pareto frontier samples, a multi-objective optimization formulation should be first established. Analogous to the previous arts [66, 67], sample functions from the posterior MGP model via some kernel functions and then solve a multi-objective optimization over the N sampled functions. This multi-objective optimization also allows us to capture the interactions between different objectives. In the process, the N kernel-based linear functions are exploited in a ridge regression taste. The principle behind this execution is that the Gaussian Process is a Bayesian generalization of the ridge regression and can be explained in a weight space view [146]. The sample function is constructed as a finitely parametrized approximation which is show in Equation (6.15).

$$\tilde{f}_i\left(\boldsymbol{x}\right) = \boldsymbol{\kappa}(\boldsymbol{x})^\top \boldsymbol{\mu}, \tag{6.15}$$

where $\boldsymbol{\kappa}(\cdot)$ is some kind of kernel function such as Matern, radial basis function, and $\boldsymbol{\mu}$ is a random variable sampled from its corresponding posterior distribution conditioned on the data $\mathcal{D}$ including all parameter configurations through tool evaluations. The re-parameterization trick is exploited to compute $\boldsymbol{\mu}$.

After formulating the multi-objective optimization over the N sampled functions $\tilde{f}_1\left(\cdot\right), \tilde{f}_2\left(\cdot\right), \ldots, \tilde{f}_N\left(\cdot\right)$, a genetic algorithm-based solver, e.g. NSGA-II, is adopted to optimize the optimization problem.

$$\mathcal{X}^* = \arg\min_{\boldsymbol{x}\in\mathcal{X}} \left(\tilde{f}_1\left(\boldsymbol{x}\right), \tilde{f}_2\left(\boldsymbol{x}\right), \ldots, \tilde{f}_N\left(\boldsymbol{x}\right)\right), \tag{6.16}$$

Algorithm 3 The Computation Process for Information Gain based Acquisition Function in t-th iteration

Input: the number of tasks N, the predictions from the multi-task GP model.

1: **for** $i \leftarrow 1$ to N **do**
2: Sample approximation function $\tilde{f}_i(\cdot)$ from each task output from the multi-task GP model; ▷ Equation (6.15)
3: **end for**
4: Multi-objective optimizer (NSGA-II [38]) optimizing over $\tilde{f}_1(\cdot), \tilde{f}_2(\cdot), \ldots, \tilde{f}_N(\cdot)$; ▷ Equation (6.16)
5: Compute and maximize $\mathrm{I}(\boldsymbol{x})$; ▷ Equation (6.17)

where $\mathcal{X}^*$ is the associated Pareto-optimal set.

Until now, the Pareto-optimal set $\mathcal{X}_t^*$ with corresponding objective values $\mathcal{Y}*$ is approximated. Next, an additional constraint is introduced. It can be proved that the value of each element of $\boldsymbol{y}$ in Equation (6.14) is upper-bounded by the maximum value of corresponding element in sampled point on Pareto frontier $\mathcal{Y}^*$. Combining the boundedness property and the fact that each sampled objective function is modeled as a GP prior, we can model each component of $\boldsymbol{y}$ as a truncated Gaussian distribution. Ultimately, directly harnessing the closed-form solutions to moments of a truncated Gaussian distribution [19] and Equation (6.13), we can get the approximation of acquisition function as shown in Equation (6.17).

$$\mathrm{I}(\boldsymbol{x}) = \sum_{i=1}^{N} \left[\frac{\nu_i(\boldsymbol{x}) \phi\left(\nu_i(\boldsymbol{x})\right)}{2\mathrm{F}\left(\nu_i(\boldsymbol{x})\right)} - \ln \mathrm{F}\left(\nu_i(\boldsymbol{x})\right) \right], \tag{6.17}$$

where ϕ and F stand for the probability density function and the cumulative density function of a standard Gaussian distribution respectively. $\nu_i(\boldsymbol{x})$ equals to $\frac{y_i^* - \mu_i(\boldsymbol{x})}{\sigma_i(\boldsymbol{x})}$ with y_i^* the maximum value among the sampled points on predicted Pareto frontier for i-th QoR metric. Then we can maximize Equation (6.17) via some Quasi-Newton optimizers like limited-memory BFGS. The computation process for information gain-based acquisition function is briefly concluded in Algorithm 3.

6.3.3 The Multi-Objective Bayesian Optimization Method

Algorithm 4 Multi-objective Bayesian Optimization Method

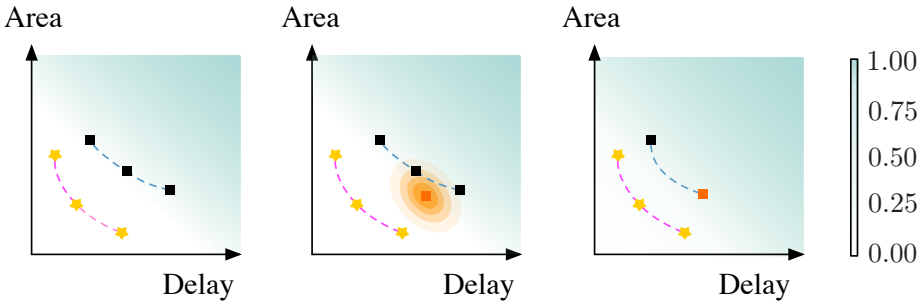
Input: the parameter configuration space $\mathcal{X}$ (i.e. the boundaries of parameters) of a certain physical design tool, initial dataset $\mathcal{D}_0$ and the number of maximum iterations T.

Output: the set of predicted Pareto-optimal parameter configurations $\tilde{\mathcal{P}}$ over $\mathcal{D}$.

1: **Initialization**: Initial the multi-task GP model with initial data set $\mathcal{D}_0$;
2: **while** not convergence or $t < T$ **do**
3: $\boldsymbol{x}_t \leftarrow \arg\min_{\boldsymbol{x}\in\mathcal{X}} \mathrm{I}(\boldsymbol{x})$; ▷ Algorithm 3
4: Enquiry the physical design tool to acquire the golden values $\boldsymbol{y}_t$ (area/power/delay values) of $\boldsymbol{x}_t$;
5: Update the training dataset $\mathcal{D}_{t-1}$ with $(\boldsymbol{x}_t, \boldsymbol{y}_t)$: $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (\boldsymbol{x}_t, \boldsymbol{y}_t)$;
6: Fine tune the multi-task GP model with $\mathcal{D}_t$;
7: $t \leftarrow t + 1$;
8: **end while**

After details of two main components, we summarize the proposed multi-objective Bayesian optimization method in Algorithm 4. In initialization stage of Algorithm 4, the surrogate model captures the priors about the unknown objective functions, and offers predictions on posterior distributions (line 1). Per iteration, the acquisition function first exploits the predicted posterior distributions to search for the candidate points for tool query (lines 3 and 4). After evaluation, the candidate points with golden values will in turn help calibrate the multi-task Gaussian process model (lines 5 and 6). The optimization process performs in this manner iteratively until converges. Finally, the Pareto-optimal parameter configurations are obtained.

For a better comprehend, we sketch some visualizations in Fig. 6.4 to illustrate the working principles for the proposed Bayesian optimization method.



(a) (b) (c)

Figure 6.4 The visualization for the proposed Bayesian optimization method optimizing multiple QoR metrics of interest (i.e. area vs. delay). The golden stars with dash carnation lines refer to the golden Pareto frontier among area and delay. The black squares with blue dash lines denote the current predicted Pareto frontier based on observed tool parameter configurations and corresponding performance values. The cyan heat map reflects the probability of points in the bi-objective space dominated by others. With a darker color, the probability gets higher. (a) New parameter configuration for tool evaluation has not been selected yet; (b) The method chooses a new parameter configuration with associated area vs. delay values near one predicted Pareto frontier point. The nested circles reflect the distribution of the output function at new data point; (c) The predicted Pareto frontier is updated with the new data point and the dark-colored area in the heat map expands, which means that the uncertainty about the Pareto frontier approximation decreases.

6.4 The Developed Physical Design Tool Parameter Tuning Flow

We visualize the global view of the proposed tuning flow in Fig. 6.5, where the arrows mark the dataflow. Our proposed Bayesian optimization framework delivers the selected parameter configurations for tool evaluation. The tool estimates the corresponding QoR metric values given selected parameter configurations with certain design netlist, technology files and associate libraries. Then, the training dataset stores the new data point and promptly calibrates multi-task GP model. With more accurate predictions from multi-task GP model, acquisition function continues seeking the next parameter

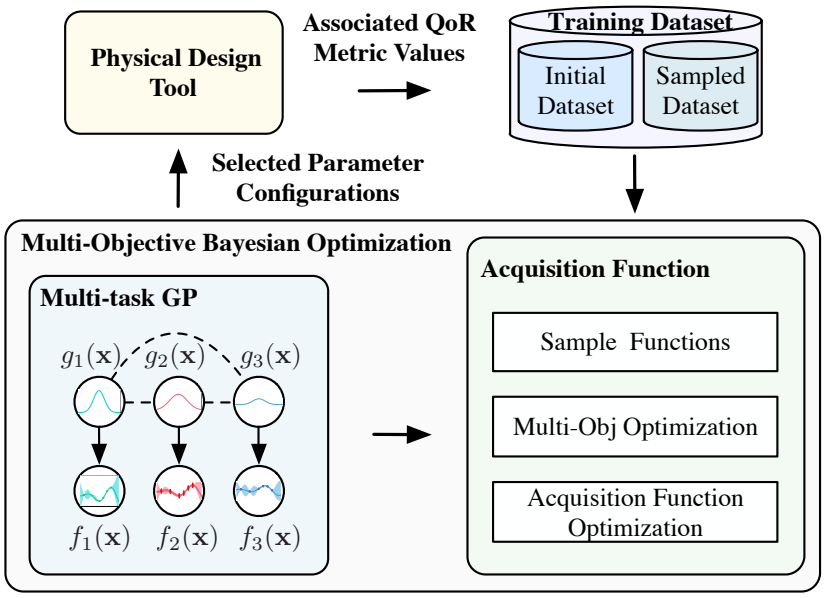


Figure 6.5 The workflow of proposed tuning flow.

configuration candidate.

6.5 Experimental Results

The implementation of our framework is in `Python` with `GPflow` [140] library. We test it on a platform with an Xeon Silver 4114 CPU processor. To evaluate the performance state of our framework, we compare it with the state-of-the-art methods [82, 94, 95, 116, 148] by exploring the parameter space of one certain physical design tool on two industrial benchmarks. The two benchmarks consist of hundreds of different input parameter configurations of the physical design tool with associated QoR metrics values. These golden values are obtained by invoking the physical design tool fed with industrial MAC designs which have far more than twenty thousand of cells under 7nm technology node. Besides, these two benchmarks represent parameter spaces which are built upon the distinct combinations of 12 physical design tool parameters with different parameter bounds and settings, and these parameter configurations in two benchmarks

Table 6.1 The statistics of parameters of a certain physical design tool on two benchmarks. Some descriptions of parameters are introduced as supplementary. For example, flowEffort configures the flow to give a trade-off between best-quality result and best turnaround time, and place_global_uniform_density enables even cell distribution for designs with less than 70% utilization, and place_global_cong_effort specifies the effort level of relieving congestion, and place_global_max_density controls the maximum density of local bins during global placement, while maxLength belongs to DRV rule parameters including max_capacitance/ max_transition/ max_fanout, and maxDensity defines the maximum value for density (area utilization).

Parameters	Benchmark1		Benchmark2	
	Min	Max	Min	Max
freq	950	1050	1000	1300
set_clock_uncertainty	50	200	20	100
flowEffort	standard	extreme	standard	extreme
place_global_uniform_density	FALSE	TRUE	FALSE	TRUE
place_global_cong_effort	AUTO	HIGH	AUTO	HIGH
place_global_max_density	0.65	0.90	0.65	0.90
maxLength	160	310	160	300
maxDensity	0.65	0.90	0.65	0.90
max_transition	0.19	0.34	0.10	0.35
max_capacitance	0.08	0.13	0.08	0.20
max_fanout	25	50	25	50
maxAllowedDelay	0.00	0.25	0.00	0.25

are sampled via Latin hypercube sampling in parameter spaces. In the view of the fact that these parameters set constraints on the area, power and delay during the physical design flow, they impact profoundly on the QoR metric qualities of designs. Thus, these benchmarks can be exploited to verify the performance of prior arts and ours. For a better understanding, we summarize the statistics of parameter in design space in TABLE 6.1. The data types can be either floating or integer, which depend on the specific parameters.

For a comprehensive comparison, the experiments are performed qualitatively and quantitatively. First, we estimate the performance

of these methods in terms of the hypervolume error, the ADRS and the number of tool runs, and then visually compare the quality of the Pareto frontiers found by the those methodologies.

TABLE 6.2 and TABLE 6.3 depict the qualities of Pareto frontiers in quantification. Note that per tool run, it costs about 3 hours, while the initialization and follow-up updating of all methods require far less time than simulations. On the back of that, we use the simulation runs to calculate the optimization cost in lieu of the conventional running time for model training and testing. Additionally, to compare methods fairly, the cardinality of the dataset for initialization is the same. Column "Multi-objective" lists three objective space to be explored: area vs. delay, power vs. delay, area vs. power vs. delay, whilst columns "HV", "ADRS" and "Simu." are the evaluation metrics referring to hypervolume error, the average distance from reference set and the number of tool runs. Columns "ISLPED'17", "TCAD'19", "MLCAD'19", "DAC'19", "ASPDAC'20" and "Ours" represent the results acquired by simple regression method (i.e. support vector machine) [116], Pareto-driven active learning-based optimization framework [94], single-objective Bayesian optimization framework [95] with scalarization trick extending to multi-objective case, a tensor decomposition and recommender system-based tuning framework [82], an active learning-based tuning approach leveraging the feature importance sampling and XGBoost regressor [148], and our proposed multi-task Gaussian process-based multi-objective Bayesian optimization framework, respectively. For `Benchmark1`, it can be seen that, with less optimization expense, our algorithm evenly outperforms [116] with a 46.2% decrease on hypervolume error and a 51.0% drop on ADRS value, and contracts 45.0% hypervolume error and 63.6% ADRS value compared with [94]. On the other hand, our method is superior to [95] on this benchmark with a drop of 33.1% hypervolume error and almost half ADRS value, and reduces beyond 40.0% on both evaluation metrics compared with [82, 148]. Considering `Benchmark2`, with fewer physical

design tool runs, our algorithm averagely surpasses [116] with the average hypervolume error and the ADRS value of 0.083 and 0.062, while it reduces 55.9% hypervolume error and 49.2% ADRS value comparing to [94]. Besides, our method behaves better than [95] by decreasing 48.1% hypervolume error and shrinking half ADRS value. With about 50% average reductions on hypervolume error and ADRS, our method shows a better performance against [82,148]. In a nutshell, the quantitative results in TABLE 6.2 and TABLE 6.3 illustrate the superiority of our method.

In addition, the visualizations of the points on Pareto frontier predicted by the methods on two benchmarks are displayed in Fig. 6.6. Pareto frontiers in area vs. delay space, power vs. delay space, and area vs. power vs. delay space are discovered. In Fig. 6.6, golden star dots represent the points on golden Pareto frontier, and thin diamond dots in cyan refer to the predictions by our method, while other different markers with distinct colors stand for the compared frameworks. It can be observed that Pareto frontiers searched by our approach are much closer to the golden frontiers than the frontiers explored by other methods. Even some predictions by ours exactly match the golden results (e.g. see in Figs. 6.6(b) and 6.6(d)). Briefly, Fig. 6.6 visually demonstrates that our method predicts the better Pareto frontiers.

6.6 Summary

In this chapter, for the first time, we have proposed an effective parameter tuning flow for the certain physical design tool, which is built upon an information gain-based multi-objective Bayesian optimization framework surrogated with a multi-task Gaussian process model. The optimization framework discovers the dependencies among multiple QoR metrics, and explores the high quality parameter configurations. The experimental results on two industrial benchmarks under advanced 7nm technology node have demon-

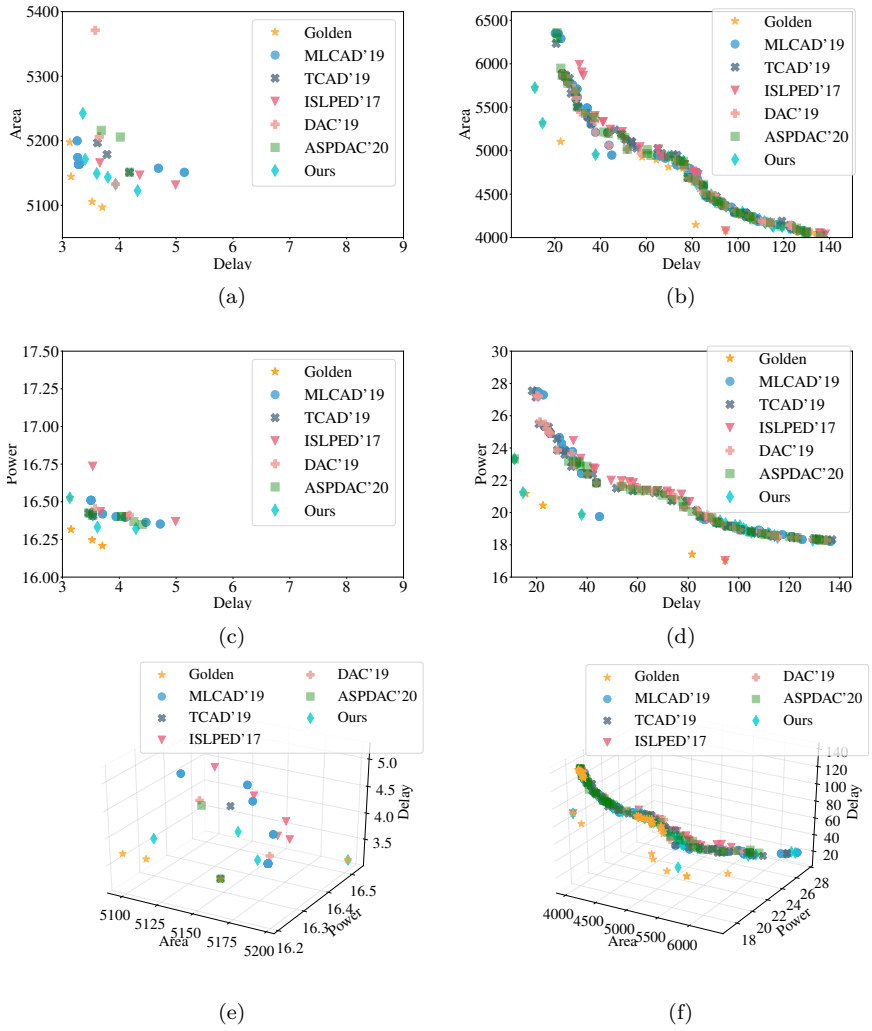


Figure 6.6 The visualizations of Pareto frontiers on two industrial Benchmarks (best viewed in color and zoomed in): (a) Pareto frontiers: area vs. delay on `Benchmark1`; (b) Pareto frontiers: area vs. delay on `Benchmark2`; (c) Pareto frontiers: power vs. delay on `Benchmark1`; (d) Pareto frontiers: power vs. delay on `Benchmark2`; (e) Pareto frontiers: area vs. power vs. delay on `Benchmark1`; (f) Pareto frontiers: area vs. power vs. delay on `Benchmark2`. The units for area, power and delay are ns, mW and um^2 respectively.

strated the efficacy and effectiveness of the proposed framework. In the future, we expect to extend our framework to tackle more VLSI design space exploration problems.

□ **End of chapter.**

Table 6.2 The quantitive comparison of the Pareto frontiers on `Benchmark1`.

Multi-objective	ISLPED'17 [116]			TCAD'19 [94]			MLCAD'19 [95]			DAC'19 [82]			ASPDAC'20 [148]			Ours		
	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.
Area-Delay	0.163	0.095	427	0.209	0.088	533	0.175	0.063	400	0.155	0.086	599	0.218	0.099	400	0.104	0.047	400
Power-Delay	0.212	0.073	423	0.232	0.065	512	0.188	0.064	400	0.245	0.080	587	0.182	0.059	400	0.140	0.039	400
Area-Power-Delay	0.299	0.098	451	0.218	0.063	475	0.181	0.068	400	0.232	0.079	566	0.202	0.067	400	0.118	0.045	400
Average	0.225	0.089	433.667	0.220	0.072	506.667	0.181	0.065	**400**	0.211	0.082	584	0.201	0.075	**400**	**0.121**	**0.044**	**400**
Ratio	1.862	2.031	1.084	1.820	1.649	1.267	1.503	1.489	**1.000**	1.741	1.856	1.460	1.658	1.705	**1.000**	**1.000**	**1.000**	**1.000**

Table 6.3 The quantitive comparison of the Pareto frontiers on `Benchmark2`.

Multi-objective	ISLPED'17 [116]			TCAD'19 [94]			MLCAD'19 [95]			DAC'19 [82]			ASPDAC'20 [148]			Ours		
	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.	HV	ADRS	Simu.
Area-Delay	0.164	0.121	566	0.142	0.068	509	0.122	0.061	400	0.129	0.075	627	0.133	0.063	400	0.053	0.032	400
Power-Delay	0.239	0.695	560	0.237	0.235	512	0.199	0.256	400	0.243	0.266	596	0.190	0.178	400	0.099	0.100	400
Area-Power-Delay	0.185	0.134	479	0.185	0.064	503	0.160	0.058	400	0.214	0.100	577	0.195	0.086	400	0.098	0.055	400
Average	0.196	0.317	535	0.188	0.122	508	0.160	0.125	**400**	0.195	0.147	600	0.173	0.109	**400**	**0.083**	**0.062**	**400**
Ratio	2.352	5.080	1.338	2.256	1.963	1.270	1.924	2.005	**1.000**	2.353	2.371	1.500	2.080	1.758	**1.000**	**1.000**	**1.000**	**1.000**

Chapter 7

Conclusion

In the thesis, we have supplied a few methodologies related to feature learning and attribute value optimization in different VLSI CAD applications at distinct stages of the chip design flow. The feature issues (e.g. hand-crafted feature and attribute value design, isolated feature extraction) we have combatted cover the physical synthesis and mask synthesis, ranging from the pre-layout phase to post-layout. In this chapter, we first conclude each proposed methodology and then discuss the future extensions.

7.1 summary

SRAF insertion as one of the resolution enhancement techniques has been widely leveraged to increase the continuously developed process window. Quite a few cleanups and simplifications are required for SRAF insertion from the initial generation to the final determination. However, previous ML-based SRAF insertion tools considerably rely on the manually designed feature (i.e. CCAS feature) which has much redundancy. We have proposed a supervised dictionary learning-based feature revision in Chapter 3 to smooth away the redundancy. Besides, our whole SRAF insertion framework maps the insertion process to a linear programming issue, which abstains from the sub-optimum brought about by the prior greedy insertion

strategy. Our flow is superior to the prior insertion tools in terms of the lithographic performance on PV band area and EPE.

Lithography variations majorly contribute to the decrease in the quality of the circuits. Besides the RETs, layout hotspot detection is another solution to relieve the variations. Existing deep learning-based detectors suffer from suboptimal performance owing to a two-stage flow scheme and less efficient representations of layout features. Specifically, the two-stage flow separates feature (e.g. DCT feature) extraction process from the consequent training networks, which degrades the performance of the detector. In Chapter 4, we have proposed a new end-to-end hotspot detection flow where layout feature embedding and hotspot detection are jointly carried out. The deep layout metric learning renders a new direction to extract features from both via layer patterns and metal layer patterns. Several latest progresses of deep learning like inception module and attention techniques to make backbone network self-adaptively concentrate on more informative parts. Furthermore, to evaluate the true state of hotspot detectors, a new via layer benchmark suite has been employed for comprehensive verification. Our detector shows a surpassed performance against the previous detectors.

Adders are the essential units in the modern VLSI. Traditional adders such as Kogge-Stone and Sklansky involve much human assiduousness and masses of trials. Meanwhile, common adder synthesis algorithms are not capable of tackling the design task in a huge adder design space. Unfortunately, current ML-based adder DSE works have not developed efficient feature extractors for adder structure. The manually crafted features may lose partial structure information. We have proposed a new end-to-end learning model, graph neural process, where a graph autoencoder for feature learning on prefix adder structures and a neural process as a regressor are simultaneously conducted. The graph neural process is incorporated into the proposed sequential optimization model-based DSE methodology to guide design space exploration for high-quality

adder designs. Apart from the proposed advanced adder feature extractor and layout-level performance predictor, there are some other factors for that the whole framework works effectively and are able to find near Pareto-optimal adder designs. One is the feasible and reasonable adder generation method and space pruning scheme. An exhaustive bottom-up and pruning based enumeration technique for prefix adder generation is exploited. It can produce all possible $n+1$ bit prefix graph structures from any n bit prefix graph. Then, we select the representative adder designs in the large design space by Quasi-random sampling which is based on a two-level (max-fan-out constraint and size) binning scheme. The approach evenly samples the prefix adders covering different architectural bins. Another reason is that our DSE framework considers the factors affecting the final layout performance cross multiple stages from architecture design to logic synthesis and physical synthesis. The corresponding experimental results in Chapter 5 have demonstrated the superiority of the proposed framework over the existing arts.

Over the past decades, physical design tools with scores of sophisticated algorithms incorporated have been developed to promote chip yield, design quality and reduce time-to-market. Manual configuring the input parameter values of physical design tools based on the expertise lasts for many years, which may be time-consuming and un-robust. In Chapter 6, we treat the input parameters as features and exploit a multi-objective Bayesian optimization framework to tune the attribute values. Multi-task Gaussian process regressor is harnessed as the surrogate model in the proposed Bayesian optimization framework, which captures the correlations among the tool parameters (or features) and objectives (i.e. QoR metrics). Two industrial benchmarks under advanced 7nm technology node have demonstrated the efficacy and effectiveness of the proposed framework.

7.2 Possible Future Directions

SRAF insertion, as one of the prevailing RETs at the mask synthesis stage, require growing computational resources as well as OPC. Recently, deep generative networks like generative adversarial networks (GAN) and cycle generative adversarial networks (CyGANs) are applied to the SRAF generation task, which is the new feasible measure to perform better than Mentor/Calibre [106] commercial tool.

Concerning hotspot detection, in the future, we expect to have more cooperations with industry (e.g. fabs or fabless companies) to solve the cutting-edge practical problems. Although even the customized hotspot detector with more advanced machine learning and deep learning techniques cannot be guaranteed to detect all hotspots, especially for new test-cases, with a selective learning mechanism (e.g., an integrated reject option where the model chooses to abstain from predicting hotspot and non-hotspot labels when misclassification risk is high), it has the potential to aid and accelerate the detection flow under the supervision of the IC designers. Besides, multi-layer layout hotspot detection and full-chip scale detection are promising directions, and layout clipper can be further investigated.

For the techniques in adder design space exploration, as mentioned in the result part of Chapter 5, our DSE framework is readily extended to different bit-width adder designs in theory. Accordingly, generalizing and transferring the DSE model with certain domain adaptation techniques among different bit-width even under distinct technology nodes may be worth studying. In addition, the multiplier DSE problem is to be solved in demand.

When it turns to the tool parameter tuning, more parameters can be explored and the associated design space pruning techniques are required. On the other hand, matrices computations are dense and will be involved many times during the tuning processes. it is

worth to thinking of methods to introduce sparsity or speedup matrices computation. Apart from the above, the sampling techniques employed in initial dataset generation need to be further developed.

□ **End of chapter.**

Appendix A

Equation Derivation and Proof

A.1 SRAF Insertion

A.1.1 Calculation of Gradient

In t-th iteration to update atoms of proposed algorithm, with $\boldsymbol{x}$ fixed, Equation (3.10) can be rewritten as in Equations (A.1) to (A.5).

$$\boldsymbol{D}_t \triangleq \arg\min_{\boldsymbol{D}} \frac{1}{2t} \left\| \boldsymbol{Y}_t - \boldsymbol{D}_{t-1}\boldsymbol{X}_t \right\|_F^2 \tag{A.1}$$

$$\triangleq \arg\min_{\boldsymbol{D}} \frac{1}{2} tr\left[\left(\boldsymbol{Y}_t - \boldsymbol{D}_{t-1}\boldsymbol{X}_t \right)^\top \left(\boldsymbol{Y}_t - \boldsymbol{D}_{t-1}\boldsymbol{X}_t \right) \right] \tag{A.2}$$

$$\triangleq \arg\min_{\boldsymbol{D}} \frac{1}{2} tr\left(\boldsymbol{D}_{t-1}^\top \boldsymbol{D}_{t-1}\boldsymbol{X}_t\boldsymbol{X}_t^\top - 2\boldsymbol{D}_{t-1}^\top \boldsymbol{Y}_t\boldsymbol{X}_t^\top \right) \tag{A.3}$$

$$\triangleq \arg\min_{\boldsymbol{D}} \left(\frac{1}{2} tr\left(\boldsymbol{D}_{t-1}^\top \boldsymbol{D}_{t-1}\boldsymbol{C}_t \right) - tr\left(\boldsymbol{D}_{t-1}^\top \boldsymbol{B}_t \right) \right) \tag{A.4}$$

$$\triangleq \arg\min_{\boldsymbol{D}} \left(\frac{1}{2} \sum_k \boldsymbol{d}_k^\top \sum_i \boldsymbol{d}_i \boldsymbol{c}_{ik} - \sum_k \boldsymbol{d}_k^\top \boldsymbol{b}_k \right). \tag{A.5}$$

In the stage of updating atoms in new algorithm, block coordinate descent algorithm, which means updating one atom (i.e., $\boldsymbol{d}_j$) while fixing other atoms, is still exploited with warm start mechanism. Therefore, the updating rule for atoms can be derived from the

following equation.

$$\frac{\partial(\text{A.5})}{\partial \boldsymbol{d}_j} = \frac{\partial\left(\frac{1}{2}\sum_k \boldsymbol{d}_k^\top \sum_i \boldsymbol{d}_i \boldsymbol{c}_{ik} - \sum_k \boldsymbol{d}_k^\top \boldsymbol{b}_k\right)}{\partial \boldsymbol{d}_j} \tag{A.6}$$

$$= \frac{\partial\left(\sum_{k\neq j} \boldsymbol{d}_j^\top \boldsymbol{d}_k \boldsymbol{c}_{kj} + \frac{1}{2}\boldsymbol{d}_j^\top \boldsymbol{d}_j \boldsymbol{c}_{jj} - \sum_k \boldsymbol{d}_k^\top \boldsymbol{b}_k\right)}{\partial \boldsymbol{d}_j} \tag{A.7}$$

$$= \sum_{k\neq j} \boldsymbol{d}_k \boldsymbol{c}_{kj} + \boldsymbol{d}_j \boldsymbol{c}_{jj} - \boldsymbol{b}_j \tag{A.8}$$

$$= \boldsymbol{D}\boldsymbol{c}_j - \boldsymbol{b}_j. \tag{A.9}$$

A.1.2 Proof of Theorem 1

Proof. As $\boldsymbol{C}_t$ and $\boldsymbol{B}_t$ are in a compact set, extracting converging sequences becomes possible. Therefore, the assumption could be made that two sequences converge to $\boldsymbol{C}_\infty$ and $\boldsymbol{B}_\infty$. As a result, $\boldsymbol{D}_t$ converges to $\boldsymbol{D}_\infty$. Assuming that $\boldsymbol{V} \in \mathbb{R}^{(n+s+1)\times(s)}$, $\hat{f}_t$ upperbounds the empirical cost f_t, i.e. $\hat{f}_t\left(\boldsymbol{D}_t + \boldsymbol{V}\right) \geq f_t\left(\boldsymbol{D}_t + \boldsymbol{V}\right)$. When $t \to \infty$, the inequality, $\hat{f_\infty}\left(\boldsymbol{D}_\infty + \boldsymbol{V}\right) \geq f\left(\boldsymbol{D}_\infty + \boldsymbol{V}\right)$, still holds.

Introduce a sequence $a_t > 0$ which converges to 0. With harnessing the Taylor expansion and using $\hat{f_\infty}\left(\boldsymbol{D}_\infty\right) = f\left(\boldsymbol{D}_\infty\right)$, the inequality (A.10) exists.

$$\begin{aligned} & f\left(\boldsymbol{D}_\infty\right) + tr\left(a_t \boldsymbol{V}^\top \nabla \hat{f_\infty}\left(\boldsymbol{D}_\infty\right)\right) \\ \geq & f\left(\boldsymbol{D}_\infty\right) + tr\left(a_t \boldsymbol{V}^\top \nabla f\left(\boldsymbol{D}_\infty\right)\right). \end{aligned} \tag{A.10}$$

This inequality holds for all $\boldsymbol{V}$, $\nabla \hat{f_\infty}\left(\boldsymbol{D}_\infty\right) = \nabla f\left(\boldsymbol{D}_\infty\right)$. A first-order necessary optimality condition for $\boldsymbol{D}_\infty$ being an optimum of $\hat{f_\infty}$ is that $-\nabla \hat{f_\infty}$ is in the normal cone of the convex set of dictionary matrices at $\boldsymbol{D}_\infty$ [17]. So the first-order necessary optimality condition for $\boldsymbol{D}_\infty$ being an optimum of f is also validated. Since $\boldsymbol{B}_t$ and $\boldsymbol{C}_t$ asymptotically get close to their accumulation points, $-\nabla f\left(\boldsymbol{D}_t\right)$ will be close to the normal cone at $\boldsymbol{D}_t$. □

A.2 Layout Hotspot Detection

A.2.1 The Proof for Theorem 2

Proof. The proof follows from [108, 143]. $F_{\mathcal{T}}$ is defined as a replacement to $\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)] - \mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot))$. From Equation (A.11a) to Equation (A.11b), we exploit the triangle inequality. Then, the upper-bound of Equation (A.11b) is attained by using Jensen's inequality and the definition of $\mathcal{L}$. With the combination of the triangle inequality, β-uniform stability and $\mathcal{B}$-boundedness, the further bound is found in Equation (A.11d).

$$
\begin{aligned}
&|F_{\mathcal{T}} - F_{\mathcal{T}_i}| \\
=\ &\Big|\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot)) - \mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)) + \mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)] \\
&-\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)]\Big| && \text{(A.11a)} \\
\leq\ &|\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)] \\
&-\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)| \\
&+\Big|\mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)) - \mathcal{L}(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot))\Big| && \text{(A.11b)} \\
\leq\ &\beta + \frac{1}{|\mathcal{T}|}\Bigg|\sum_{j\neq i}\sum_{k\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l) - \ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_l)) \\
&+\sum_{j\neq i}\sum_{k\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i) - \ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_k,\boldsymbol{x}_i')) \\
&+\sum_{j\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i,\boldsymbol{x}_l) - \ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_j,\boldsymbol{x}_i',\boldsymbol{x}_l)) \\
&+\sum_{k\neq i}\sum_{l\neq i}(\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_k,\boldsymbol{x}_l) - \ell(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i',\boldsymbol{x}_k,\boldsymbol{x}_l))\Bigg| && \text{(A.11c)} \\
\leq\ &\beta + \beta + \frac{3\mathcal{B}}{n} \leq 2\beta + \frac{3\mathcal{B}}{n}. && \text{(A.11d)}
\end{aligned}
$$

Based on the upper-bound obtained in Equation (A.11), we utilize McDiarmid's inequality [105] to obtain Equation (A.12).

$$
\Pr[F_{\mathcal{T}} \geq \epsilon + \mathbb{E}[F_{\mathcal{T}}]] \leq \exp\left(\frac{-2n\epsilon^2}{(2n\beta + 3B)^2}\right). \tag{A.12}
$$

With δ set to be $\exp\left(\frac{-2n\epsilon^2}{(2n\beta+3\mathcal{B})^2}\right)$, ϵ equals to $(2n\beta+3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}$. Hence, with confidence $1-\delta$, Equation (A.13) exists.

$$F_{\mathcal{T}} \leq \mathbb{E}[F_{\mathcal{T}}] + (2n\beta+3\mathcal{B})\sqrt{\frac{\log\frac{1}{\delta}}{2n}}. \tag{A.13}$$

For an effective bound, $\beta = o\left(\frac{1}{\sqrt{n}}\right)$. Assume $\beta = \mathcal{O}\left(n^p\right)$. Since $\lim_{n\to+\infty}\frac{-n}{(2n\beta+3\mathcal{B})^2} = -\infty$, $1 > 2*(1+p)$ holds and $\beta = o\left(\frac{1}{\sqrt{n}}\right)$.

Equation (A.14) shows the searching computing of the upper-bound of $\mathbb{E}[F_{\mathcal{T}}]$. Note that from Equation (A.14a) to Equation (A.14b), we harness a fact that replacing the examples with i.i.d exemplars does not change the expected computation. More specifically, $\mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\right)] = \mathbb{E}_{\mathcal{T}\sim\mathcal{D}}[\mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot)\right)]$.

$$\mathbb{E}[F_{\mathcal{T}}] = \mathbb{E}\left[\mathbb{E}_{\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[\ell(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)] - \mathcal{L}\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot)\right)\right] \tag{A.14a}$$

$$\leq \mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}[|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)|] \tag{A.14b}$$

$$= \mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\left[\left|\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) + \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) + \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\right] \tag{A.14c}$$

$$\leq \mathbb{E}_{\mathcal{T},\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\sim\mathcal{D}}\left[\frac{1}{n}\sum_{(\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k)\in\mathcal{T}}\left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right| + \left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_{i,j}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right| + \left|\ell\left(f_{\boldsymbol{w}_{\mathcal{T}_i}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right) - \ell\left(f_{\boldsymbol{w}_{\mathcal{T}}}(\cdot),\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k\right)\right|\right] \leq 3\beta. \tag{A.14d}$$

In Equation (A.14), $f_{\boldsymbol{w}_{\mathcal{T}_{i,j,k}}}(\cdot)$ is the mapping function learned over the training set $\mathcal{T}$ with $\boldsymbol{x}_i,\boldsymbol{x}_j,\boldsymbol{x}_k$ replaced by $\boldsymbol{x}'_i,\boldsymbol{x}'_j,\boldsymbol{x}'_k$. Combing the results of Equation (A.13) and Equation (A.14), Inequality (4.10) holds. Therefore, with $\beta = o\left(\frac{1}{\sqrt{n}}\right)$, the generalization gap will

converge in the order of $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$ with high confidence $1-\delta$. The proof completes. □

□ **End of chapter.**

Bibliography

[1] Synopsys Design Compiler. [Online]. Available: "http://www.synopsys.com". Accessed: Apr. 23, 2016.

[2] Synopsys IC Compiler. [Online]. Available: "http://www.synopsys.com". Accessed: Apr. 23, 2016.

[3] Synopsys SAED Library. [Online]. Available: http://www.synopsys.com/Community/UniversityProgram/Pages/32-28nm-generic-library.aspx. Accessed: April. 23, 2016.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al. TensorFlow: A system for large-scale machine learning. In *Proc. OSDI*, pages 265–283, 2016.

[5] A. Agnesina, K. Chang, and S. K. Lim. VLSI placement parameter optimization using deep reinforcement learning. In *Proc. ICCAD*, pages 1–9, 2020.

[6] M. Aharon, M. Elad, and A. Bruckstein. k-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[7] M. B. Alawieh, W. Li, Y. Lin, L. Singhal, M. A. Iyer, and D. Z. Pan. High-definition routing congestion prediction for large-scale fpgas. In *Proc. ASPDAC*, pages 26–31, 2020.

[8] M. B. Alawieh, Y. Lin, Z. Zhang, M. Li, Q. Huang, and D. Z. Pan. GAN-SRAF: Sub-resolution assist feature generation using conditional generative adversarial networks. In *Proc. DAC*, pages 149:1–149:6, 2019.

[9] Z. Allen-Zhu and L. Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. In *Innovations in Theoretical Computer Science Conference (ITCS)*, 2017.

[10] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon. Performance indicators in multiobjective optimization. *Optimization Online*, 2018.

[11] C. Bai, Q. Sun, J. Zhai, Y. Ma, B. Yu, and M. D. Wong. BOOM-Explorer: RISC-V BOOM microarchitecture design space exploration framework. In *Proc. ICCAD*, pages 1–9, 2021.

[12] S. Banerjee, Z. Li, and S. R. Nassif. ICCAD-2013 CAD contest in mask optimization and benchmark suite. In *Proc. ICCAD*, pages 271–274, 2013.

[13] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences (SIIMS)*, 2(1):183–202, 2009.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *Proc. NIPS*, volume 24, 2011.

[15] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[16] E. V. Bonilla, K. Chai, and C. Williams. Multi-task Gaussian process prediction. *Proc. NIPS*, 20:153–160, 2007.

[17] J. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer Science & Business Media, 2010.

[18] L. Bottou and O. Bousquet. The trade-offs of large scale learning. In *Proc. NIPS*, pages 161–168, 2008.

[19] J. Burkardt. The truncated normal distribution. *Department of Scientific Computing Website, Florida State University*, pages 1–35, 2014.

[20] J. P. Cain, M. Fakhry, P. Pathak, J. Sweis, F. E. Gennari, and Y.-C. Lai. Pattern-based analytics to estimate and track yield risk of designs down to 7nm. In *SPIE Advanced Lithography*, volume 10148, 2017.

[21] J. P. Cain, M. Fakhry, P. Pathak, J. Sweis, F. E. Gennari, and Y.-C. Lai. Pattern-based analytics to estimate and track yield risk of designs down to 7nm. In *Proc. SPIE*, volume 10148, page 1014805, 2017.

[22] L. Capodieci. Data analytics and machine learning for continued semiconductor scaling. In *SPIE News*, 2016.

[23] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu. DAMO: deep agile mask optimization for full chip scale. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[24] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu. DevelSet: Deep neural level set for instant mask optimization. In *Proc. ICCAD*, pages 1–9, 2021.

[25] J. Chen, Y. Lin, Y. Guo, M. Zhang, M. B. Alawieh, and D. Z. Pan. Lithography hotspot detection using a double inception module architecture. *JM3*, 18(1):013507, 2019.

[26] R. Chen, S. Hu, Z. Chen, S. Zhu, B. Yu, P. Li, C. Chen, Y. Huang, and J. Hao. A unified framework for layout pattern analysis with deep causal estimation. In *Proc. ICCAD*, pages 1–9, 2021.

[27] R. Chen, W. Zhong, H. Yang, H. Geng, F. Yang, X. Zeng, and B. Yu. Faster region-based hotspot detection. *IEEE TCAD*, 2020.

[28] R. Chen, W. Zhong, H. Yang, H. Geng, X. Zeng, and B. Yu. Faster region-based hotspot detection. In *Proc. DAC*, pages 1–6, 2019.

[29] T. Chen, B. Lin, H. Geng, S. Hu, and B. Yu. Leveraging spatial correlation for sensor drift calibration in smart building. *IEEE TCAD*, 2020.

[30] T. Chen, B. Lin, H. Geng, and B. Yu. Sensor drift calibration via spatial correlation model in smart building. In *Proc. DAC*, pages 1–6, 2019.

[31] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu. Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks. In *Proc. ASPDAC*, pages 898–903, 2021.

[32] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu. Deep H-GCN: Fast analog IC aging-induced degradation estimation. *IEEE TCAD*, 2021.

[33] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan. Semi-supervised hotspot detection with self-paced multi-task learning. In *Proc. ASPDAC*, pages 420–425, 2019.

[34] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *Proc. CVPR*, pages 1335–1344, 2016.

[35] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Proc. CVPR*, pages 1153–1162, 2016.

[36] S. Dai, Y. Zhou, H. Zhang, E. Ustun, E. F. Young, and Z. Zhang. Fast and accurate estimation of quality of results in high-level synthesis with machine learning. pages 129–132, 2018.

[37] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. CVPR*, volume 1, pages 886–893, 2005.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[39] J.-A. Désidéri. Multiple-gradient descent algorithm (MGDA) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5-6):313–318, 2012.

[40] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan. EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation. In *Proc. ASPDAC*, pages 263–270, 2012.

[41] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[42] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu. Dual attention network for scene segmentation. In *cvpr*, pages 3146–3154, 2019.

[43] M. J. Gangeh, A. K. Farahat, A. Ghodsi, and M. S. Kamel. Supervised dictionary learning and sparse representation-a review. *arXiv preprint arXiv:1502.05928*, 2015.

[44] M. J. Gangeh, A. Ghodsi, and M. S. Kamel. Kernelized supervised dictionary learning. *IEEE Transactions on Signal Processing*, 61(19):4753–4767, 2013.

[45] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *Proc. DAC*, pages 52:1–52:6, 2014.

[46] X. Gao, S. C. Hoi, Y. Zhang, J. Wan, and J. Li. SOML: Sparse online metric learning with application to image retrieval. In *Proc. AAAI*, pages 1206–1212, 2014.

[47] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional neural processes. In *Proc. ICML*, pages 1690–1699, 2018.

[48] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[49] H. Geng, Y. Ma, Q. Xu, J. Miao, S. Roy, and B. Yu. High-speed adder design space exploration via graph neural processes. *IEEE TCAD*, 2021.

[50] H. Geng, F. Yang, X. Zeng, and B. Yu. When wafer failure pattern classification meets few-shot learning and self-supervised learning. In *Proc. ICCAD*, pages 1–8, 2021.

[51] H. Geng, H. Yang, B. Yu, X. Li, and X. Zeng. Sparse VLSI layout feature extraction: A dictionary learning approach. In *Proc. ISVLSI*, pages 488–493, 2018.

[52] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu. Hotspot detection via attention-based deep layout metric learning. In *Proc. ICCAD*, 2020.

[53] H. Geng, H. Yang, L. Zhang, J. Miao, F. Yang, X. Zeng, and B. Yu. Hotspot detection via attention-based deep layout metric learning. *IEEE TCAD*, 2021.

[54] H. Geng, W. Zhong, H. Yang, Y. Ma, J. Mitra, and B. Yu. SRAF insertion via supervised dictionary learning. pages 406–411, 2019.

[55] H. Geng, W. Zhong, H. Yang, Y. Ma, J. Mitra, and B. Yu. Sraf insertion via supervised dictionary learning. *IEEE TCAD*, 39(10):2849–2859, 2020.

[56] G. H. Golub, P. C. Hansen, and D. P. O'Leary. Tikhonov regularization and total least squares. *SIAM Journal on Matrix Analysis and Applications*, 21(1):185–194, 1999.

[57] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[58] C. Goutte and E. Gaussier. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In *Proc. ECIR*, pages 345–359. Springer, 2005.

[59] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear gauss–seidel method under convex constraints. *Operations research letters*, 26(3):127–136, 2000.

[60] A. Gu and A. Zakhor. Optical proximity correction with linear regression. *IEEE TSM*, 21(2):263–271, 2008.

[61] J. Guo, F. Yang, S. Sinha, C. Chiang, and X. Zeng. Improved tangent space based distance metric for accurate lithographic hotspot classification. In *Proc. DAC*, pages 1173–1178, 2012.

[62] Q. Guo, T. Chen, Y. Chen, Z.-H. Zhou, W. Hu, and Z. Xu. Effective and efficient microprocessor design space exploration using unlabeled design configurations. In *Proc. IJCAI*, 2011.

[63] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Proc. NIPS*, pages 1024–1034, 2017.

[64] B. Harwood, B. Kumar, G. Carneiro, I. Reid, T. Drummond, et al. Smart mining for deep metric learning. In *Proc. ICCV*, pages 2821–2829, 2017.

[65] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778, 2016.

[66] D. Hernández-Lobato, J. Hernandez-Lobato, A. Shah, and R. Adams. Predictive entropy search for multi-objective Bayesian optimization. In *Proc. ICML*, pages 1492–1501, 2016.

[67] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Proc. NIPS*, pages 918–926, 2014.

[68] J. Hu, J. Lu, and Y.-P. Tan. Discriminative deep metric learning for face verification in the wild. In *Proc. CVPR*, pages 1875–1882, 2014.

[69] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *Proc. NIPS*, pages 1262–1270, 2016.

[70] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong, et al. Machine learning for electronic design automation: A survey. *ACM TODAES*, 26(5):1–46, 2021.

[71] B. Jiang, L. Liu, Y. Ma, H. Zhang, B. Yu, and E. F. Young. Neural-ILT: migrating ILT to neural networks for mask printability and complexity co-optimization. In *Proc. ICCAD*, pages 1–9, 2020.

[72] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng. Efficient layout hotspot detection via binarized residual neural network. In *Proc. DAC*, page 147, 2019.

[73] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *Proc. CVPR*, pages 1697–1704, 2011.

[74] R. Jin, S. Wang, and Y. Zhou. Regularized distance metric learning: Theory and algorithm. In *Proc. NIPS*, pages 862–870, 2009.

[75] I. Jolliffe. *Principal Component Analysis*. Wiley Online Library, 2005.

[76] J. Jun, M. Park, C. Park, H. Yang, D. Yim, M. Do, D. Lee, T. Kim, J. Choi, G. Luk-Pat, et al. Layout optimization with assist features placement by model based rule tables for 2x node random contact. In *Proc. SPIE*, volume 9427, 2015.

[77] M. Kandemir. Asymmetric transfer learning with deep Gaussian processes. In *Proc. ICML*, pages 730–738, 2015.

[78] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. In *Proc. ICLR*, 2019.

[79] T. N. Kipf and M. Welling. Variational graph auto-encoders. *NeurIPS Workshop on Bayesian Deep Learning*, 2016.

[80] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *Theory of Computing*, 100(8):786–793, 1973.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Proc. NIPS*, 25:1097–1105, 2012.

[82] J. Kwon, M. M. Ziegler, and L. P. Carloni. A learning-based recommender system for autotuning design fiows of industrial high-performance processors. In *Proc. DAC*, pages 1–6, 2019.

[83] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[84] D. Li, S. Yao, Y.-H. Liu, S. Wang, and X.-H. Sun. Efficient design space exploration via statistical sampling and adaboost learning. In *Proc. DAC*, pages 1–6, 2016.

[85] W. Li, G. Chen, H. Yang, R. Chen, and B. Yu. Learning point clouds in EDA. In *Proc. ISPD*, pages 55–62, 2021.

[86] W. Li, Y. Qu, G. Chen, Y. Ma, and B. Yu. TreeNet: Deep point cloud embedding for routing tree construction. In *Proc. ASPDAC*, pages 164–169, 2021.

[87] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu. DRC hotspot prediction at sub-10nm process nodes using customized convolutional network. In *Proc. ISPD*, pages 135–142, 2020.

[88] J.-M. Lin and Y.-W. Chang. TCG: A transitive closure graph-based representation for non-slicing floorplans. In *Proc. DAC*, pages 764–769, 2001.

[89] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang. A novel fuzzy matching model for lithography hotspot detection. In *Proc. DAC*, pages 68:1–68:6, 2013.

[90] C. Lo and P. Chow. Multi-fidelity optimization for high-level synthesis directives. In *International Conference on Field Programmable Logic and Applications (FPL)*, pages 272–2727, 2018.

[91] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim. GAN-CTS: A generative adversarial framework for clock tree prediction and optimization. In *Proc. ICCAD*, pages 1–8, 2019.

[92] W. Lyu, F. Yang, C. Yan, D. Zhou, and X. Zeng. Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design. In *Proc. ICML*, pages 3306–3314, 2018.

[93] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu. A unified framework for simultaneous layout decomposition and mask optimization. In *Proc. ICCAD*, pages 81–88, 2017.

[94] Y. Ma, S. Roy, J. Miao, J. Chen, and B. Yu. Cross-layer optimization for high speed adders: A Pareto driven machine learning approach. *IEEE TCAD*, 38(12):2298–2311, 2019.

[95] Y. Ma, Z. Yu, and B. Yu. CAD tool design space exploration via Bayesian optimization. In *Proc. MLCAD*, 2019.

[96] C. A. Mack. Scattering bars. *Solid State Technology*, 2003.

[97] P. C. Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[98] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proc. ICML*, pages 689–696, 2009.

[99] H. M. Makrani, F. Farahmand, H. Sayadi, S. Bondi, S. M. P. Dinakarrao, H. Homayoun, and S. Rafatirad. Pyramid: Machine learning framework to estimate the optimal timing and resource usage of a high-level synthesis design. In *Proc. FPL*, pages 397–403, 2019.

[100] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[101] T. Matsunaga and Y. Matsunaga. Area minimization algorithm for parallel prefix adders under bitwise delay constraints. In *Proc. GLSVLSI*, pages 435–440, 2007.

[102] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan. A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In *Proc. SPIE*, volume 9427, 2015.

[103] T. Matsunawa, B. Yu, and D. Z. Pan. Optical proximity correction with hierarchical bayes model. In *Proc. SPIE*, volume 9426, 2015.

[104] T. Matsunawa, B. Yu, and D. Z. Pan. Laplacian eigenmaps- and Bayesian clustering-based layout pattern sampling and its applications to hotspot detection and optical proximity correction. *JM3*, 15(4):043504–043504, 2016.

[105] C. McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[106] Mentor Graphics. Calibre verification user's manual, 2008.

[107] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[108] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[109] D. Z. Pan, B. Yu, and J.-R. Gao. Design for manufacturing with emerging nanolithography. *IEEE TCAD*, 32(10):1453–1472, 2013.

[110] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[111] D. Park and Y. Kim. Fast Pareto front exploration for design of reconfigurable energy storage. *IEEE TCAD*, 38(3):526–537, 2018.

[112] A. Paszke, S. Gross, S. Chintala, and G. Chanan. Pytorch, 2017.

[113] D. Patel, R. Bonam, and A. A. Oberai. Engineering neural networks for improved defect detection and classification. In *Proc. SPIE*, volume 10959, 2019.

[114] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. Asilomar*, pages 40–44. IEEE, 1993.

[115] S. Roy, M. Choudhury, R. Puri, and D. Z. Pan. Polynomial time algorithm for area and power efficient adder synthesis in high-performance designs. In *Proc. ASPDAC*, pages 249–254, 2015.

[116] S. Roy, Y. Ma, J. Miao, and B. Yu. A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders. In *Proc. ISLPED*, pages 1–6, 2017.

[117] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint*, 2016.

[118] T. G. Rudner, V. Fortuin, Y. W. Teh, and Y. Gal. On the connection between neural processes and Gaussian processes with deep kernels. In *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

[119] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, pages 815–823, 2015.

[120] S. D. Shang, L. Swallow, and Y. Granik. Model-based SRAF insertion, Oct. 11 2011.

[121] S. Shim, S. Choi, and Y. Shin. Light interference map: A prescriptive optimization of lithography-friendly layout. *IEEE TSM*, 29(1):44–49, 2016.

[122] S. Shim, W. Chung, and Y. Shin. Synthesis of lithography test patterns through topology-oriented pattern extraction and classification. In *Proc. SPIE*, volume 9053, 2014.

[123] S. Shim and Y. Shin. Topology-oriented pattern extraction and classification for synthesizing lithography test patterns. *JM3*, 14(1):013503–013503, 2015.

[124] J. Sklansky. Conditional sum addition logic. *IRE Trans. on Electronic Computers*, EC-9(2):226–231, 1960.

[125] K. Skretting and K. Engan. Recursive least squares dictionary learning algorithm. *IEEE Transactions on Signal Processing*, 58(4):2121–2130, 2010.

[126] Q. Sun, C. Bai, H. Geng, and B. Yu. Deep neural network hardware deployment optimization via advanced active learning. In *Proc. DATE*, pages 1510–1515, 2021.

[127] Q. Sun, T. Chen, S. Liu, J. Miao, J. Chen, H. Yu, and B. Yu. Correlated multi-objective multi-fidelity optimization for HLS directives design. In *Proc. DATE*, pages 46–51, 2021.

[128] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, pages 1–9, 2015.

[129] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. CVPR*, pages 2818–2826, 2016.

[130] H. Takahashi, H. Ogura, S. Sato, A. Takahashi, and C. Kodama. A feature selection method for weak classifier based hotspot detection. In *Proc. SPIE*, volume 11328, 2020.

[131] E. Teoh, V. Dai, L. Capodieci, Y.-C. Lai, and F. Gennari. Systematic data mining using a pattern database to accelerate yield ramp. In *Proc. SPIE*, volume 9053, page 905306, 2014.

[132] C. G. Tianqi Chen. XGBoost: A scalable tree boosting system. In *Proc. KDD*, pages 785–794, 2016.

[133] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58:267–288, 1996.

[134] Y. Tomioka, T. Matsunawa, C. Kodama, and S. Nojima. Lithography hotspot detection by two-stage cascade classifier using histogram of oriented light propagation. In *Proc. ASP-DAC*, pages 81–86, 2017.

[135] A. J. Torres. ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *Proc. ICCAD*, pages 349–350, 2012.

[136] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

[137] S. S.-E. Tseng, W.-C. Chang, I. H.-R. Jiang, J. Zhu, and J. P. Shiely. Efficient search of layout hotspot patterns for matching sem images using multilevel pixelation. In *Proc. SPIE*, volume 10961, page 109610B, 2019.

[138] E. Ustun, S. Xiang, J. Gui, C. Yu, and Z. Zhang. LAMDA: Learning-assisted multi-stage autotuning for fpga design closure. In *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 74–77, 2019.

[139] L. Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

[140] M. van der Wilk, V. Dutordoir, S. John, A. Artemev, V. Adam, and J. Hensman. A framework for interdomain and multioutput Gaussian processes. *arXiv:2003.01115*, 2020.

[141] M. Volpp, L. P. Fröhlich, K. Fischer, A. Doerr, S. Falkner, F. Hutter, and C. Daniel. Meta-learning acquisition functions for transfer learning in Bayesian optimization. *Proc. ICLR*, 2020.

[142] C. H. Wallace, P. A. Nyhus, and S. S. Sivakumar. Sub-resolution assist features, Dec. 15 2009.

[143] B. Wang, H. Zhang, P. Liu, Z. Shen, and J. Pineau. Multitask metric learning: Theory and algorithm. In *Proc. AISTATS*, pages 3362–3371, 2019.

[144] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. In *cvpr*, pages 3156–3164, 2017.

[145] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE TCAD*, 33(11):1671–1680, 2014.

[146] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[147] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon. Cbam: Convolutional block attention module. In *eccv*, pages 3–19, 2018.

[148] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza. FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning. In *Proc. ASP-DAC*, pages 19–25, 2020.

[149] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *Proc. ICCAD*, pages 1–8. IEEE, 2018.

[150] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *Proc. NIPS*, pages 521–528, 2003.

[151] J. Xu, K. N. Krishnamoorthy, E. Teoh, V. Dai, L. Capodieci, J. Sweis, and Y.-C. Lai. Design layout analysis and DFM optimization using topological patterns. In *Proc. SPIE*, volume 9427, 2015.

[152] X. Xu, T. Matsunawa, S. Nojima, C. Kodama, T. Kotani, and D. Z. Pan. A machine learning based framework for sub-resolution assist feature generation. In *Proc. ISPD*, pages 161–168, 2016.

[153] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *Proc. DAC*, pages 131:1–131:6, 2018.

[154] H. Yang, S. Li, C. Tabery, B. Lin, and B. Yu. Bridging the gap between layout pattern sampling and hotspot detection via batch active learning. *IEEE TCAD*, 40(7):1464–1475, 2020.

[155] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE TCAD*, 38(6):1175–1187, 2019.

[156] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Y. Young. Layout hotspot detection with feature tensor generation and deep biased learning. In *Proc. DAC*, pages 62:1–62:6, 2017.

[157] H. Yang, W. Zhong, Y. Ma, H. Geng, R. Chen, W. Chen, and B. Yu. VLSI mask optimization: From shallow to deep learning. *Integration, the VLSI Journal*, 77:96–103, 2021.

[158] Y. Yankelevsky and M. Elad. Structure-aware classification using supervised dictionary learning. In *Proc. ICASSP*, pages 4421–4425, 2017.

[159] W. Ye, M. B. Alawieh, M. Li, Y. Lin, and D. Z. Pan. LithoGPA: Gaussian process assurance for lithography hotspot detection. In *Proc. DATE*, pages 54–59, 2019.

[160] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan. LithoGAN: End-to-End lithography modeling with generative adversarial networks. In *Proc. DAC*, pages 107:1–107:6, 2019.

[161] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan. Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering. *JM3*, 14(1):011003, 2015.

[162] R. Yu, Z. Dou, S. Bai, Z. Zhang, Y. Xu, and X. Bai. Hard-aware point-to-set deep metric for person re-identification. In *Proc. ECCV*, pages 188–204, 2018.

[163] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang. Accurate process-hotspot detection using critical design rule extraction. In *Proc. DAC*, pages 1167–1172, 2012.

[164] Z. Yu, G. Chen, Y. Ma, and B. Yu. A GPU-enabled level set method for mask optimization. In *Proc. DATE*, pages 1835–1838, 2021.

[165] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *Proc. ICCV*, pages 814–823, 2017.

[166] B. R. Zeydel, T. T. J. H. Kluter, and V. G. Oklobdzija. Efficient mapping of addition recurrence algorithms in CMOS. *Proc. ARITH*, pages 107–113, 2005.

[167] H. Zhang, H. Yang, B. Yu, and E. F. Y. Young. VLSI layout hotspot detection based on discriminative feature extraction. In *Proc. APCCAS*, pages 542–545, 2016.

[168] H. Zhang, B. Yu, and E. F. Y. Young. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. In *Proc. ICCAD*, pages 47:1–47:8, 2016.

[169] Q. Zhang and B. Li. Discriminative K-SVD for dictionary learning in face recognition. In *Proc. CVPR*, pages 2691–2698. IEEE, 2010.

[170] S. Zhang, W. Lyu, F. Yang, C. Yan, D. Zhou, X. Zeng, and X. Hu. An efficient multi-fidelity Bayesian optimization approach for analog circuit synthesis. In *Proc. DAC*, pages 1–6, 2019.

[171] S. Zhang, F. Yang, D. Zhou, and X. Zeng. An efficient asynchronous batch Bayesian optimization approach for analog circuit synthesis. In *Proc. DAC*, pages 1–6, 2020.

[172] W. Zhang, X. Li, S. Saxena, A. Strojwas, and R. Rutenbar. Automatic clustering of wafer spatial signatures. In *Proc. DAC*, pages 71:1–71:6, 2013.

[173] B. Zhu, R. Chen, X. Zhang, F. Yang, X. Zeng, B. Yu, and M. D. Wong. Hotspot detection via multi-task learning and transformer encoder. In *Proc. ICCAD*, pages 1–8, 2021.

[174] E. Zitzler, D. Brockhoff, and L. Thiele. The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary multi-criterion optimization*, pages 862–876, 2007.

[175] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel. Active learning for multi-objective optimization. In *Proc. ICML*, pages 462–470, 2013.