FIST: A Feature-Importance Sampling and Tree-Based Method for Automatic Design Flow Parameter Tuning

Zhiyao Xie*, Guan-Qi Fang[+], Yu-Hung Huang[+], Haoxing Ren[±], Yanqing Zhang[±], Brucek Khailany[±], Shao-Yun Fang[+], Jiang Hu[¥], Yiran Chen*, Erick Carvajal Barboza[¥]

Duke University*, National Taiwan University of Science and Technology[+], Nvidia[±], Texas A&M University [¥]

Outline

- Introduction
- Method
- Results
- Discussion
- Application
- Conclusion

Introduction: Design Flow Parameter Tuning

- Design parameters have strong impact on overall design quality
- Manual parameter tuning process can be very time-consuming
  - Industrial design flows can take several hours or days to run
  - Huge search space
- Design space exploration (DSE) vs design flow parameter tuning
  - DSE includes high-level synthesis design space exploration
  - Design flow parameter tuning is more about logic synthesis & physical design
  - Design flows take much longer to run & have more parameters
  - Design flow parameter tuning has more prior data (other designs)

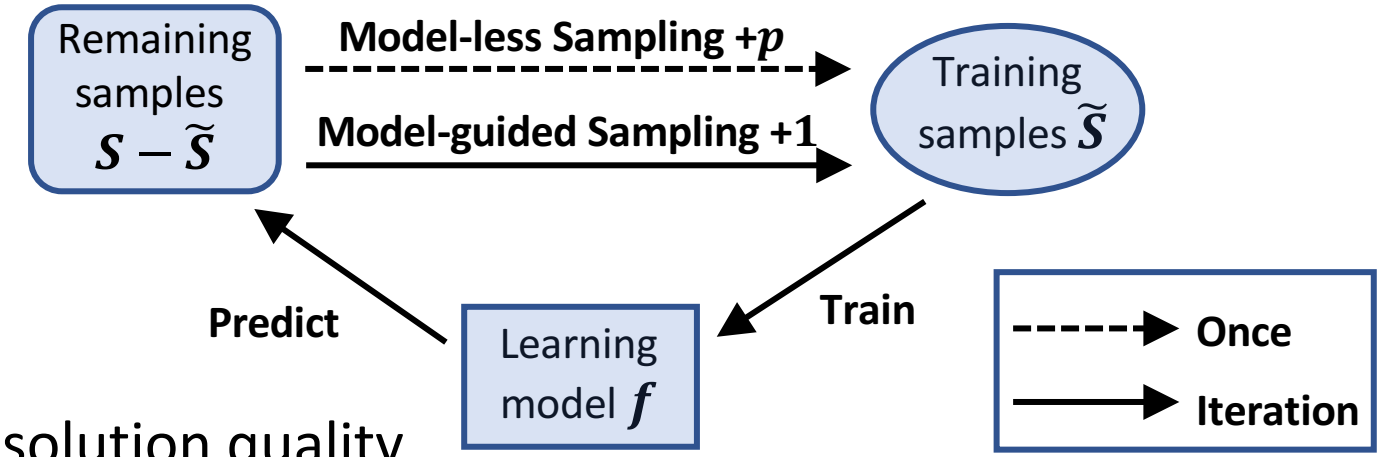
Introduction: Definitions

- ***parameters*** or ***features:*** parameters tuned in logic synthesis or physical design scripts
- **sample (s)**: one combination of parameter values
- **solution quality** or **label**: sample's PPA measured after the complete flow
- **parameter space (S)**: some / all parameter value combinations
- **sampling**: select samples from parameter space (to run design flow)

$$|S| = \prod_{i=1}^{c} n_i$$

Introduction: An Highly Cited Work in DSE

- Idea: iterative refinement
- 1. Model-less Sampling
  - Select **p** samples
  - Run these **p** design flows, get their solution quality
  - Train the model **f** with such **p** samples
- 2. Model-guided Sampling (Refinement)
  - Use **f** to select the most promising unselected sample **s**
  - Run design flow **s**, get its solution quality
  - Add **s** to existing dataset $\tilde{S}$, retrain model **f**



Liu et al., 13' DAC *

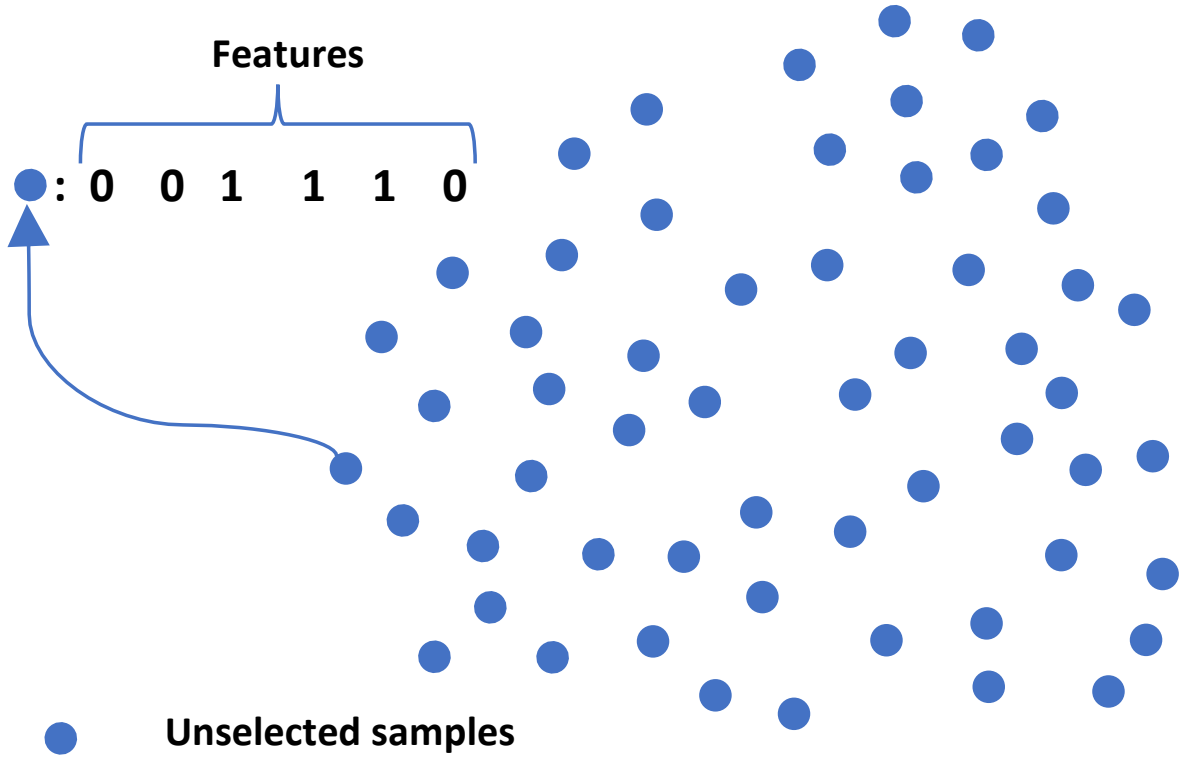
Iterate

* H.-Y. Liu and L. P. Carloni, "On learning-based methods for design space exploration with high-level synthesis," in *DAC*, 2013.

Method: Ideas that Motivate FIST

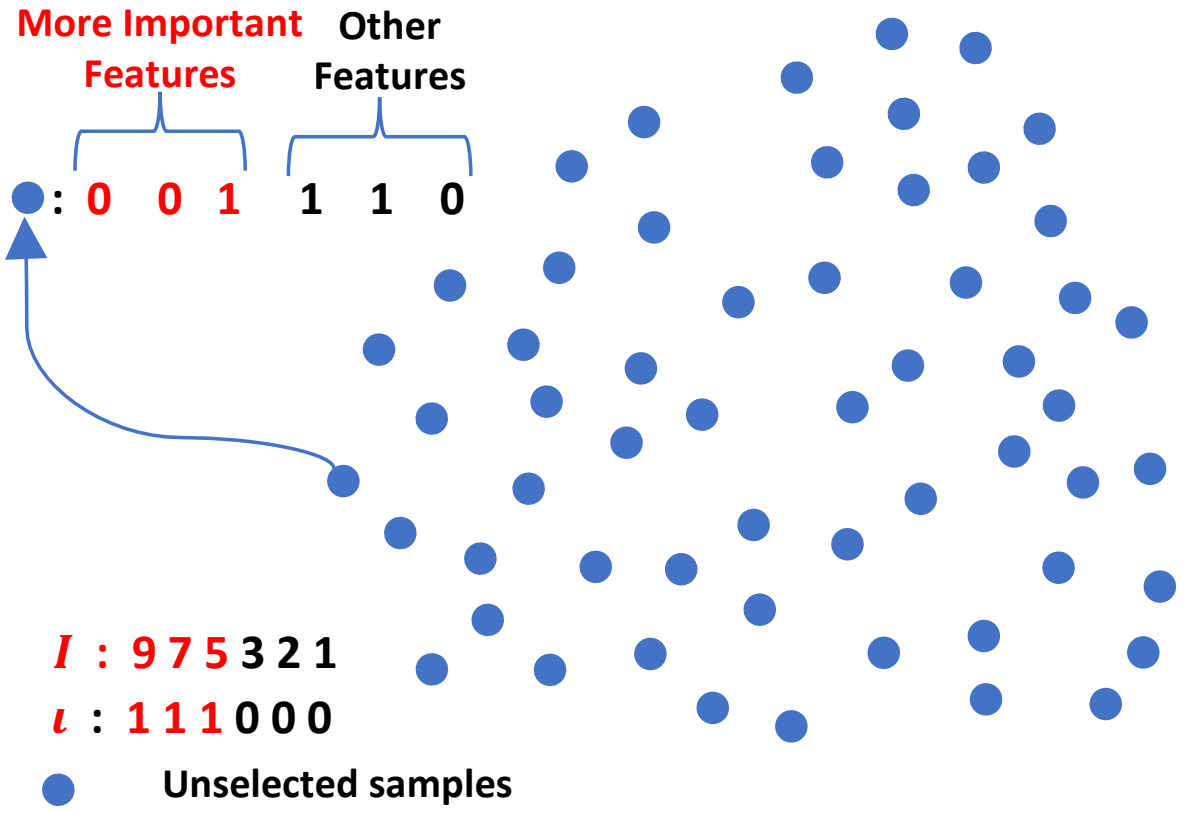
- Learn previous knowledge from already synthesized designs
- Knowledge: parameters' impact (importance) on solution quality, similar across different designs
- Samples with the same values on important features have similar final solution quality. (Put them into the same cluster)
- Size of training dataset increases through iteration

Method: An Example of FIST



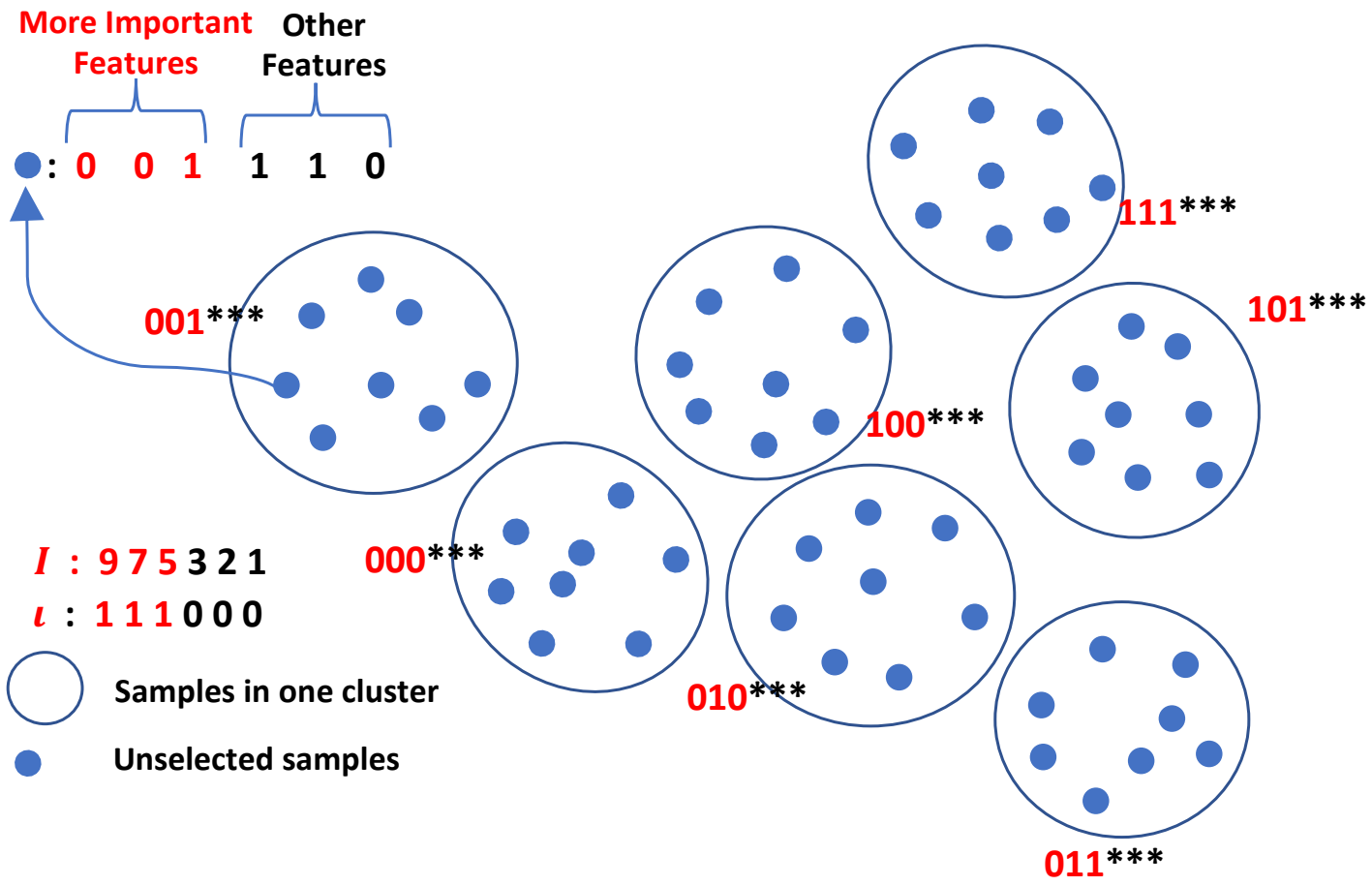
Method: An Example of FIST

- Calculate feature importance



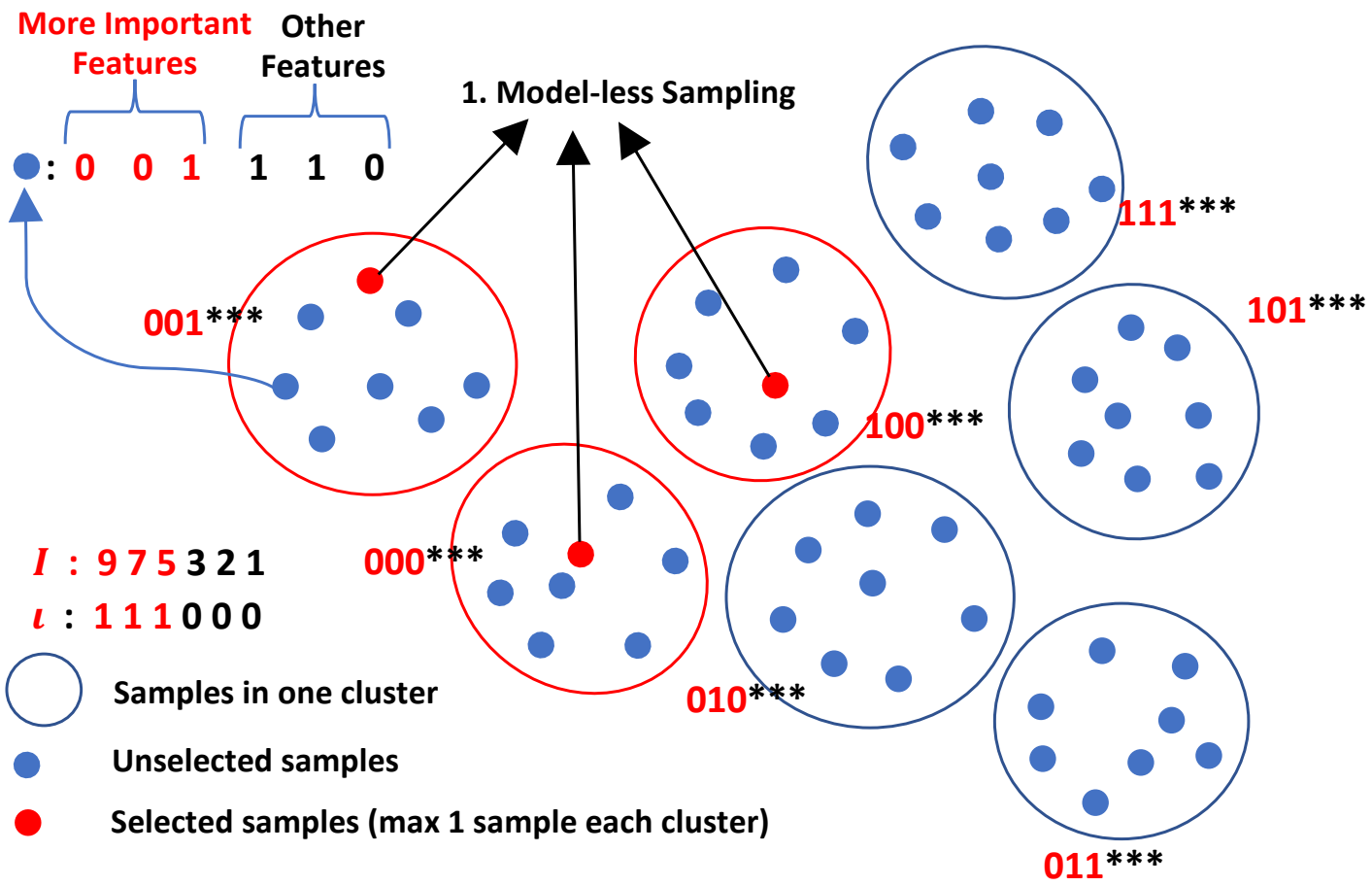
Method: An Example of FIST

- Calculate feature importance
- Cluster samples with the same values on important features
- Assume the same solution quality in the whole cluster



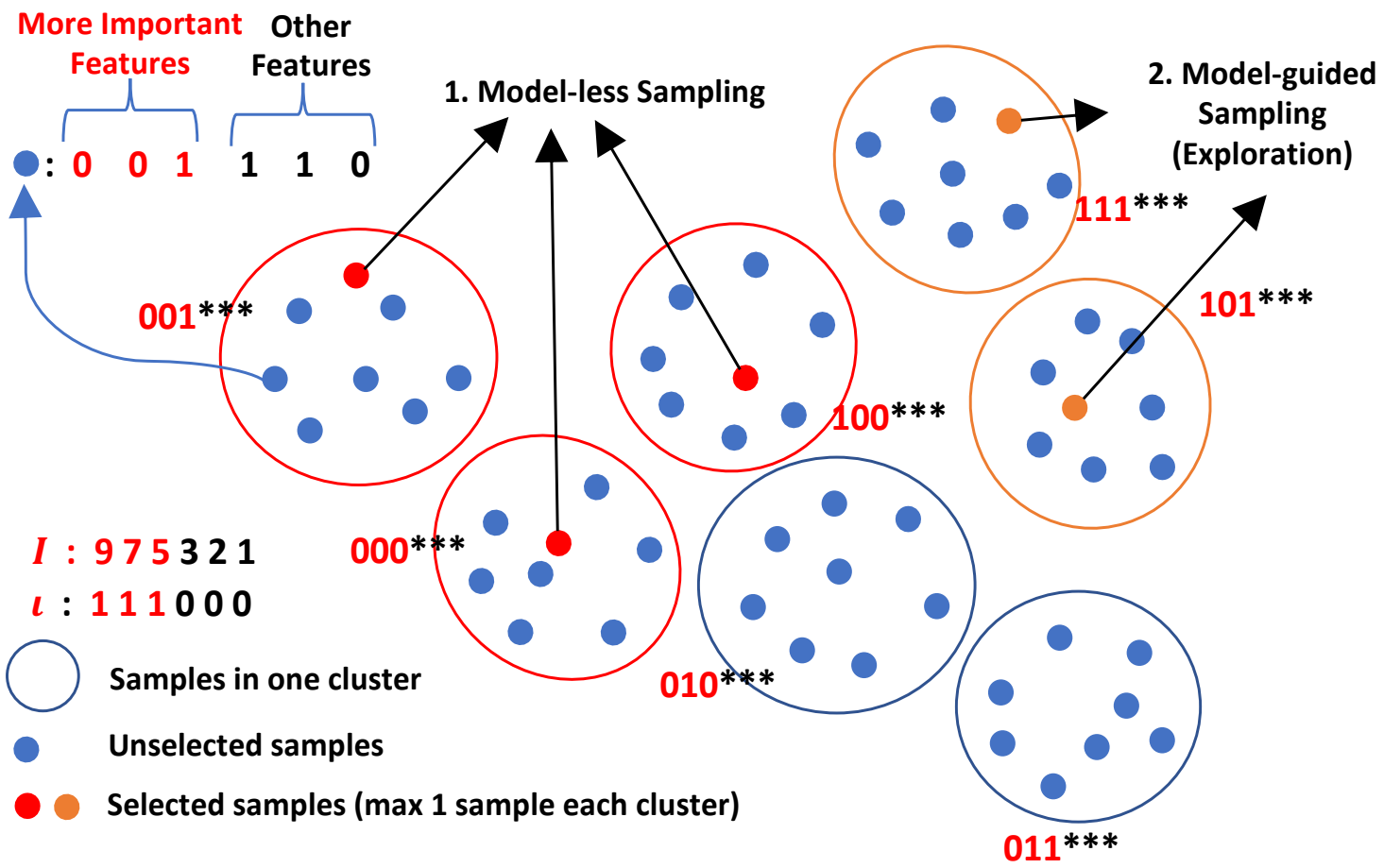
Method: An Example of FIST

- Calculate feature importance
- Cluster samples with the same values on important features
- Assume the same solution quality in the whole cluster
- Select one sample, get solution quality, then put the whole cluster into the training set



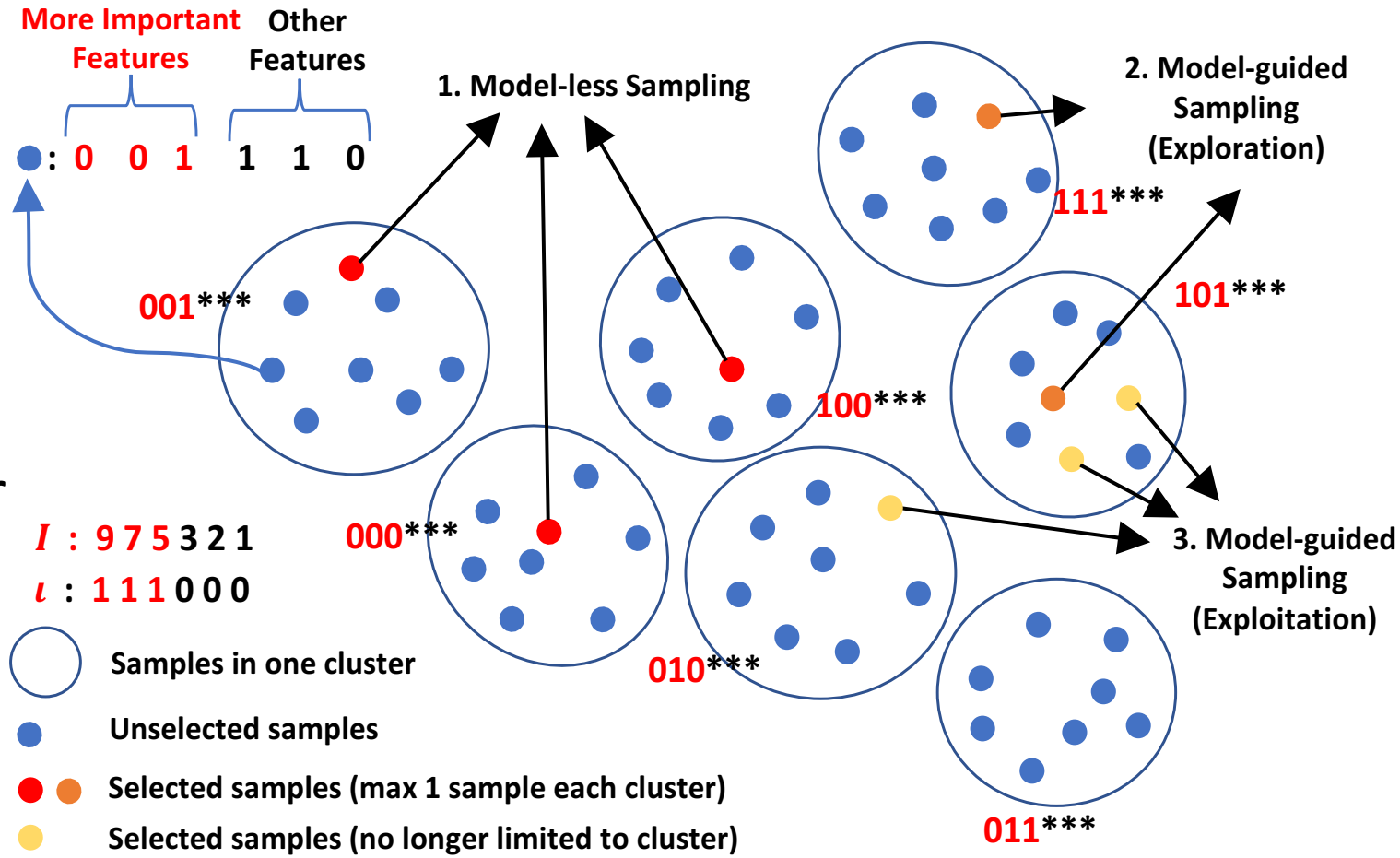
Method: An Example of FIST

- Calculate feature importance
- Cluster samples with the same values on important features
- Assume the same solution quality in the whole cluster
- Select one sample, get solution quality, then put the whole cluster into the training set



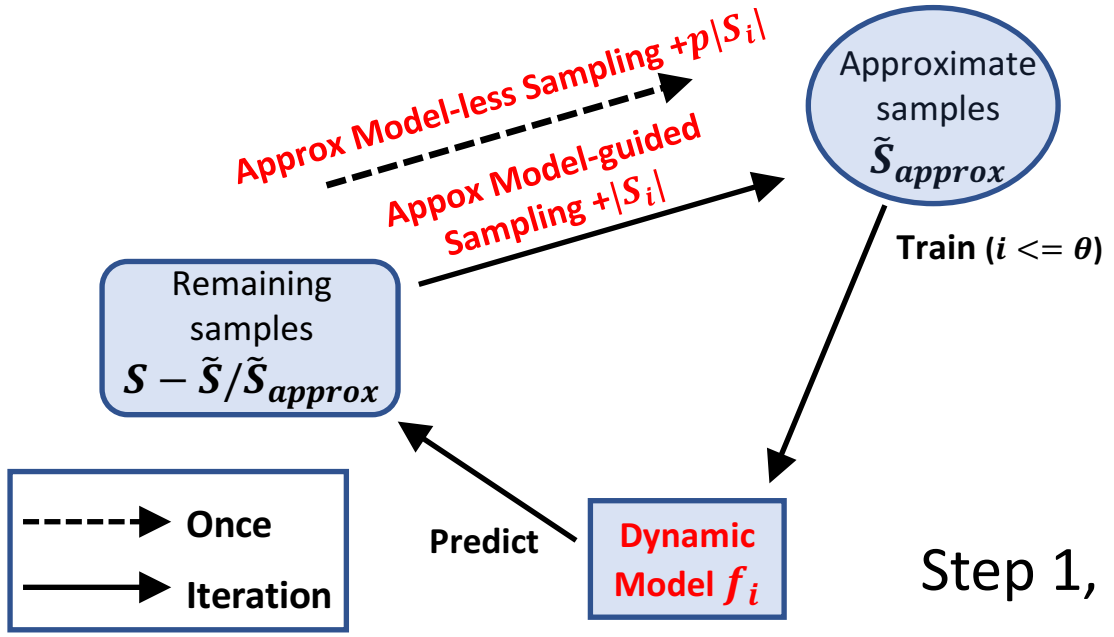
Method: An Example of FIST

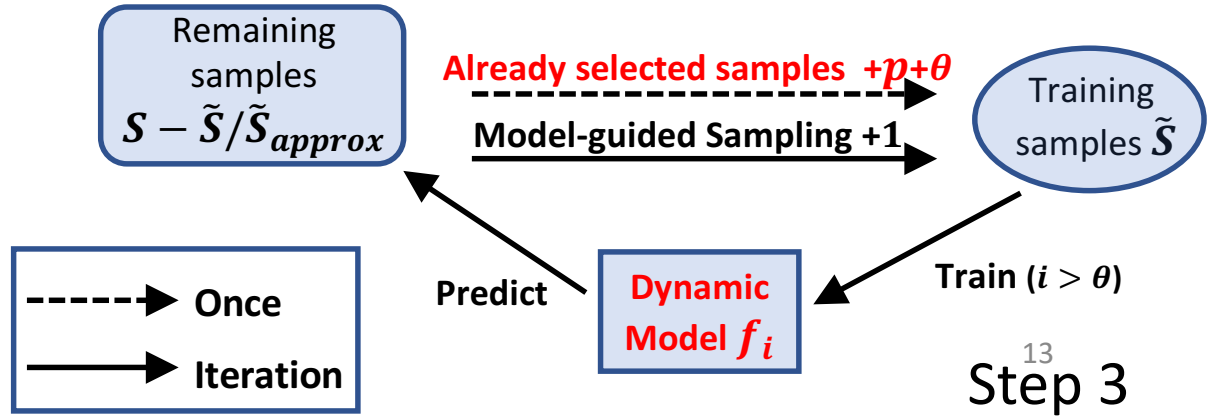
- Calculate feature importance
- Cluster samples with the same values on important features
- Assume the same solution quality in the whole cluster
- Select one sample, get solution quality, then put the whole cluster into the training set
- Benefit: sample more data at the beginning, avoid selecting similar samples



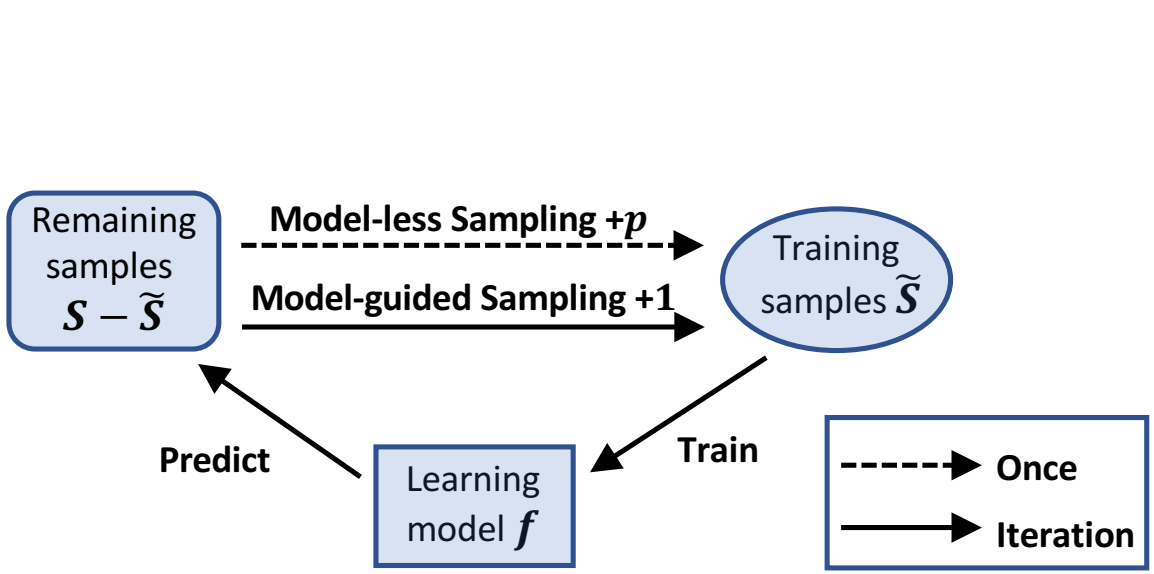
Method: FIST Summary

- 1. Model-less Sampling
  - Select **p cluster** of samples
- Model-guided Sampling (refinement)
  - 2. Exploration (i < θ)
    - Keep selecting by clusters
  - 3. Exploitation (i > θ)
    - Now select by sample
- ML model:
  - Use XGBoost instead of Random Forest
  - Increase max tree depth (model complexity) during iteration process

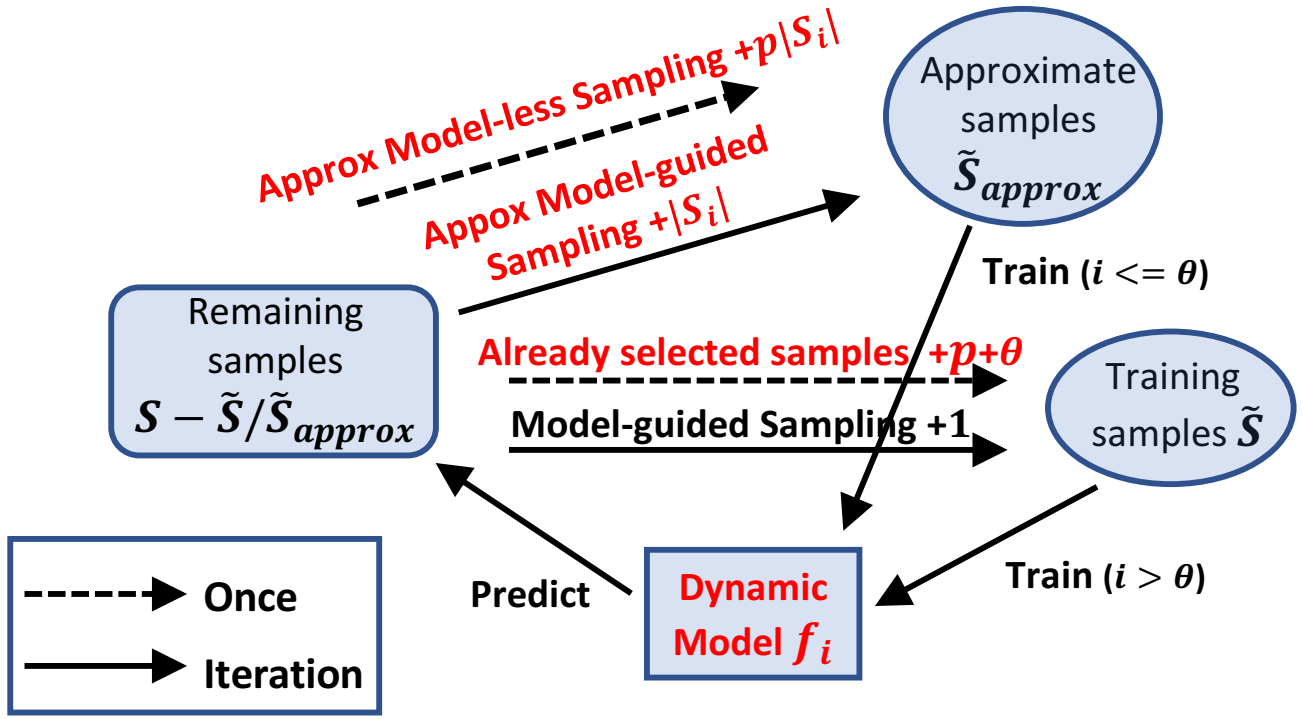




Method: FIST Compared with Previous Works



Previous DSE method



FIST

Cluster samples based on the values of important features

In initial stages i < θ, we select whole clusters instead of one sample

Result: Experiment Setup

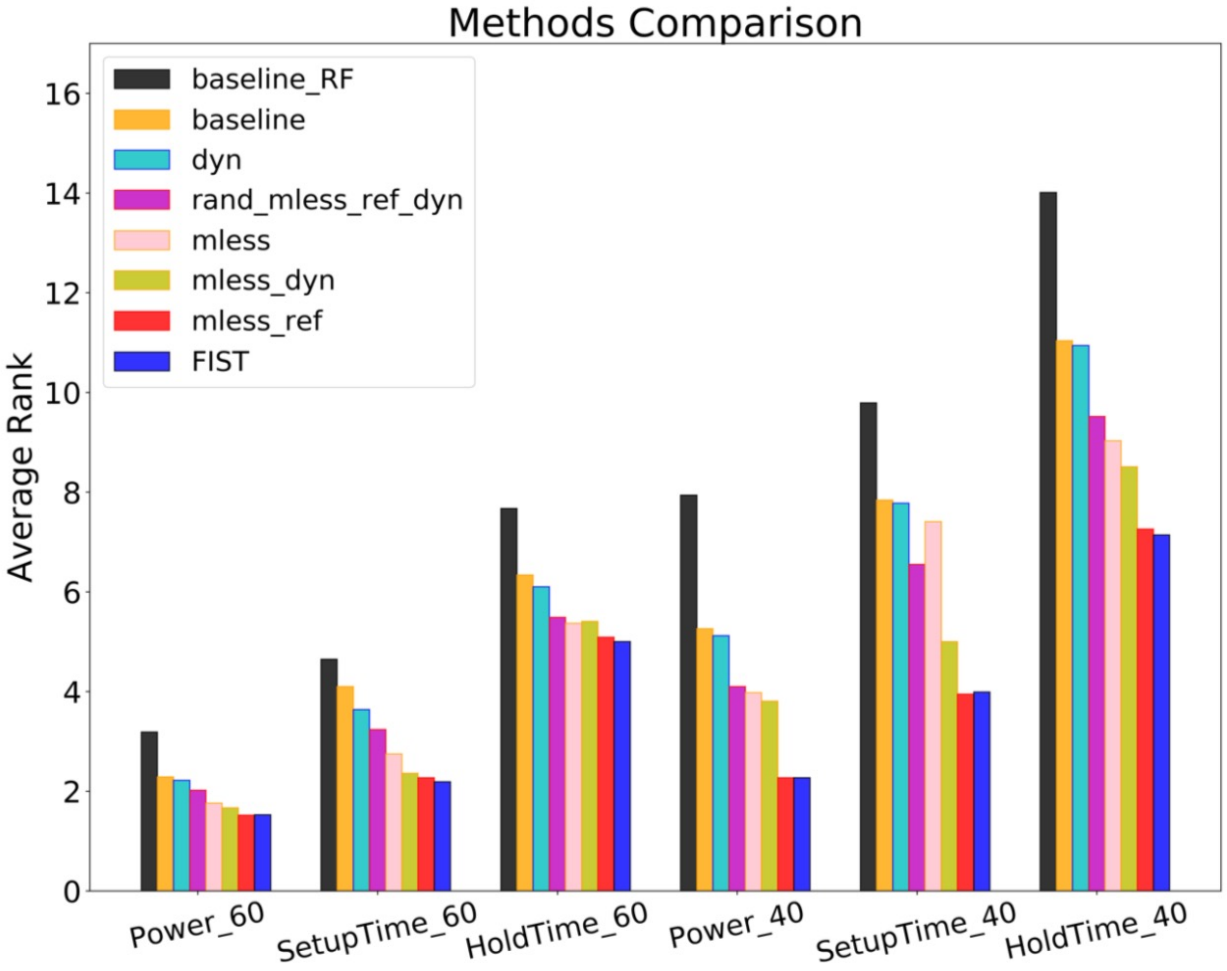
- Nine designs from ITC 99
- 9 logic synthesis parameters
- For each design, exhaustively run 1728 samples in their parameter space

Synthesis parameters include:

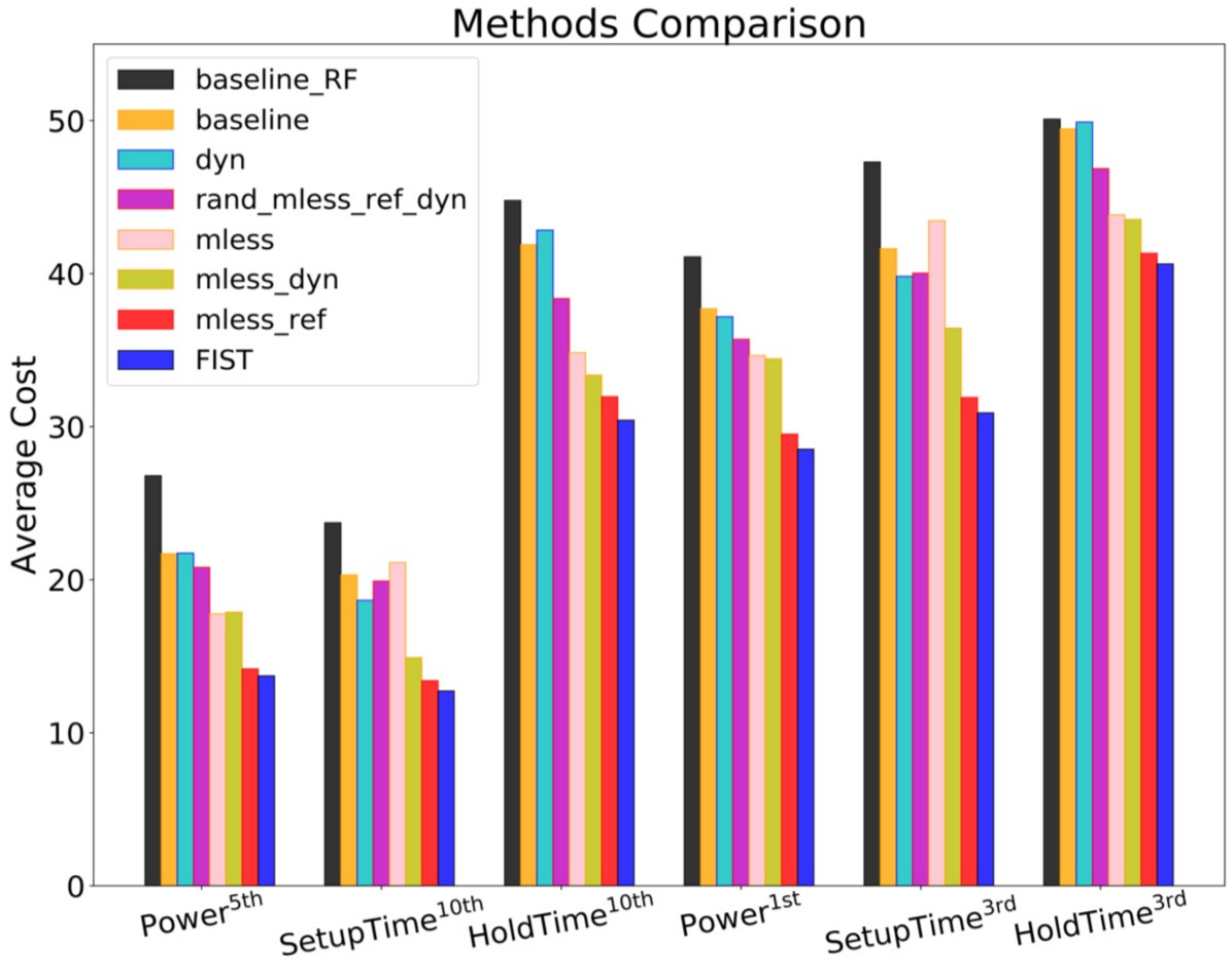
set max fanout, set max transition, set max area, high fanout net threshold, set max capacitance, insert clock gating, leakage power optimization, compile type, dynamic power optimization.

Denotation	Methods
baseline_RF	random sampling & Random Forest model
baseline	random sampling & XGBoost (same below)
dyn	dynamic-depth tree model
mless	model-less sampling by clustering
ref	model-guided sampling in refinement by clustering

Result: Single-Objective Tuning

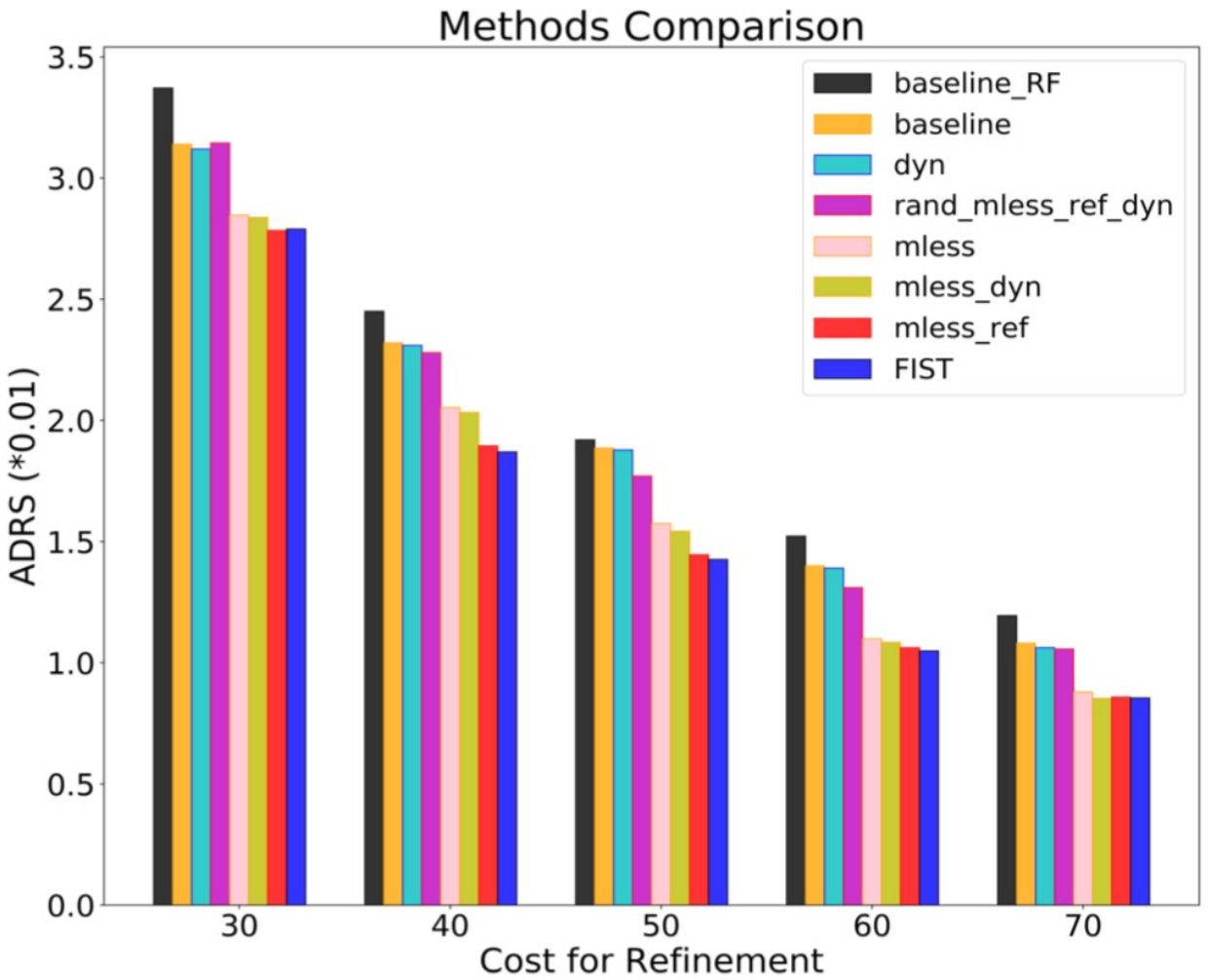


Best solution rank with the same sample cost

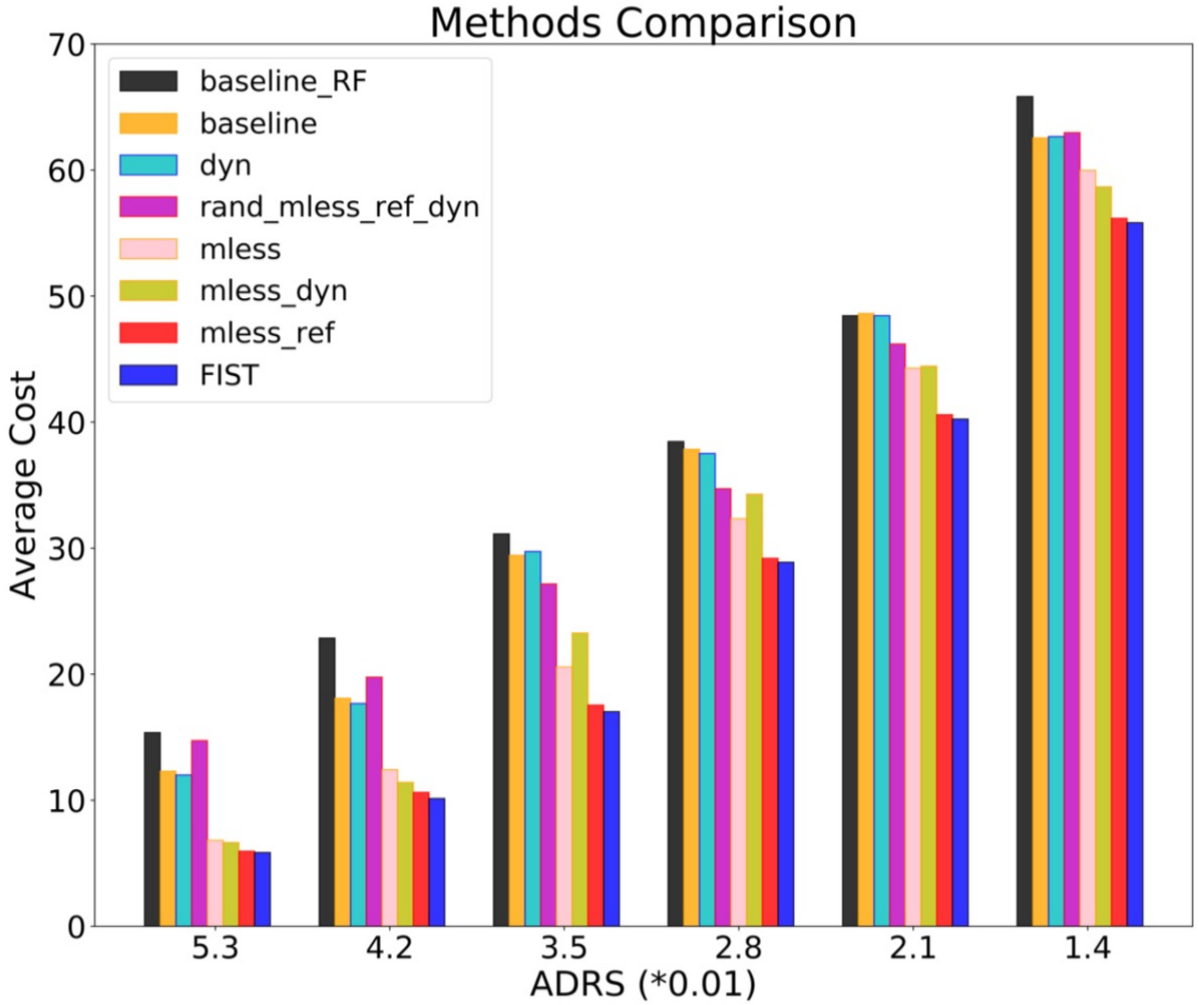


Sample cost to reach the same solution rank

Result: Multiple-Objectives Tuning



Best ADRS with the same sample cost.



Sample cost to reach the same ADRS

ADRS (Average Distance from Reference Set) considers multiple design objectives

Discussion: Similarity in Design Quality

- in-cluster sampling has much lower σ -> design quality of samples from the same cluster are indeed similar
- learned σ is close to ground truth -> the learned feature importance is accurate

Standard deviation σ of solution qualities among samples

	Power	SetupTime	HoldTime
Random sampling	3.35	0.377	0.288
In-cluster sampling (learn)	0.49	0.152	0.175
Cross-cluster sampling (learn)	3.42	0.384	0.282
In-cluster sampling (true)	0.54	0.135	0.146
Cross-cluster sampling (true)	3.39	0.383	0.294

Application: Setup

- FIST-based automatic parameter tuning flow for industrial designs
- Modules from a 16nm deep learning inference accelerator

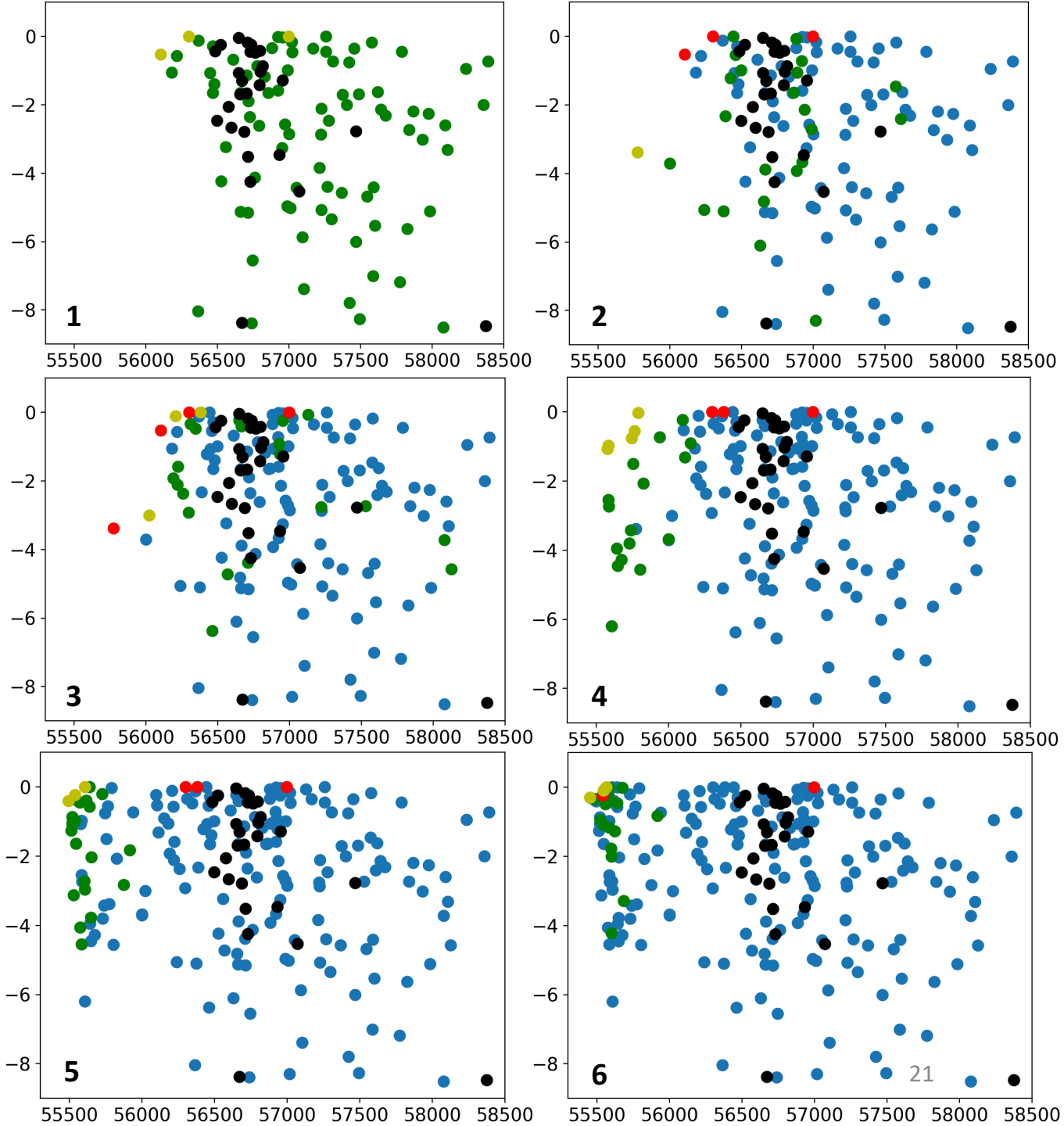
Physical design parameter	Settings
postroute iterations	0, 1, 2, 3, 4
cts.optimize.enable_local_skew	False, True
clock_opt.hold.effort	low, medium, high
postroute (clock_tran_fix)	disable, enable
postroute (useful_skew, timing_opt)	0, 1
useful_skew (power_opt)	0, 1
clock buffer max fanout	22, 36, 48, 96
target skew	0.025, 0.05, 0.1, 0.3
setup uncertainty	-0.025, -0.05, -0.1
hvt cell swap enable during leakage optimization	0, 1
extra hold uncertainty for SRAM macro	-0.025, -0.05, -0.1, -0.15
max util	0.7, 0.78, 0.85
hold uncertainty	-0.002, -0.005, -0.008 -0.01, -0.012

Application: Setup

- Module PE: a 71K-gate Processing Element
- Module RISC-V: a 117K-gate RISC-V microprocessor
- Parameter space consists of 1,382,400 samples
- Impossible to collect data exhaustively like ITC99, thus compare with 30 hand-tuned solutions from experienced designers
- Budget b ~= 200, <**0.02%** parameter space. Initial sampling p = 100
- Although taking more trials, FIST is fully automatic

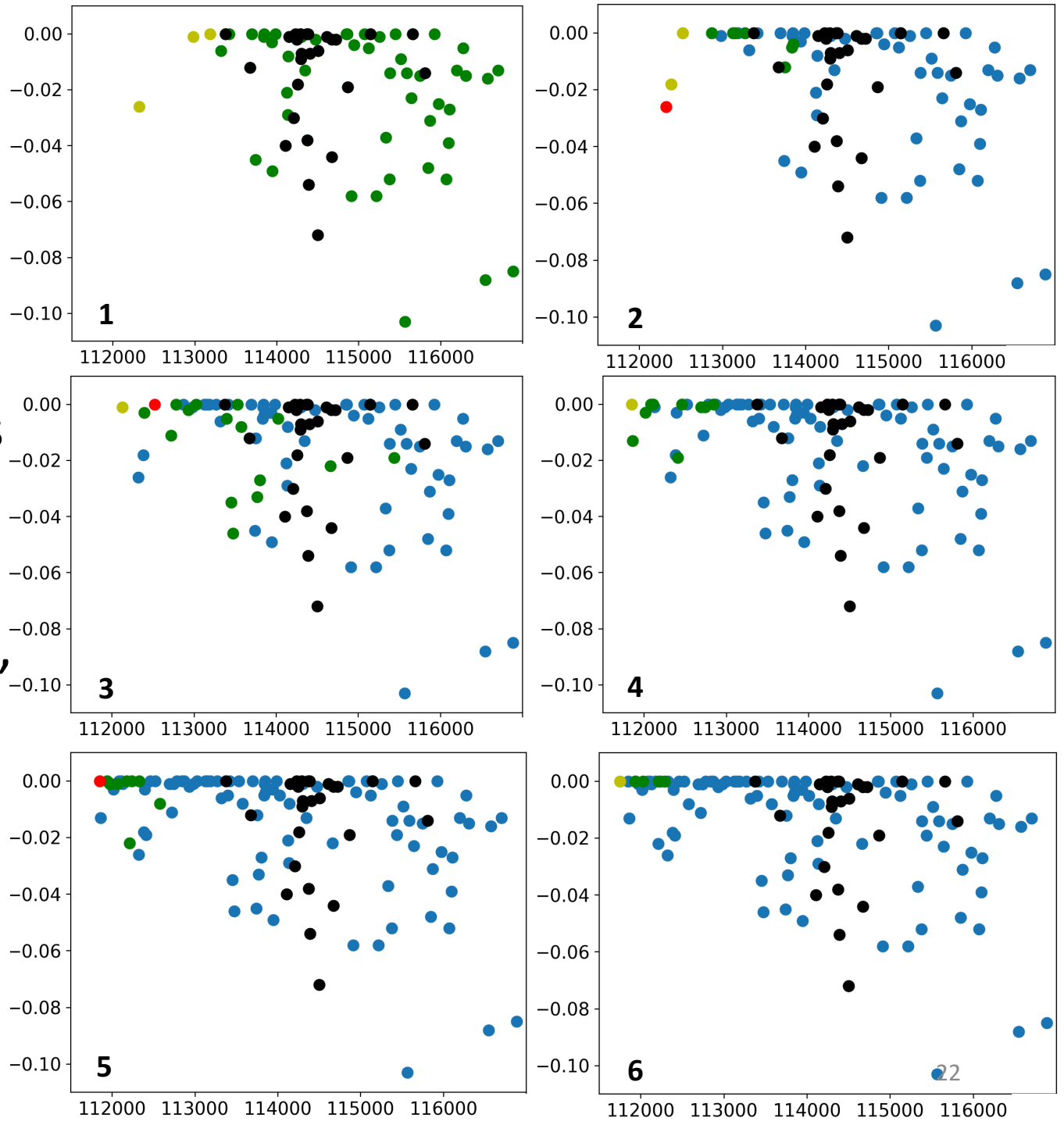
- Parameter tuning process in six stages on PE
- Area (µm2) vs setup TNS (ns)
- **Black** are baselines from human
- Red and yellow are Pareto points
- Step 1 perform model-less samp
- Step 2, 3 perform ‘exploration’
- Step 4, 5, 6 perform ‘exploitation’

- 1.82% improvement in area
- All samples from 4, 5, 6 outperform human solutions



- Parameter tuning process in six stages on RISC-V
- Area (µm2) vs setup TNS (ns)
- **Black** are baselines from human
- Red and yellow are Pareto points
- Step 1 perform model-less samp
- Step 2, 3 perform ‘exploration’
- Step 4, 5, 6 perform ‘exploitation’

- 1.43% improvement in area
- All samples from 4, 5, 6 outperform human solutions



Conclusion

- Proposed an efficient machine learning approach for automatic parameter tuning.
- Leveraged prior knowledge from other already explored designs.
- Introduced approximate (cluster) sampling, similar to ideas in semi-supervised learning.
- Incorporated FIST to the automatic tuning process of two complicated industrial designs with a huge parameter space.

Thanks!