STANFORD DAWN

# Design Space Exploration

## - and Its Application to Computer Hardware Engineering -

**Luigi Nardi, Ph.D.**
**Stanford University**

@Google, July 30, 2019

# Main Papers Behind This Talk

1. Nardi, et al., "Practical Design Space Exploration", MASCOTS, 2019.

2. Koeplinger, et al., "Spatial: A Language and Compiler for Application Accelerators", PLDI, 2018.

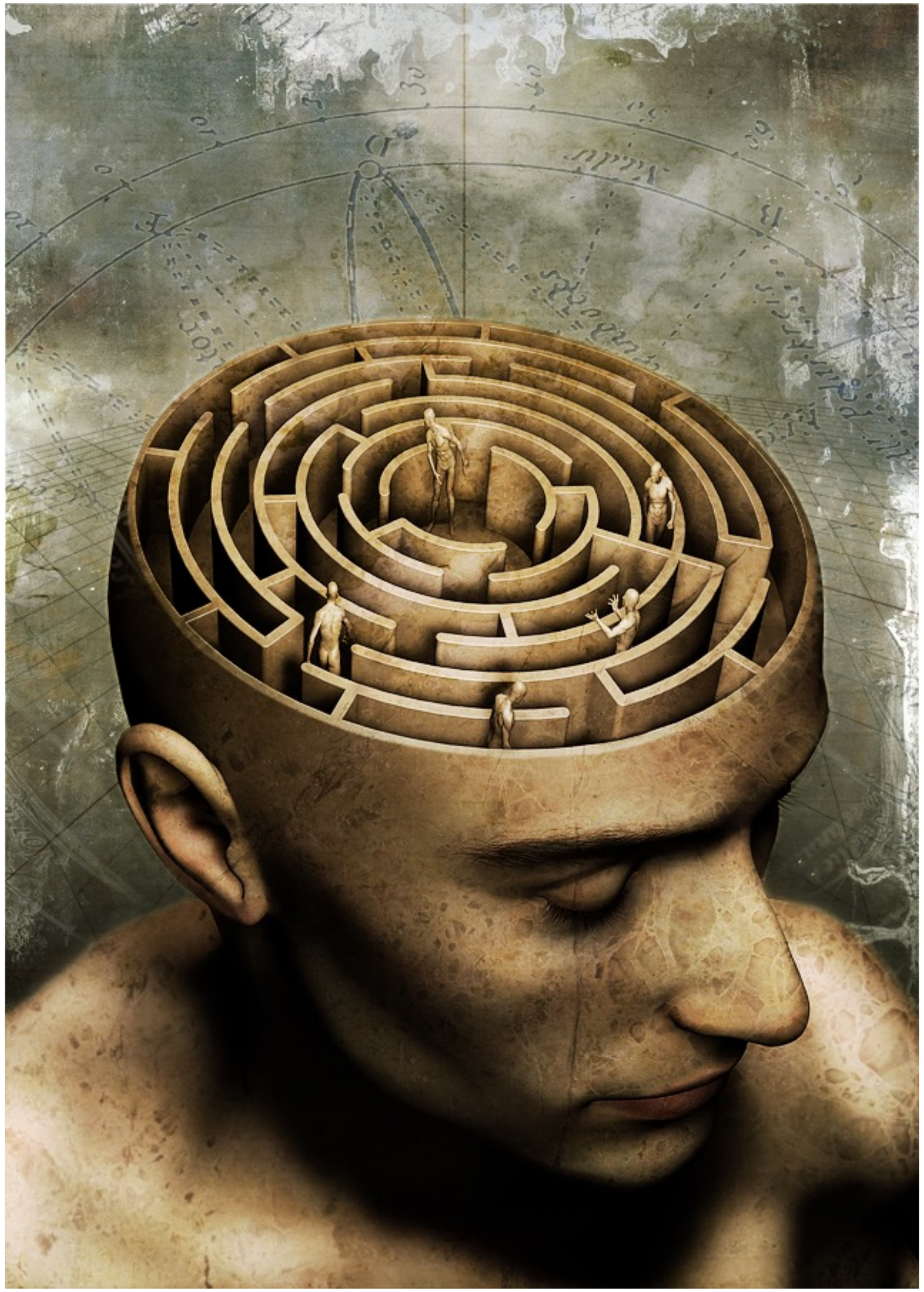
3. Nardi, et al., "Algorithmic performance-accuracy trade-off in 3D vision applications using HyperMapper", iWAPT-IPDPS, 2017.

4. Saeedi, et al., "Application-oriented design space exploration for SLAM algorithms", ICRA, 2017.

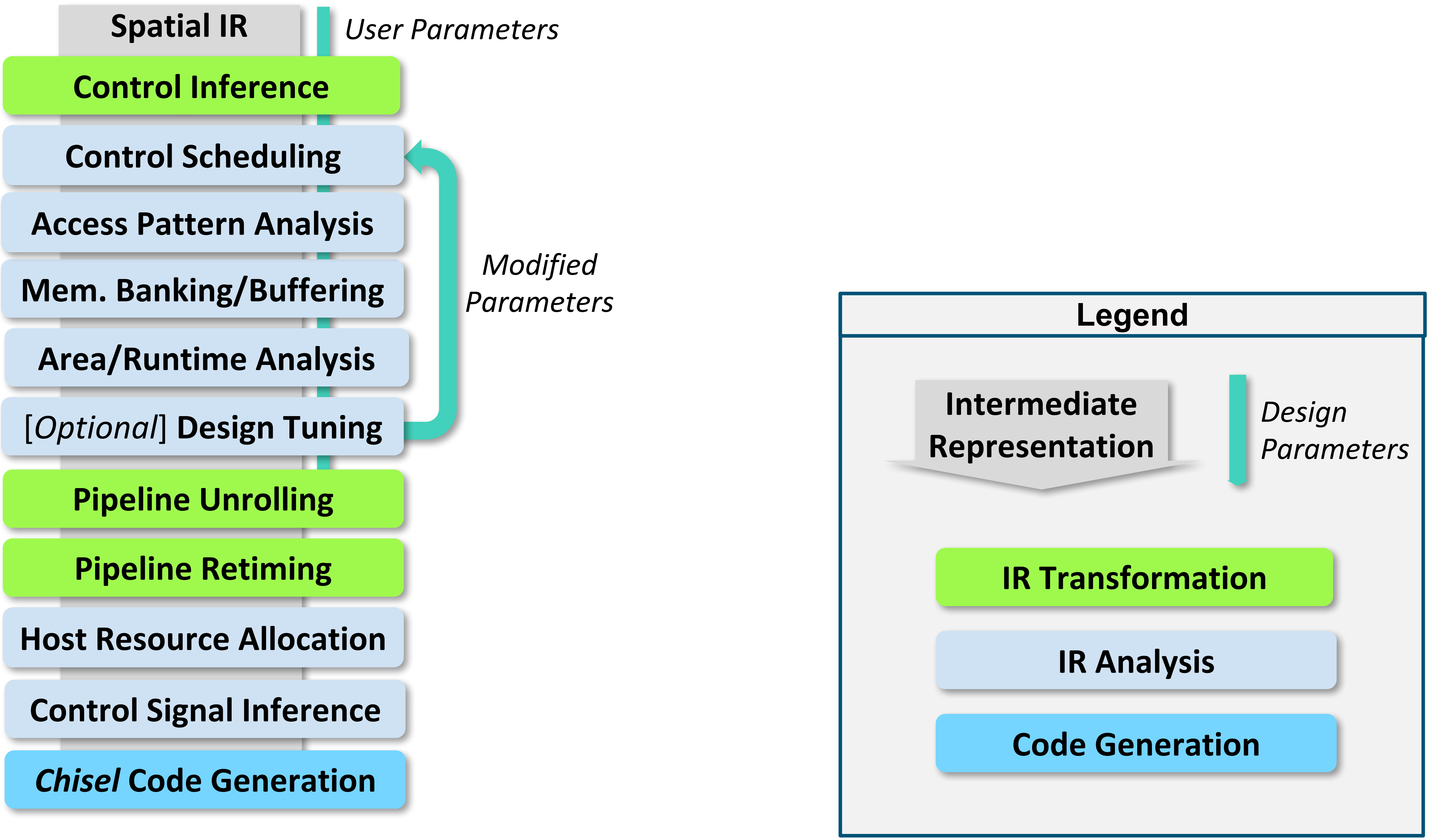
5. Bodin, et al., "Integrating algorithmic parameters into benchmarking and design space exploration in 3D scene understanding", PACT, 2016.

# Outline

1. **Design Space Exploration**
   - **Motivation Example**
   - **Problem Setting**
   - **Important Features**
   - **Taxonomy**
2. The HyperMapper Framework
   - Pareto and Hypervolumes
   - Prior Distribution
   - Pareto-based Active Learning
3. Experimental Results



# The Spatial Compiler



Spatial IR

*User Parameters*

Control Inference

Control Scheduling

Access Pattern Analysis

Mem. Banking/Buffering

Area/Runtime Analysis

[*Optional*] Design Tuning

*Modified Parameters*

Pipeline Unrolling

Pipeline Retiming

Host Resource Allocation

Control Signal Inference

*Chisel* Code Generation

**Legend**

Intermediate Representation

*Design Parameters*

IR Transformation

IR Analysis

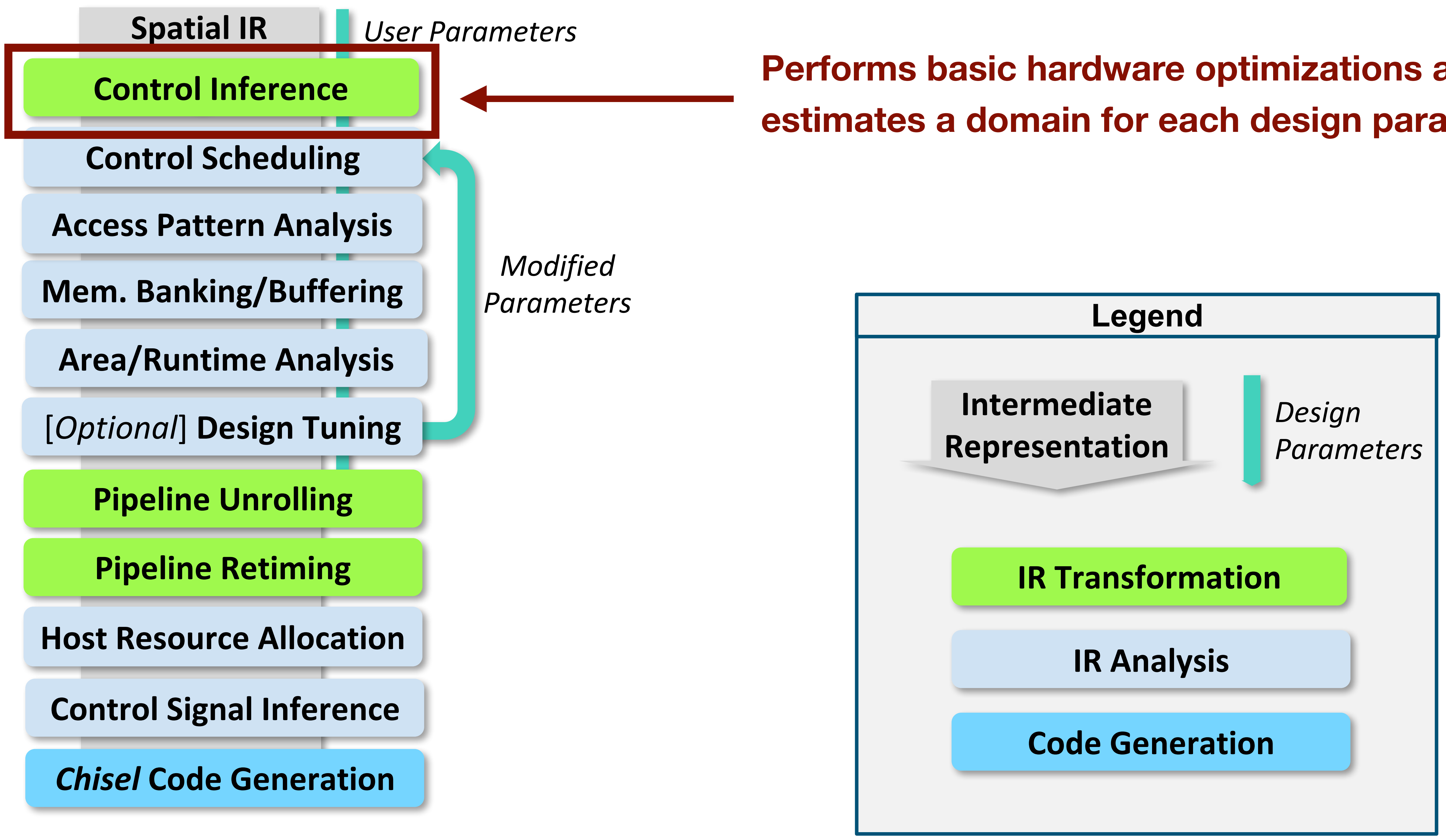
Code Generation

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler



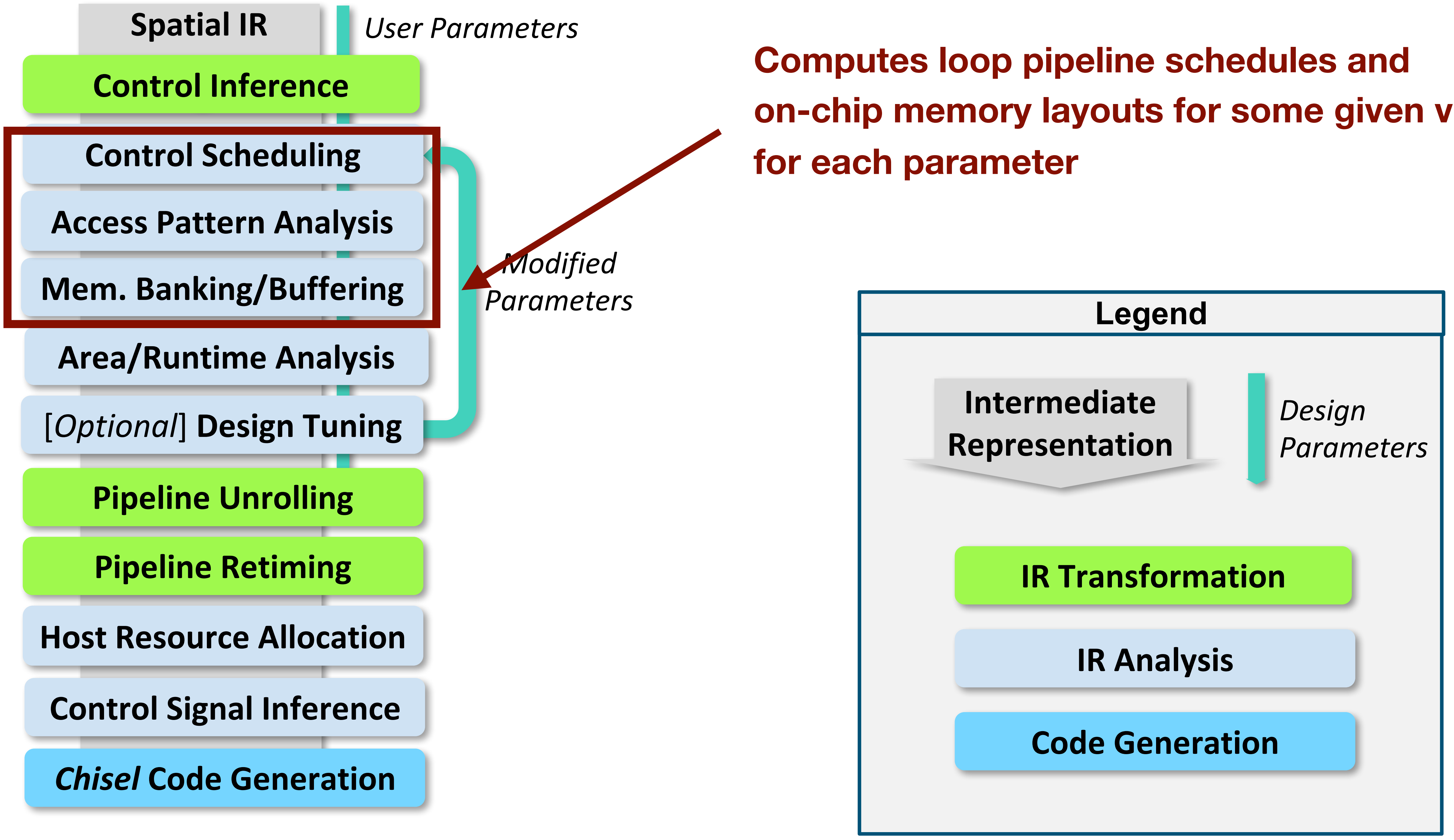
**Performs basic hardware optimizations and estimates a domain for each design parameter**

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler



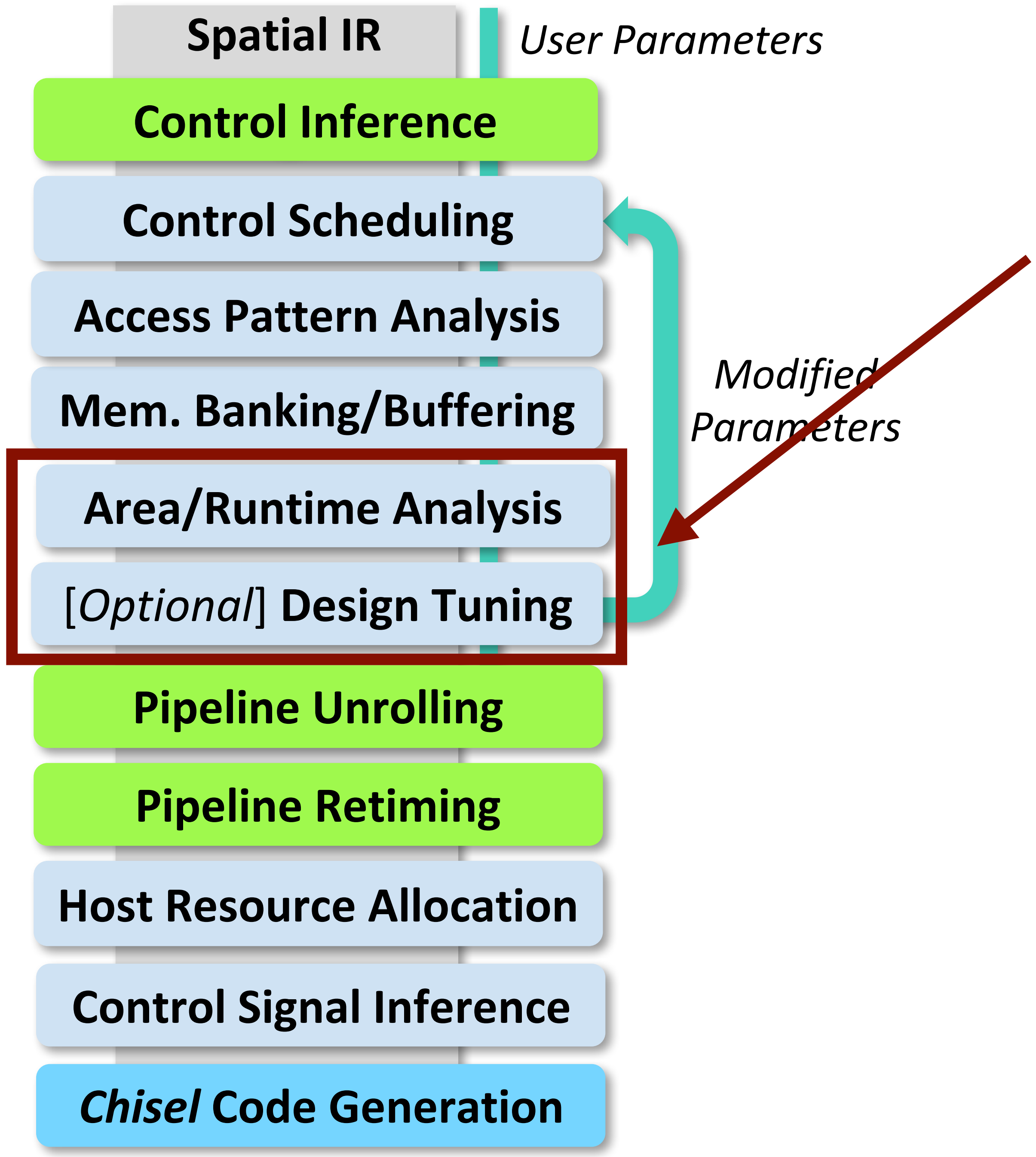
Computes loop pipeline schedules and on-chip memory layouts for some given value for each parameter

- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

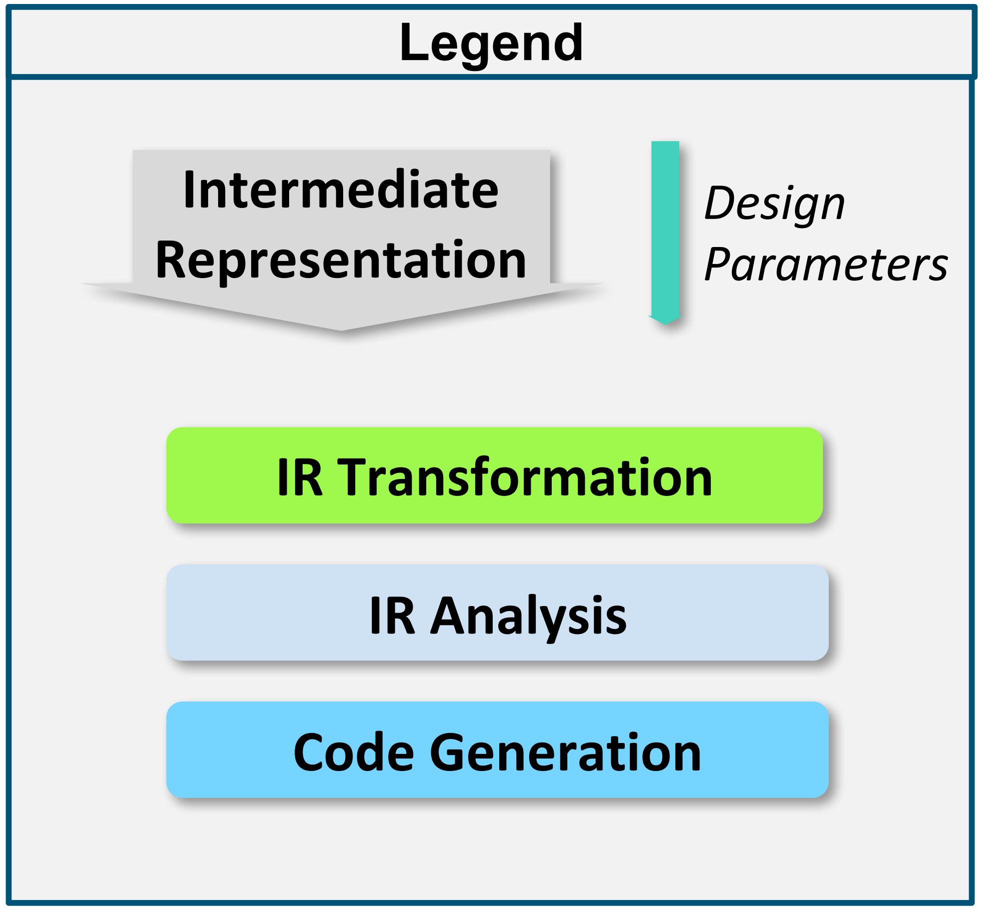
[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler



**Estimates the amount of hardware resources and the runtime of the application**

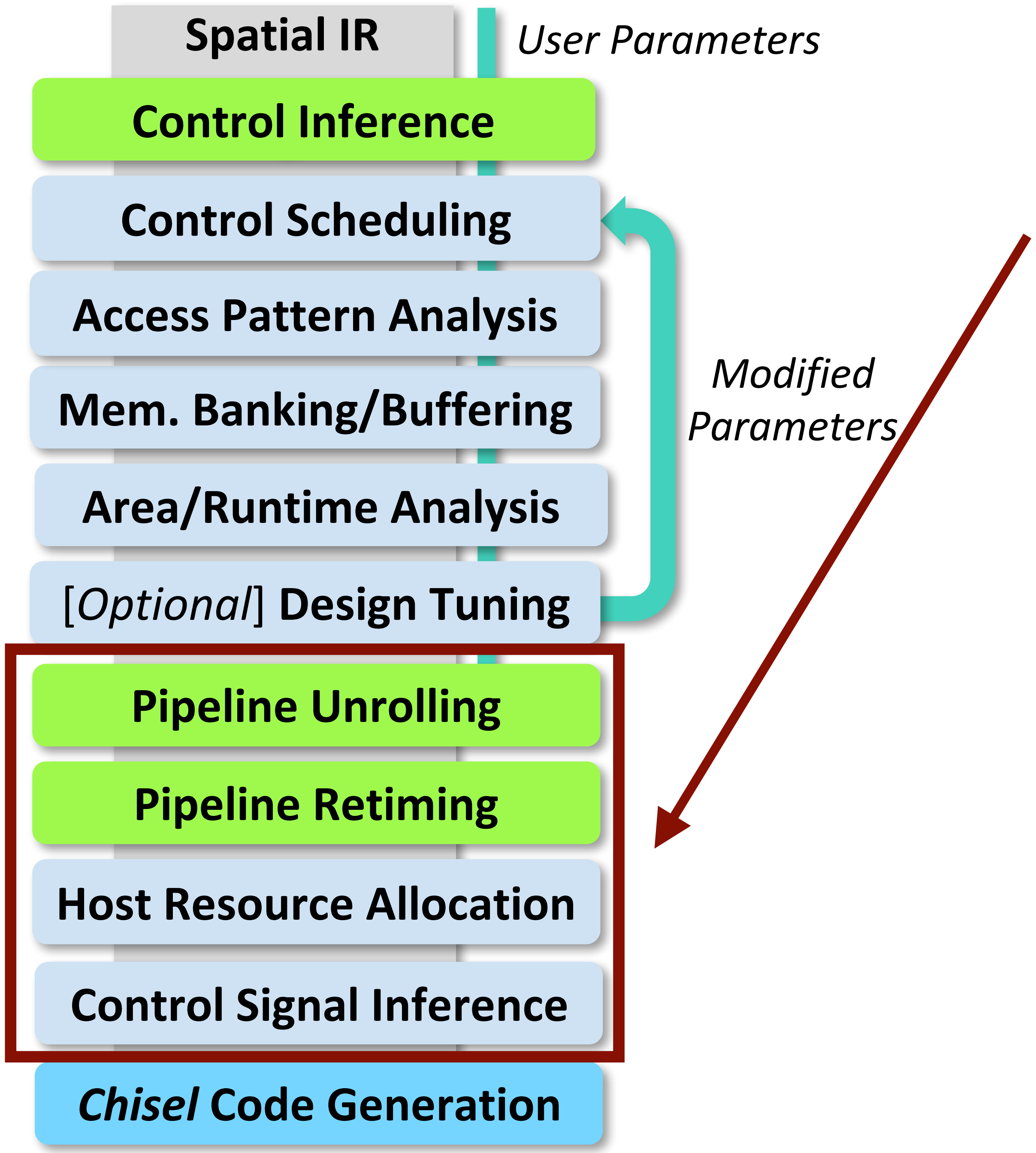


- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

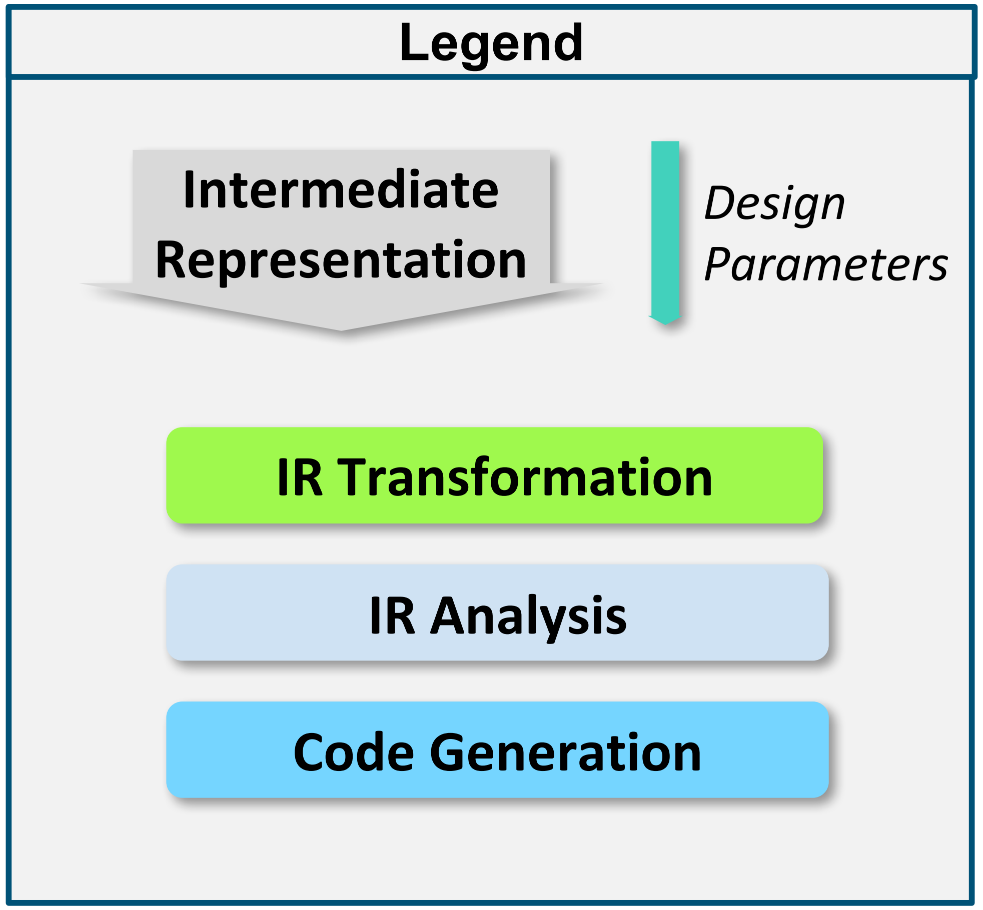
[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler



**Unrolls loops, retimes pipelines, and performs on-chip memory layout.**
**The optimizations are computed based on the analyses of the previous phase**

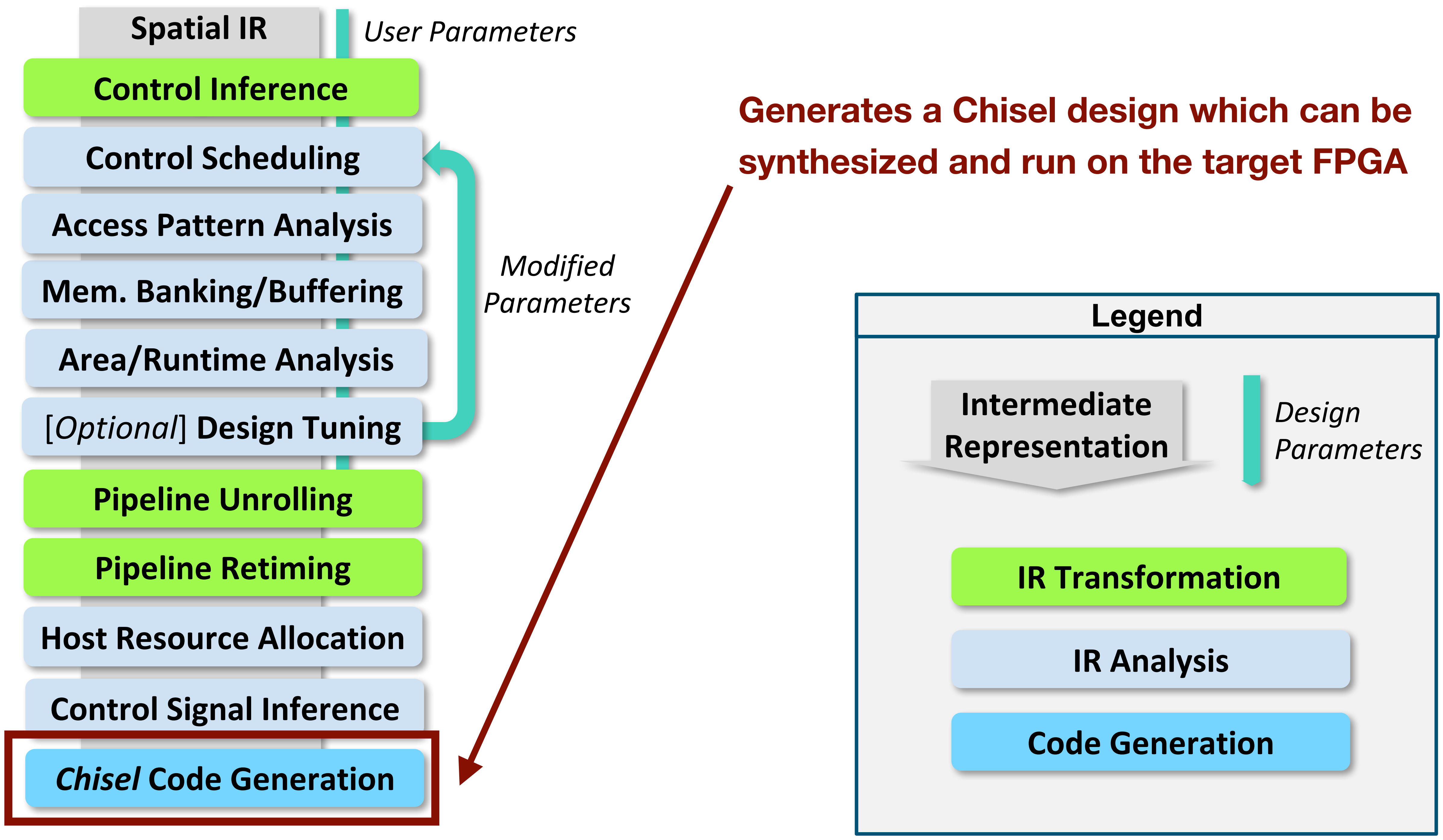


- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler



Spatial IR

*User Parameters*

Control Inference

Control Scheduling

Access Pattern Analysis

Mem. Banking/Buffering

Area/Runtime Analysis

[*Optional*] Design Tuning

*Modified Parameters*

Pipeline Unrolling

Pipeline Retiming

Host Resource Allocation

Control Signal Inference

*Chisel* Code Generation

**Generates a Chisel design which can be synthesized and run on the target FPGA**

**Legend**

Intermediate Representation

*Design Parameters*

IR Transformation

IR Analysis

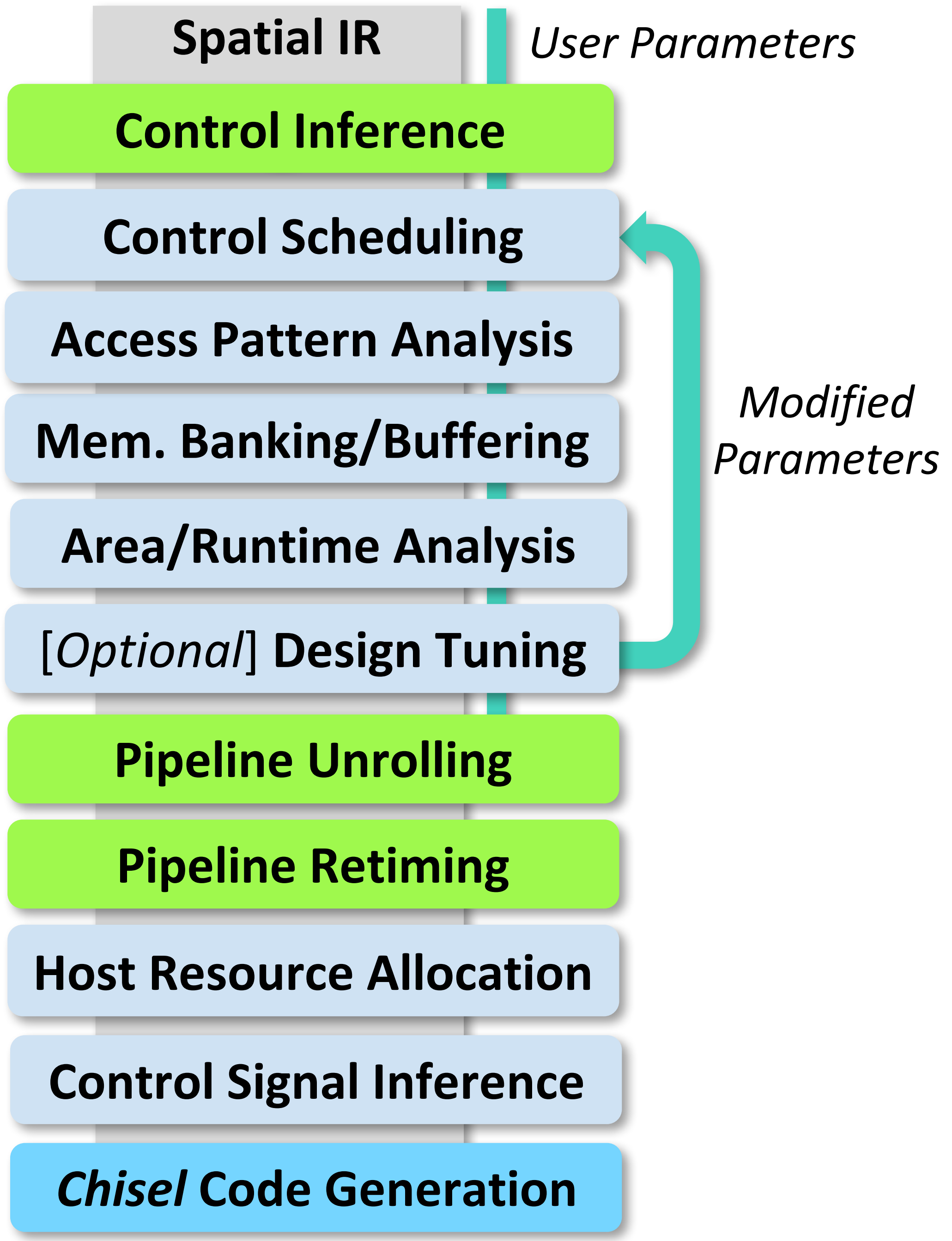
Code Generation

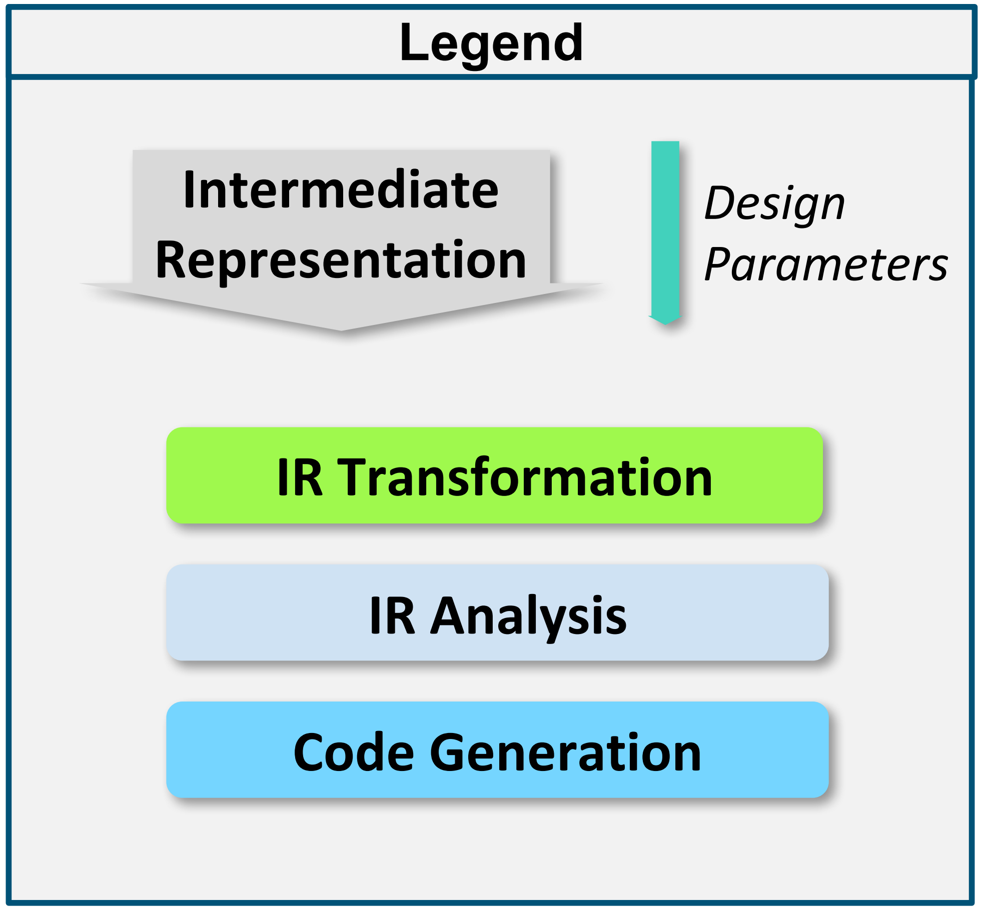
- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler





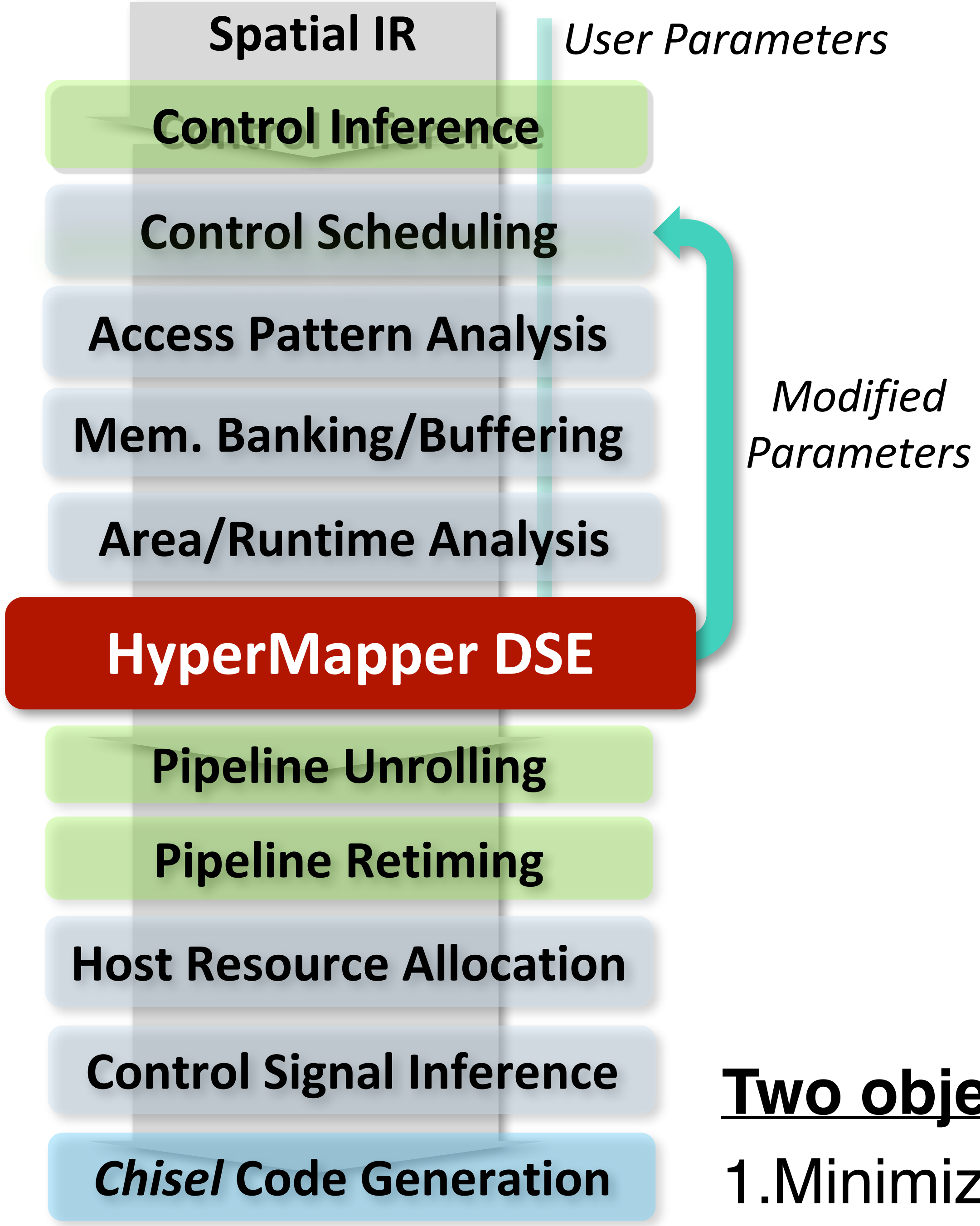
- Goal of Spatial [Koeplinger, et al.]: design of application accelerators
- On reconfigurable architectures FPGAs and CGRAs
- Spatial compiler lowers user programs into synthesizable Chisel [Bachrach, et al.]

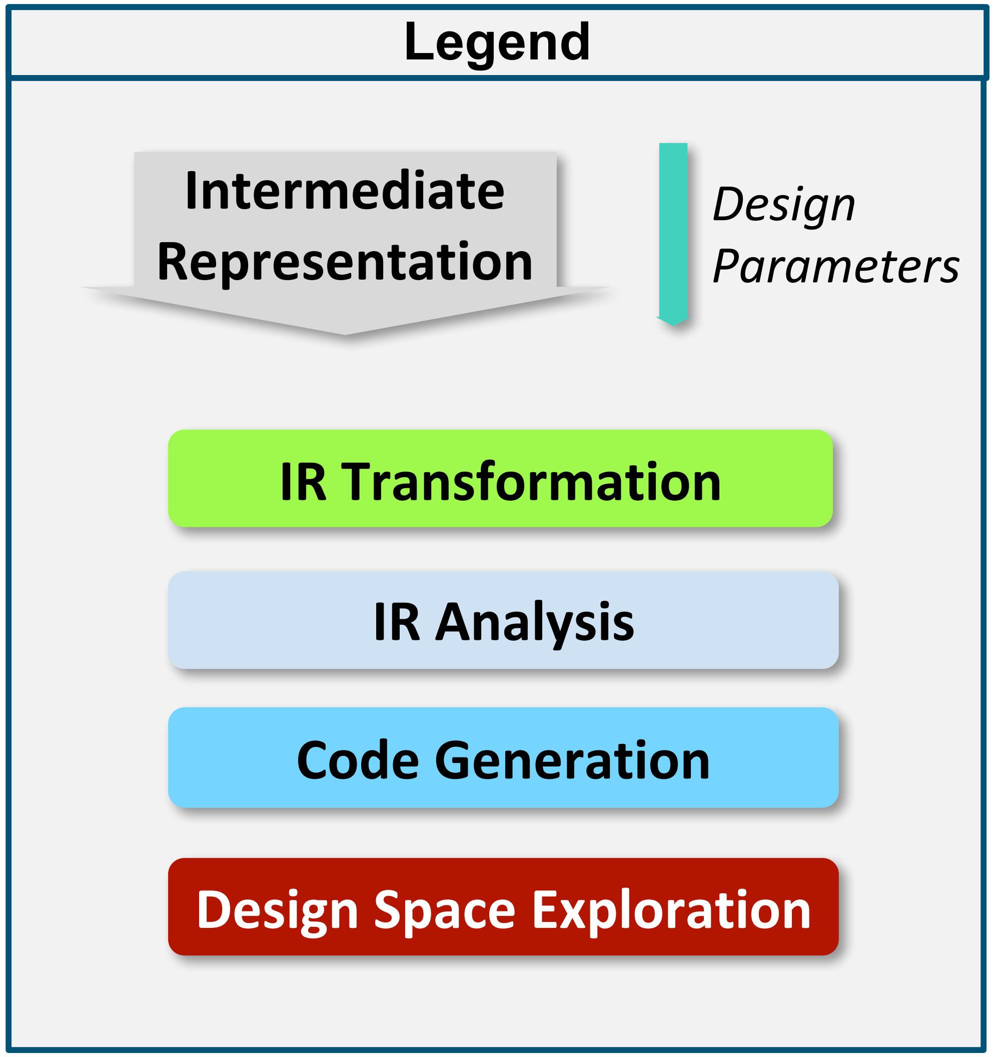
[Koeplinger, et al.] "Spatial: a language and compiler for application accelerators." PLDI 2018

[Bachrach, et al.] "Chisel: constructing hardware in a Scala embedded language." DAC 2012.

# The Spatial Compiler

This talk



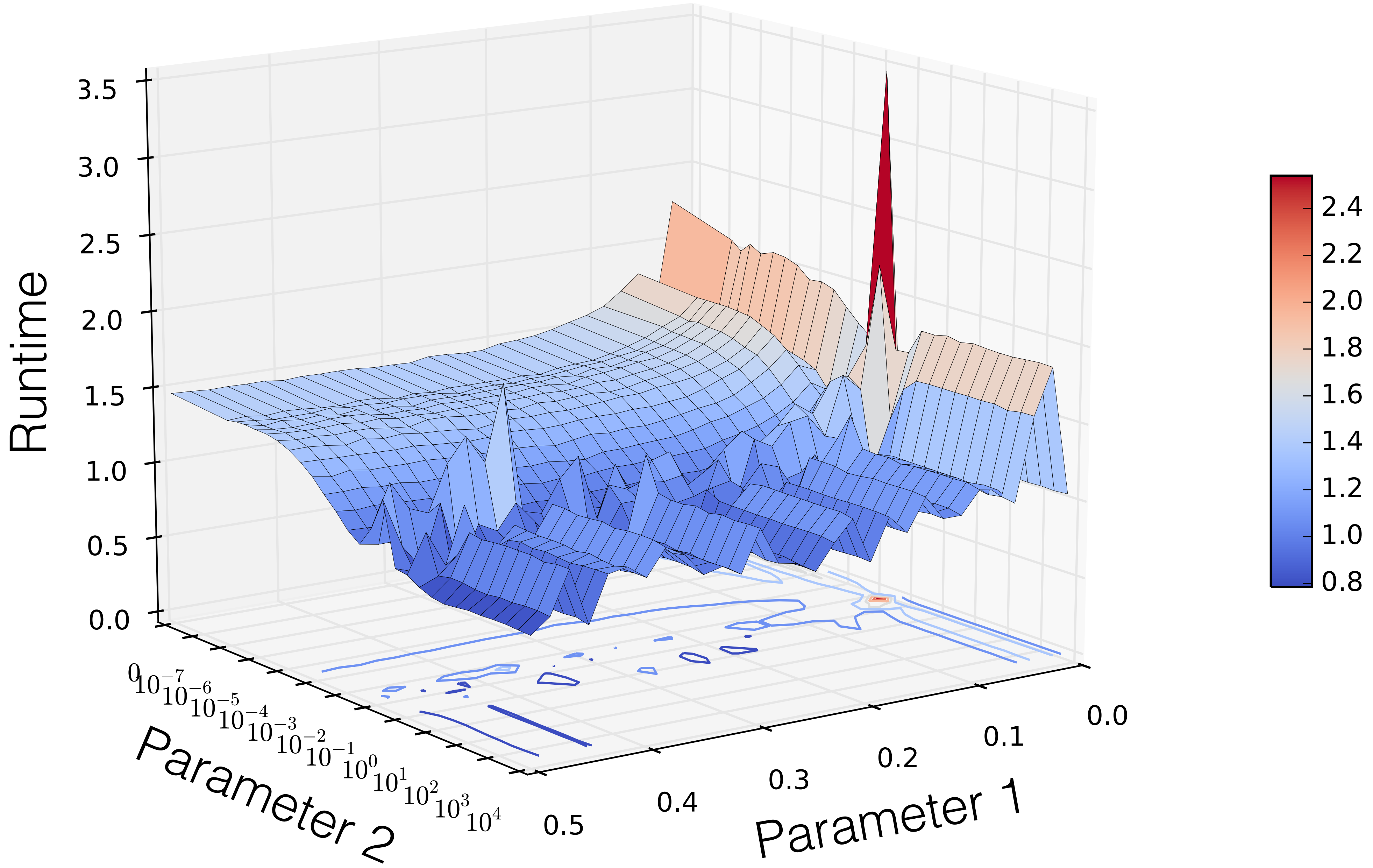


**Two objectives**:

1. Minimize clock cycles (runtime)
2. Minimize FPGA logic utilization
    - Useful for fitting multiple applications
    - Proxy for energy consumption

**One constraint**: design must fit in the chip

# Motivation - Mono-objective

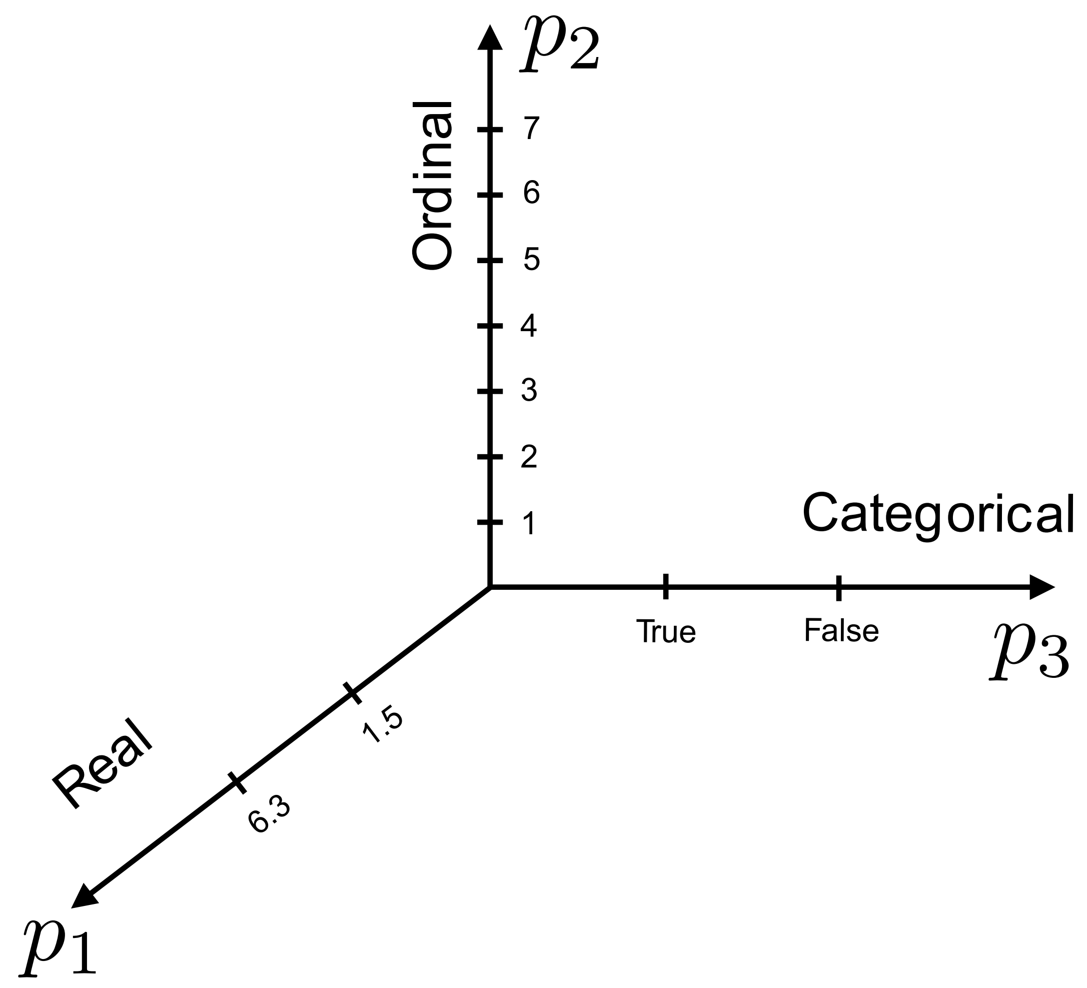


- Benchmark: SLAMBench 1.0 runtime response surface is: non-linear, multi-modal and non-smooth

# Derivative-Free Optimization (DFO)
# 3-parameters and 2-objective Pictorial

**Input space**

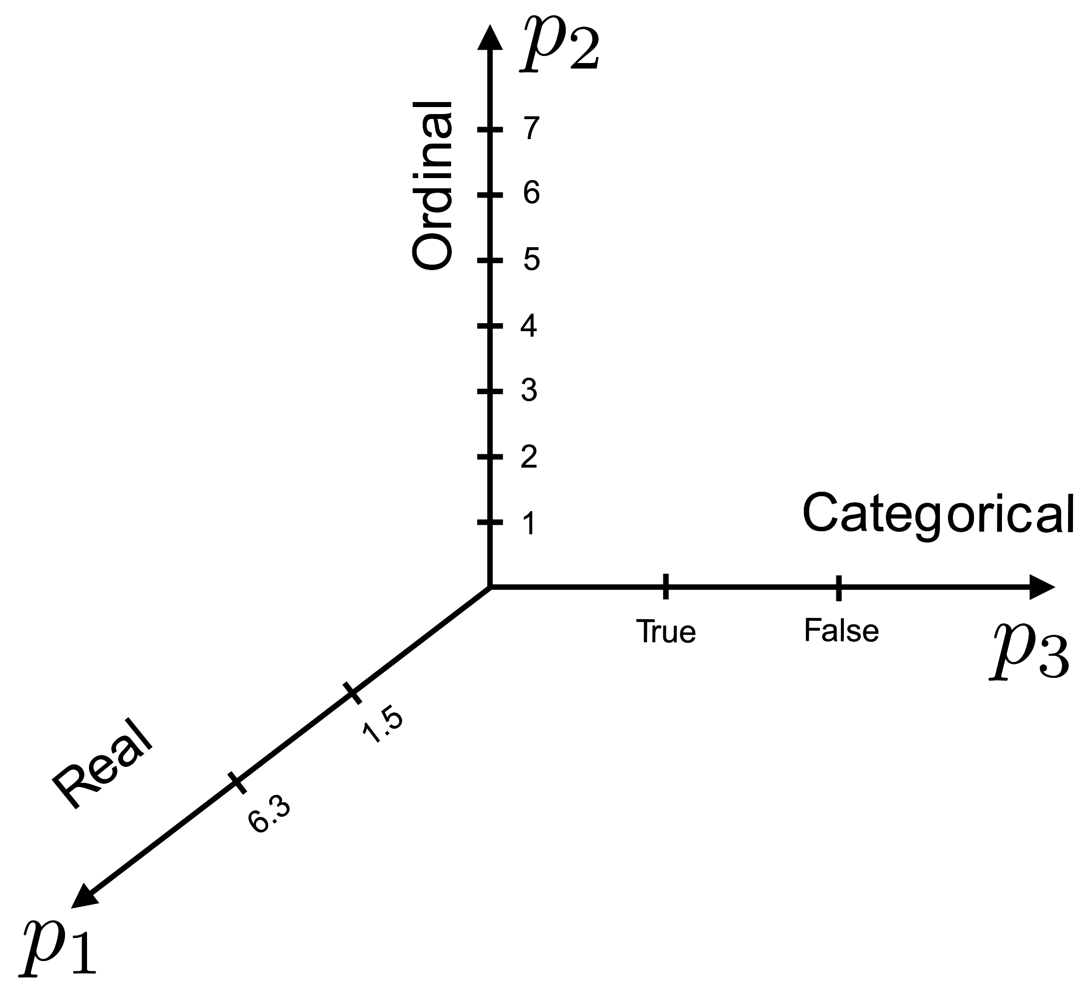
(a.k.a. search or design space)



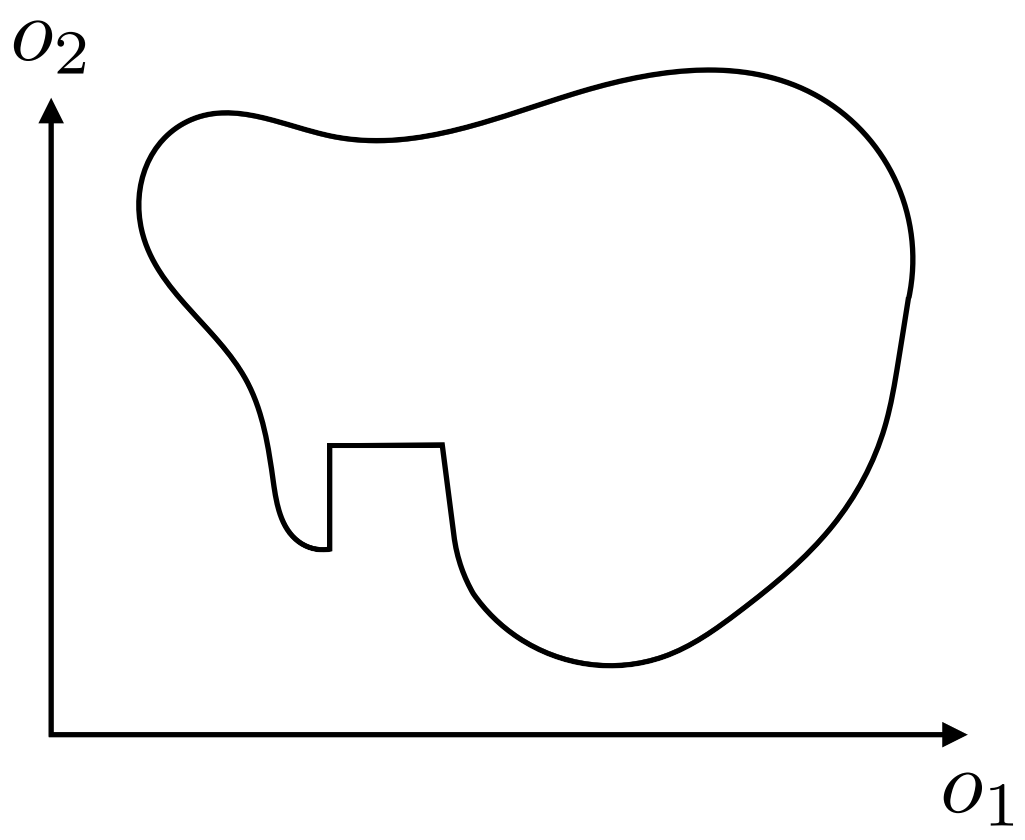
Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO)
# 3-parameters and 2-objective Pictorial

**Input space**
(a.k.a. search or design space)

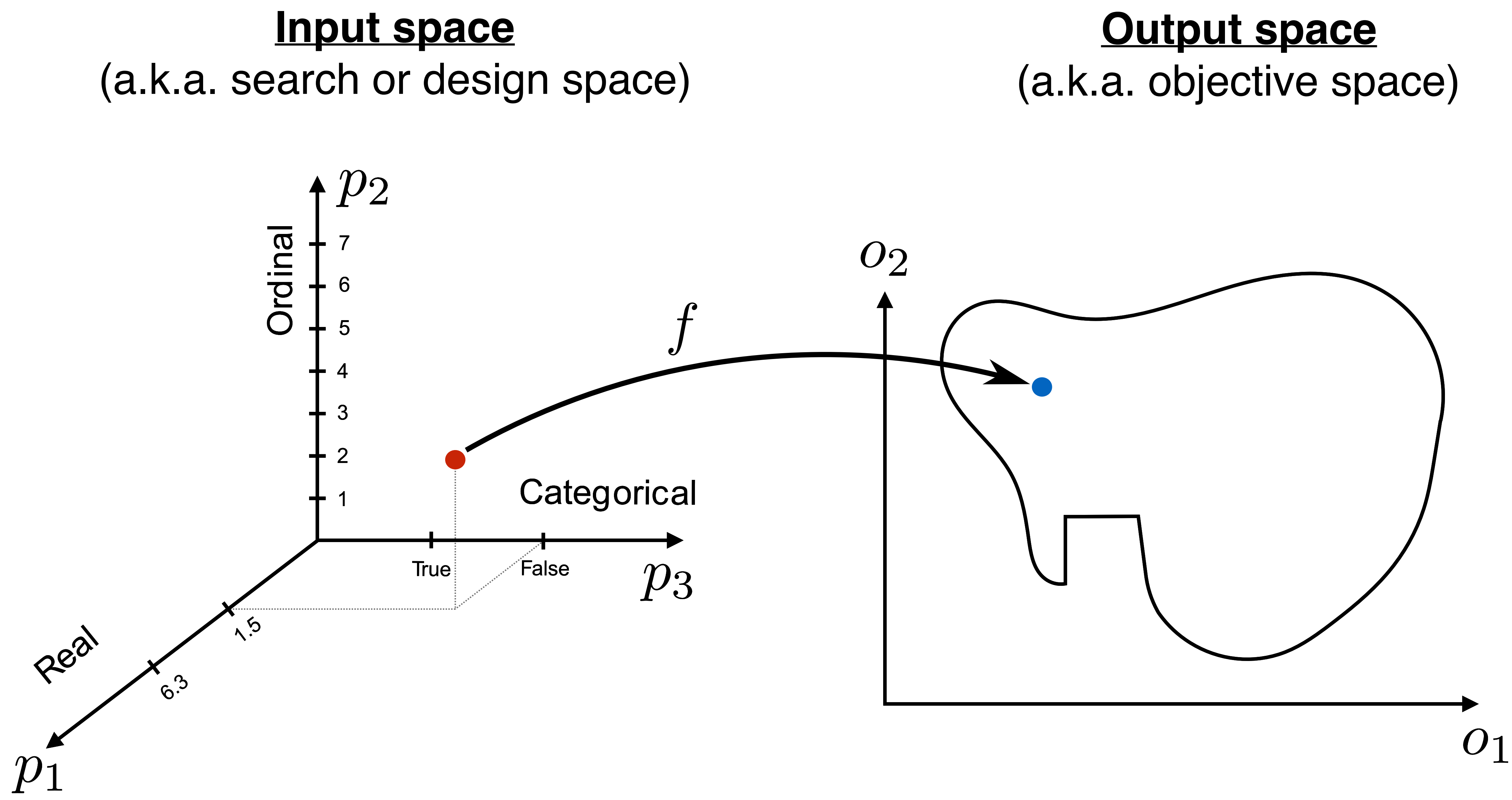


**Output space**
(a.k.a. objective space)



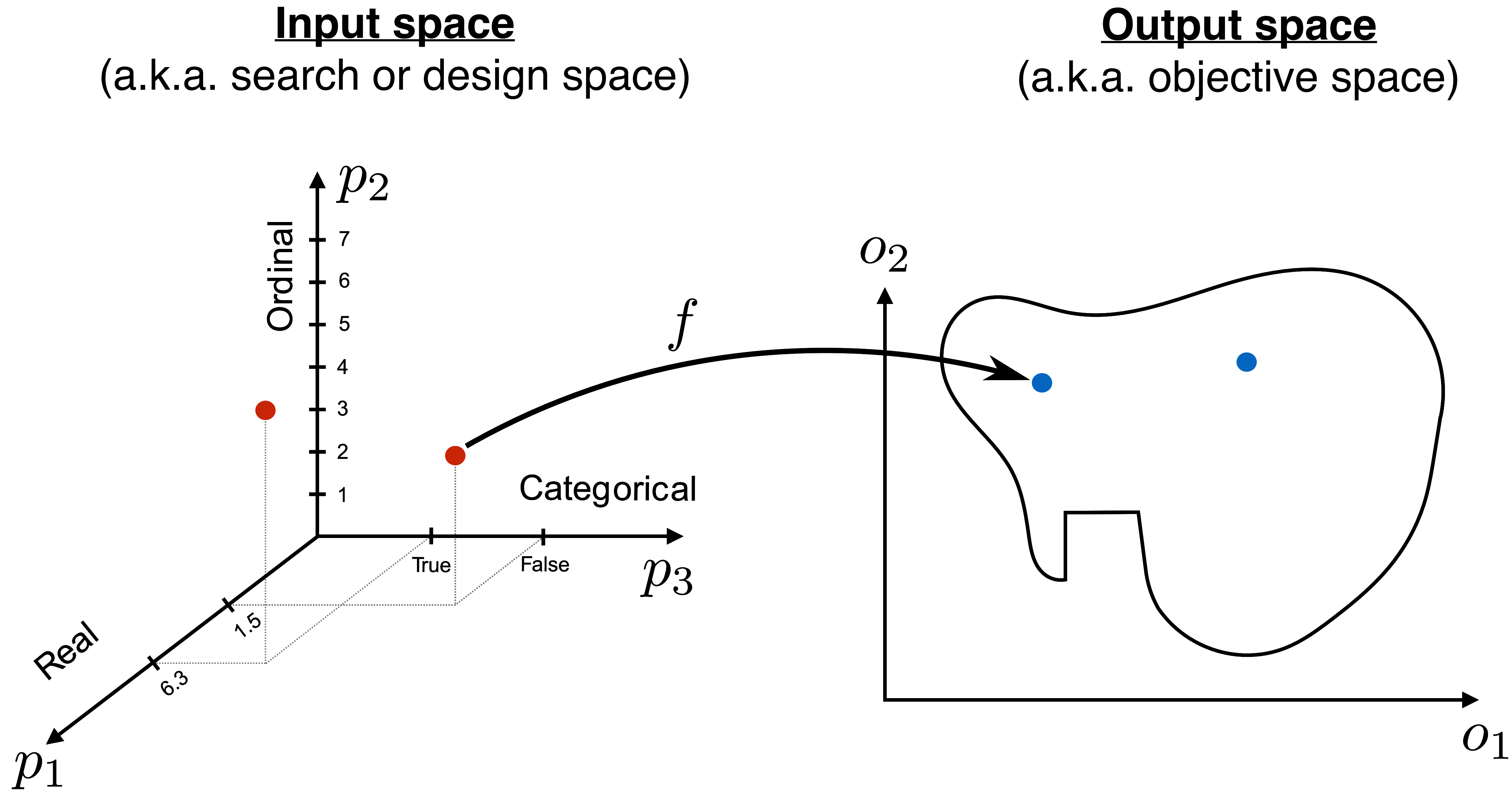
Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO)
# 3-parameters and 2-objective Pictorial



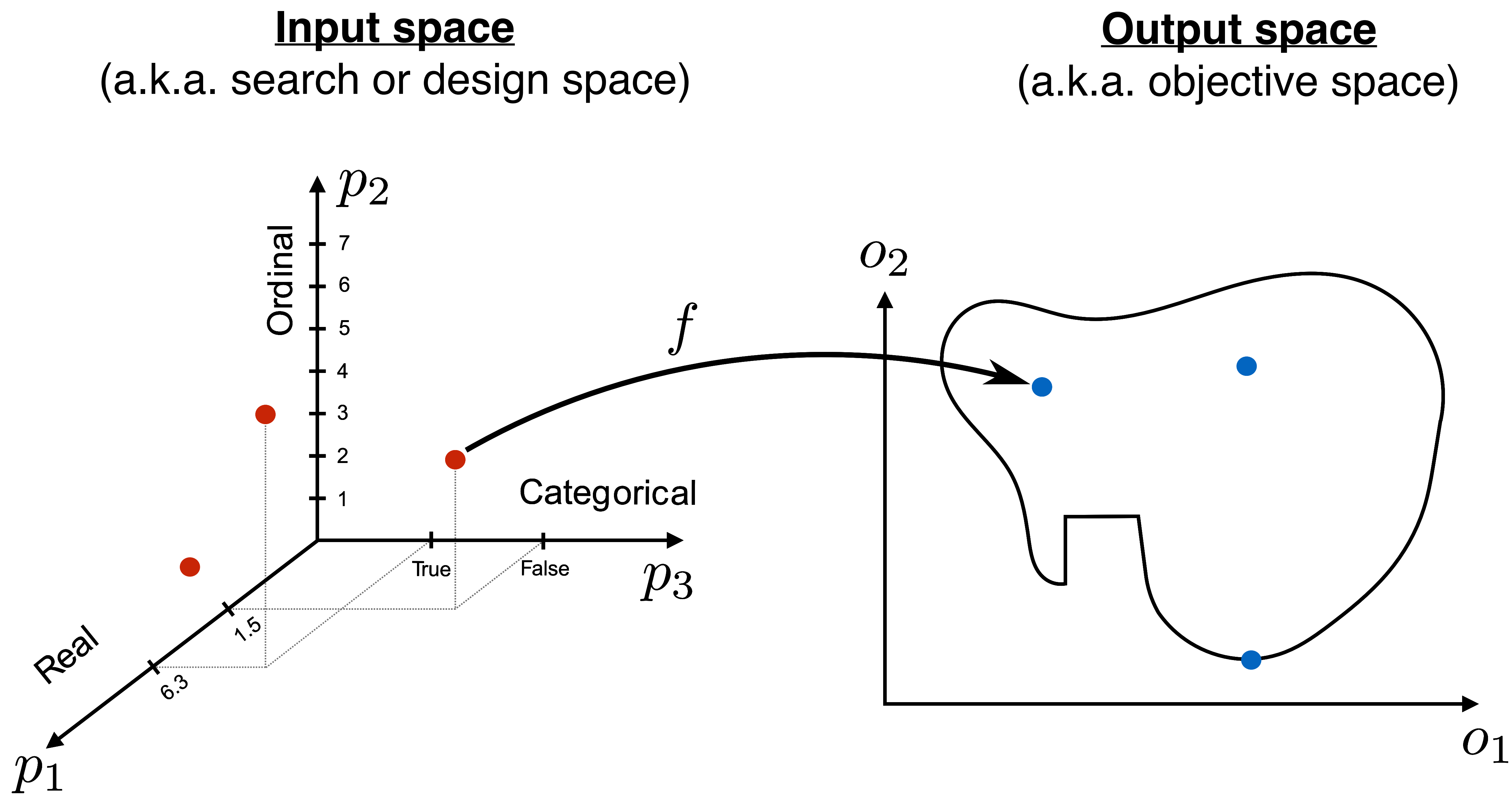
Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO) 3-parameters and 2-objective Pictorial



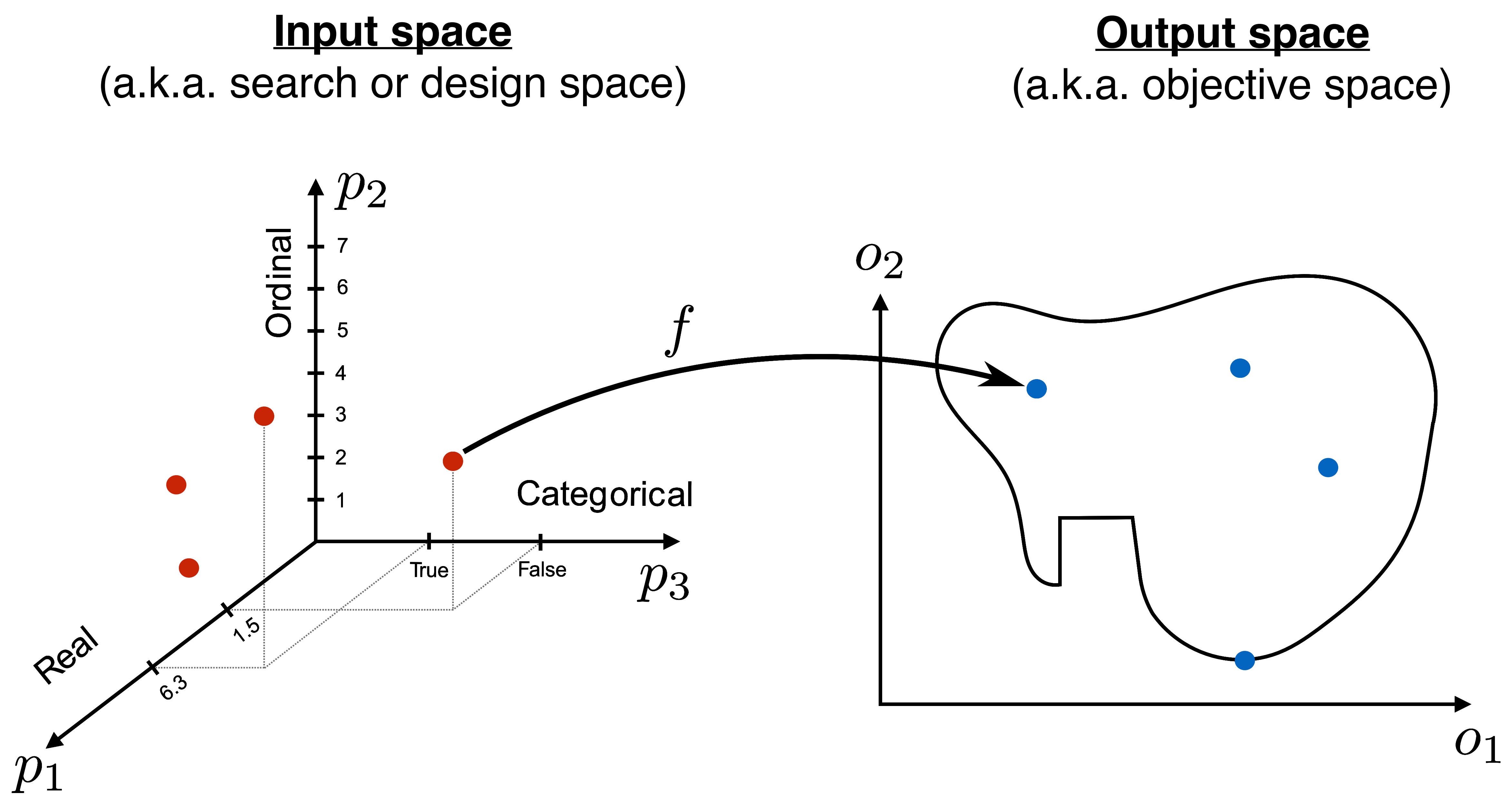
Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO) 3-parameters and 2-objective Pictorial



Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO) 3-parameters and 2-objective Pictorial

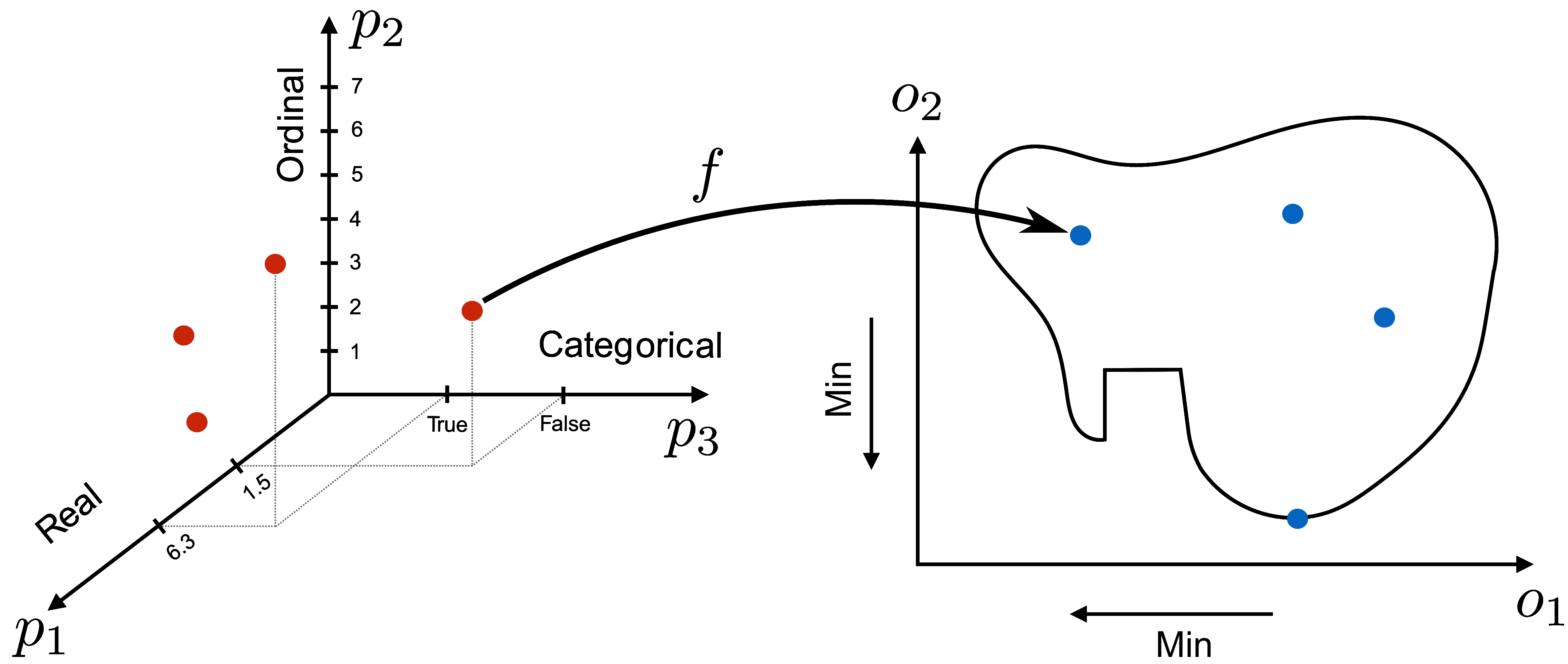


Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO) 3-parameters and 2-objective Pictorial

**Input space**
(a.k.a. search or design space)

**Output space**
(a.k.a. objective space)

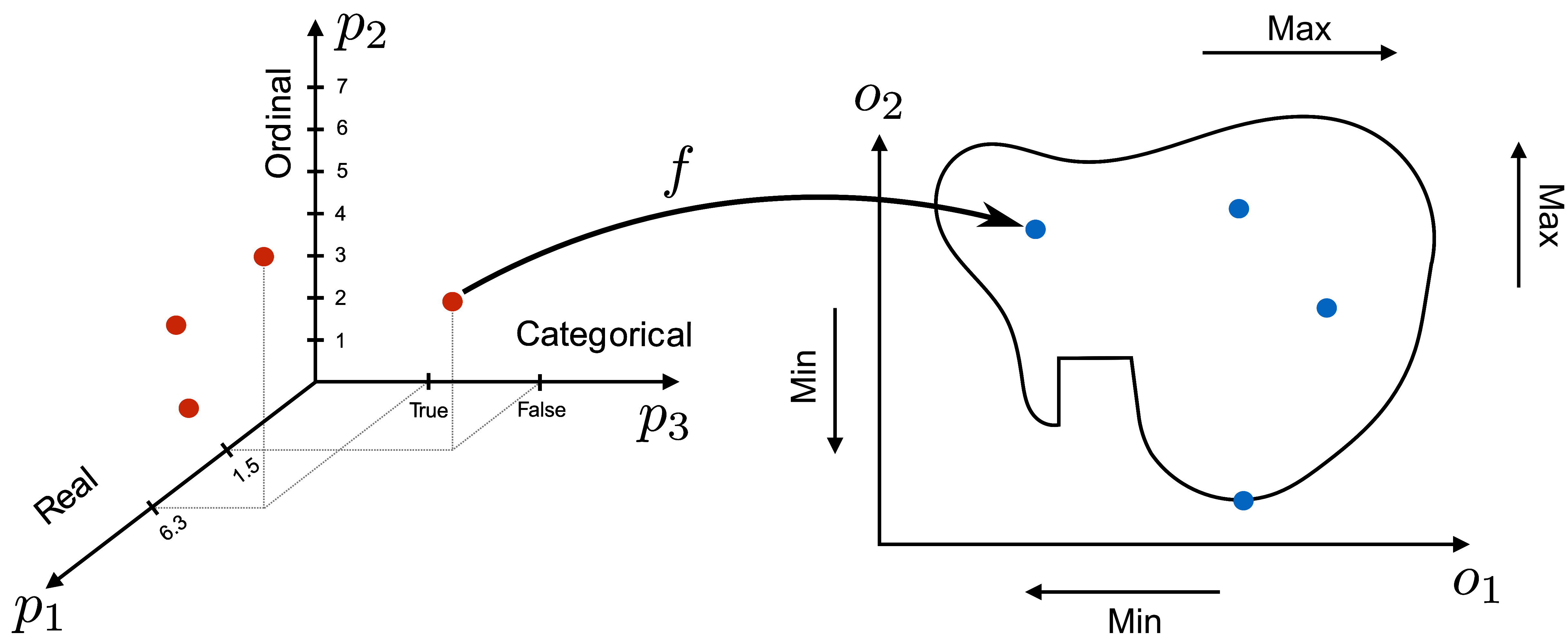


Derivatives are unavailable: e.g., categorical variables

# Derivative-Free Optimization (DFO) 3-parameters and 2-objective Pictorial

**Input space**
(a.k.a. search or design space)

**Output space**
(a.k.a. objective space)



Derivatives are unavailable: e.g., categorical variables

# Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
    - Example: tile size is an ordinal, parallelism is a categorical

# Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
    - Example: tile size is an ordinal, parallelism is a categorical
2. User prior knowledge in the search (Prior)
    - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 4

# Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
   - Example: tile size is an ordinal, parallelism is a categorical
2. User prior knowledge in the search (Prior)
   - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 4
3. Unknown feasibility constraints (Constr.)
   - Example: does a design fit in the FPGA?

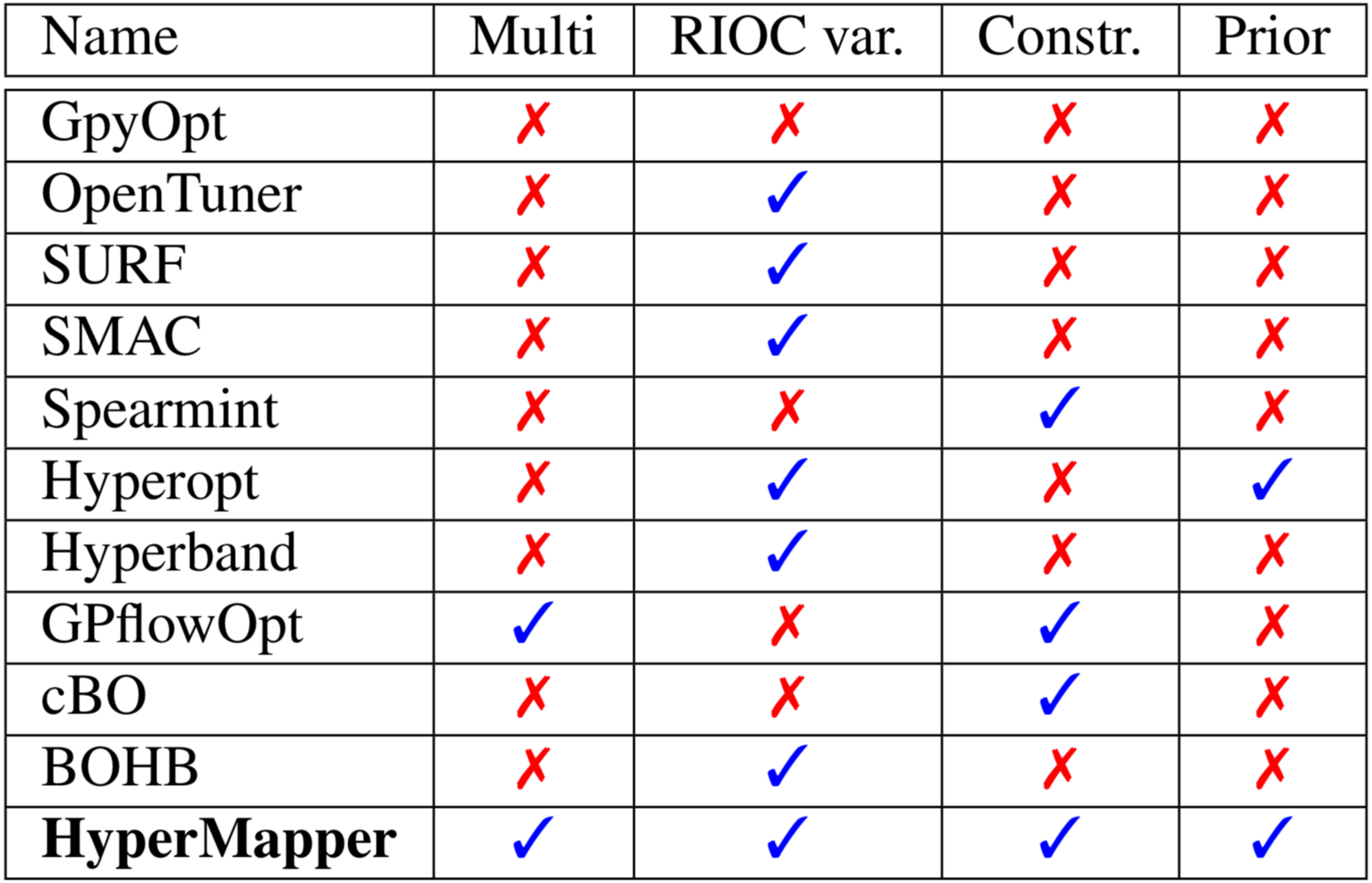
# Practical DSE: Important Features

1. Real, integer, ordinal and categorical variables (RIOC var.)
    - Example: tile size is an ordinal, parallelism is a categorical
2. User prior knowledge in the search (Prior)
    - Example: # threads on a CPU with 4 cores? Gaussian-shaped around 4
3. Unknown feasibility constraints (Constr.)
    - Example: does a design fit in the FPGA?
4. Multi-objective optimization (Multi)
    - Example: trade-off runtime and area

# DFO Tools Taxonomy

## None of the tools available support all these DSE features

We introduce a new framework dubbed **HyperMapper**

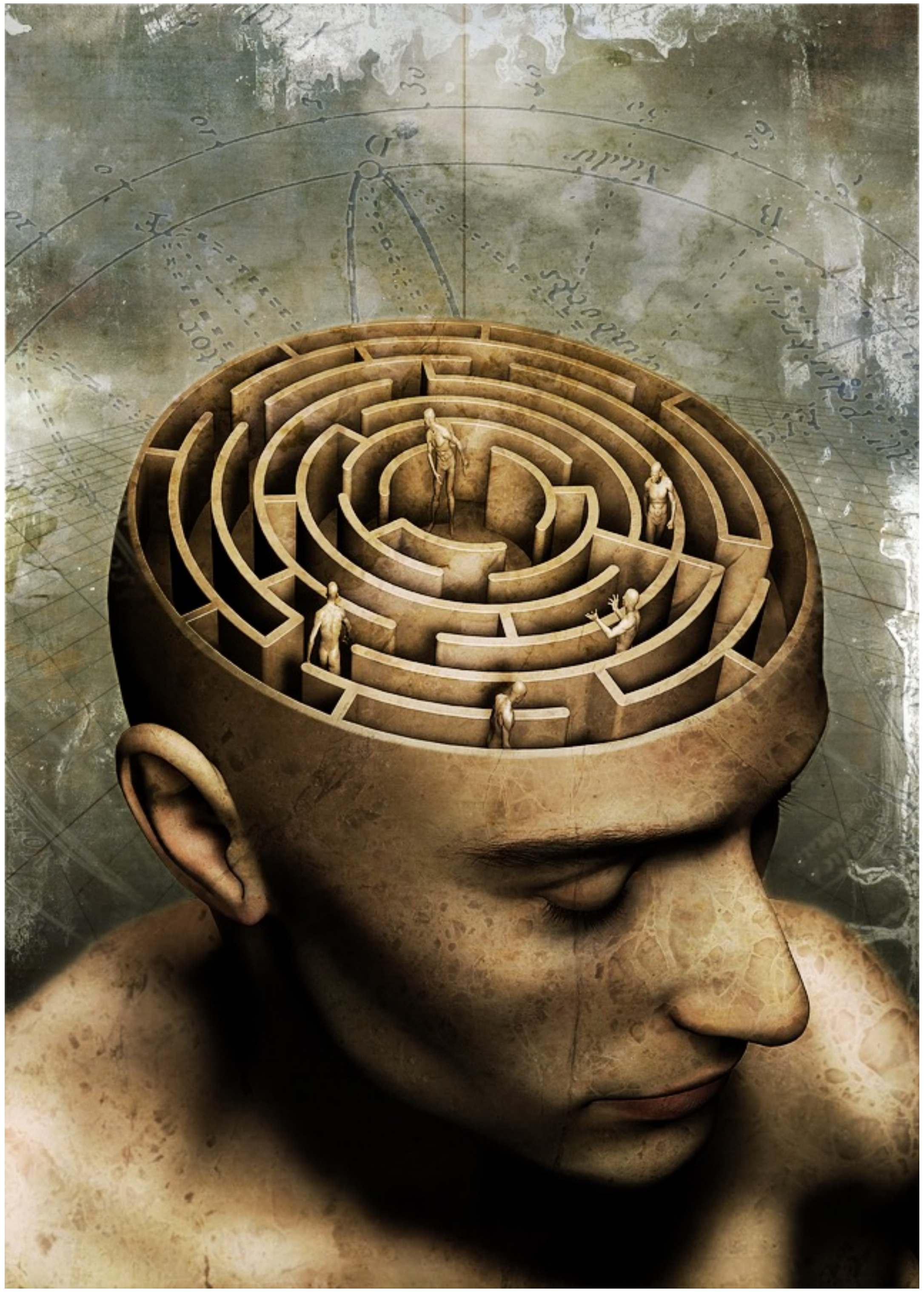


| Name | Multi | RIOC var. | Constr. | Prior |
|---|---|---|---|---|
| GpyOpt | ✗ | ✗ | ✗ | ✗ |
| OpenTuner | ✗ | ✓ | ✗ | ✗ |
| SURF | ✗ | ✓ | ✗ | ✗ |
| SMAC | ✗ | ✓ | ✗ | ✗ |
| Spearmint | ✗ | ✗ | ✓ | ✗ |
| Hyperopt | ✗ | ✓ | ✗ | ✓ |
| Hyperband | ✗ | ✓ | ✗ | ✗ |
| GPflowOpt | ✓ | ✗ | ✓ | ✗ |
| cBO | ✗ | ✗ | ✓ | ✗ |
| BOHB | ✗ | ✓ | ✗ | ✗ |
| **HyperMapper** | ✓ | ✓ | ✓ | ✓ |

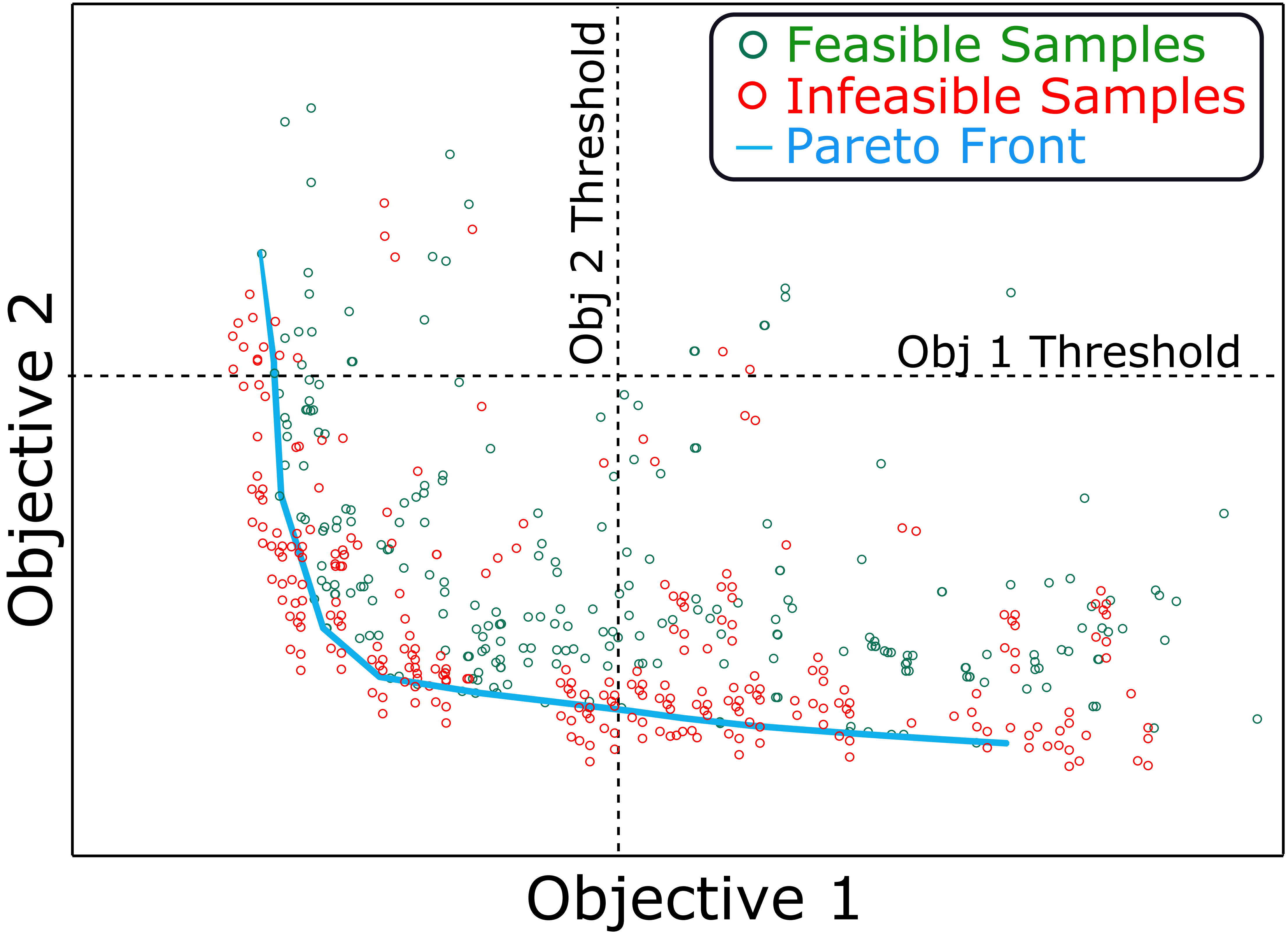
[Bodin, et al.] "Integrating algorithmic parameters into benchmarking and design space exploration in 3D scene understanding", PACT, 2016.
[Nardi, et al.] "Algorithmic performance-accuracy trade-off in 3D vision applications using HyperMapper", IPDPS-iWAPT, 2017.
[Nardi, et al., 2019] "Practical Design Space Exploration", MASCOTS, 2019.

# Outline

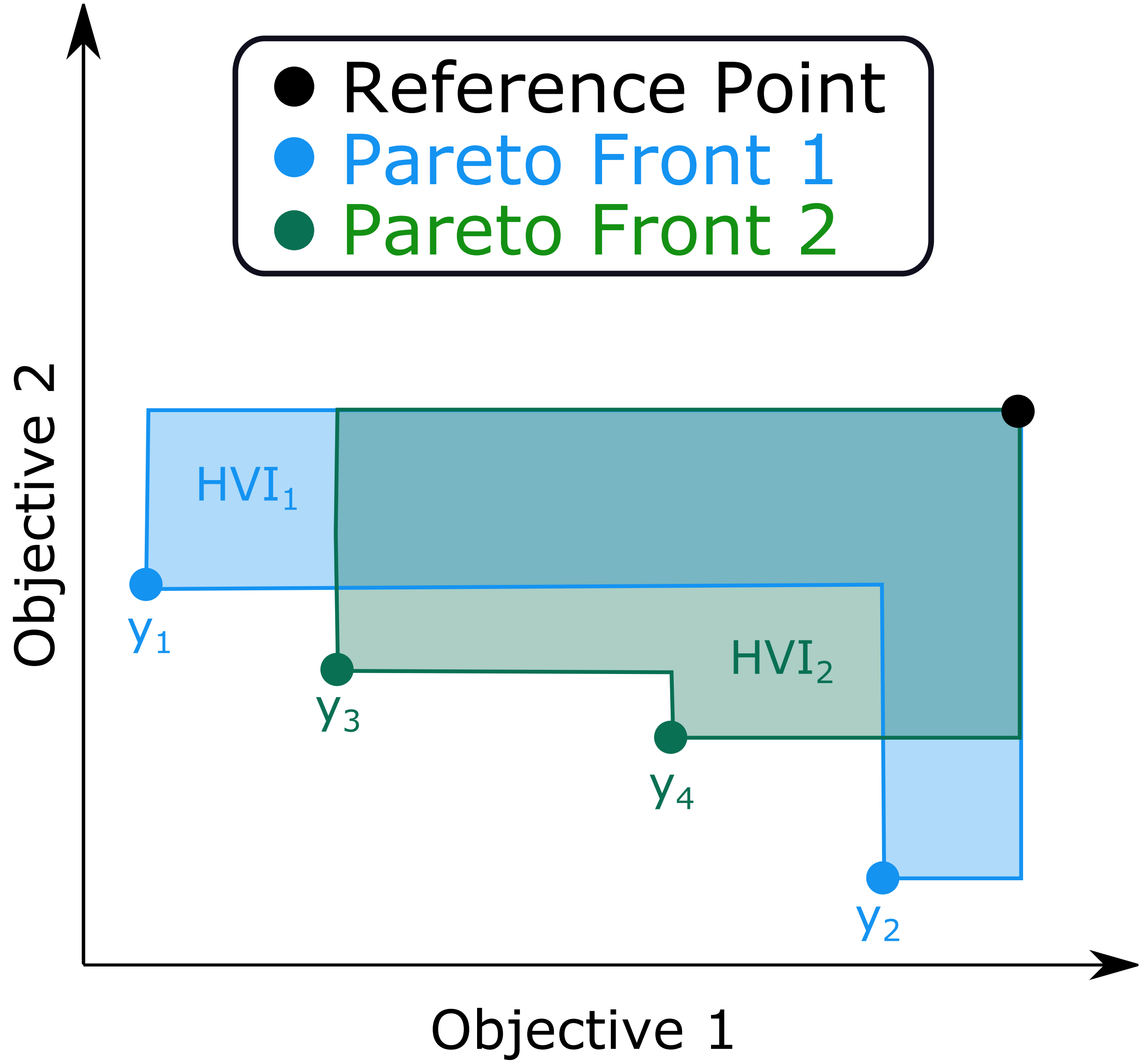
1. Design Space Exploration
   - Motivation Example
   - Problem Setting
   - Important Features
   - Taxonomy
2. **The HyperMapper Framework**
   - **Pareto and Hypervolumes**
   - **Prior Distribution**
   - **Pareto-based Active Learning**
3. Experimental Results



# Multi-Objective Goal: Pareto Front



# HyperVolume Indicator (HVI)



- HVI is a hypervolume
    - with respect to a reference
- It is always a scalar
    - even with multiple objectives
- Used to compare 2 Paretos
- The lower the better: **$\text{HVI}_{\text{total}}$ - $\text{HVI}_{\text{i}}$**

# Injecting Prior Knowledge in the DSE

- Need a probability distribution that:

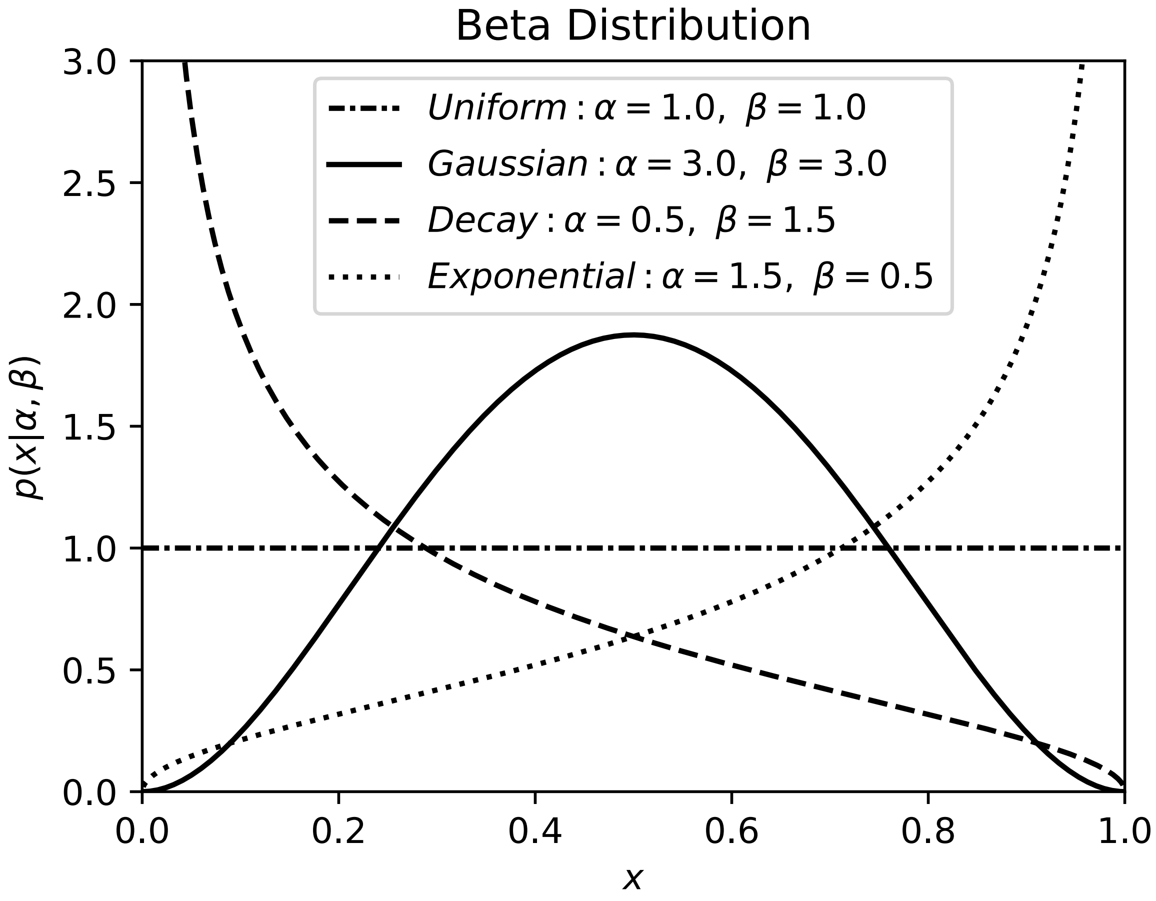
  1. Has a finite domain

  2. Can flexibly model shapes including:

     1. Bell-shape, 2. U-shape and 3. J-shape

- Beta distribution - parameters alpha and beta

- In addition, need of a discrete distribution for categorical variables



PDF given by:

$$f(x|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

for $x \in [0,1]$ and $\alpha, \beta > 0,$

where $\Gamma$ is the Gamma function

# Surrogate Model

- Predicts an outcome given an input x

# Surrogate Model

- Predicts an outcome given an input x
- One surrogate model per objective
    - Random forests (RF) for regression

# Surrogate Model

- Predicts an outcome given an input x
- One surrogate model per objective
    - Random forests (RF) for regression
- One surrogate model per feasibility constraint
    - Random forests for classification

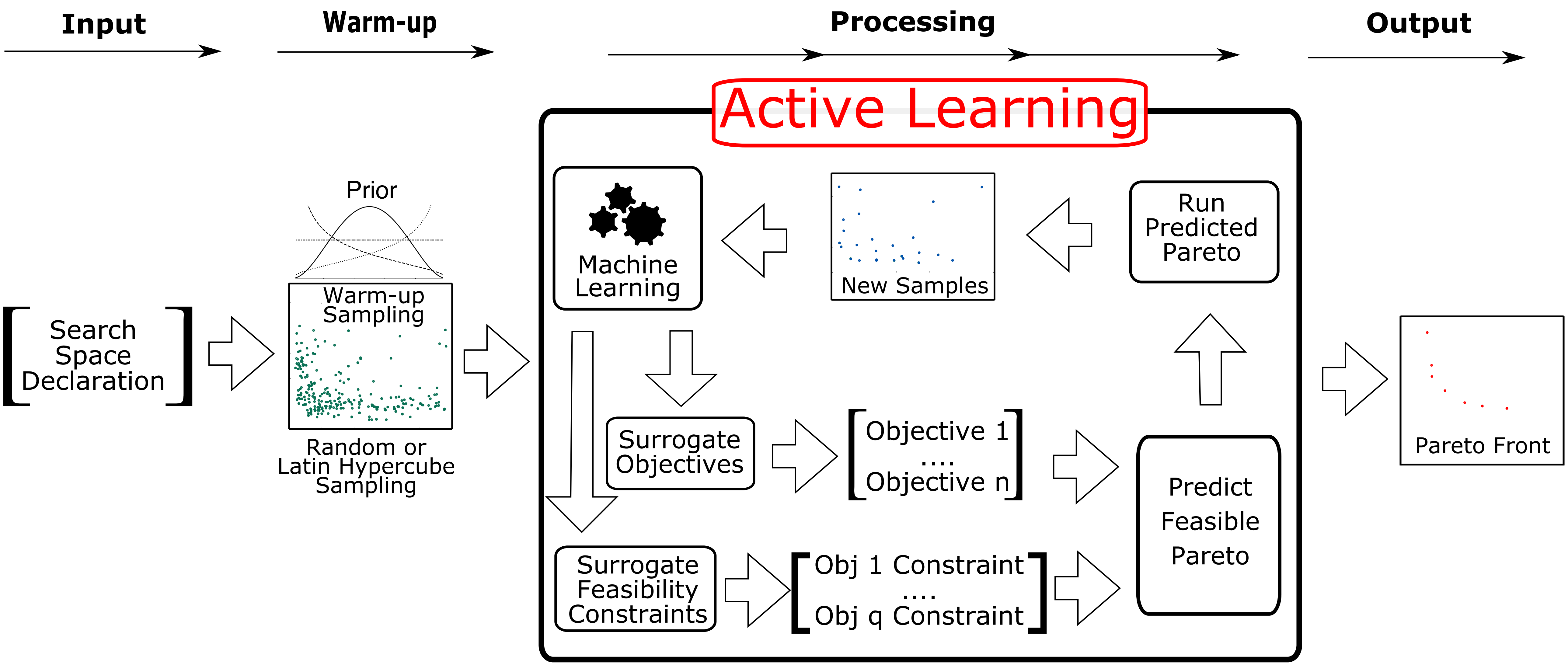
# Surrogate Model

- Predicts an outcome given an input x
- One surrogate model per objective
    - Random forests (RF) for regression
- One surrogate model per feasibility constraint
    - Random forests for classification
- Intuition of why RF is a good model:
    - Good at non-linearity, multi-modality and non-smoothness

# Surrogate Model

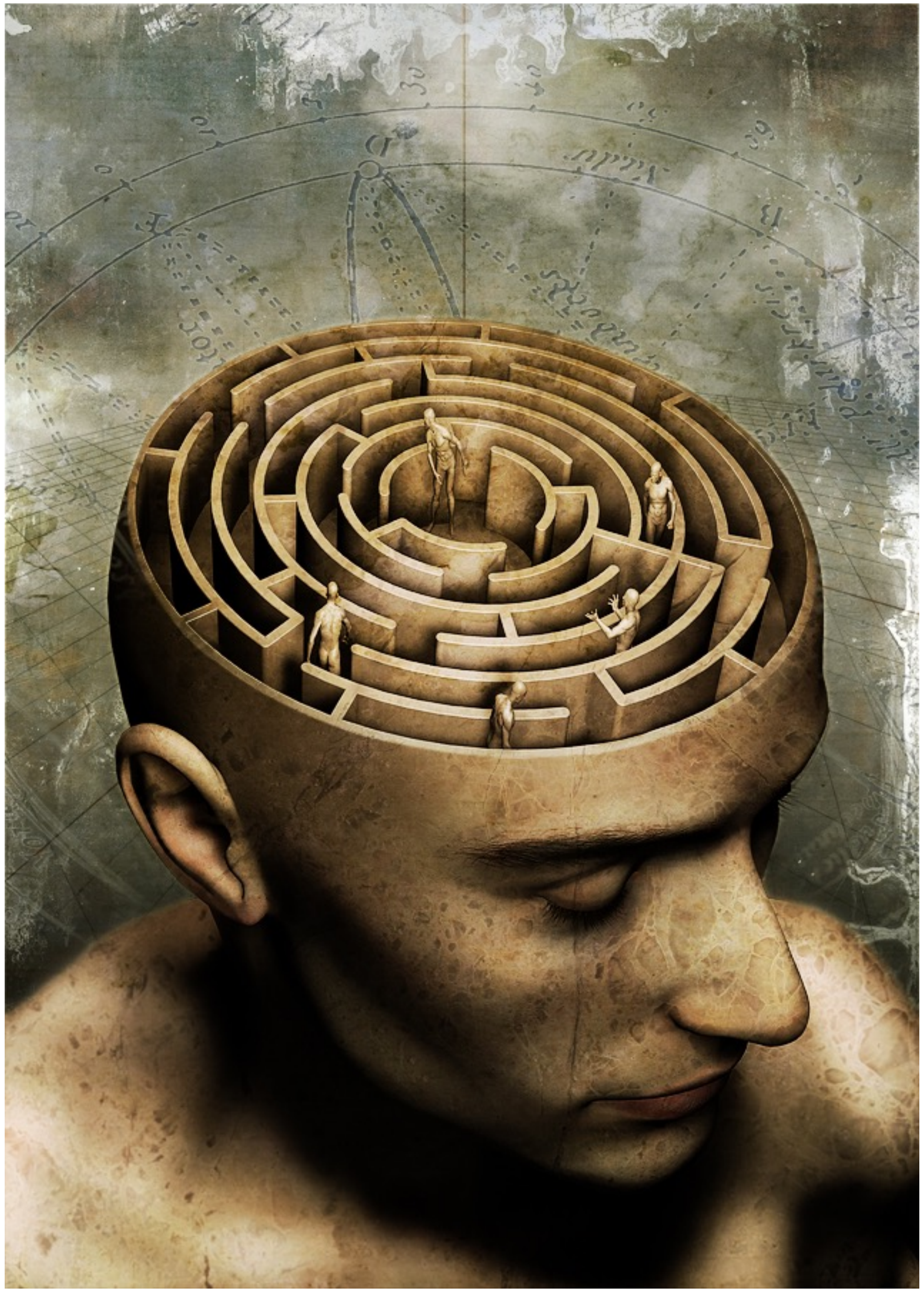
- Predicts an outcome given an input x
- One surrogate model per objective
  - Random forests (RF) for regression
- One surrogate model per feasibility constraint
  - Random forests for classification
- Intuition of why RF is a good model:
  - Good at non-linearity, multi-modality and non-smoothness
- Has not to be perfect (interested in optima not the best model)
  - Coefficient of determination ($R^2$) is the wrong indicator of success

# The HyperMapper Framework

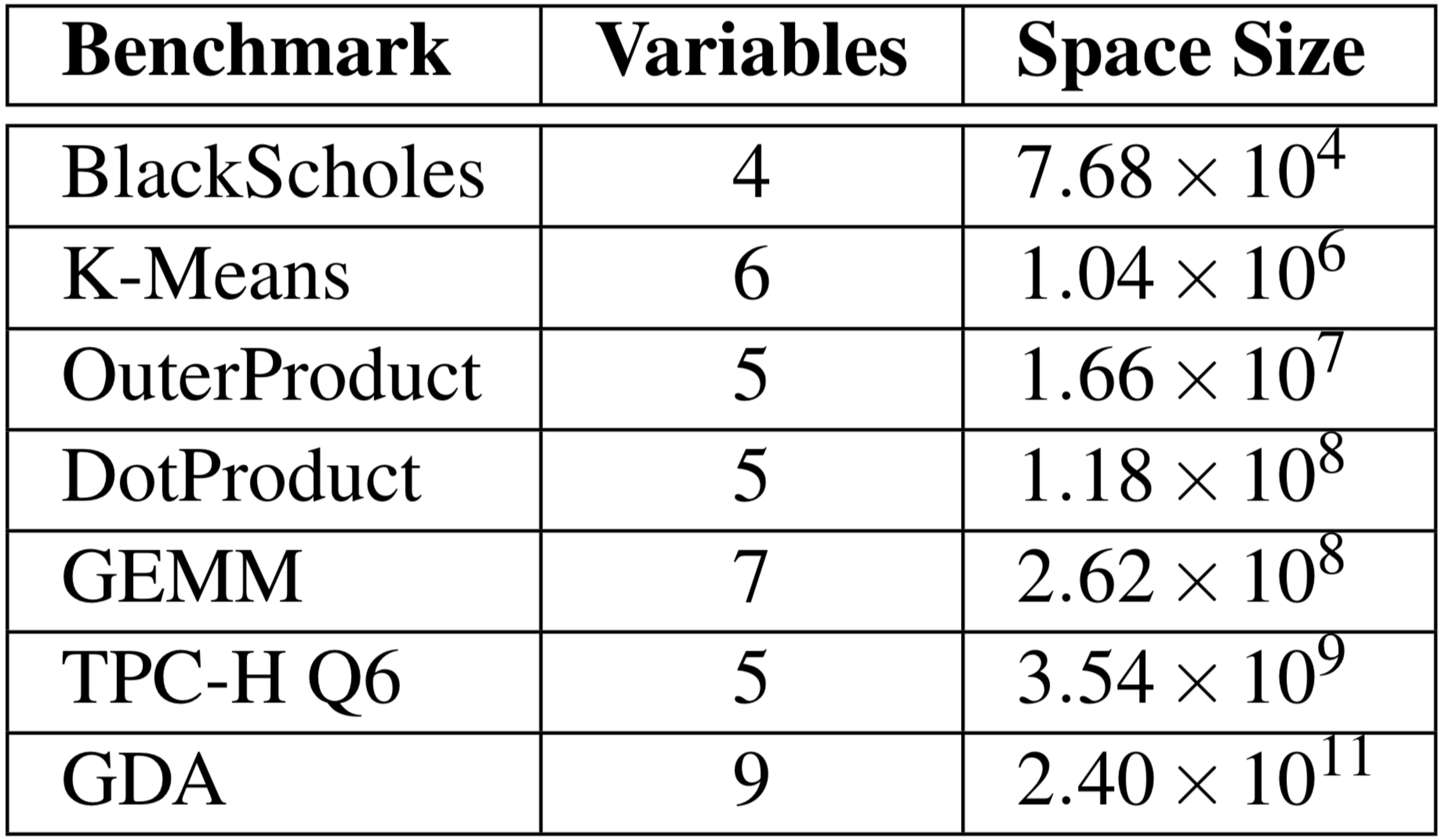


# Outline

1. Design Space Exploration
   - Motivation Example
   - Problem Setting
   - Important Features
   - Taxonomy
2. The HyperMapper Framework
   - Pareto and Hypervolumes
   - Prior Distribution
   - Pareto-based Active Learning
3. **Experimental Results**



# Spatial Search Spaces (Optimization Knobs)



| **Benchmark** | **Variables** | **Space Size** |
|---|---|---|
| BlackScholes | 4 | $7.68 \times 10^{4}$ |
| K-Means | 6 | $1.04 \times 10^{6}$ |
| OuterProduct | 5 | $1.66 \times 10^{7}$ |
| DotProduct | 5 | $1.18 \times 10^{8}$ |
| GEMM | 7 | $2.62 \times 10^{8}$ |
| TPC-H Q6 | 5 | $3.54 \times 10^{9}$ |
| GDA | 9 | $2.40 \times 10^{11}$ |

## Input

Compiler automatically provides params:

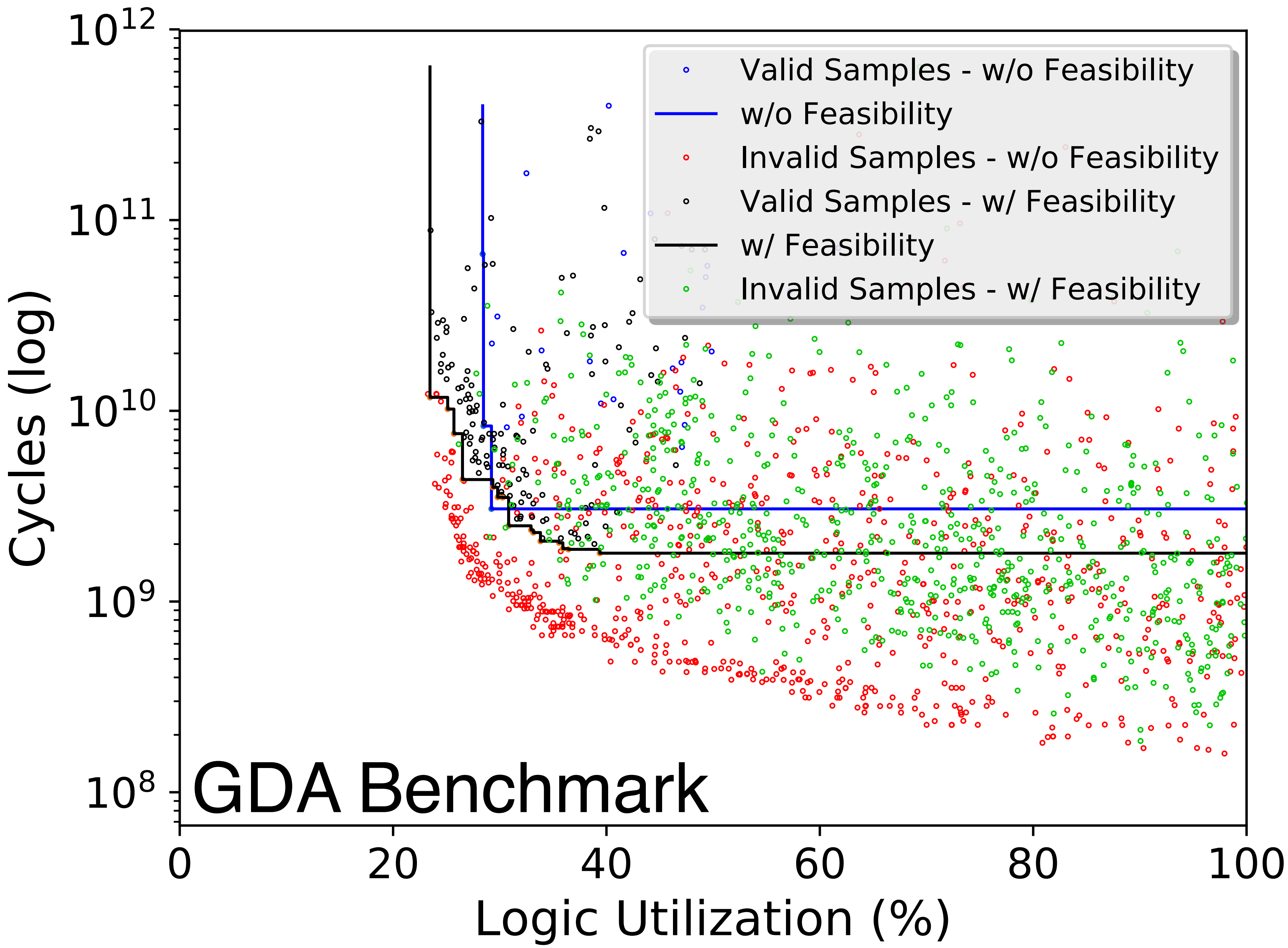
- Tile size (ordinal)
- Inner and outer loop pipelining (ordinal)
- Meta-pipe (categorical)
- Unrolling factor (ordinal)
- Memory banking (ordinal)
- Parallelism (categorical)

## Output

Compiler evaluation provides:

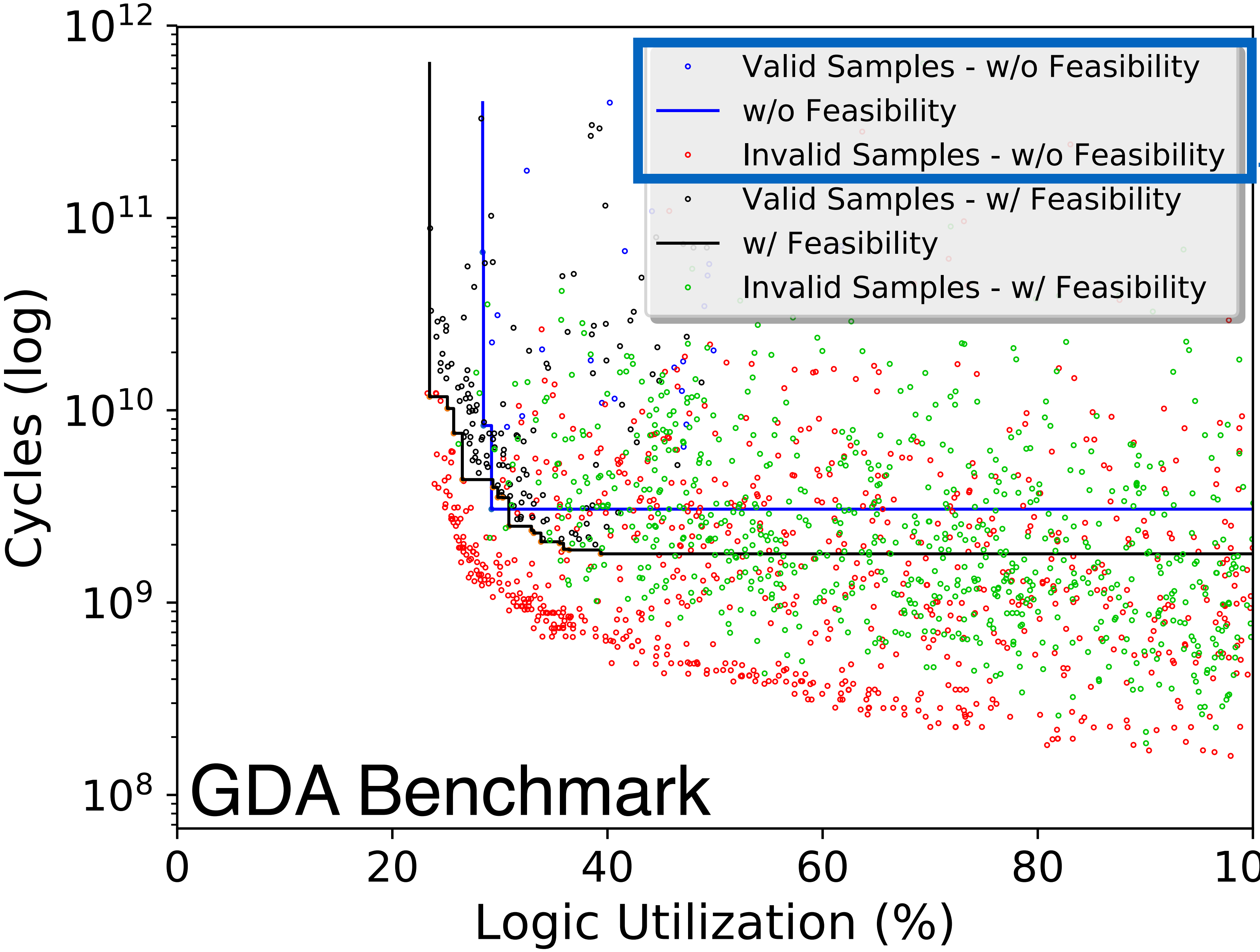
- Clock cycles (runtime): objective 1
- FPGA logic utilization: objective 2
- Feasibility constraint:
  - true if design fits in the chip

# Feasibility Constraint Model Performance



Valid and invalid mean that a design does or does not fit in the FPGA

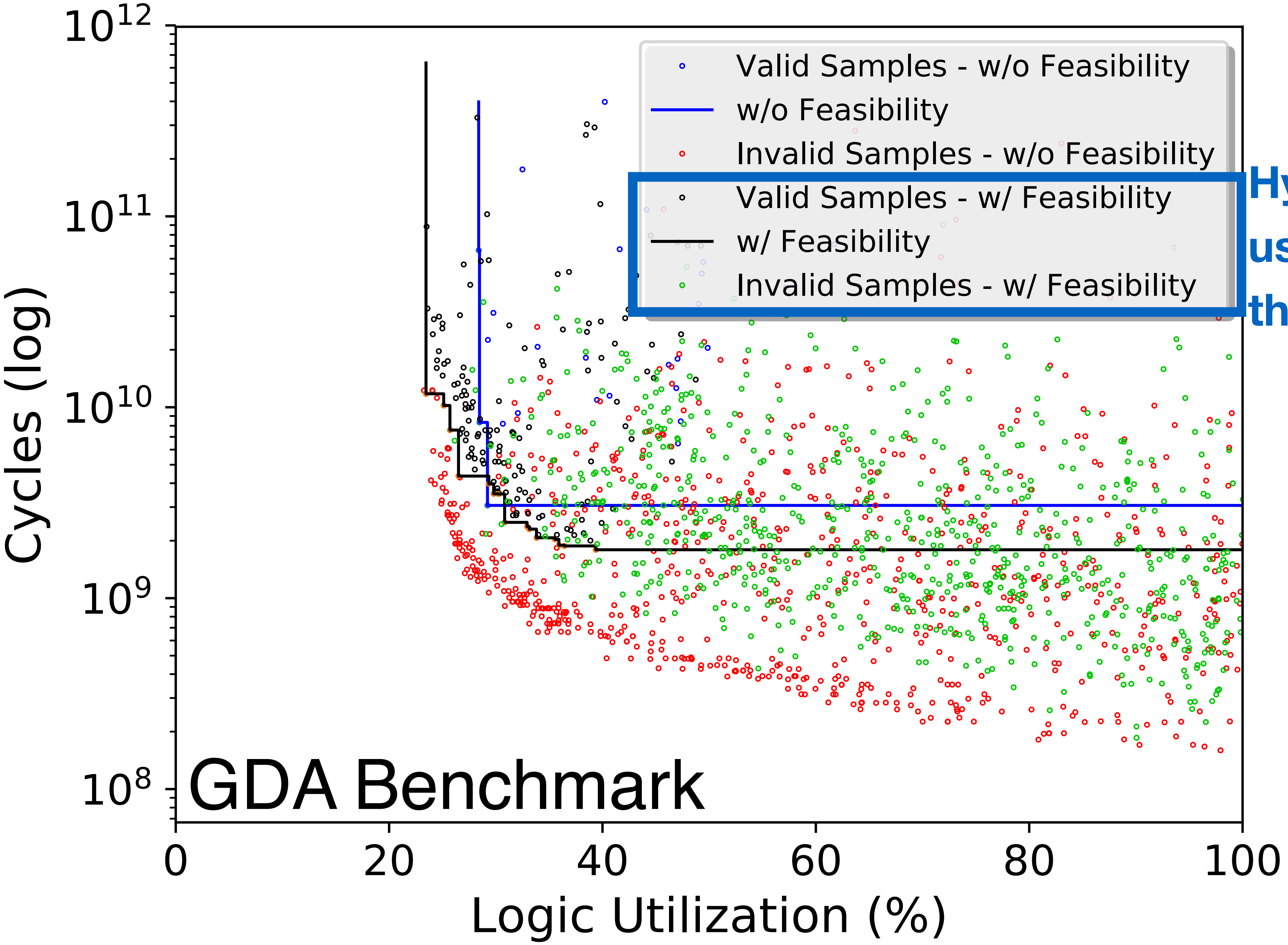
# Feasibility Constraint Model Performance



HyperMapper not using the model for the feasibility constraint

Valid and invalid mean that a design does or does not fit in the FPGA

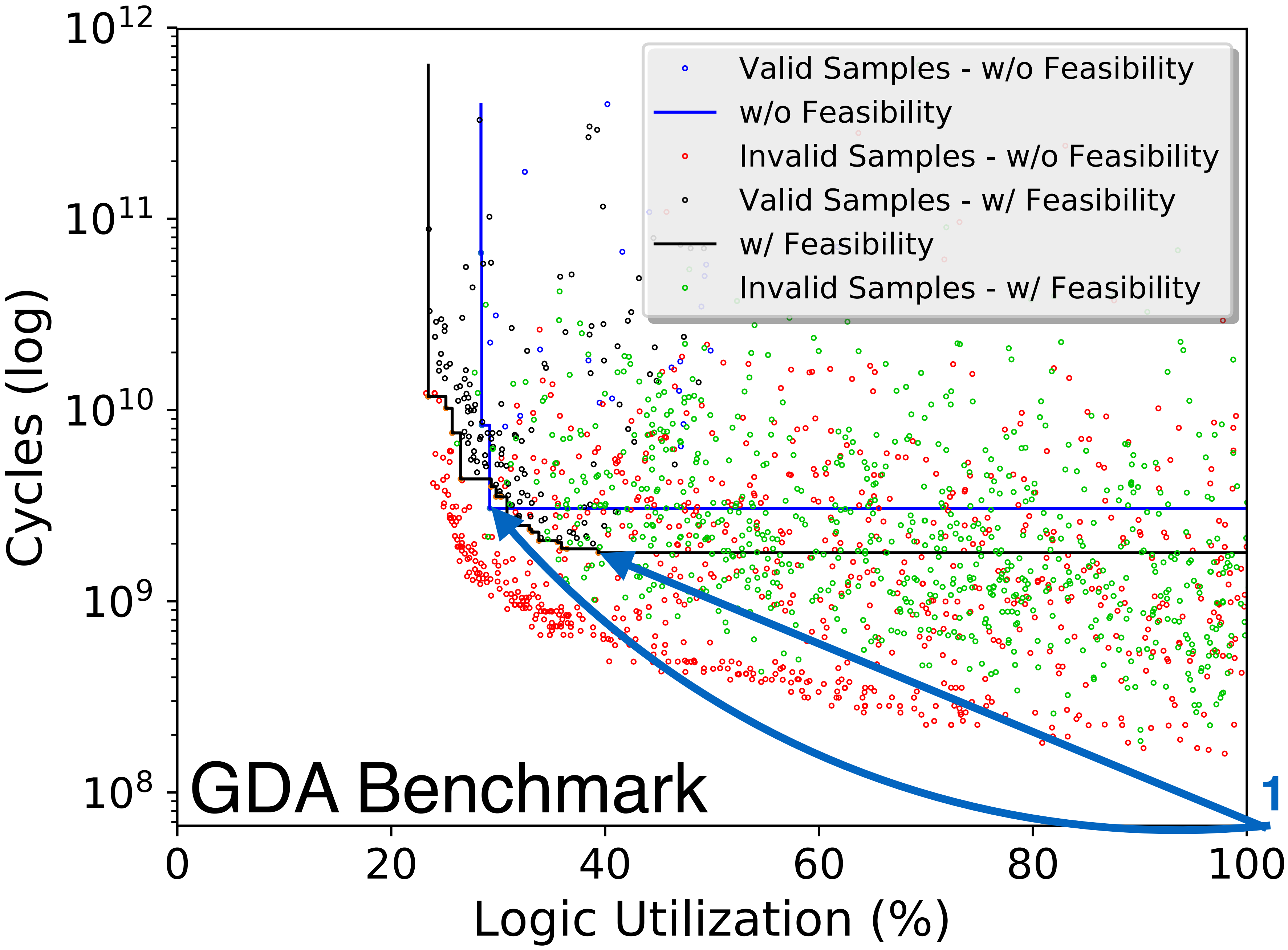
# Feasibility Constraint Model Performance



**HyperMapper using the model for the feasibility constraint**

Valid and invalid mean that a design does or does not fit in the FPGA

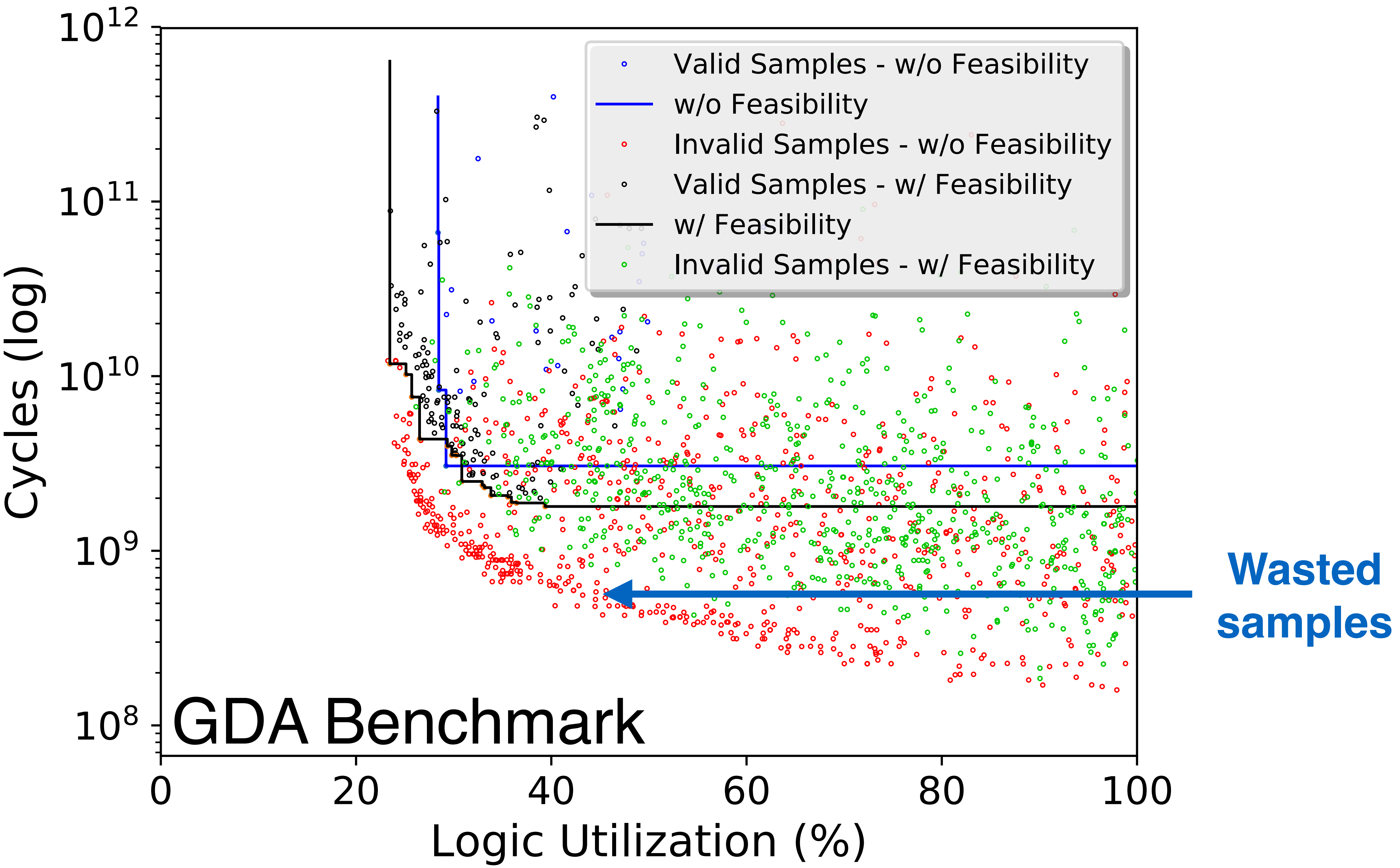
# Feasibility Constraint Model Performance



**About 2x in Cycles, 10% difference in Logic**

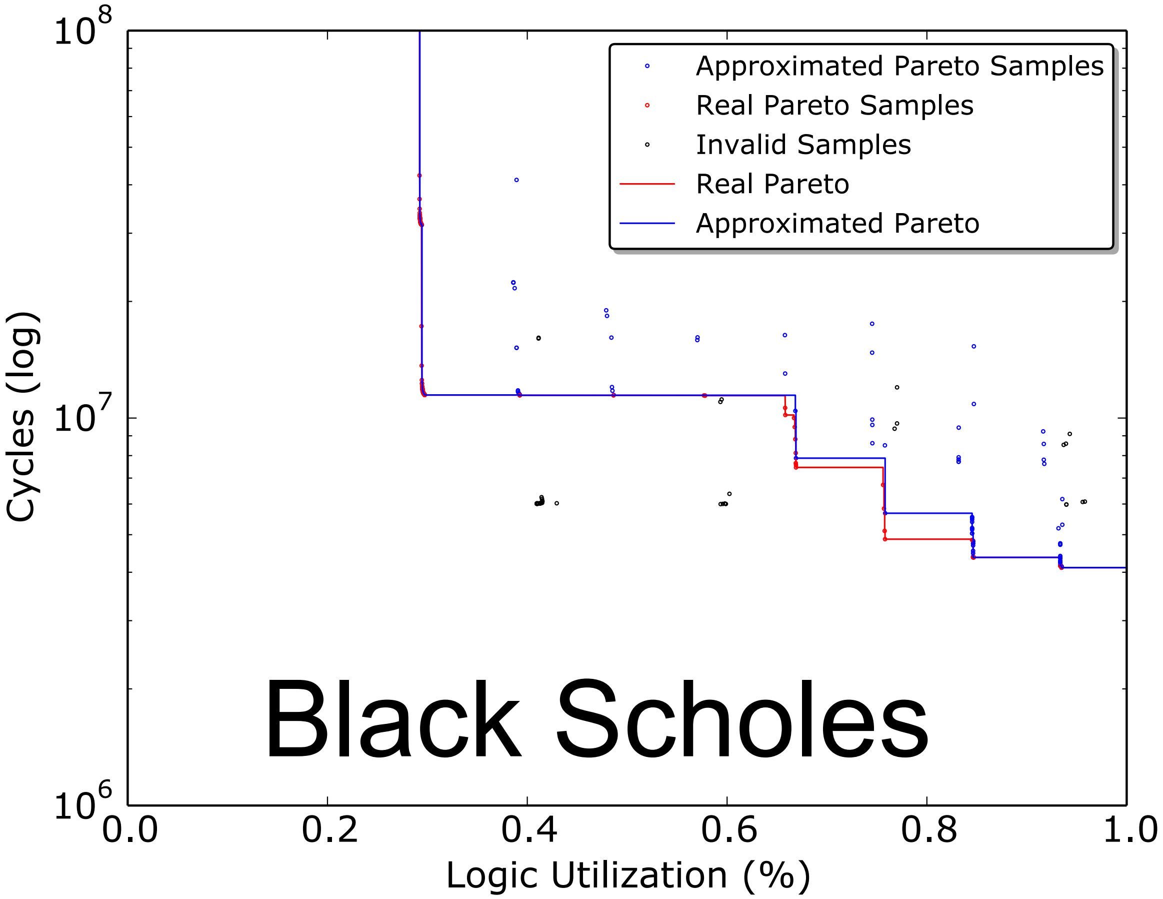
Valid and invalid mean that a design does or does not fit in the FPGA

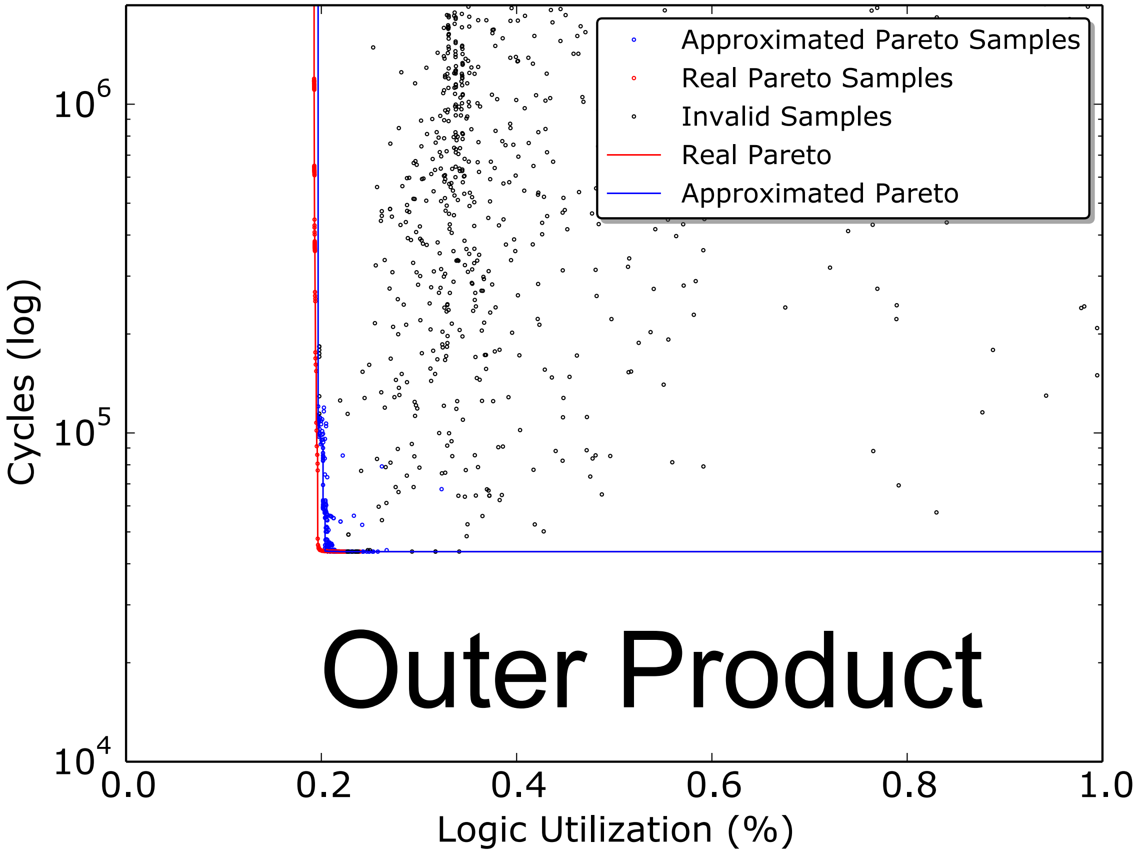
# Feasibility Constraint Model Performance

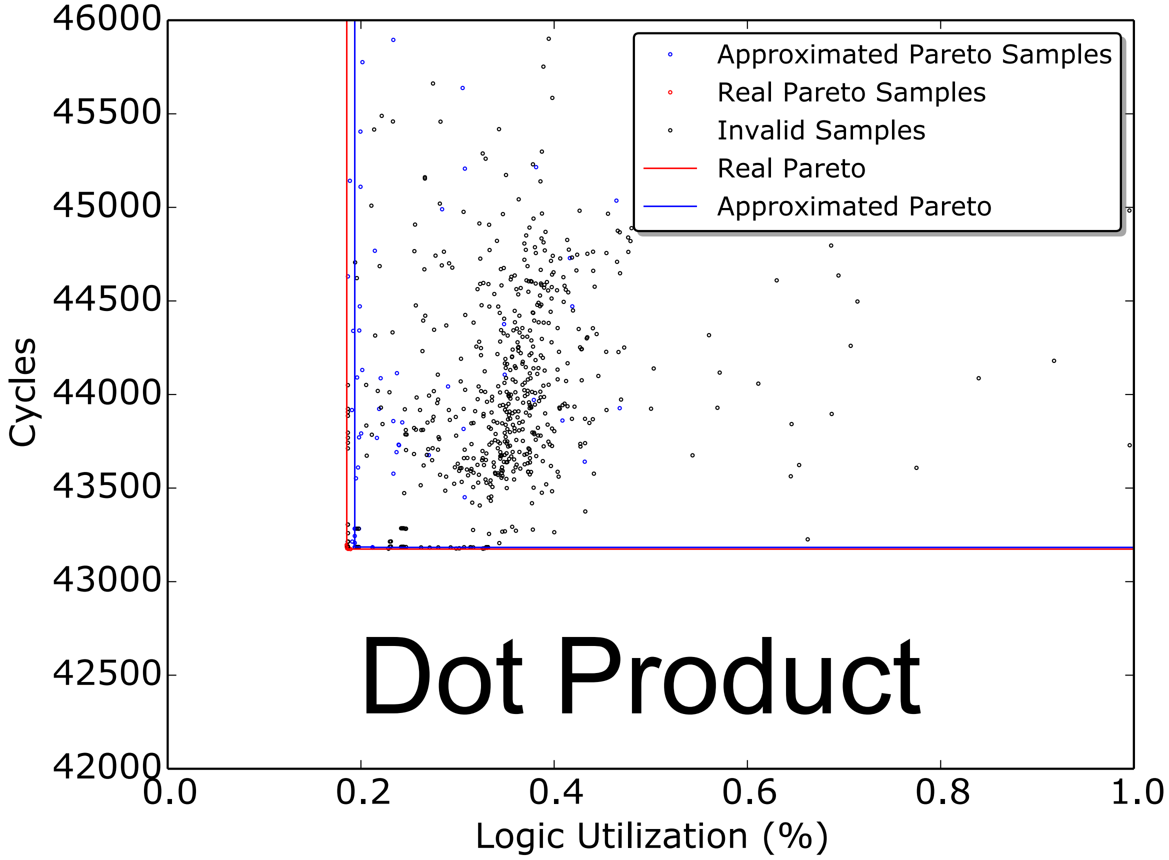


Valid and invalid mean that a design does or does not fit in the FPGA

# Small benchmarks - Cycles/Logic Utilization



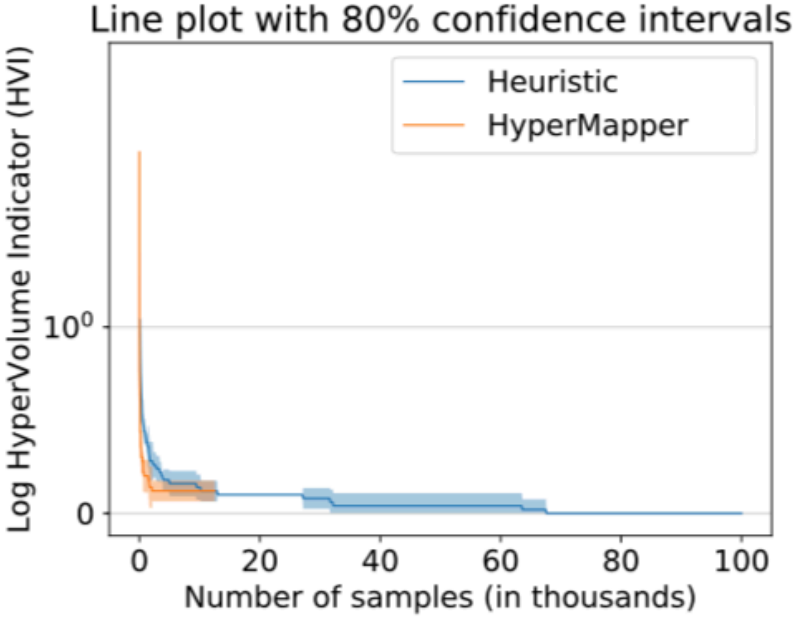




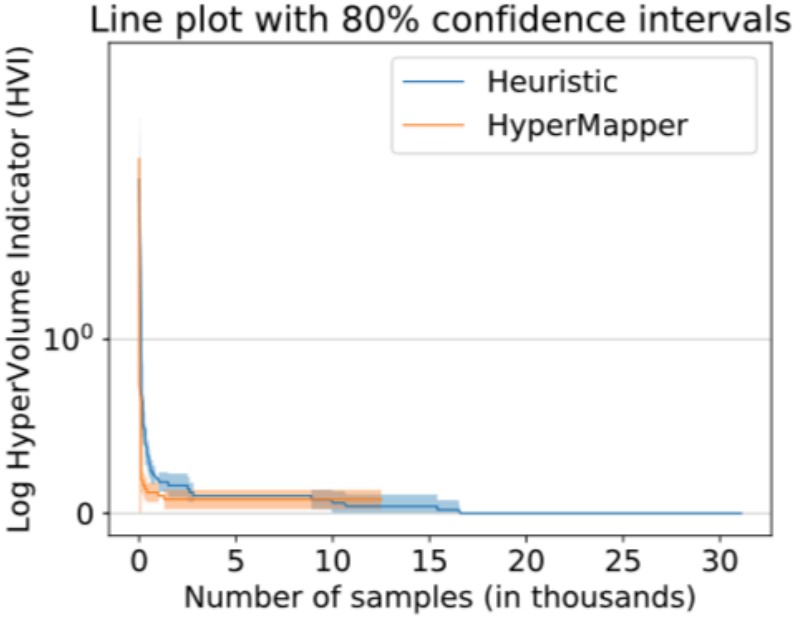
- 3 small benchmarks: compute real Pareto
- Real Pareto is exhaustive search
- Exhaustive: 6 to 12 hours on 16 cores
- Approximated Pareto is HyperMapper

# HVI Cycles/Logic Utilization

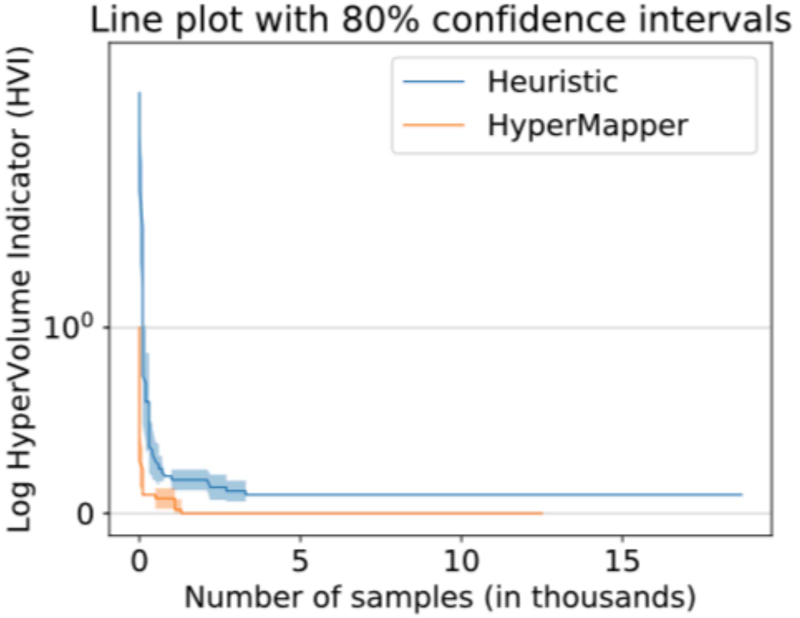
- Hypervolume indicator (HVI): scalar to compare Paretos
- Baseline is a pruning (based on heuristics) and 100K random sampling



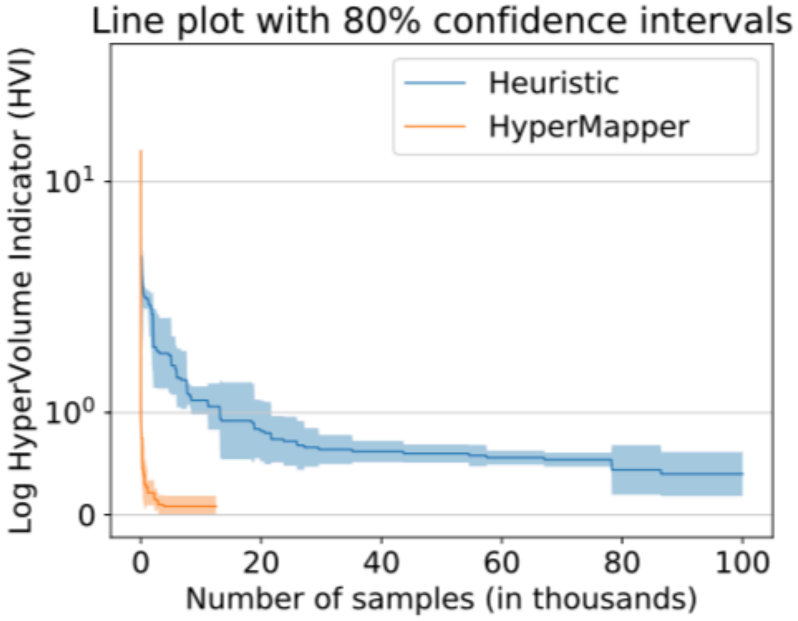
GEMM



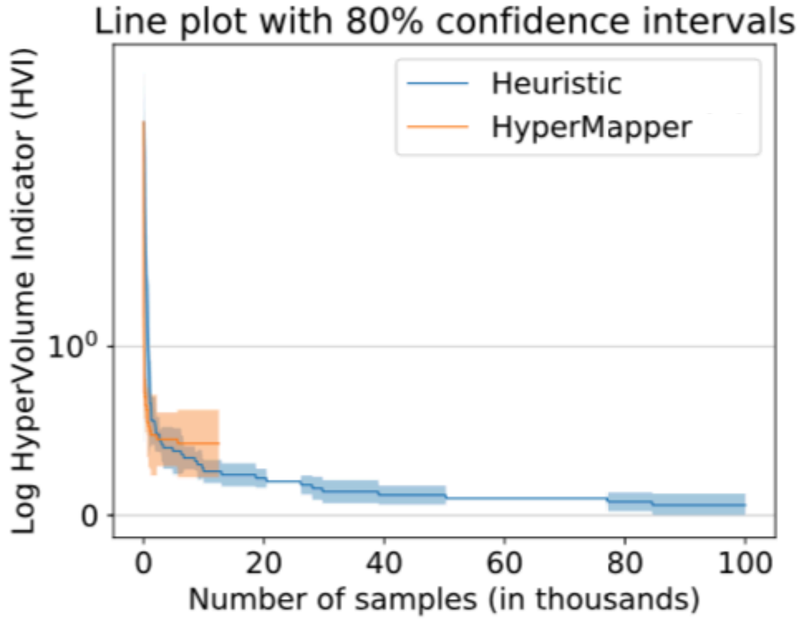
OuterProduct



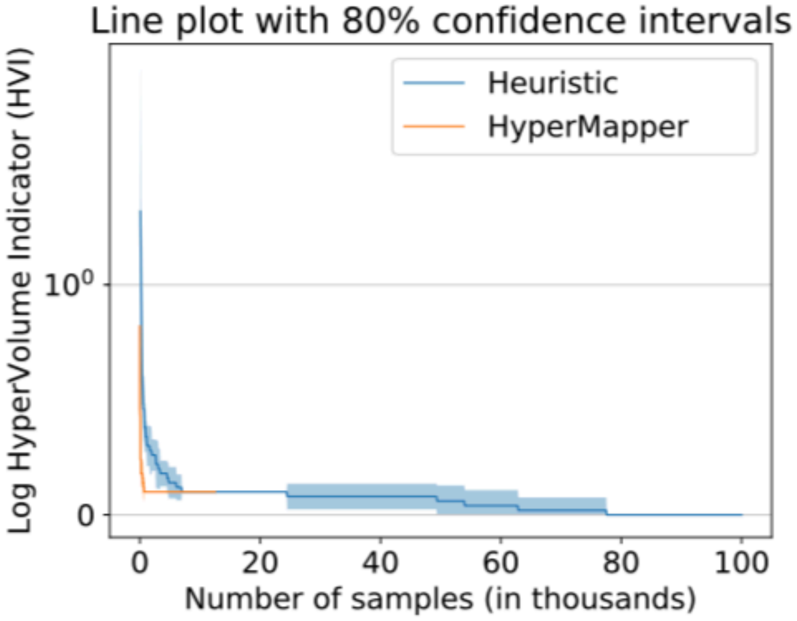
K-Means



GDA



T-PCH Q6



DotProduct

# Conclusion and Future Work

- Goal: HyperMapper is a multi-objective optimization framework
- Demonstrated on: hardware design, CV applications and robotics
- HyperMapper provides Intelligence Augmentation (IA):
    - White-box surrogate model: easy to understand and interpret
    - Prior: non-blind search - grey-box optimization approach

- Future:
    - Use of prior knowledge in the multi-objective case
        - For the moment only used in the DoE warm-up phase
    - Application of HyperMapper to the popular Halide CV language
    - Support other Black-box optimization algorithms
    - Support multi-fidelities to increase efficiency on DNNs and CV
    - Support batch evaluation for when a cluster is available

# Info on HyperMapper

- Join HyperMapper on **Slack**: hypermapper.slack.com
- **Tutorials**: to come
- **Repo**: https://github.com/luinardi/hypermapper
- **Wiki**: https://github.com/luinardi/hypermapper/wiki
- Early adopters:
  - Microsoft (DBMS)
  - Stanford (hardware design)
  - UCSD (Spector benchmarks)
  - UT Austin (Capri approximate computing)
  - ICL (computer vision and robotics)
  - Etc.

# Demo
# HyperMapper