Multi-Objective Optimization for High-Level Synthesis

Marcela Zuluaga [1]
Andreas Krause [1]
Guillaume Sergent [2]
Markus Püschel [1]

[1] *Department of Computer Science, ETH Zurich*

[2] *ENS de Lyon*

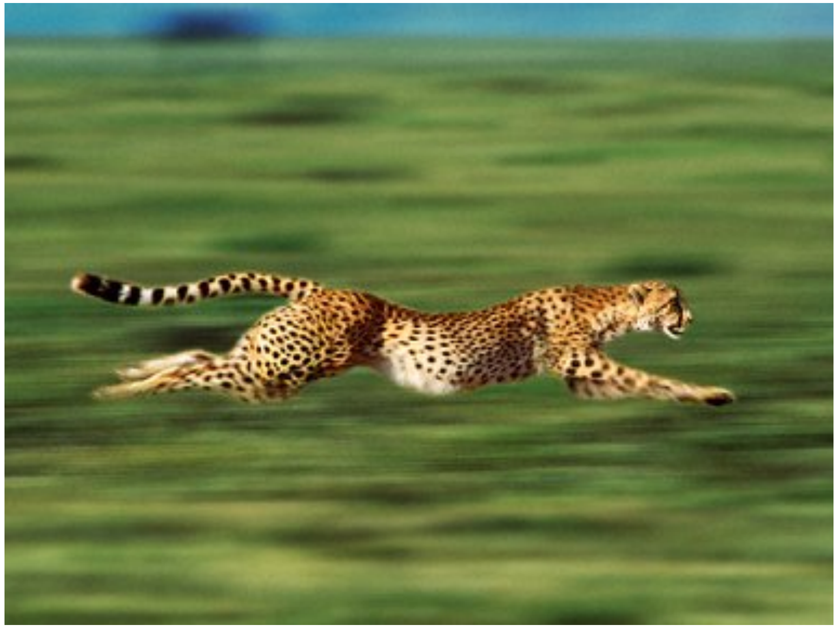
SPIRAL
www.spiral.net

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

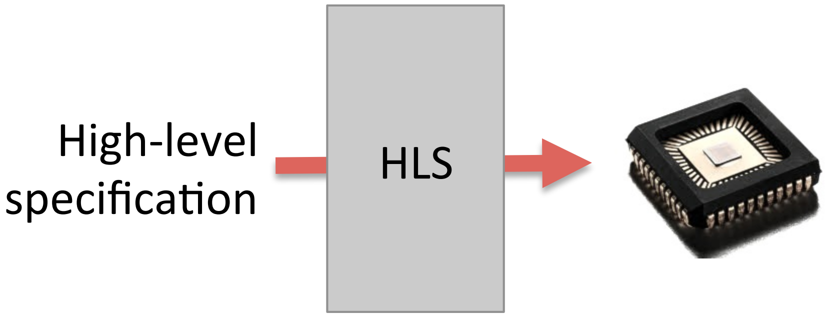
High Level Synthesis 4 High Performance Computing

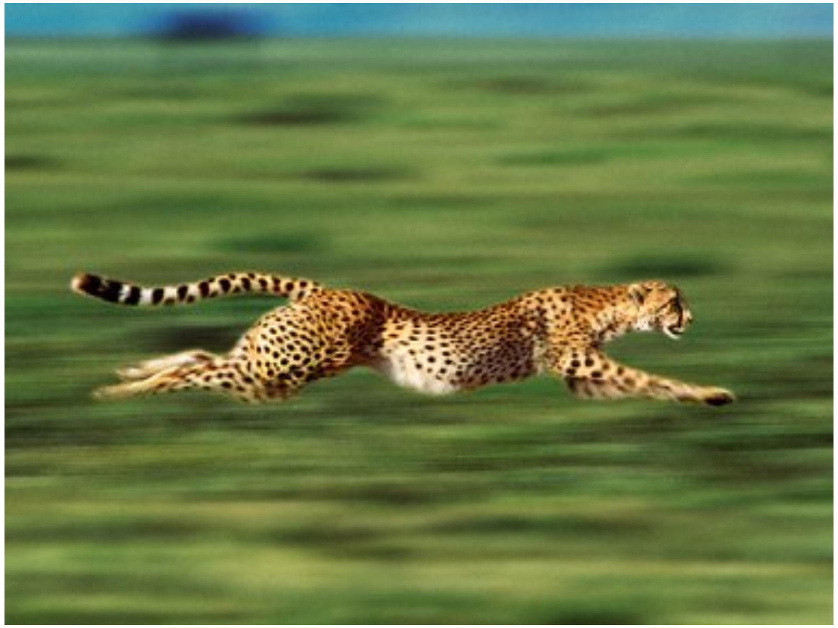
High Level Synthesis 4 High Performance Computing

High Level Synthesis 4 High Performance Computing



High Level Synthesis 4 High Performance Computing





High Level 4 High Performance Synthesis Computing

High-level specification

HLS

Best

Throughput

Latency

Area

Energy

High Level Synthesis 4 High Performance Computing

High-level specification → HLS →

→ **Best**

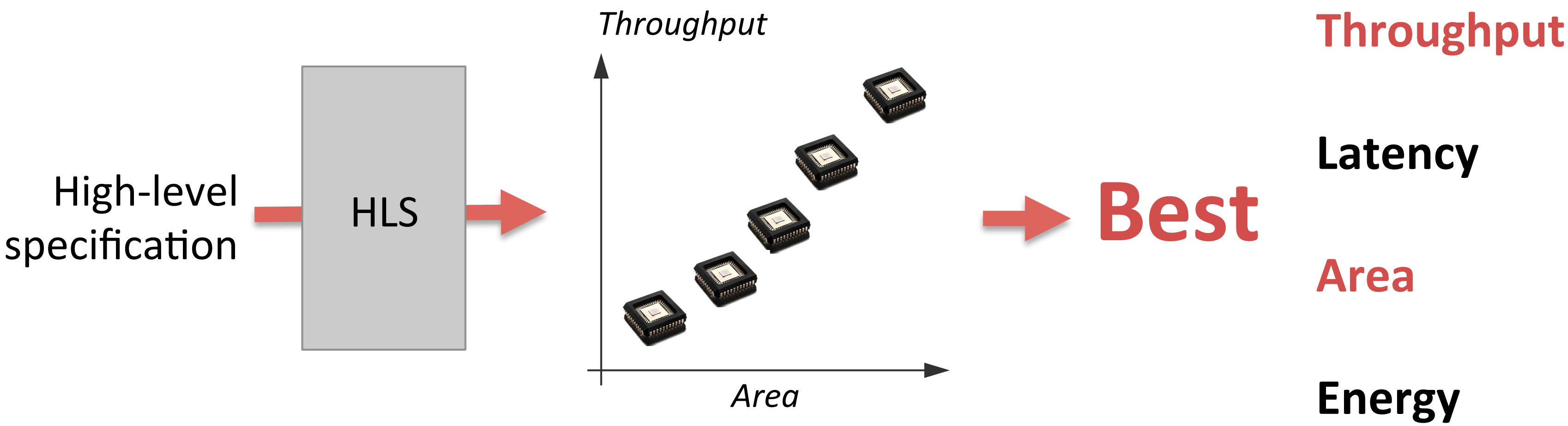
Throughput

Latency

Area

Energy

High Level Synthesis 4 High Performance Computing



IP Generator for Sorting Networks

http://www.spiral.net/hardware/sort/sort.html

parameter	value	range	explanation
Problem specification			
input set size	64	4–16384	number of samples to sort (?)
data type	fixed point		fixed or floating point (?)
	16 bits	4–32 bits	fixed point precision (?)
Parameters controlling implementation			
architecture	SN1		sorting network algorithm (?)
streaming width	2		number of samples per cycle (?)
iterative reuse	fully streaming		iterative or fully streaming (?)
BRAM budget	1000		maximum # of BRAMs to utilize (-1 for no limit) (?)

✔

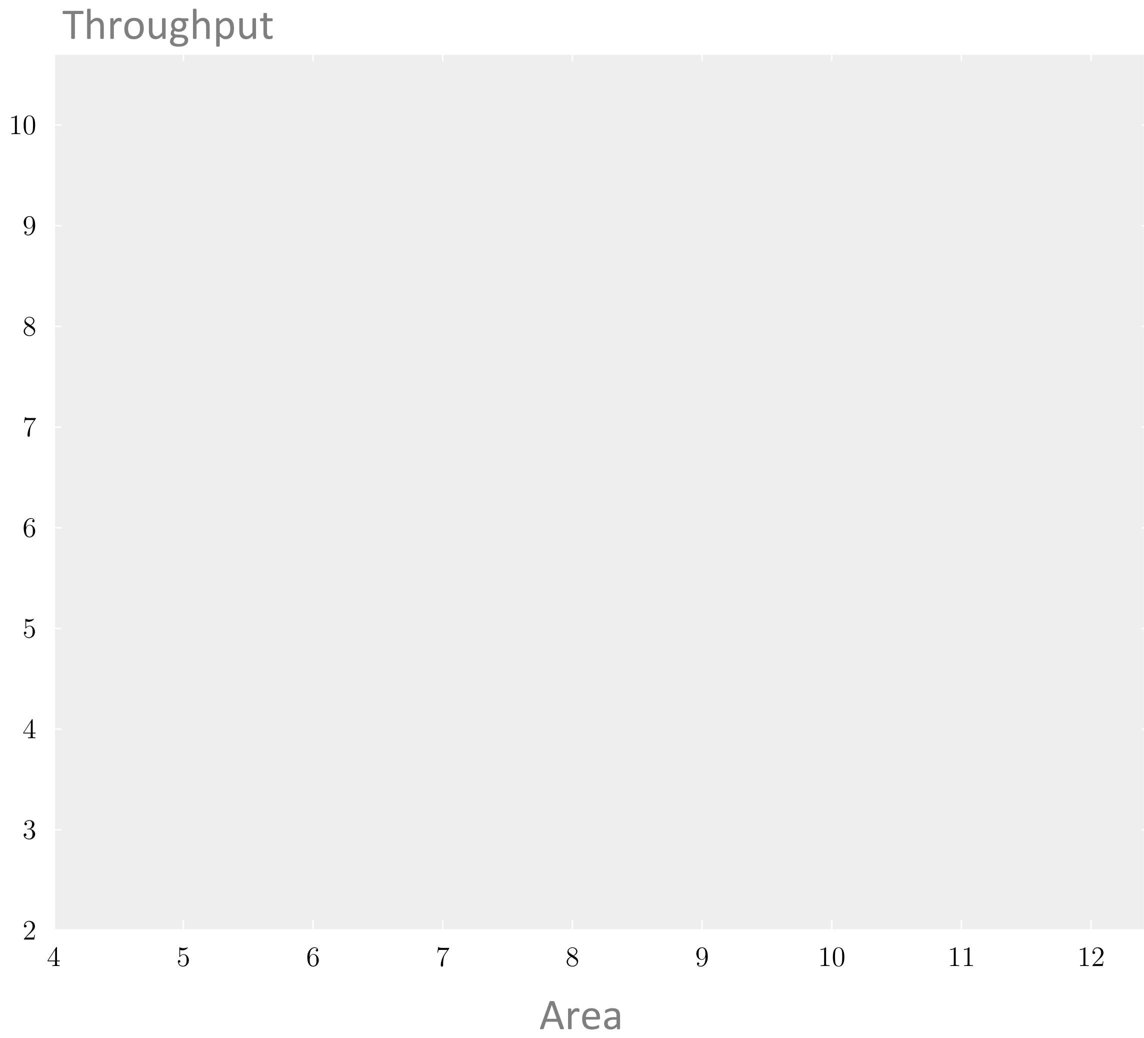
?

Generate Verilog

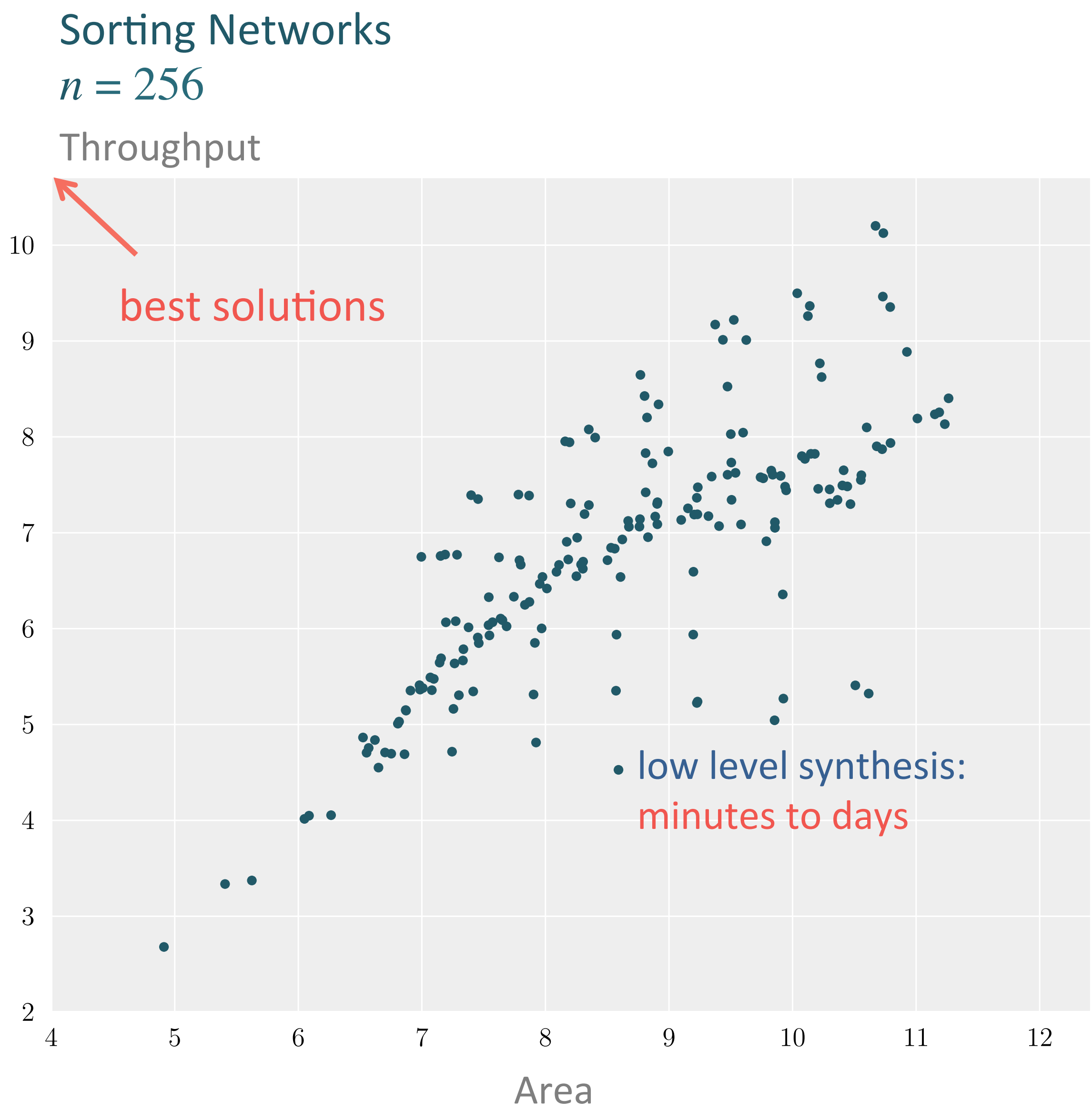
Exploring the Design Space

Sorting Networks

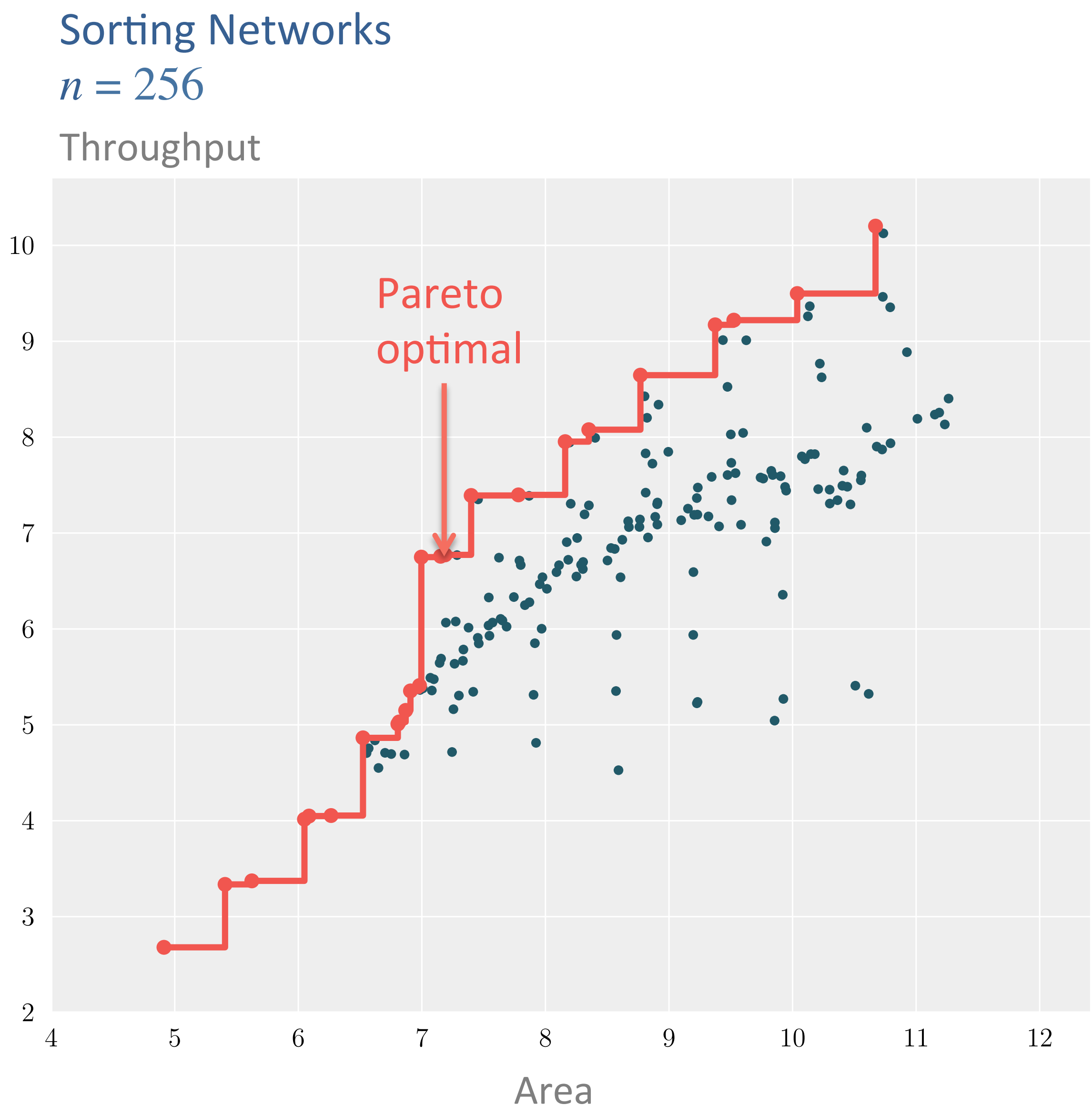
$n = 256$



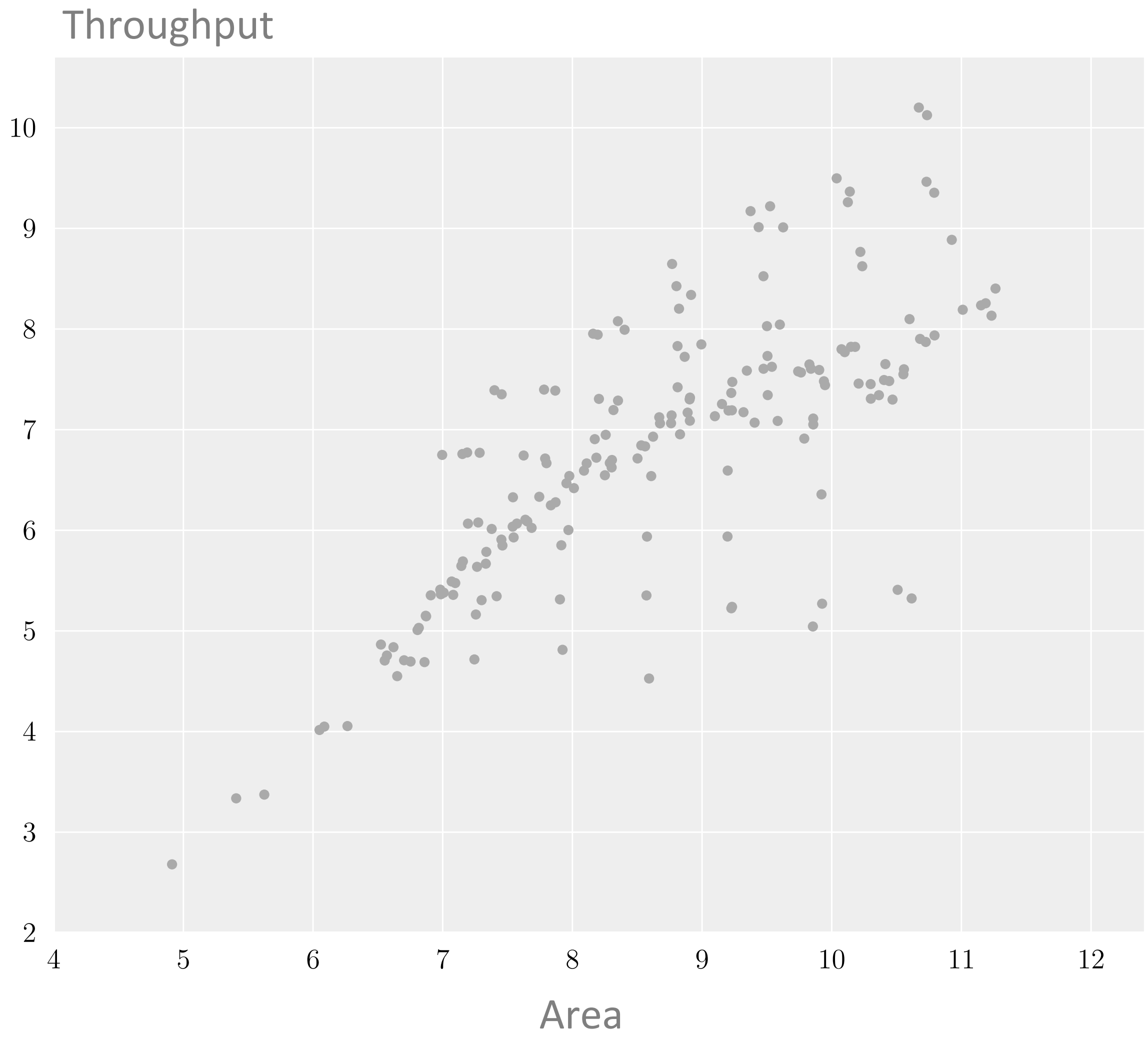
Exploring the Design Space



Exploring the Design Space

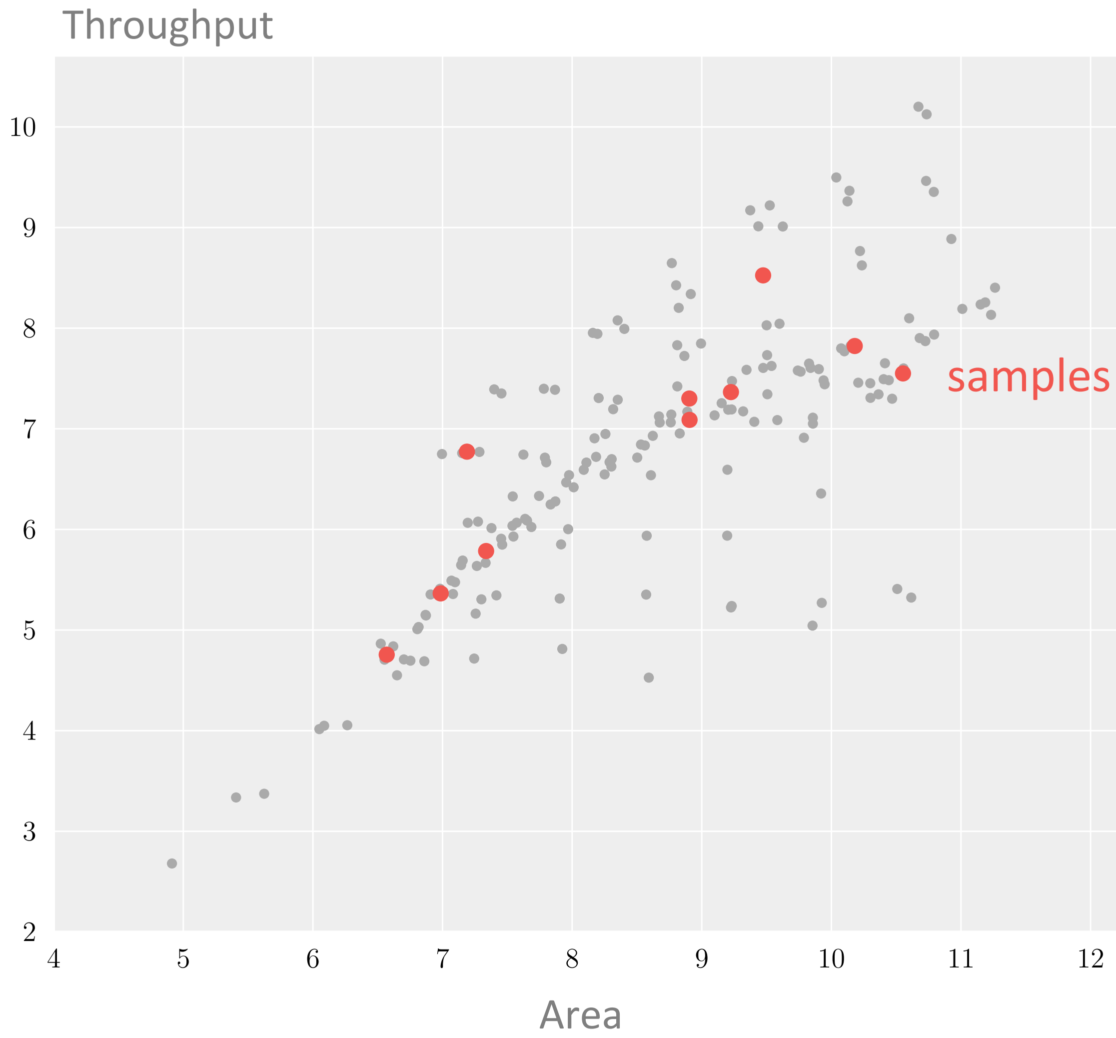


Goal



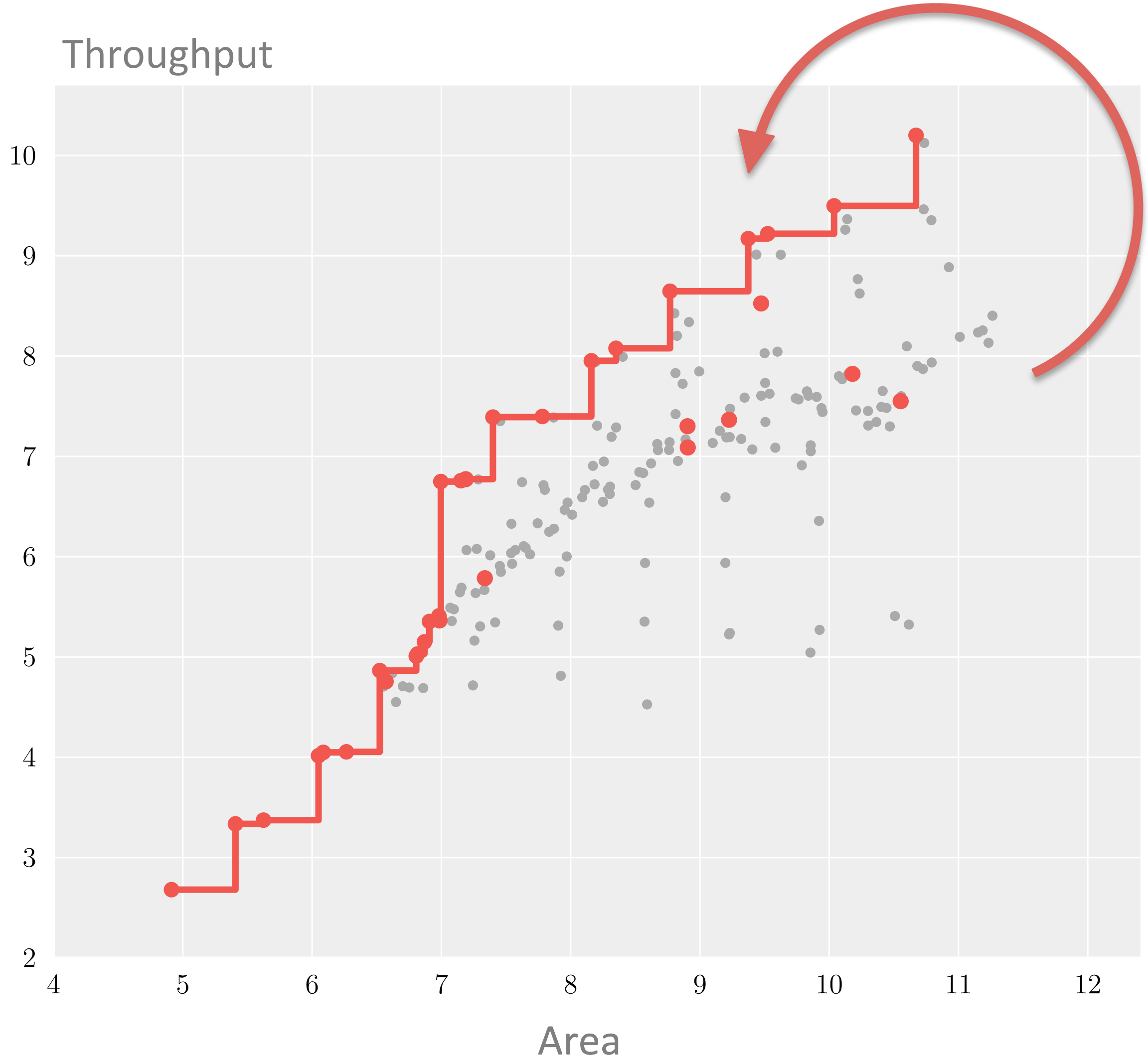
Goal

Sample as few designs as possible



Goal

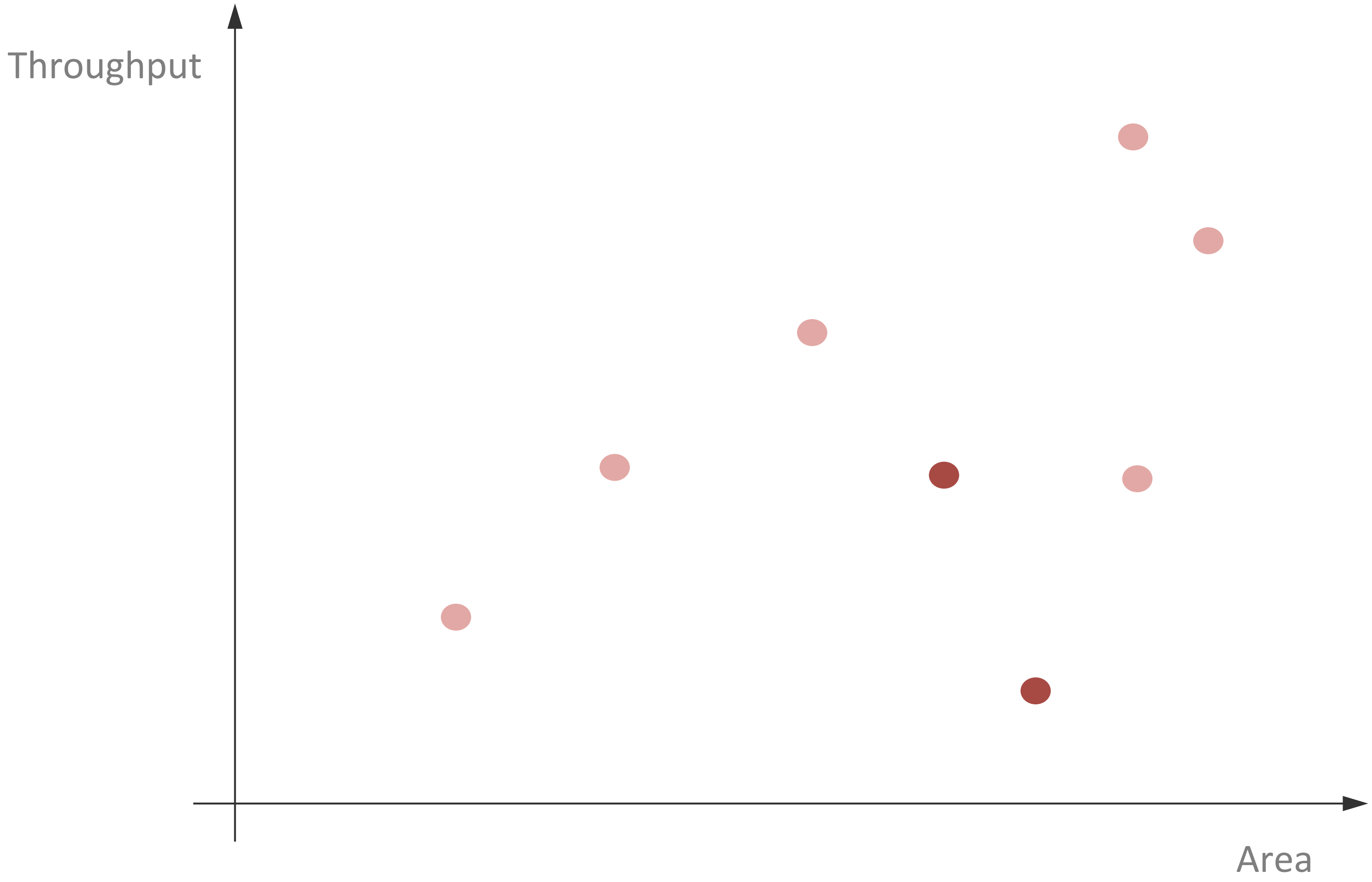
Sample as few designs as possible *to predict* Pareto optimal designs



Running the Algorithm

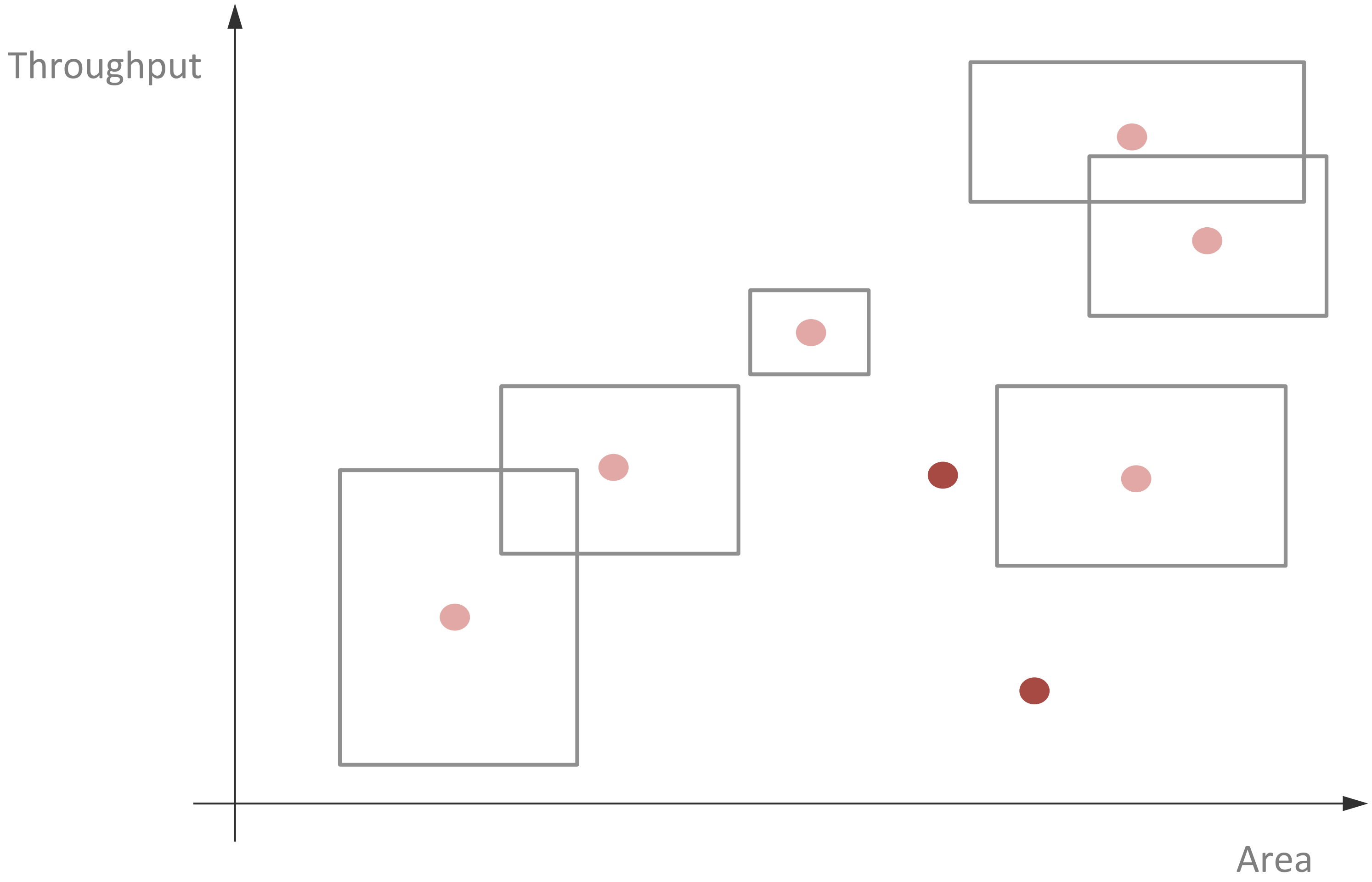
Initialization

Running the Algorithm
Modeling

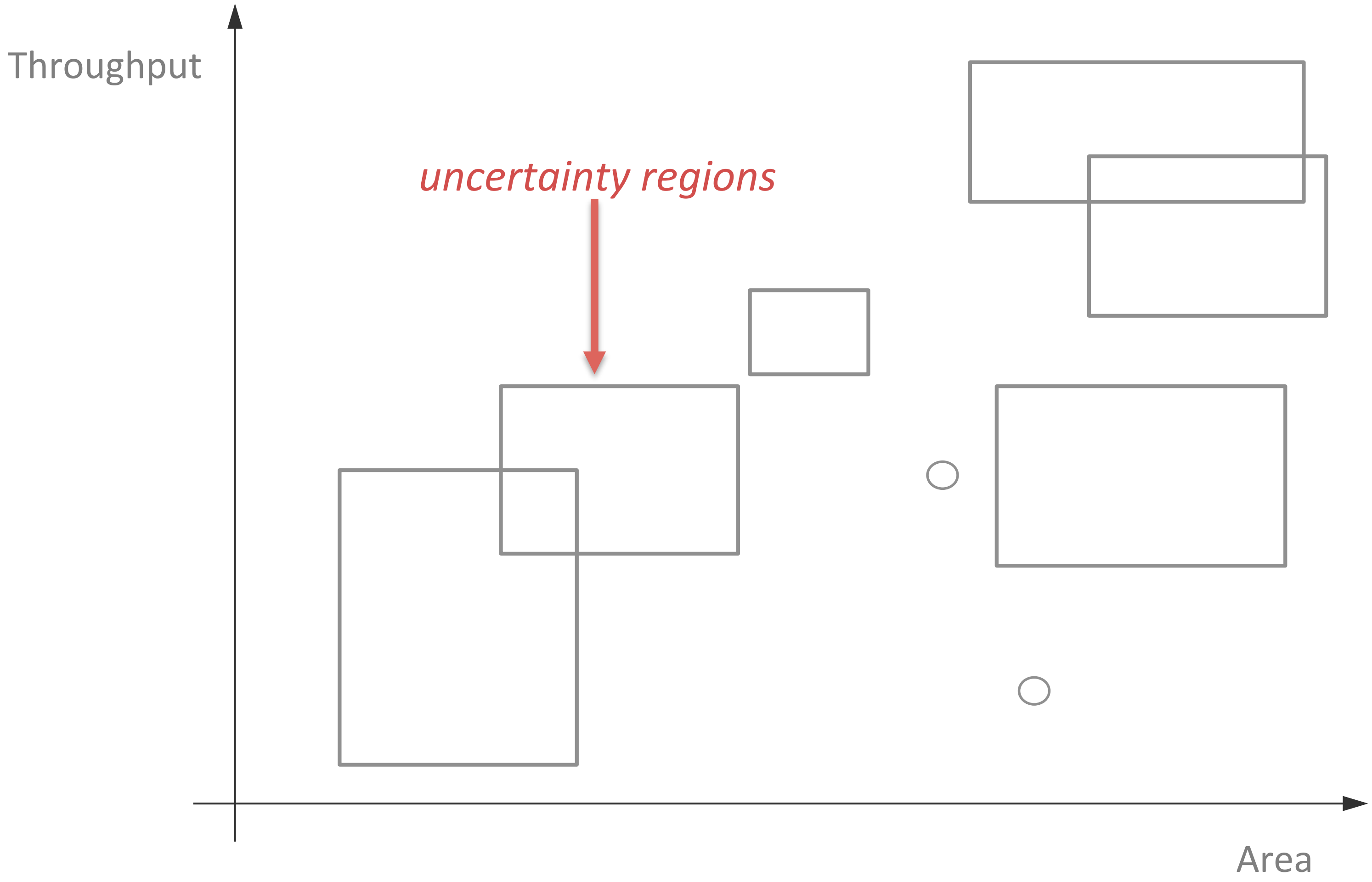


Running the Algorithm

Modeling

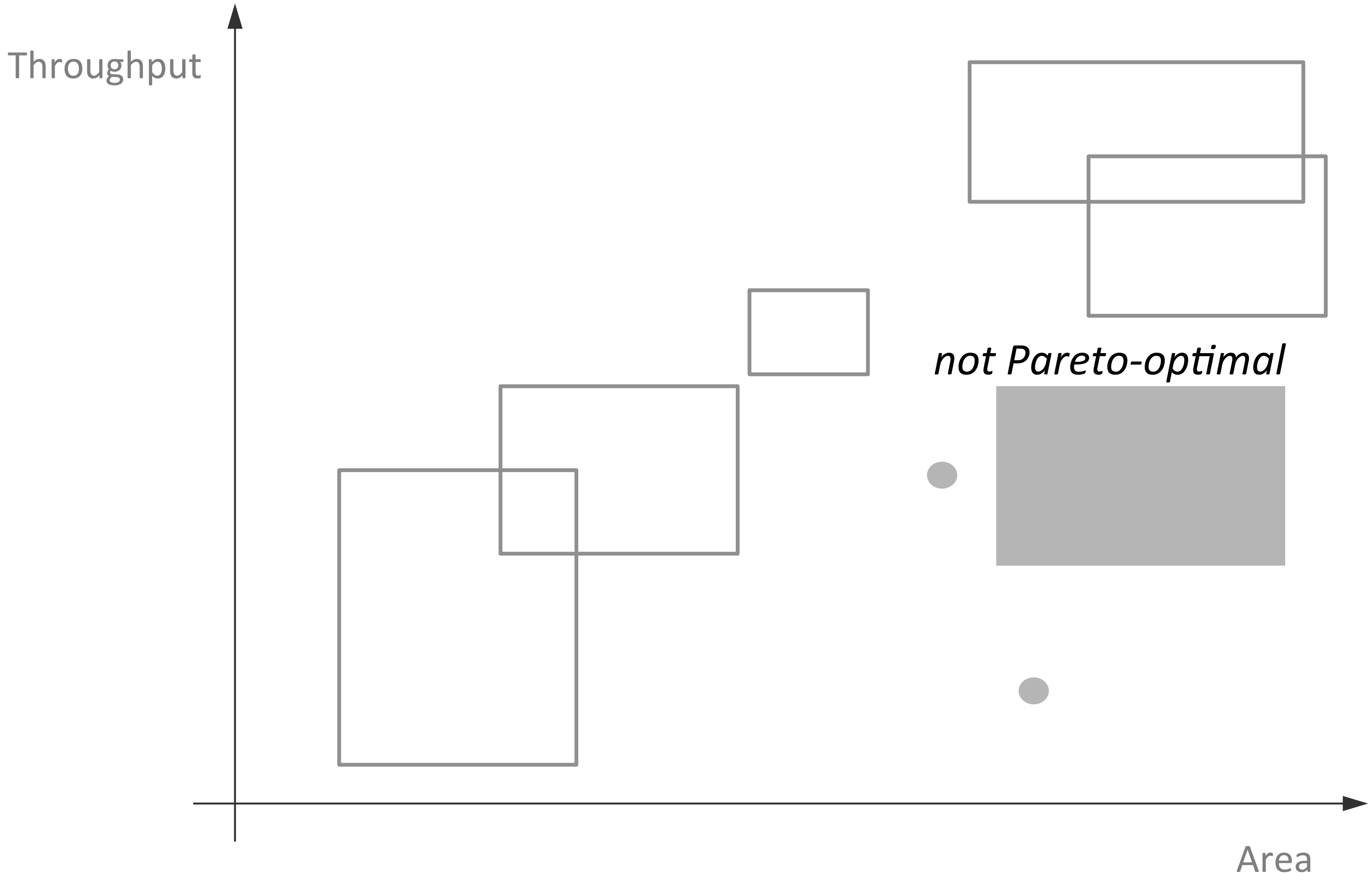


Running the Algorithm
Modeling



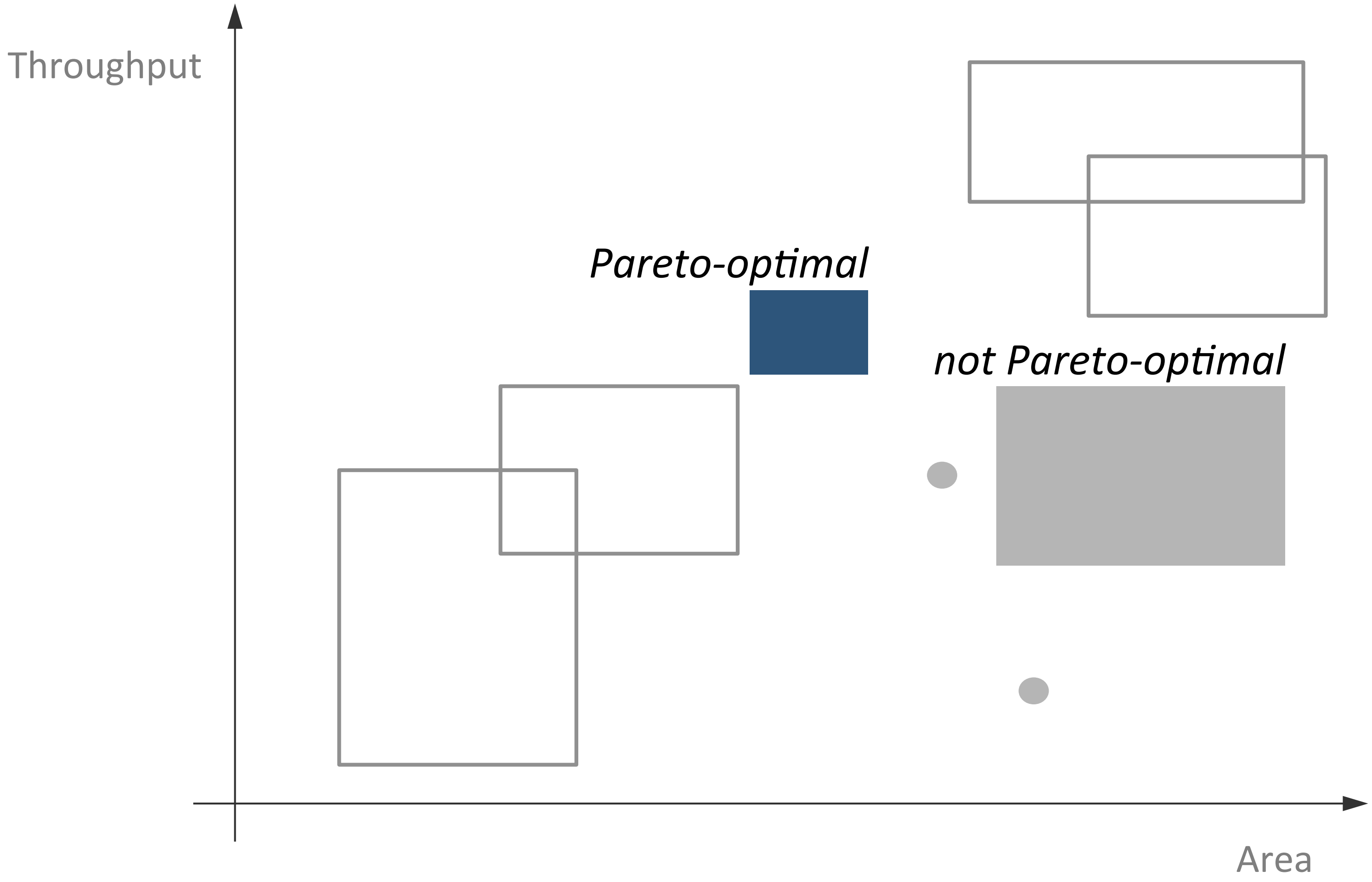
Running the Algorithm

Classification



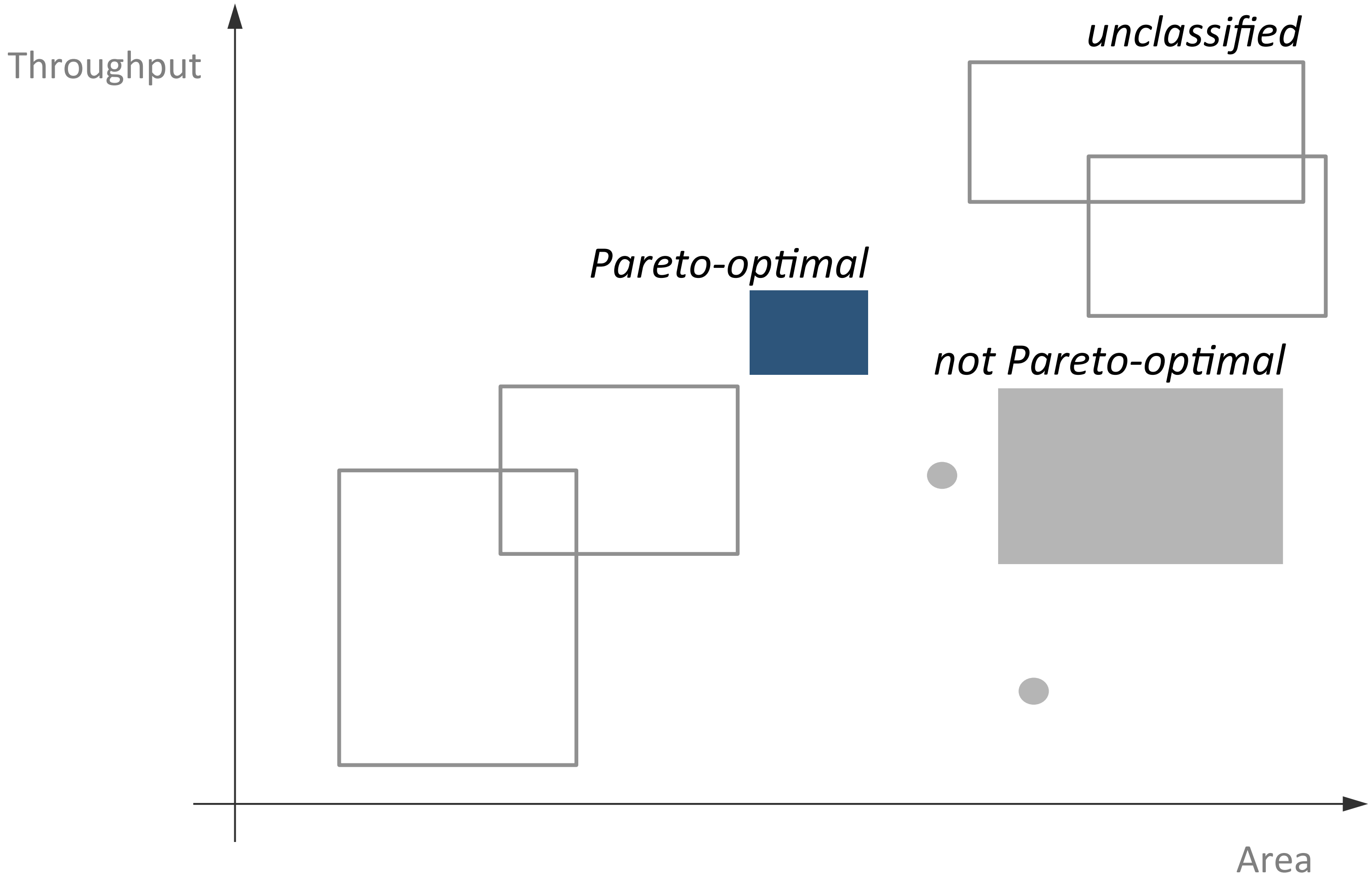
Running the Algorithm

Classification



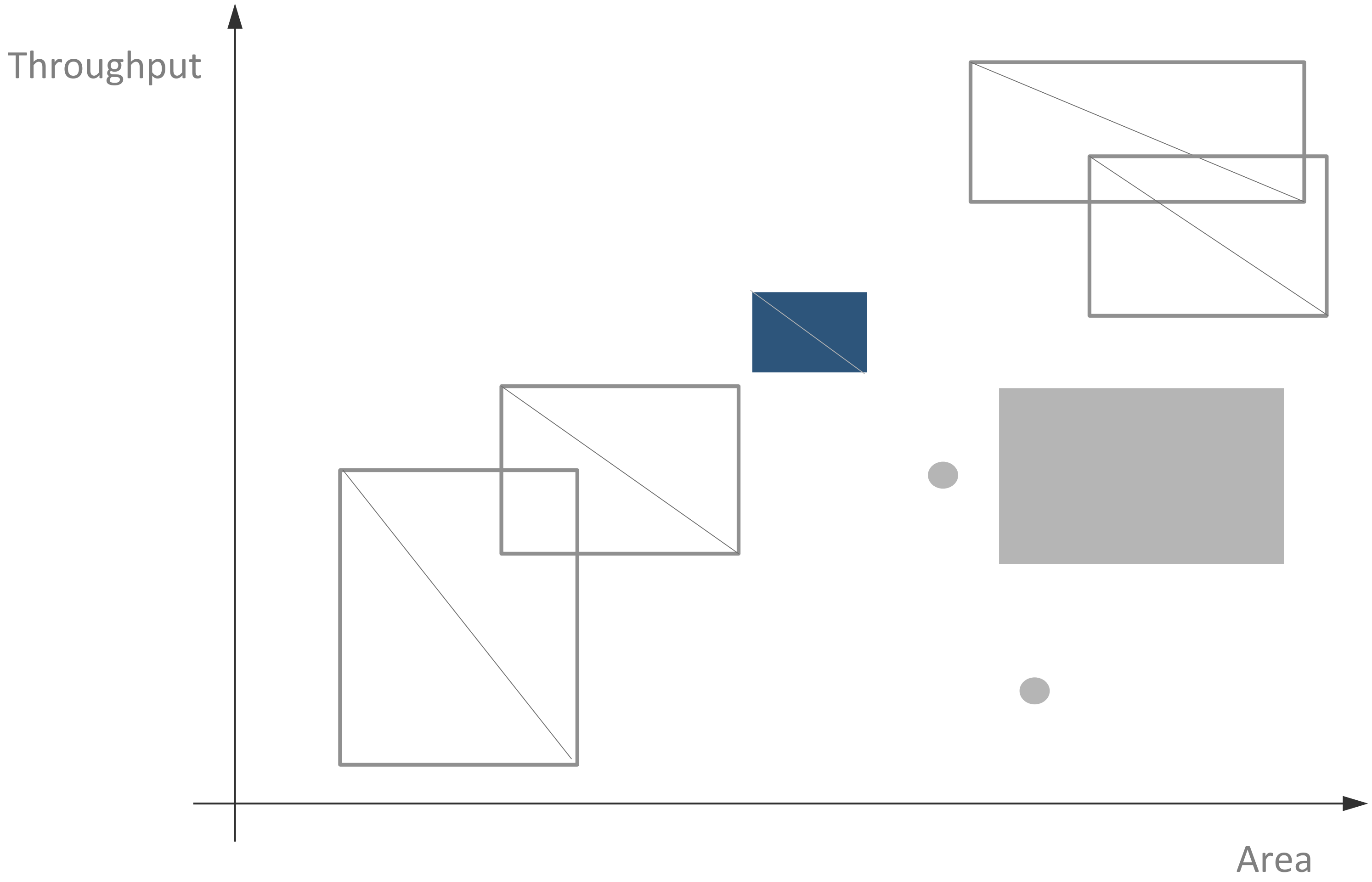
Running the Algorithm

Classification



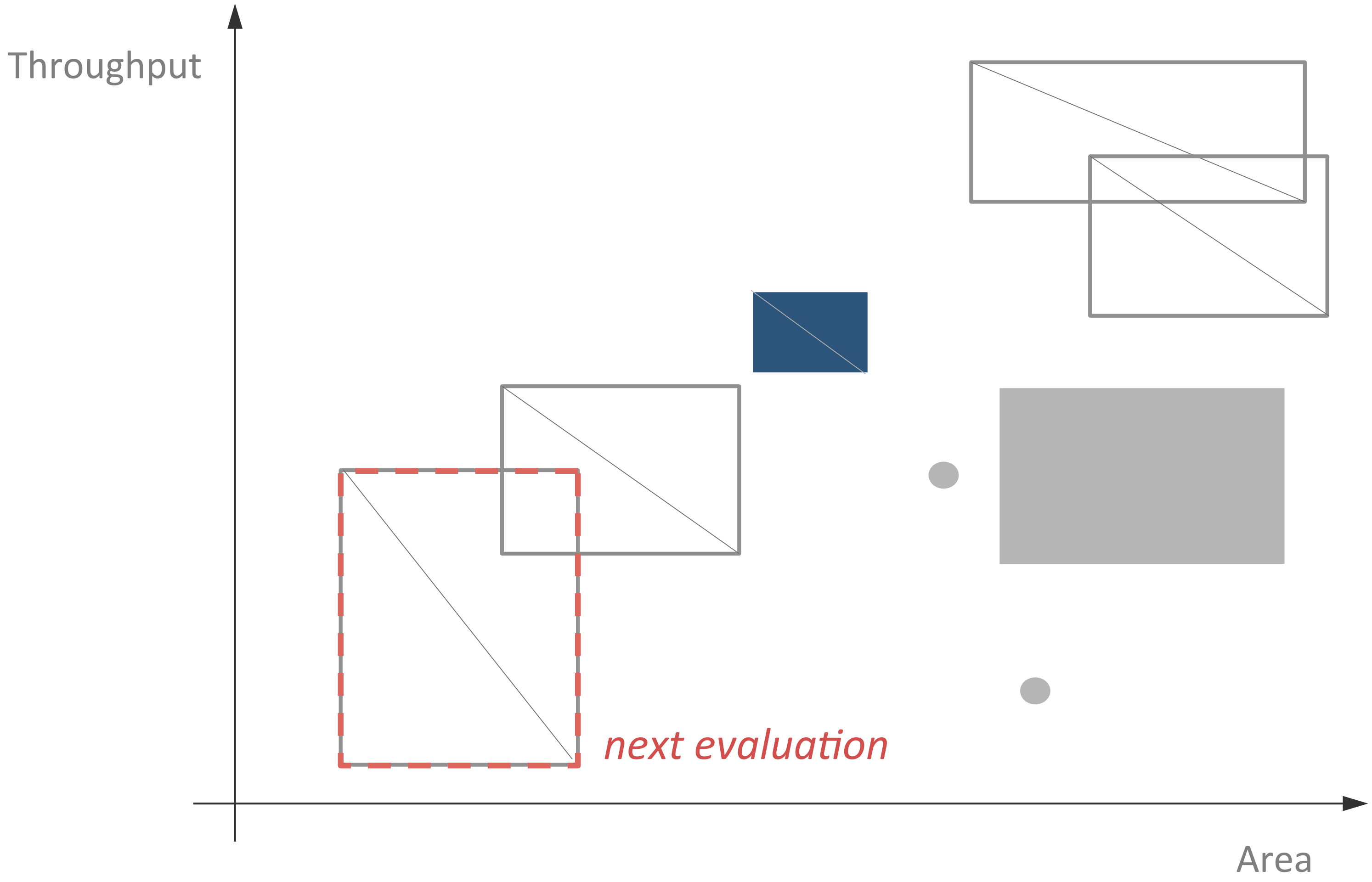
Running the Algorithm

Sampling



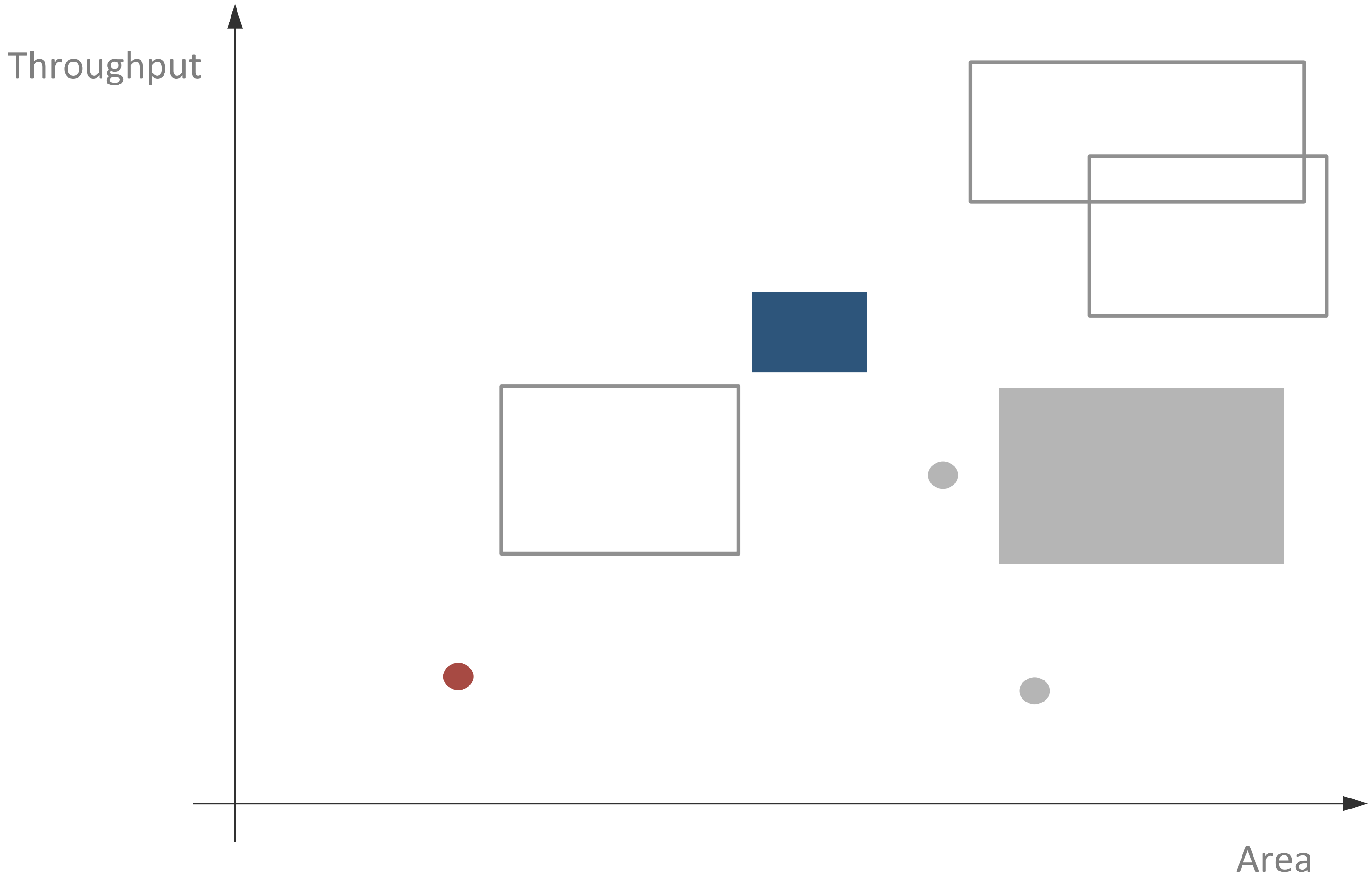
Running the Algorithm

Sampling

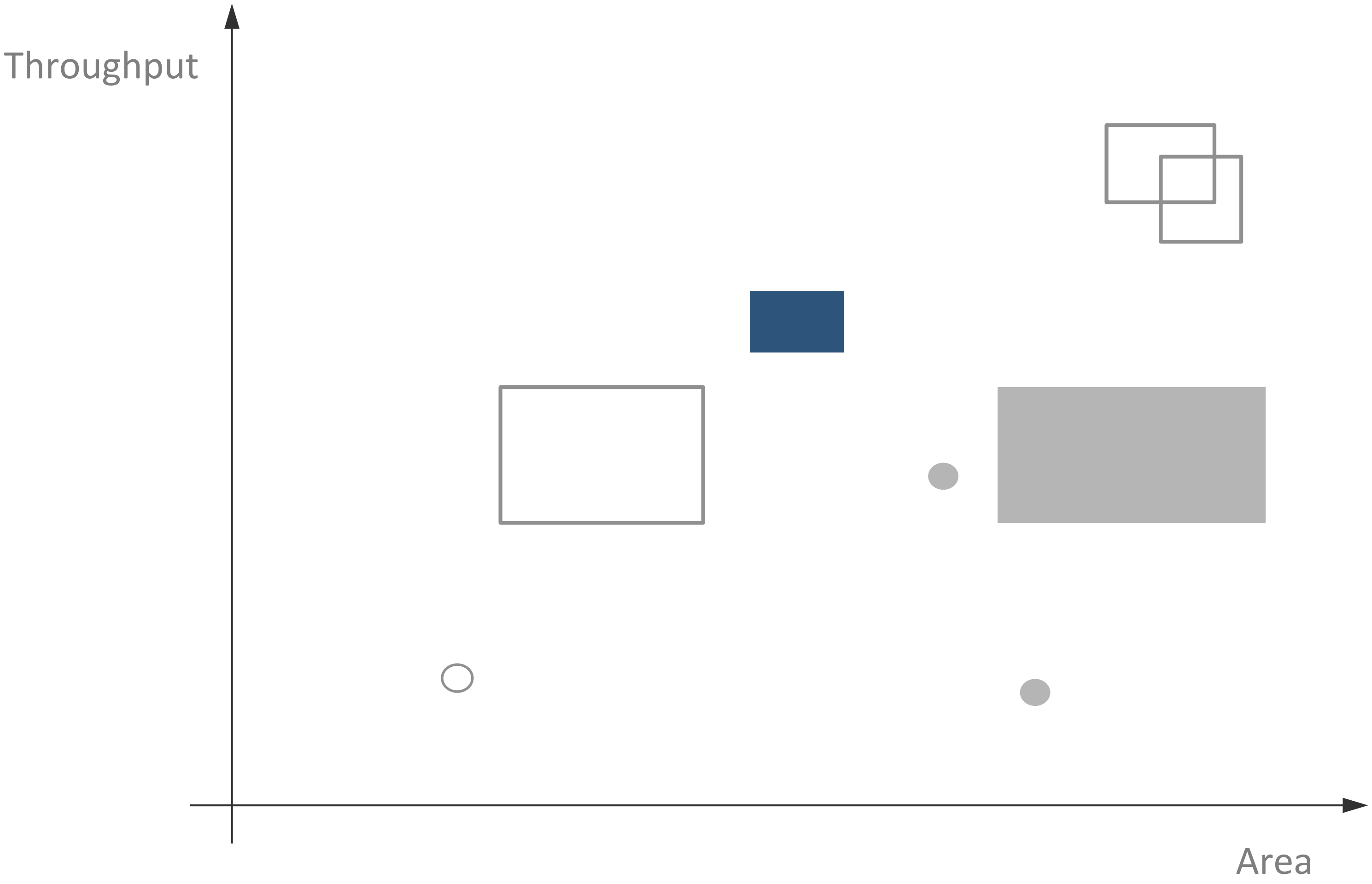


Running the Algorithm

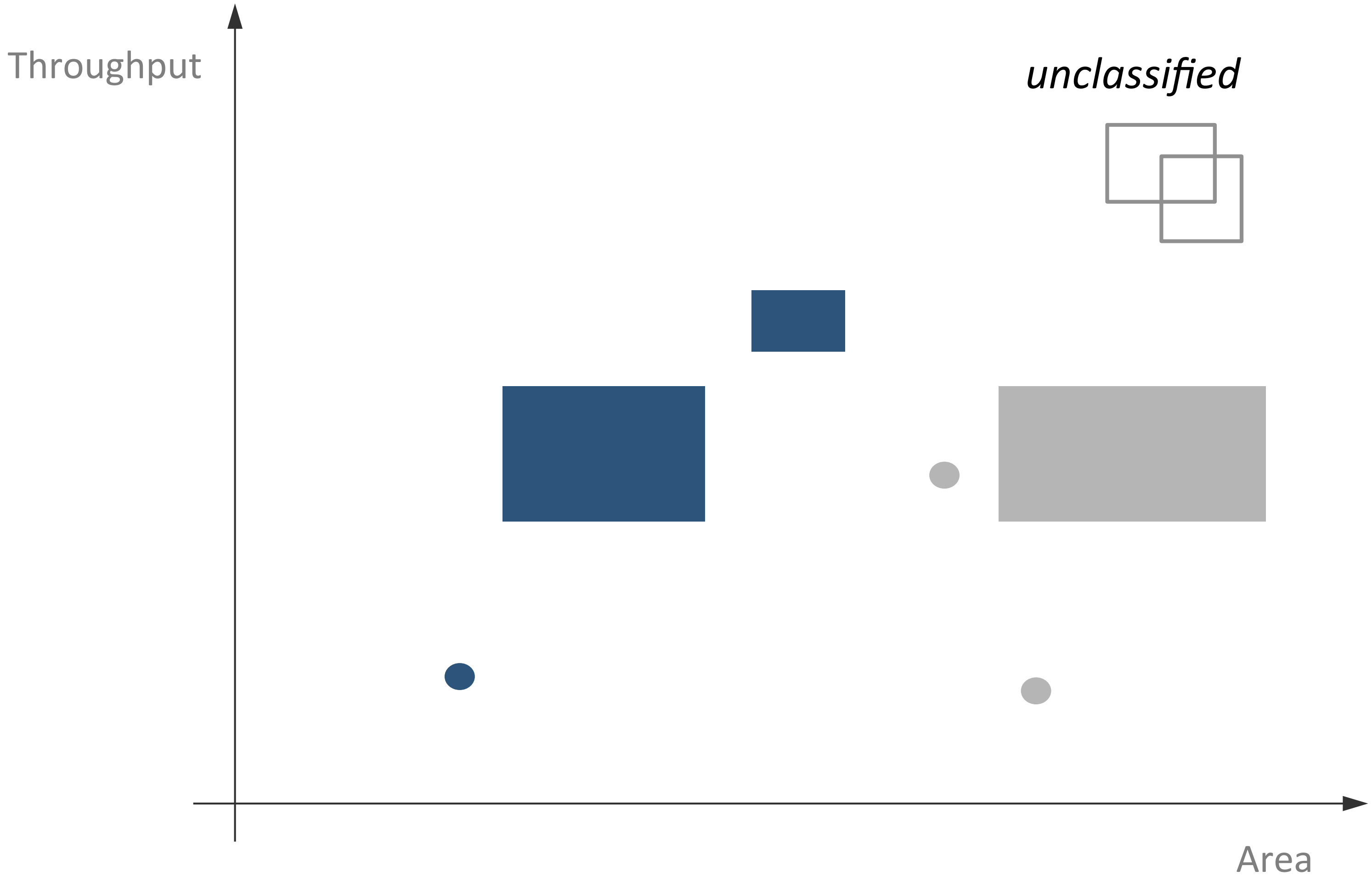
Evaluating the sample



Running the Algorithm
Modeling

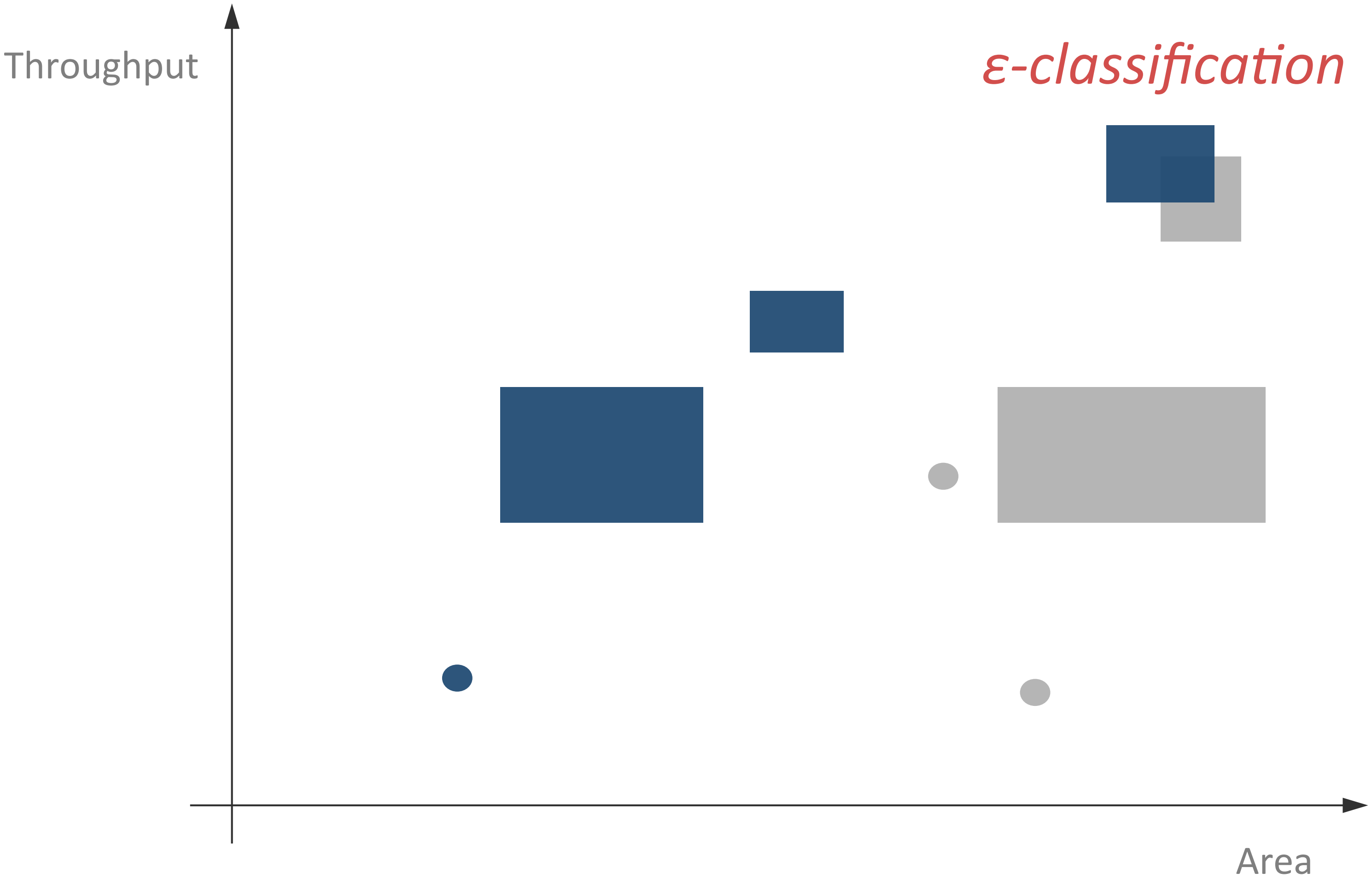


Running the Algorithm
Classification



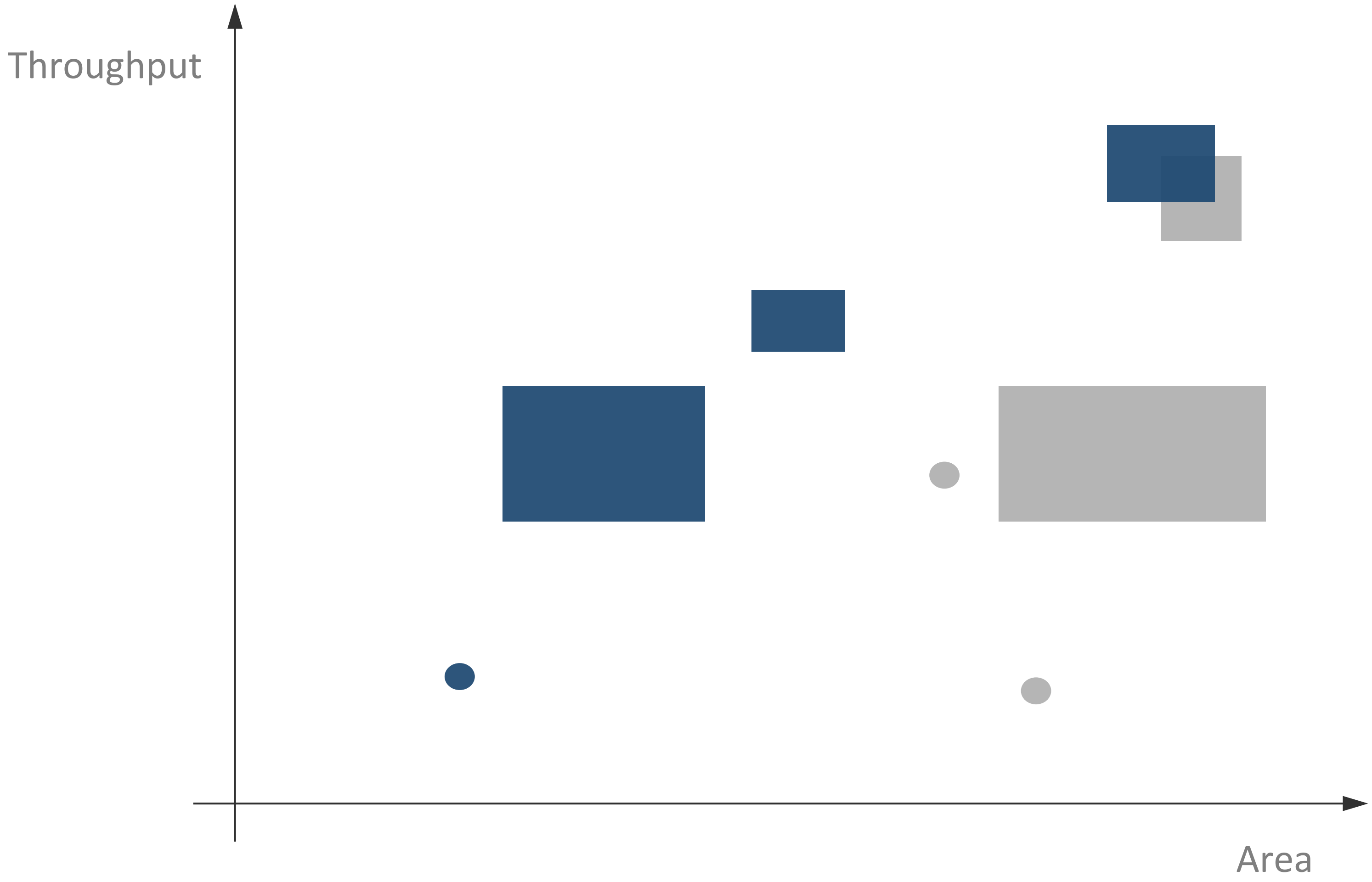
Running the Algorithm

Reducing training cost



Running the Algorithm

Termination: all points classified



The PAL Algorithm

Input: design space E; GP prior $\mu_{0,i}, \sigma_0, k_i$ for all $1 \leq i \leq n$; ϵ; β_t for $t \in \mathbb{N}$

Output: predicted-Pareto set $\hat{P}$

$P_0 = \emptyset, N_0 = \emptyset, U_0 = E$ {classification sets}

$S_0 = \emptyset$ {evaluated set}

$R_0(\boldsymbol{x}) = \mathbb{R}^n$ for all $\boldsymbol{x} \in E$

$t = 0$

repeat

— *Modeling* —

Obtain $\boldsymbol{\mu}_t(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$ $\{\boldsymbol{\mu}_t(\boldsymbol{x}) = \boldsymbol{y}(\boldsymbol{x})$ and $\boldsymbol{\sigma}_t(\boldsymbol{x}) = 0$ for all $\boldsymbol{x} \in S_t\}$

$R_t(\boldsymbol{x}) = R_{t-1}(\boldsymbol{x}) \cap Q_{\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t, \beta_{t+1}}(\boldsymbol{x})$ for all $\boldsymbol{x} \in E$

— *Classification* —

$P_t = P_{t-1},\ N_t = N_{t-1},\ U_t = U_{t-1}$

for all $\boldsymbol{x} \in U_t$ **do**

if there is no $\boldsymbol{x}' \neq \boldsymbol{x}$ such that $\min(R_t(\boldsymbol{x})) + \boldsymbol{\epsilon} \preceq \max(R_t(\boldsymbol{x}')) - \boldsymbol{\epsilon}$ **then**

$P_t = P_t \cup \{\boldsymbol{x}\}$, $U_t = U_t \setminus \{\boldsymbol{x}\}$

else if there exists $\boldsymbol{x}' \neq \boldsymbol{x}$ such that $\max(R_t(\boldsymbol{x})) - \boldsymbol{\epsilon} \preceq \max(R_t(\boldsymbol{x}')) + \boldsymbol{\epsilon}$ **then**

$N_t = N_t \cup \{\boldsymbol{x}\}$, $U_t = U_t \setminus \{\boldsymbol{x}\}$

end if

end for

— *Sampling* —

Find $w_t(\boldsymbol{x})$ for all $\boldsymbol{x} \in (U_t \cup P_t) \setminus S_t$

Choose $\boldsymbol{x}_{t+1} = \arg\max_{x \in (U_t \cup P_t) \setminus S_t} \{w_t(\boldsymbol{x})\}$

$t = t + 1$

Sample $\boldsymbol{y}_t(\boldsymbol{x}_t) = \boldsymbol{f}(\boldsymbol{x}_t) + \boldsymbol{\nu}_t$

until $U_t = \emptyset$

$\hat{P} = P_t$

Theoretical Guarantee

Given a target error η, PAL is guaranteed to stop in less than T iterations:

Theorem 1. *Let* $\delta \in (0,1)$. *Running* PAL *with* $\beta_t = 2\log(n|E|\pi^2 t^2/(6\delta))$, *the following holds with probability* $1-\delta$.
To achieve a maximum hypervolume error of η, *it is sufficient to choose*

$$\epsilon = \frac{\eta(n-1)!}{2na^{n-1}},$$

where $a = \max_{\boldsymbol{x}\in E, 1\le i\le n}\{\sqrt{\beta_1 k_i(\boldsymbol{x},\boldsymbol{x})}\}$.
In this case, the algorithm terminates after at most T *iterations, where* T *is the smallest number satisfying*

$$\boxed{\sqrt{\frac{T}{C_1\beta_T\gamma_T}} \ge \frac{na^{n-1}}{\eta(n-1)!}}.$$

Here, $C_1 = 8/\log(1-\sigma^{-2})$, *and* γ_T *depends on the type of kernel used.*

Related Work

Evolutionary Algorithms

J. Knowles. *ParEGO: a Hybrid Algorithm with On-line Landscape Approximation for Expensive Multi-objective Optimization Problems. 2006*

M. Emmerich, K. Giannakoglou, and B. Naujoks. *Single- and Multi-objective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. 2006*

Scalarization to Single-Obective

Q. Zhang, W. Liu, E. Tsang, and B. Virginas. *Expensive Multi-objective Optimization by MOEA/D with Gaussian Process Model. 2010*

Heuristic-Based Methods

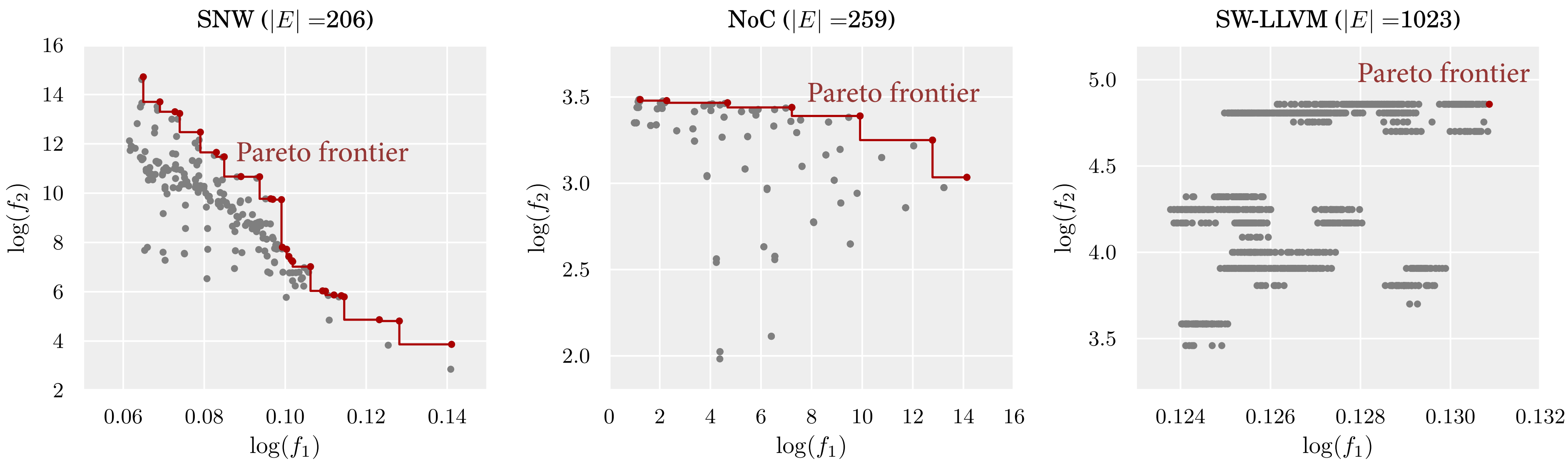
G. Palermo, C. Silvano, and V. Zaccaria. *ReSPIR: A Response Surface-Based Pareto Iterative Refinement for Application-Specific Design Space Exploration. 2009*

Gaussian Process Optimization

N. Srinivas, A. Krause, S. Kakade, and M. Seeger. *Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. 2010*

Experiments

Data sets



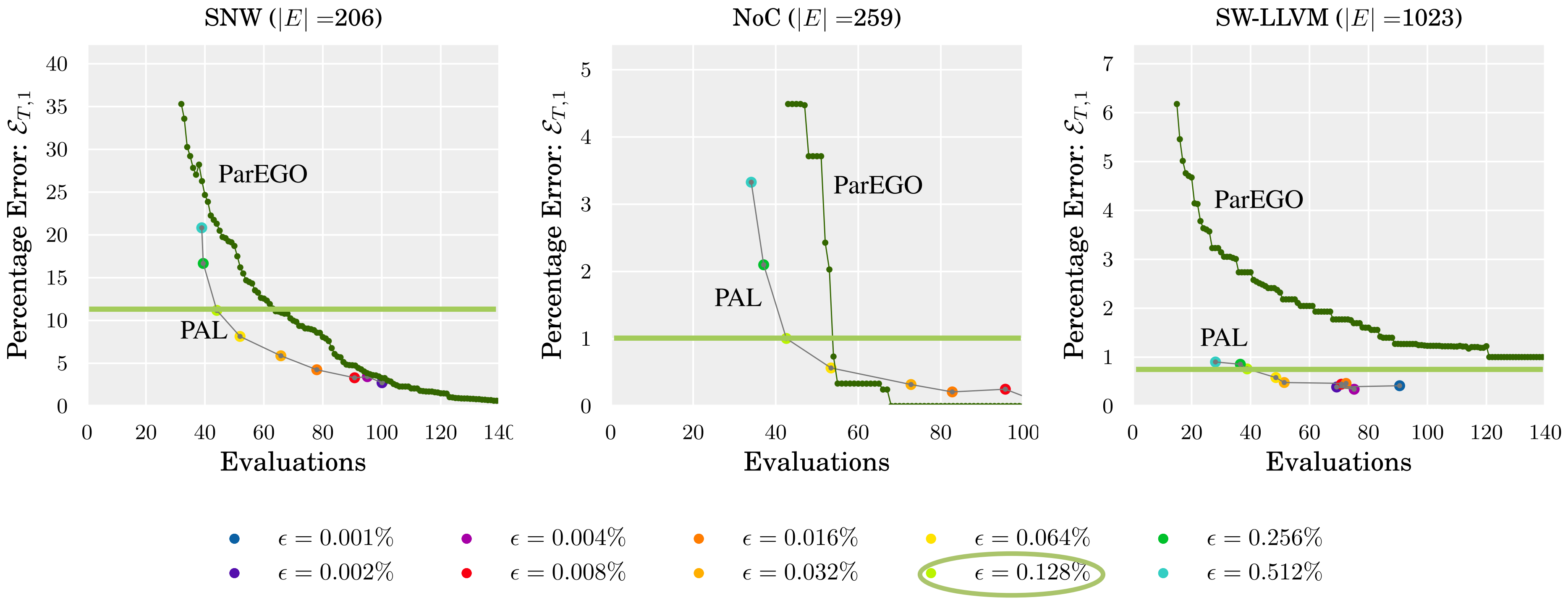
Marcela Zuluaga, Andreas Krause, Peter Milder, Markus Püschel. *Streaming Sorting Networks. DAC 2012*

Oscar Almer, Nigel Topham,, Björn Franke. *A Learning- Based Approach to the Automated Design of MP-SoC Networks. ARCS 2011*

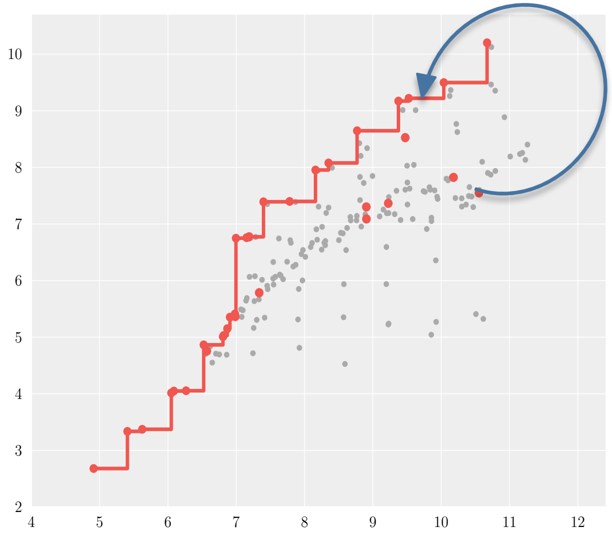
Predicting Performance via Automated Feature-Interaction Detection . *N. Siegmund, S. S. Kolesnikov, C. Kastner , S. Apel, D. Batory, M. Rosenmuller, and G. Saake. ICSI 2012*

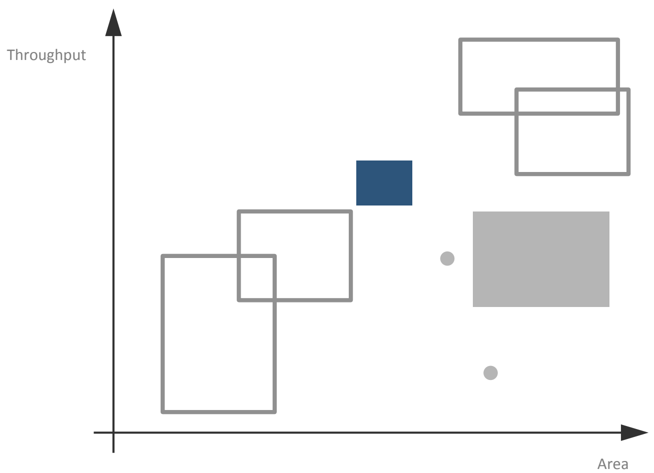
Experiments

Results and comparison with ParEGO



Conclusions





Machine learning technique to *predict Pareto* optimal solutions

- Gaussian process modeling
- Few evalutations: *"smart" sampling*
- Stopping criteria
- Convergence guarantees

References:

- "Smart" Design Space Sampling to Predict Pareto-Optimal Solutions. *Marcela Zuluaga, Andreas Krause, Peter Milder, Markus Püschel. LCTES 2012.*
- Pareto Active Learning. *Marcela Zuluaga, Andreas Krause, Guillaume Sergent, Markus Püschel. To appear in ICML 2013.*

More machine learning algorithms and IP genarators at *http://www.spiral.net/*