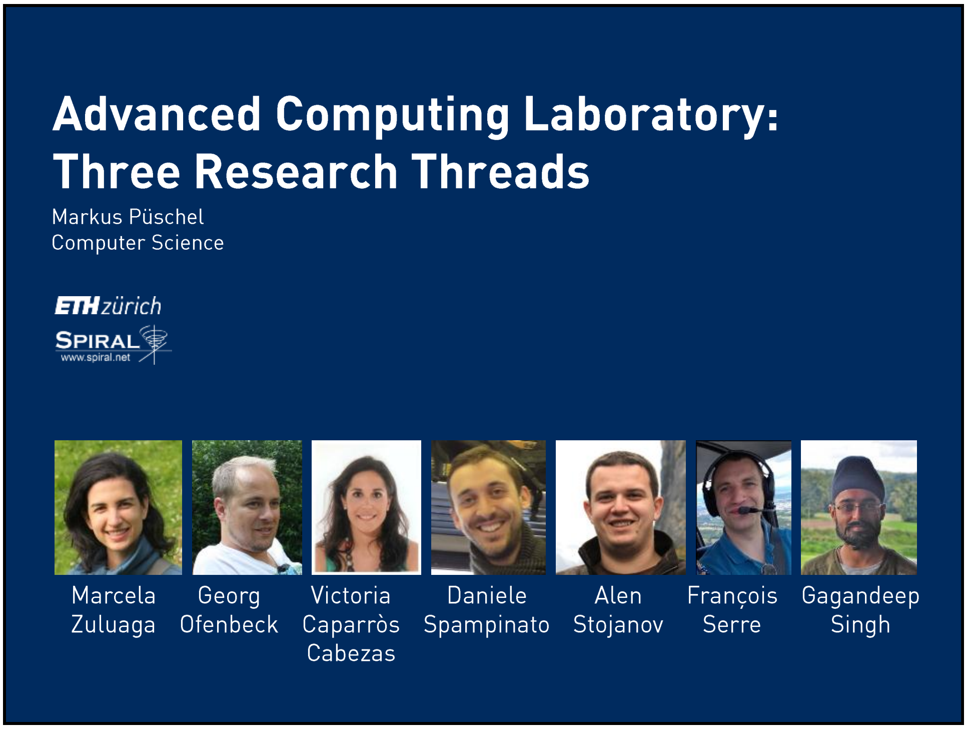
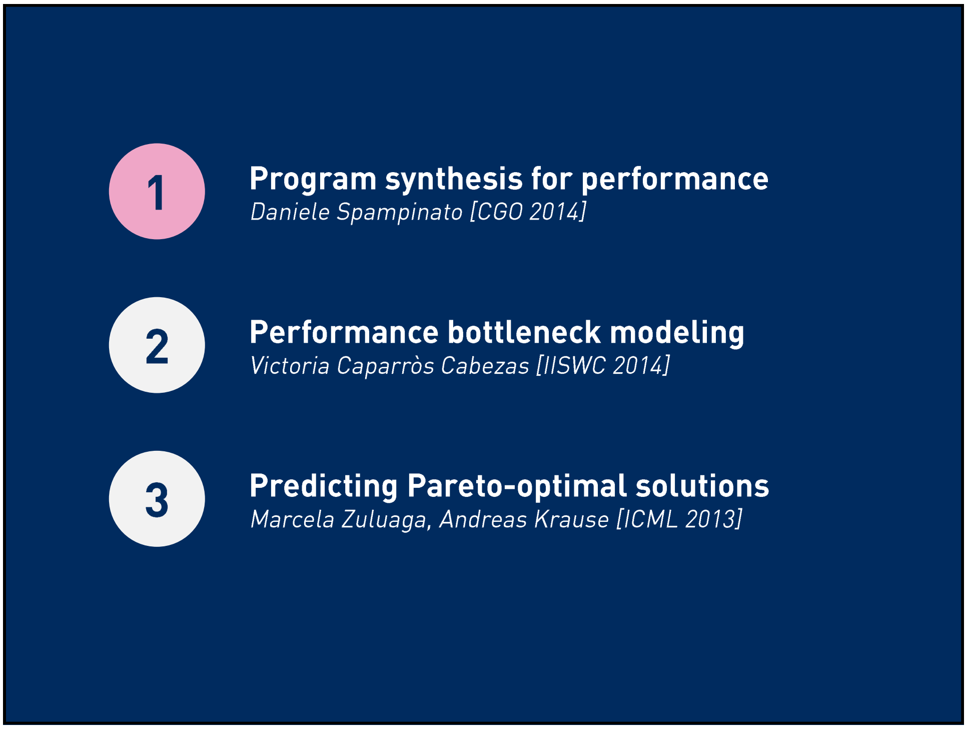


Advanced Computing Laboratory: Three Research Threads

Markus Püschel
Computer Science

ETH zürich
SPIRAL
www.spiral.net

Marcela Zuluaga | Georg Ofenbeck | Victoria Caparròs Cabezas | Daniele Spampinato | Alen Stojanov | François Serre | Gagandeep Singh



1. **Program synthesis for performance**
   Daniele Spampinato [CGO 2014]

2. **Performance bottleneck modeling**
   Victoria Caparròs Cabezas [IISWC 2014]

3. **Predicting Pareto-optimal solutions**
   Marcela Zuluaga, Andreas Krause [ICML 2013]

Vision:
Program Synthesis For Performance

Generate highest performance code for mathematical computations directly from a mathematical description

Approach

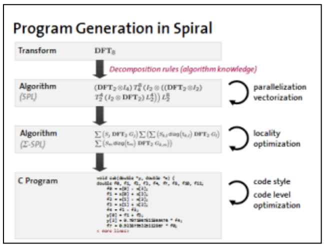
Mathematical DSLs

Rewriting systems for difficult optimizations

Compiler

Learning and search for fine-tuning

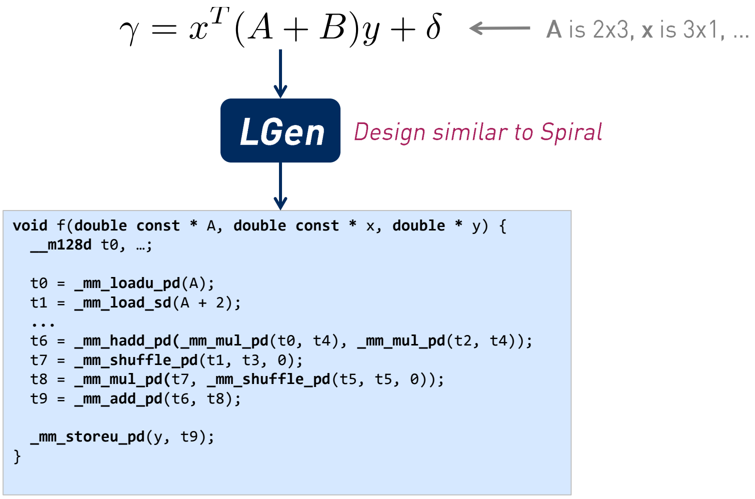
Use advanced software platforms
for the development of generators



Program Generation in Spiral
Transform
$\mathbf{DFT}_8$
Decomposition rules (algorithm knowledge)
Algorithm
(SPL)
$(\mathbf{DFT}_2 \otimes I_4)\, T^8_4\, (I_2 \otimes ((\mathbf{DFT}_2 \otimes I_2)\, T^4_2\, (I_2 \otimes \mathbf{DFT}_2)\, L^4_2))\, L^8_2$
parallelization
vectorization
Algorithm
(Σ-SPL)
locality
optimization
C Program
< more lines>
code style
code level
optimization

Example: Linear transforms
www.spiral.net

LGen: Generator for Linear Algebra

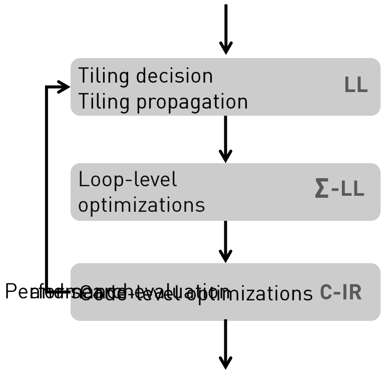


```
void f(double const * A, double const * x, double * y) {
  __m128d t0, …;

  t0 = _mm_loadu_pd(A);
  t1 = _mm_load_sd(A + 2);
  ...
  t6 = _mm_hadd_pd(_mm_mul_pd(t0, t4), _mm_mul_pd(t2, t4));
  t7 = _mm_shuffle_pd(t1, t3, 0);
  t8 = _mm_mul_pd(t7, _mm_shuffle_pd(t5, t5, 0));
  t9 = _mm_add_pd(t6, t8);

  _mm_storeu_pd(y, t9);
}
```

Architecture of LGen



Basic linear algebra computation (BLAC)

Tiling decision
Tiling propagation **LL**

Loop-level optimizations **Σ-LL**

Code-level optimizations **C-IR**

Performance evaluation

Optimized C function

$$y = Ax$$

$$[y = Ax]_{2,1}$$

$$y = \sum_{i,j} S_i(G_i \cdots)$$

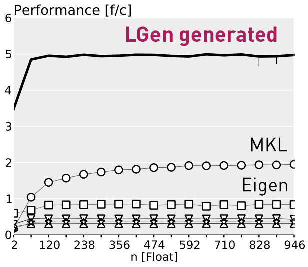
```
...
Mov (mmMulPs A[0,0], x[0,0]), t[0,0]
...
```

```
for(int i = … ) {
  …
  t = _mm_mul_ps(a, x);
  …
}
```

Example Benchmarks

Intel Xeon Westmere

$$C = \alpha AB + \beta C$$

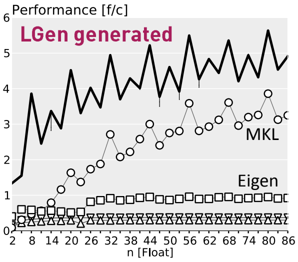


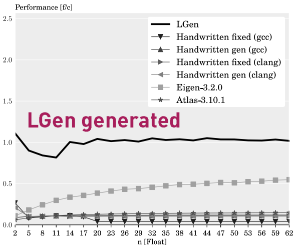
$$A \in \mathbb{R}^{n\times 4}$$

$$B \in \mathbb{R}^{4\times 4}$$

Intel Xeon Westmere **ARM Cortex-A8**

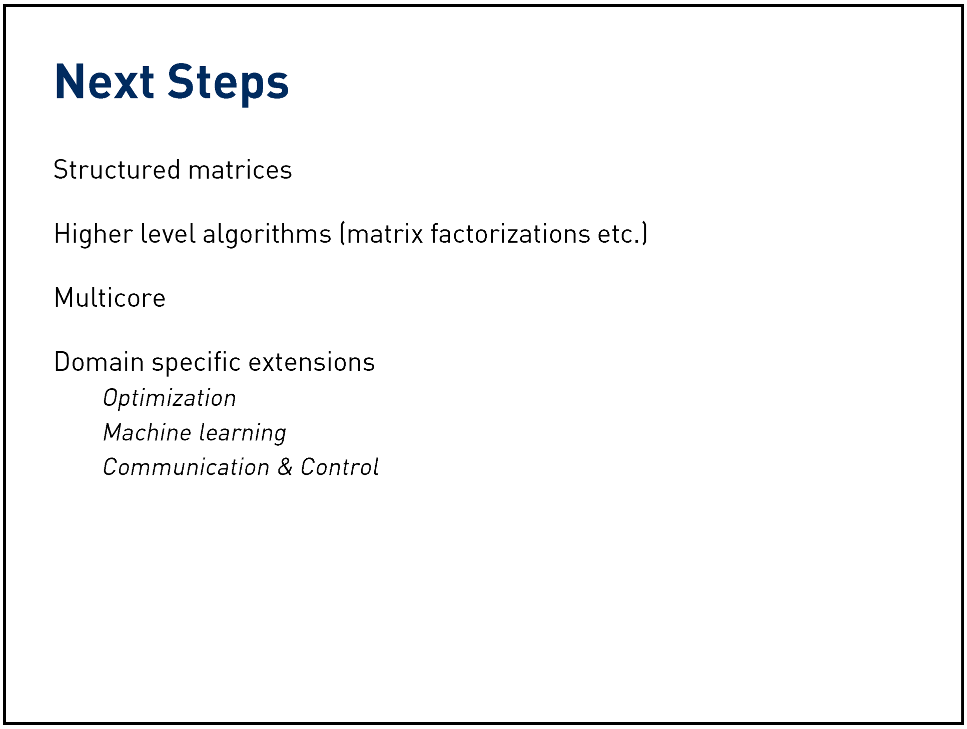
$$C = \alpha(A_0 + A_1)^T B + \beta C$$





$$A_0 \in \mathbb{R}^{4\times n}$$

$$B \in \mathbb{R}^{4\times n}$$



Next Steps

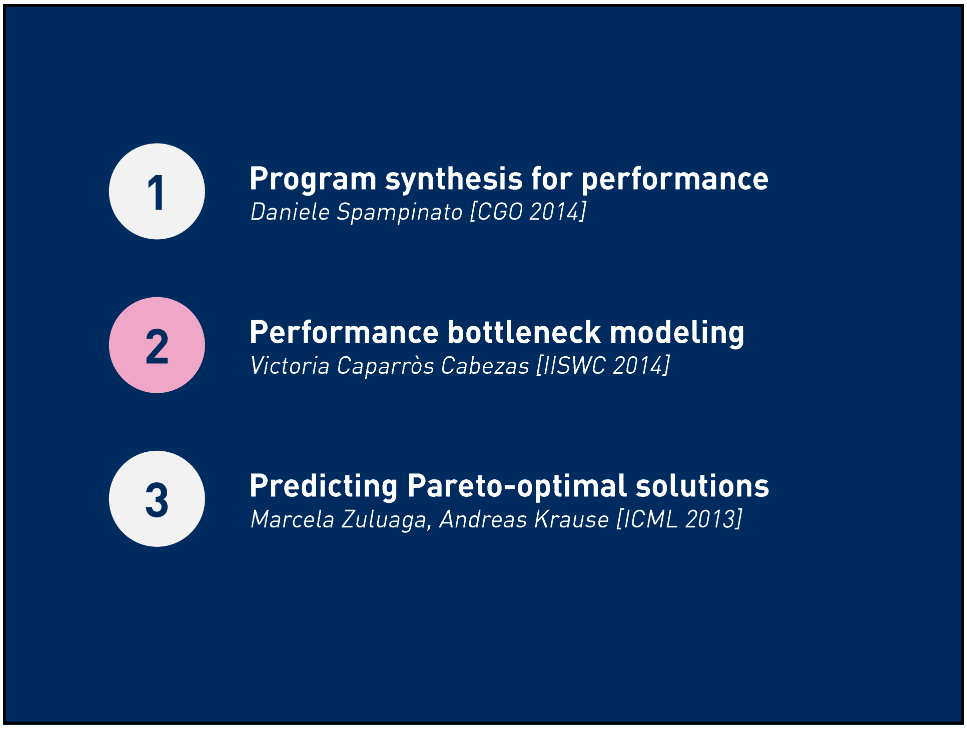
Structured matrices

Higher level algorithms (matrix factorizations etc.)

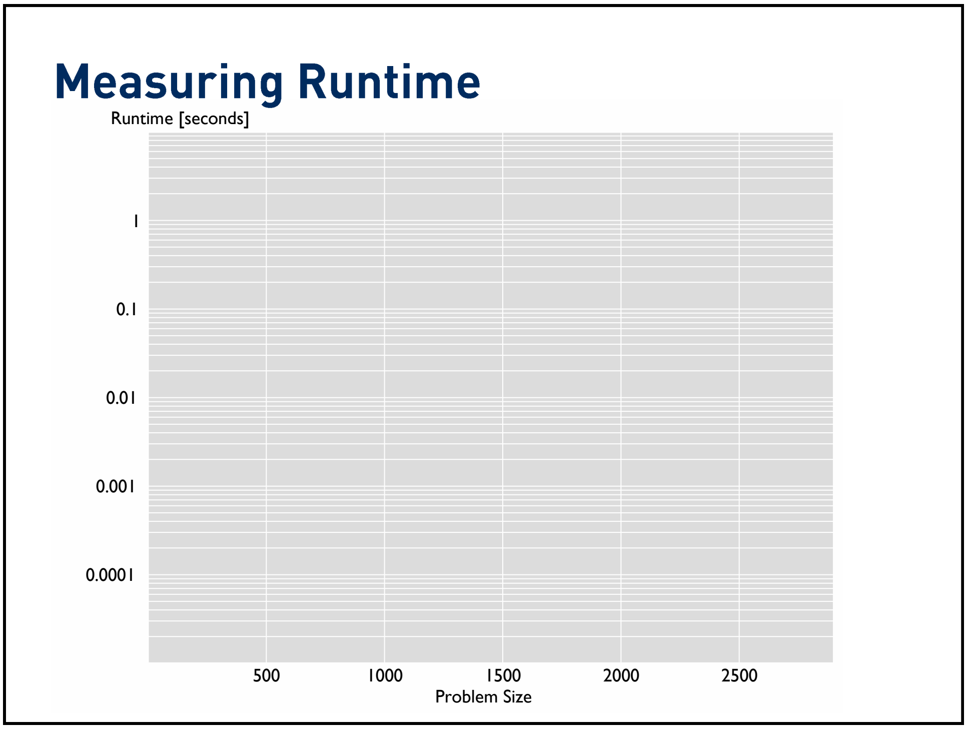
Multicore

Domain specific extensions

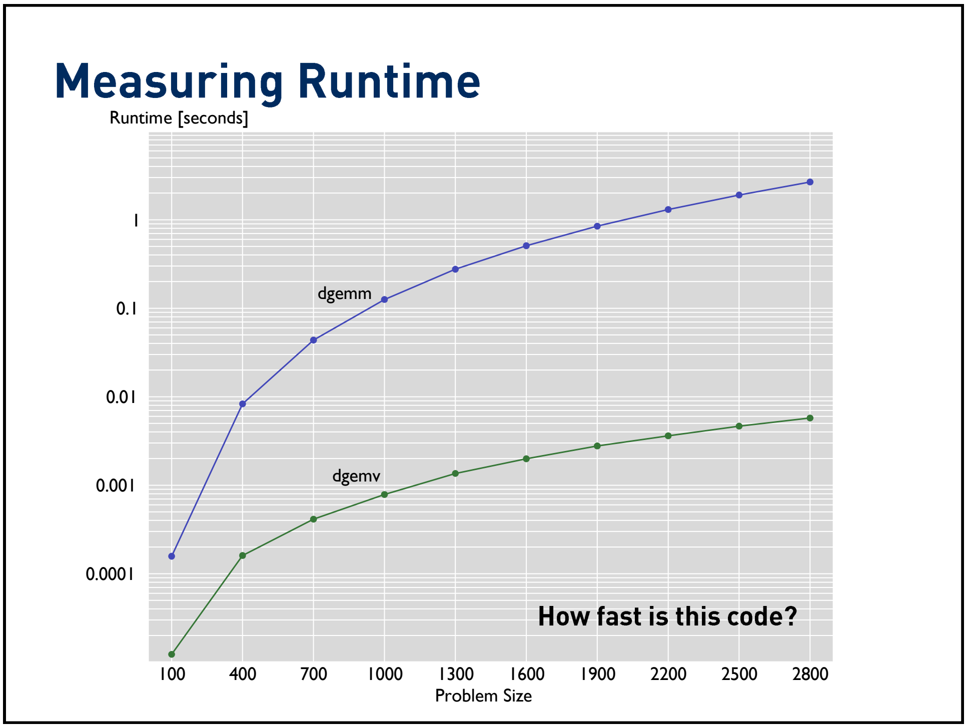
- *Optimization*
- *Machine learning*
- *Communication & Control*



1. **Program synthesis for performance**
   Daniele Spampinato [CGO 2014]
2. **Performance bottleneck modeling**
   Victoria Caparròs Cabezas [IISWC 2014]
3. **Predicting Pareto-optimal solutions**
   Marcela Zuluaga, Andreas Krause [ICML 2013]

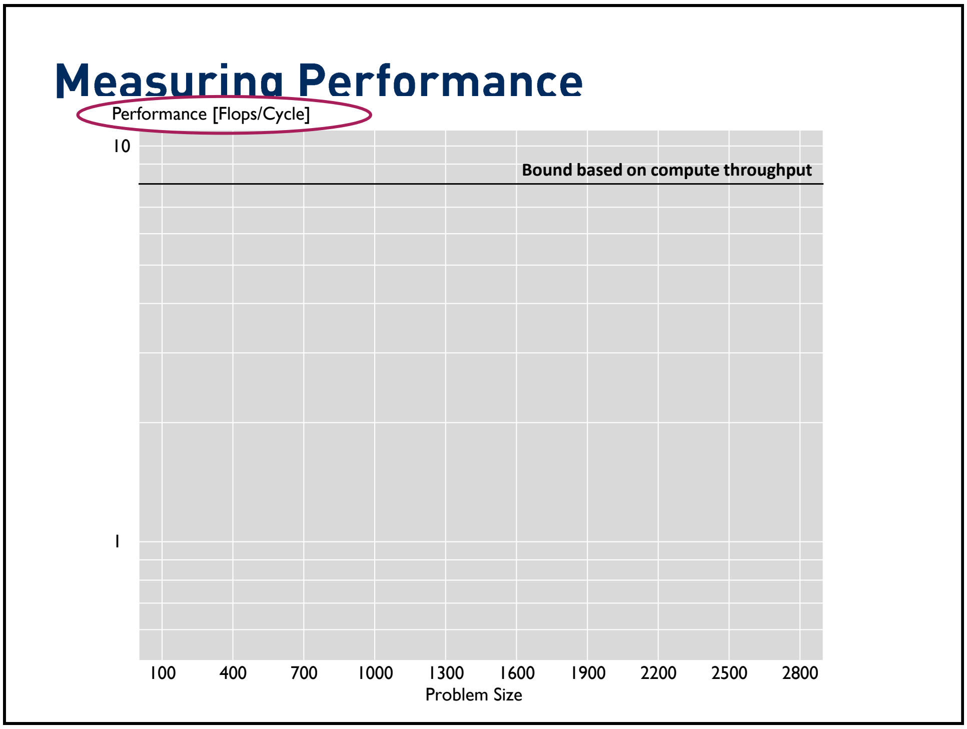


Measuring Runtime

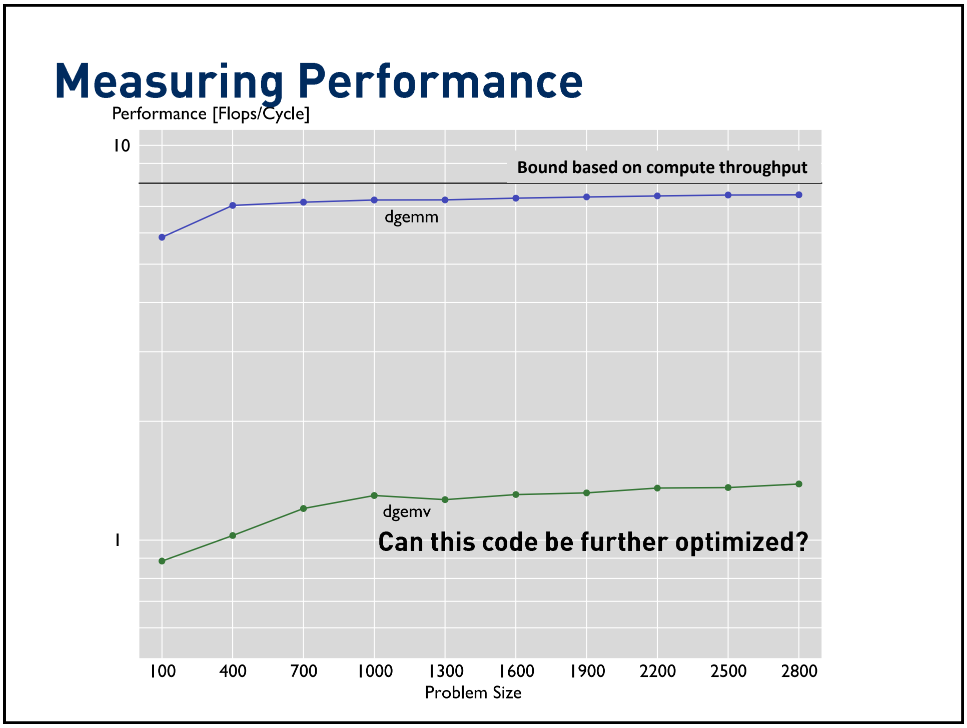


Measuring Runtime

How fast is this code?

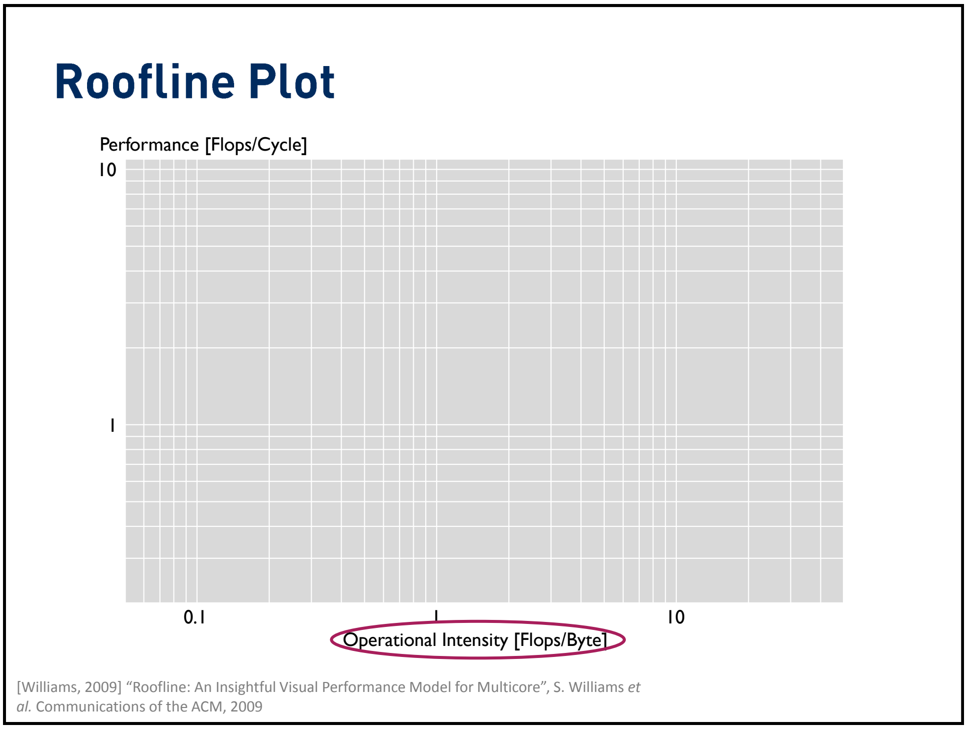


Measuring Performance



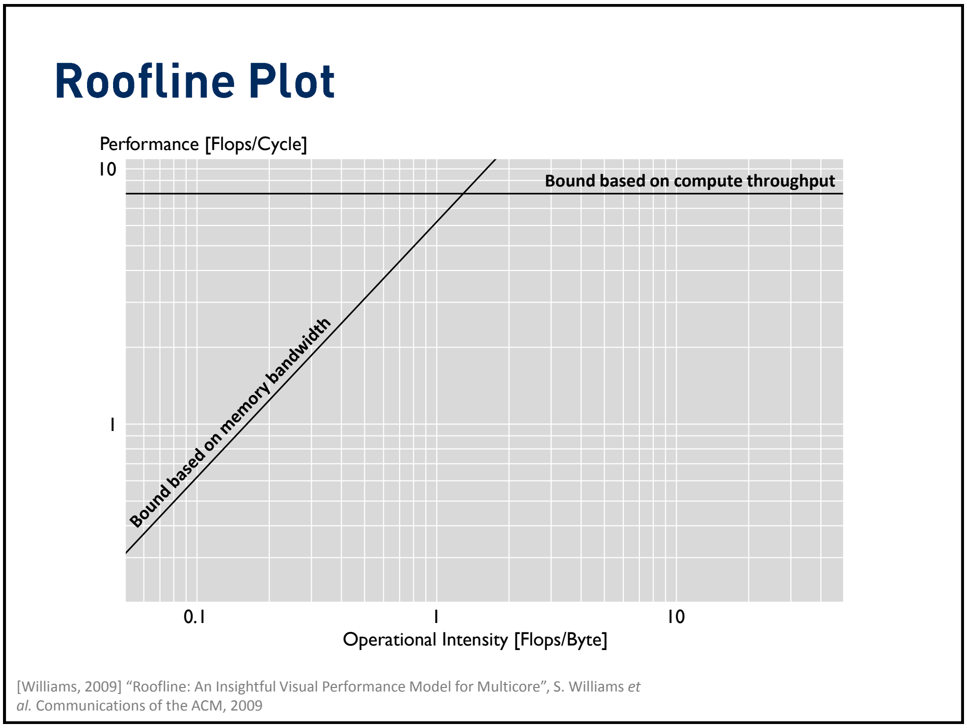
Measuring Performance

Can this code be further optimized?



Roofline Plot

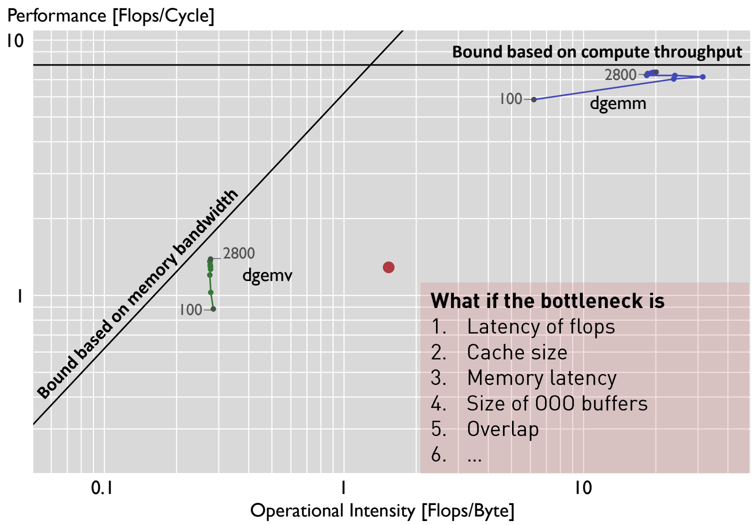
[Williams, 2009] "Roofline: An Insightful Visual Performance Model for Multicore", S. Williams *et al.* Communications of the ACM, 2009



Roofline Plot

[Williams, 2009] "Roofline: An Insightful Visual Performance Model for Multicore", S. Williams *et al.* Communications of the ACM, 2009

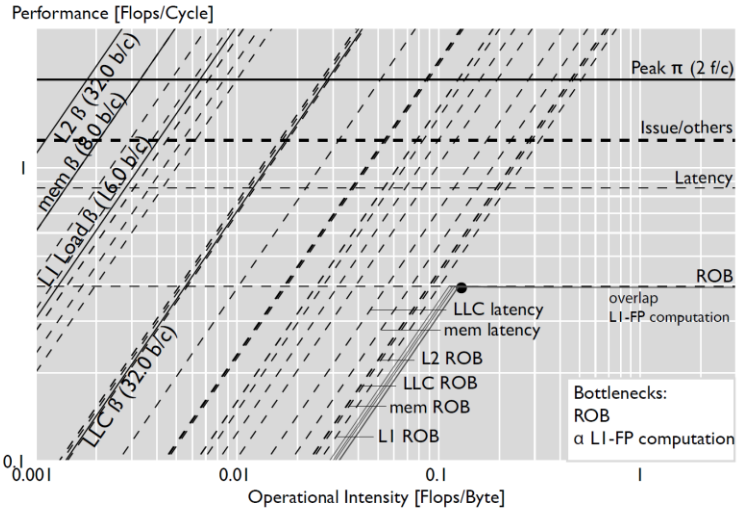
Roofline Plot



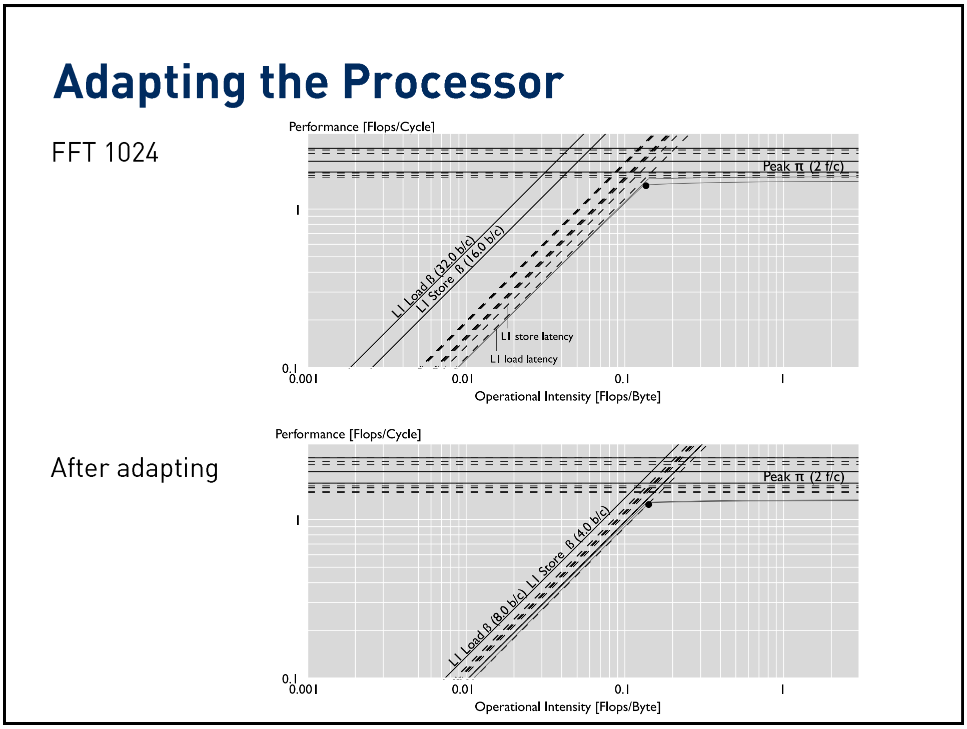
What if the bottleneck is

1. Latency of flops
2. Cache size
3. Memory latency
4. Size of OOO buffers
5. Overlap
6. …

Bottleneck Modelling



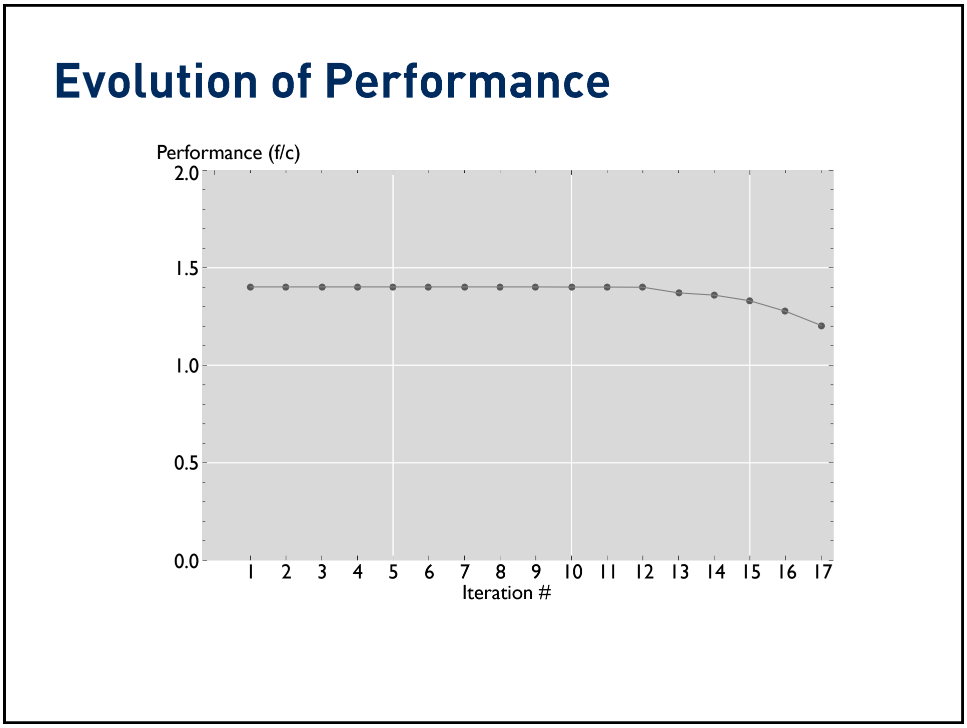
Example FFT 2^{20}



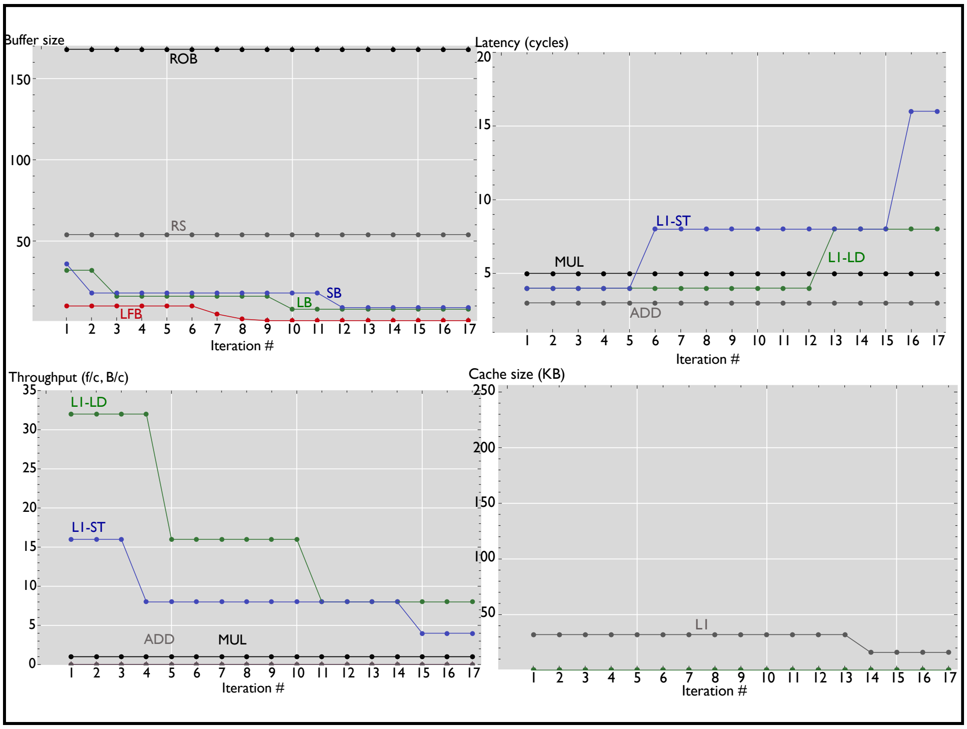
Adapting the Processor

FFT 1024

After adapting



Evolution of Performance



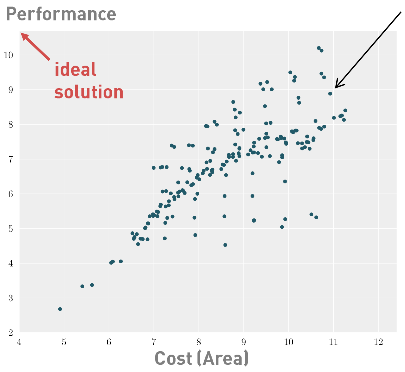
1 **Program synthesis for performance**
Daniele Spampinato [CGO 2014]

2 **Performance bottleneck modeling**
Victoria Caparròs Cabezas [IISWC 2014]

3 **Predicting Pareto-optimal solutions**
Marcela Zuluaga, Andreas Krause [ICML 2013]

Multi-Objective Optimization

Case Study: Different Hardware Implementations of a 256-input Sorter

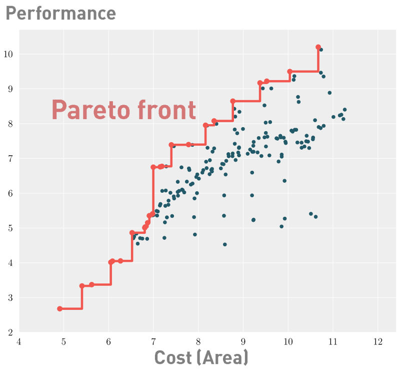


each determined by a set of 4 parameters

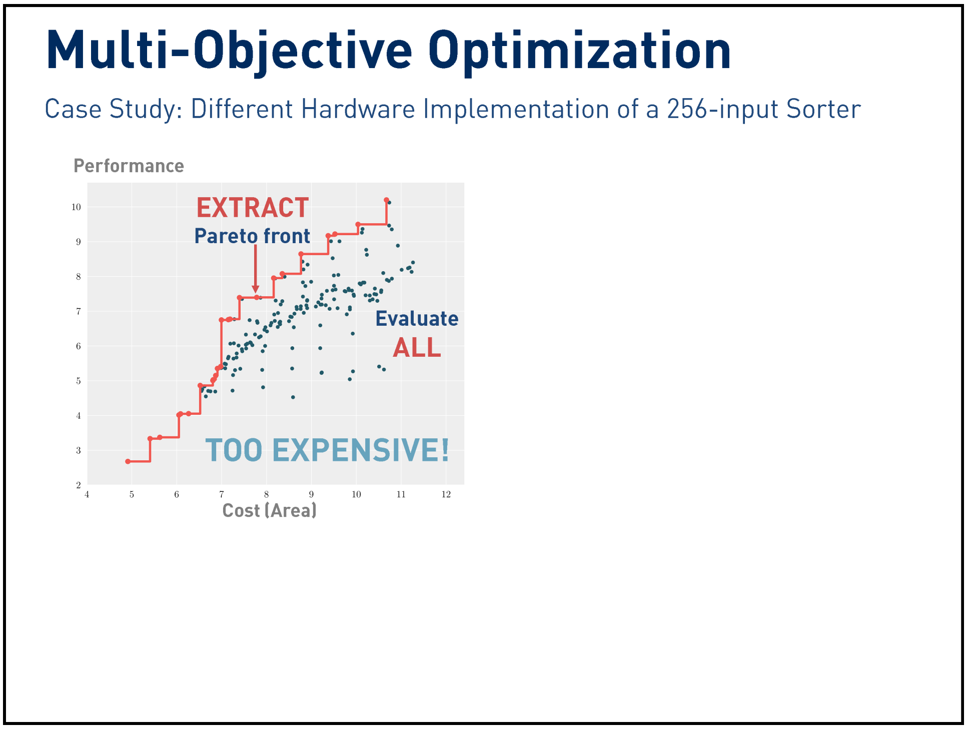
Which ones are relevant?

Multi-Objective Optimization

Case Study: Different Hardware Implementation of a 256-input Sorter



How to get it?



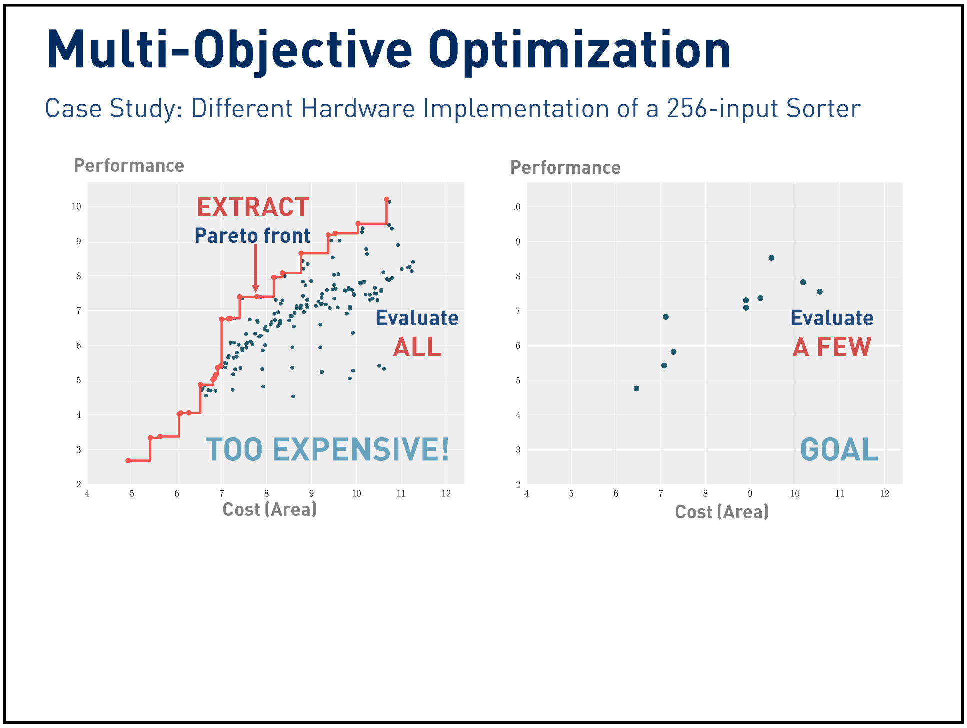
Multi-Objective Optimization

Case Study: Different Hardware Implementation of a 256-input Sorter

EXTRACT
Pareto front

Evaluate
ALL

TOO EXPENSIVE!



Multi-Objective Optimization

Case Study: Different Hardware Implementation of a 256-input Sorter

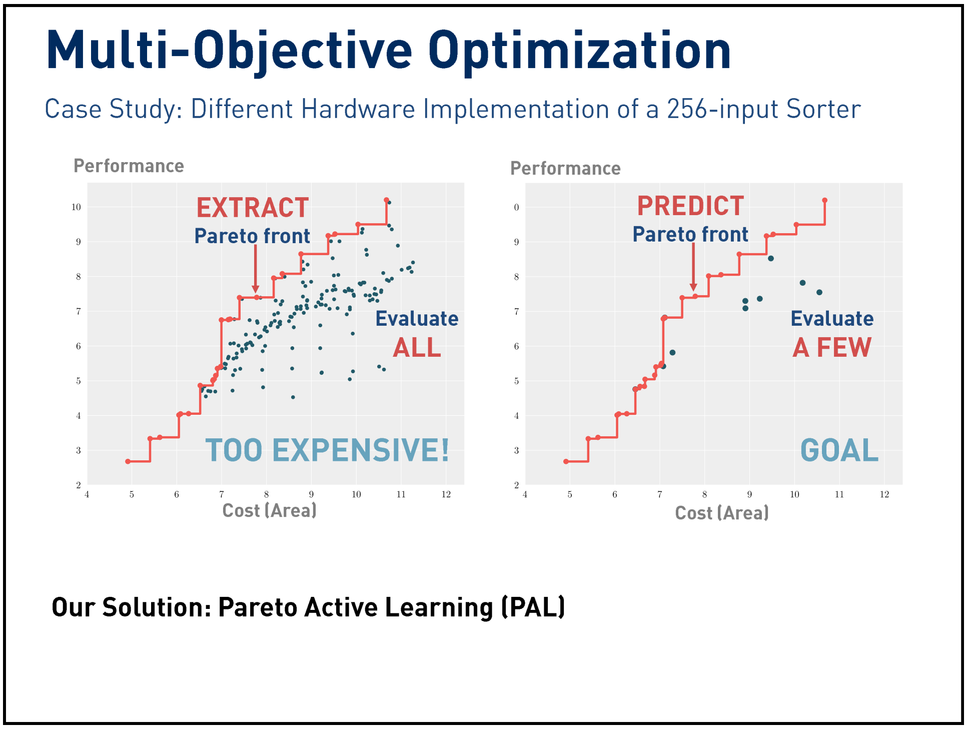
EXTRACT
Pareto front

Evaluate
ALL

TOO EXPENSIVE!

Evaluate
A FEW

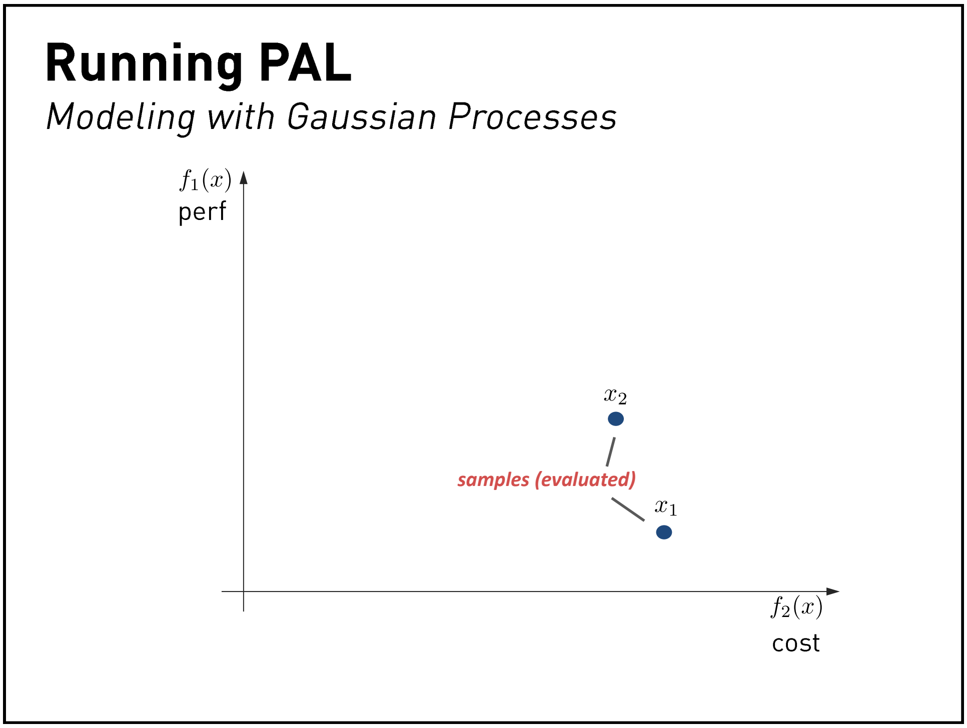
GOAL



Multi-Objective Optimization

Case Study: Different Hardware Implementation of a 256-input Sorter

Our Solution: Pareto Active Learning (PAL)

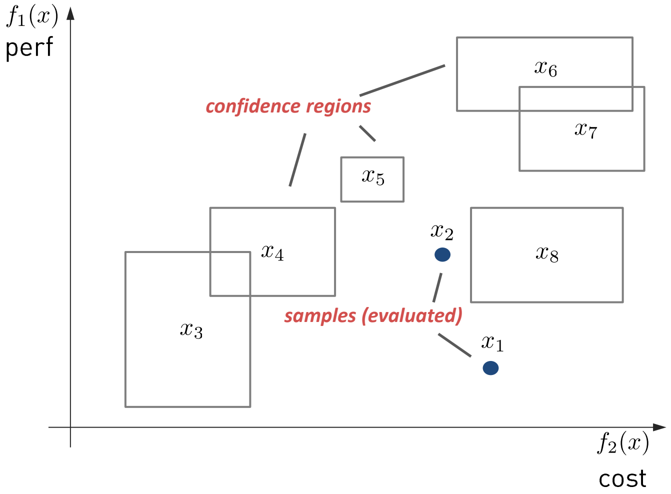


Running PAL

Modeling with Gaussian Processes

Running PAL

Modeling with Gaussian Processes

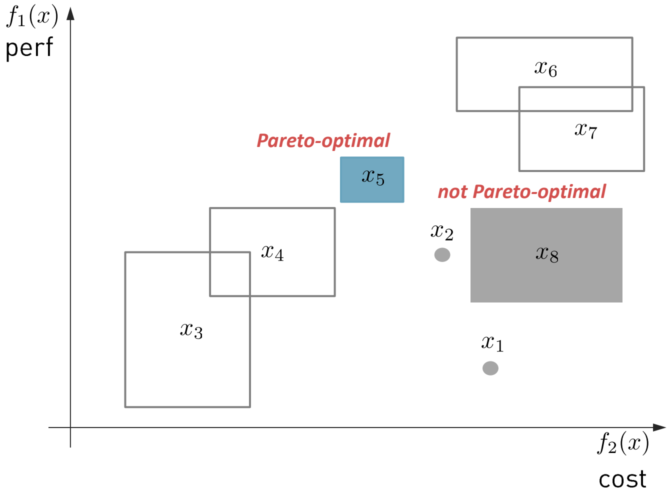


confidence regions

samples (evaluated)

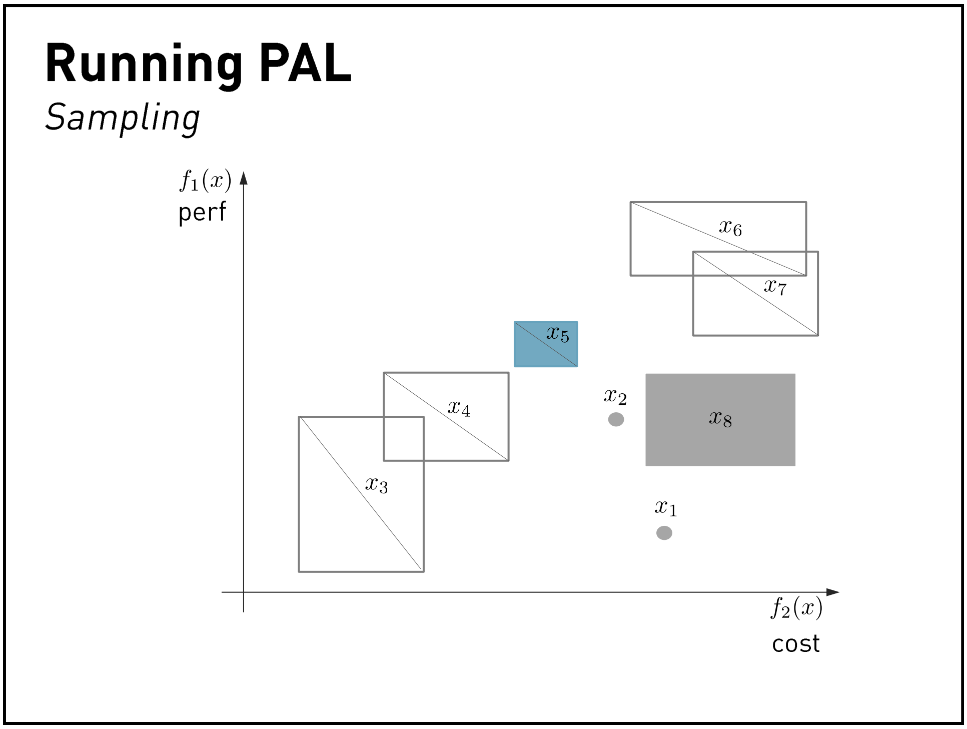
Running PAL

Classification



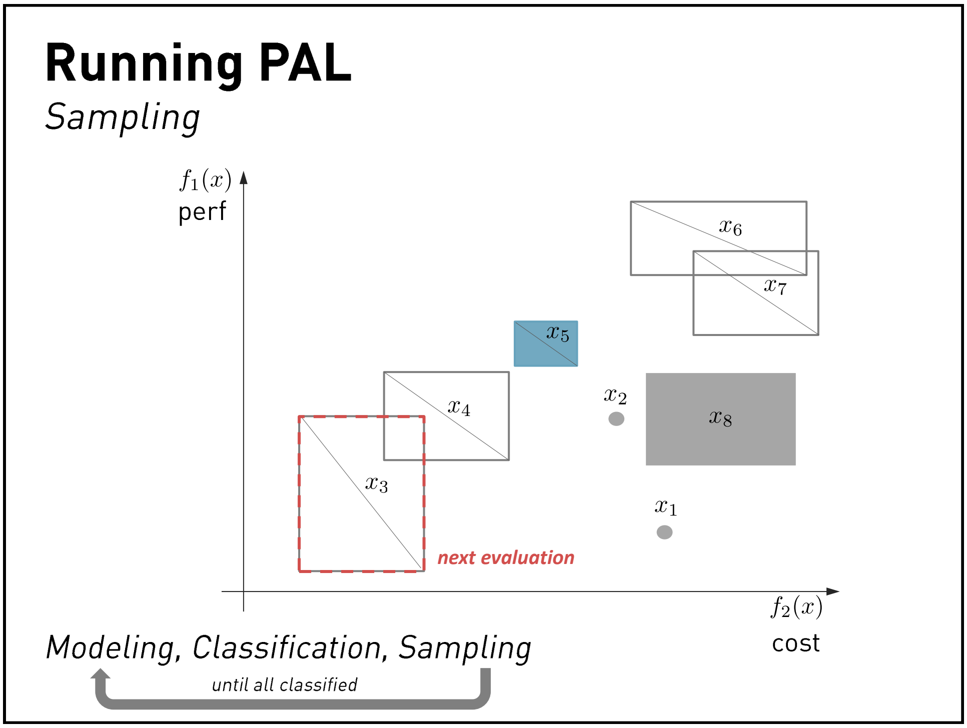
Pareto-optimal

not Pareto-optimal



Running PAL

Sampling



Running PAL

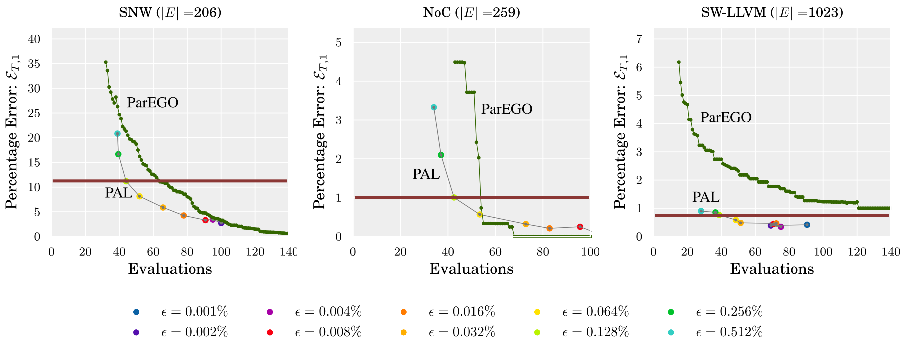
Sampling

Modeling, Classification, Sampling

until all classified

Convergence Behaviour

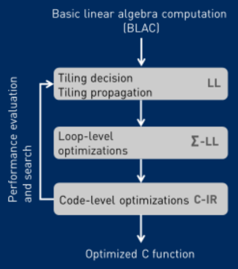
ParEGO: State-of-the-art evolutionary algorithm



Program synthesis for performance

Bottleneck modelling

Predicting Pareto fronts


Basic linear algebra computation (BLAC)

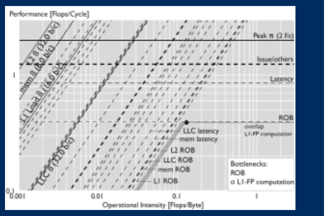
Tiling decision
Tiling propagation LL

Loop-level optimizations Σ-LL

Code-level optimizations C-IR

Optimized C function

Performance evaluation and search

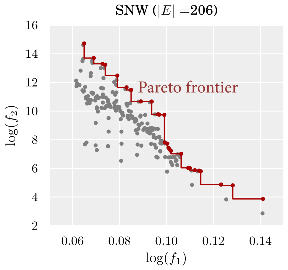


PREDICT
Pareto front
Evaluate
A FEW
GOAL

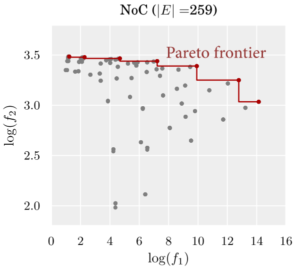
I am happy to discuss more
www.acl.inf.ethz.ch

Experiments: Data sets

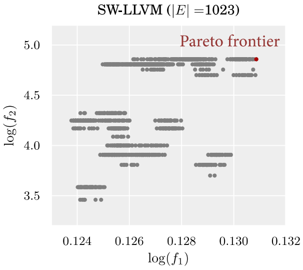
4 features



4 features



11 features



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*