配置优化

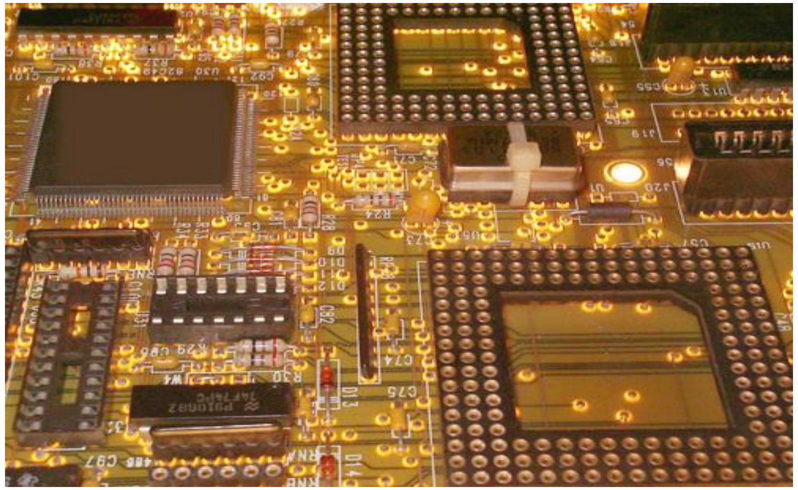
郭健美

2023年秋

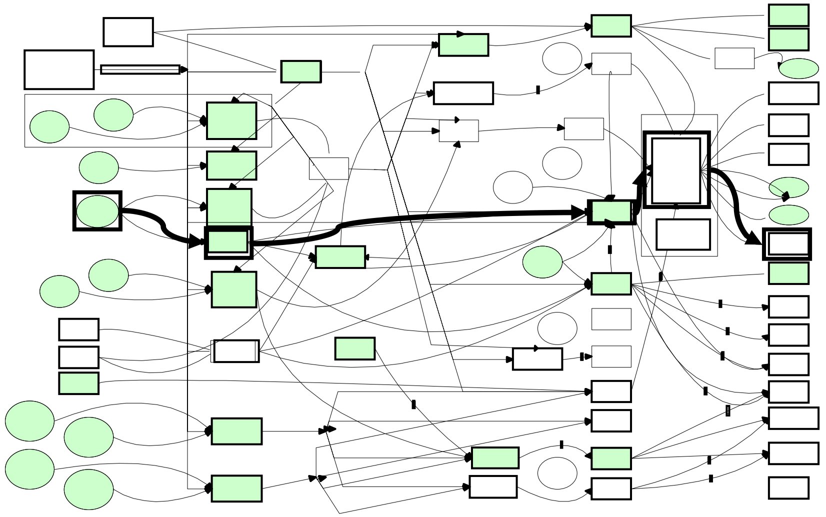
SOLE 系统优化实验室
华东师范大学

HW-SW Codesign is Ubiquitous






Automotive Electrical System Design

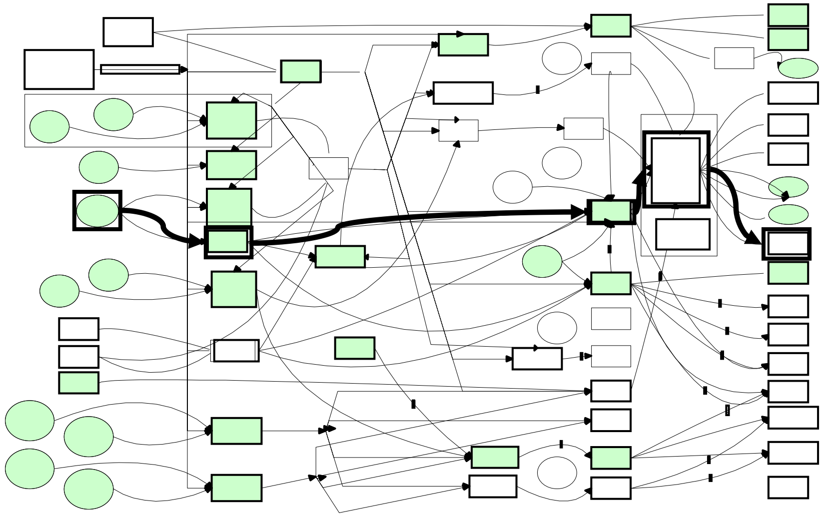


Functionality
- 1000’s Features
- 1000’s Messages

Architecture
- 60+ ECUs and
- 10+ Communication Busses

Design Objectives (Quality Req.)
- mass
- cost
- fuel consumption
- performance
- bus bandwidth
- safety
- security
- etc.

Automotive Electrical System Design



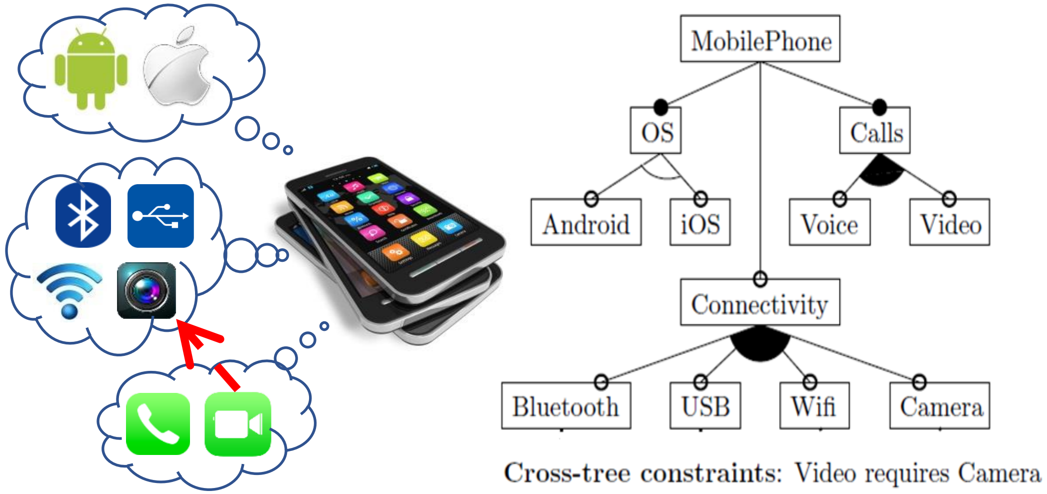
Functionality
- 1000’s Features
- 1000’s Messages

Architecture
- 60+ ECUs and
- 10+ Communication Busses

Design Objectives (Quality Req.)
- mass
- cost
- fuel consumption
- performance
- bus bandwidth
- safety
- security
- etc.

Huge Design Space!

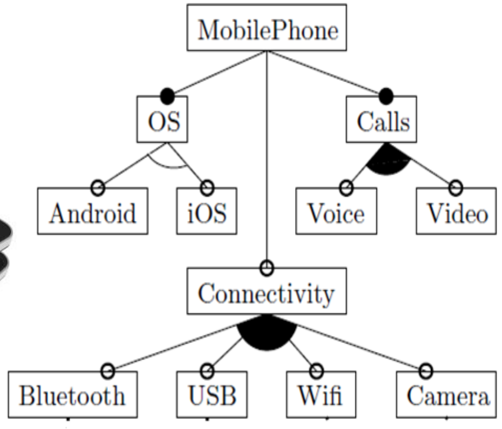
A Model-Driven Methodology



Feature modeling

- Enables formal representation
- Supports automated analysis
using off-the-shelf constraint solvers

A Model-Driven Methodology



Cross-tree constraints: Video requires Camera

Feature modeling

- Enables formal representation
- Supports automated analysis

using off-the-shelf constraint solvers

Program synthesis!

Translation rules → Propositional formulas → Constraint solvers → Feasible designs

Type	Propositional Formulas
Mandatory	$C_1 \leftrightarrow P$
Optional	$C_1 \rightarrow P$
And-group	$(P \rightarrow \bigwedge_{i \in M} C_i) \wedge (\bigvee_{1 \le i \le n} C_i \rightarrow P)$
Or-group	$P \leftrightarrow \bigvee_{1 \le i \le n} C_i$
Alternative-group	$(P \leftrightarrow \bigvee_{1 \le i \le n} C_i) \wedge \bigwedge_{i<j} (\neg C_i \vee \neg C_j)$
Requires	$F_1 \rightarrow F_2$
Excludes	$\neg(F_1 \wedge F_2)$

A constraint satisfaction problem

[Kang et al., **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. CMU SEI, SEI-90-TR-21, 1990]
[Czarnecki and Eisenecker, **Generative Programming: Methods, Tools, and Applications**. Addison-Wesley, 2000]
[Batory, **Feature Models, Grammars, and Propositional Formulas**. In *Proc. SPLC*, 2005]

Configurable software and variability are ubiquitous

Configure software to tailor functional behavior

a command-line tool to encode a video stream

```
x264 --quiet
     --no-progress
     --no-asm
     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

feature1
feature2
feature3
feature4
feature5

configuration1 (variant1)

output stream
input stream

[Guo et al., **Variability-Aware Performance Prediction: A Statistical Learning Approach**. In *Proc. ASE*, 2013]

Configure software to meet a certain performance goal

configuration1

```
x264 --quiet
     --no-progress
     --no-asm
     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

424 seconds

configuration2

```
x264
     --no-progress
     --no-asm
     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

651 seconds

A non-linear regression problem

Predictors

Response

feature1: “quiet”

Boolean variables

Numeric variable

A small random sample

Conf.	Features																Perf. (s)
c_i	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	p_i
c_1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	1	651
c_2	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	0	536
c_3	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	1	581
c_4	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	381
c_5	1	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	424
c_6	1	1	0	0	1	0	1	1	1	1	0	0	1	0	0	1	615
c_7	1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	477
c_8	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0	0	263
c_9	1	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	272
c_{10}	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	247
c_{11}	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	612
c_{12}	1	0	1	1	1	0	0	0	1	0	0	1	1	0	1	0	510
c_{13}	1	1	1	1	0	1	1	0	1	0	1	0	1	0	0	1	555
c_{14}	1	1	0	0	1	0	1	1	1	0	0	1	1	1	0	0	264
c_{15}	1	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	576
c_{16}	1	0	1	0	1	0	1	1	1	0	1	0	1	1	0	0	268
c	1	1	0	1	1	0	0	0	0	1	0	0	1	0	1	0	?

Challenges?

Combinatorial explosion!

Table 2 Overview of the subject systems; LOC: lines of code; N: number of all features; $|W|$: size of the whole population of all configurations

System	Domain	Language	LOC	N	$\|W\|$
AJSTATS	Code analyzer	C	14 782	19	30 256
APACHE	Web server	C	230 277	9	192
BDB-C	Database system	C	219 811	18	2 560
BDB-J	Database system	Java	42 596	26	180
CLASP	Answer set solver	C++	30 871	19	700
HIPA[cc]	Video processing library	C++	25 605	52	13 485
LLVM	Compiler infrastructure	C++	47 549	11	1 024
LRZIP	Compression library	C++	9 132	19	432
SQLITE	Database system	C	312 625	39	4 653
X264	Video encoder	C	45 743	16	1 152

Table 3 Overview of subject SPLs

System	Version	#Features	#Constraints
ECOS	3.0	1244	3146
FREEBSD	8.0.0	1396	62,183
FIASCO	2011081207	1638	5228
UCLINUX	20100825	1850	2468
LINUX	2.6.28.6	6888	343,944

[J. Guo, et al. Data-efficient performance learning for configurable systems. Empir. Softw. Eng. 23(3): 1826-1867, 2018.]
[J. Guo, et al. SMTIBEA: a hybrid multi-objective optimization algorithm for configuring large constrained software product lines. Softw. Syst. Model. 18(2): 1447-1466, 2019.]

Measurement effort!

Q11: How long does it take to run? Does CPU 2017 take longer than CPU 2006?

Run time depends on the system, suite, compiler, tuning, and how many copies or threads are chosen.
One example system is shown below; your times will differ.

Example run times - simple options chosen

Metric	Config Tested	Individual Benchmarks	Full Run (Reportable)
SPECrate 2017 Integer	1 copy	6 to 10 minutes	2.5 hours
SPECrate 2017 Floating Point	1 copy	5 to 36 minutes	4.8 hours
SPECspeed 2017 Integer	4 threads	6 to 15 minutes	3.1 hours
SPECspeed 2017 Floating Point	16 threads	6 to 75 minutes	4.7 hours

One arbitrary example using a year 2016 system. Your system will differ.
2 iterations chosen, base only, no peak. Does not include compile time.

https://www.spec.org/cpu2017/Docs/overview.html#Q11

Feature interactions!

configuration1

```
x264 --quiet
     --no-progress
     --no-asm
     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

324 seconds

configuration2

```
x264
     --no-progress
     --no-asm
     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

551 seconds

configuration3

```
x264 --quiet
     --no-progress

     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

487 seconds

configuration4

```
x264
     --no-progress

     --rc-lookahead 60
     --ref 9
     -o trailer_480p24.x264
     trailer_2k_480p24.y4m
```

661 seconds

P(**quiet**)
= 551 – 324
= **227** seconds

Feature interactions

P'(**quiet**)
= 661– 487
= **174** seconds

特征交互会带来严重的性能和质量问题!

ABS防抱死系统

传统液压制动系统

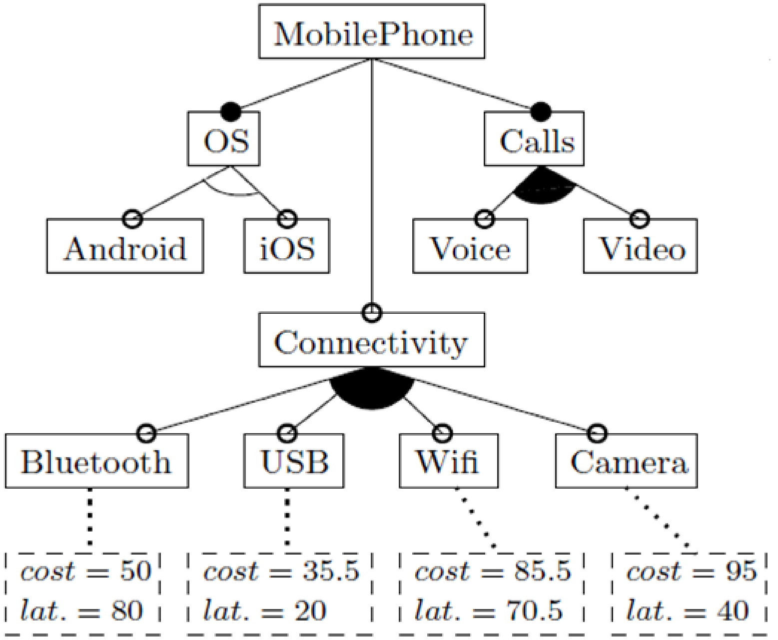
\+

新型再生制动系统

⇒

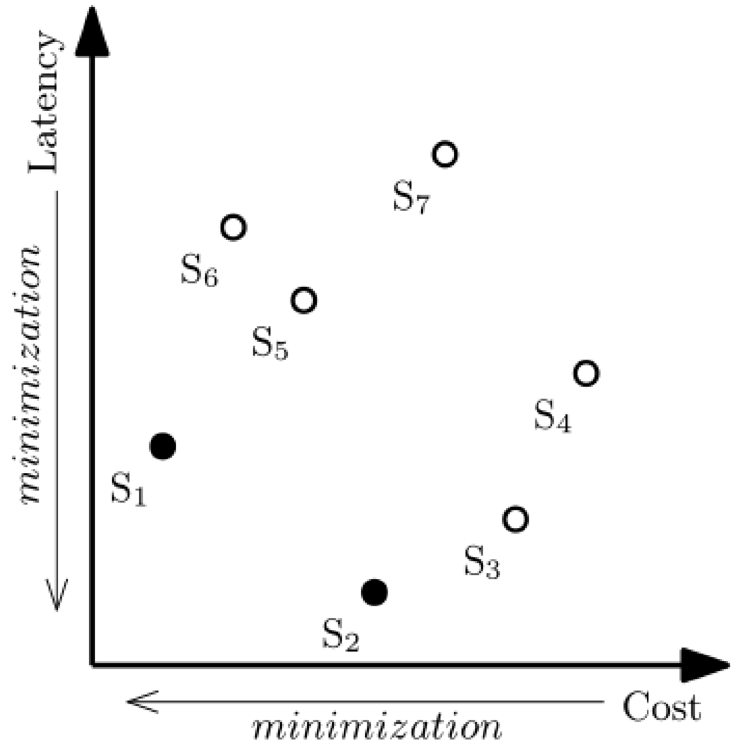
丰田普锐斯2010版混动车召回事件

A multi-objective combinatorial optimization problem



Cross-tree constraints: Video requires Camera

Objectives: minimizing cost, minimizing latency



[Guo et al., **Scaling Exact Multi-Objective Combinatorial Optimization by Parallelization**. In *Proc. ASE*, 2014]

[Guo et al., **To preserve or not to preserve invalid solutions in search-based software engineering: a case study in software product lines**. In *Proc. ICSE*, 2018]

如何搜索最优配置?

Design of experiments

- One-factor-at-a-time design
- Full factorial design
- Fractional factorial design
- …

Example

Personal workstation design

1. Processor: 68000, Z80, or 8086.
2. Memory size: 512K, 2M, or 8M bytes
3. Number of Disks: One, two, three, or four
4. Workload: Secretarial, managerial, or scientific.
5. User education: High school, college, or post-graduate level.

Five **Factors** at 3x3x4x3x3 **levels**

Types of Experimental Designs

- **Simple Designs**: Vary one factor at a time

$$\# \text{ of Experiments} = 1 + \sum_{i=1}^{k} (n_i - 1)$$

  - Not statistically efficient.
  - Wrong conclusions if the factors have interaction.
  - Not recommended.
- **Full Factorial Design**: All combinations.

$$\# \text{ of Experiments} = \prod_{i=1}^{k} n_i$$

  - Can find the effect of all factors.
  - Too much time and money.
  - May try 2^k design first.

Types of Experimental Designs (Cont)

- Fractional Factorial Designs: Less than Full Factorial
  - Save time and expense.
  - Less information.
  - May not get all interactions.
  - Not a problem if negligible interactions

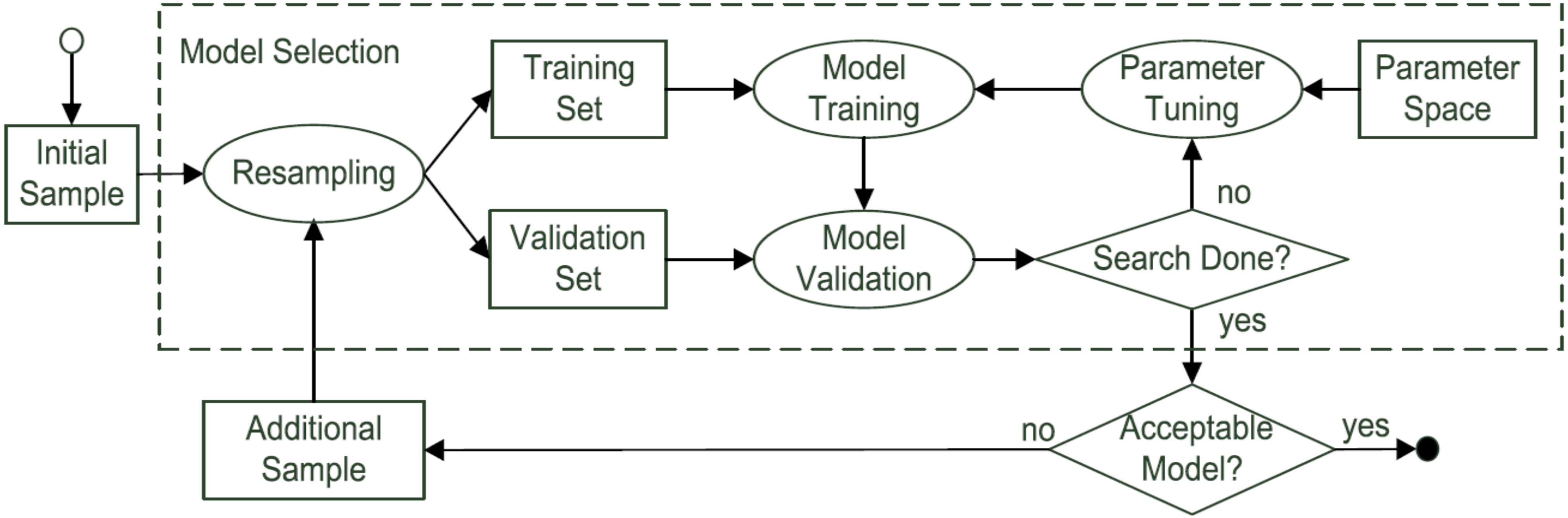
A Sample Fractional Factorial Design

- Workstation Design:
  (3 CPUs)(3 Memory levels)(3 workloads)(3 ed levels)
  = 81 experiments

Experiment Number	CPU	Memory Level	Workload Type	Educational Level
1	68000	512K	Managerial	High School
2	68000	2M	Scientific	Post-graduate
3	68000	8M	Secretarial	College
4	Z80	512K	Scientific	College
5	Z80	2M	Secretarial	High School
6	Z80	8M	Managerial	Post-graduate
7	8086	512K	Secretarial	Post-graduate
8	8086	2M	Managerial	College
9	8086	8M	Scientific	High School

Machine Learning

- CART
- Random forests
- Bayesian optimization
- …



[J. Guo, et al. Data-efficient performance learning for configurable systems. Empir. Softw. Eng. 23(3): 1826-1867, 2018.]

Data-Efficient Learning for Performance Modeling

- **Large-data problems**: object detection and recognition, machine translation, text-to-speech, recommender systems, and information retrieval (DATA IS CHEAP)
- **Small-data problems**: personalized healthcare, robot reinforcement learning, sentiment analysis, and community detection (DATA IS EXPENSIVE)

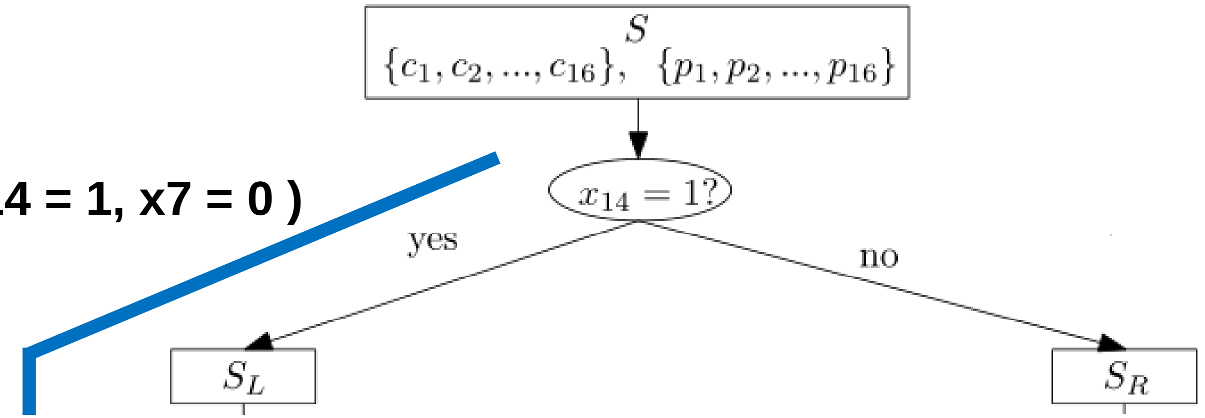
ICML’16 Workshop on

Data-Efficient Machine Learning

Classification And Regression Trees (CART)

[Breiman et al, 1984]

(x14 = 1, x7 = 0)



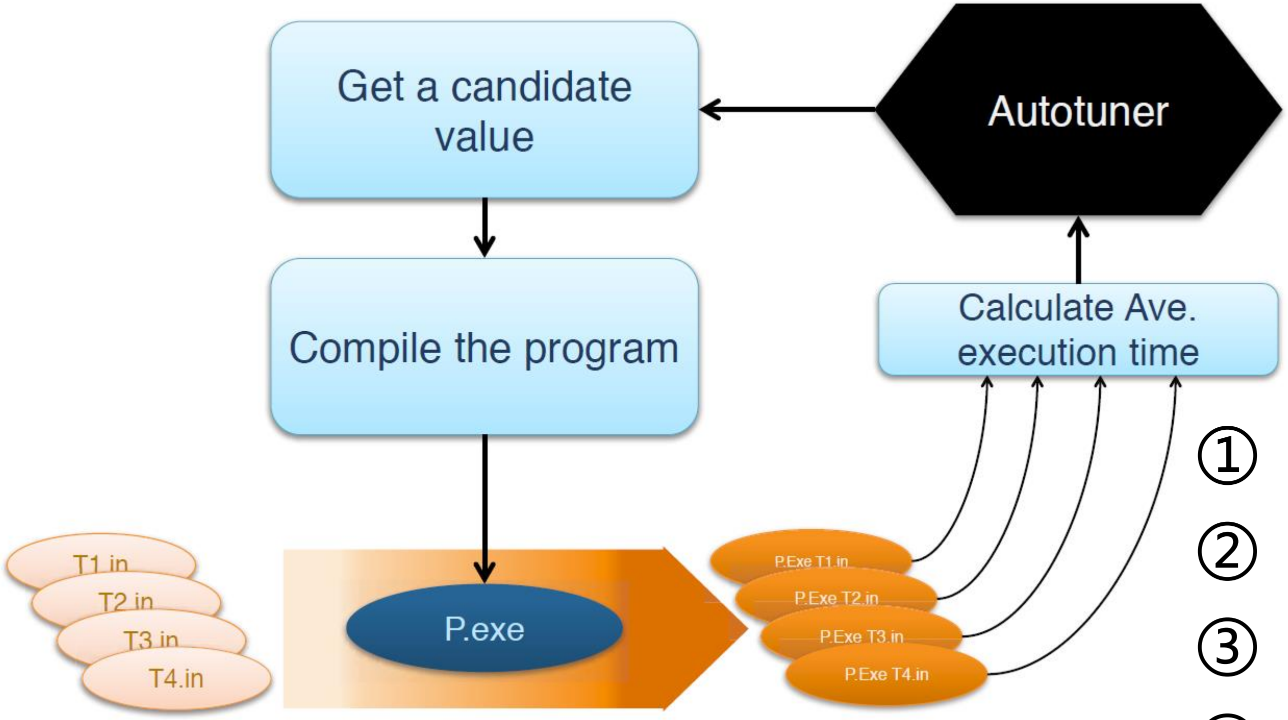
Predicted value
for any configuration
selecting feature_14 and
deselecting feature_7

OpenTuner

- Performance Engineering is all about finding the right:
  - block size in matrix multiply (voodoo parameters)
  - strategy in the dynamic memory allocation project
  - flags in calling GCC to optimize the program
  - schedule in Halide
  - schedule in GraphIt

https://opentuner.org/

Autotuning A Program



① Define a space of acceptable values

② Choose a value at random from that space

③ Evaluate the performance given that value

④ If satisfied with the performance or time limit exceeded,
then finish

⑤ Choose a new value from the feedback

⑥ Goto 3

Domain Knowledge

- New architectural features
- New instructions
- …

Photo from: https://www.hospitalityupgrade.com/techTalk/April-2016/A-Beacon-in-the-Dark/

- **Domain Knowledge**
  - New architectural features
  - New instructions
  - …

LSE (Large Systems Extensions) introduced in ARMv8.1

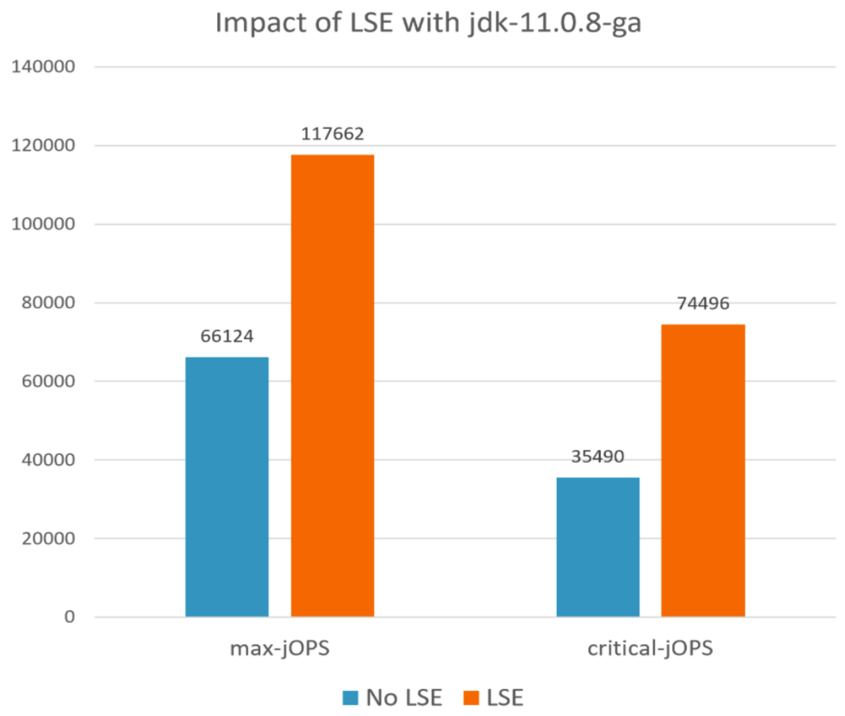
- New implementation of atomic operations (e.g., CAS)
- Benefit highly-concurrent workloads using heavy synchronizations
- Enable it in GCC flag `-march`

[S. Huang. Improving Java performance on OCI Ampere A1 compute instances. Arm Community, 2021.]
[Z. Wang, C. Wu. Alibaba Dragonwell Powers Java Applications in Alibaba Cloud (Arm ECS Instances). Alibaba Cloud Community 2021]

- **Domain Knowledge**
  - New architectural features
  - New instructions
  - …

LSE (Large Systems Extensions) introduced in ARMv8.1

- New implementation of atomic operations (e.g., CAS)
- Benefit highly-concurrent workloads using heavy synchronizations
- Enable it in GCC flag `-march`



OCI Ampere A1 instance

max-jOPS 1.8X
critical-jOPS 2.1X

[S. Huang. Improving Java performance on OCI Ampere A1 compute instances. Arm Community, 2021.]
[Z. Wang, C. Wu. Alibaba Dragonwell Powers Java Applications in Alibaba Cloud (Arm ECS Instances). Alibaba Cloud Community 2021]

Domain Knowledge

- New architectural features
- New instructions
- …

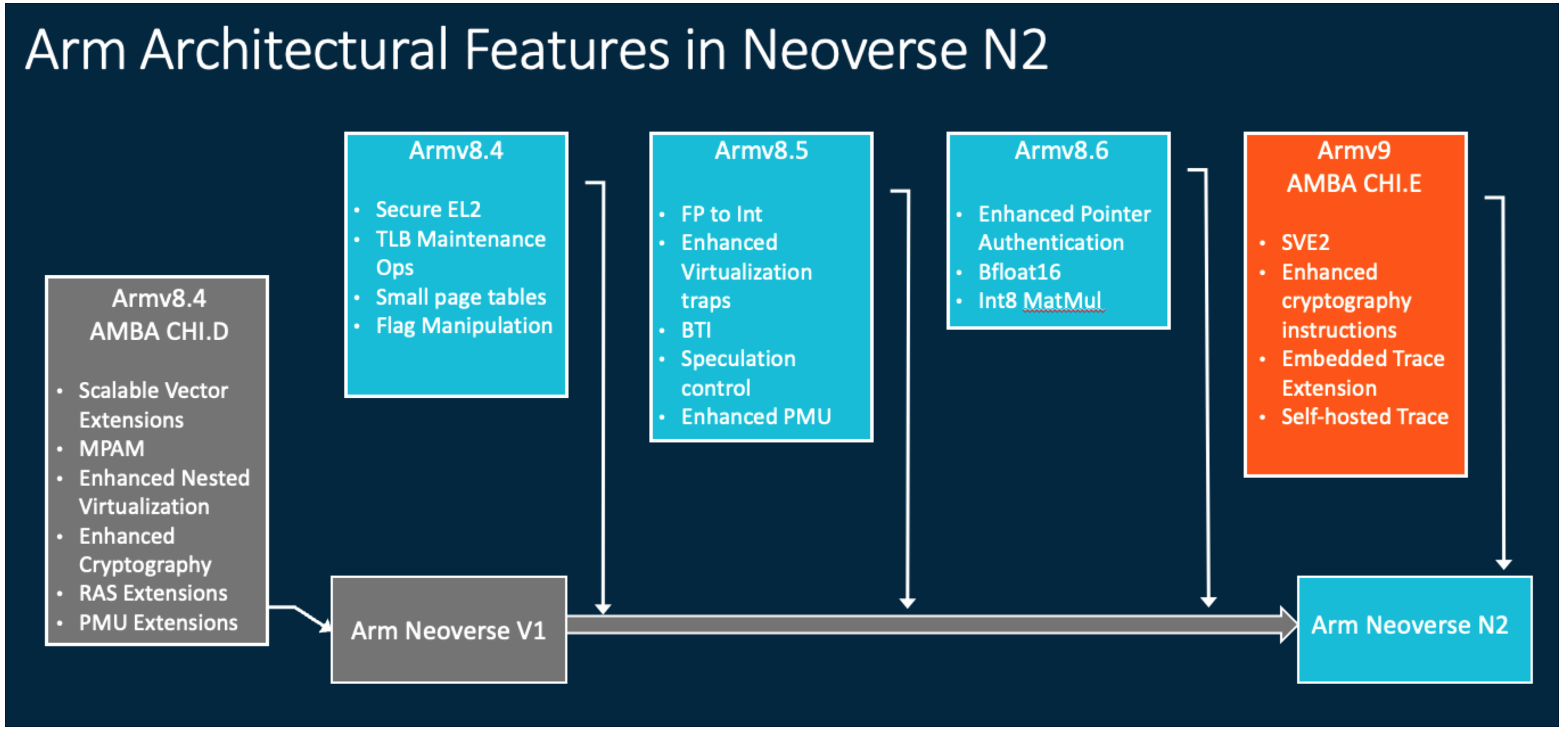


Photo from https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/arm-neoverse-n2-industry-leading-performance-efficiency

Exploration of Configuration Space

- **Design of experiments**
  - One-factor-at-a-time design
  - Full factorial design
  - Fractional factorial design
  - …
- **Machine learning**
  - CART
  - Random forests
  - Bayesian optimization
  - …
- **Domain Knowledge**
  - New architectural features
  - New instructions
  - …

Huge Configuration Space!