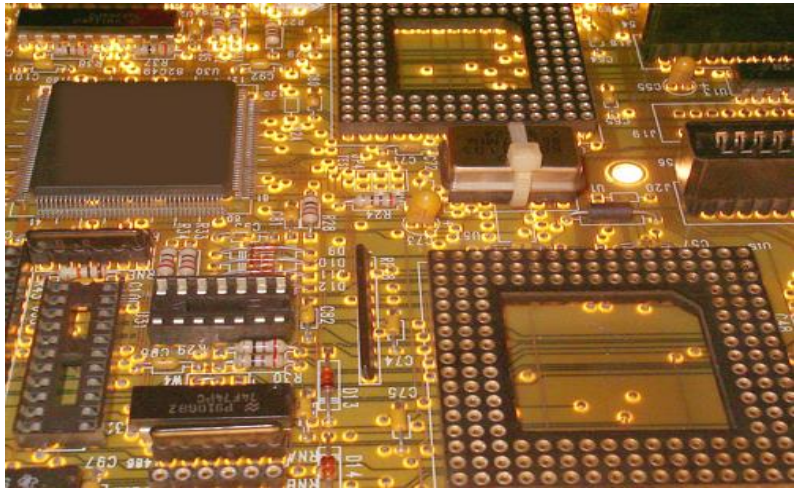
配置优化

郭健美

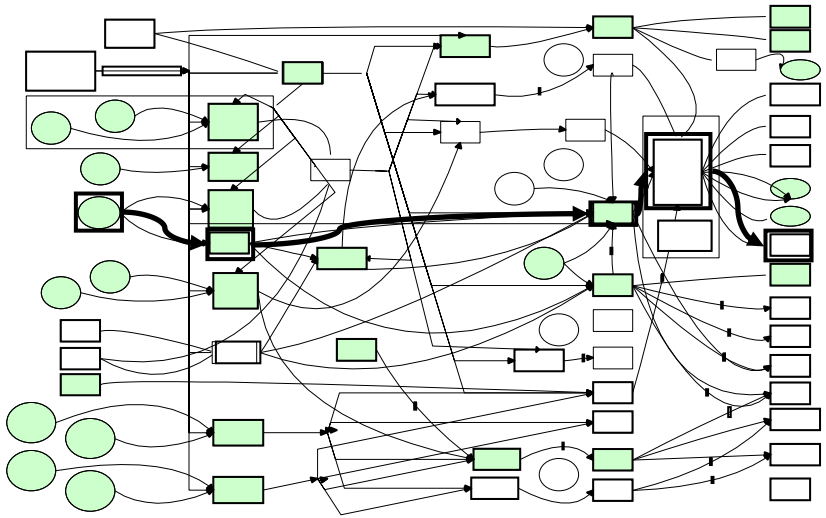
2024年秋

SOLE 系统优化实验室
华东师范大学

软硬件的配置普遍存在



汽车电气系统设计



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. Murashkin, et al.: Automated decomposition and allocation of automotive safety integrity levels using exact solvers. SAE International Journal of Passenger Cars-Electronic and Electrical Systems 8 (2015-01-0156): 70-78 (2015)]

各类可配置的软件系统

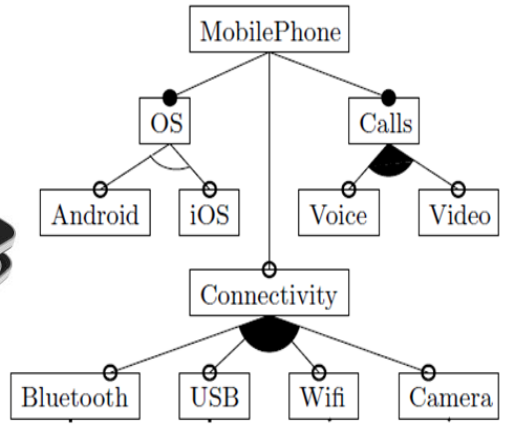
[L. T. Passos, et al.: Coevolution of variability models and related artifacts: a case study from the Linux kernel. SPLC 2013: 91-100]

内容

- 基本概念
- 技术挑战
- 实验设计
- 基于机器学习的方法
- 领域知识驱动的方法

如何生成有效的配置?

模型驱动工程 Model-Driven Engineering



Cross-tree constraints: Video requires Camera

Feature modeling

- Enables formal representation
- Supports automated analysis

using off-the-shelf constraint solvers

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	$\neg(F_1 \wedge F_2)$

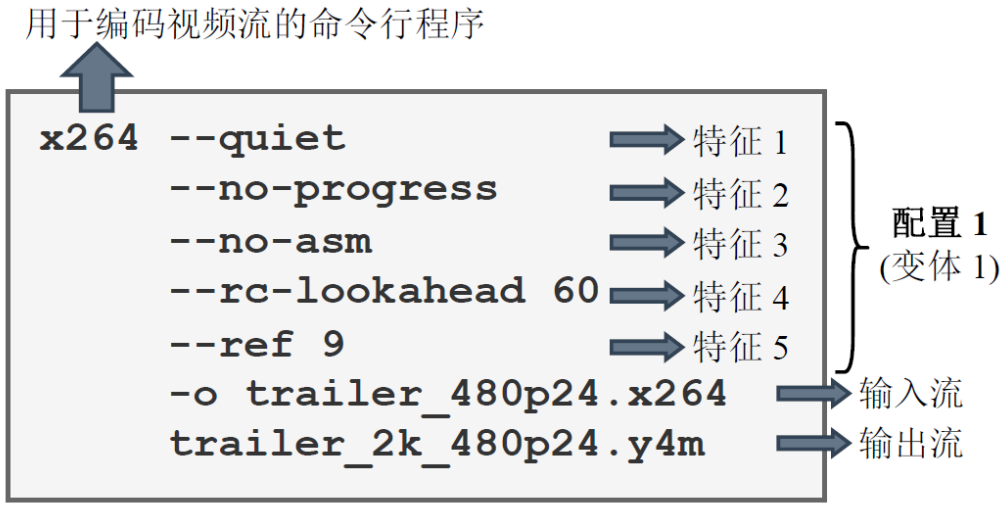
A constraint satisfaction problem

Program synthesis

[K. C. Kang, et al.: Feature-Oriented Domain Analysis (FODA) Feasibility Study. CMU SEI, SEI-90-TR-21, 1990]
[K. Czarnecki, U. W. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison-Wesley, 2000]
[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)]

基本概念

- 配置优化（configuration optimization）的主要目标是找到满足所期望功能和非功能属性的软件系统最优配置
- 以 x264 为例



配置	特征																性能
x_i	$x_i^{(1)}$	$x_i^{(2)}$	$x_i^{(3)}$	$x_i^{(4)}$	$x_i^{(5)}$	$x_i^{(6)}$	$x_i^{(7)}$	$x_i^{(8)}$	$x_i^{(9)}$	$x_i^{(10)}$	$x_i^{(11)}$	$x_i^{(12)}$	$x_i^{(13)}$	$x_i^{(14)}$	$x_i^{(15)}$	$x_i^{(16)}$	y_i
x_1	1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	1	651
x_2	1	1	1	1	1	1	0	1	1	1	0	0	1	0	1	0	536
x_3	1	1	1	1	0	0	0	0	1	1	0	0	1	0	0	1	581
x_4	1	0	0	0	0	0	1	0	1	1	0	0	1	0	1	0	381
x_5	1	1	0	1	0	0	0	1	1	1	0	0	1	0	1	0	424
x_6	1	1	0	0	1	0	1	1	1	1	0	0	1	0	0	1	615
x_7	1	0	1	0	1	1	1	0	1	1	0	0	1	0	1	0	477
x_8	1	0	1	0	0	0	0	1	1	0	0	1	1	1	0	0	263
x_9	1	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	272
x_{10}	1	1	1	1	0	0	0	1	1	0	0	1	1	1	0	0	247
x_{11}	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	1	612
x_{12}	1	0	1	1	1	0	0	0	1	0	0	1	1	0	1	0	510
x_{13}	1	1	1	1	0	1	1	0	1	0	1	0	1	0	0	1	555
x_{14}	1	1	0	0	1	0	1	1	1	0	0	1	1	1	0	0	264
x_{15}	1	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	576
x_{16}	1	0	1	0	1	0	1	1	1	0	1	0	1	1	0	0	268

[J. Guo, et al., Variability-aware performance prediction: A statistical learning approach. ASE 2013: 301-311]

技术挑战

- 配置空间的组合爆炸
- 性能测量的高昂代价
- 复杂隐蔽的特征交互

配置空间的组合爆炸

系统	领域	编程语言	程序代码行数	特征数	配置空间大小
AJStats	代码分析	C	14 782	19	30 256
Apache	Web 服务器	C	230 277	9	192
BDB-C	数据库系统	C	219 811	18	2 560
BDB-J	数据库系统	Java	42 596	26	180
Clasp	回答集求解器	C++	30 871	19	700
HIPAcc	视频处理库	C++	25 605	52	13 485
LLVM	编译器基础设施	C++	47 549	11	1 024
lrzip	压缩库	C++	9 132	19	432
SQLite	数据库系统	C	312 625	39	4 653
x264	视频编码器	C	45 743	16	1 152

系统	版本	特征数	约束数
eCos	3.0	1 244	3 146
FreeBSD	8.0.0	1 396	62 183
Fiasco	2011081207	1 638	5 228
uClinux	20100825	1 850	2 468
Linux	2.6.28.6	6 888	343 944

模型计数（model counting）问题
Sharp-SAT / #SAT 问题

ChatGPT 的可配置参数/特征：GPT-3 约 1 750亿个，GPT-4 约 1.8 万亿个

[J. Guo, et al., Scaling exact multi-objective combinatorial optimization by parallelization. ASE 2014: 409-420]
[J. Guo, et al. Data-efficient performance learning for configurable systems. Empir. Softw. Eng. 23(3): 1826-1867, 2018.]
https://www.technologyreview.com/2023/03/14/1069823/gpt-4-is-bigger-and-better-chatgpt-openai/

性能测量的高昂代价

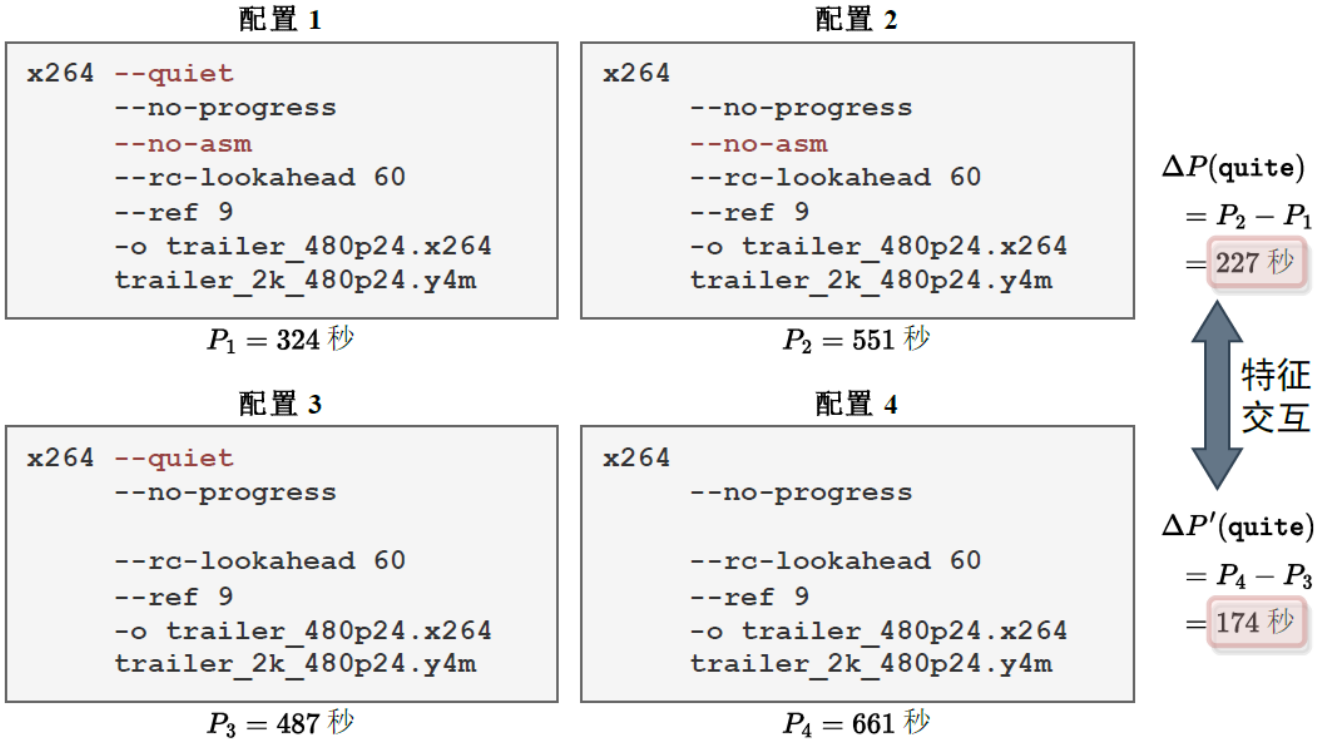
套件	设置	单个基准测试程序运行时间	完整运行时间
SPECrate 2017 Integer	单副本	6~10 分钟	2.5 小时
SPECrate 2017 Floating Point	单副本	5~36 分钟	4.8 小时
SPECspeed 2017 Integer	4 线程	6~15 分钟	3.1 小时
SPECspeed 2017 Floating Point	16 线程	6~75 分钟	4.7 小时

- 设想下为配置空间中每个配置执行一次性能测量所需的总时间
- 为了避免随机误差，可能对每个配置执行多次测量

[A. Sarkar, et al.: Cost-Efficient Sampling for Performance Prediction of Configurable Systems. ASE 2015: 342-352]

复杂隐蔽的特征交互

特征交互（feature interaction）表示一个系统功能特征的行为由于另一个特征（或一组特征）的出现而受到影响。它通常会产生无法预期的系统行为，也会导致难以预测的性能。



[Y. Zhang, et al.: A mathematical model of performance-relevant feature interactions. SPLC 2016: 25-34]

特征交互可能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

如何找到最优配置?

- 实验设计
- 基于机器学习的方法
- 领域知识驱动的方法

实验设计

- 单次单因子设计（one-factor-at-a-time design）
- 全因子设计（full factorial design）
- 部分因子设计（fractional factorial design）
- $2^k r$因子设计
- 随机搜索
- 自动调优（autotuning）

单次单因子设计

假设有 k 个因子，每个因子有 $n_i \ (i = 1, 2, \cdots, k)$ 个水平，则所需实验数量为：

$$1 + \sum_{i=1}^{k} (n_i - 1)$$

全因子设计

假设有 k 个因子，每个因子有 $n_i \ (i = 1, 2, \cdots, k)$ 个水平，则所需实验数量为：

$$\prod_{i=1}^{k} n_i$$

部分因子设计

- 从全因子设计中选出一部分进行实验，以此减少实验数量
- 部分因子设计选出的一部分配置并非随机选择，通常需要遵循一定策略。一种常见的做法是尽量保证每一个因子对各自水平的覆盖程度。

实验	特征															
	$x_i^{(1)}$	$x_i^{(2)}$	$x_i^{(3)}$	$x_i^{(4)}$	$x_i^{(5)}$	$x_i^{(6)}$	$x_i^{(7)}$	$x_i^{(8)}$	$x_i^{(9)}$	$x_i^{(10)}$	$x_i^{(11)}$	$x_i^{(12)}$	$x_i^{(13)}$	$x_i^{(14)}$	$x_i^{(15)}$	$x_i^{(16)}$
1	1	0	1	0	0	1	1	1	0	0	0	0	1	1	1	1
2	0	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1
3	1	0	0	1	1	0	0	1	1	1	0	0	0	0	1	1
4	0	1	0	0	1	1	0	1	1	1	1	0	0	0	0	1
5	1	0	1	0	0	1	1	0	1	1	1	1	0	0	0	0
6	0	1	1	1	0	0	1	0	0	1	1	1	1	0	0	0
7	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0	0
8	0	1	0	0	1	1	0	0	0	0	0	1	1	1	1	0

如何筛选因子？

$2^k r$ 因子设计

- 共 k 个因子
- 每个因子有 2 个备选水平
- 每组实验被重复执行 r 次

- 用于确定 k 个因子的影响

2^2 因子设计

以 x264 为例，考虑两个布尔型特征 no-8x8dct （即禁用自适应空间大小）和 no-mbtree （即禁用宏块树速率控制），分别简记为 A 和 B，并定义两个特征变量 x_A 和 x_B

$$x_A = \begin{cases} -1 & \text{当特征 A 未启用} \\ 1 & \text{当特征 A 被启用} \end{cases}$$

$$x_B = \begin{cases} -1 & \text{当特征 B 未启用} \\ 1 & \text{当特征 B 被启用} \end{cases}$$

2^2 因子设计

测量四组实验的性能值 y

实验	x_A	x_B	y
1	-1	-1	275
2	-1	1	310
3	1	-1	261
4	1	1	291

2^2 因子设计

为了量化两个因子及其交互对性能的影响，构建一个非线性回归（nonlinear regression）模型如下：

$$y = c_0 + c_A x_A + c_B x_B + c_{AB} x_A x_B$$

$$\begin{cases} c_0 - c_A - c_B + c_{AB} = y_1 \\ c_0 - c_A + c_B - c_{AB} = y_2 \\ c_0 + c_A - c_B - c_{AB} = y_3 \\ c_0 + c_A + c_B + c_{AB} = y_4 \end{cases} \quad \begin{cases} c_0 = \frac{1}{4}(y_1 + y_2 + y_3 + y_4) \\ c_A = \frac{1}{4}(-y_1 - y_2 + y_3 + y_4) \\ c_B = \frac{1}{4}(-y_1 + y_2 - y_3 + y_4) \\ c_{AB} = \frac{1}{4}(y_1 - y_2 - y_3 + y_4) \end{cases}$$

$$y = \frac{1137}{4} - \frac{33}{4}x_A + \frac{65}{4}x_B - \frac{5}{4}x_A x_B$$

2^2 因子设计

符号表（sign table）方法

实验	x_A	x_B	y
1	-1	-1	275
2	-1	1	310
3	1	-1	261
4	1	1	291

I	x_A	x_B	x_Ax_B	y
1	-1	-1	1	275
1	-1	1	-1	310
1	1	-1	-1	261
1	1	1	1	291
1137	-33	65	-5	(加和)
$\frac{1137}{4}$	$-\frac{33}{4}$	$\frac{65}{4}$	$-\frac{5}{4}$	(平均)

$$y = \frac{1137}{4} - \frac{33}{4}x_A + \frac{65}{4}x_B - \frac{5}{4}x_Ax_B$$

2^2 因子设计

分析各因子对性能的影响

- 计算性能值的总离差平方和 (Total Sum of Squares, TSS)

$$\mathrm{TSS} = \sum_{i=1}^{2^2} (y_i - \bar{y})^2$$

- TSS可推导分解为三部分SSA (Sum of Squares due to A)，SSB (Sum of Squares due to B) 和 SSAB (Sum of Squares due to the interaction of A and B)

$$\begin{aligned}\mathrm{TSS} &= \mathrm{SSA} + \mathrm{SSB} + \mathrm{SSAB} \\ &= 2^2 {c_A}^2 + 2^2 {c_B}^2 + 2^2 {c_{AB}}^2 \\ &= \frac{33^2}{4} + \frac{65^2}{4} + \frac{5^2}{4} \\ &= \frac{5339}{4}\end{aligned}$$

- 计算各个因子所造成的离差平方和占总离差平方和的比例，即 $\frac{SSA}{TSS}$、$\frac{SSB}{TSS}$ 和 $\frac{SSAB}{TSS}$ 得到 特征A、特征B、以及它们的交互 对性能的影响分别为 20.4%、79.1%、0.5%。

随机搜索

- 简单易行，开销可控

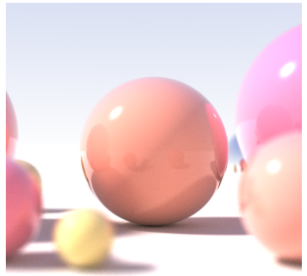
- 如何做到完全的“真随机”而非“伪随机”？
  - 分层采样（stratified sampling）
  - 蒙特卡洛采样（Monte Carlo sampling）
  - 拉丁超立方采样（Latin Hypercube Sampling, LHS）
  - 索伯采样（Sobol sampling）

[C. Kaltenecker, et al.: Distance-based sampling of software configuration spaces. ICSE 2019 : 1084-1094]

自动调优

1. 定义配置空间 C
2. 从配置空间中根据不同的实验设计策略选择一个配置 c
3. 测量并评估配置 c 的性能值 p
4. 如果调优时间达到限制，结束流程并从所有评测过的配置中选出 BKC
5. 如果调优时间未达到限制，继续下一步
6. 如果 p 满足性能要求，结束流程并输出配置 c 及其性能值 p
7. 如果 p 不满足性能要求，则从配置空间中再选择一个新的配置 c'，令 $c = c'$
8. 返回步骤 3

例子：OpenTuner

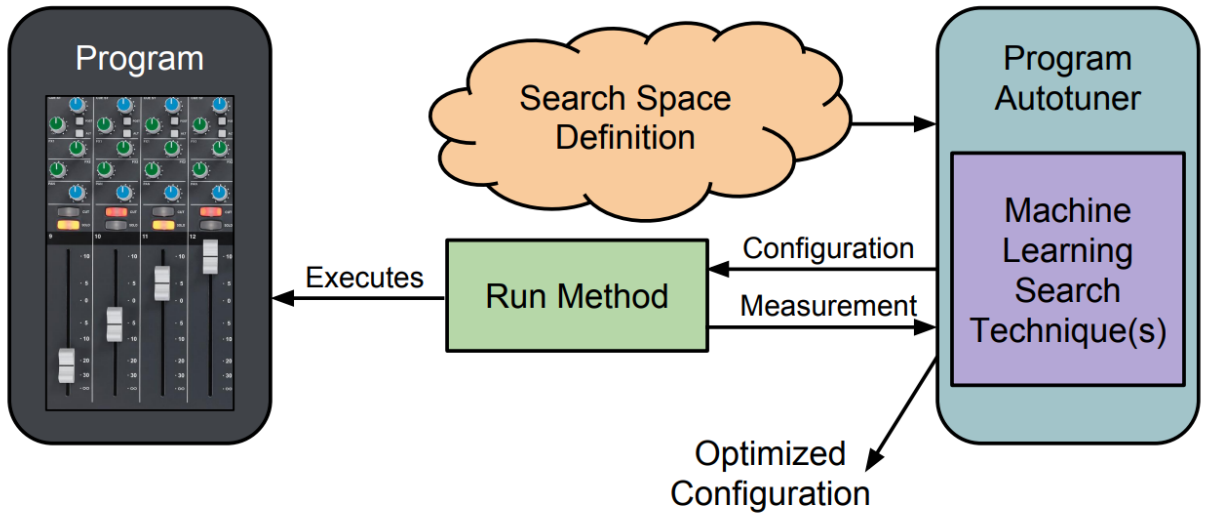


An example ray tracer program: `raytracer.cpp`

```
$ g++ -O3 -o raytracer_a raytracer.cpp
$ time ./raytracer_a
./raytracer_a  0.17s user 0.00s system 99% cpu 0.175 total
```

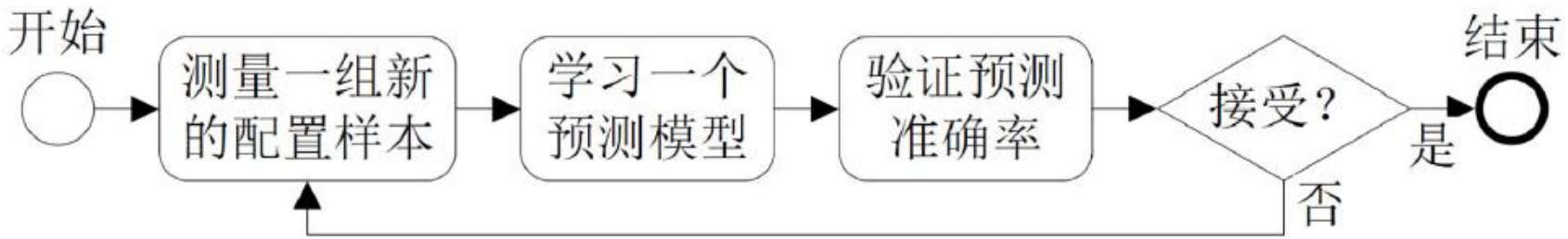
1.47x speedup with:

```
$ g++ -O3 -o raytracer_b apps/raytracer.cpp -funsafe-math-optimizations -fwrapv
    ↪ -fno-expensive-optimizations --param=max-peel-branches=115 -fweb -fno-
    ↪ cx-fortran-rules  --param=max-inline-recursive-depth=25 -fno-btr-bb-
    ↪ exclusive -fno-tree-ch --param=iv-max-considered-uses=69 -fgcse-las -
    ↪ ftree-loop-distribution --param=max-goto-duplication-insns=11 --param=
    ↪ max-hoist-depth=44 -fsched-stalled-insns-dep --param=max-once-peeled-
    ↪ insns=165 --param=max-pipeline-region-insns=316 --param=iv-consider-all
    ↪ -candidates-bound=75
$ time ./raytracer_b
./raytracer_b  0.12s user 0.00s system 99% cpu 0.119 total
```



https://opentuner.org/

基于机器学习的方法



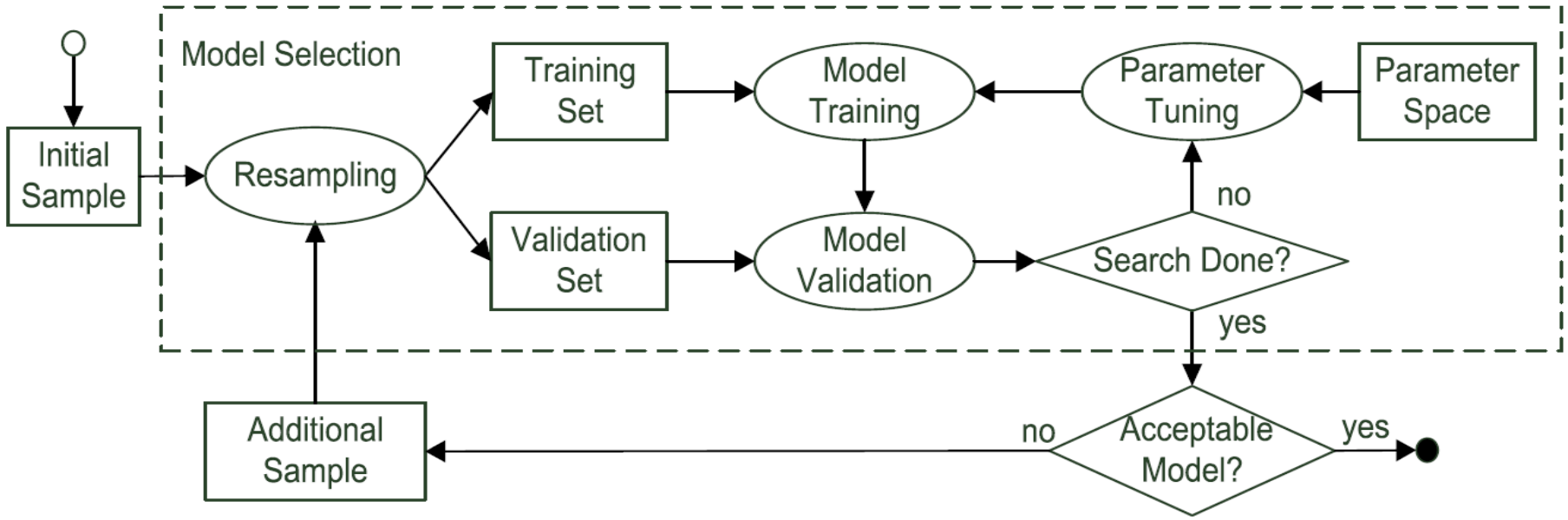
Data-Efficient Performance Learning

- **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

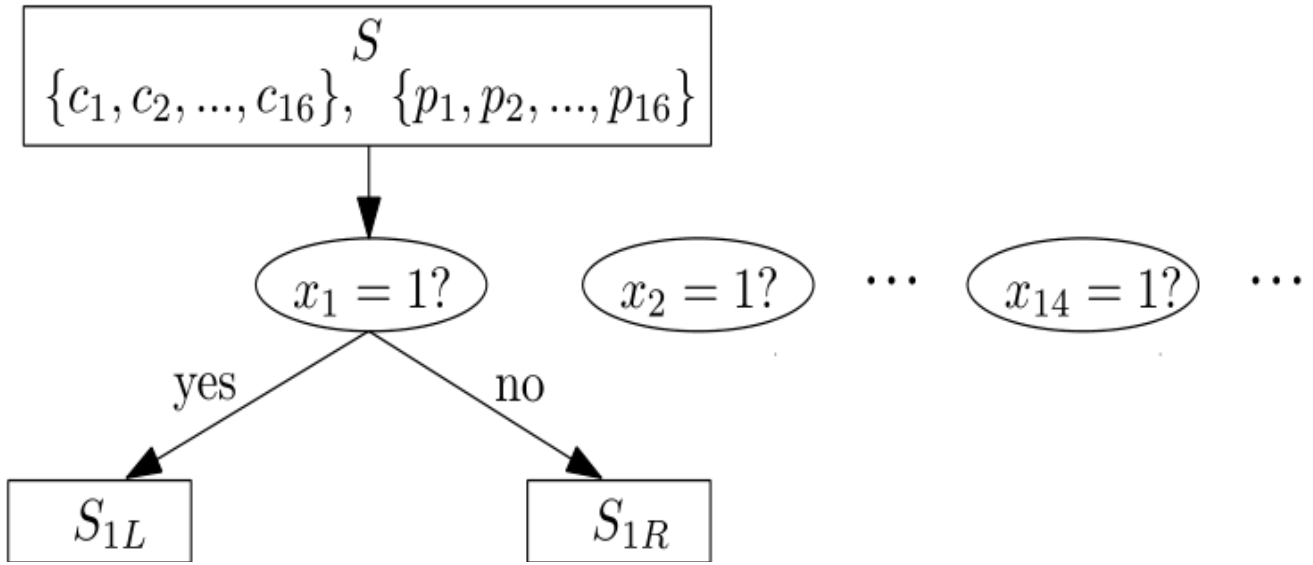
Data-Efficient Machine Learning

基于机器学习的方法



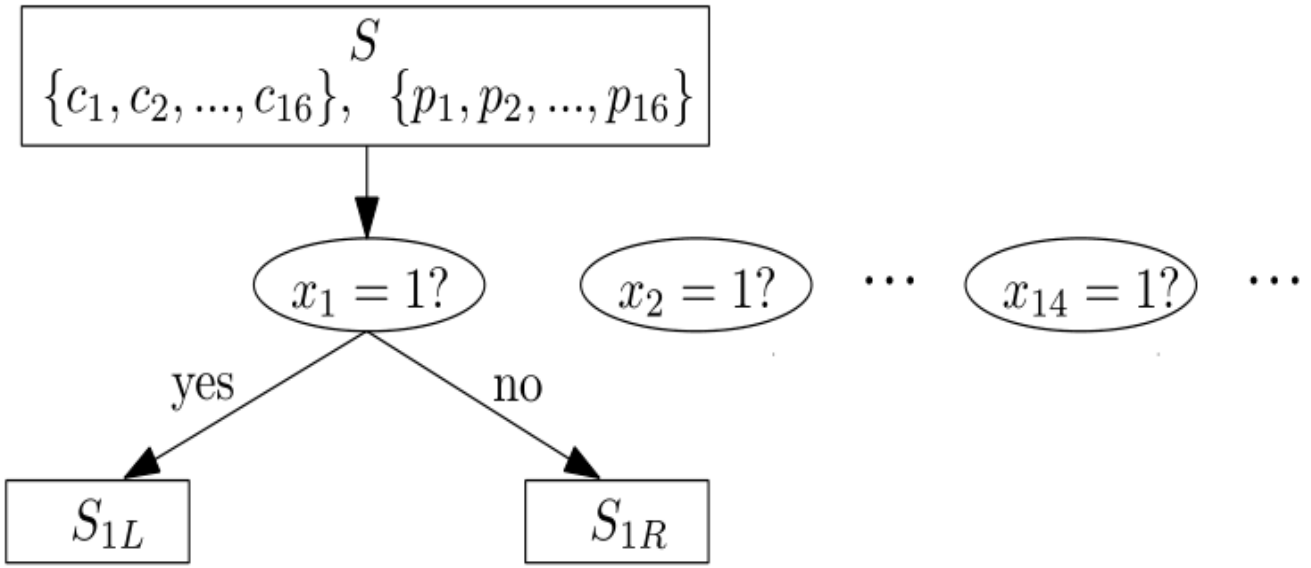
[J. Guo, et al. Data-efficient performance learning for configurable systems. Empir. Softw. Eng. 23(3): 1826-1867, 2018.]

例子：分类回归树 Classification And Regression Trees (CART)



[J. Guo, et al., Variability-aware performance prediction: A statistical learning approach. ASE 2013: 301-311]

例子：分类回归树 Classification And Regression Trees (CART)



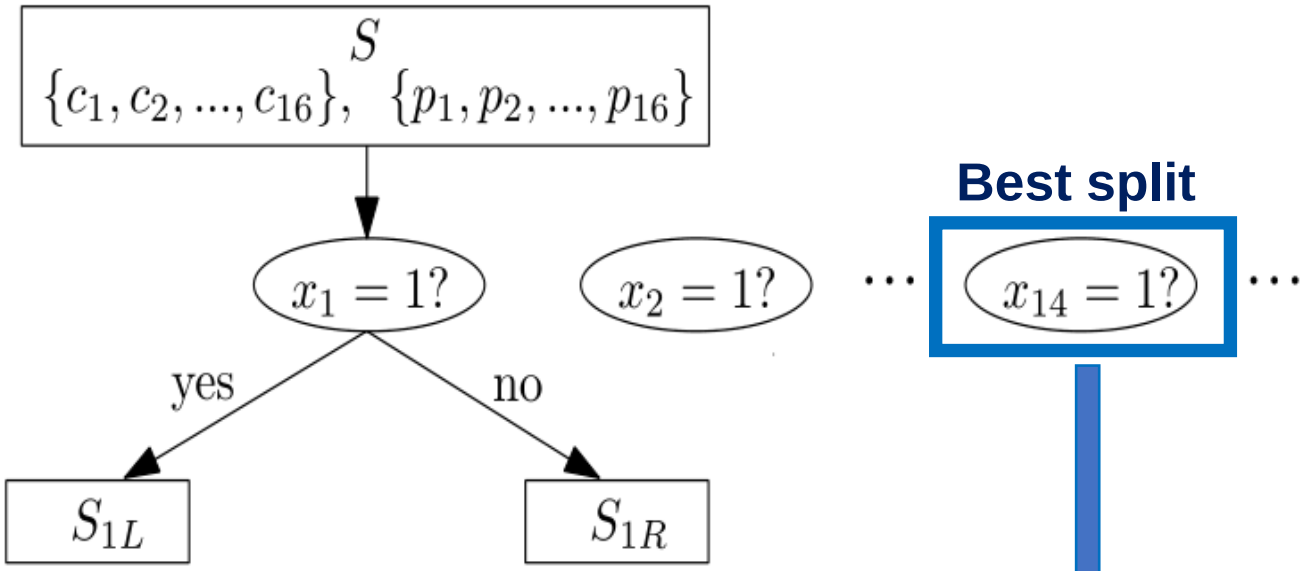
Sample mean $\ell_{S_{iL}} = \frac{1}{|S_{iL}|} \sum_{p_j \in S_{iL}} p_j \qquad \ell_{S_{iR}} = \frac{1}{|S_{iR}|} \sum_{p_j \in S_{iR}} p_j$

Squared error loss $\sum_{p_j \in S_{iL}} (p_j - \ell_{S_{iL}})^2 \qquad \sum_{p_j \in S_{iR}} (p_j - \ell_{S_{iR}})^2$

Sum of squared error loss $\sum_{p_j \in S_{iL}} (p_j - \ell_{S_{iL}})^2 + \sum_{p_j \in S_{iR}} (p_j - \ell_{S_{iR}})^2$ **is minimal**

[J. Guo, et al., Variability-aware performance prediction: A statistical learning approach. ASE 2013: 301-311]

例子：分类回归树 Classification And Regression Trees (CART)

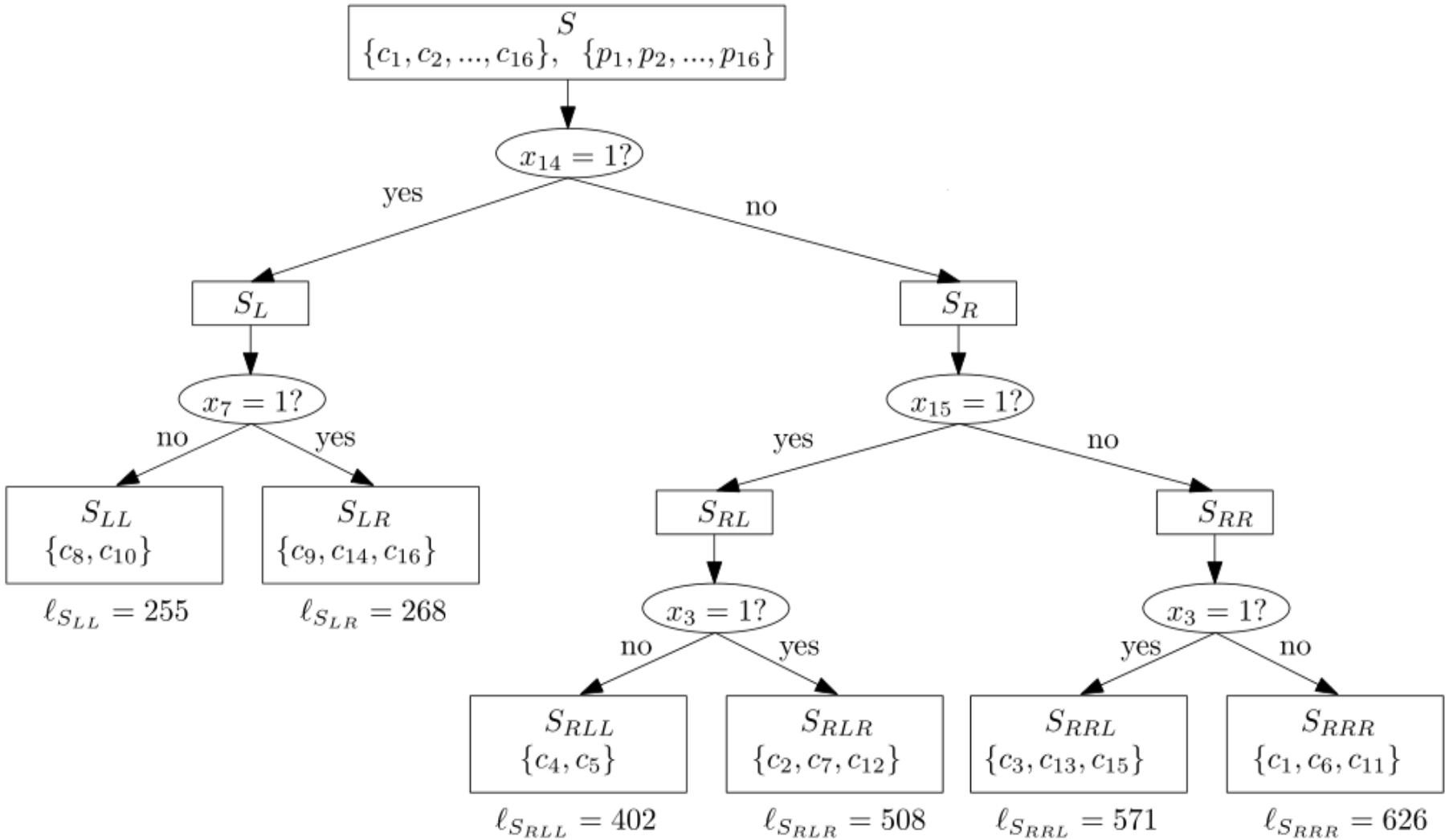


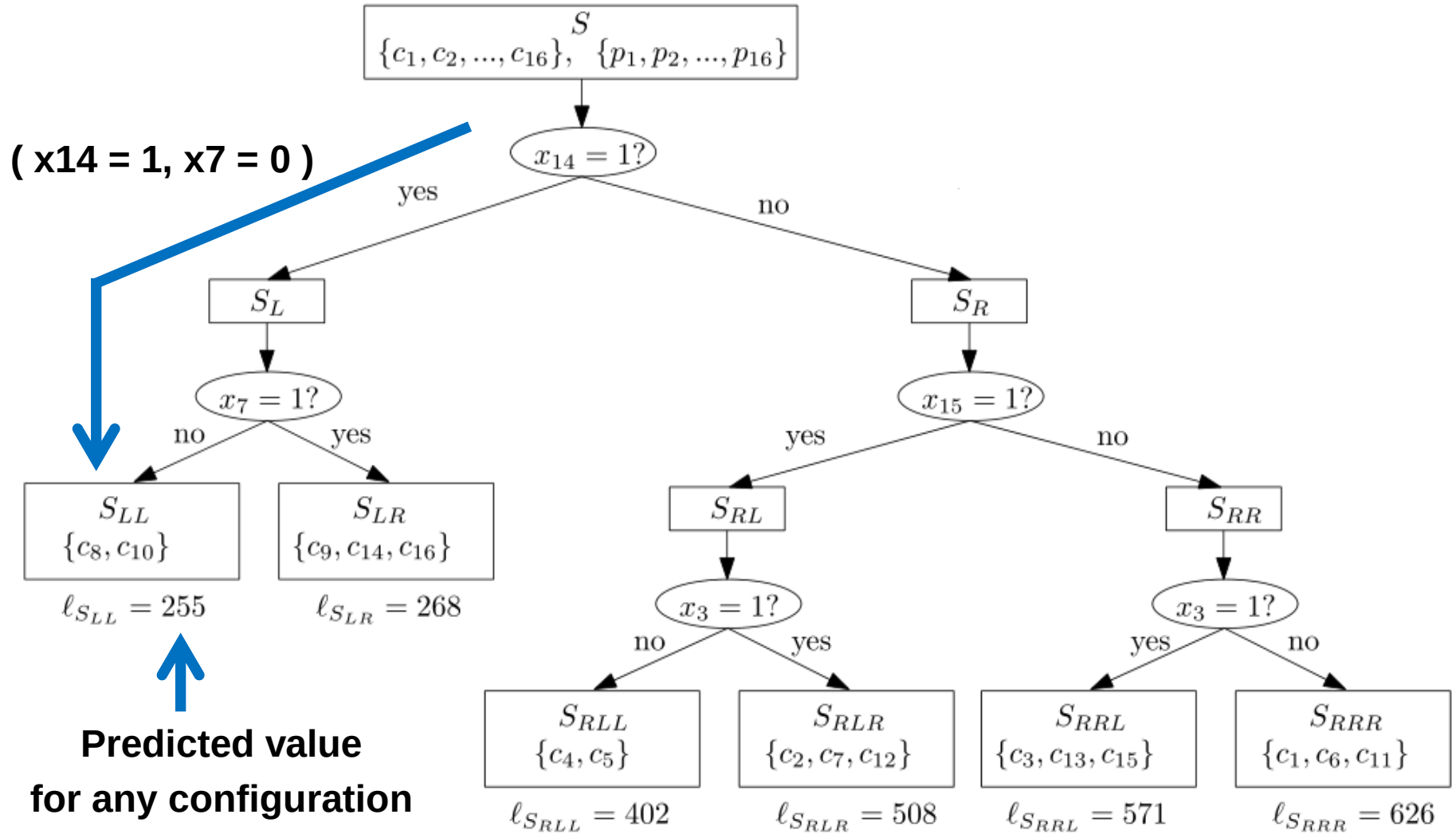
Sample mean $\ell_{S_{iL}} = \frac{1}{|S_{iL}|} \sum_{p_j \in S_{iL}} p_j$ $\ell_{S_{iR}} = \frac{1}{|S_{iR}|} \sum_{p_j \in S_{iR}} p_j$

Squared error loss $\sum_{p_j \in S_{iL}} (p_j - \ell_{S_{iL}})^2$ $\sum_{p_j \in S_{iR}} (p_j - \ell_{S_{iR}})^2$

Sum of squared error loss $\sum_{p_j \in S_{iL}} (p_j - \ell_{S_{iL}})^2 + \sum_{p_j \in S_{iR}} (p_j - \ell_{S_{iR}})^2$ **is minimal**

[J. Guo, et al., Variability-aware performance prediction: A statistical learning approach. ASE 2013: 301-311]


S
$\{c_1, c_2, ..., c_{16}\}$, $\{p_1, p_2, ..., p_{16}\}$
$x_{14} = 1?$
yes
no
S_L
S_R
$x_7 = 1?$
$x_{15} = 1?$
no
yes
yes
no
S_{LL}
$\{c_8, c_{10}\}$
S_{LR}
$\{c_9, c_{14}, c_{16}\}$
S_{RL}
S_{RR}
$\ell_{S_{LL}} = 255$
$\ell_{S_{LR}} = 268$
$x_3 = 1?$
$x_3 = 1?$
no
yes
yes
no
S_{RLL}
$\{c_4, c_5\}$
S_{RLR}
$\{c_2, c_7, c_{12}\}$
S_{RRL}
$\{c_3, c_{13}, c_{15}\}$
S_{RRR}
$\{c_1, c_6, c_{11}\}$
$\ell_{S_{RLL}} = 402$
$\ell_{S_{RLR}} = 508$
$\ell_{S_{RRL}} = 571$
$\ell_{S_{RRR}} = 626$



Predicted value for any configuration selecting feature_14 and deselecting feature_7

领域知识驱动的方法

- New architectural features
- New instructions
- …

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

领域知识驱动的方法

- New architectural features
- New instructions
- …

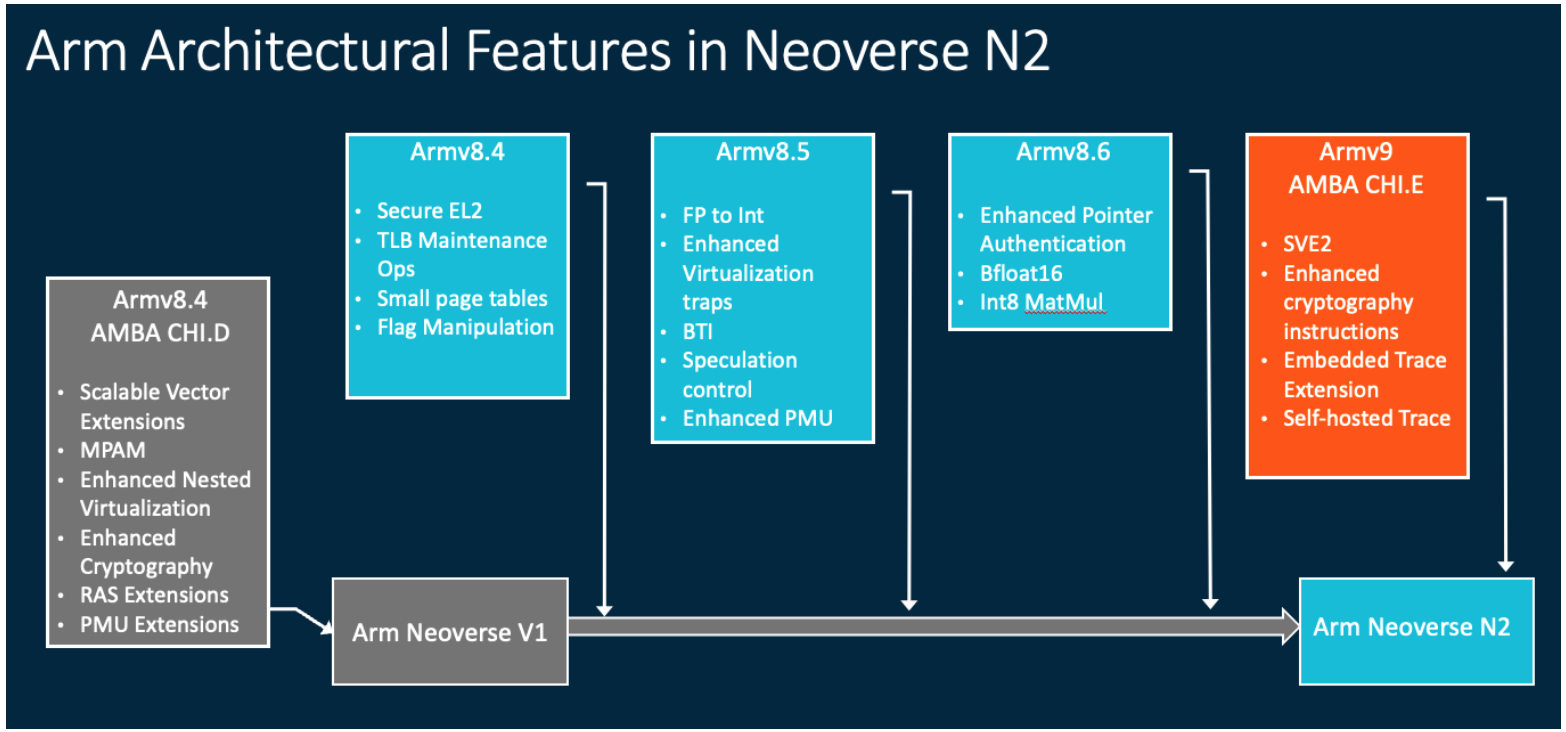
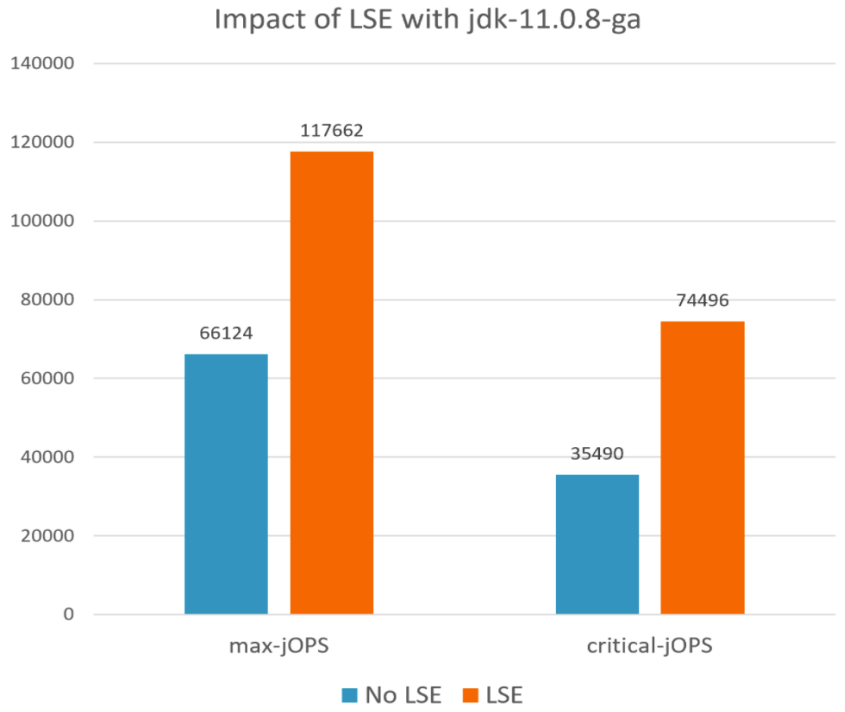


Photo from https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/arm-neoverse-n2-industry-leading-performance-efficiency

例子：引入新型指令

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]

小结

- 软件系统的可配置性是普遍存在的，不同的配置可能会导致不同的性能
- 配置优化需要考虑配置组合爆炸、测量代价和特征交互三大挑战
- 实验设计可以帮助判别和筛选有影响力的因子
- 可以通过实验设计、机器学习和领域知识找到最优配置，实际中，这些方法常常被结合起来使用