
Bayesian Optimization Methods for
Computer Systems and
Architecture Research
IISWC 2021

Devesh Tiwari

Rohan Basu Roy

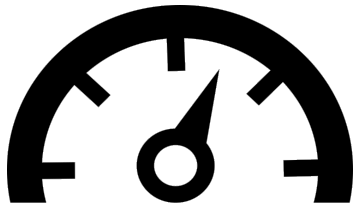
Tirthak Patel

We come from computer systems, architecture and HPC backgrounds

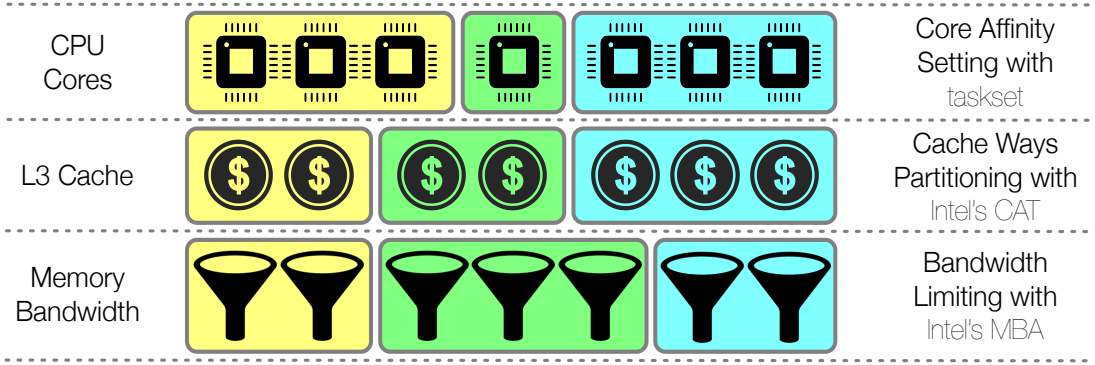
Systems engineers' point of view of Machine Learning (Bayesian Optimization)

So why apply machine learning or Bayesian inference?

System Goal
Maximize Utilization



User Goal
Maximize Throughput

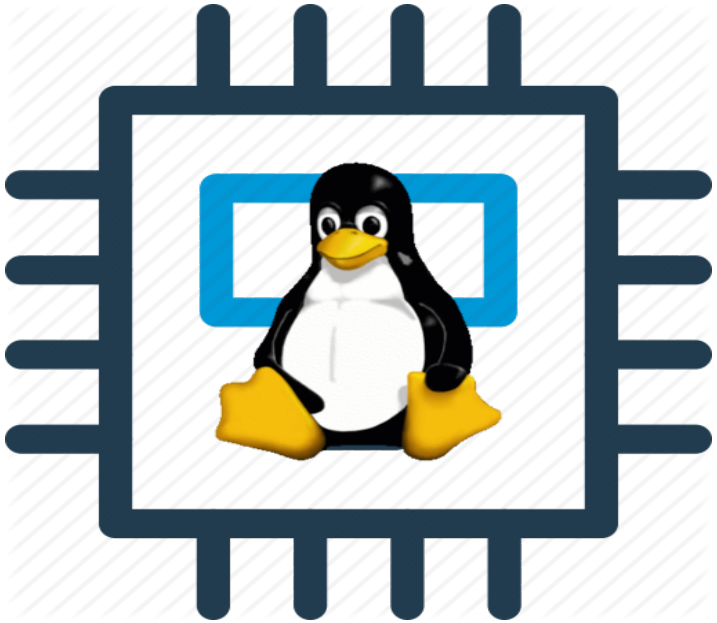


Parameter Tuning

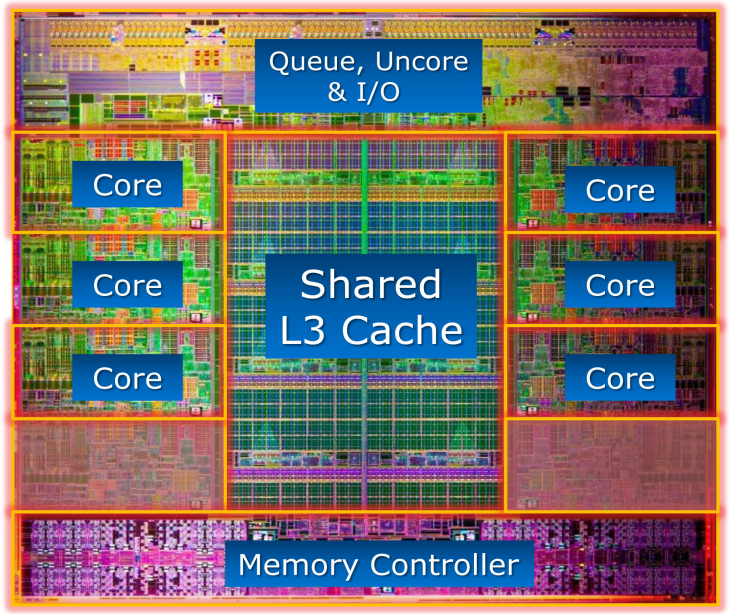
Microservice Scheduling

NETFLIX

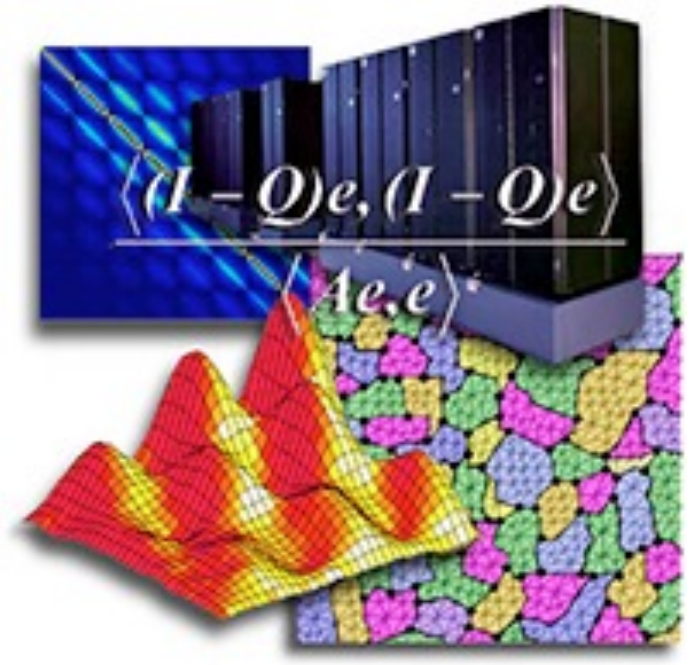
Plenty of fish in the sea
…. but we need to find "the one"



H/W and kernel parameters



Resource partitioning configurations

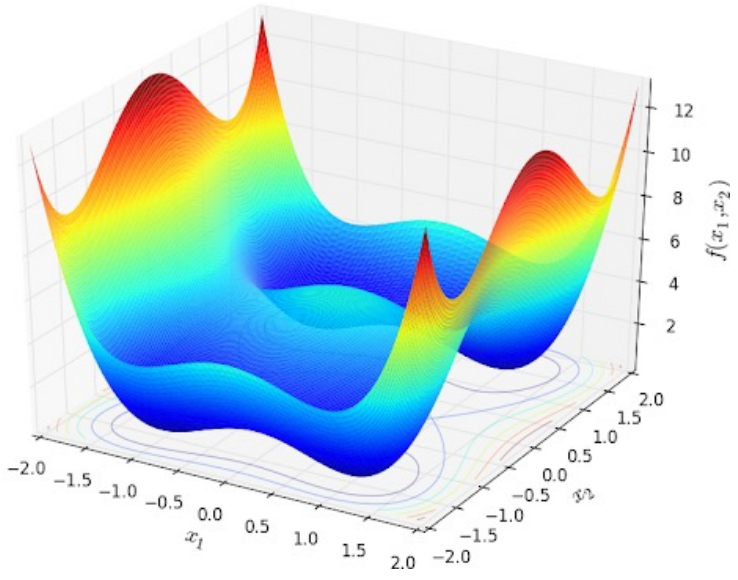


S/W parameters

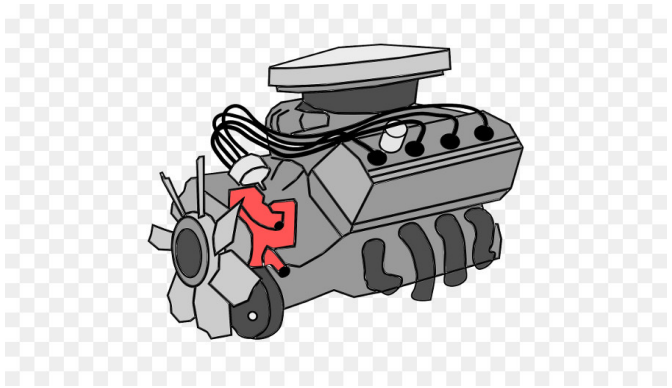
The mining for gold began long ago

Daniel Krige
1950 gold
mining

Jonas Mockus 70s
combinatorial
optimization



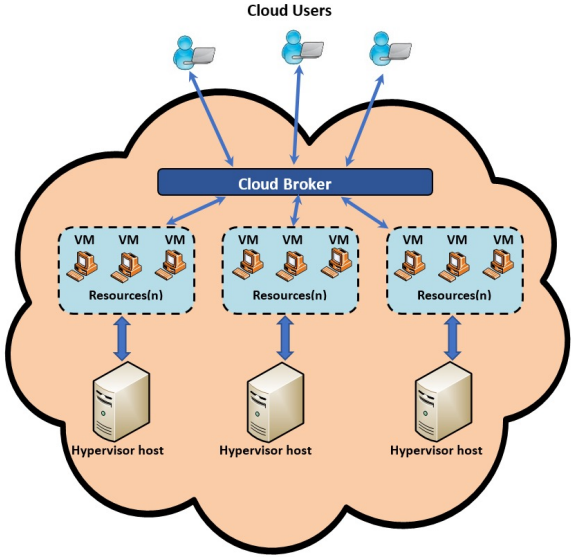
Jones 80s car
motors



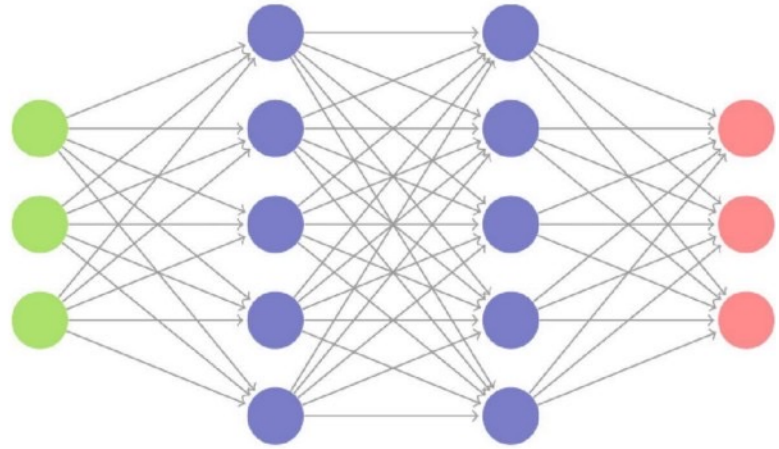
Where is a Bayesian approach useful?

Running on supercomputers

Scheduling on clouds



Hyperparameter tuning

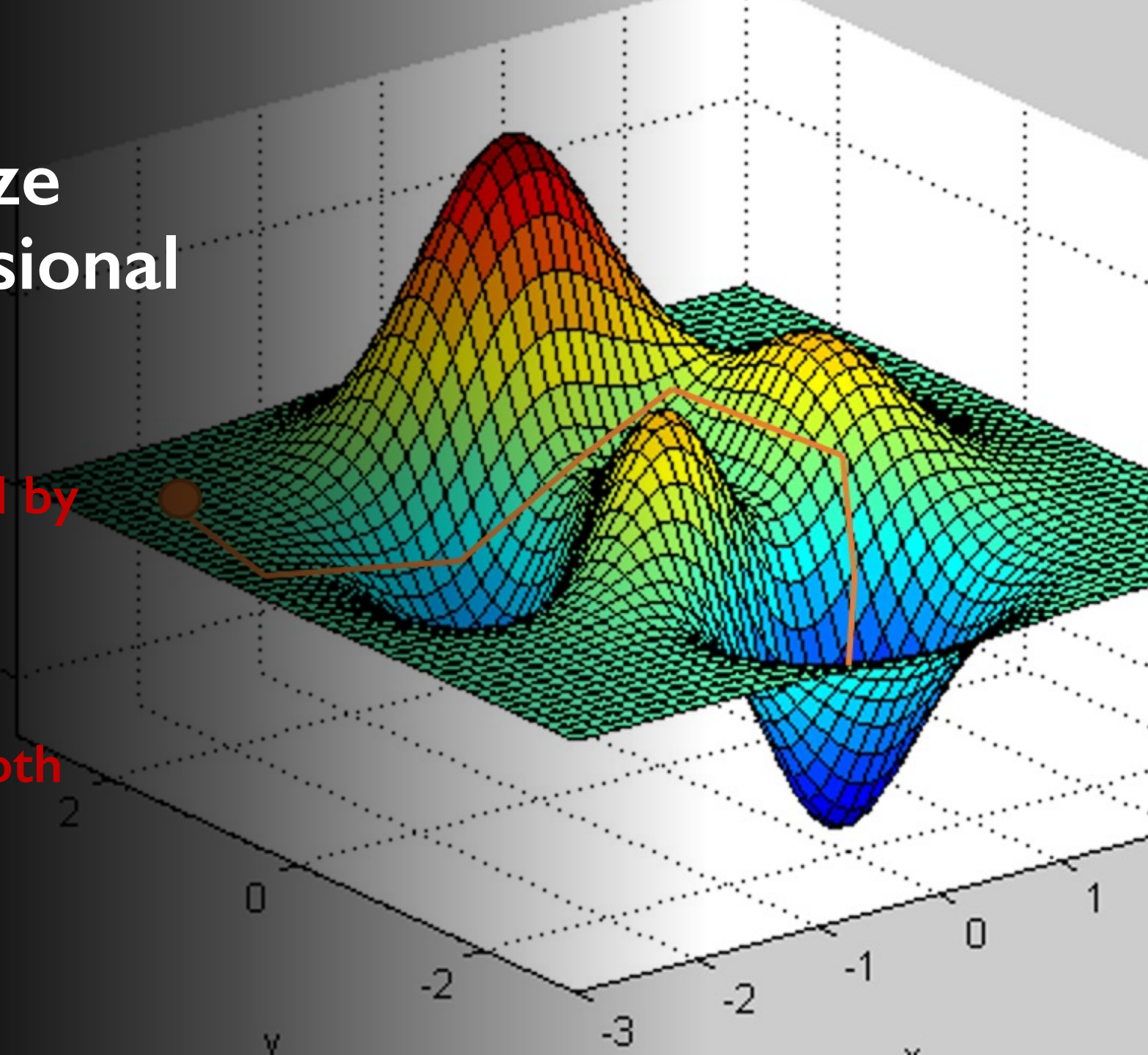


Useful when it is expensive to perform individual experiments

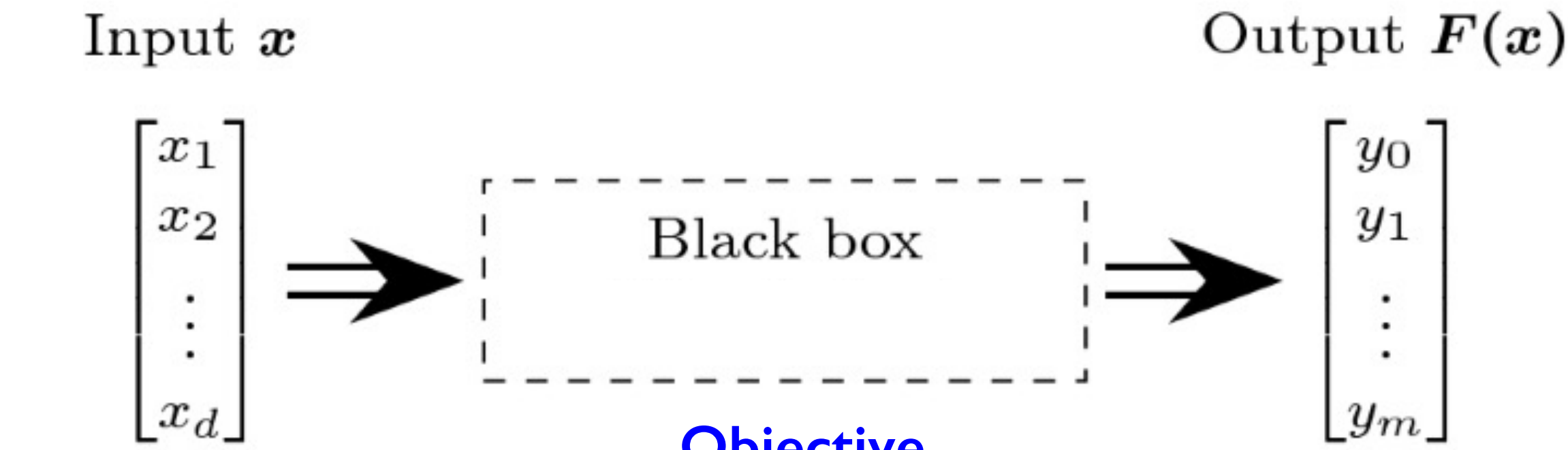
We want to optimize over a multi-dimensional search space

- ✓ The search space is defined by several constraints

- ✓ The search space can be both convex and non-convex



The objective function is unknown to us

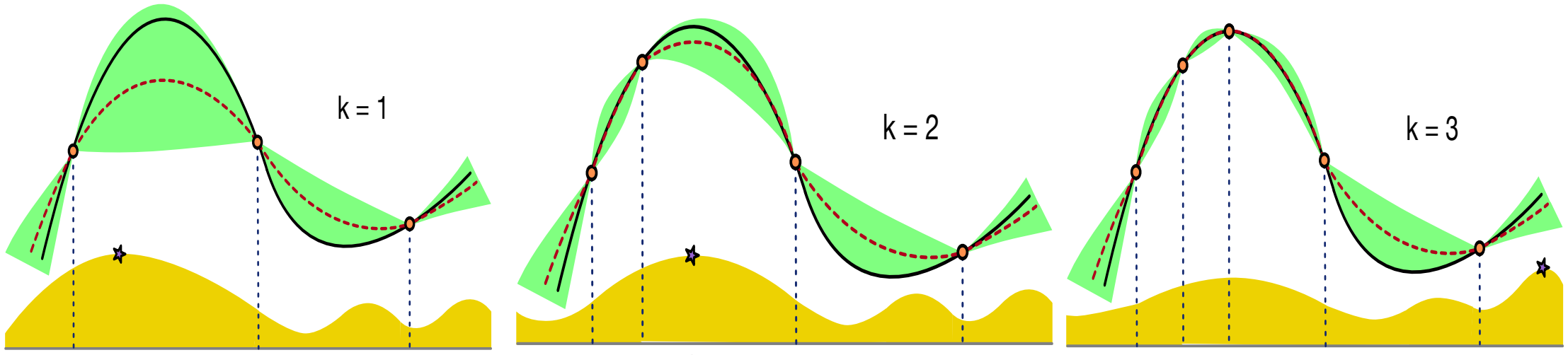


Parameter configurations

Objective Function

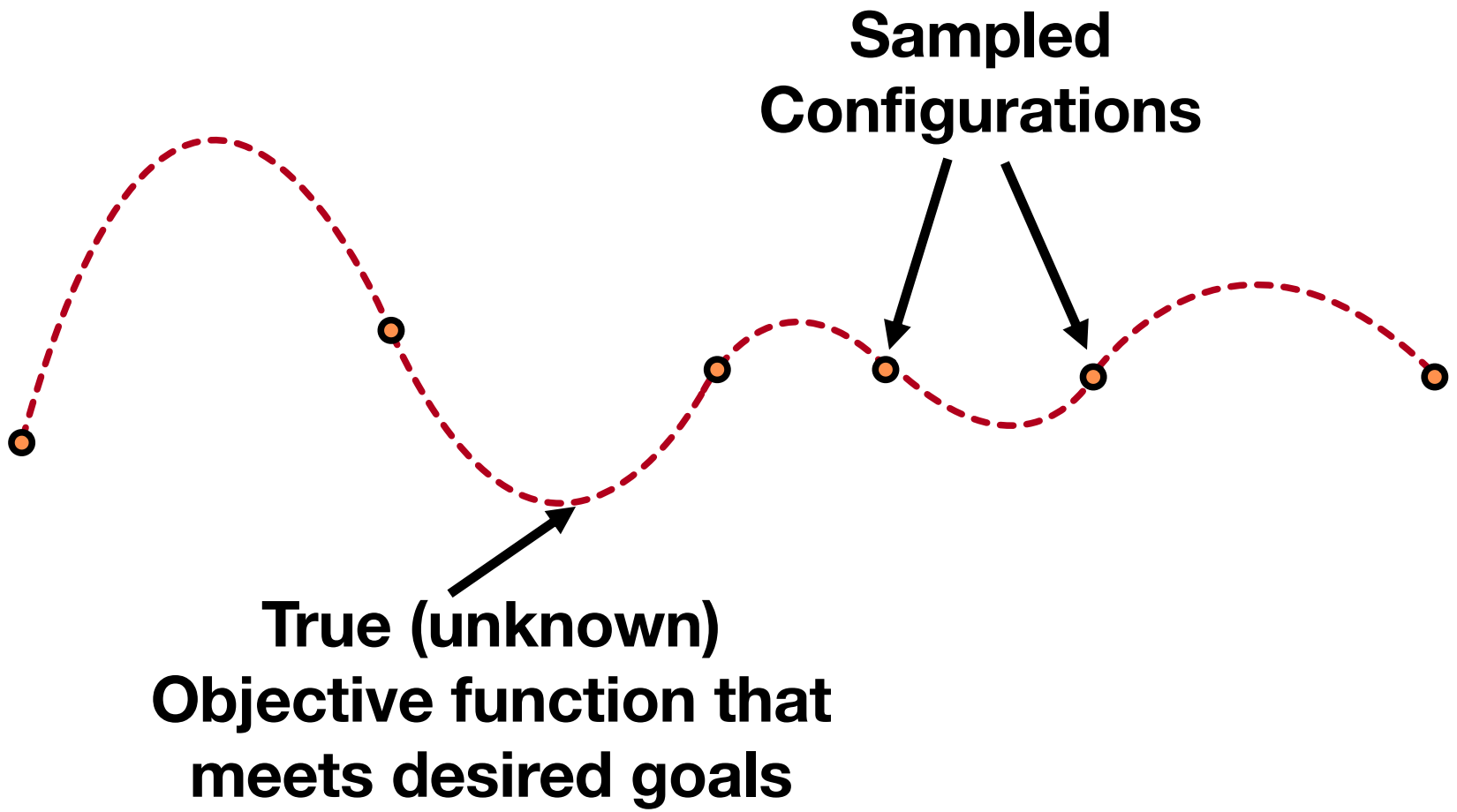
Sampled Output

How do we learn the true objective function?

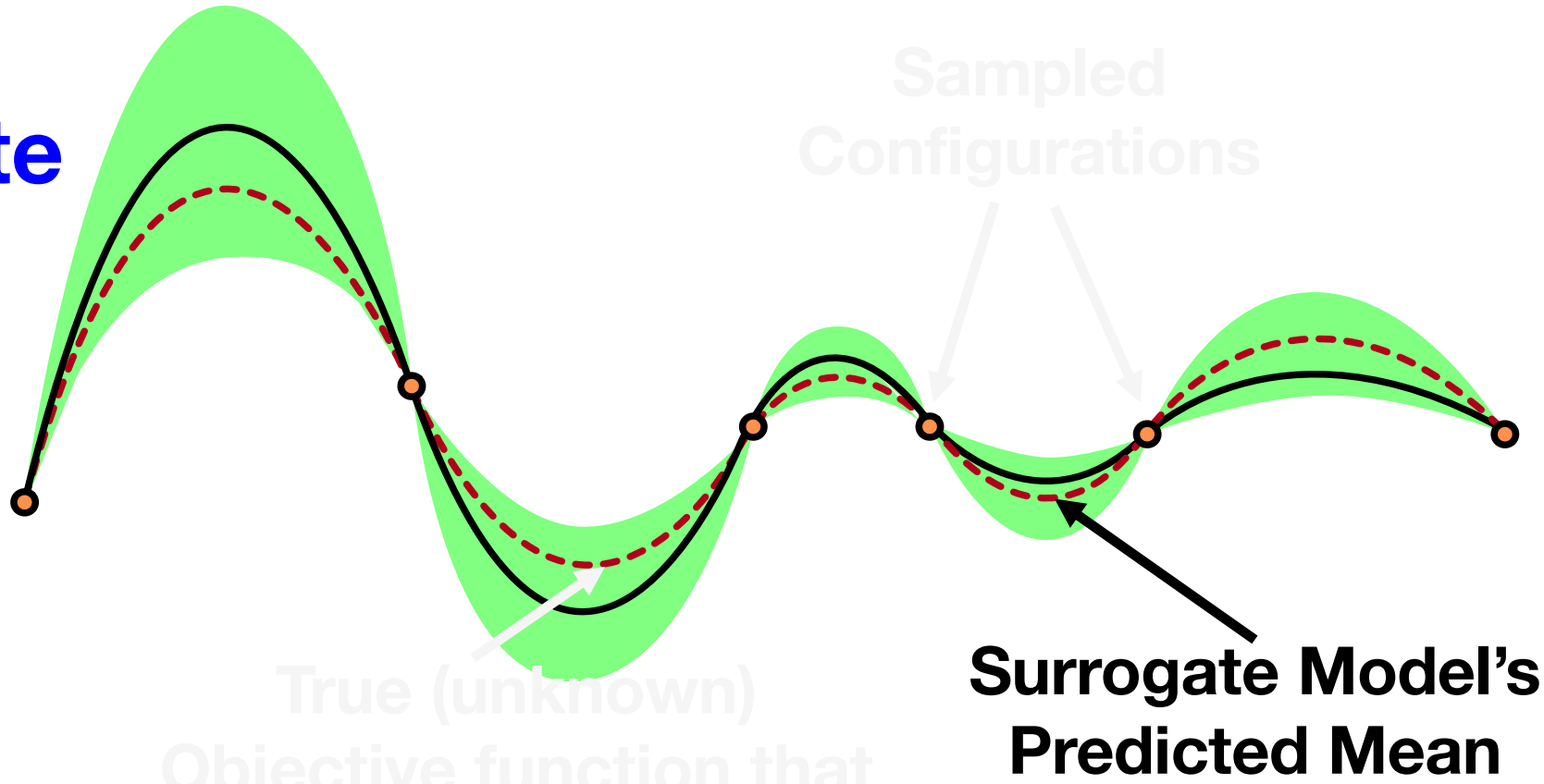


We learn via repeated sampling of the true objective functions, which are individually expensive

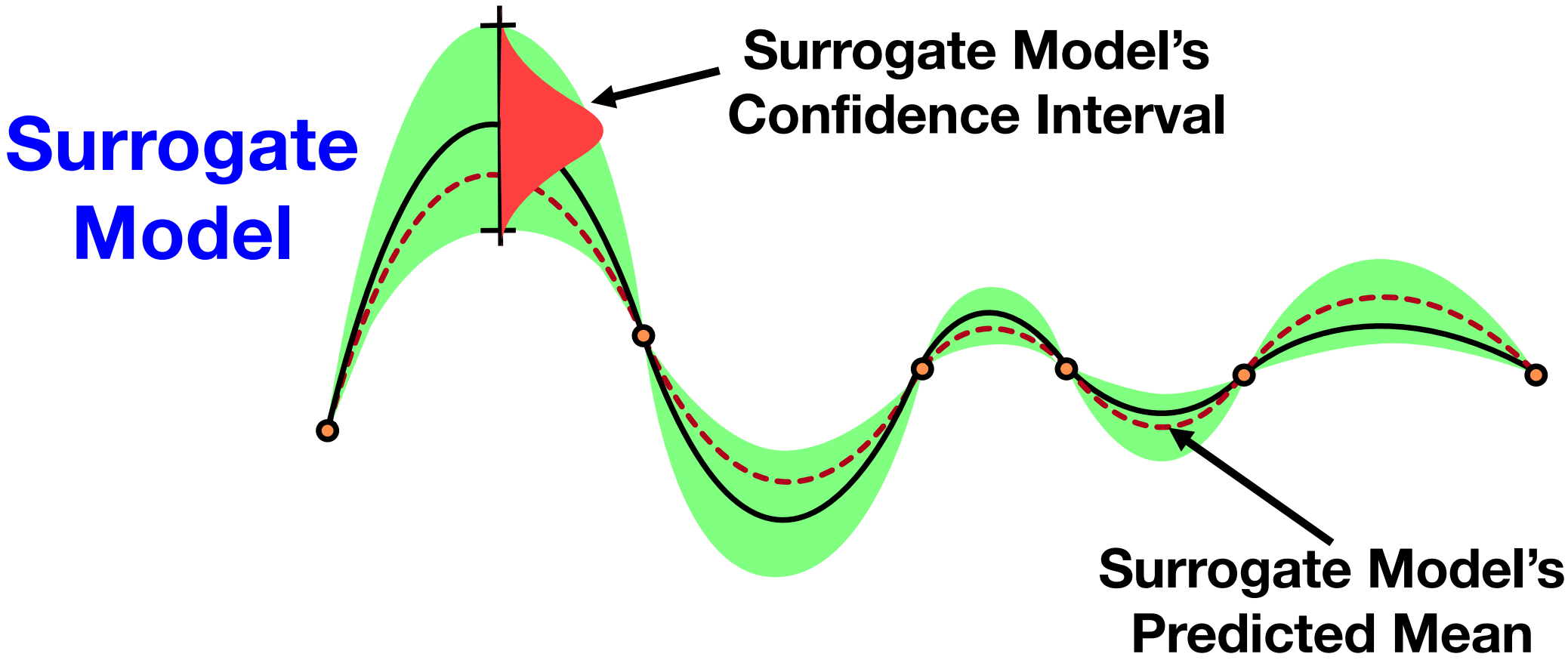
The overall mechanics of a Bayesian Optimization (BO)



The overall mechanics of a Bayesian Optimization (BO)

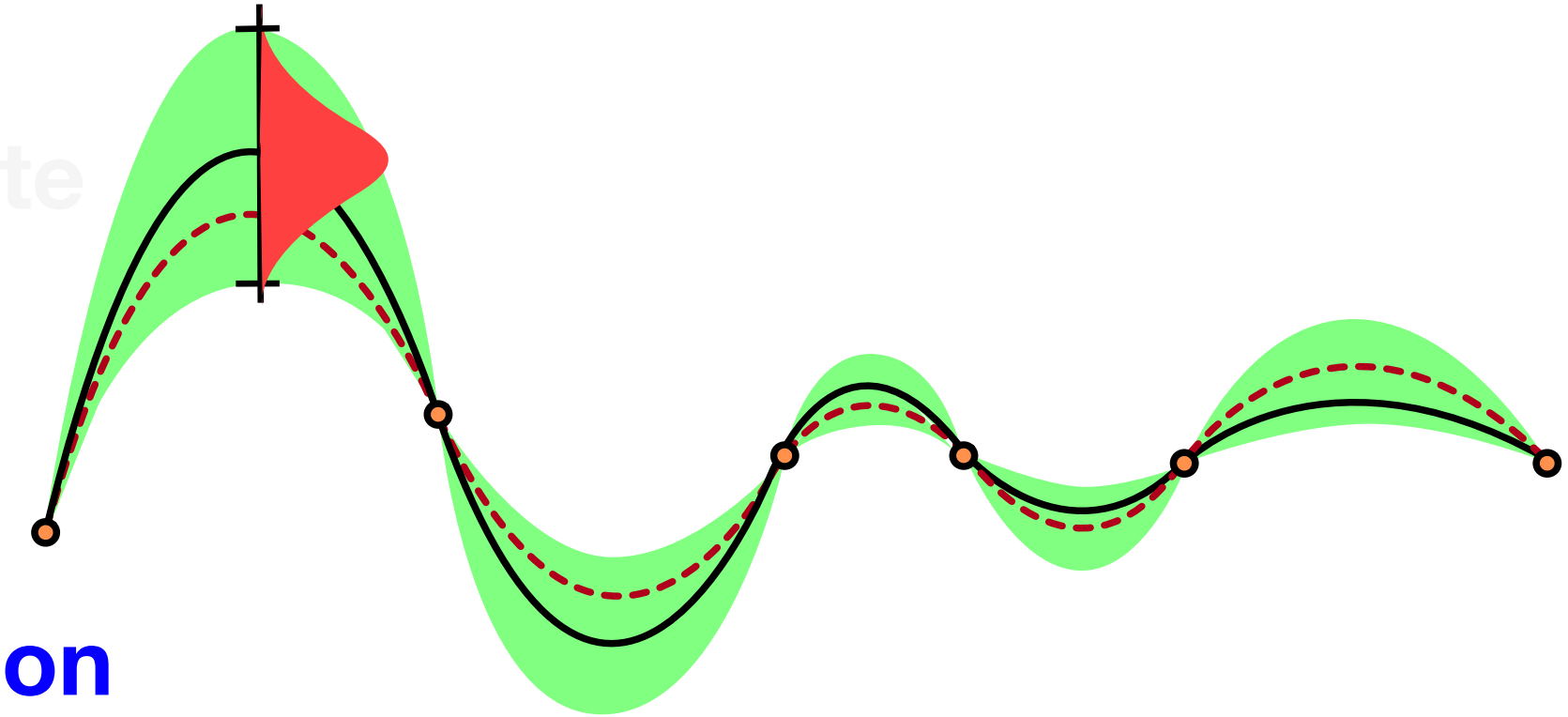


The surrogate model is a Gaussian process

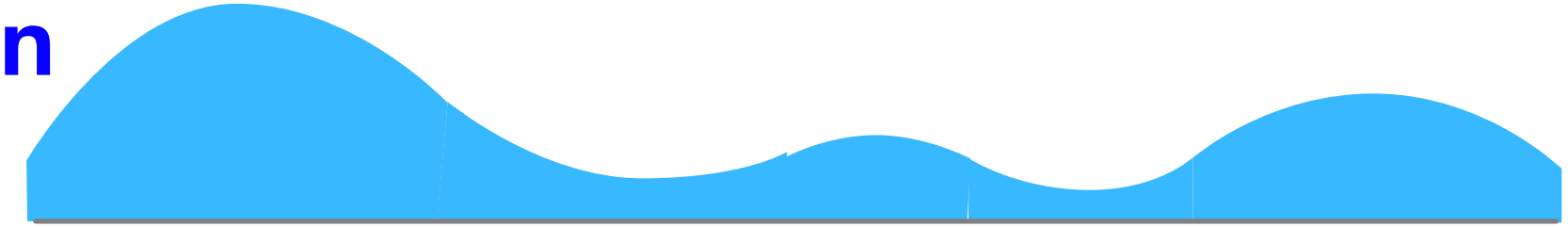


Bayesian Optimization Based Configuration Exploration

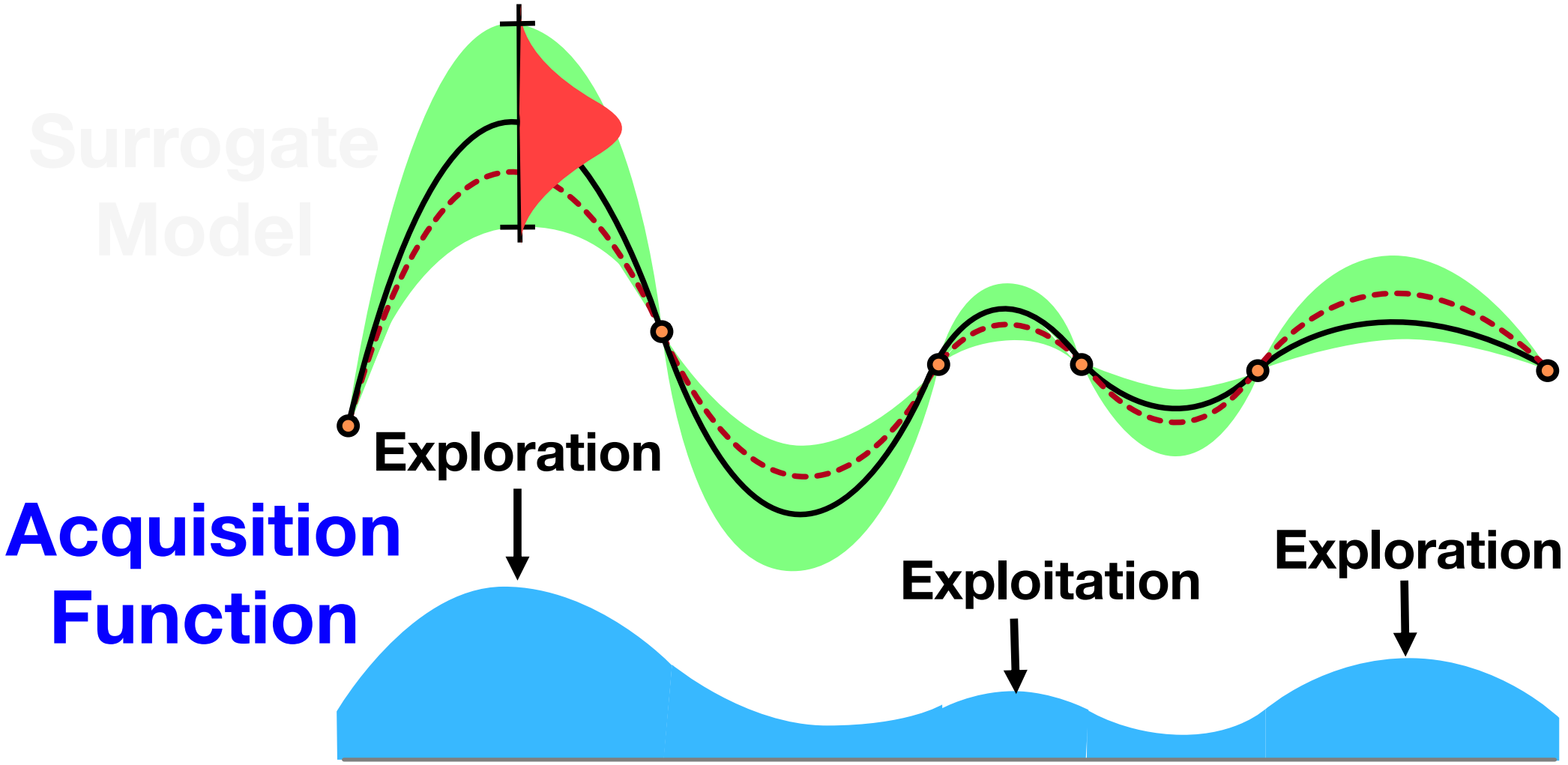
Surrogate Model



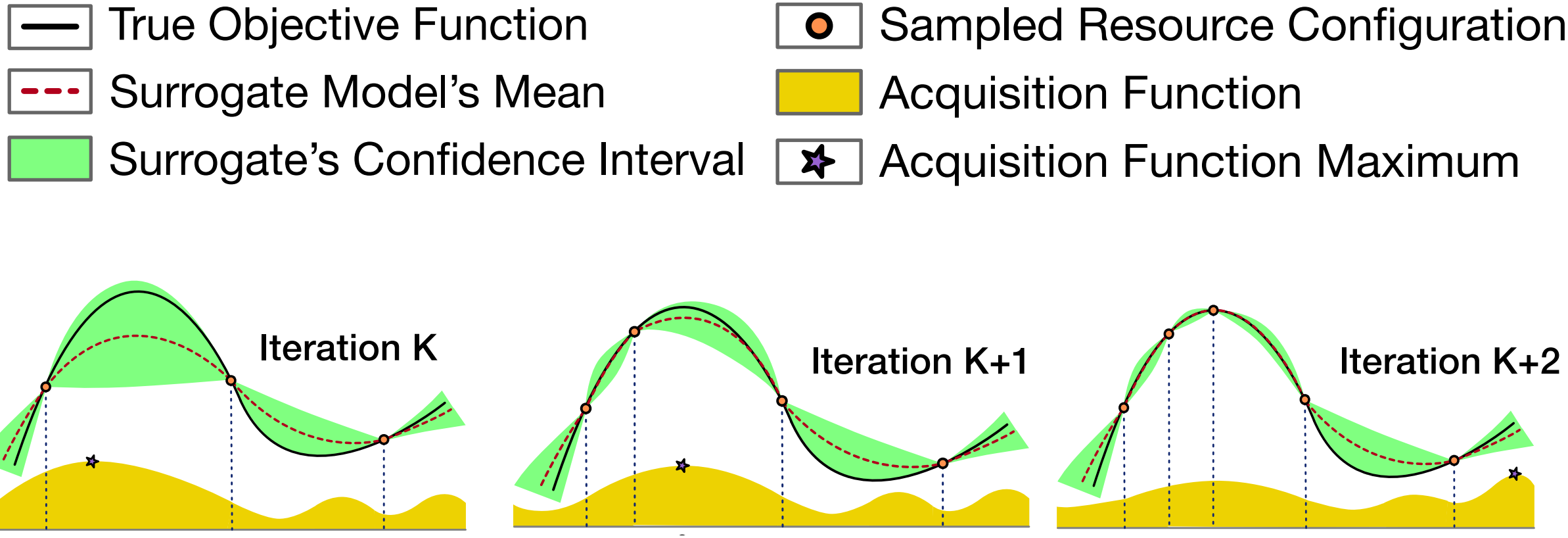
Acquisition Function



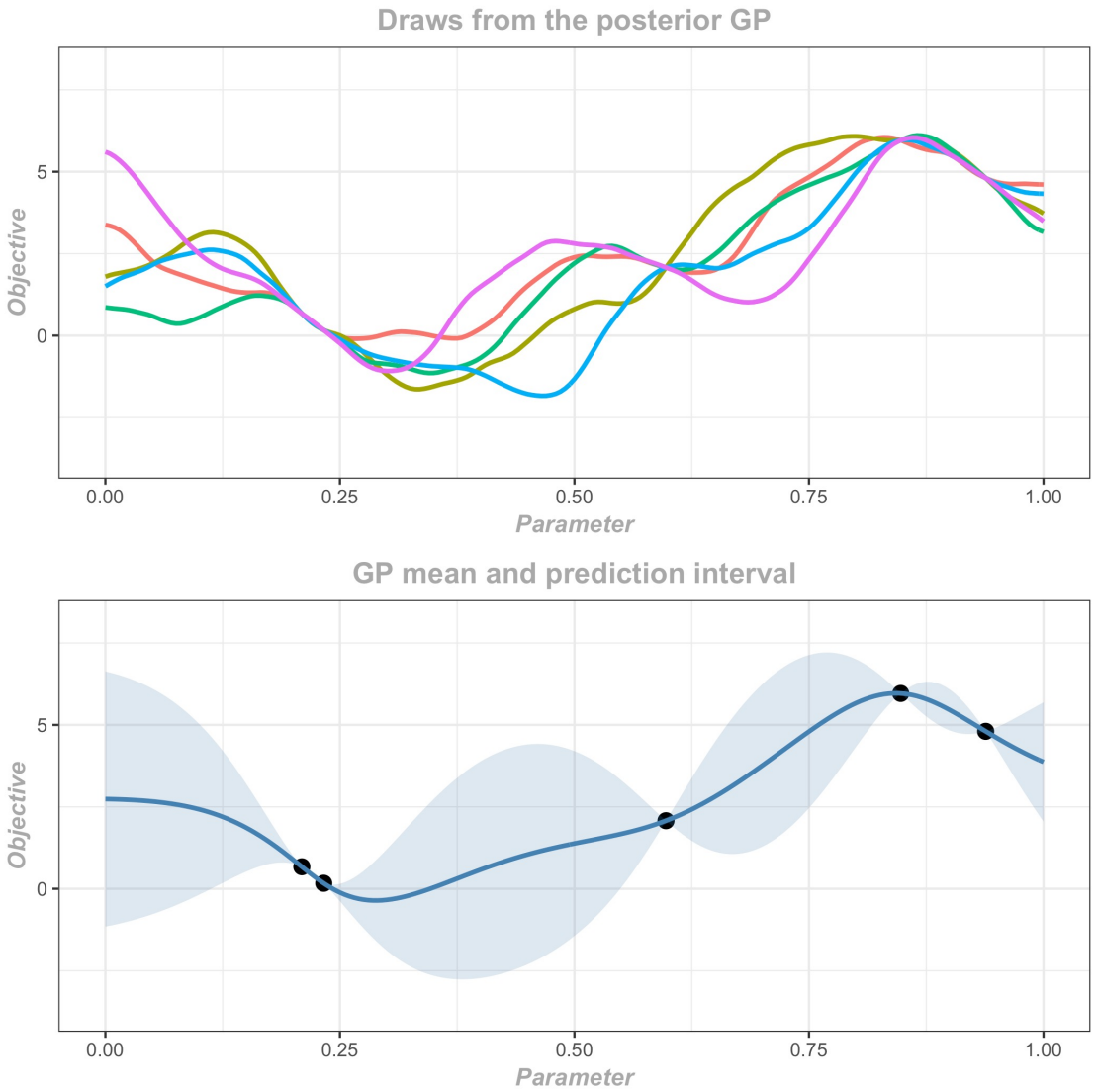
Bayesian Optimization Based Configuration Exploration



Exploration vs Exploitation In Action



BO is non-parametric, the surrogate is a probabilistic Gaussian process



Gaussian processes are defined by a mean and a covariance function

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

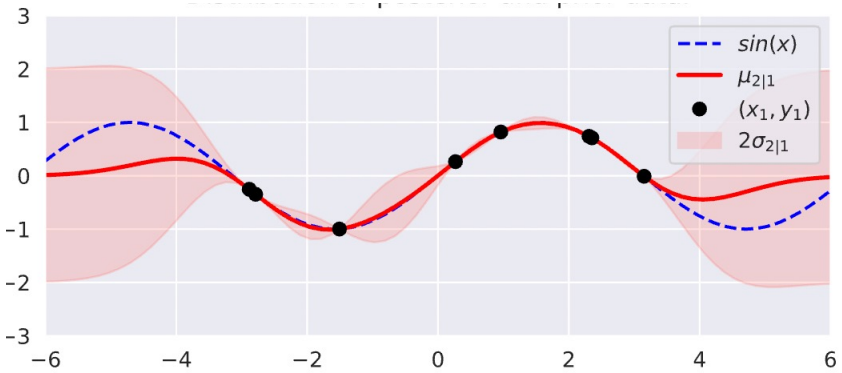
$$k(x_a, x_b) = \exp\left(-\frac{1}{2\sigma^2}\|x_a - x_b\|^2\right)$$

RBF Covariance

Covariance Function

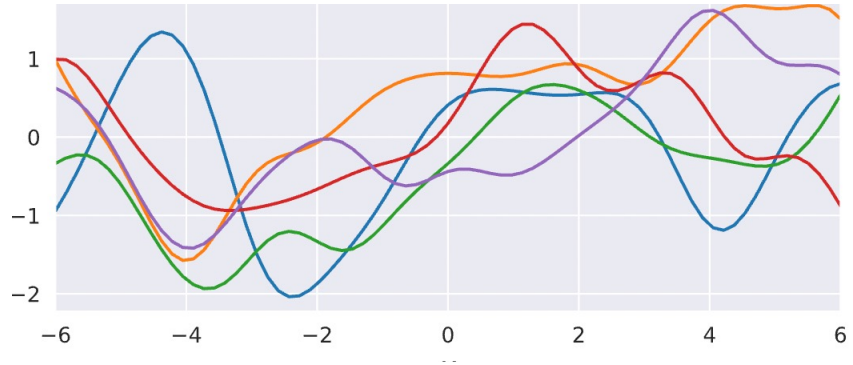
The covariance function determines the distribution between two unsampled configurations

The distribution of the covariance function should be similar to the search space



True Objective

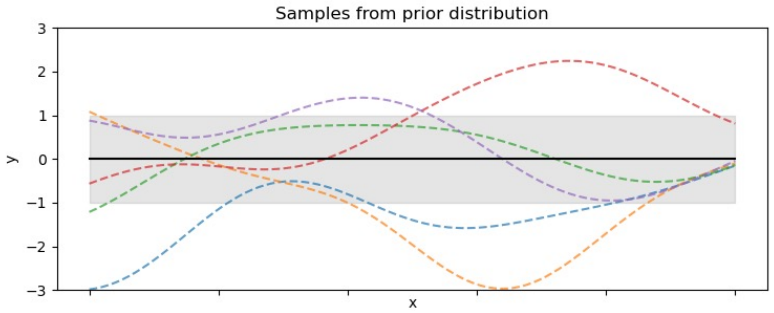
Good covariance



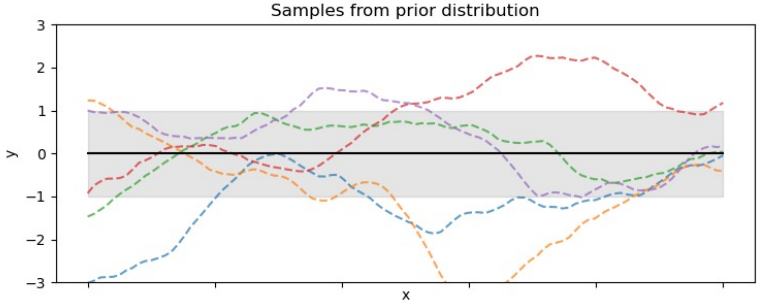
Bad Covariance

The different types of covariance functions

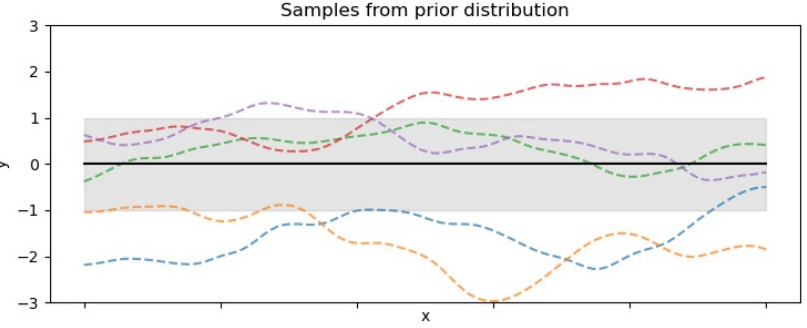
Radial Basis Function kernel



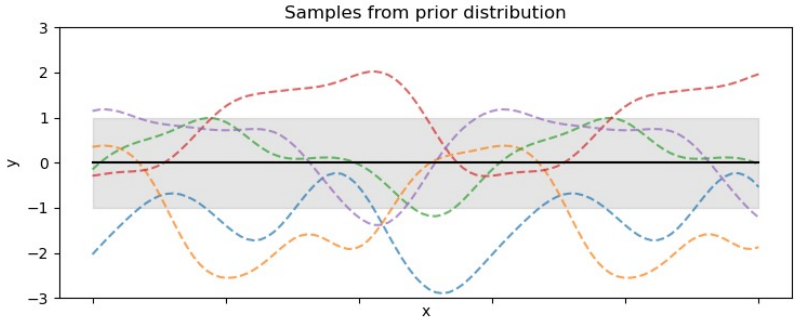
Mattern kernel



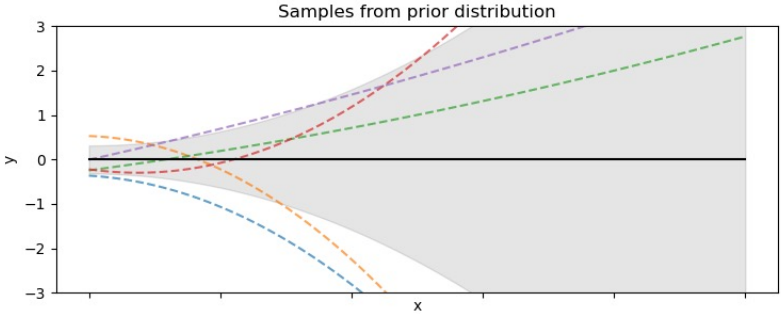
Rational Quadratic kernel



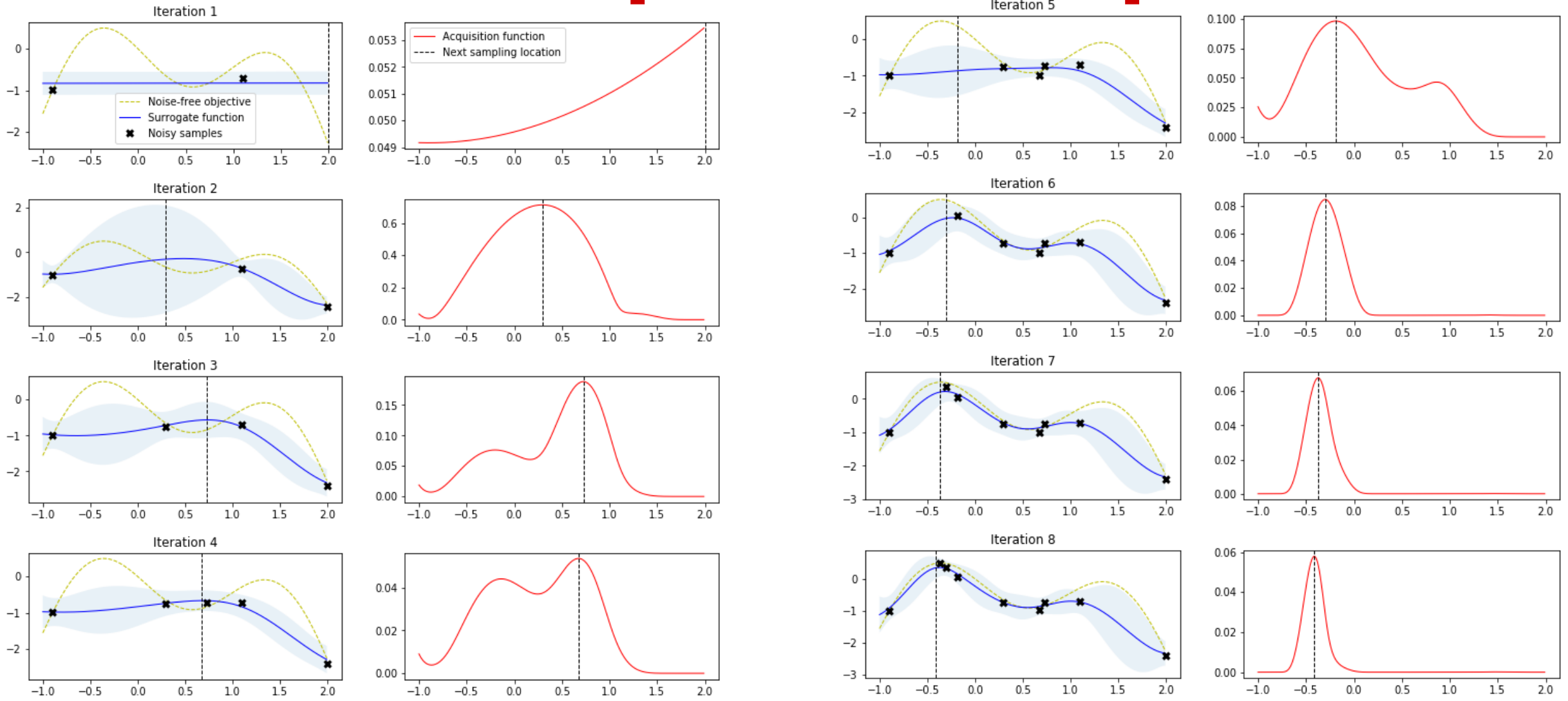
Periodic kernel



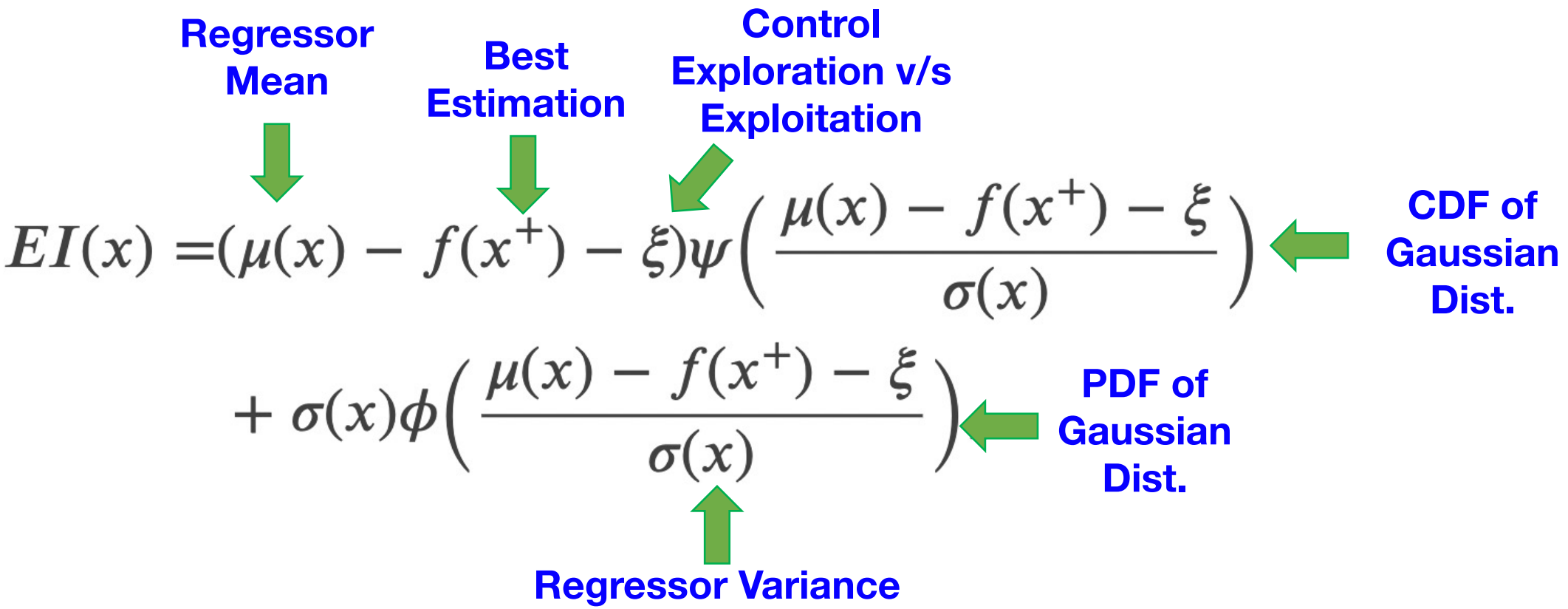
Dot product kernel



The acquisition function of a BO drives the optimization process



Acquisition functions use predicted mean and variance of Gaussian process



Acquisition function balances between exploration v/s exploitation

The Gaussian surrogate and the acquisition function together defines the kernel of BO

$$PI(x) = \psi\left(\frac{\mu(x) - f(x^+) - \xi}{\sigma(x)}\right)$$

Expected improvement, probability of improvement, upper confidence bound, and lower confidence bound are mostly used acquisition functions

The pseudocode of a BO

Bayesian Optimization Algorithm

Assuming goal is to maximize unknown function f(x) on data D:

for n loops do

-> select new x_{n+1} by optimization of α which is an acquisition function $x_{n+1} = \max \alpha(x; D_n)$

-> get new observation y_{n+1} from objective function

-> augment data $D_{n+1} = \{D_n, (x_{n+1}, y_{n+1})\}$

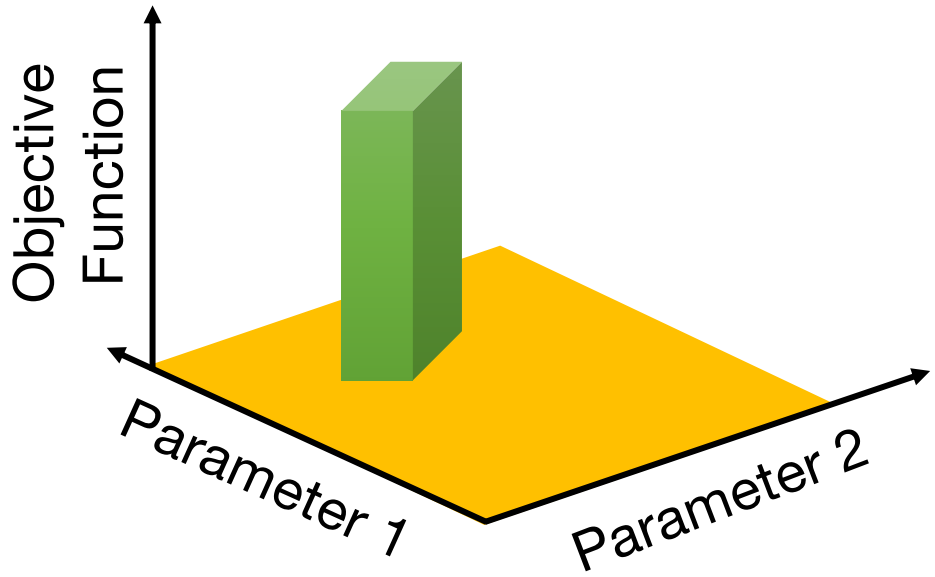
-> update model

end for

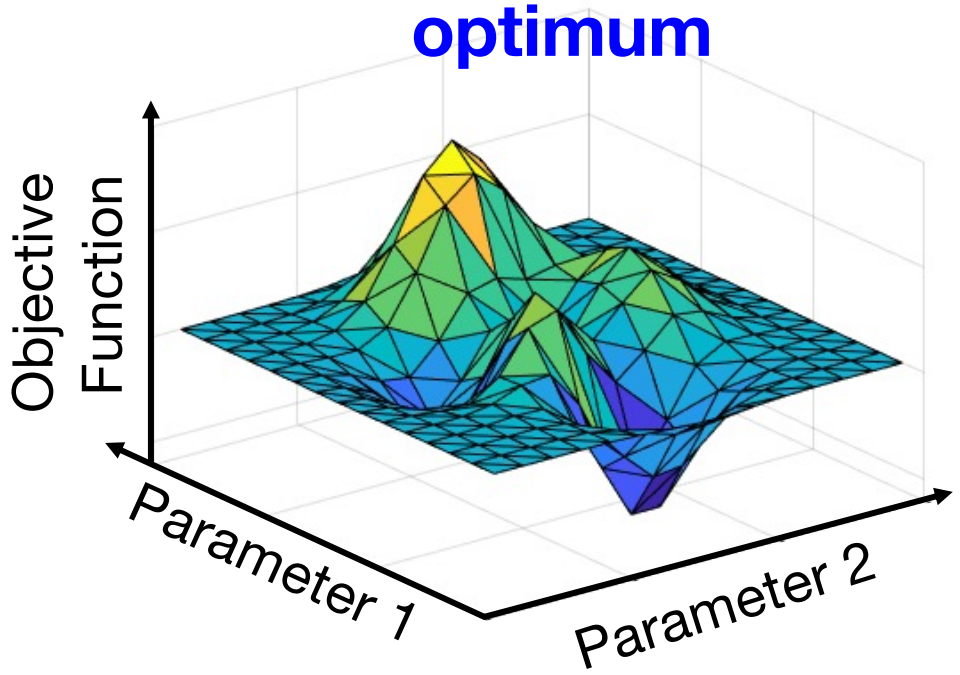
Very critical, Why?

BO's objective function needs to be 'smooth' to guide its optimization

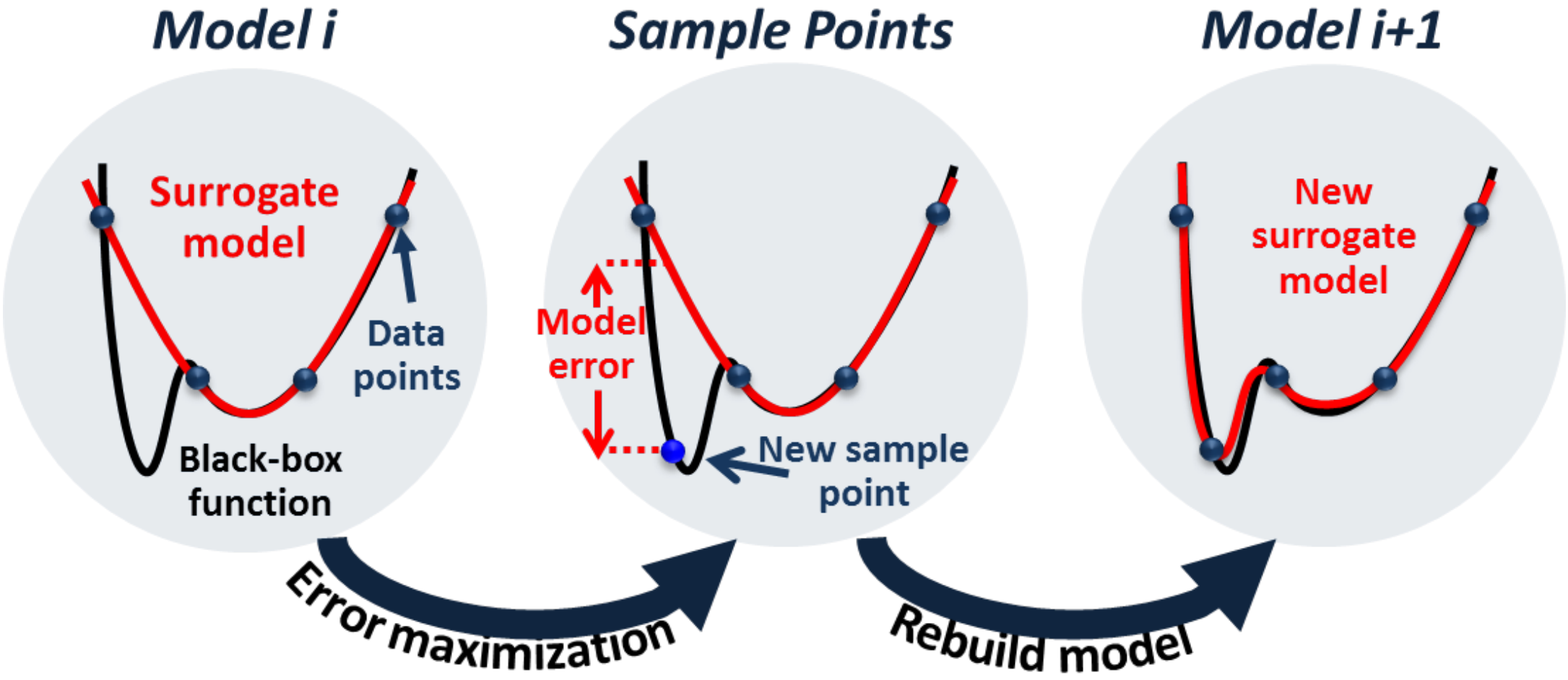
Not good for navigating toward the optimum



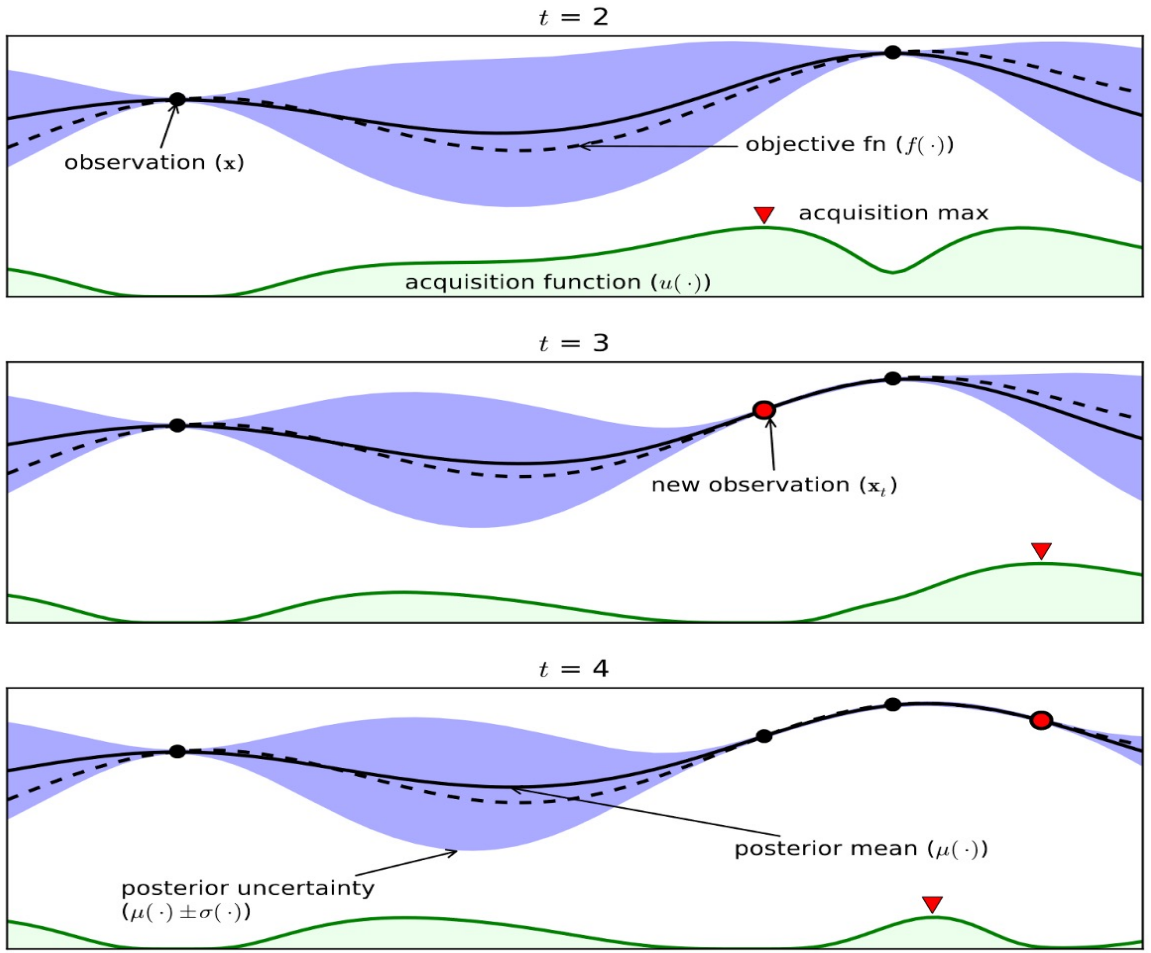
Helps guide toward the optimum



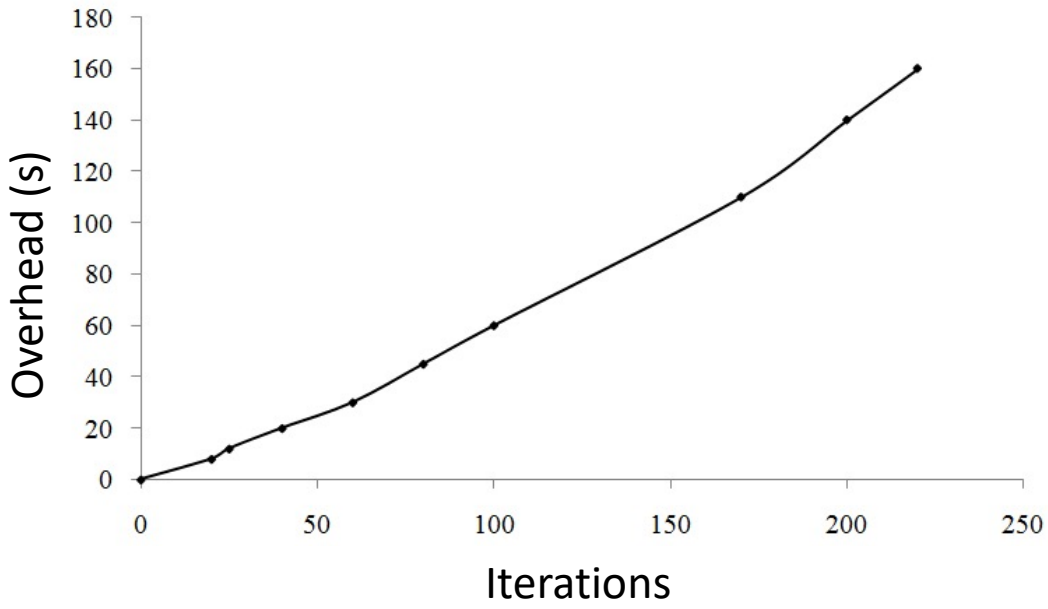
Source of inefficiency: the surrogate model gets updated in every iteration



Source of inefficiency: the acquisition function searches among all configurations in every iteration



The overhead of a BO increases with every iteration

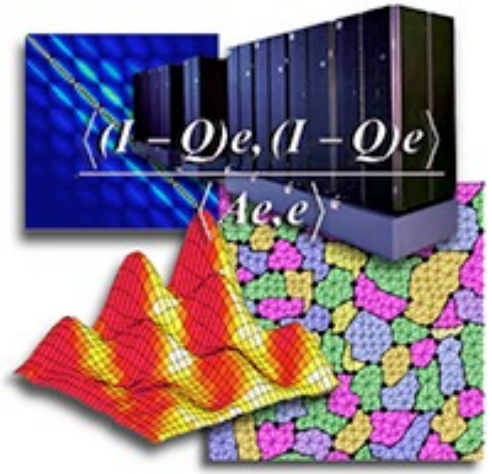


We can speed up a BO by intelligently skipping predictions and evaluations


enough talk
show me examples

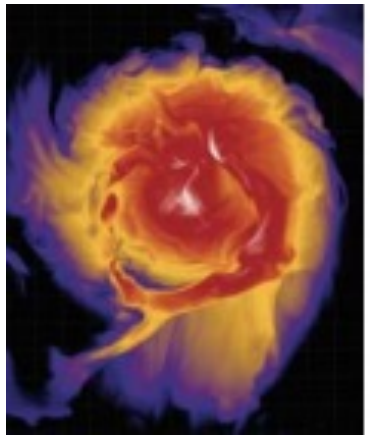
BLISS: Auto-tuning Complex Applications using a Pool of Diverse Lightweight Learning Models (PLDI' 21)

Parallel applications have a large number of tunable software parameters

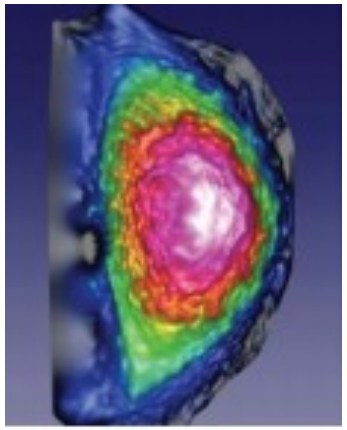


Applied Math

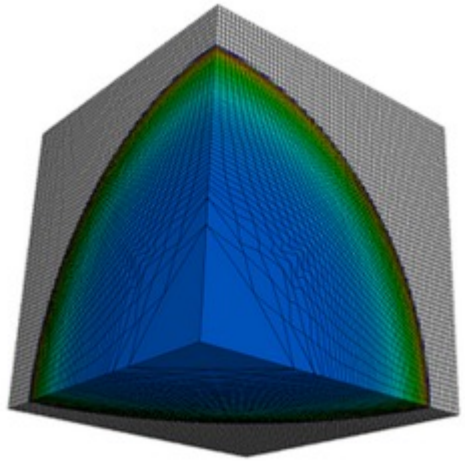
Climate



Nuclear

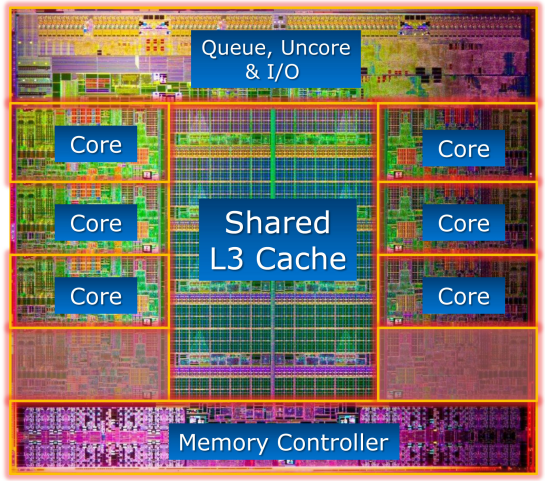


Physics



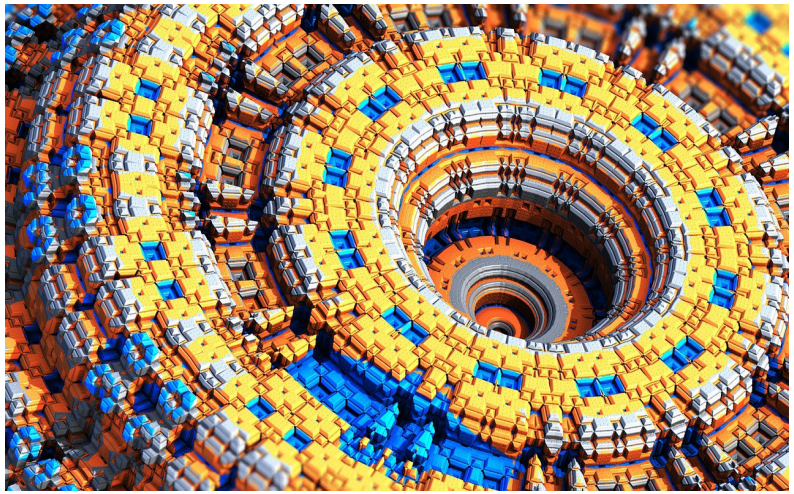
Materials

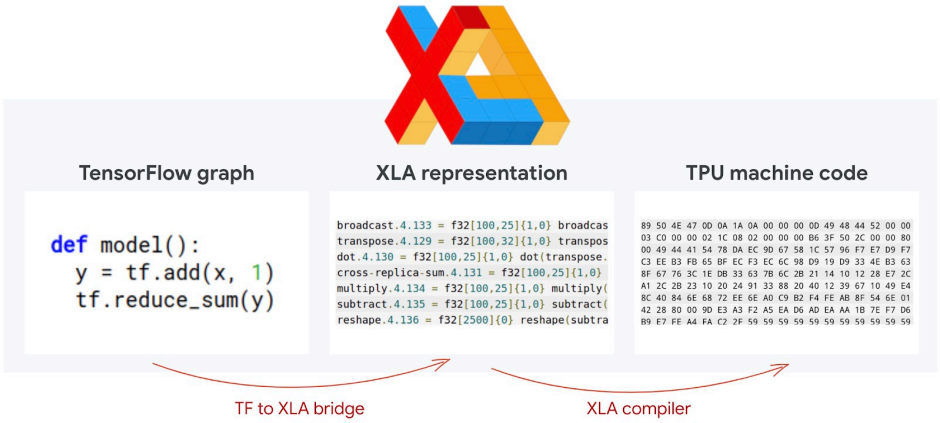
… and are being run on complex hardware with vast number of hardware knobs

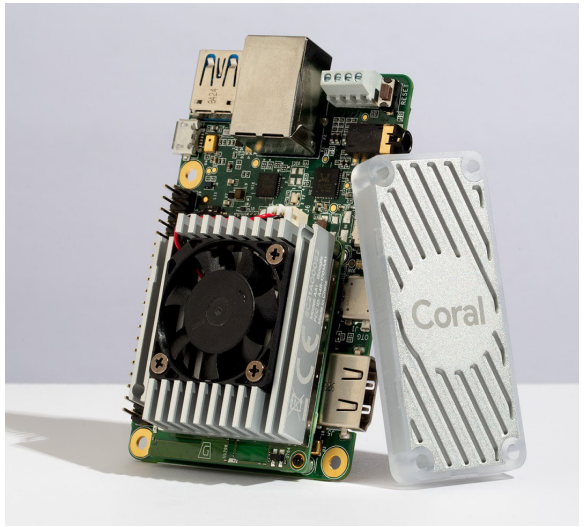


Large parameter search space!

Where can BLISS be applied?





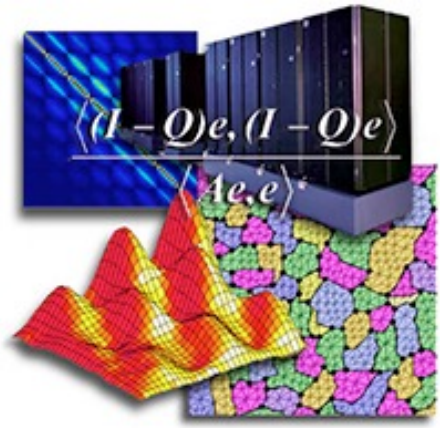


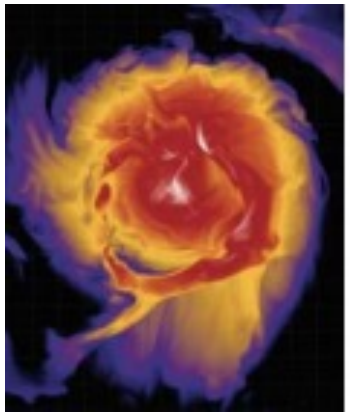
Multi-dimensional Parameter search

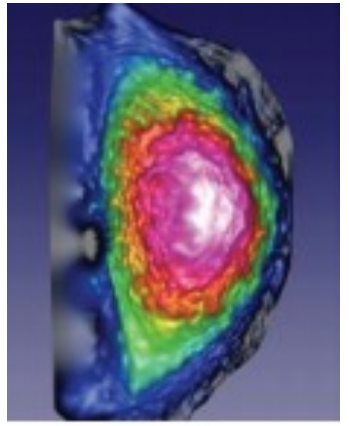
Optimizing TPU parameters for XLA kernel performance modelling

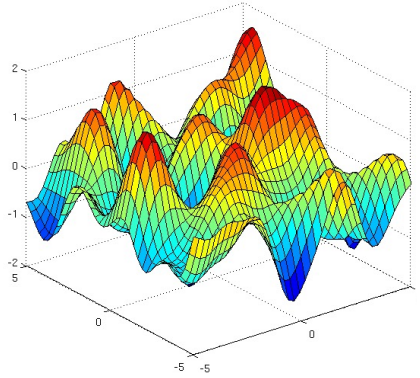
Joint DNN and H/W tuning in Edge TPU

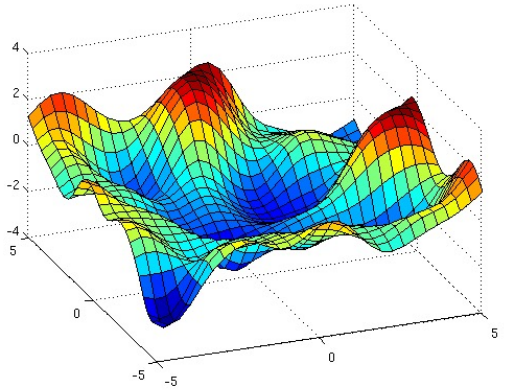
Different applications have significantly different search space terrain and need different underlying surrogate models to capture that

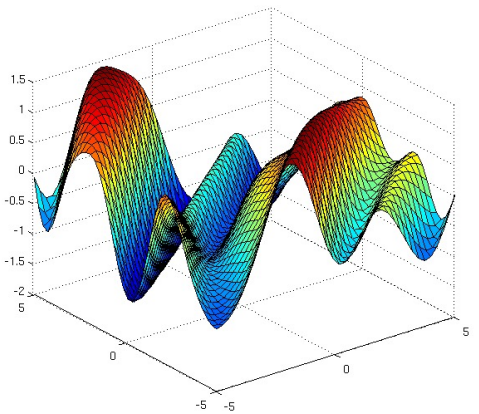




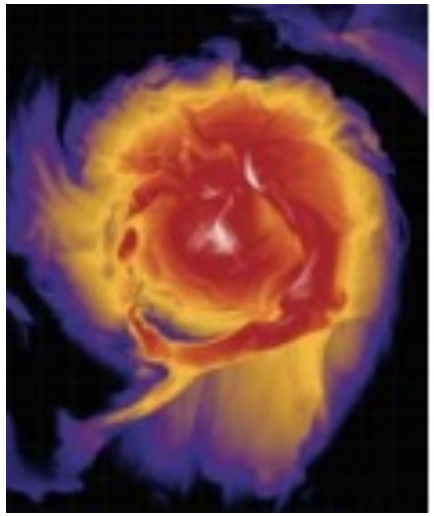


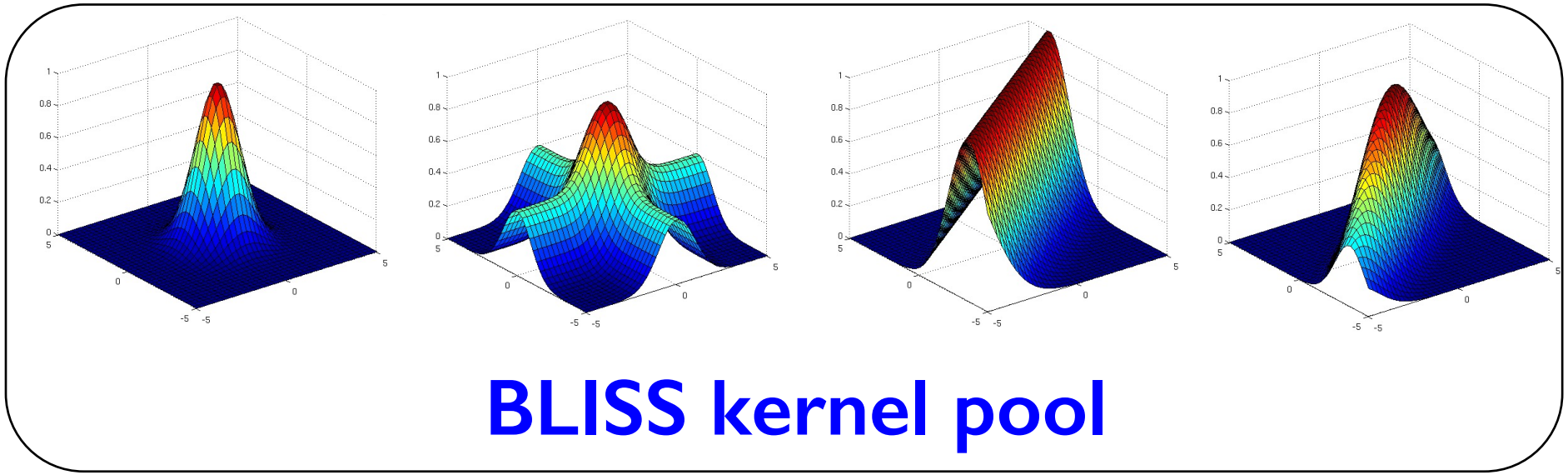






BLISS employs a pool of lightweight BO kernels (surrogate model + acquisition function)

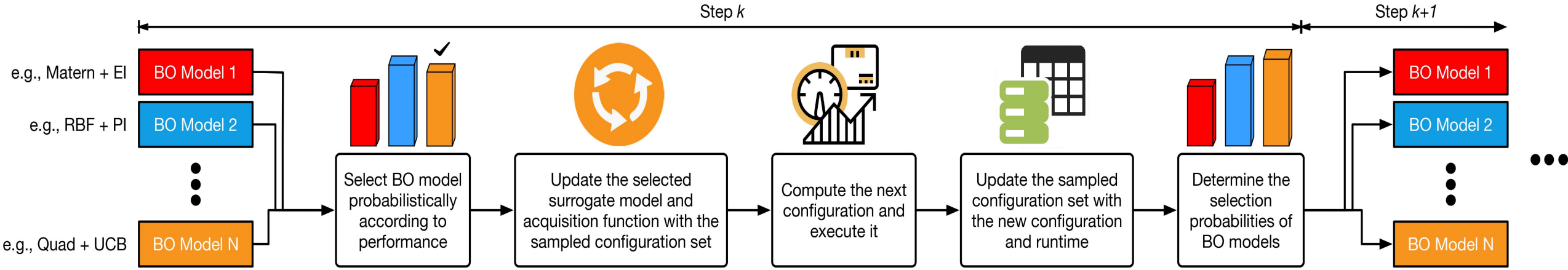




BLISS kernel pool

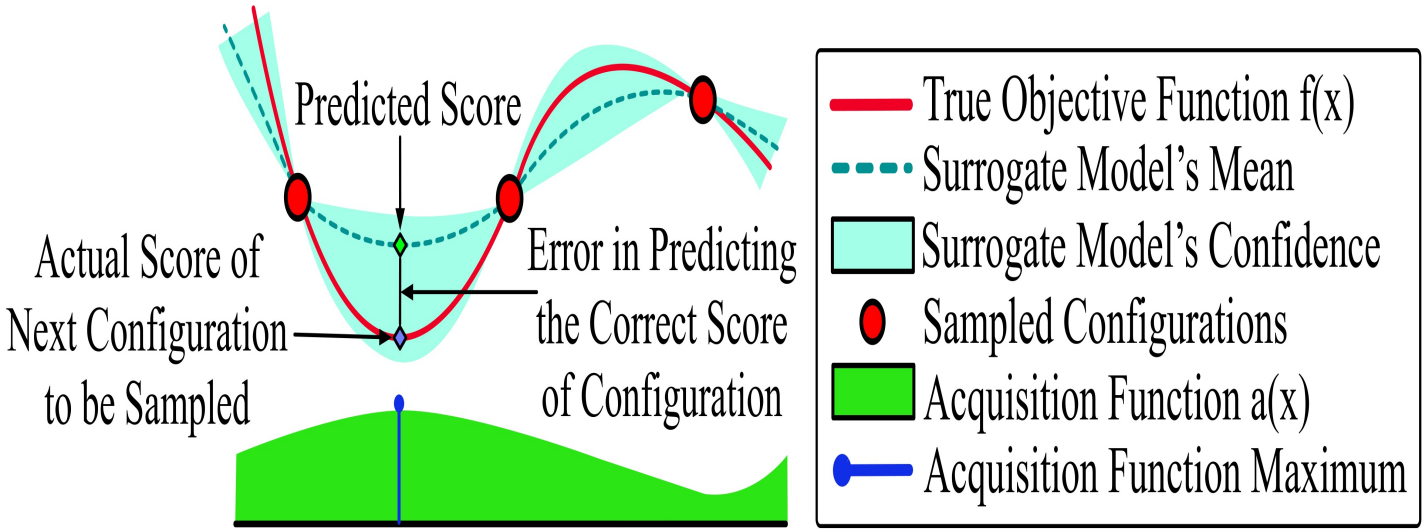
For a given application, the most suitable kernel choice is not known apriori and may change over the search space.
BLISS kernel pool mechanism solves this problem.

BLISS probabilistically and dynamically selects the best kernel over time



Kernels learn in a collaborative fashion and all kernels remain active for selection throughout the exploration.

BLISS employs maturity-based predictions to skip sample evaluations

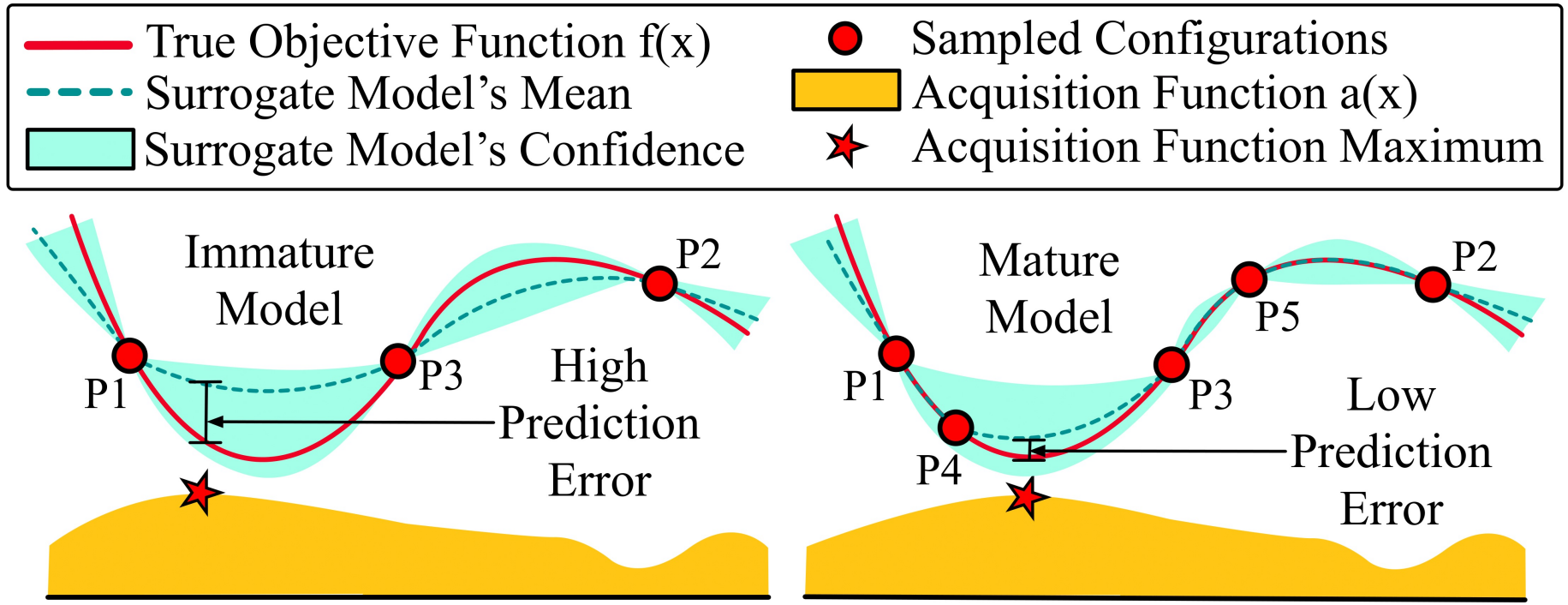


$$l = \begin{cases} \dfrac{\sum_{W_p} \left\lceil \left(1 - \frac{|T_{\text{pred}} - T_{\text{true}}|}{T_{\text{true}}}\right) * l_{max} \right\rceil}{W_p}, & \text{if } |T_{\text{pred}} - T_{\text{true}}| < T_{\text{true}} \\ 0, & \text{otherwise} \end{cases}$$

$$d = \begin{cases} \dfrac{\sum_{W_m} \left\lceil \left(\frac{|T_{\text{pred}} - T_{\text{true}}|}{T_{\text{true}}}\right) * d_{\max} \right\rceil}{W_m}, & \text{if } |T_{\text{pred}} - T_{\text{true}}| < T_{\text{true}} \\ d_{\max}, & \text{otherwise} \end{cases}$$

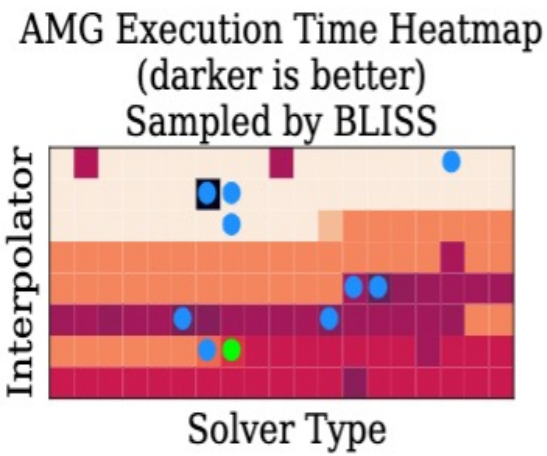
BLISS maintains low overhead and performs fast auto-tuning by intelligently deciding which samples to skip sampling

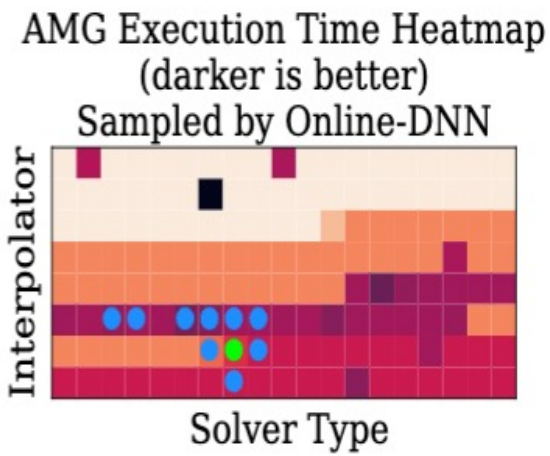
BLISS uses search space pruning for hardware portability

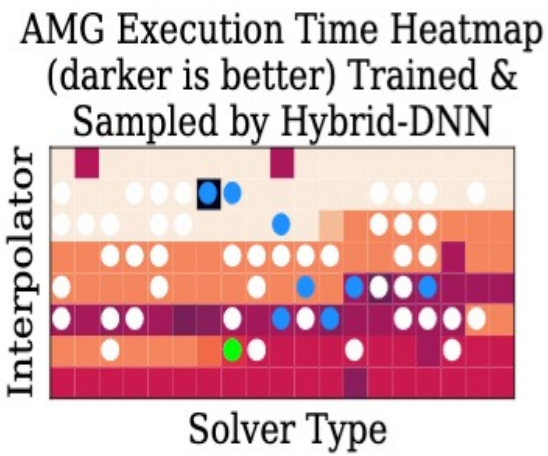


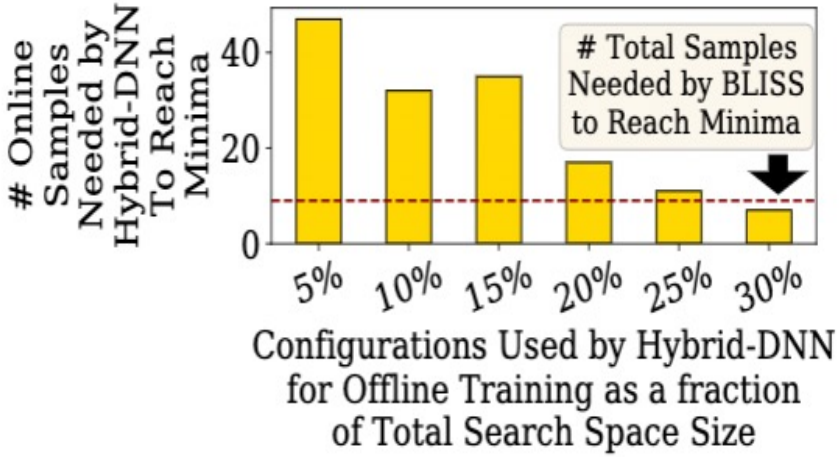
Portability aid helps to speed up the development of performance models across different hardware

DNN based solutions can get stuck in local minima or require larger number of training samples



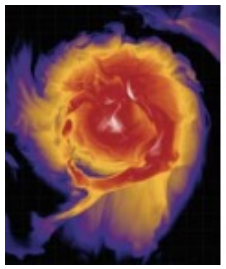


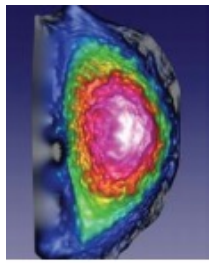




With an increase in the number of dimensions, BLISS achieves more improvement over DNN based solutions

SATORI: Efficient and Fair Resource Partitioning by Sacrificing Short-Term Benefits for Long-Term Gains (ISCA '21)





hadoop

APACHE Spark™

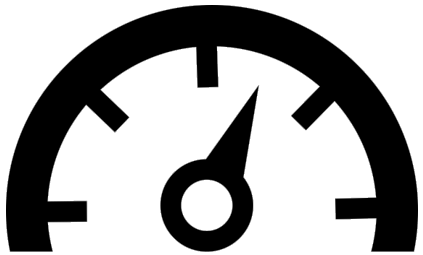
What is the Problem?

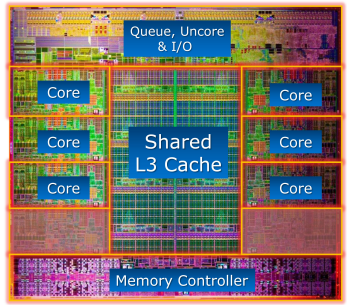
System Goal

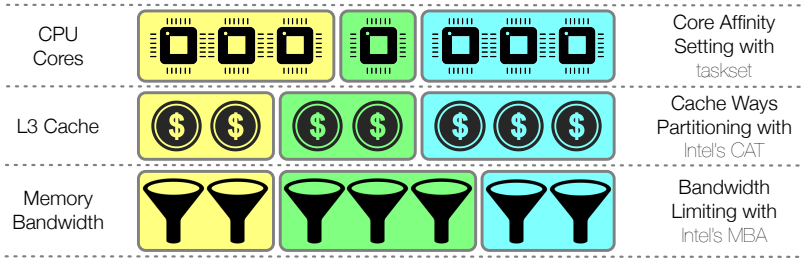
Improvement in system job throughput

User Goal

Fairness in performance degradation

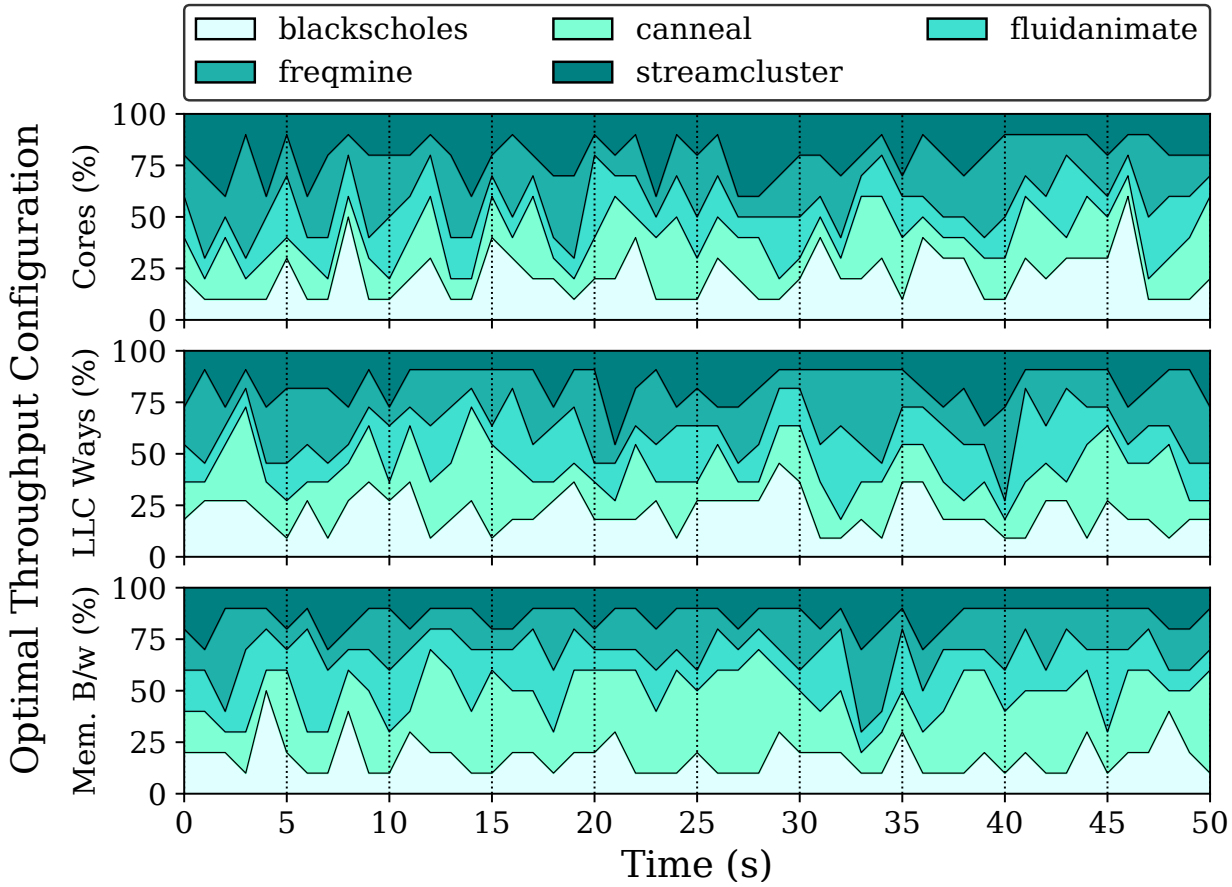






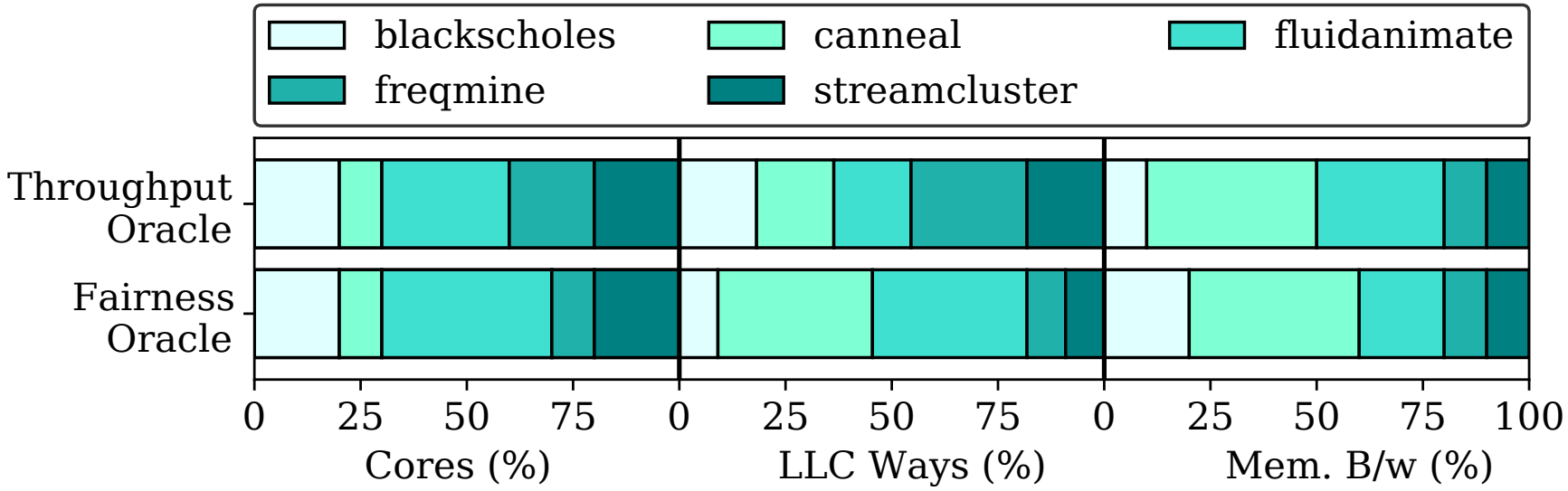
Observation I

The optimal resource partitioning configuration frequently changes over time



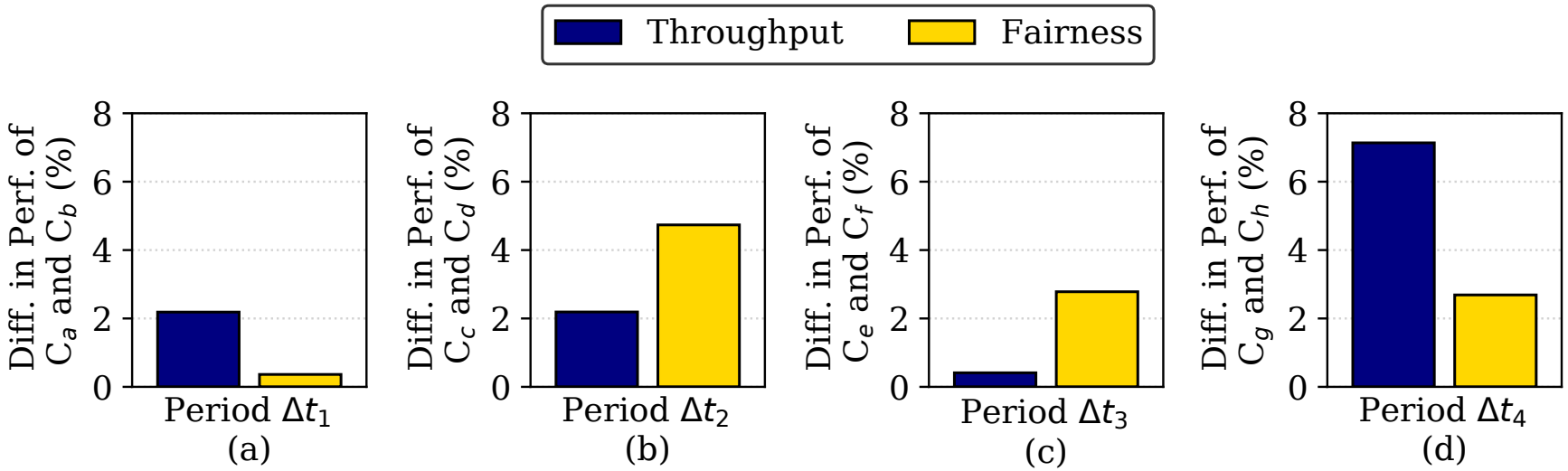
Observation II

Optimal resource partitioning configurations for conflicting goals (e.g., system throughput vs fairness) can be very different from each other

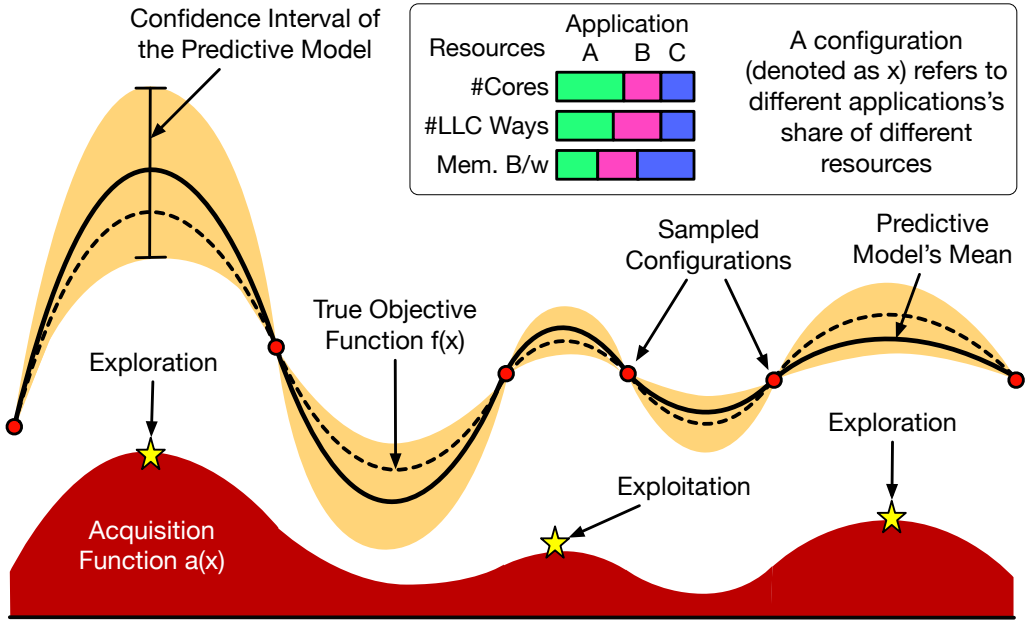


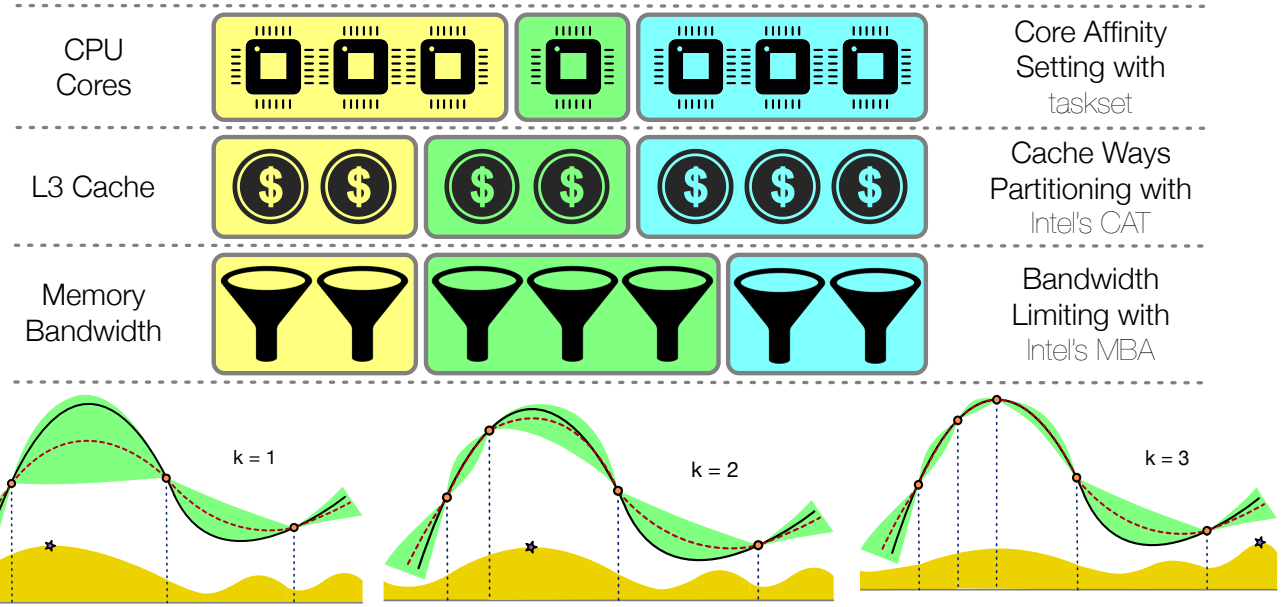
Observation III

There is an opportunity to re-balance conflicting goals over time, (i.e., prioritizing one goal over the other for a short duration) yields higher benefits. Relaxing "equal emphasis" on both goals at all times can be beneficial for both goals.



SATORI's Bayesian Optimizer Dynamically Balances Fairness and Throughput





$$f(x) = \sum_{i=1}^{K} W_i \times \text{Goal}_i(x) = W_T \times \text{T}(x) + W_F \times \text{F}(x)$$

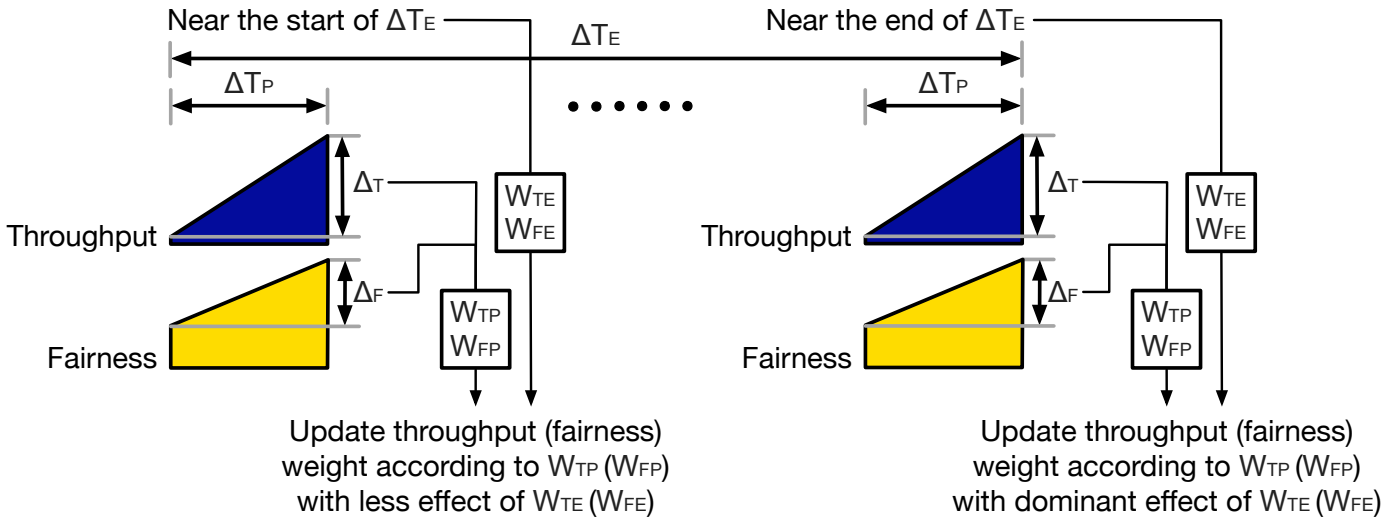
SATORI Creates New Mechanisms for Dynamic Prioritization of Conflicting Goals

$$W_{TE} = \frac{1}{2}t_e - \sum_{i=1}^{t_e} W_{T_i} \quad \& \quad W_{FE} = \frac{1}{2}t_e - \sum_{i=1}^{t_e} W_{F_i}$$

$$W_{TP} = \frac{1}{4} + \frac{1}{2}\frac{\Delta_F}{\Delta_T + \Delta_F} \quad \& \quad W_{FP} = \frac{1}{4} + \frac{1}{2}\frac{\Delta_T}{\Delta_T + \Delta_F}$$

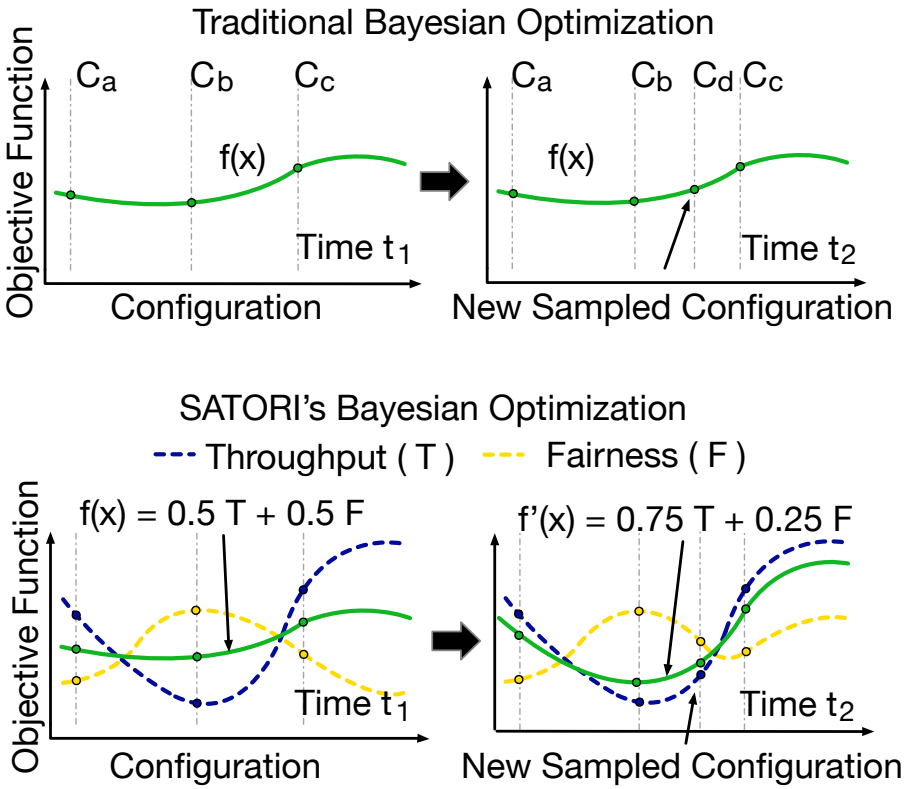
$$W_T = \frac{t_e}{T_E}W_{TE} + \left(1 - \frac{t_e}{T_E}\right)W_{TP}$$

$$W_F = \frac{t_e}{T_E}W_{FE} + \left(1 - \frac{t_e}{T_E}\right)W_{FP}$$



Novel concept of Equalization and Prioritization periods for balancing the dynamic prioritization

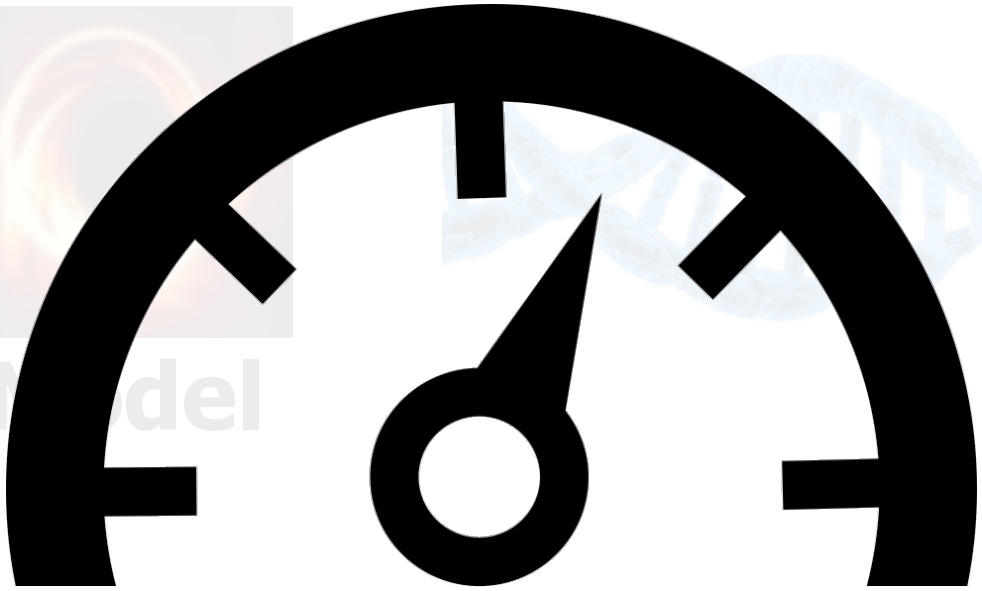
SATORI Creates New Mechanisms For Dynamic Prioritization of Conflicting Goals



SATORI maintains separate throughput and fairness performances and constructs a fresh objective function every iteration based on dynamic weights.

CLITE: Efficient and QoS-Aware Co-location of Multiple Latency-Critical Jobs for Warehouse Scale Computers (HPCA '20)

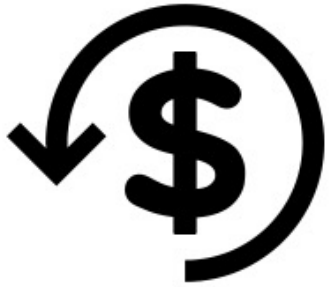
Latency-critical (LC) jobs have strict QoS requirements!



Throughput-oriented / background (BG) jobs want highest possible performance!

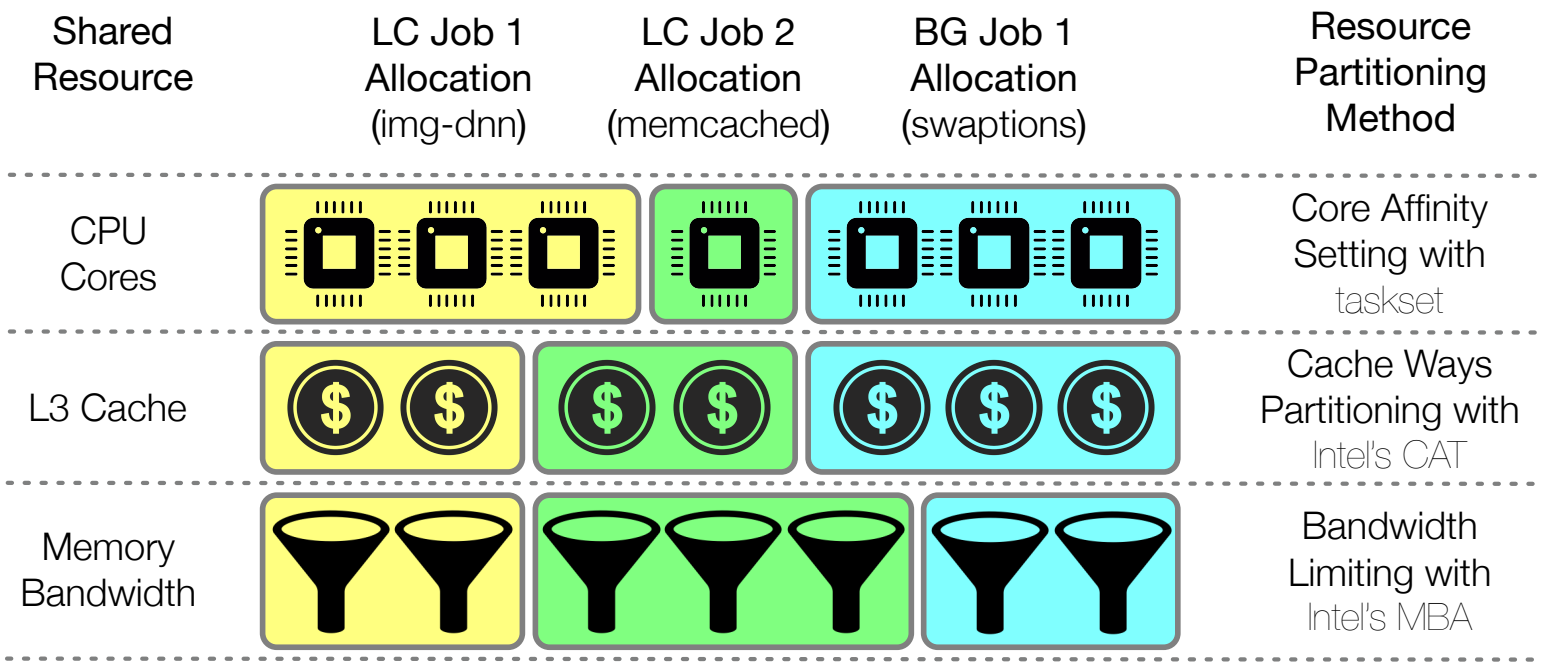
Co-locating Latency-critical and Throughput-oriented Jobs

Improved resource utilization and reduced hardware provisioning cost

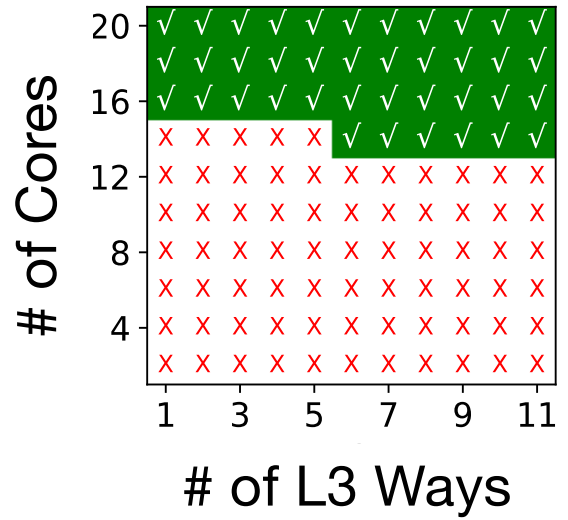


Jeopardizes QoS requirements (monetary loss)

Multi-core resource partitioning enables interference-free co-location



QoS-safe region for Img-dnn

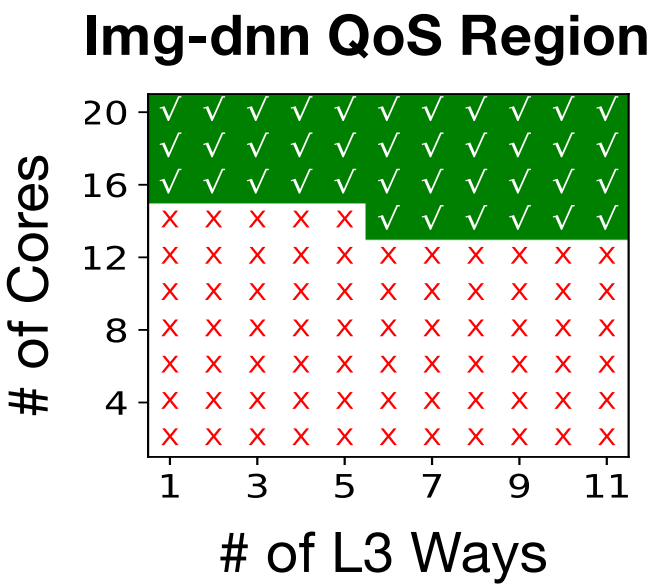


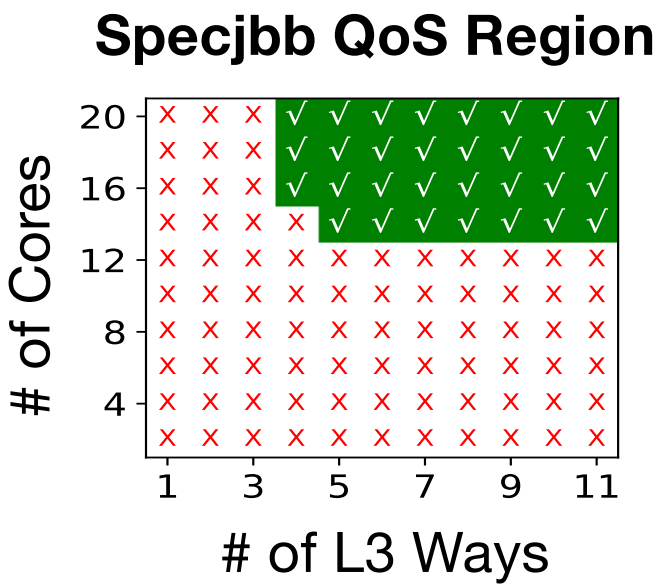
A resource configuration refers to all apps' share for different resources

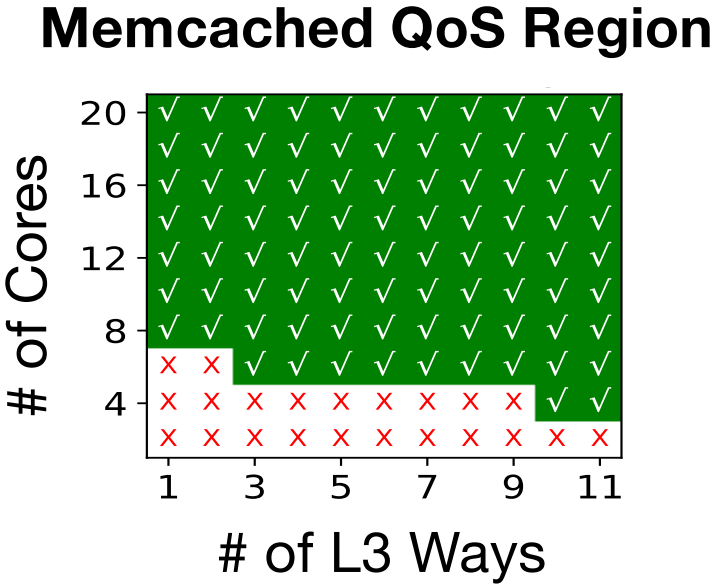
# of cores: <20% LC1, 40% LC2, 20% BG1>

#L3 cache ways: <40% LC1, 35% LC2, 35% BG1>

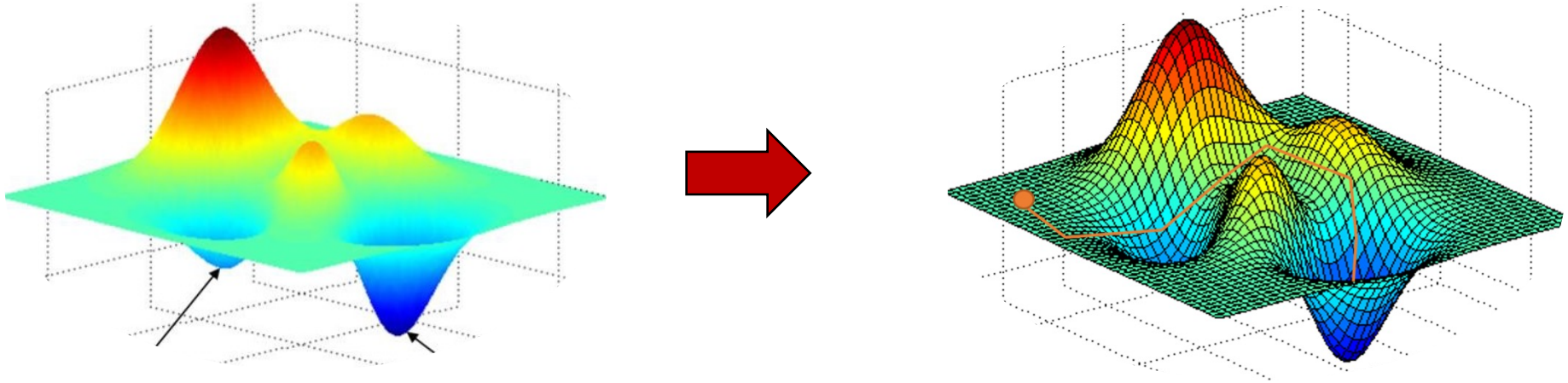
Different workloads have significantly different QoS-safe regions







Non-smooth CLITE Search Space



CLITE devises a new score assignment method for objective function evaluation

Score function when QoS is not met

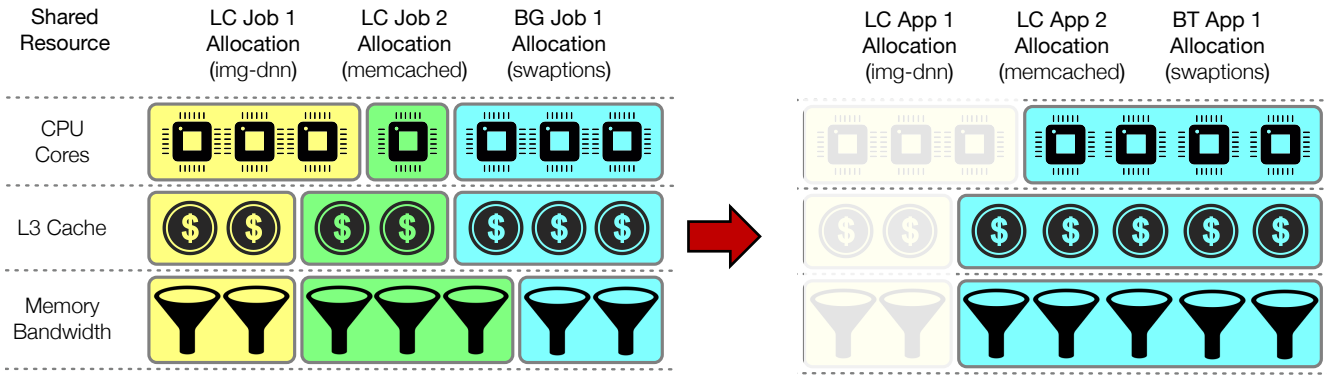
$$0.5 \times \sqrt[N_{LC}]{\prod_{n=1}^{N_{LC}} \min(1.0, \frac{\text{QoS Target}_n}{\text{Current Latency}_n})}$$

First meet the QoS then optimize BG jobs

$$0.5 + 0.5 \times \sqrt[N_{BG}]{\prod_{n=1}^{N_{BG}} \frac{\text{CoLocation Perf}_n}{\text{Isolation Perf}_n}}$$

Mitigating the Curse of High Dimensionality of the Search Space

"Dropout-copy" by fixing the allocation of the best performing job so far

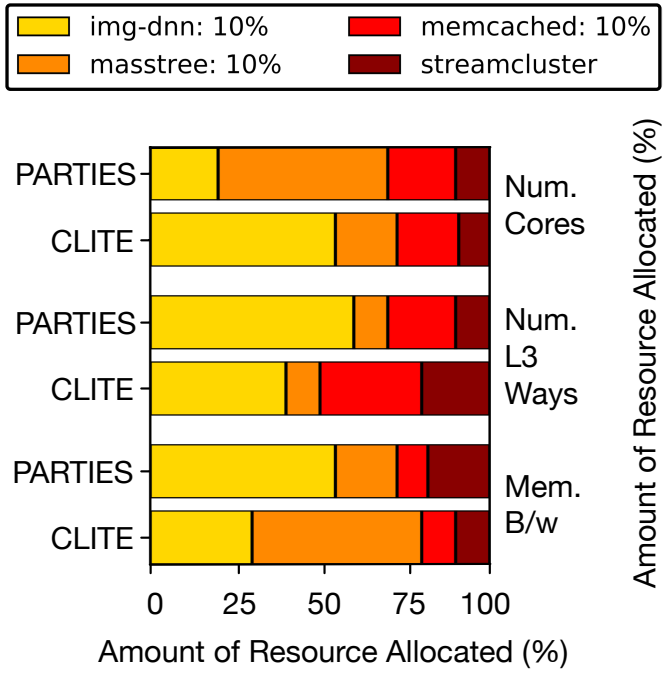


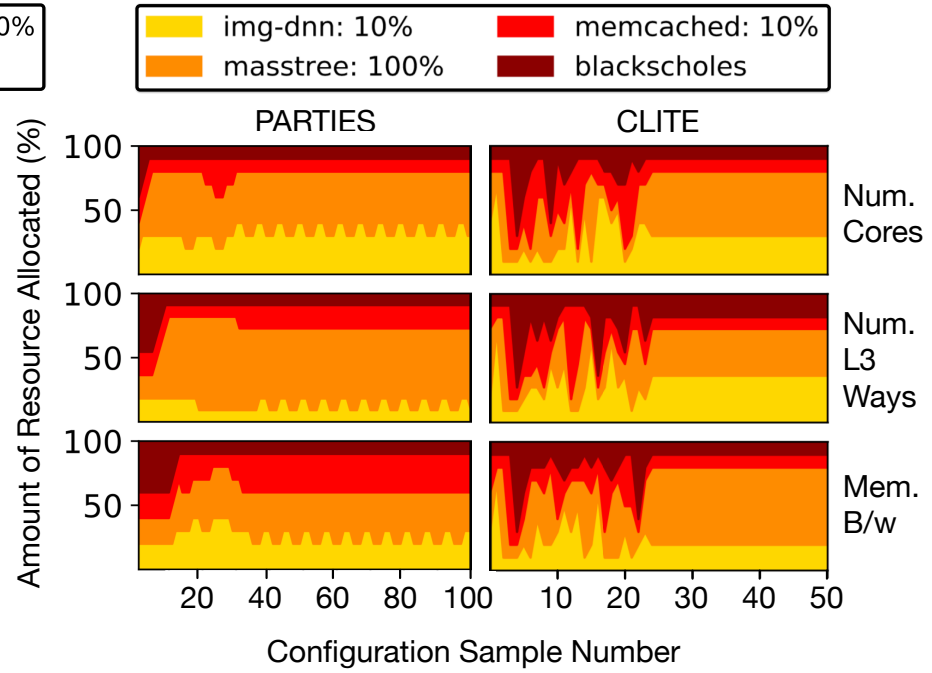
Quickly pruning "likely-to-be-suboptimal" samples

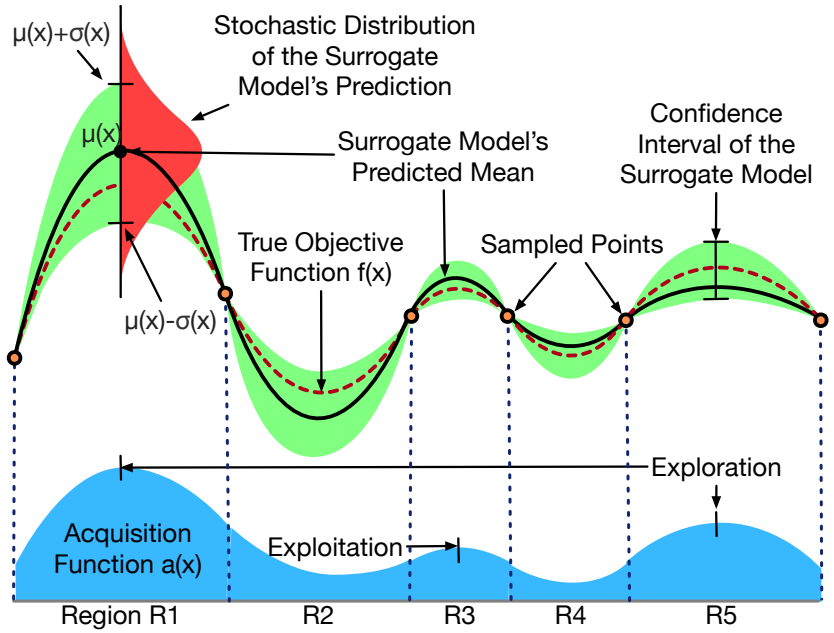
J	Set of all jobs
R	Set of all resources
$x(j, r)$	Allocation of resource r to job j
$a(x(q, r))$	Acquisition function at a given configuration
N_{jobs}	Number of jobs
$N_{units}(r)$	Number of units of resource r

$$\text{maximize } a(x(j,r))$$
$$\text{s.t. } \forall j,r \in J,R \rightarrow 1 \leq x(j,r) \leq N_{\text{units}}(r) - N_{\text{jobs}} + 1$$
$$\& \; \forall r \in R \rightarrow \sum_{j \in J} x(j,r) = N_{\text{units}}(r)$$

Why does BO work?







BO-based exploration can search a much wider range of configurations quickly and reach a close-to-optimal resource allocation configuration.

Bayesian Optimization can be beneficial for your optimization problem if there exists a multi-dimensional search space, following some sort of distribution

BLISS

SATORI

CLITE