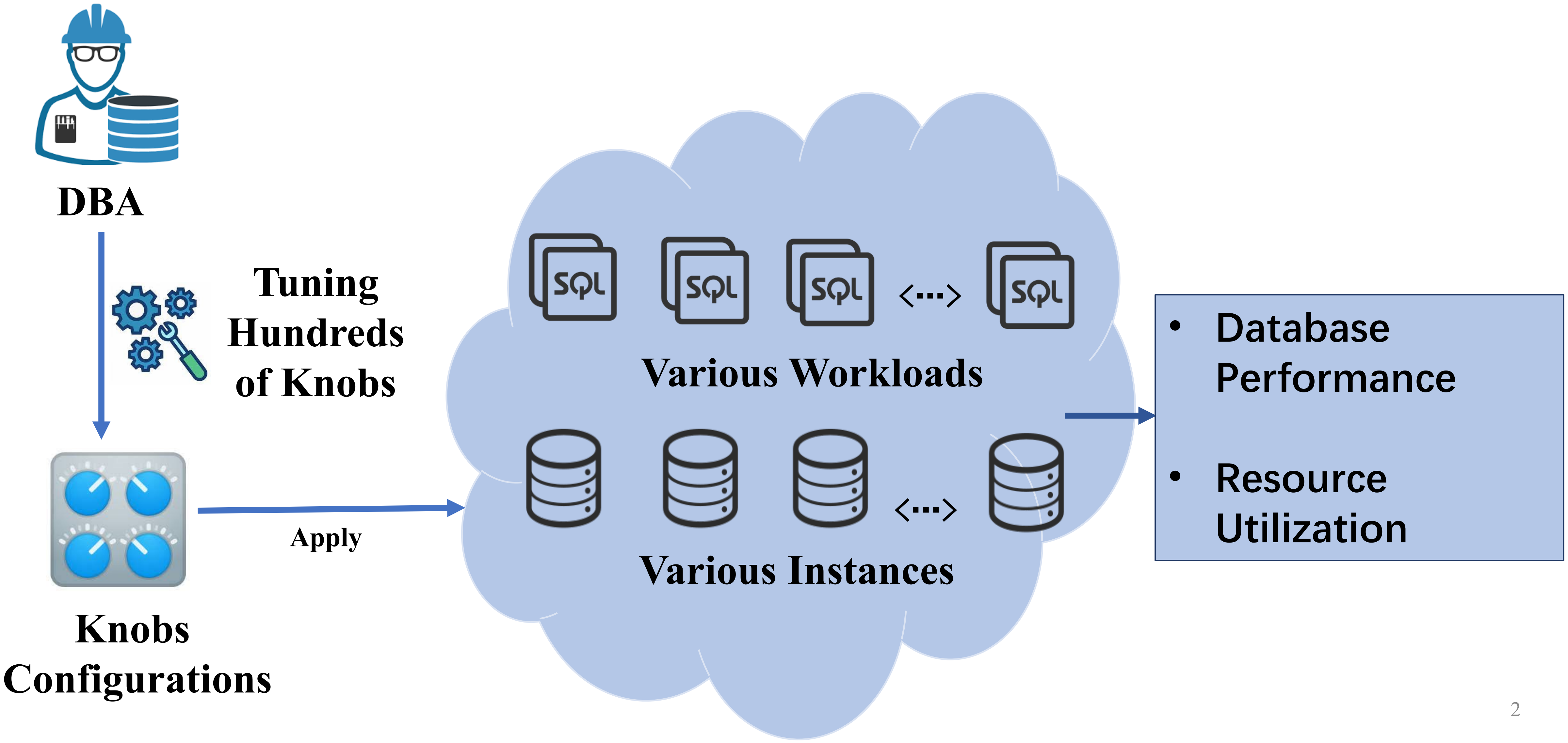
Xi'an, Shaanxi, China
SIGMOD/PODS 2021

ResTune
Resource Oriented Tuning Boosted by Meta-Learning for Cloud Databases

Xinyi Zhang*

Hong Wu*, Zhuo Chang, Shuowei Jin,
Jian Tan, Feifei Li, Tieying Zhang, Bin Cui

DBMS Tuning in Cloud



Limitation of Existing Methods

#1
Not Optimizing Resource Usage

All Methods

#2
Time Consuming

Search Based

#3
High Training overhead

Reinforcement Learning Based

#4
Weak Adaptability

Bayesian Optimization Based

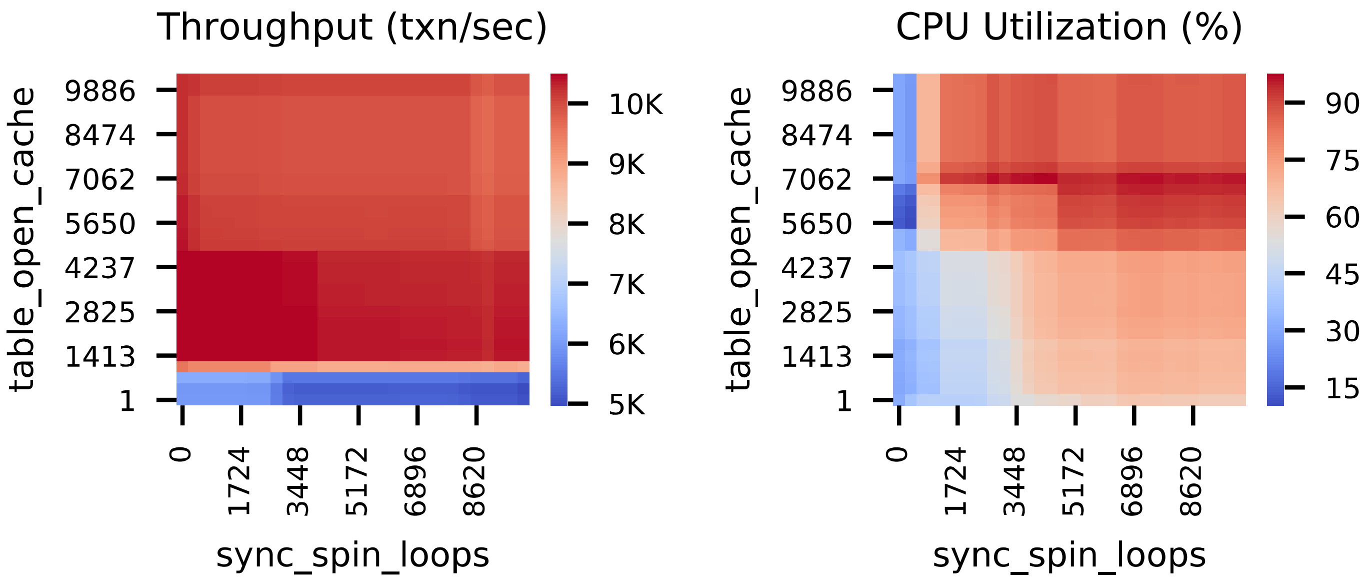
Our Goal

- **Goal 1:** To optimize the performance and the resource utilization simultaneously.

- **Goal 2**: To boost the tuning process with different past tuning tasks from different instance types and different workloads

Observations

The throughput and CPU usage on a real workload with 2 controlling knobs:



Observation 1: Throughput is not the bottleneck in most cases.

Observation 2: A wide range of configurations has different CPU usages but the same throughput.

Resource Oriented Tuning Problem

- We formalize the resource-oriented tuning problem as an optimization problem with SLA constraints
  - Consider a database with a continuous configuration space Θ:

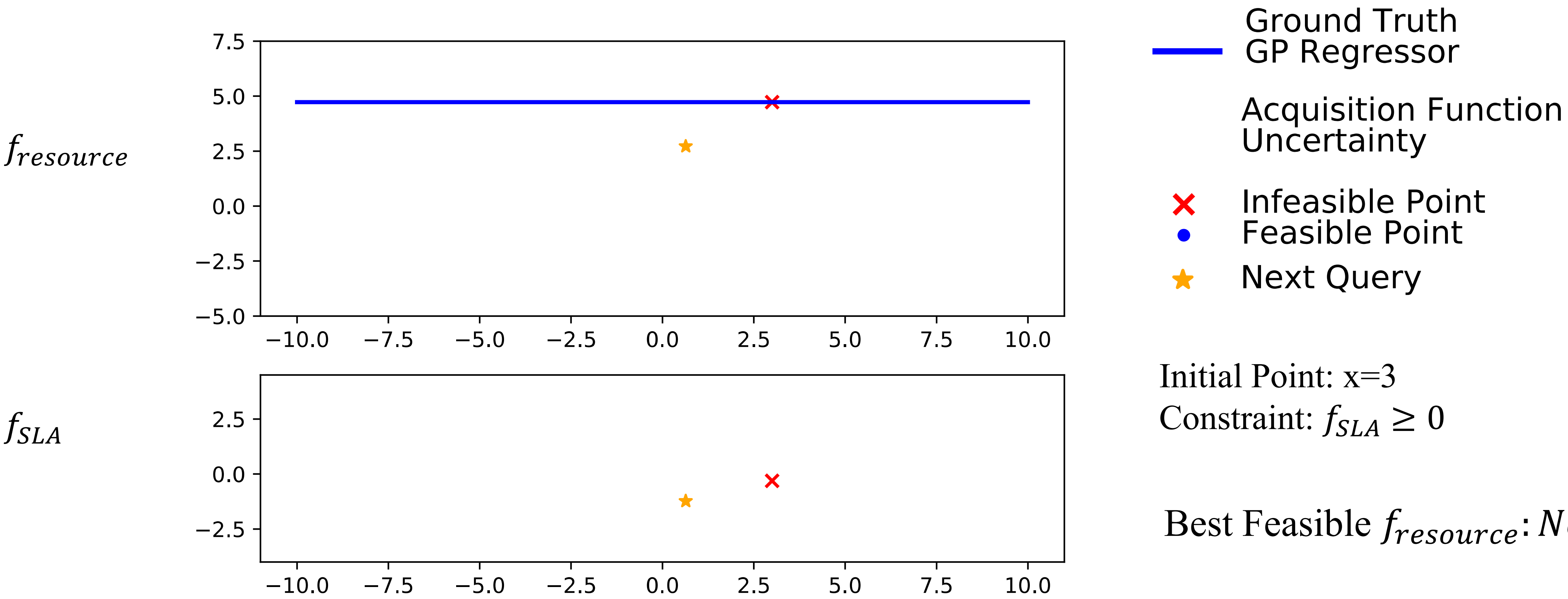
$$\arg\min_{\theta} f_{resource}(\theta)$$

$$\text{s.t. } f_{Throughput} \geq SLA_{Throughput}$$

$$f_{Latency} \leq SLA_{Latency}$$

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.



Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

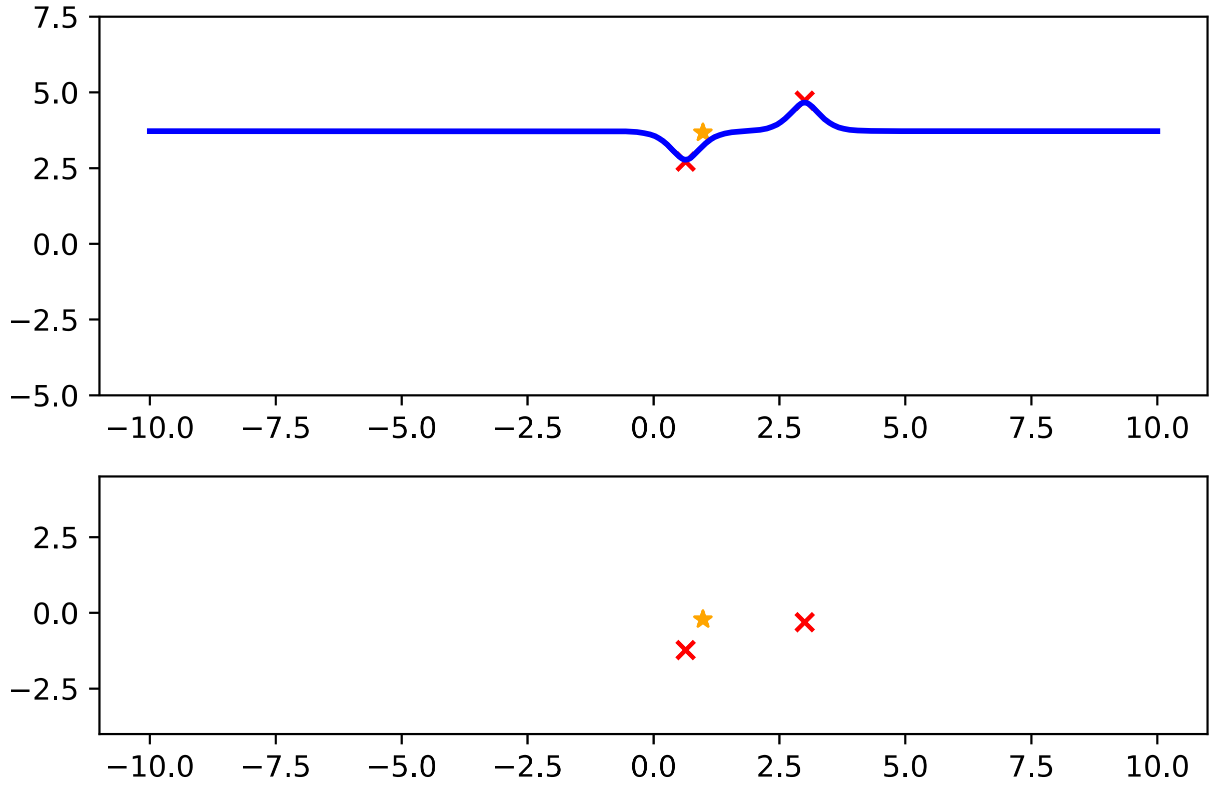
Iteration 1

Solving Constrained Optimization

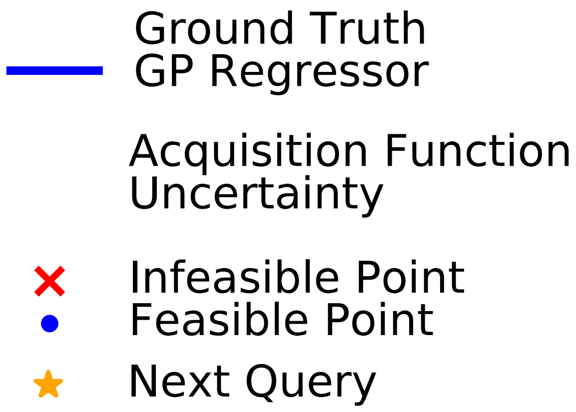
- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}



Iteration 2



Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

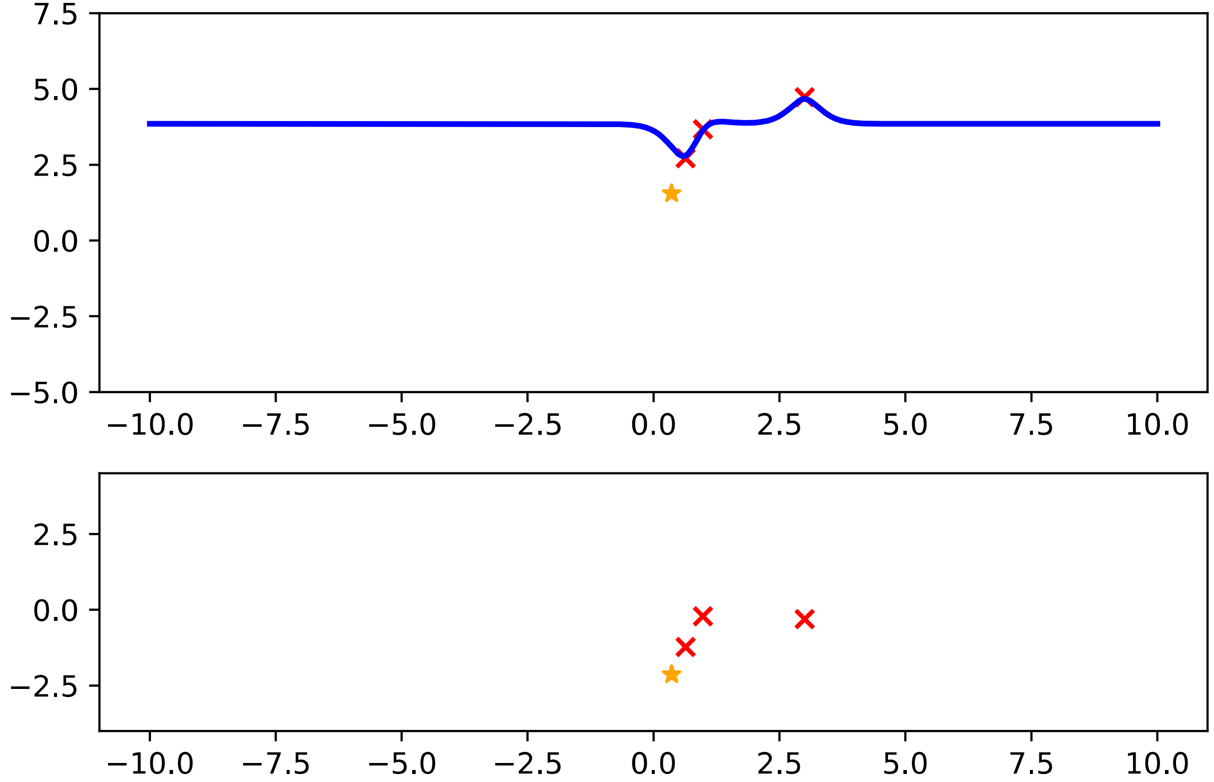
Best Feasible $f_{resource}$: $None$

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}



Iteration 3

Ground Truth
GP Regressor

Acquisition Function
Uncertainty

Infeasible Point
Feasible Point

Next Query

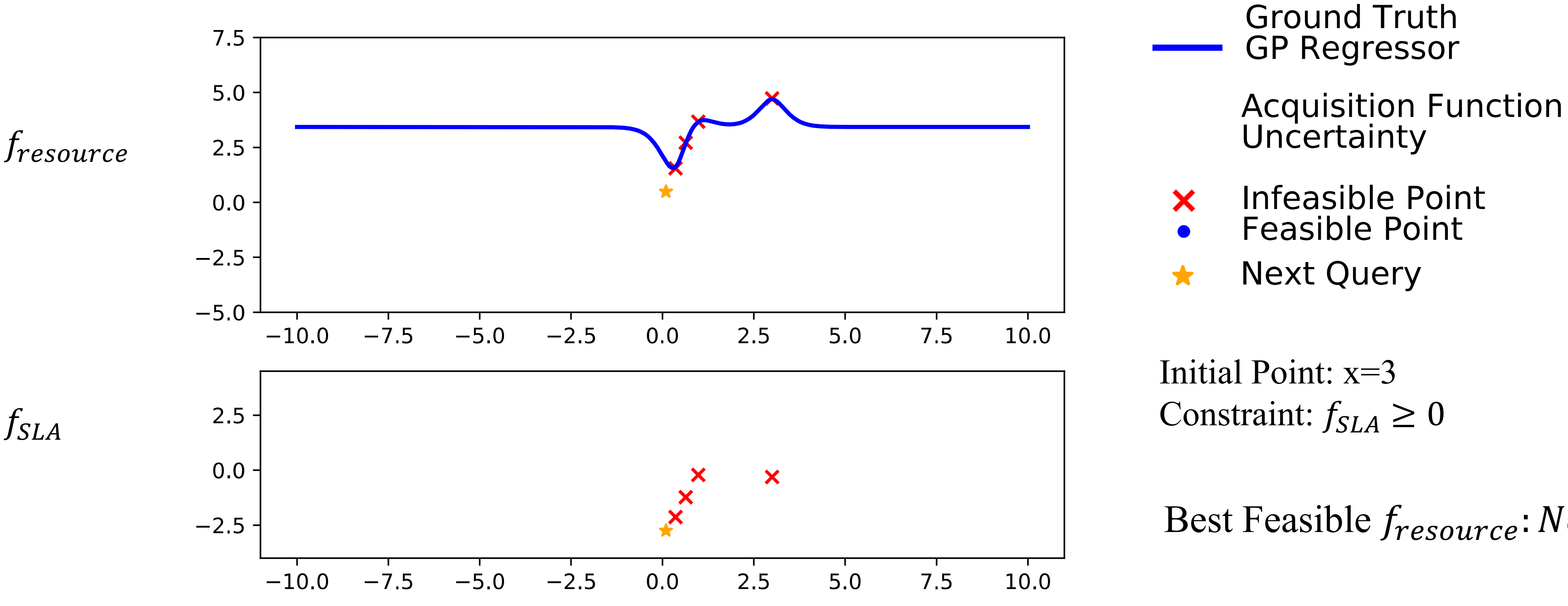
Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.



Iteration 4

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

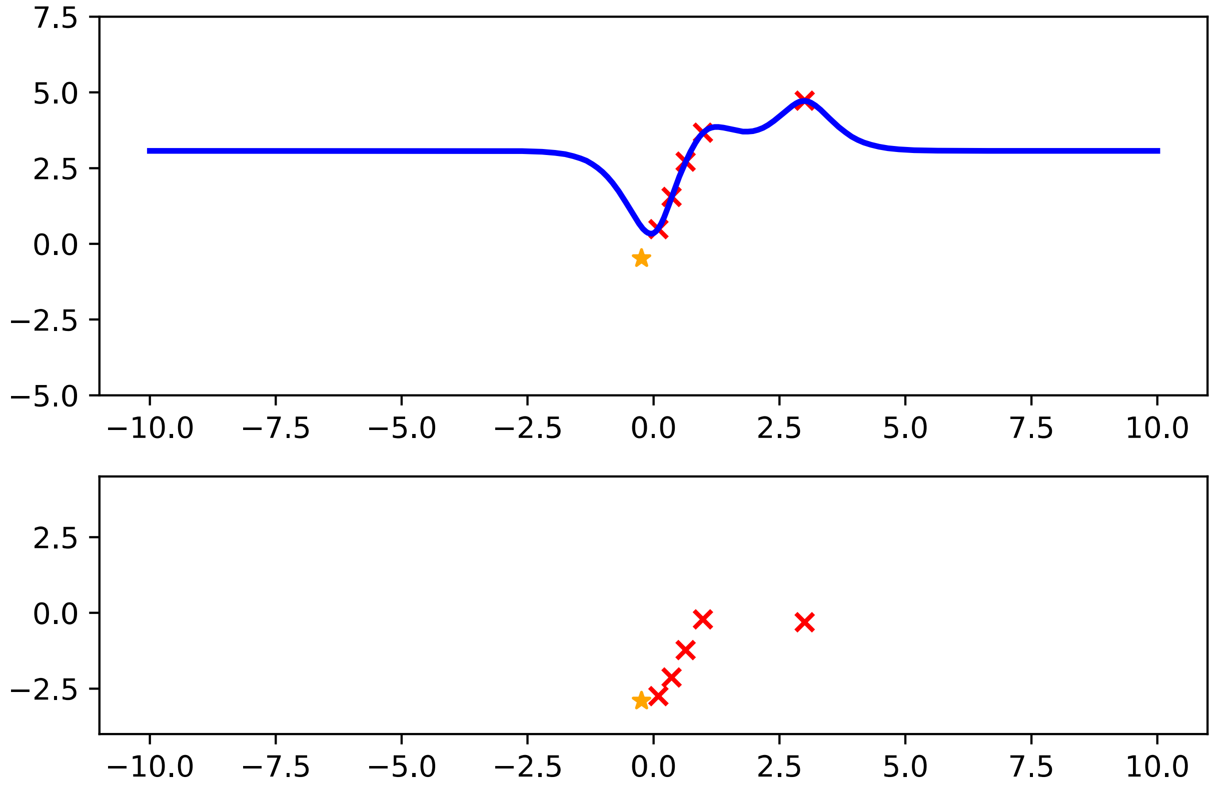
Best Feasible $f_{resource}$: $None$

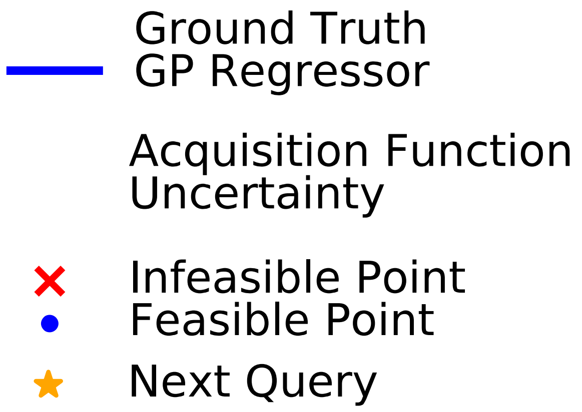
Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}





Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

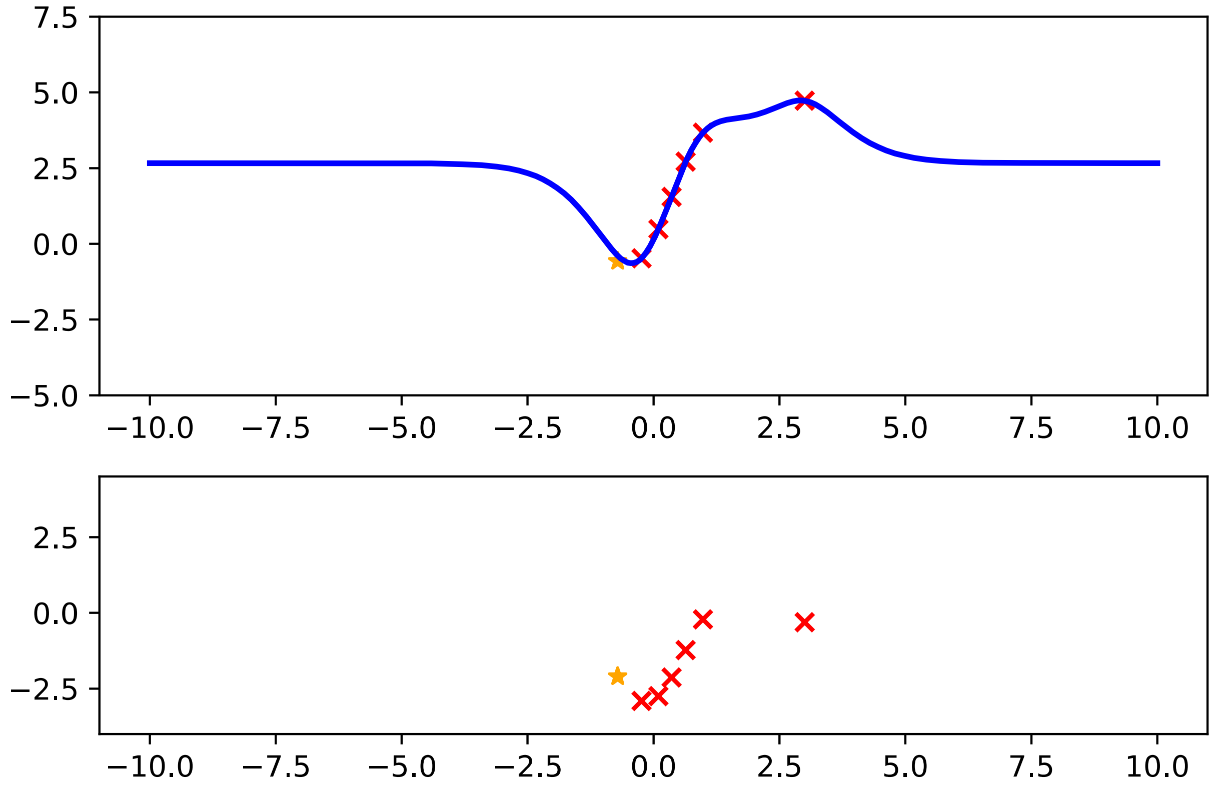
Iteration 5

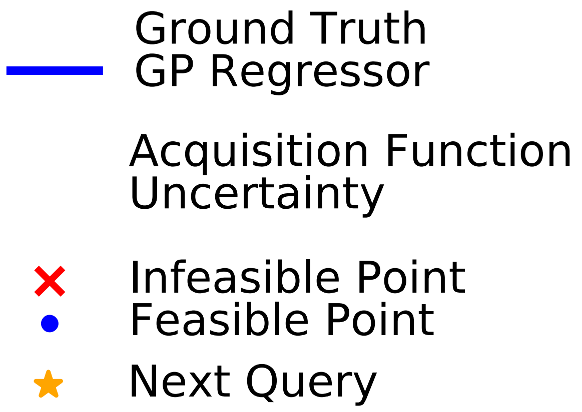
Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}





Initial Point: x=3

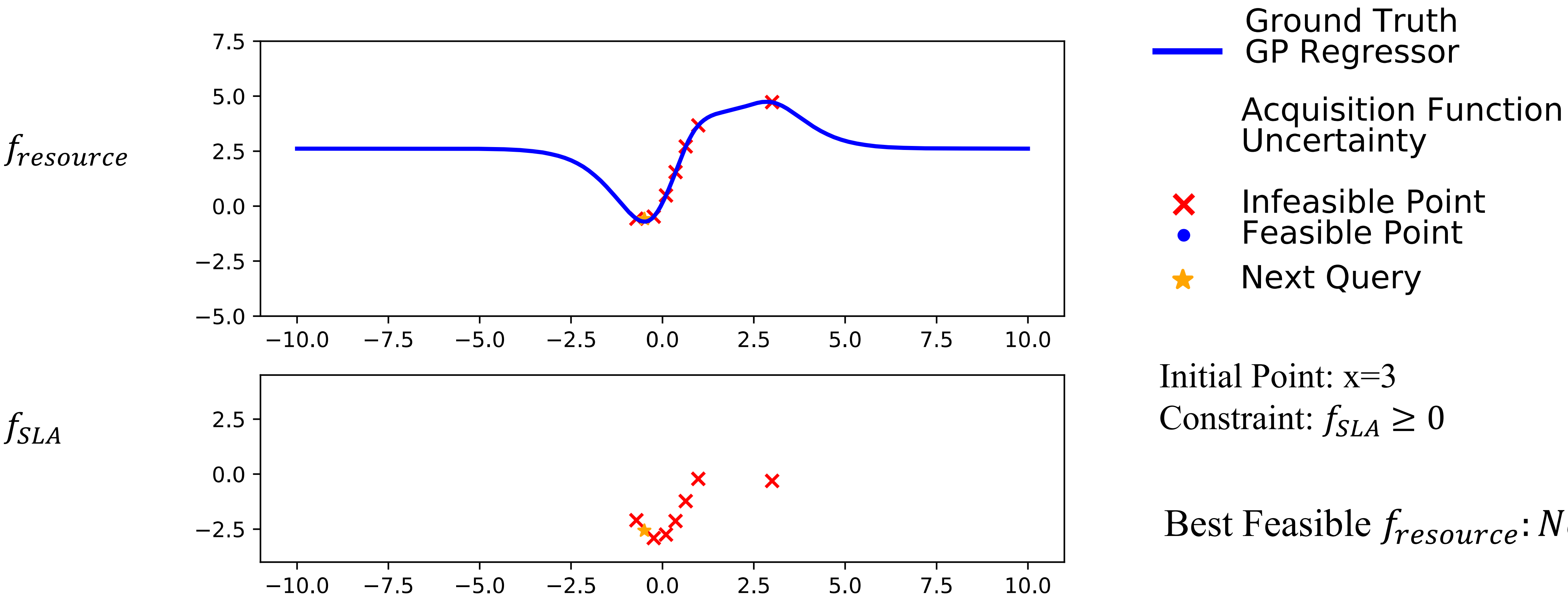
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

Iteration 6

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.



Iteration 7

Initial Point: x=3

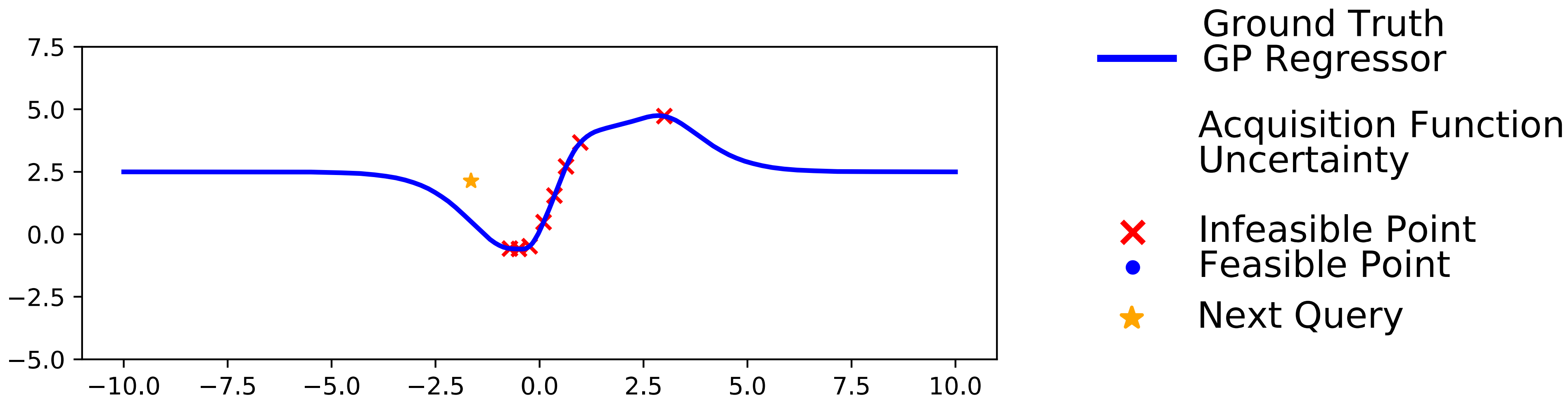
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

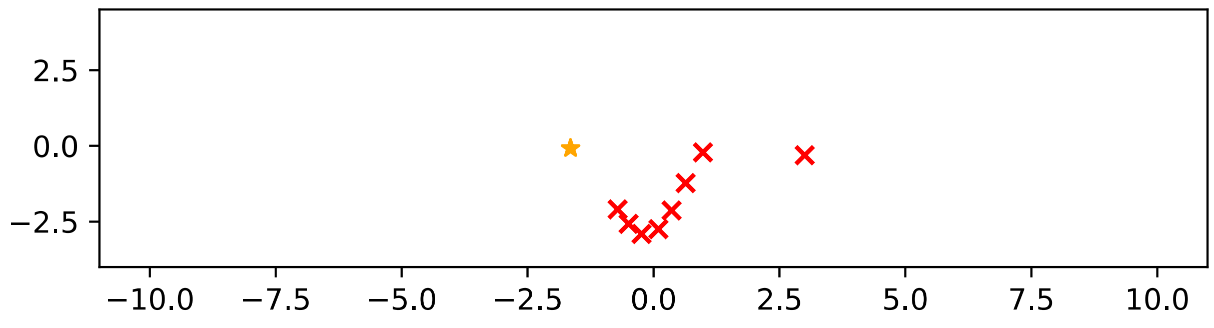
Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$



f_{SLA}



Iteration 8

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

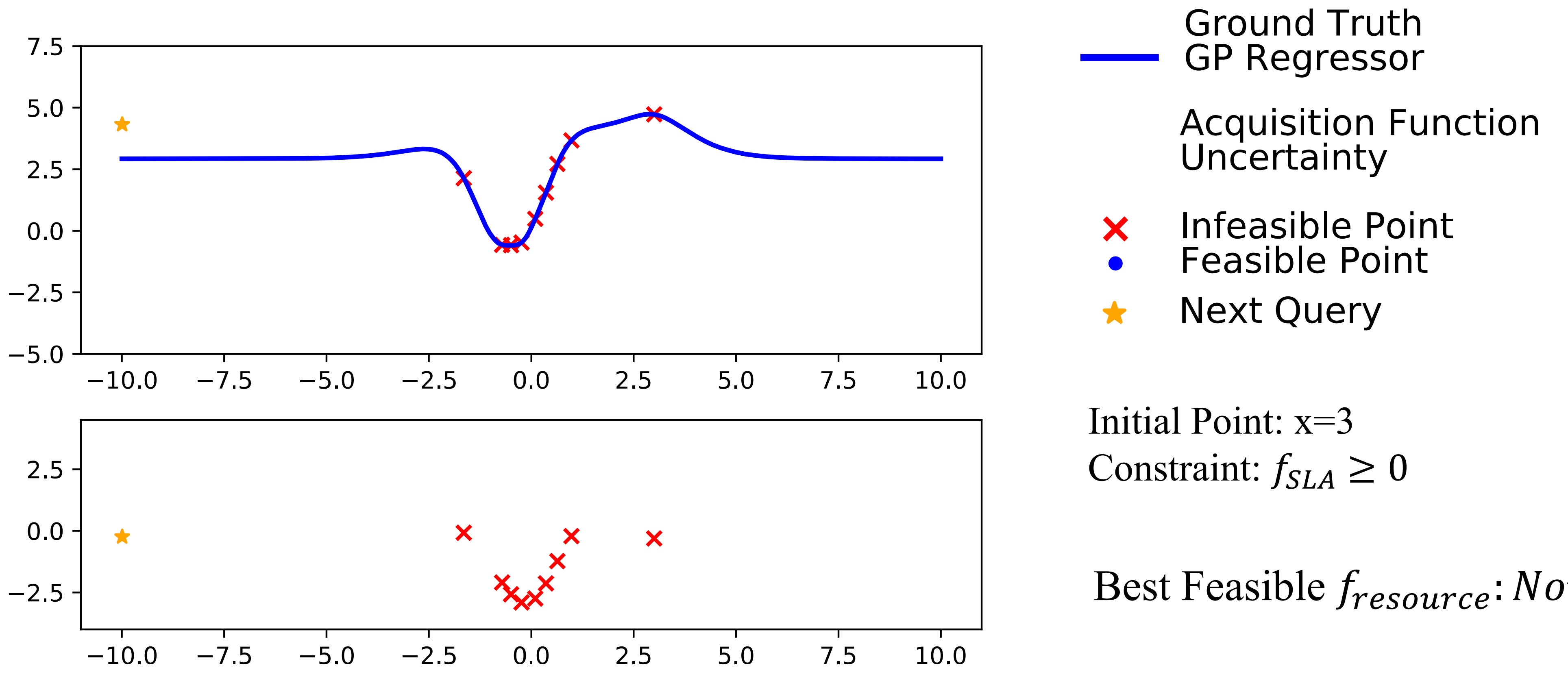
Best Feasible $f_{resource}$: $None$

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}



Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

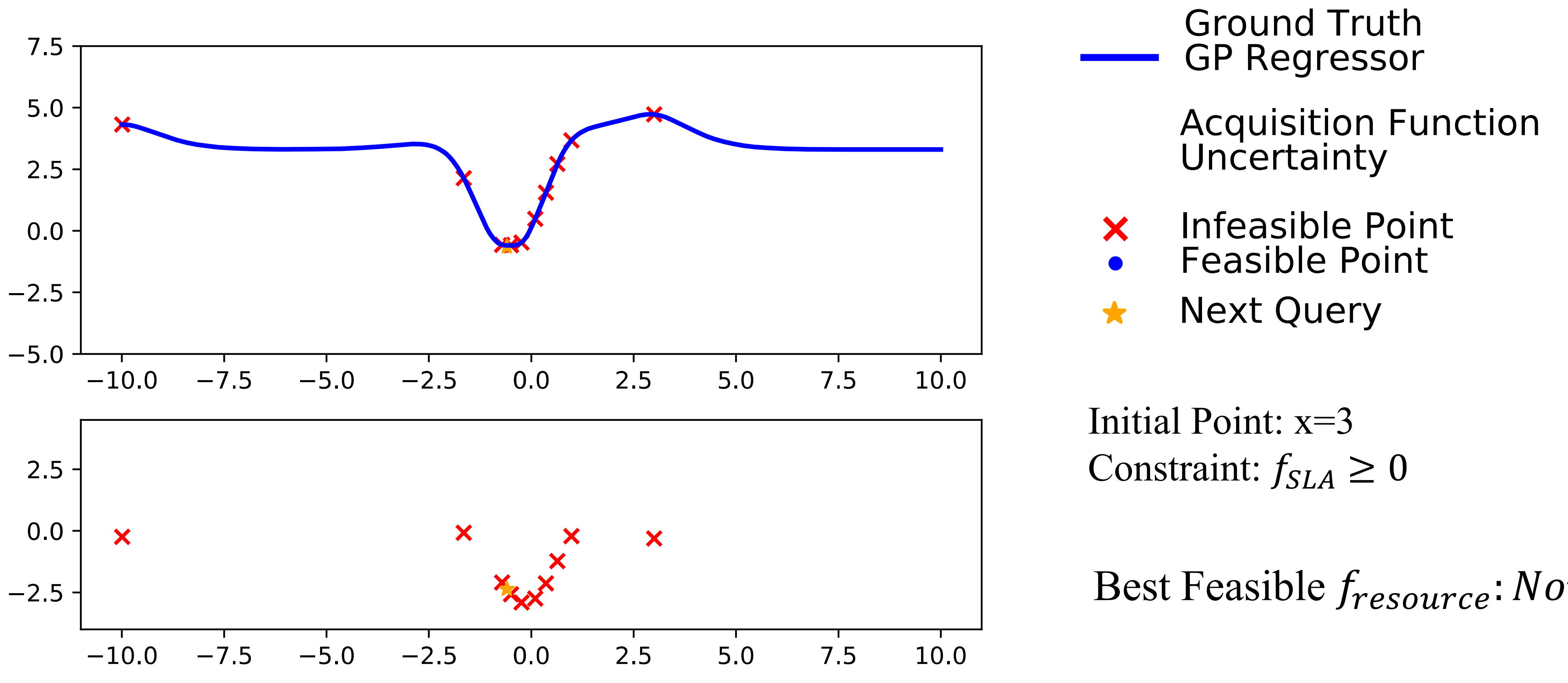
Iteration 9

Solving Constrained Optimization

- Tradition Bayesian Optimization uses acquisition function (.e.g, the Expected Improvement α_{EI}) to guide the search of the optimal.

$f_{resource}$

f_{SLA}



Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: $None$

Iteration 10

Solving Constrained Optimization

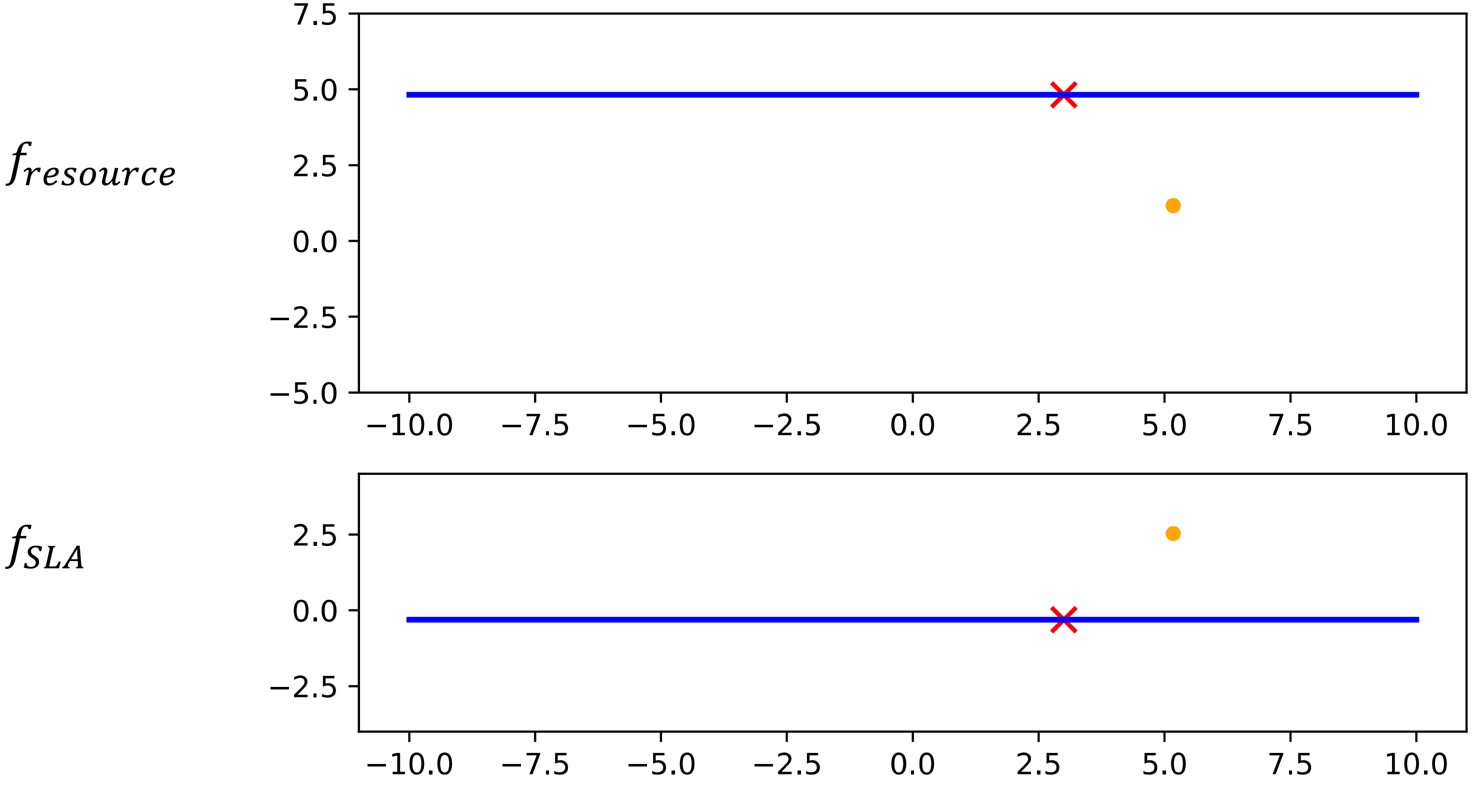
How to balance?

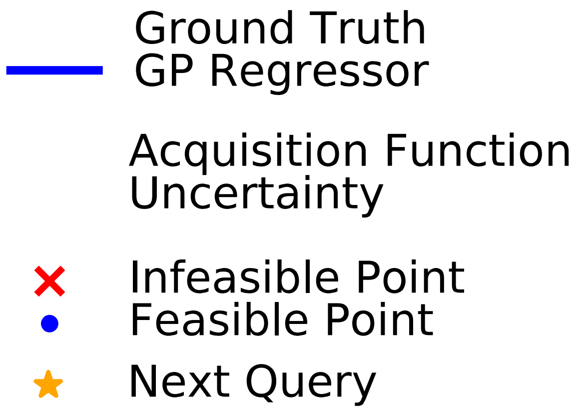
- To solve our constrained optimization problem, we extend the acquisition function:

$$\alpha_{CEI} = \alpha_{EI} \times \ Prob(feasibility)$$

  - We also use Gaussian Process to model $Prob(feasibility)$

Guiding Search in Feasible Region



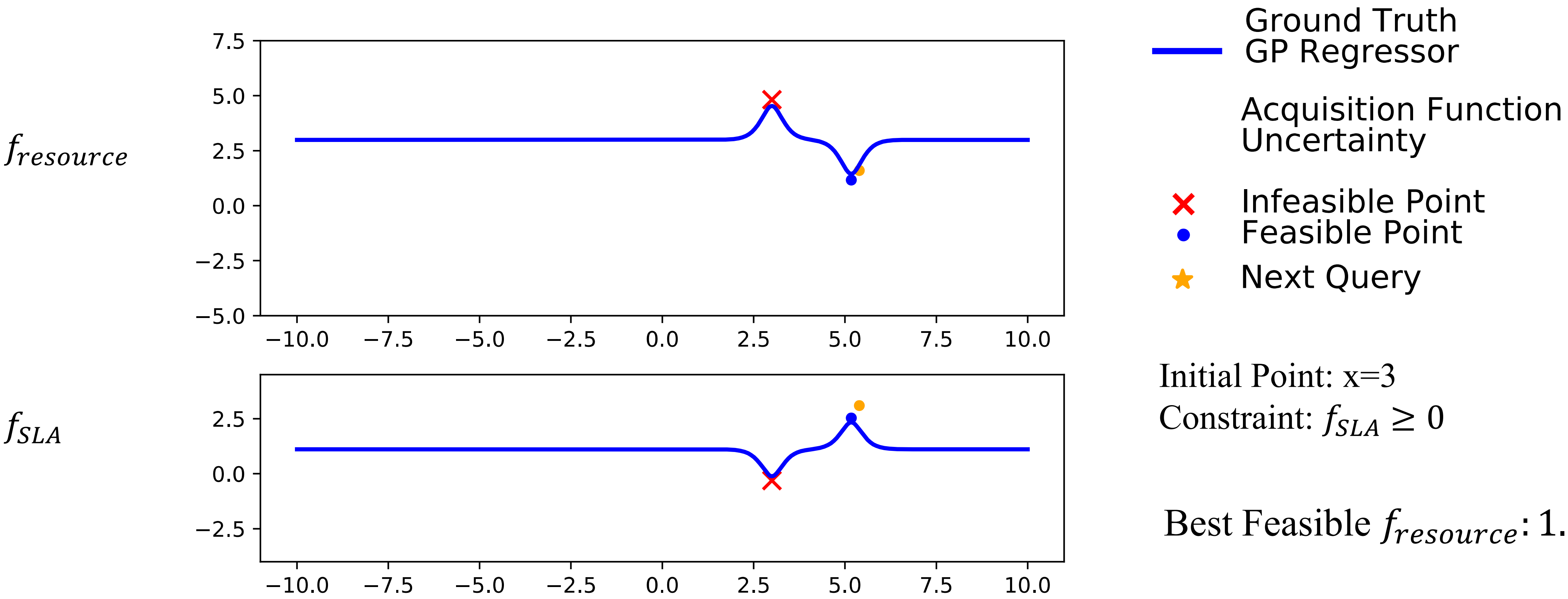


Iteration 1

Initial Point: x=3
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 1.1656

Guiding Search in Feasible Region



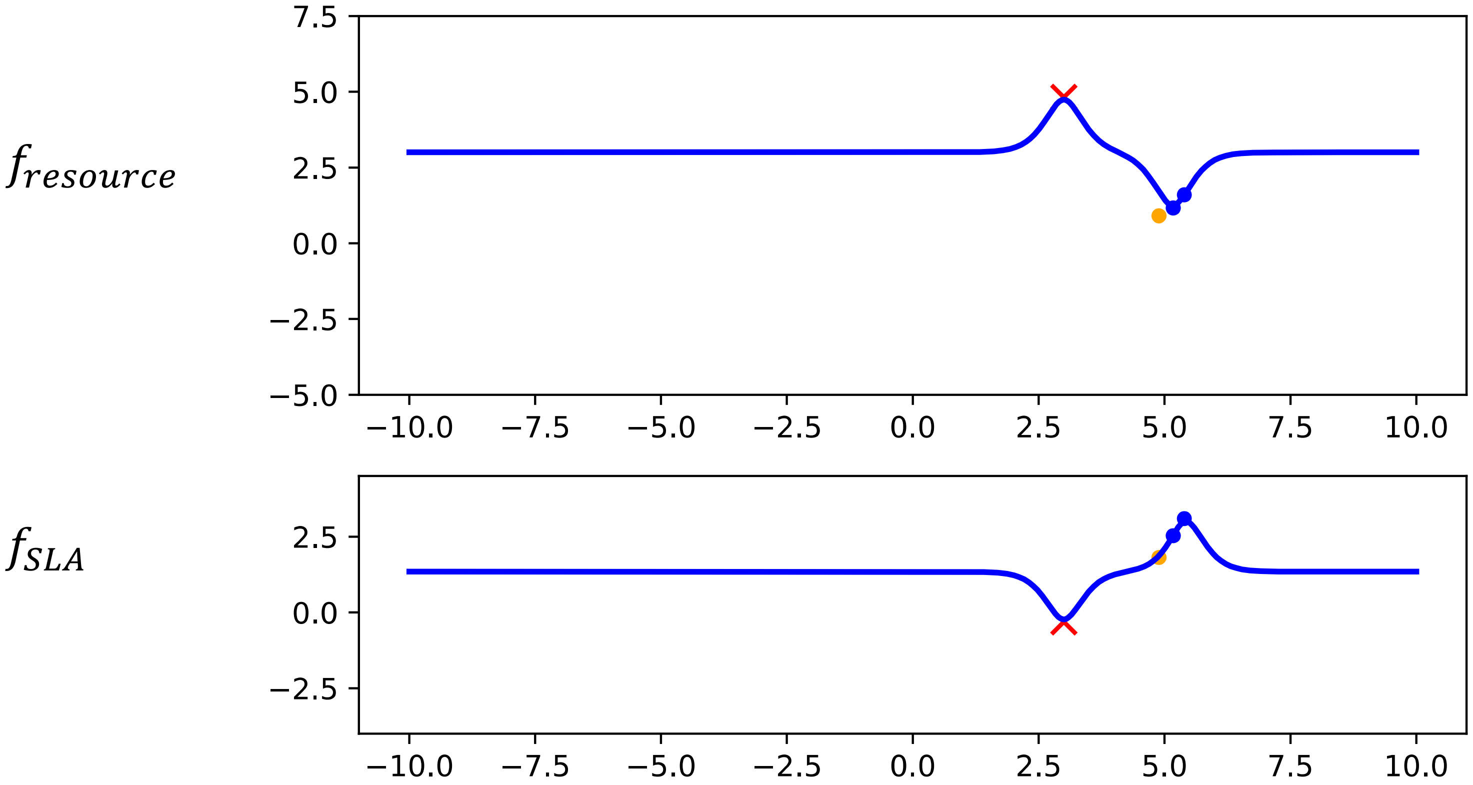
Iteration 2

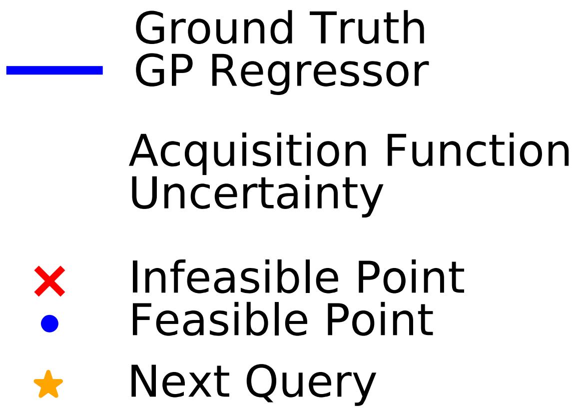
Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 1.1656

Guiding Search in Feasible Region



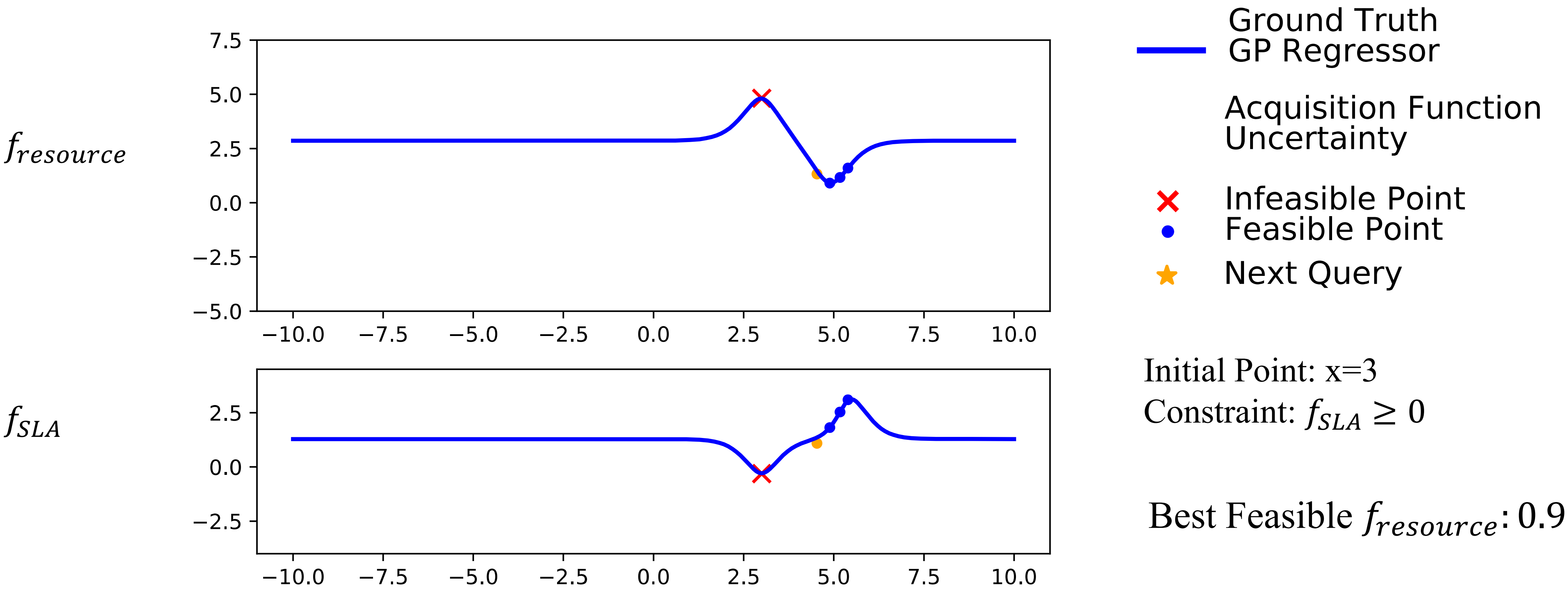


Initial Point: x=3
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.9071

Iteration 3

Guiding Search in Feasible Region



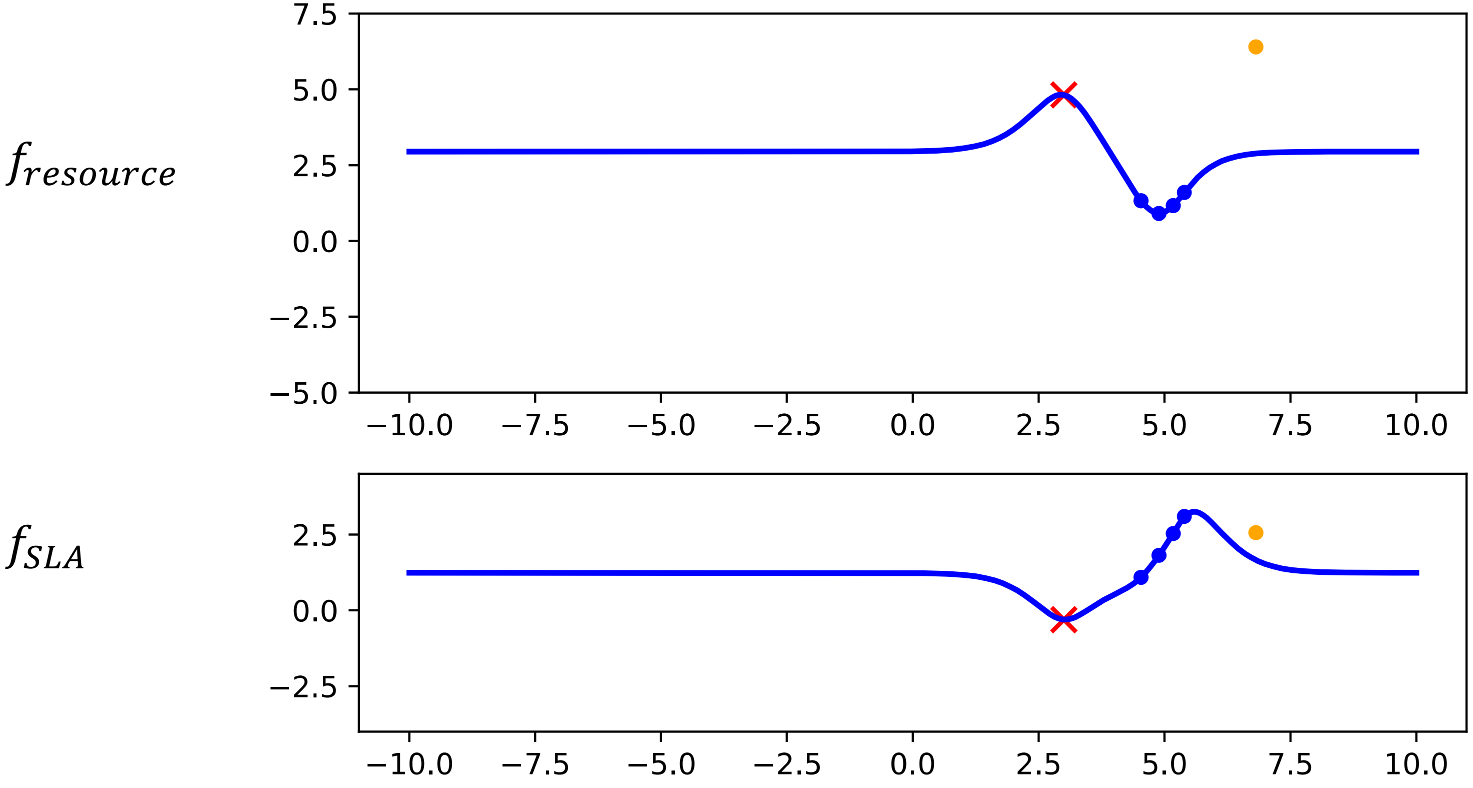
Iteration 4

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.9071

Guiding Search in Feasible Region



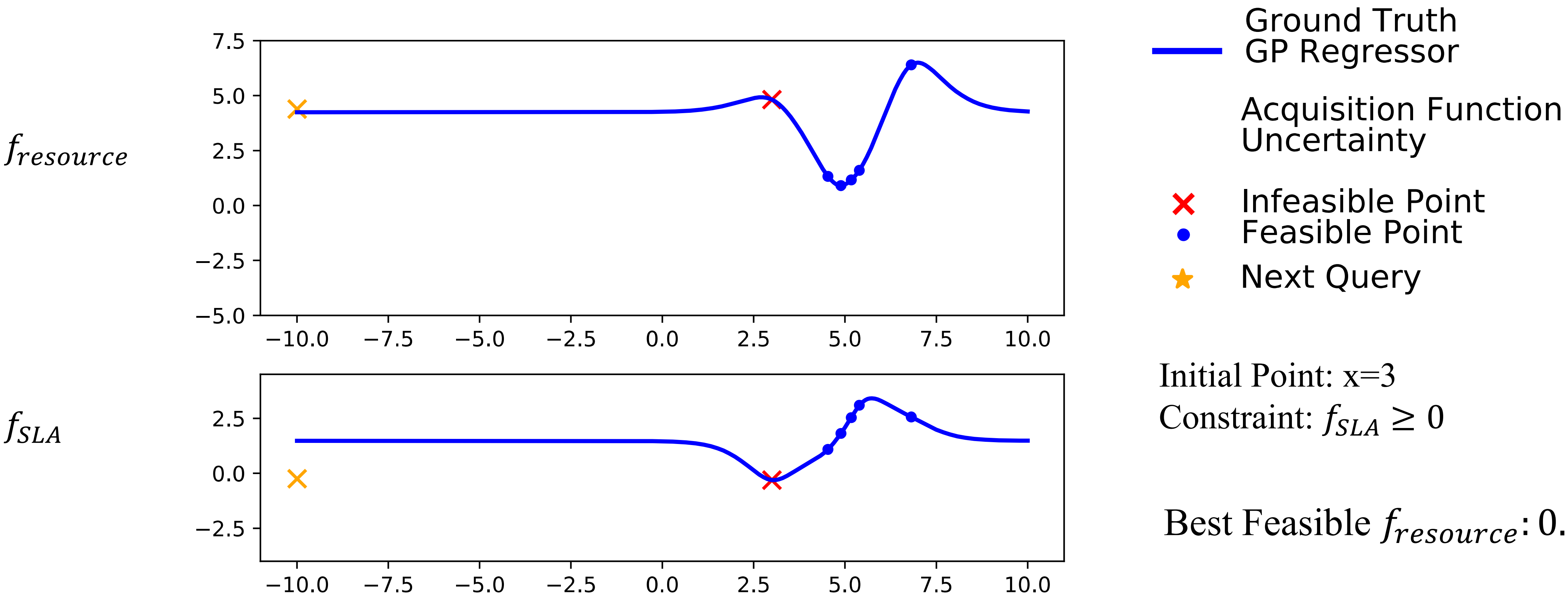
Iteration 5

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.9071

Guiding Search in Feasible Region



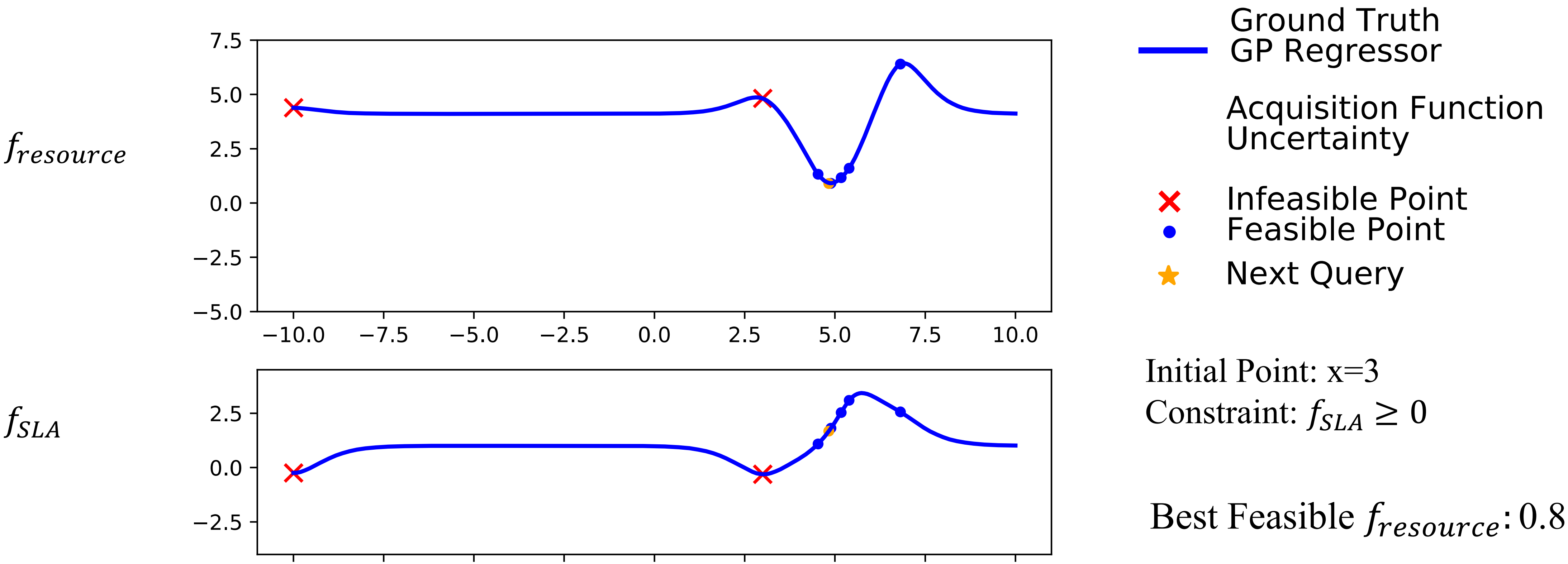
Iteration 6

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.9071

Guiding Search in Feasible Region



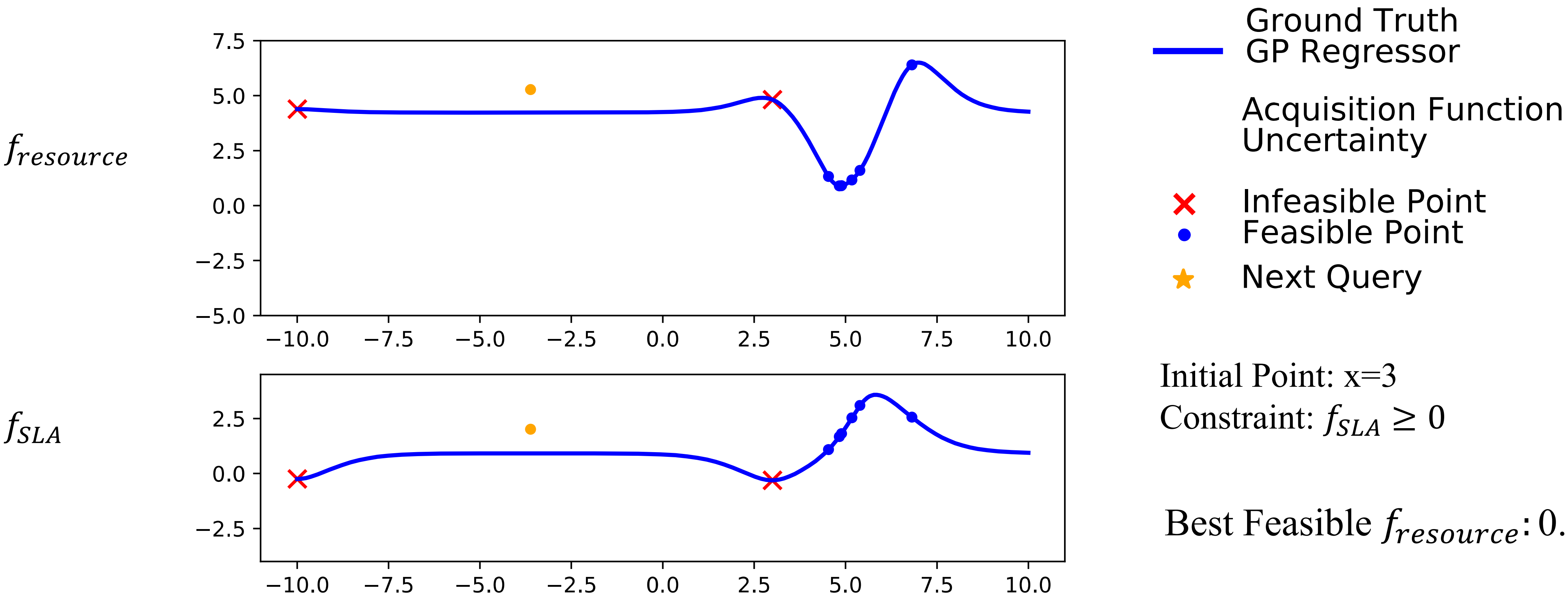
Iteration 7

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.8991

Guiding Search in Feasible Region

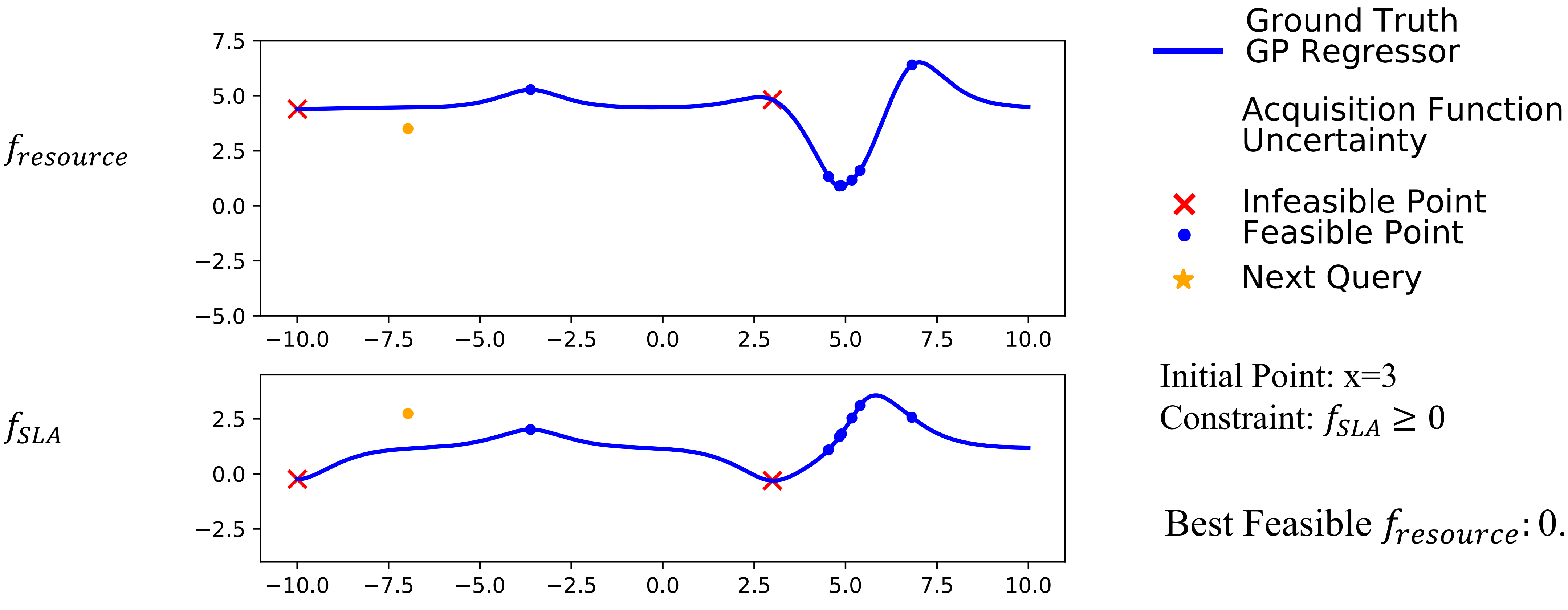


Initial Point: x=3
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.8991

Iteration 8

Guiding Search in Feasible Region



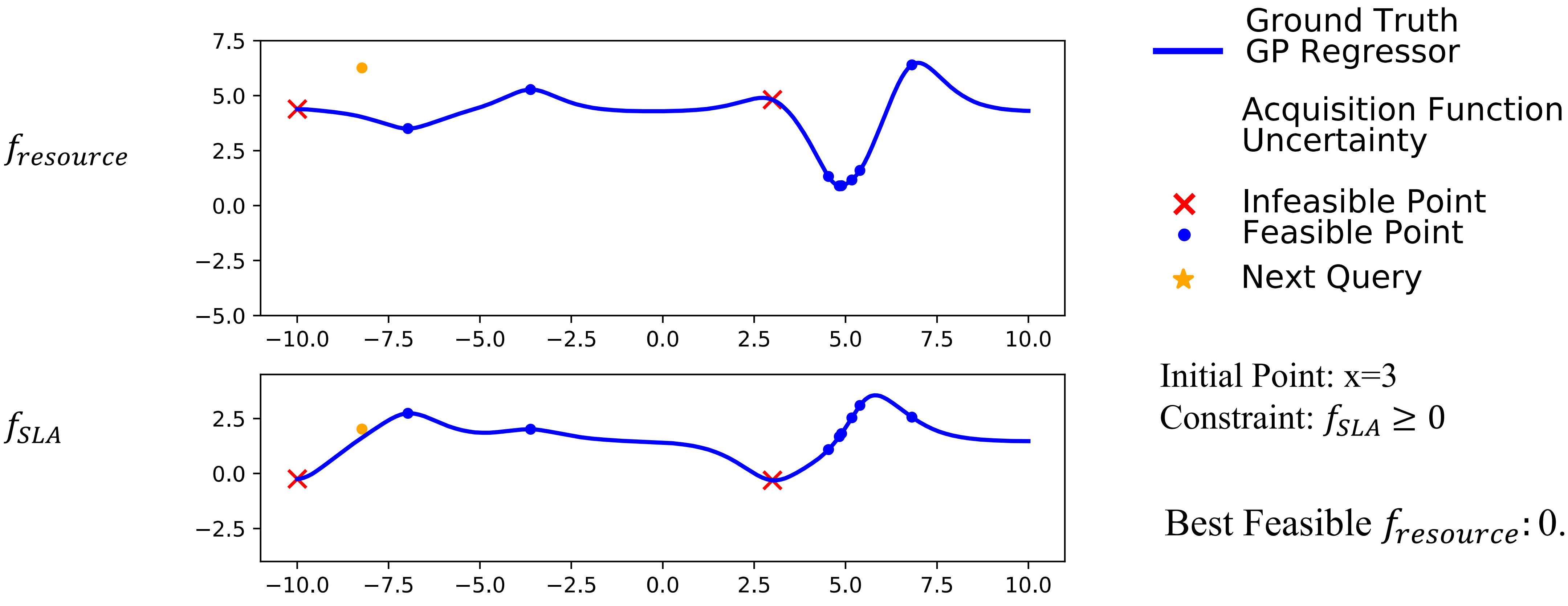
Iteration 9

Initial Point: x=3

Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.8991

Guiding Search in Feasible Region



Iteration 10

Initial Point: x=3

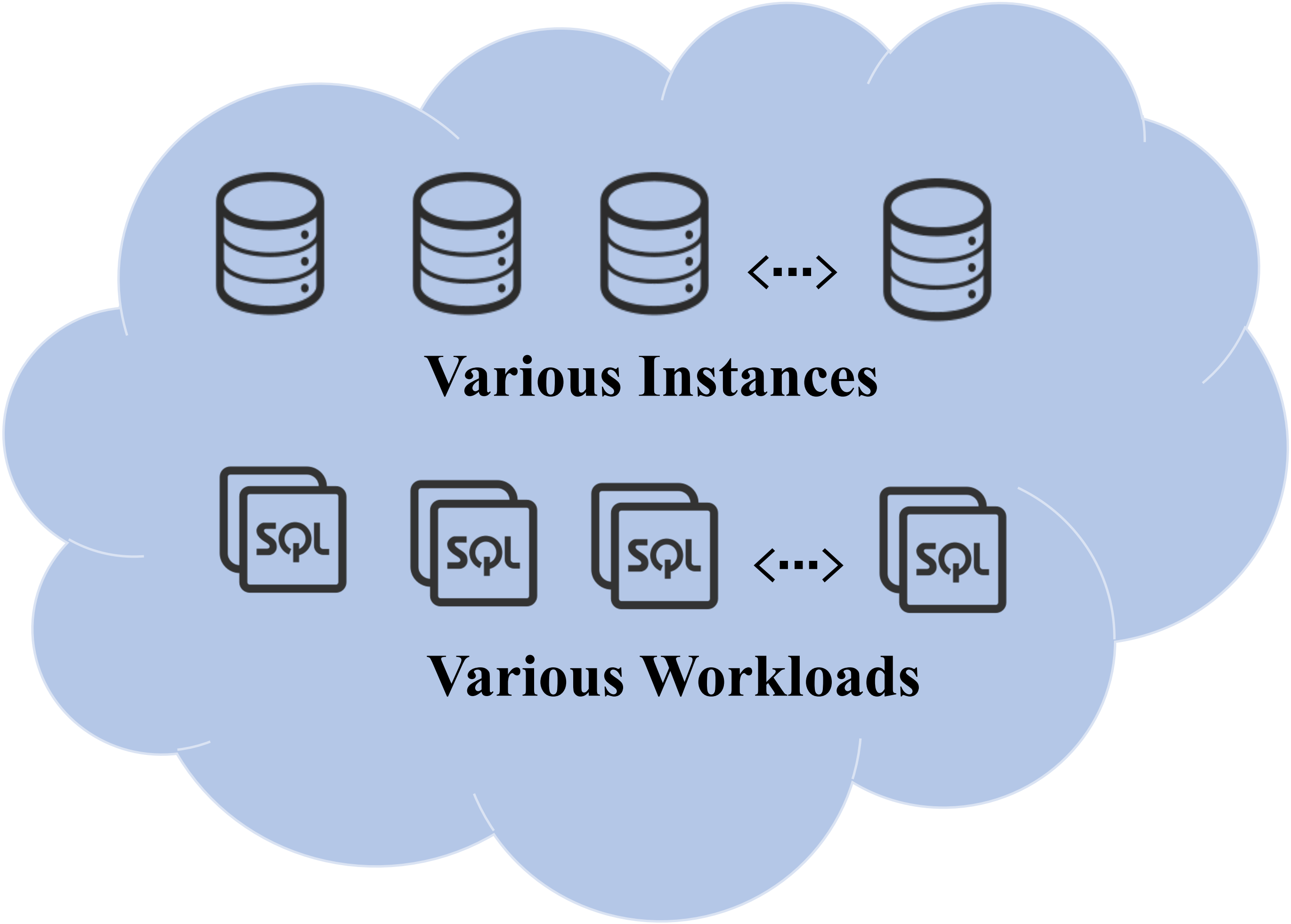
Constraint: $f_{SLA} \geq 0$

Best Feasible $f_{resource}$: 0.8991

Boosting Tuning Process

- The same workloads running on different hardware share information for tuning knobs.

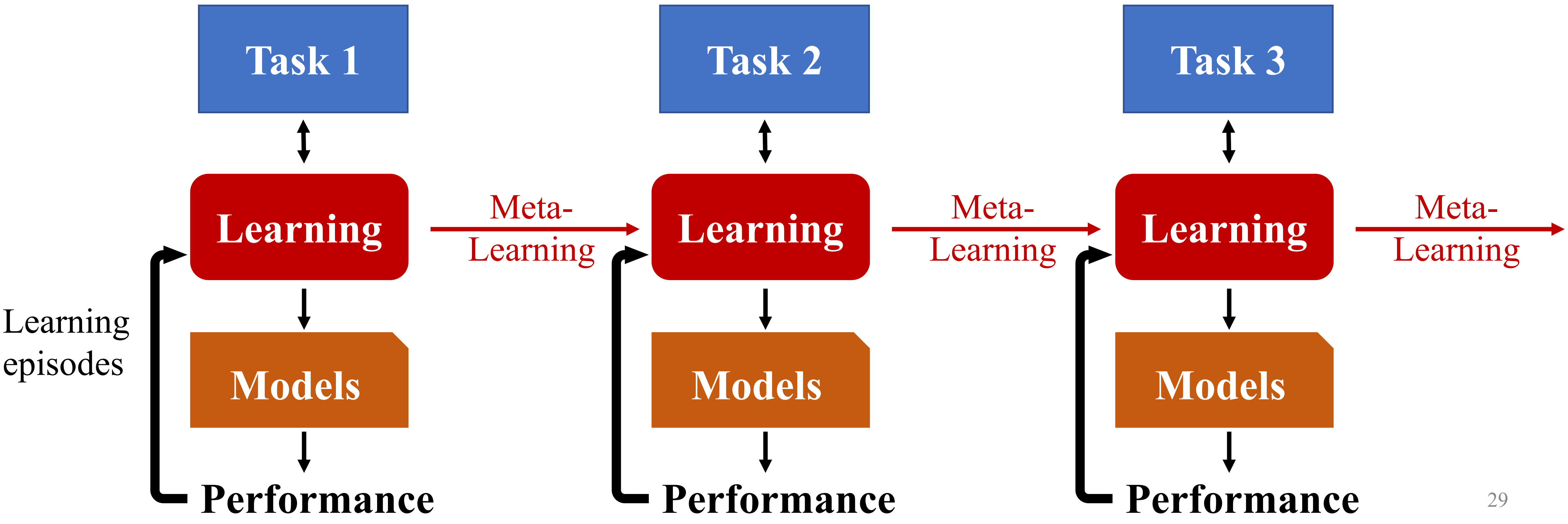
- Even for different workloads, the relationship between hidden features can lead to knowledge sharing.



Numerous tuning tasks on the cloud

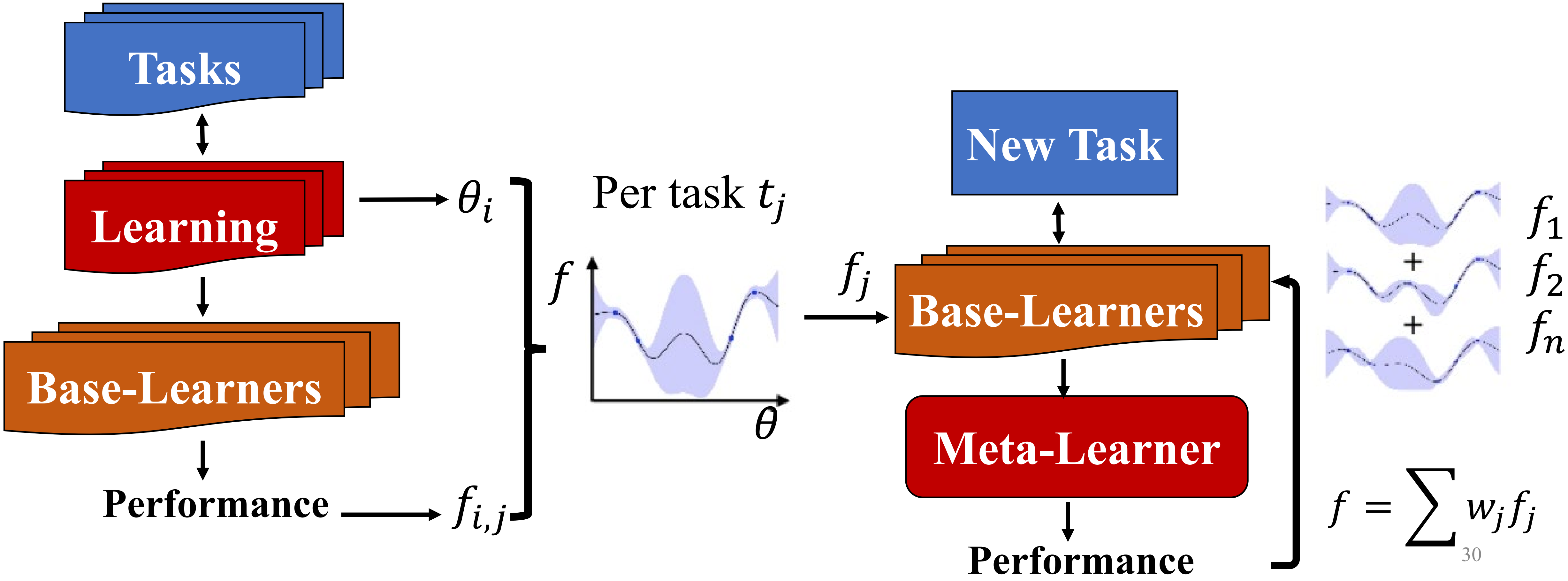
Boosting Tuning Process: Meta-Learning

- Human learns across tasks.
- Why? Require less trial-and-error, less data



Knowledge Extraction

The prior knowledge is extracted from historical tuning tasks by ensemble.



How to determine the weights?

Learning from Meta-Feature → **Learning from Model Predictions**

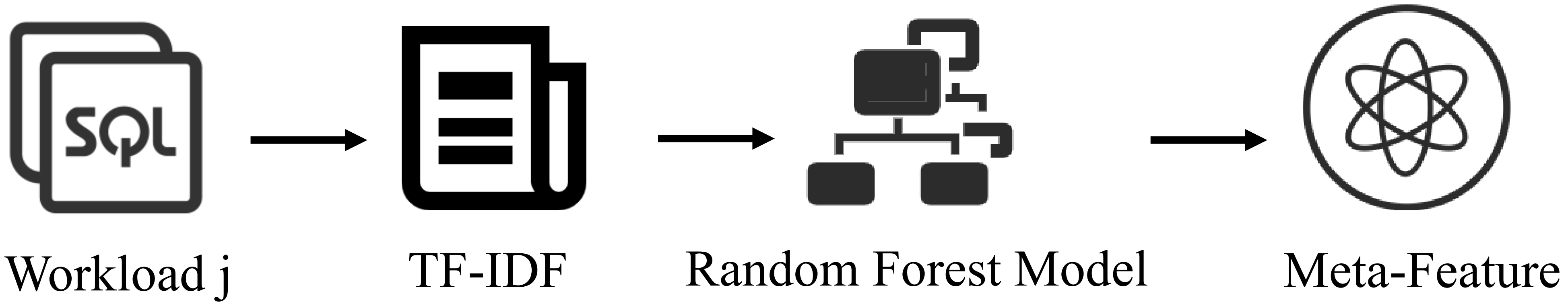
- Static
- Good initialization

- Dynamic
- Avoid over-fitting

Learning from Meta-Feature

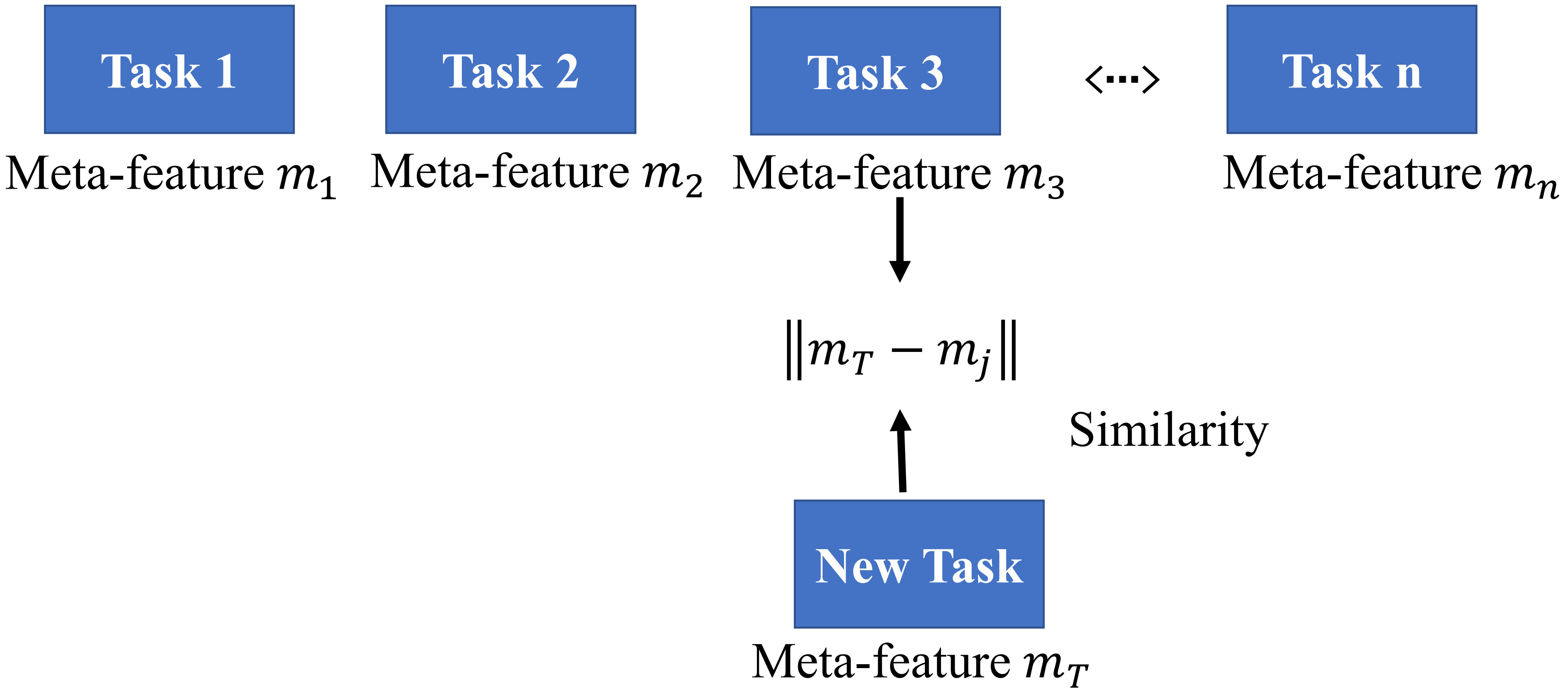
- Meta-features: measurable properties of tasks
- ResTune learns the meta-feature by workload characterization.

A Workload characterization pipeline



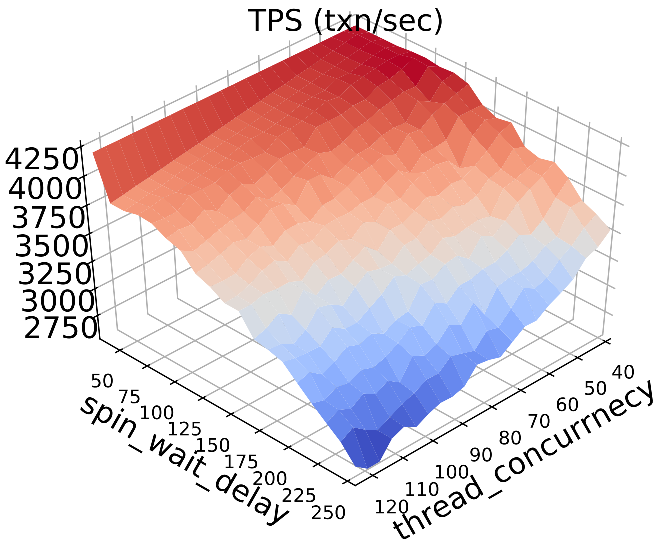
Learning from Meta-Feature

- The static weight is calculated by the distance between meta-features.

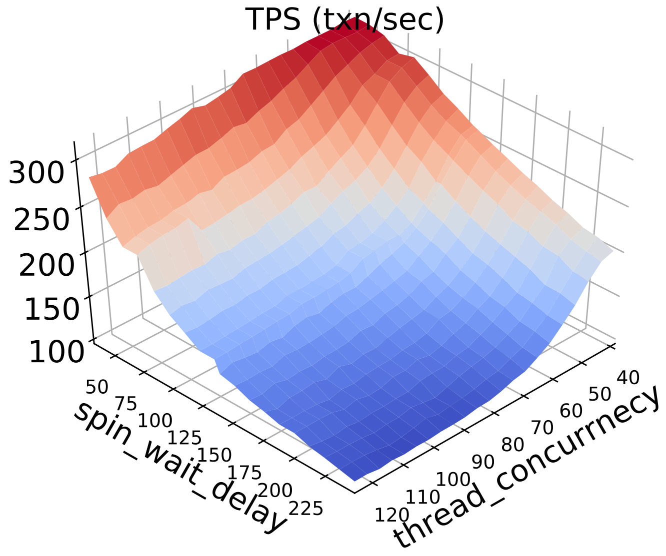


Learning form Model Predictions

- We define base-learners' similarity in terms of how accuracy base-learner can predict the performance of the target task.
- Challenge: The performances can differ in scale significantly among various hardware environments in the cloud.



Instance A

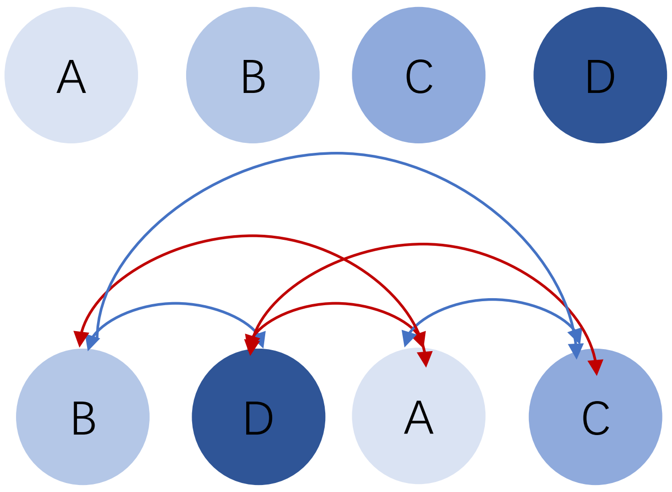


Instance B

Learning form Model Predictions

- Our observation: the actual values of the predictions do not matter, since we only need to identify the location of the optimum!
- We calculate the ranking loss of base learners against target observations.

Target ranking:
(Ground Truth)



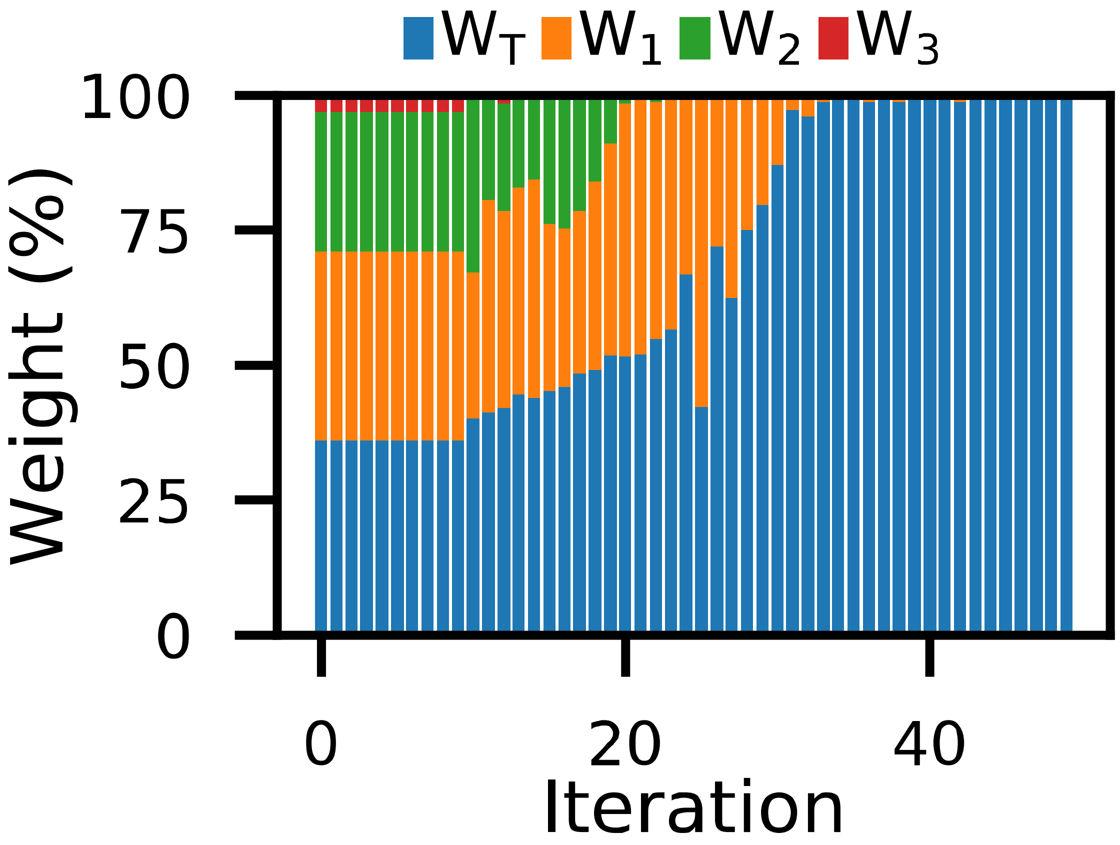
Base-learner j ranking:

$$\text{Ranking Loss for j} = \frac{\#\,Misranking\ pairs}{\#Pairs}$$
$$= \frac{6}{12}$$

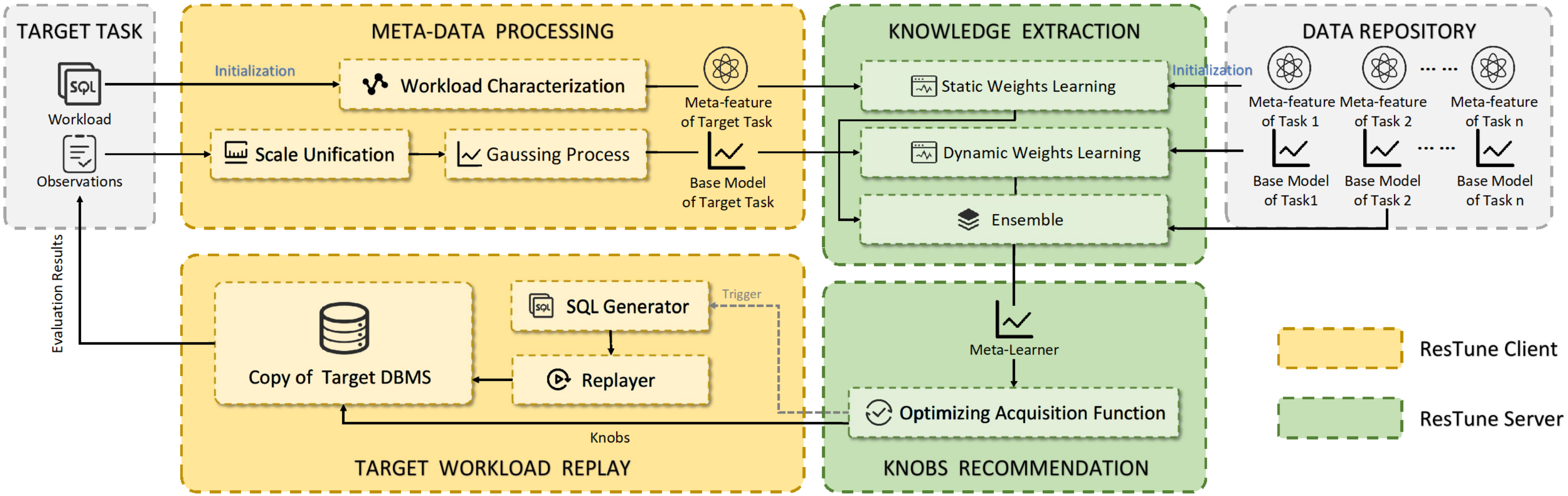
Adaptive weight schema

- Static Weight Assignment:
  - Meta-features gives a coarse-grained abstraction about task properties.
  - Suggesting knobs that are promising according to similar historical tasks.

- Dynamic Weight Assignment:
  - Ranking of model predictions measures the similarity of tasks in the optimization problem.
  - Avoiding over-fitting by shrinking historical base learners' weight.



System Architecture of ResTune



Experimental Study

- DBMS: version 5.7 of MySQL RDS
- Hardware instances:

	A	B	C	D	E	F
CPU	48 cores	8 cores	4 cores	16 cores	32 cores	64 cores
RAM	12GB	12GB	8GB	32GB	64GB	128GB

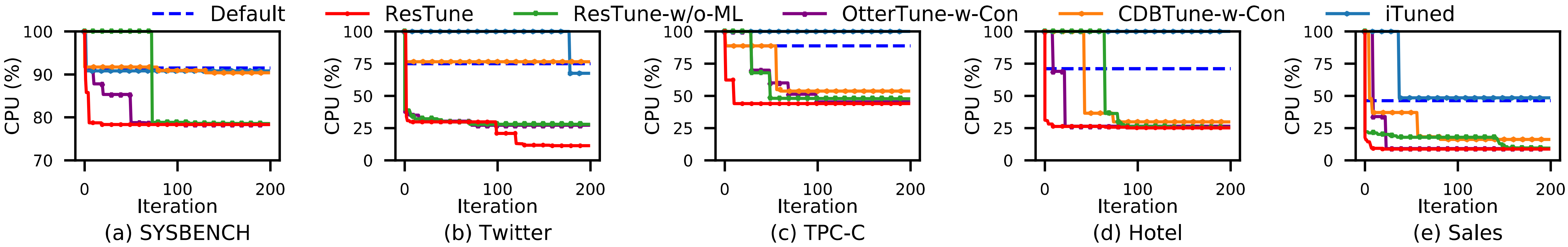
- Workloads:
  - Three Benchmark workloads: SYSBENCH、TPC-C、Twitter
  - Two real world workloads: Hotel、Sales
- Data Repository:
  - We collect workload features and observation histories of 34 past tuning tasks on instances A and B as our meta-data

Experimental Study

- Baselines:
  - **Default**: The default knobs provided by experienced DBA;
  - **iTuned**: We change its objective to minimizing the resource utilization;
  - **OtterTune-w-Con**: We replace OtterTune's acquisition function to our designed CEI to guide search in feasible region;
  - **CDBTune-w-Con**: We modify its reward function to encourage the agent to minimize resource usage and satisfy the SLA;
  - **ResTune-w/o-ML**: ResTune without Meta-Learning;
  - **ResTune**: Our approach that uses the meta-learner to boost the tuning.

iTuned [VLDB 2009]; OtterTune [SIGMOD 2017] ; CDBTune [SIGMOD 2019]:

Efficiency Comparison



Performances of various workload on Instance A

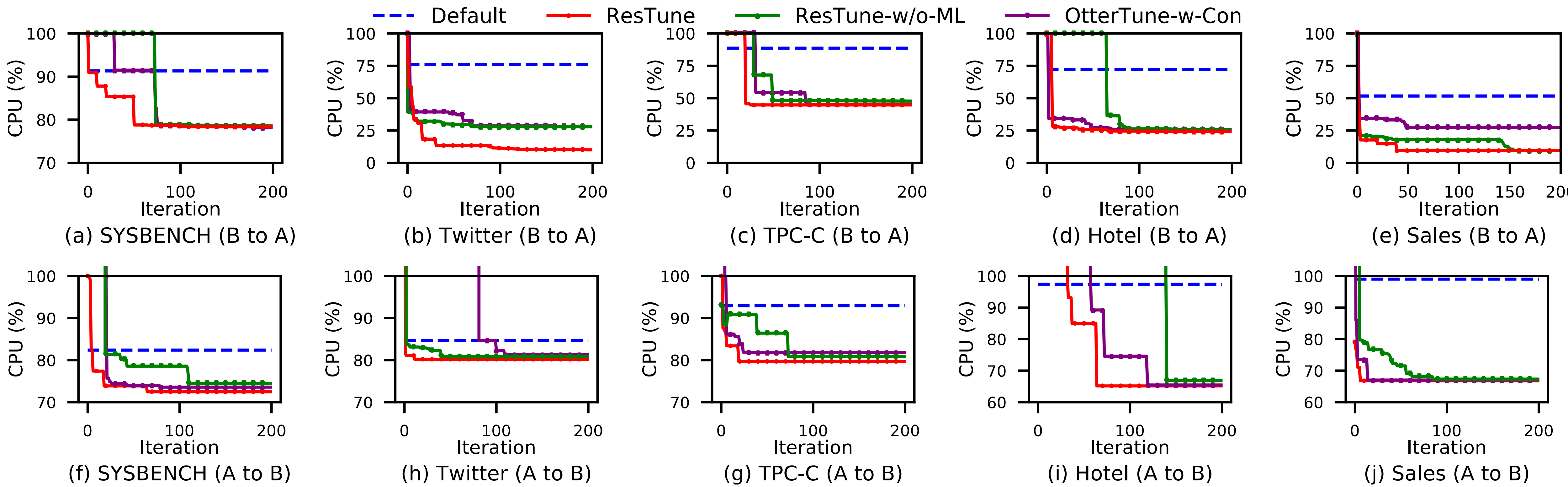
Takeaway:

- ResTune can reduce the default CPU usage by 50.1% on average and guarantee the SLA.
- ResTune-w/o-ML performs much better than iTuned and CDBTune-w-Con.
- With meta-learning design, ResTune achieves 18.6X speedup than OtterTune-w-Con in SYSBENCH and 7.38X speedup on average.

Evaluation on Adaptability

- Hardware Adaption
  - B to A
  - A to B
  - AB to C, D, E and F respectively

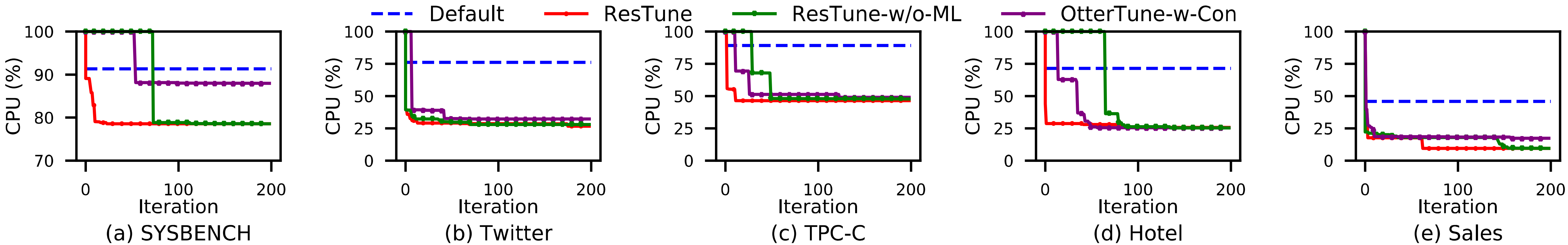
- Workload Adaption
  - holding out the target workload’s data from the data repository



(a) SYSBENCH (B to A) (b) Twitter (B to A) (c) TPC-C (B to A) (d) Hotel (B to A) (e) Sales (B to A)

(f) SYSBENCH (A to B) (h) Twitter (A to B) (g) TPC-C (A to B) (i) Hotel (A to B) (j) Sales (A to B)

Performance Adapting to Different Hardware Environments



(a) SYSBENCH (b) Twitter (c) TPC-C (d) Hotel (e) Sales

Performance Adapting to Different Worklaods

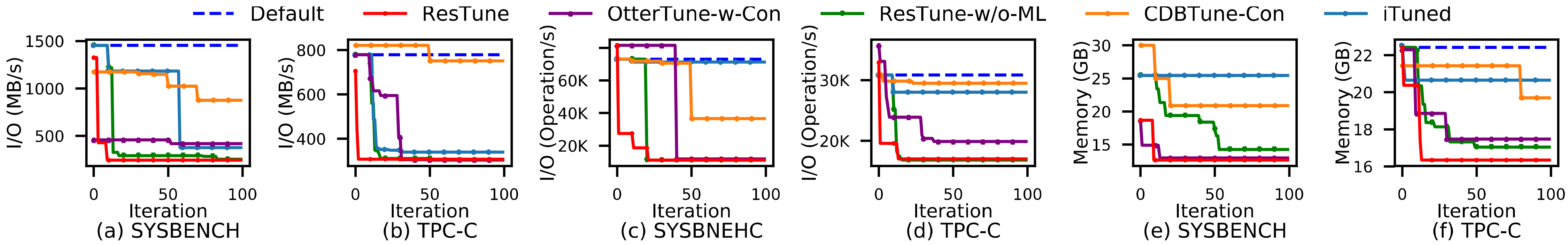
Evaluation on Adaptability

Instance			C	D	E	F
SYSBENCH	Improvement	Restune	5.02%	8.13%	17.16%	20.38%
		Restune-w/o-ML	3.34%	7.58%	16.76%	19.96%
	Iteration	Restune	37	64	100	35
		Restune-w/o-ML	57	80	115	53
		Speed Up	35%	20%	14%	34%
TPC-C	Improvement	Restune	4.96%	19.22%	33.26%	47.60%
		Restune-w/o-ML	2.78%	18.28%	33.09%	42.62%
	Iteration	Restune	12	25	45	18
		Restune-w/o-ML	99	47	79	25
		Speed Up	87.87%	46.80%	43.03%	28%

Hardware Adaptation on More Instances

Tuning other types of Resources

- Other types of resources
  - I/O (BPS and IOPS)
  - Memory



Performance Tuning Other Types of Resources

- Takeaway:
  - ResTune reduces 87% of I/O, and 39% of memory on average.

Thanks for Listening!

My E-mail: zhang_xinyi@pku.edu.cn

Execution Time Breakdown

Phase	ResTune	ResTune-w/o-ML	iTuned	CDBTune-w-Con	OtterTune-w-Con
Meta-Data Processing	0.653s~1.983s	/	/	/	/
Model Update	0.312s~2.298s	0.649s	0.151s	0.586s	11.347s
Knob Recommendation	5.115s	1.907s	0.912s	0.005s	4.457s
Target Workload Replay	**182.237s(95.1%)**	**182.237s(98.6%)**	**182.186(99.4%)**	**182.336s(99.7%)**	**182.337s(92.0%)**
Total Time	191.630s	184.793s	183.245s	182.927s	198.141s