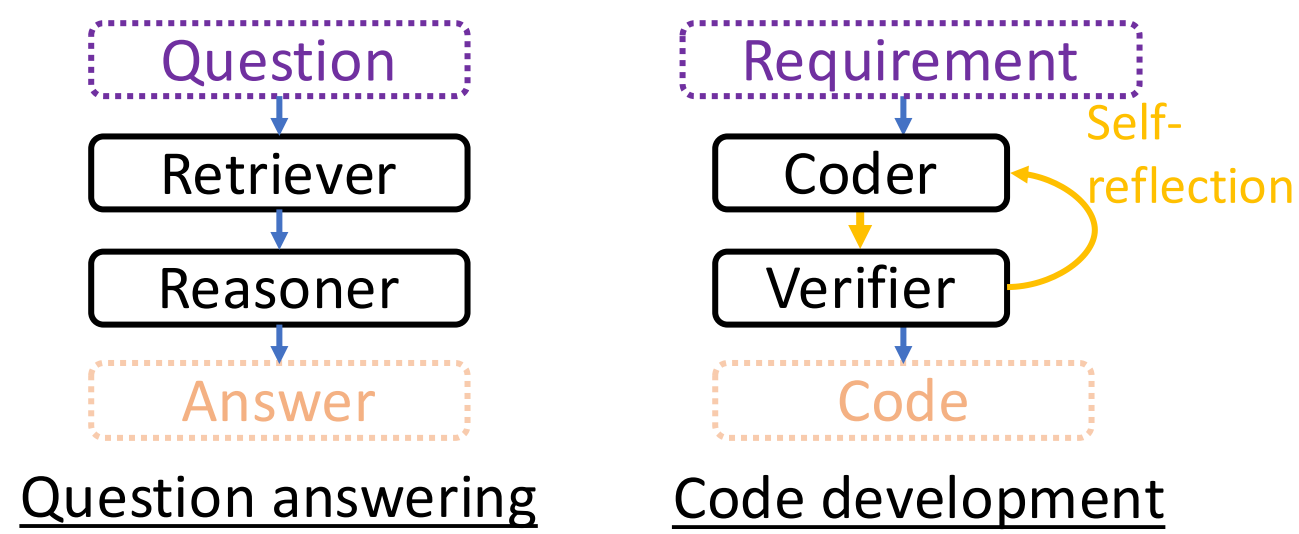


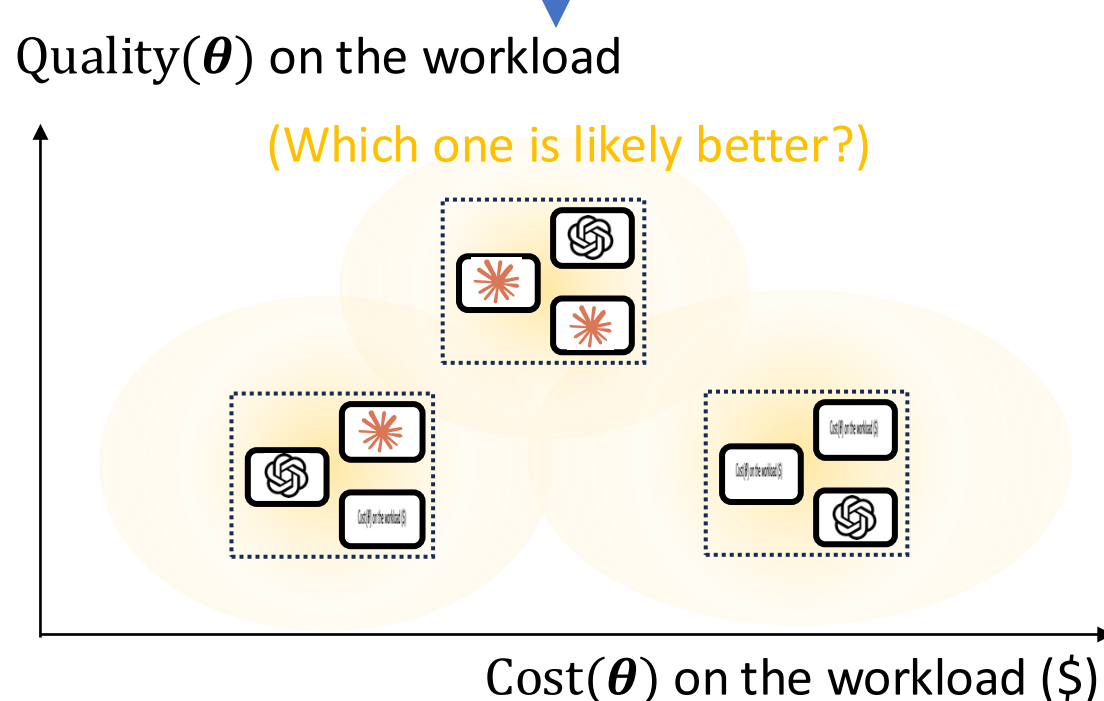
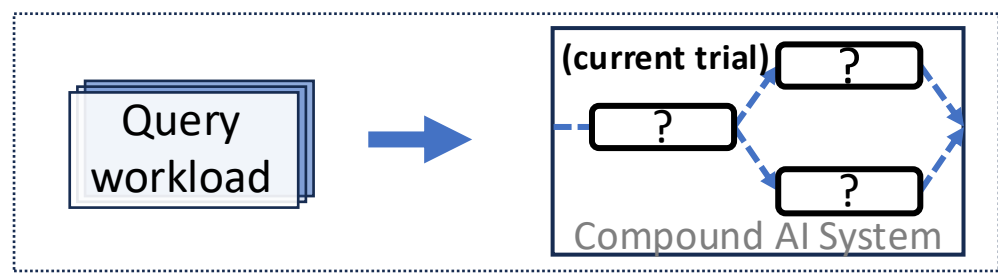
Compound AI System

It is a system made up of multiple AI agents



Existing Solutions' Recipe

Existing solutions use *the whole workload* to figure out which configuration to try next

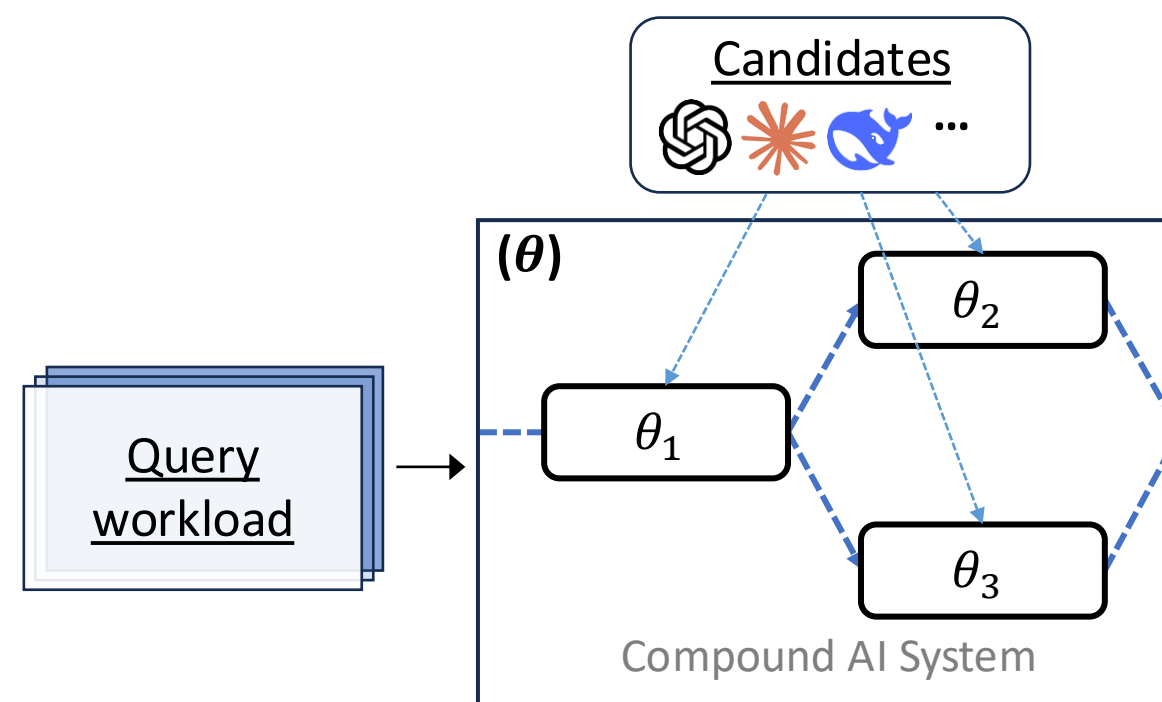


Using trial and error, existing methods are finding better configurations as the next incumbents
(General baselines: Bayesian optimization, bandits)

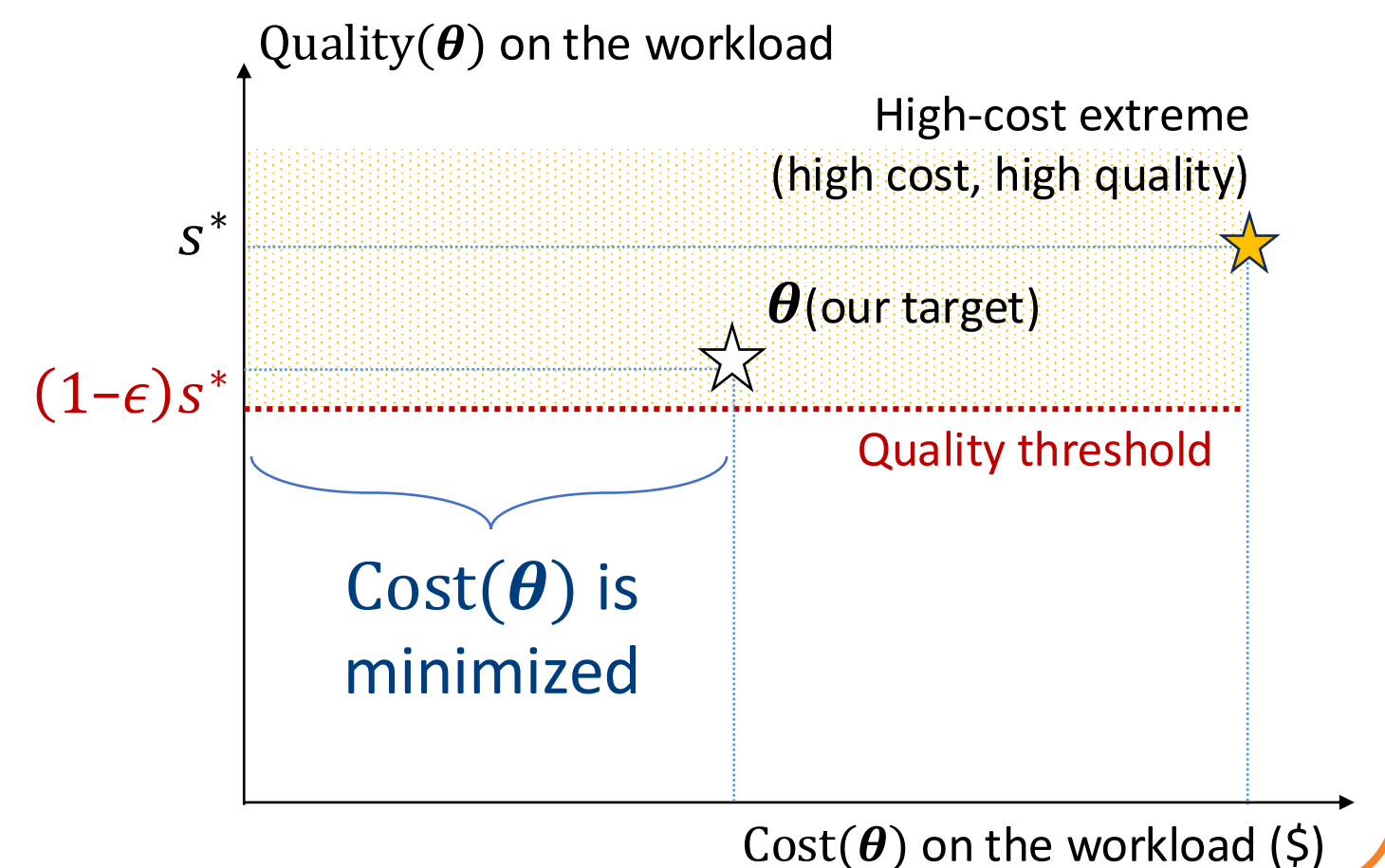
Problem Definition

Setup (left): Compound AI system + Query workload + Candidate LLMs

Input (right, on Y-axis): s^* (quality of High-cost extreme), ϵ (allowable quality degradation)



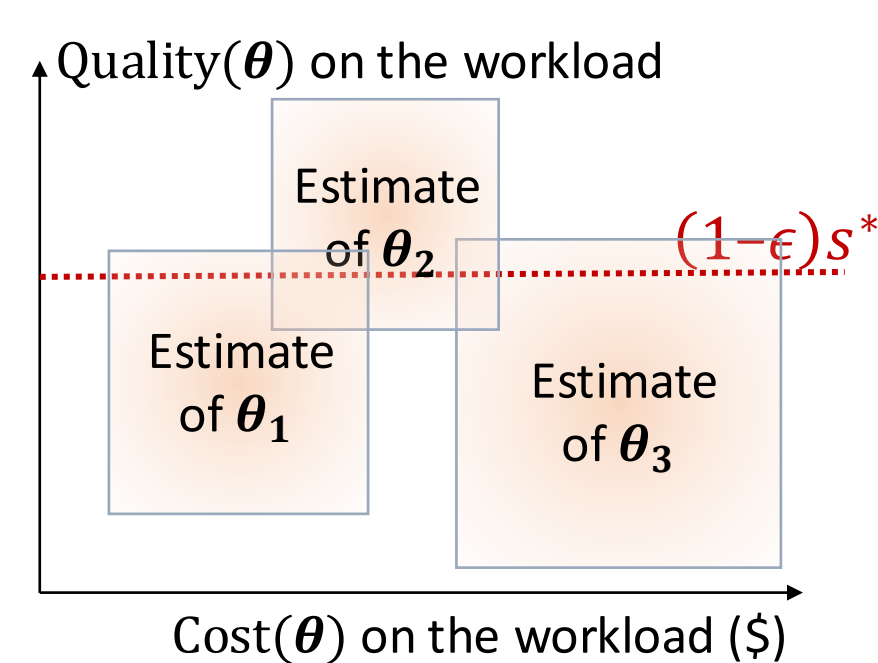
Objective: Minimize $\text{Cost}(\theta)$ subject to $\text{Quality}(\theta) \geq (1-\epsilon)s^*$



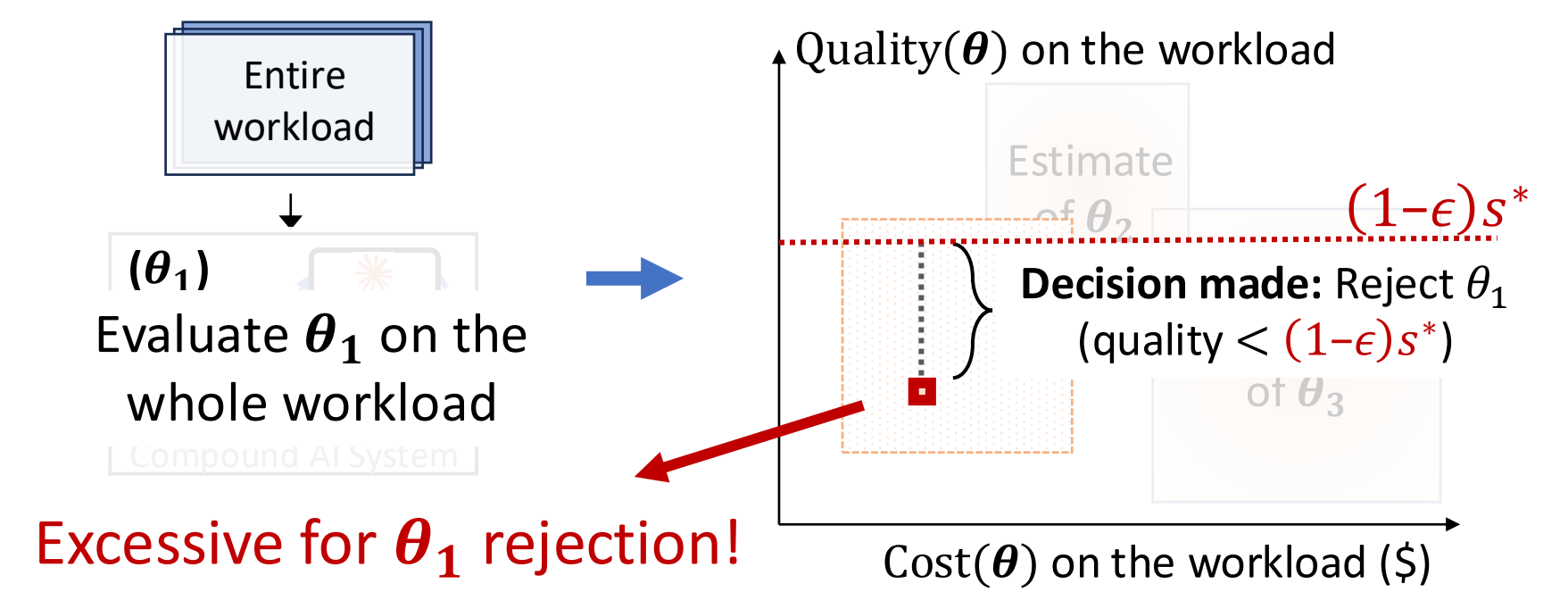
Existing Solutions: Bandit as an example

(Bandit methods keep estimates for configurations)

Left: Cost- and quality- estimate ranges are visualized as squares

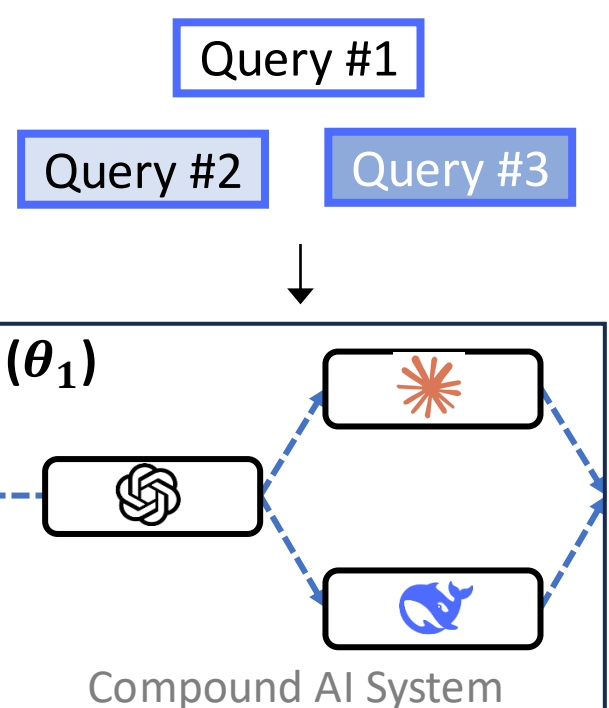


Right: It evaluates θ_1 to get its cost and quality precisely for *the whole workload*, making the estimate of θ_1 shrink to a point
Issue: it wastes money by unnecessarily shrinking squares



SCOPE: Which configuration and query to try next

SCOPE's motivation: chooses queries to shrink the square **most quickly**

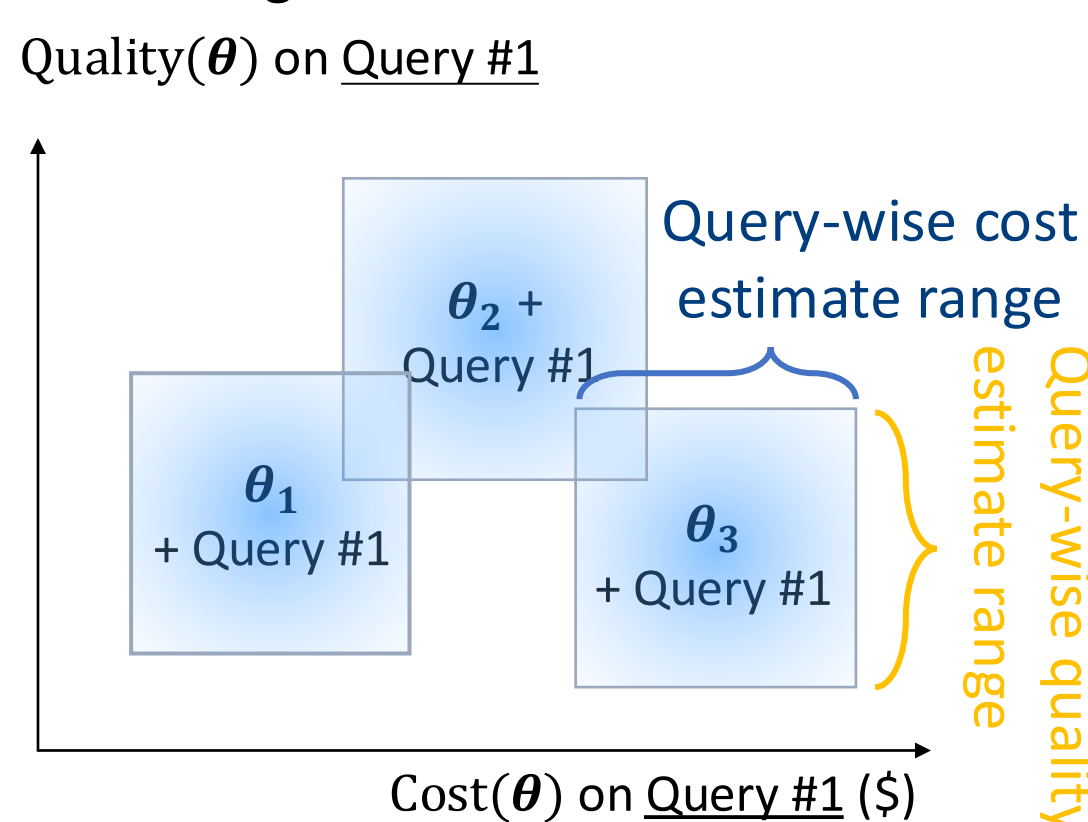


(Recall: Existing methods can only choose θ_1)

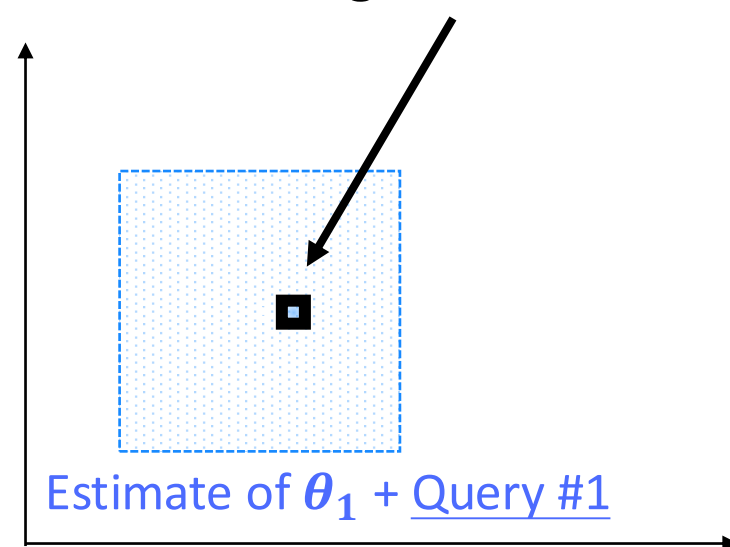
Pseudocode of SCOPE

- For (trials $t = 1, 2, \dots$ while budget remains):
- Select a configuration $\theta^{(t)}$
- While NOT (all queries are evaluated):
- Select the query q with the largest side length
- Evaluate $\theta^{(t)}$ on query q
- Break while when either stopping rule is met

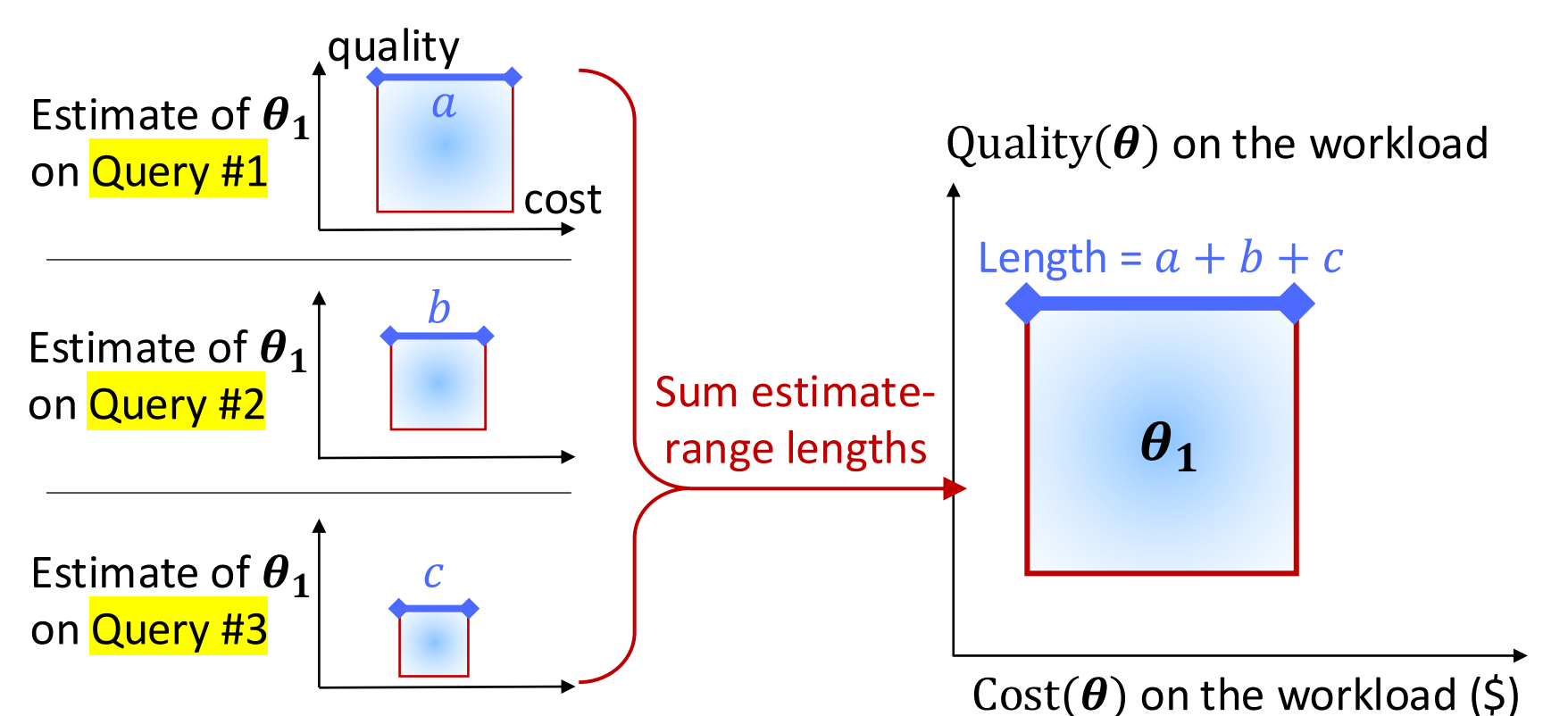
1. SCOPE keeps estimates for each query and configuration



E.g., after SCOPE evaluates θ_1 on Query #1, estimate ranges shrink to a point

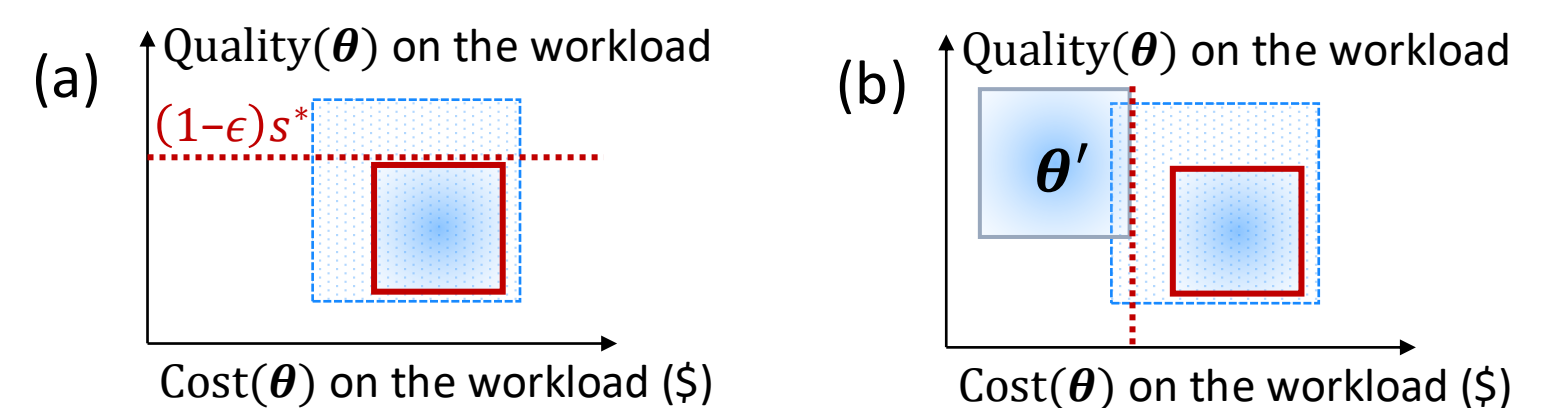


2. SCOPE sums query estimate ranges into workload estimate ranges



3. SCOPE *greedily* evaluates the query with the **largest side length**
(Apparently, this is the right way to implement SCOPE's motivation!)

4. SCOPE stops each evaluation (Line 6) based on two stopping rules
a. The quality estimate falls below $(1-\epsilon)s^*$
b. The cost estimate is dominated by the incumbent

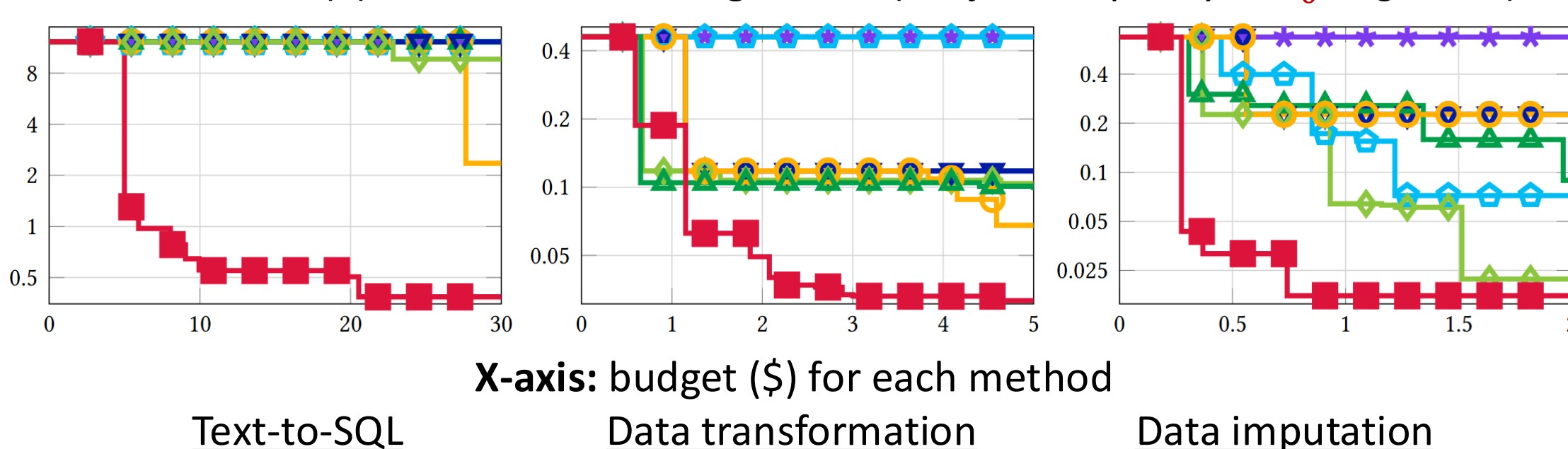


Experiments

3 systems and 23 candidate LLMs ($\Rightarrow$ up to $\sim 6,400,000$ possible configurations)
(We compare how fast a curve drops below)

SCOPE cEI CONFIG LLAMBO Abacus LLMSelector SafeOpt

Y-axis: cost (\$) of best found configuration (subject to quality $\geq s_0$; log-scale)



Theoretical Results

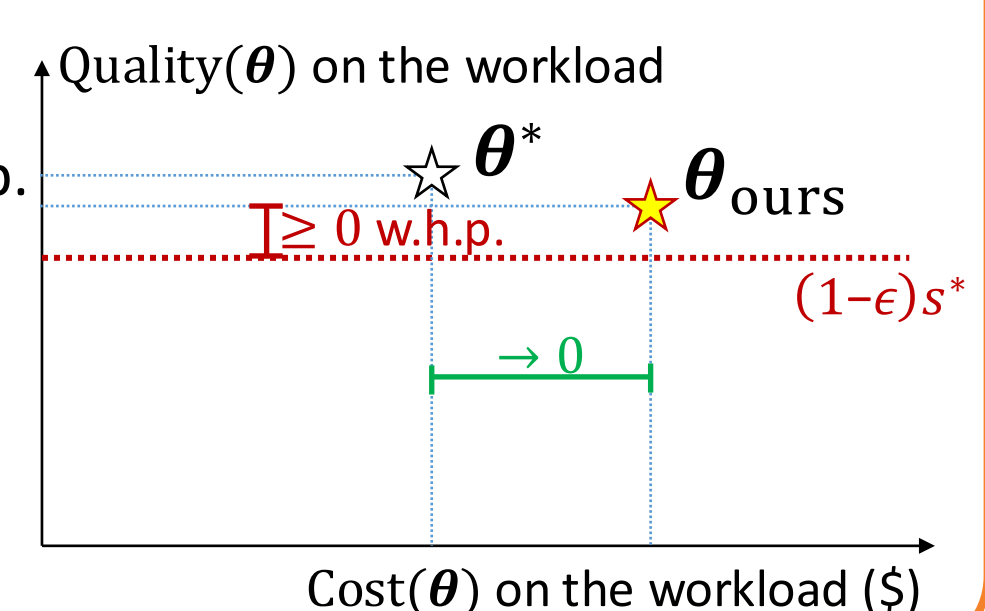
Let θ^* be the *best feasible* configuration at the lowest cost:

$$\theta^* \in \arg \min_{\theta: \text{Quality}(\theta) \geq (1-\epsilon)s^*} \text{Cost}(\theta)$$

SCOPE's output (θ_{ours}) satisfies:

- $\text{Quality}(\theta_{\text{ours}}) \geq (1-\epsilon)s^*$ w.h.p.
- $|\text{Cost}(\theta_{\text{ours}}) - \text{Cost}(\theta^*)| \rightarrow 0$ when we increase # of trials

(Both hold even when evaluations include noise)



Code: github.com/waetr/SCOPELLM-optimizer/
Email: yiqian@comp.nus.edu.sg