SCOPE: Cost-Efficient Model Selection for Compound AI Systems under Quality Constraints

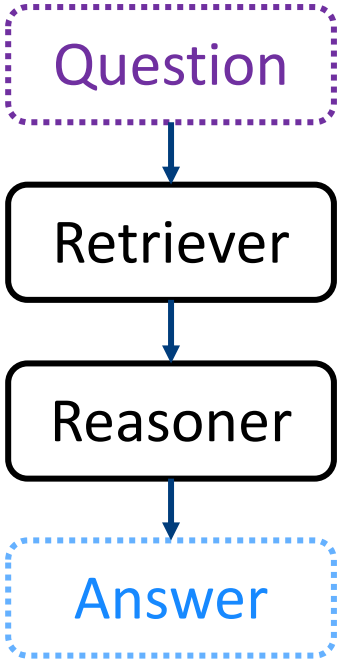
Yiqian Huang[1], Shiqi Zhang[1], Tianyuan Jin[2], Xiaokui Xiao[1]

[1] NUS National University of Singapore

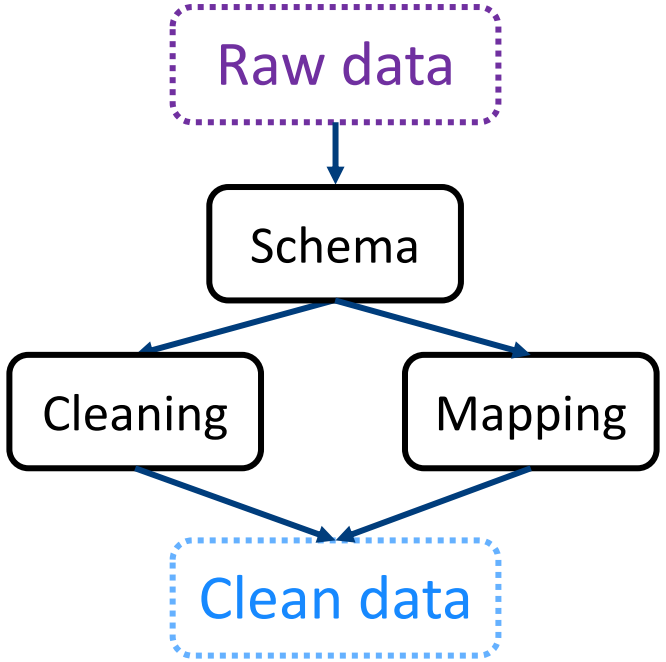
[2] 香港科技大学(广州) THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY (GUANGZHOU)

Compound AI Systems

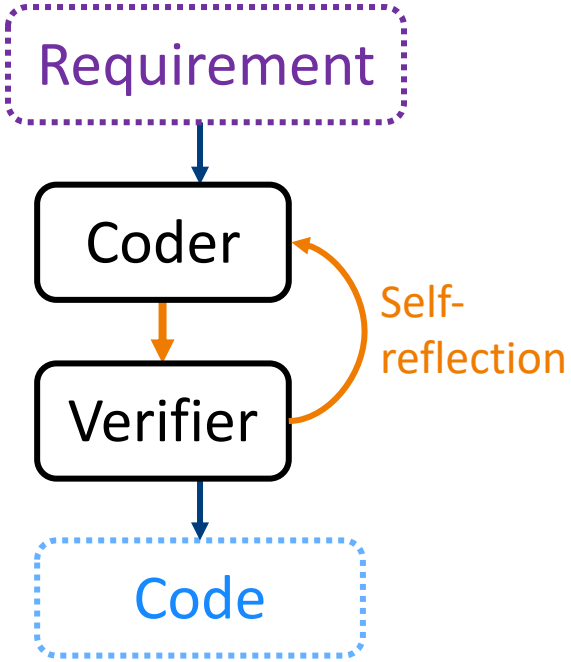
- A compound AI system is a system made up of multiple AI agents



Question answering



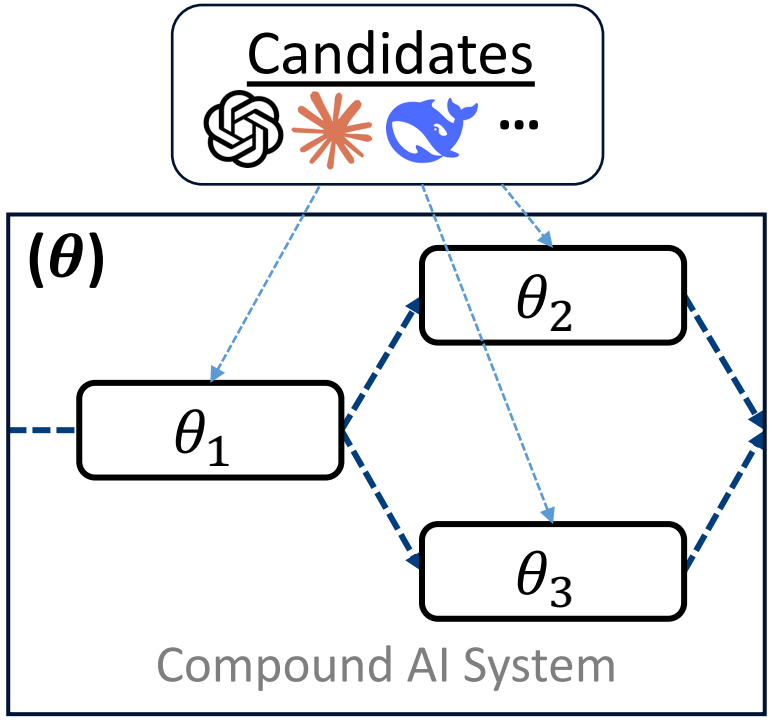
Data processing



Code development

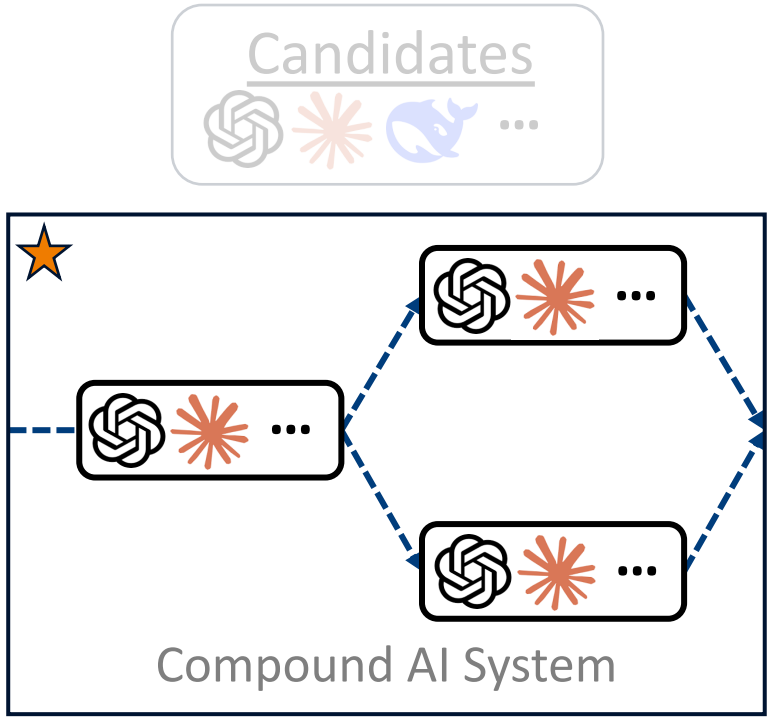
Model Selection

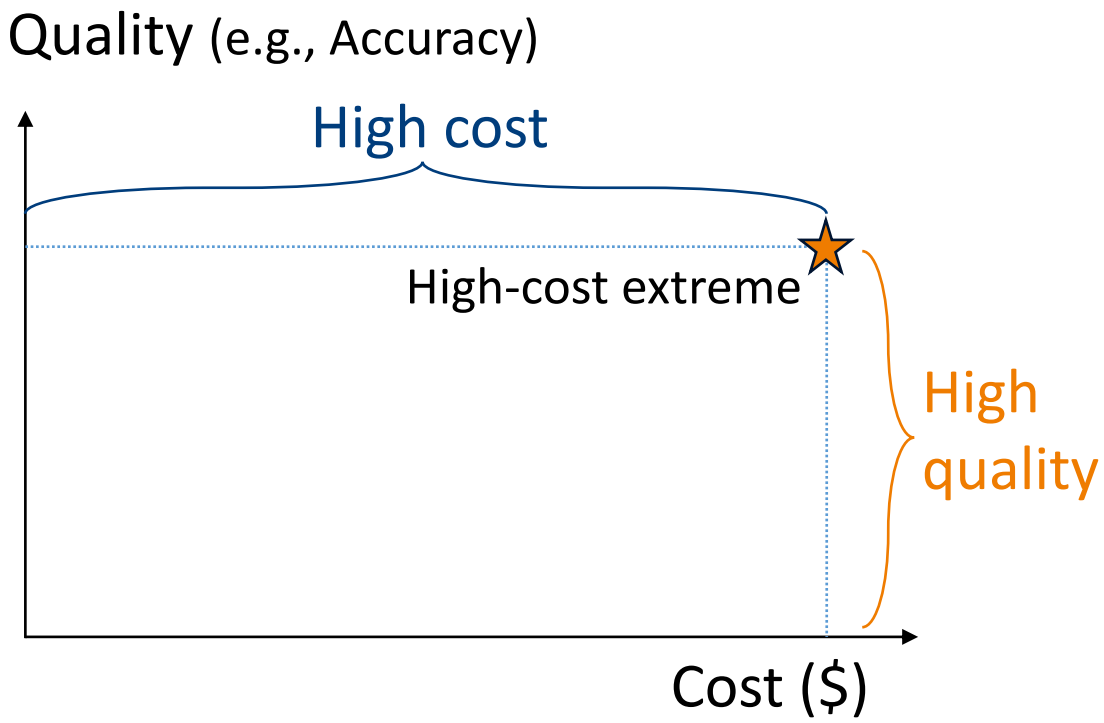
- To run the system, we should decide which models to use



Model Selection

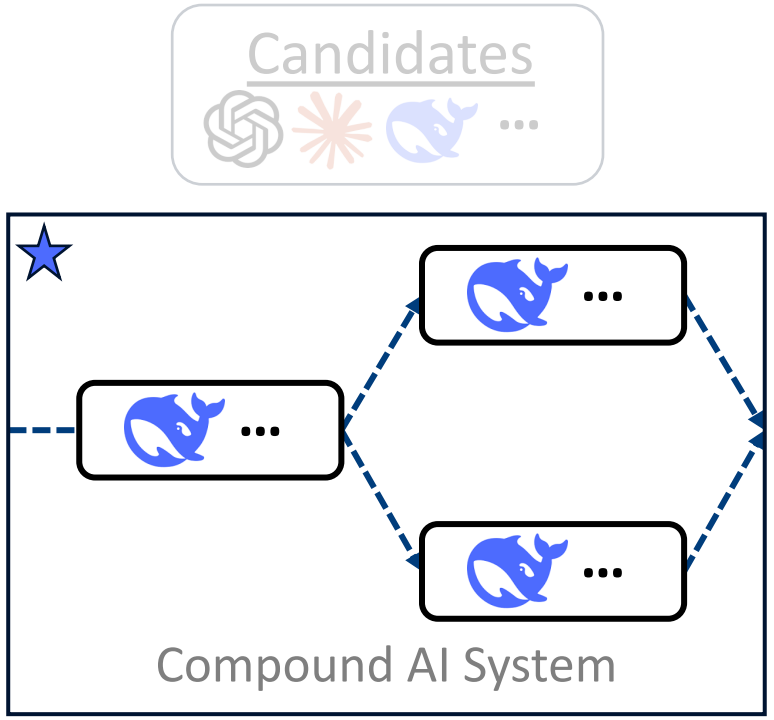
- We may use **High-cost extreme** ★ (the most expensive models)

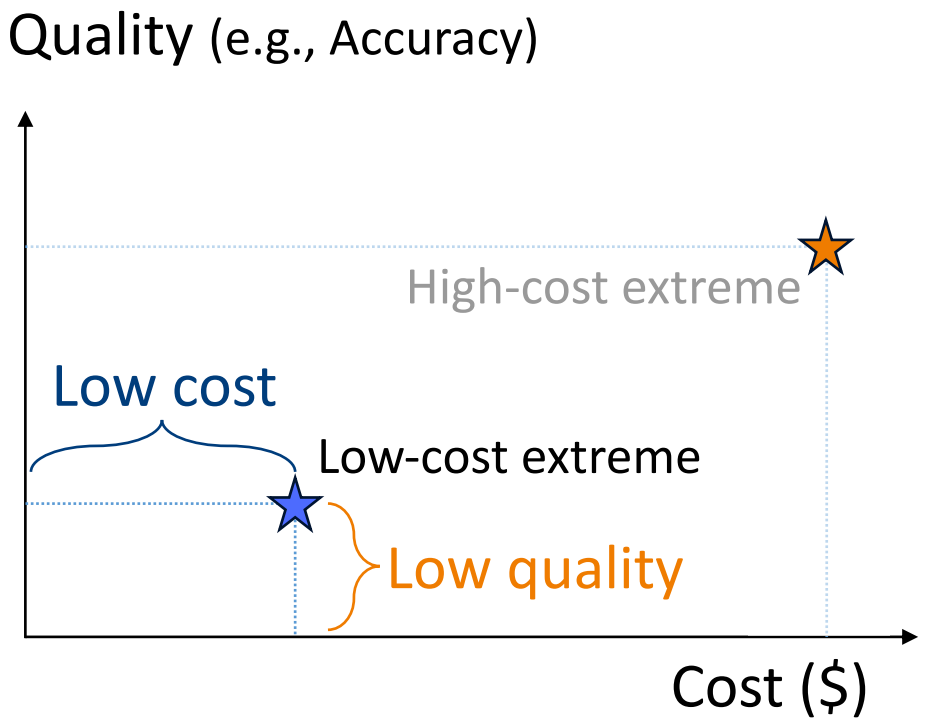




Model Selection

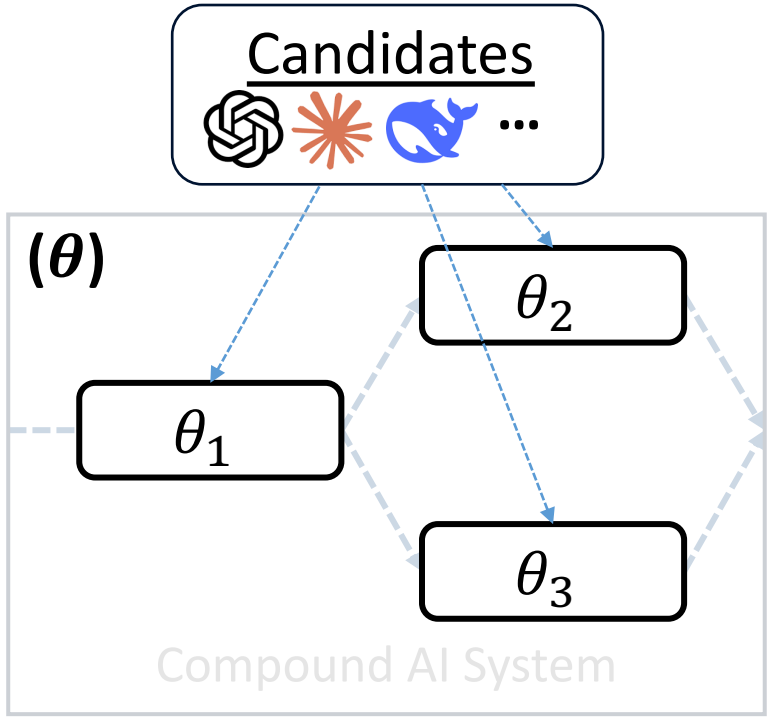
- Or we may use **Low-cost extreme** ★ (the most affordable models)

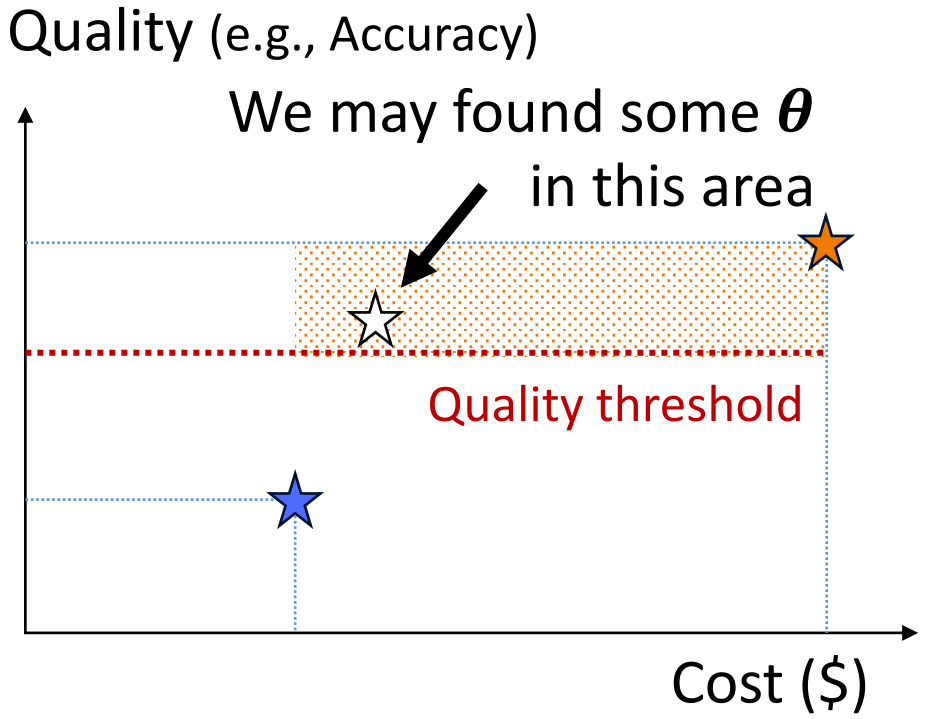




Model Selection

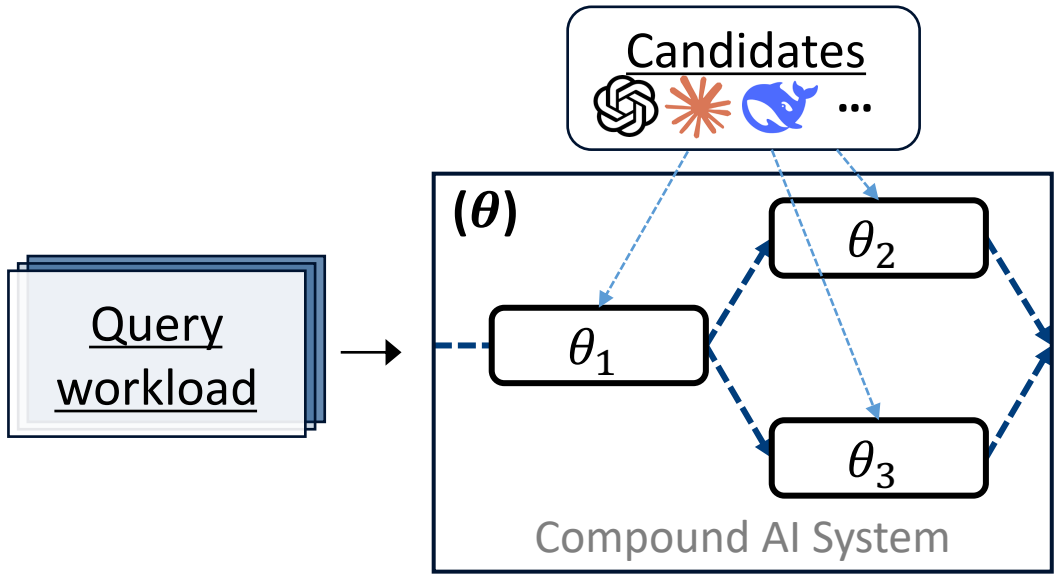
- Can we *preserve most of* the quality from ★ while saving cost?





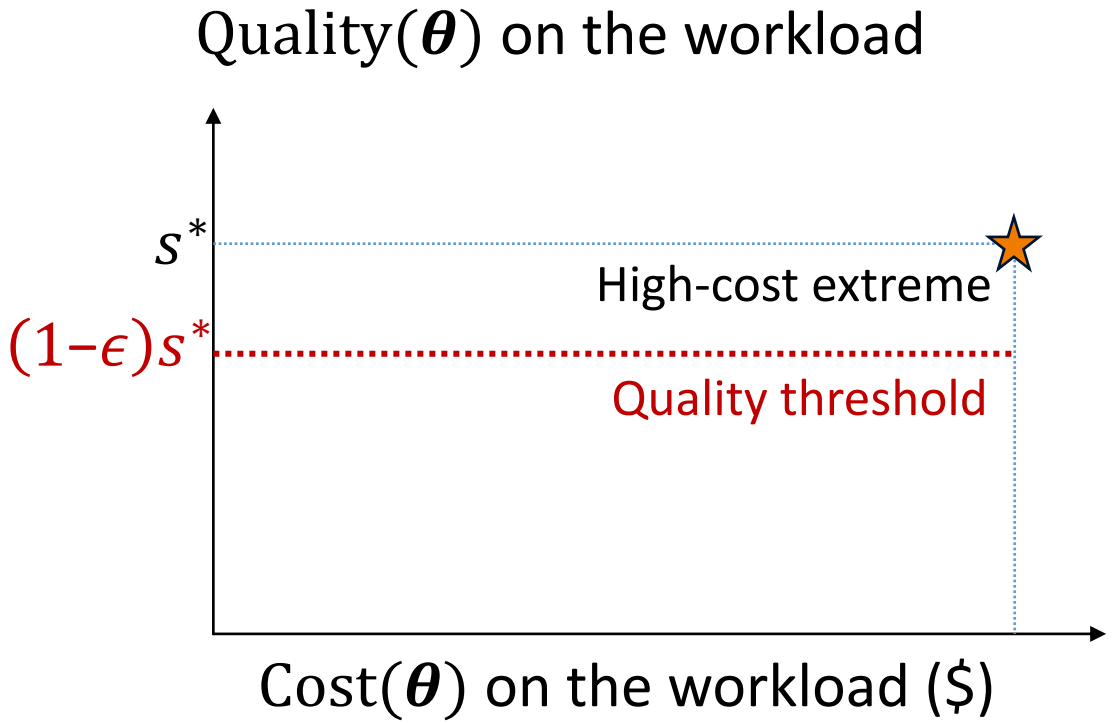
Problem Definition

- **Problem setup**: Compound AI system + Query workload + Candidate LLMs



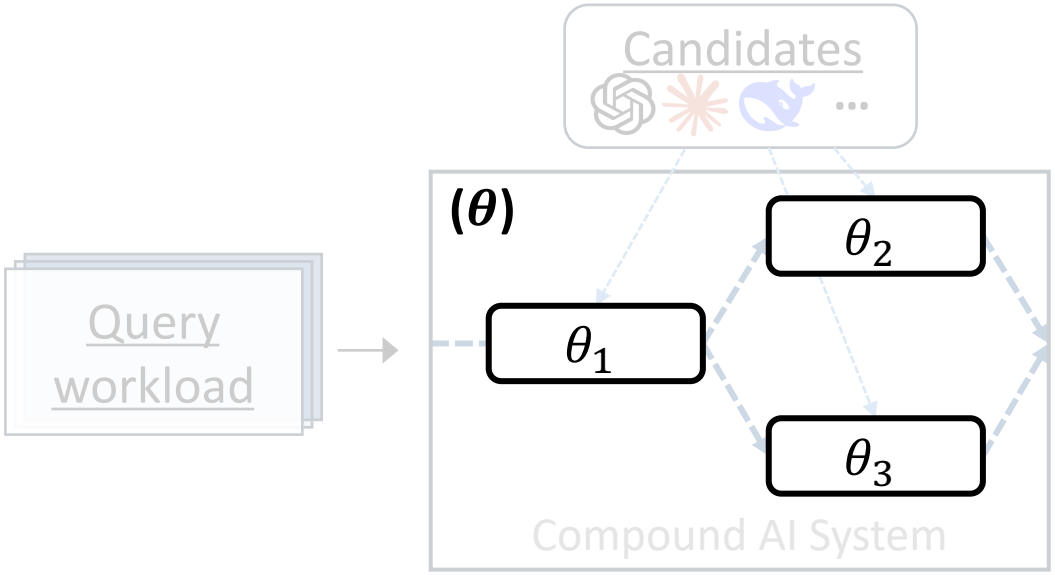
Problem Definition

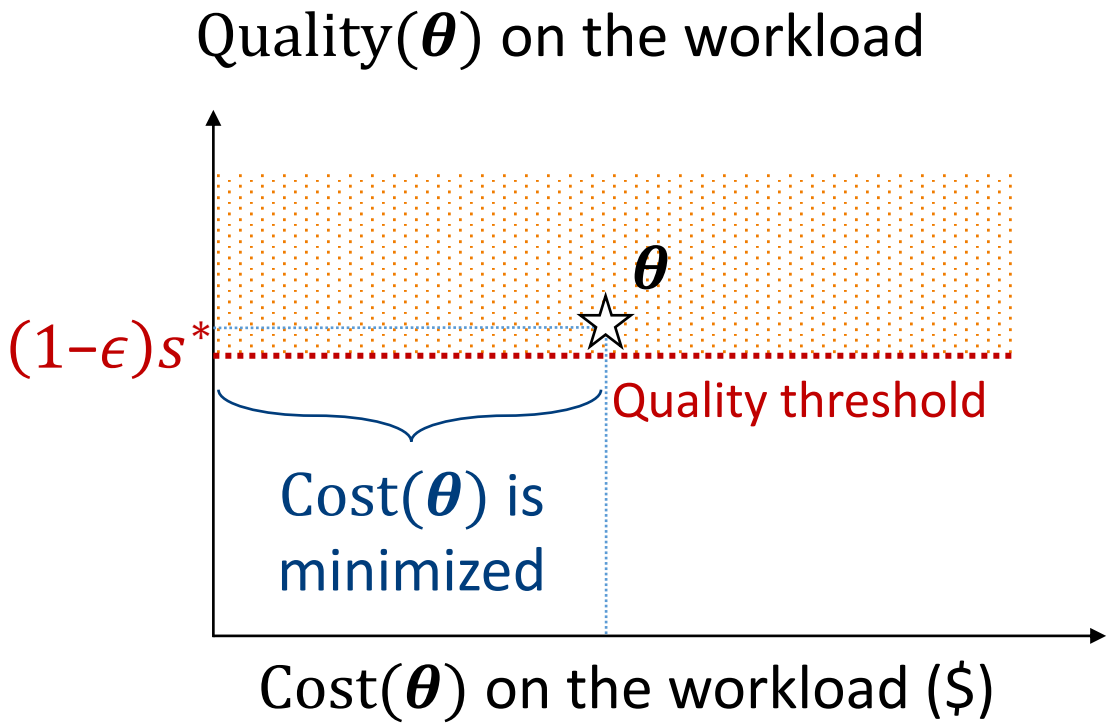
- **Problem setup**: Compound AI system + Query workload + Candidate LLMs
- **Input:** s^* (quality of high-cost extreme), ϵ (allowable quality degradation)



Problem Definition

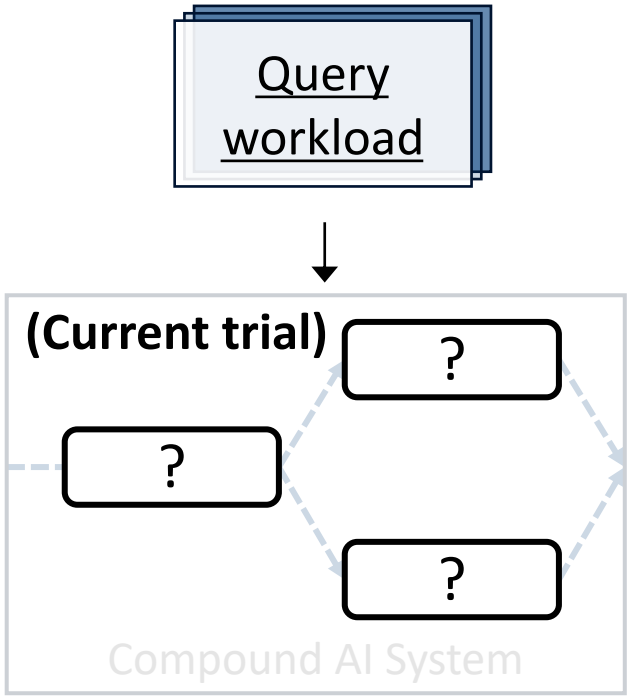
- **Objective:** Minimize Cost($\boldsymbol{\theta}$) subject to Quality($\boldsymbol{\theta}$) $\geq (1-\epsilon)s^*$

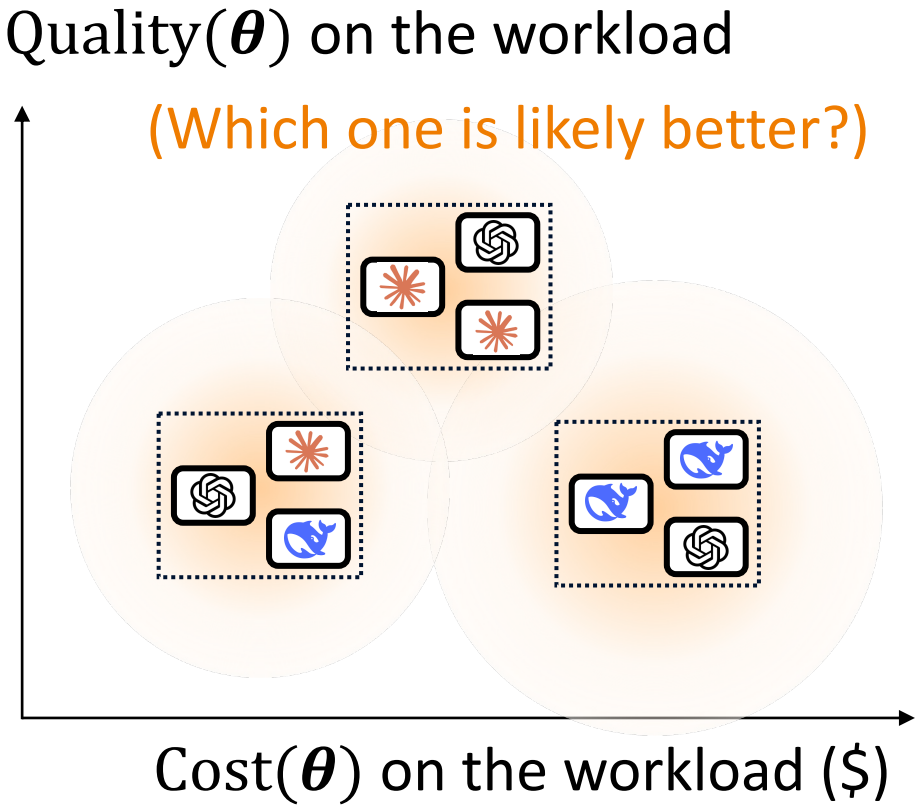




Existing Solutions

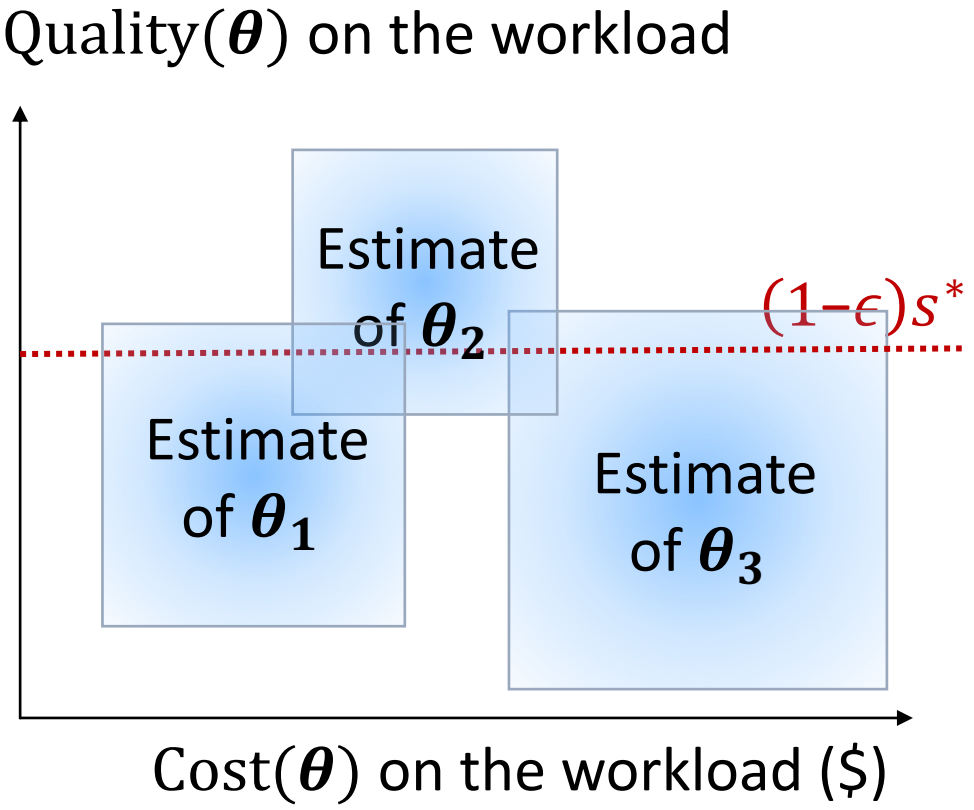
- Existing solutions use *the whole workload* to choose the next configuration





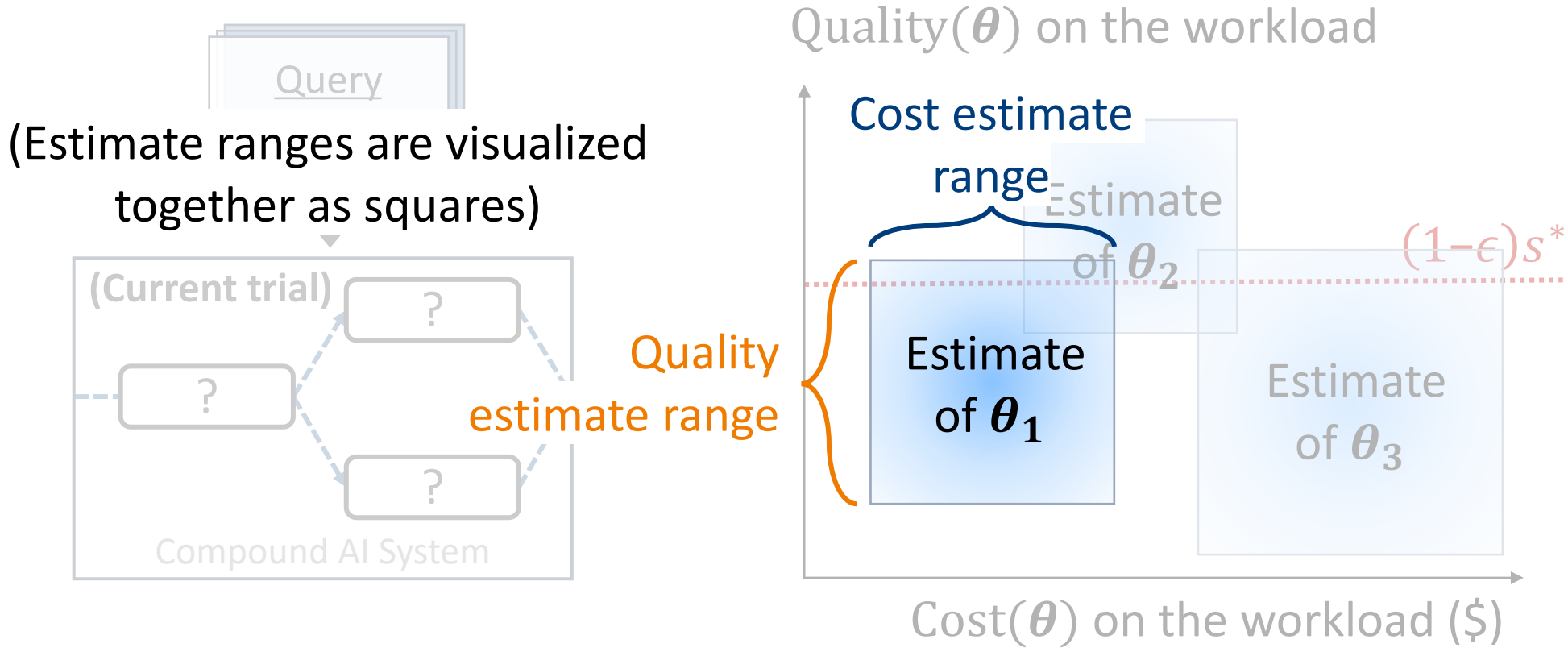
Existing Solutions (Bandit as an example)

- Example: Bandit methods keep estimates for configurations



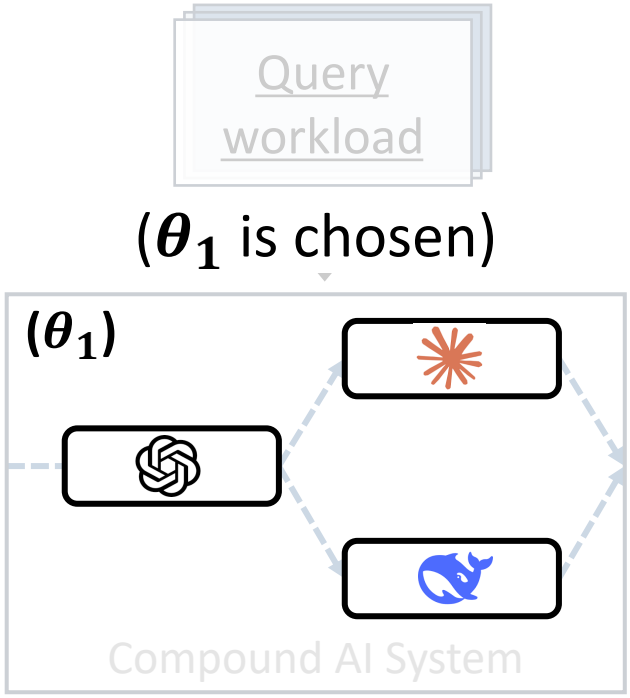
Existing Solutions (Bandit as an example)

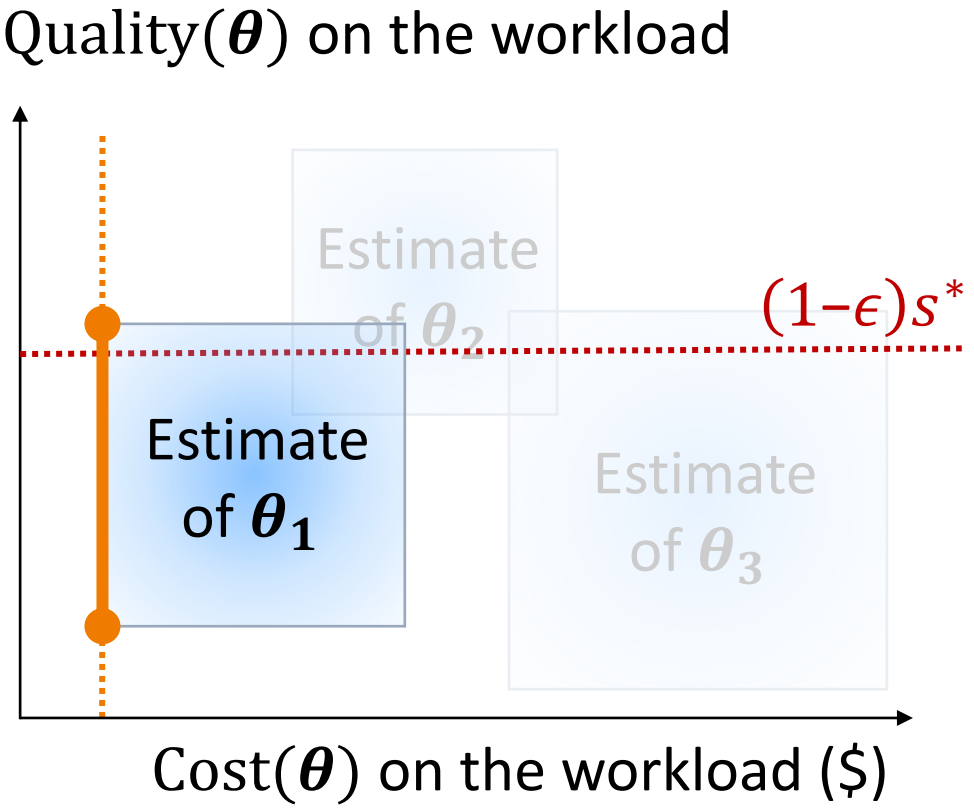
- Example: Bandit methods keep estimates for configurations



Existing Solutions (Bandit as an example)

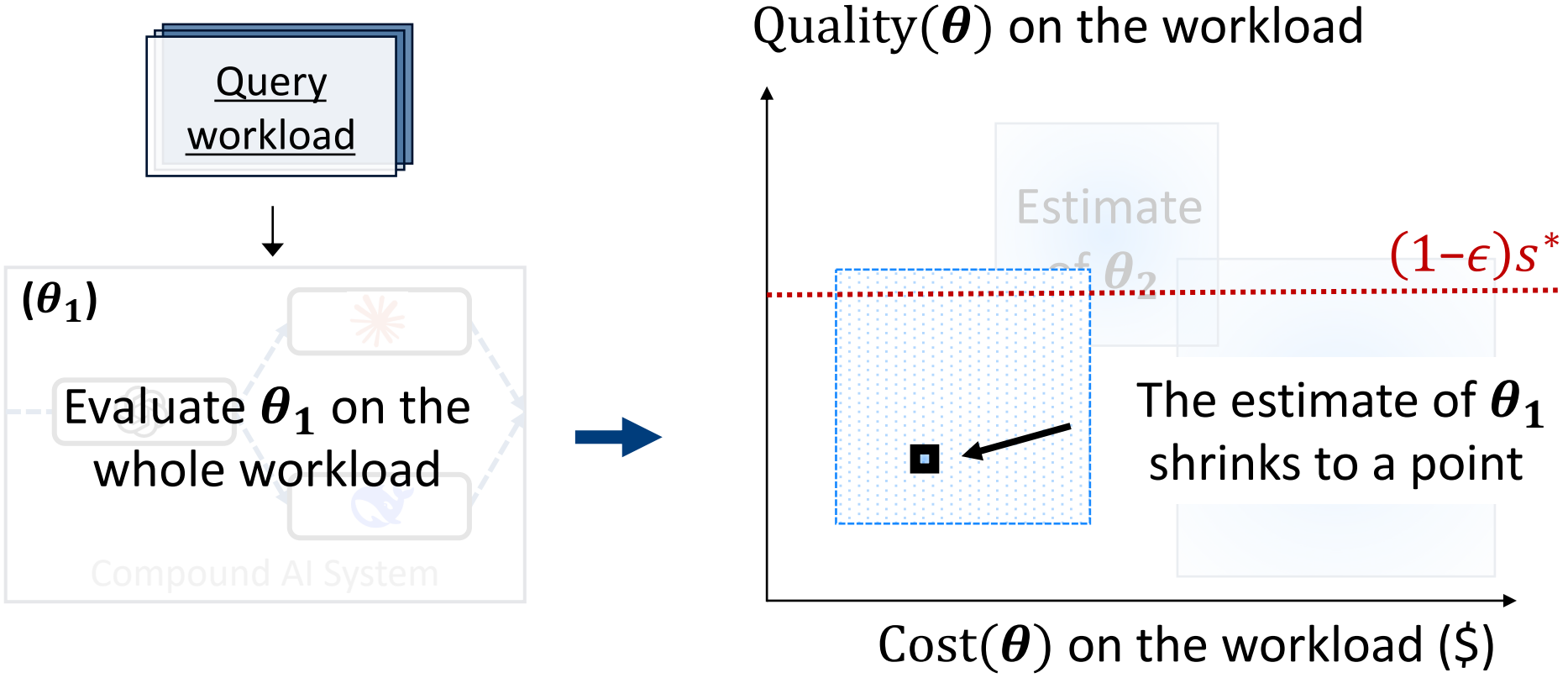
- It chooses one with the *best cost estimate* (the leftmost edge)





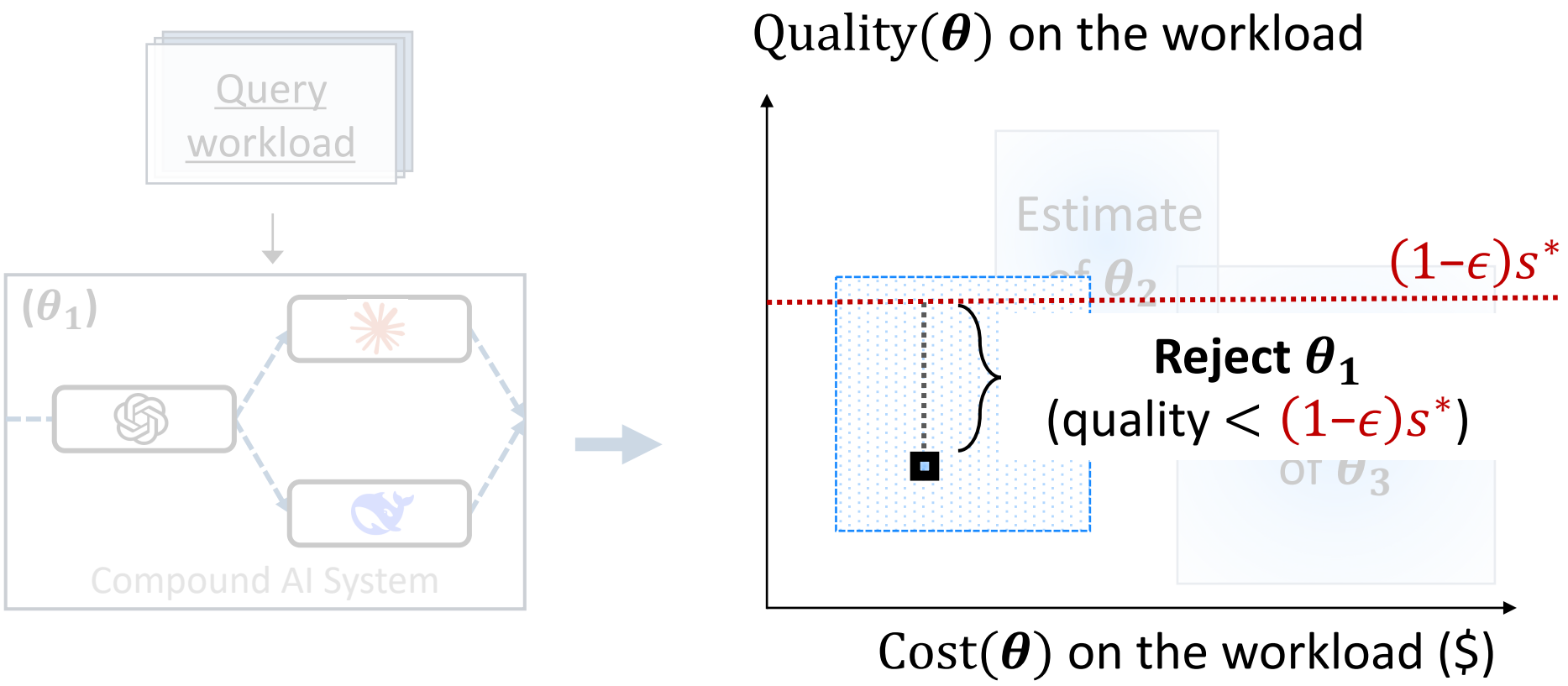
Existing Solutions (Bandit as an example)

- It evaluates $\boldsymbol{\theta_1}$ to get its cost and quality precisely *for the whole workload*



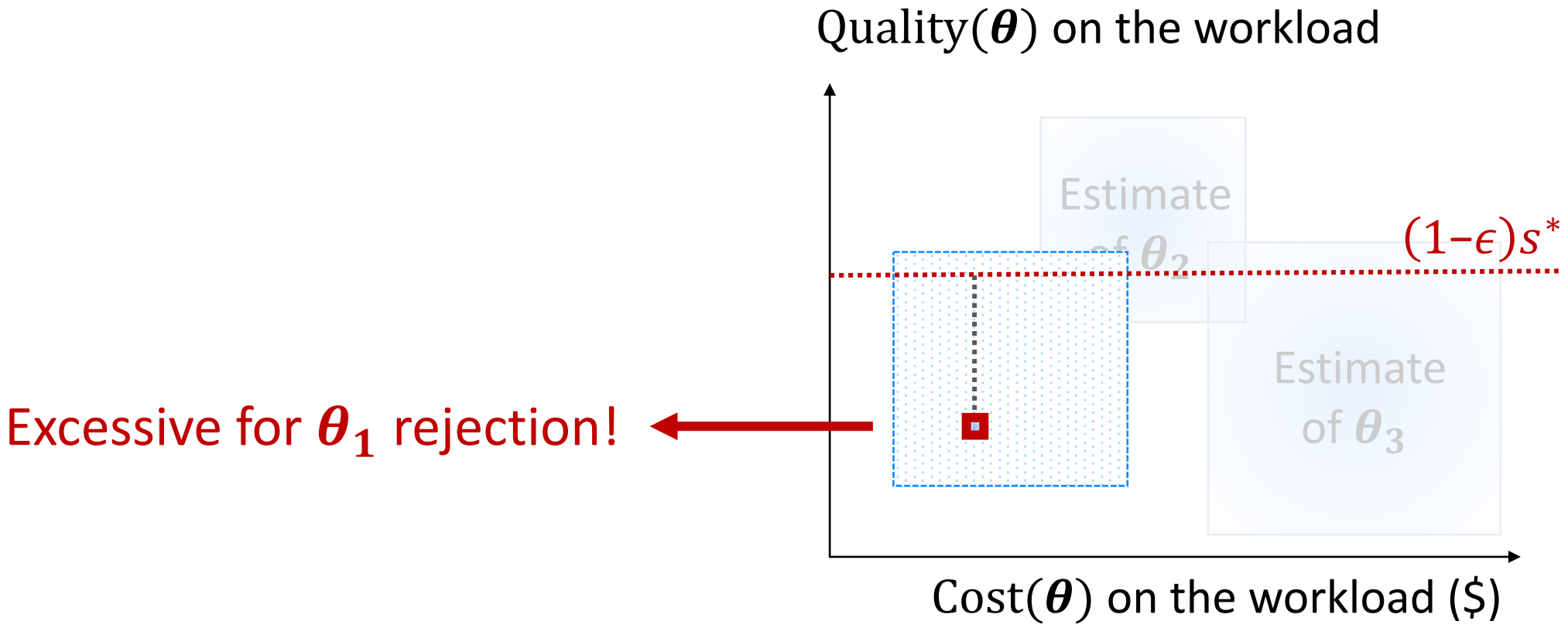
Existing Solutions (Bandit as an example)

- After evaluation, it accepts $\boldsymbol{\theta_1}$ as the next incumbent, or rejects it



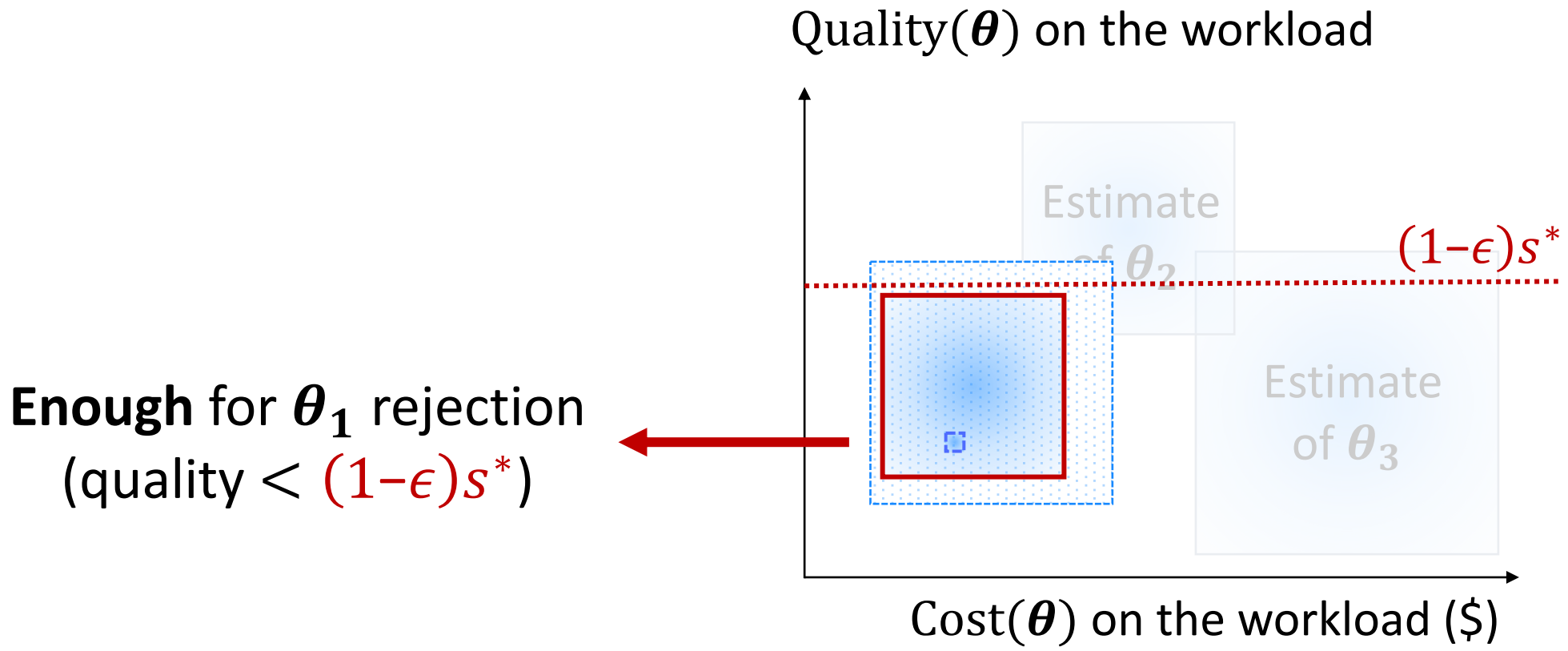
SCOPE (Main idea)

- **Issue:** Existing solutions waste money by <u>unnecessarily</u> shrinking squares



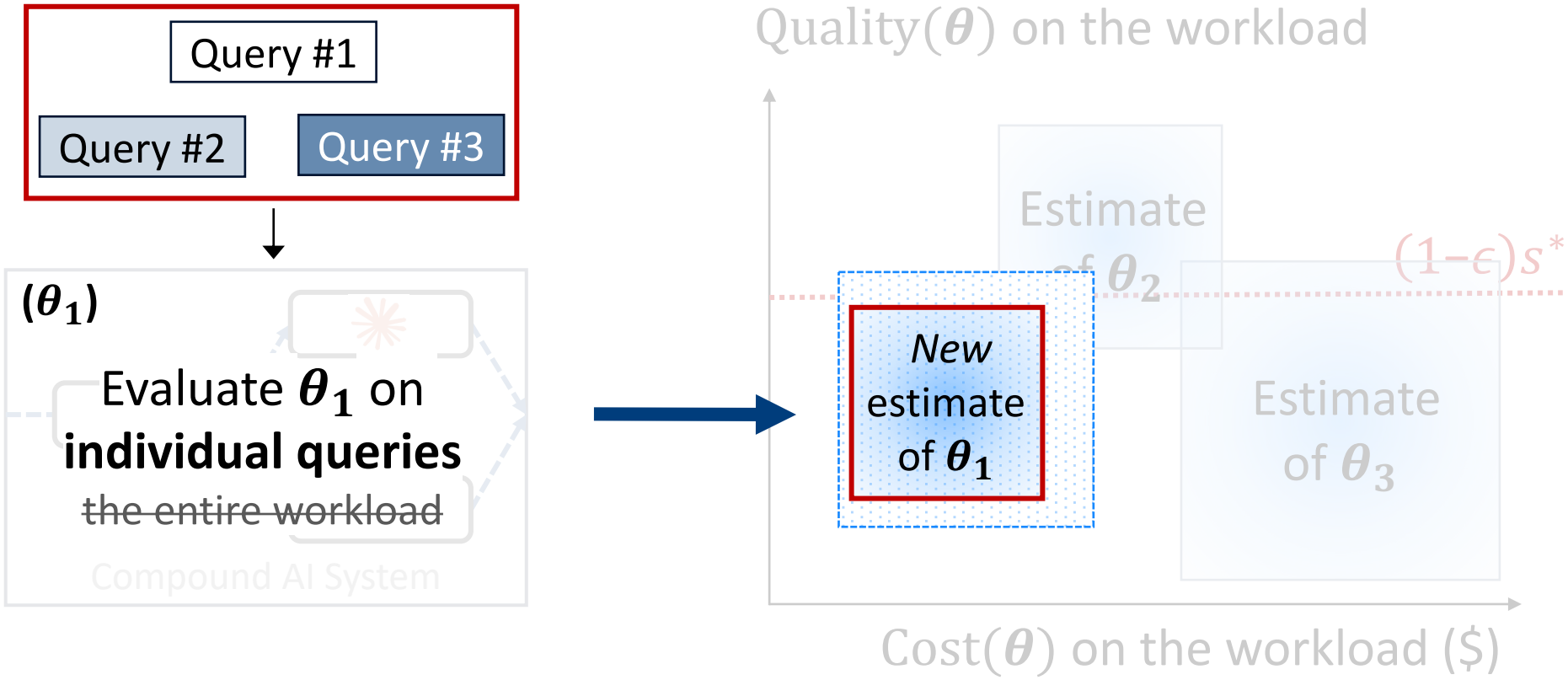
SCOPE (Main idea)

- However, smaller squares are just enough…



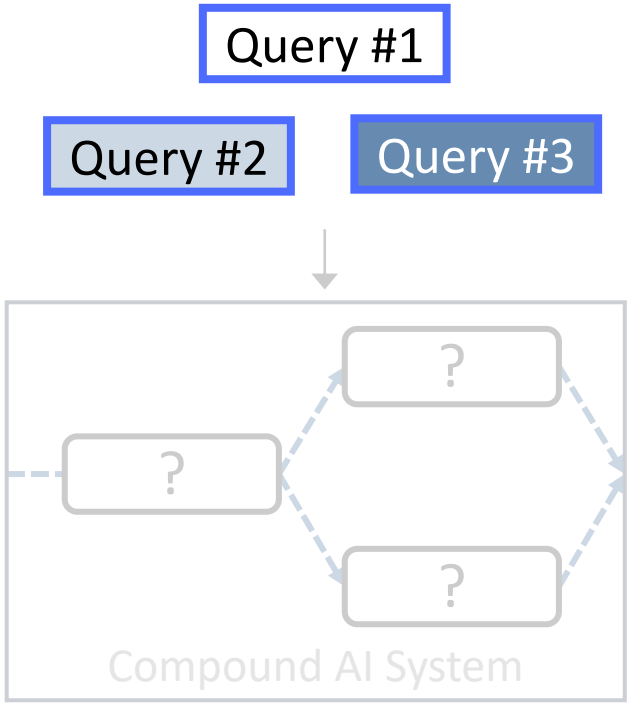
SCOPE (Main idea)

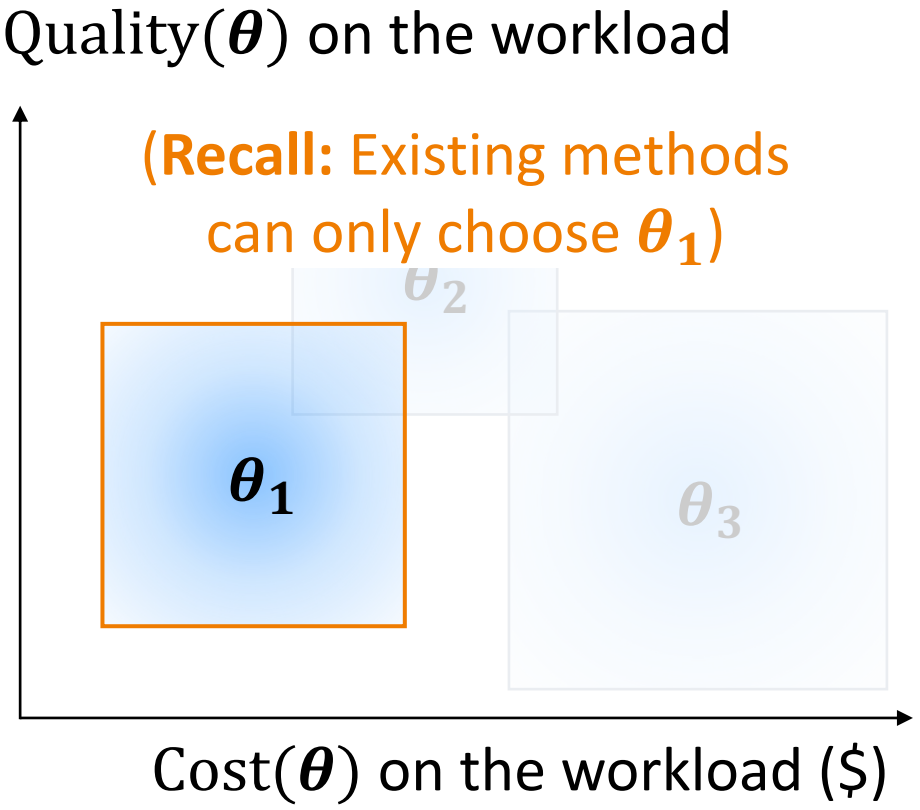
- We get such a smaller square by evaluating the workload *in part* (queries)



SCOPE (Main idea)

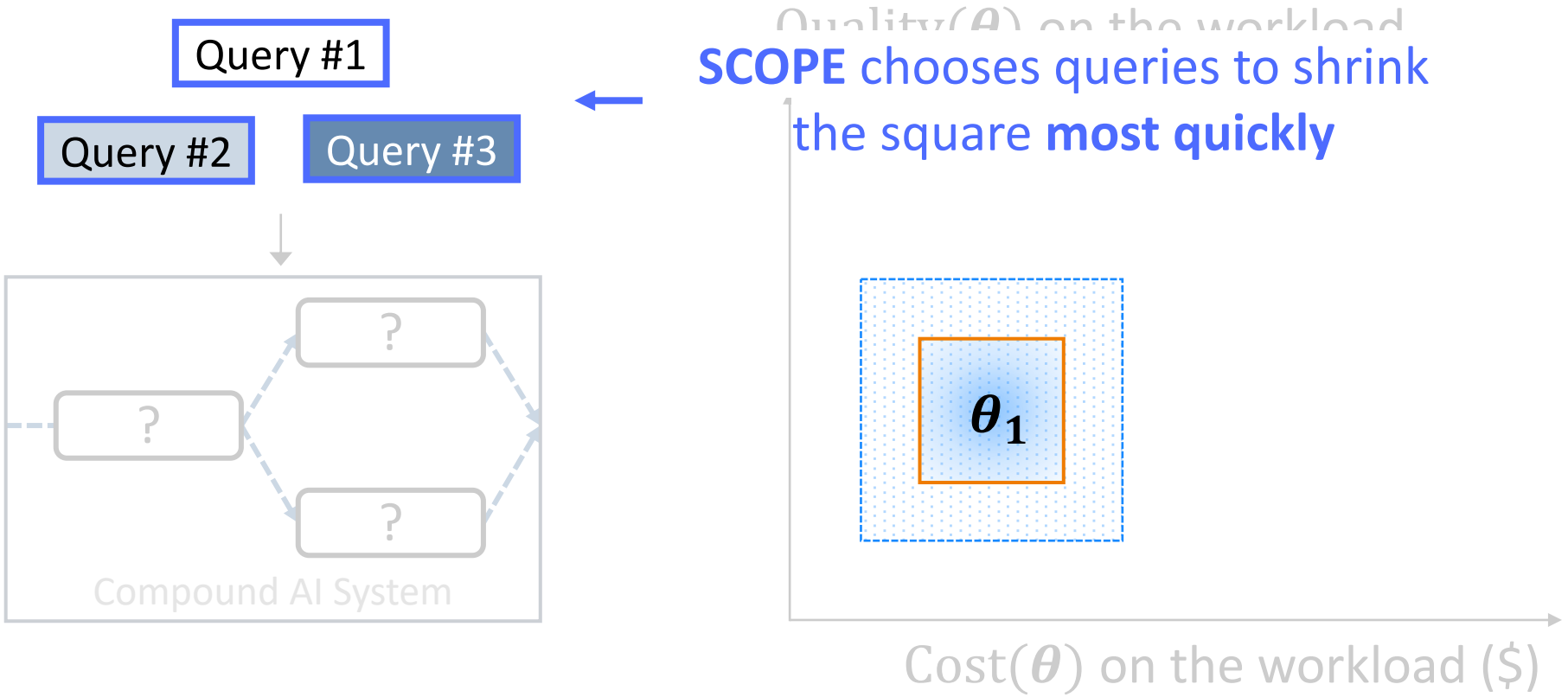
- This is **SCOPE**: it answers *which configuration and query to try next*





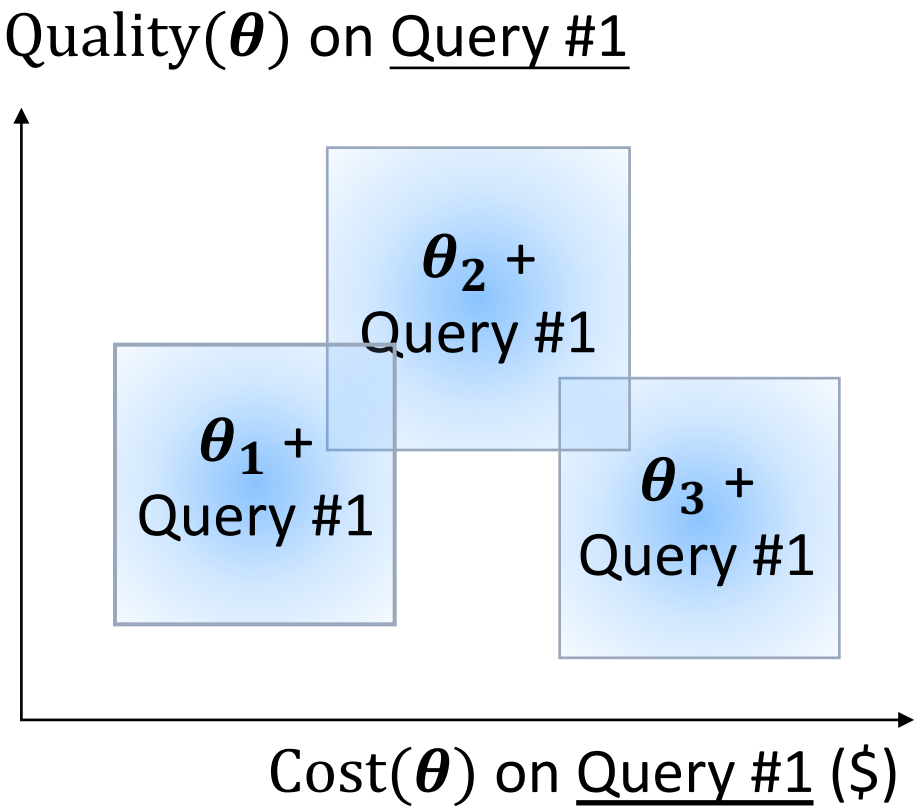
SCOPE (Main idea)

- This is **SCOPE**: it answers *which configuration and query to try next*



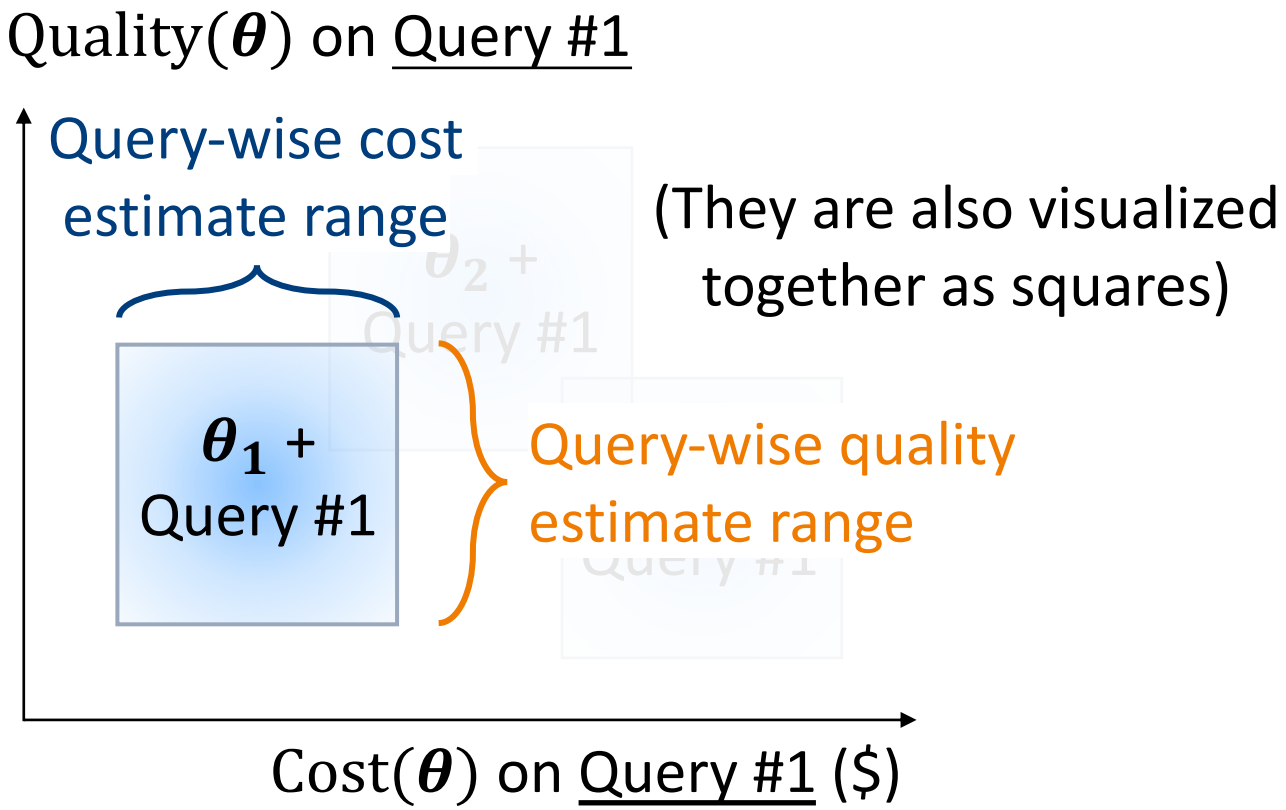
SCOPE (in more detail)

- **Q:** How can we shrink the square most quickly?
- **A:** SCOPE keeps estimates for each query and configuration



SCOPE (in more detail)

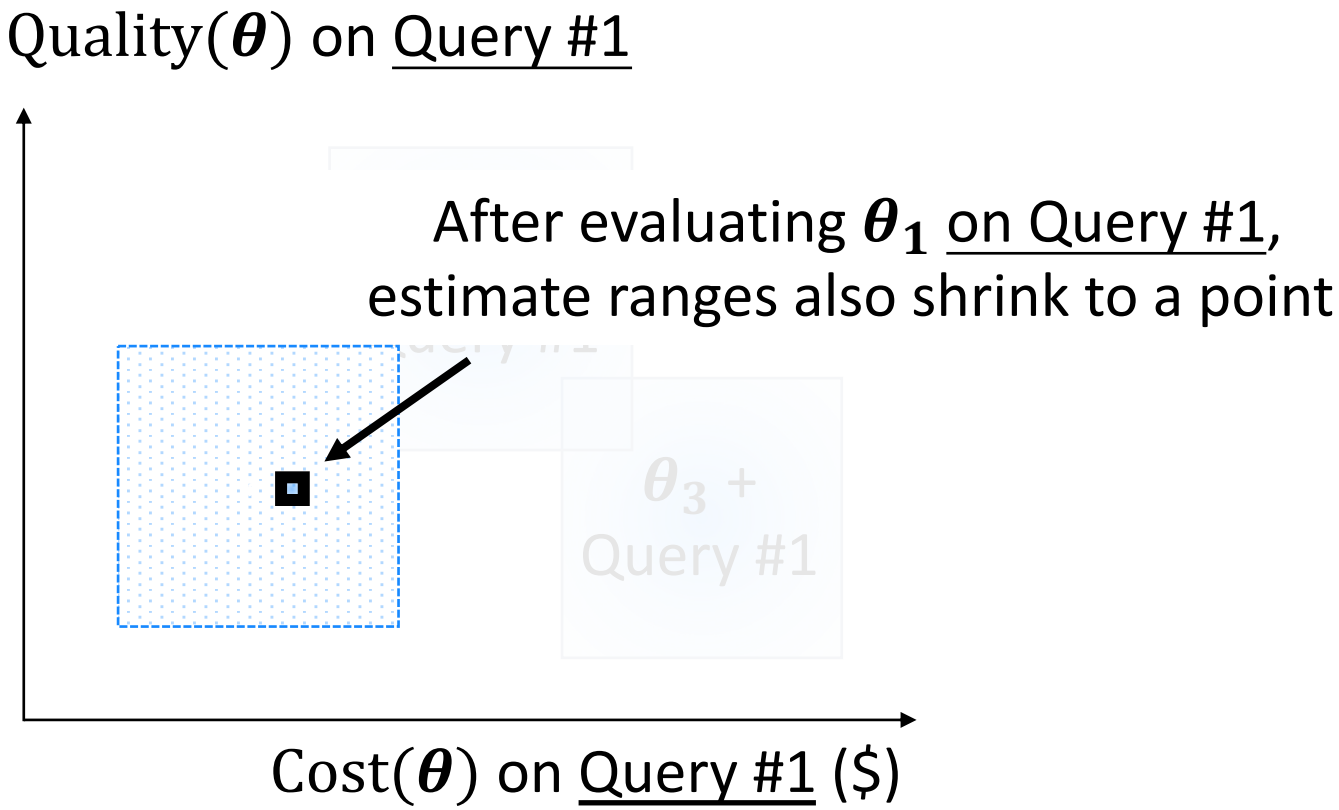
- **Q:** How can we shrink the square most quickly?
- **A:** SCOPE keeps estimates for each query and configuration



(They are also visualized together as squares)

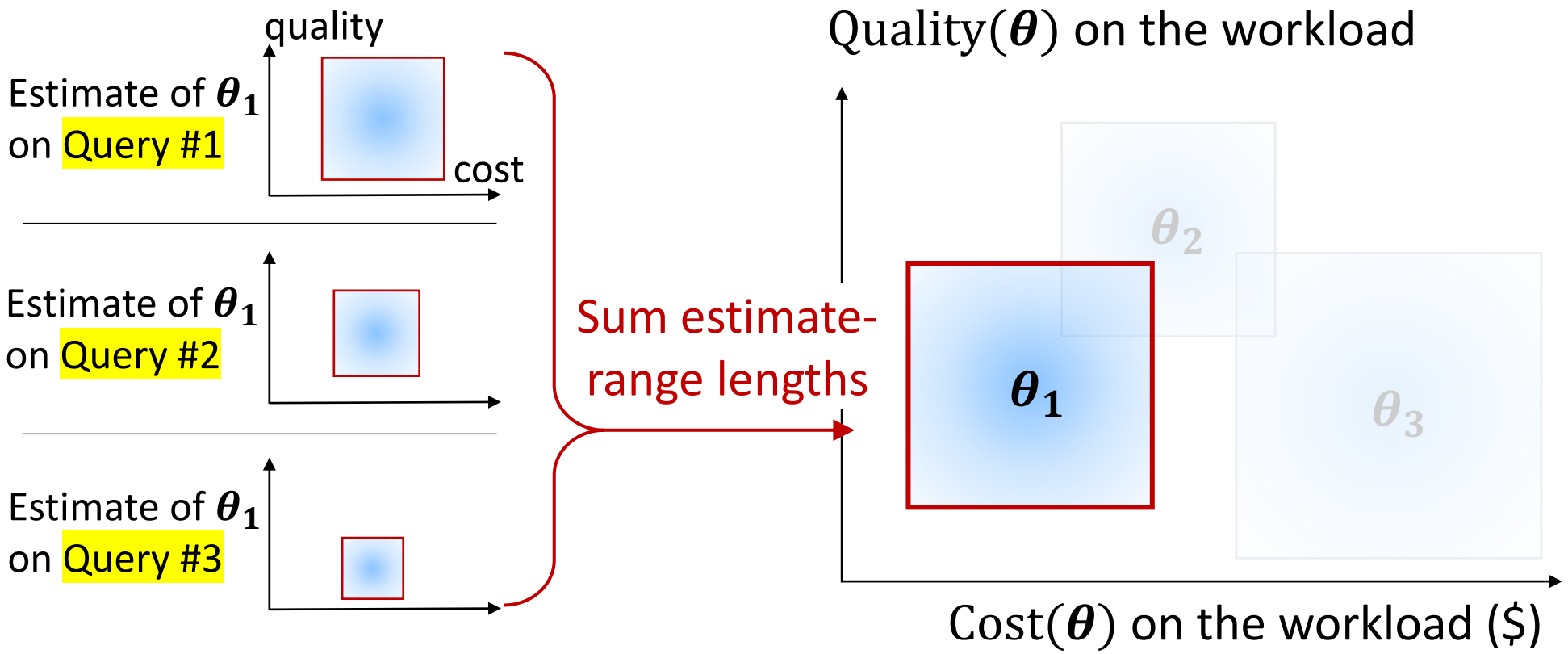
SCOPE (in more detail)

- **Q:** How can we shrink the square most quickly?
- **A:** SCOPE keeps estimates for each query and configuration



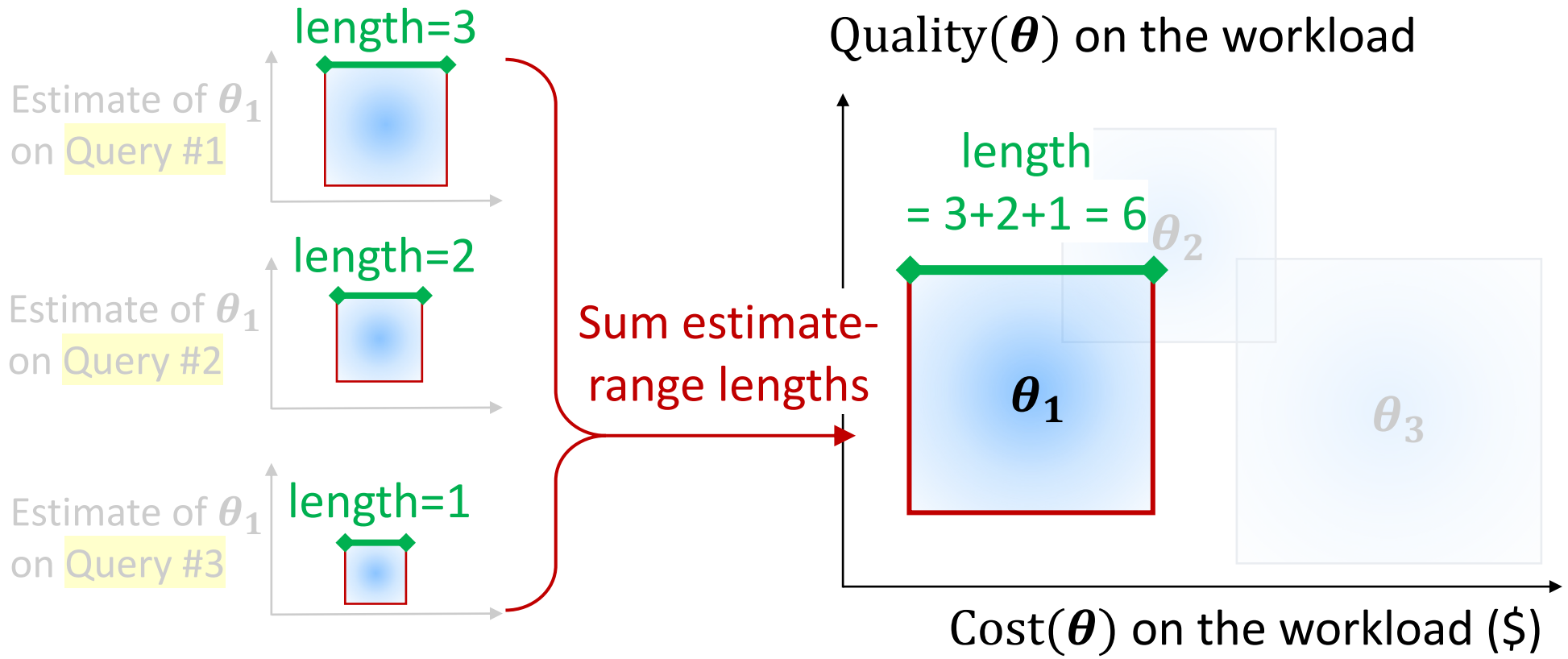
SCOPE (in more detail)

- **Q:** How can we shrink the square most quickly?
- **A:** SCOPE sums query estimate ranges into workload estimate ranges



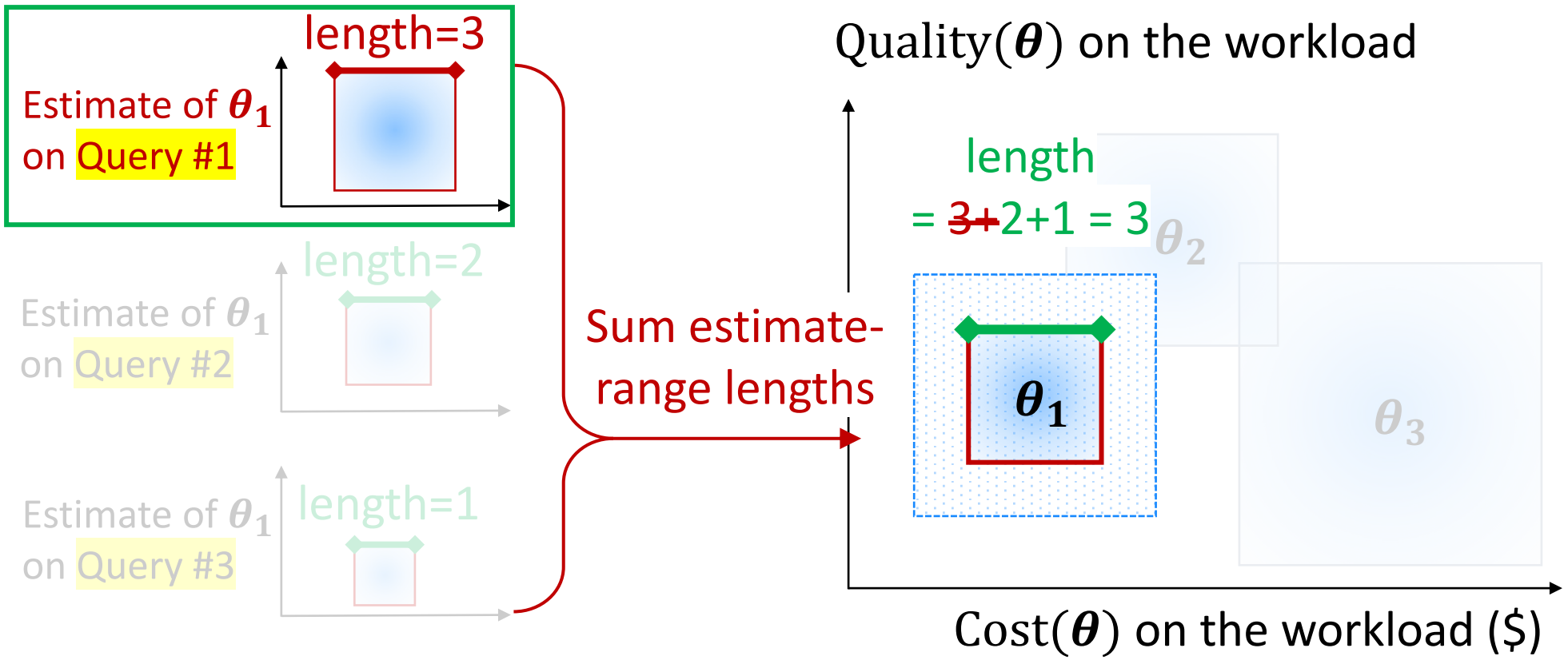
SCOPE (in more detail)

- **Q:** How can we shrink the square most quickly?
- **A:** SCOPE sums query estimate ranges into workload estimate ranges



SCOPE (in more detail)

- **Q:** How can we shrink the square most quickly?
- **A:** Greedily evaluate the query with the largest side length!



SCOPE (in more detail)

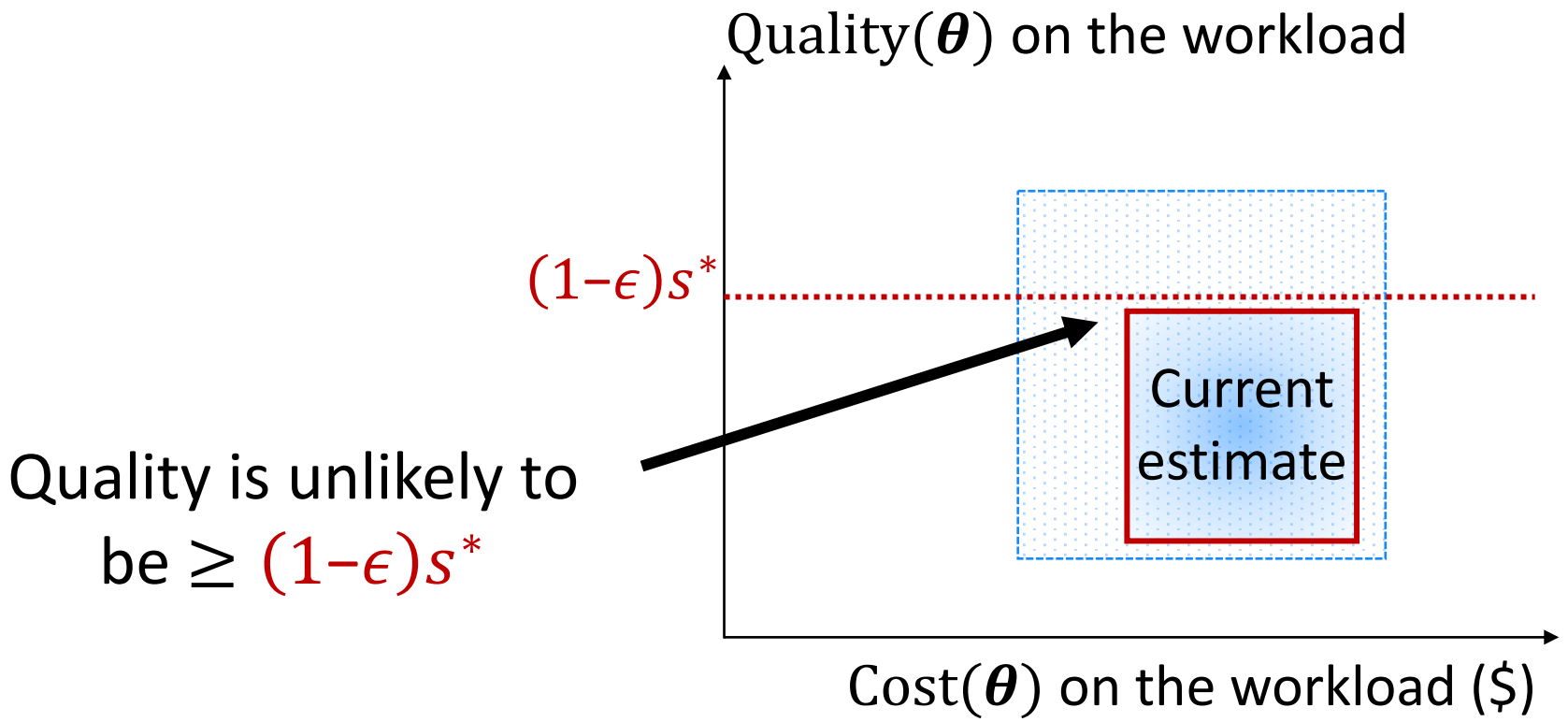
Pseudocode of **SCOPE**

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

SCOPE (in more detail)

SCOPE's two stopping rules

1. The quality estimate falls below $(1-\epsilon)s^*$

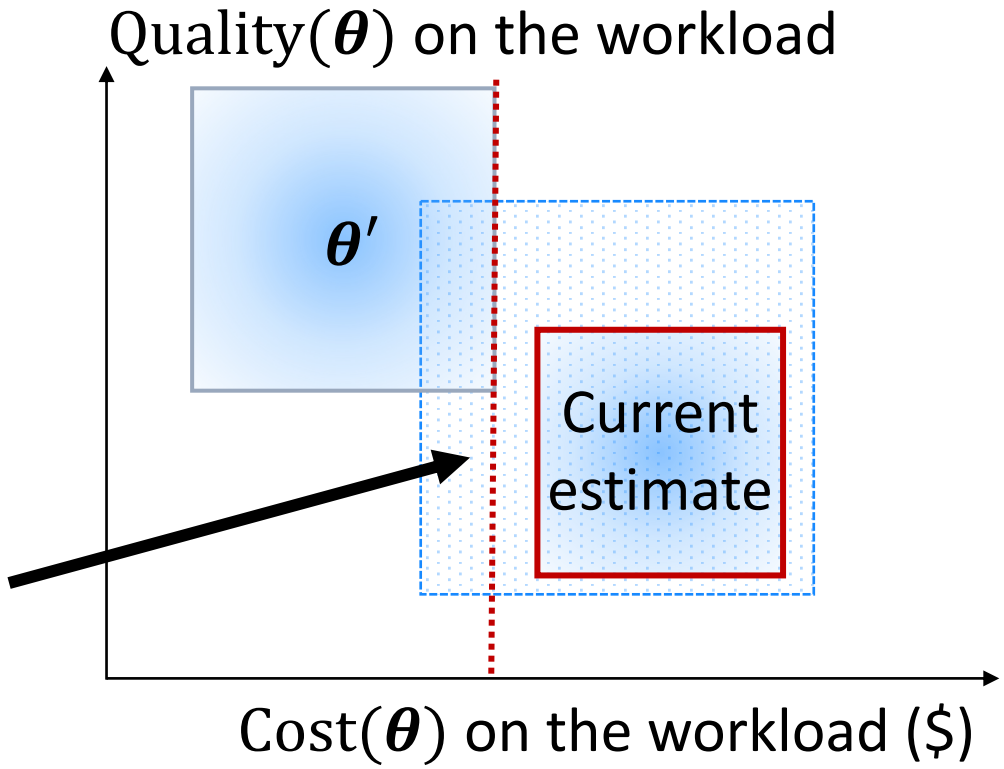


SCOPE (in more detail)

SCOPE's two stopping rules

2. The cost estimate is dominated by the incumbent

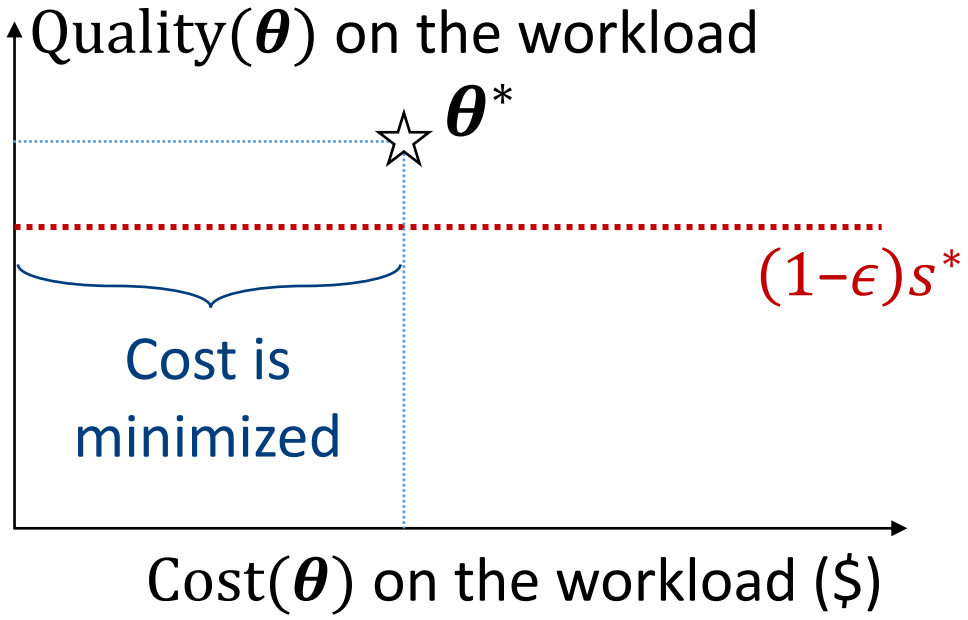
Cost is unlikely to beat $\mathrm{Cost}(\boldsymbol{\theta}')$



Theoretical Guarantee

- Let $\boldsymbol{\theta}^*$ be the *best feasible* configuration at the lowest cost:

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



Theoretical Guarantee

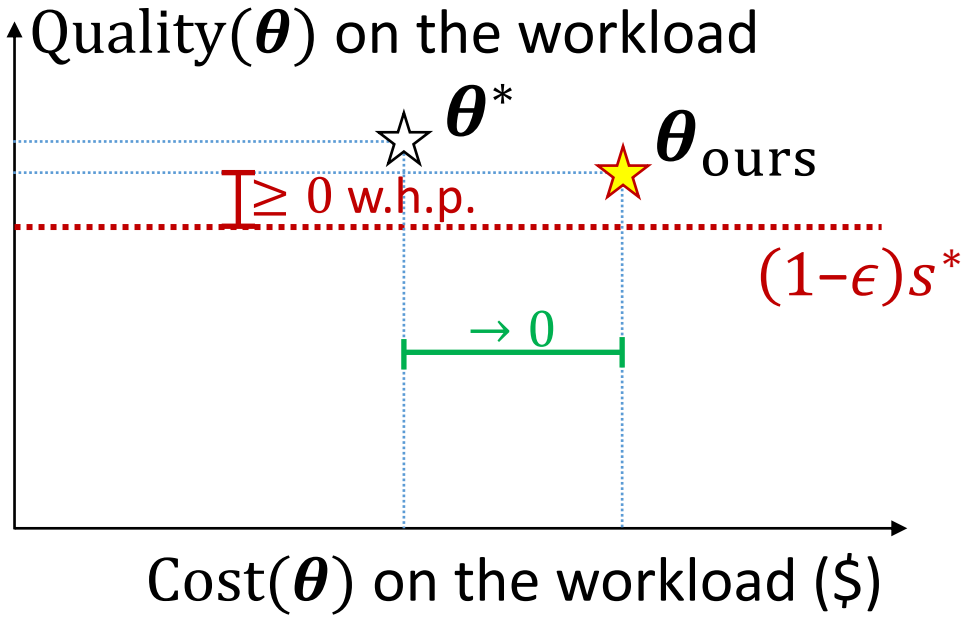
- Let $\boldsymbol{\theta}^*$ be the *best feasible* configuration at the lowest cost:

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

- SCOPE's output ($\boldsymbol{\theta}_{\text{ours}}$) satisfies:

1. $\text{Quality}(\boldsymbol{\theta}_{\text{ours}}) \geq (1-\epsilon)s^*$ w.h.p.
2. $|\text{Cost}(\boldsymbol{\theta}_{\text{ours}}) - \text{Cost}(\boldsymbol{\theta}^*)| \to \mathbf{0}$

   when we increase # of trials



Experiments

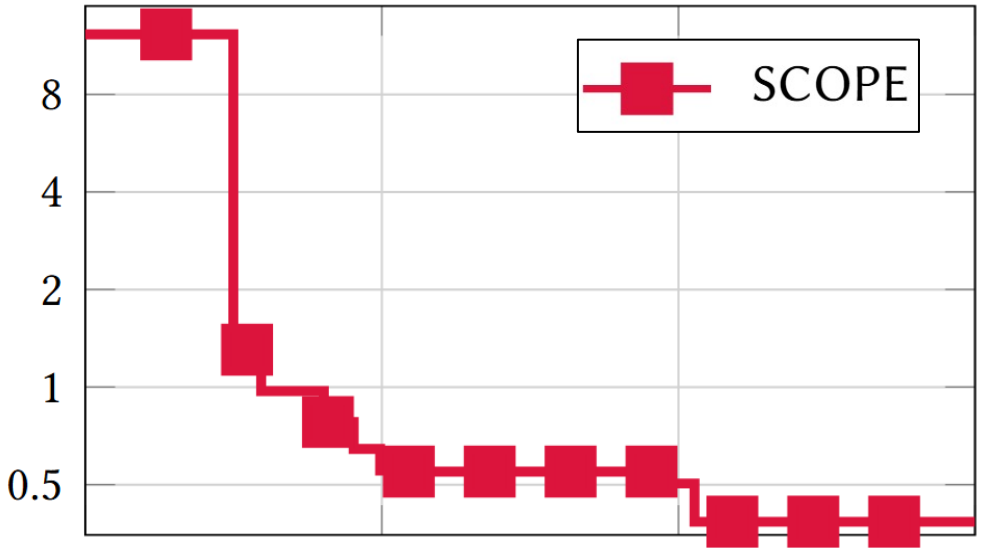
- 3 systems: for Text-to-SQL, Data transformation, Data imputation
- **23 candidate LLMs** (⇒ up to ~6,400,000 possible configurations)
- One workload for each system; $\epsilon = 0.01$

Experiments

- We compare how fast a curve drops
- We will see something like this:

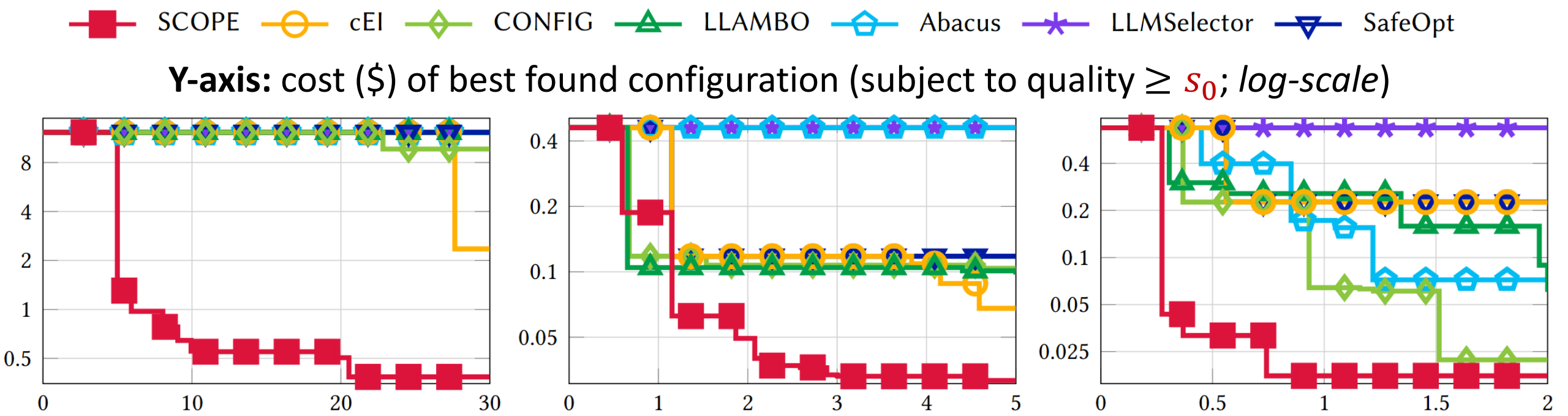
X-axis: budget ($) for each method

Y-axis: cost ($) of best found configuration (subject to quality $\geq (1-\epsilon)s^*$; *log-scale*)



Experiments

- We compare how fast a curve drops



More in the paper

- How does **SCOPE** select a configuration ($\boldsymbol{\theta}^{(t)}$)?
- Details of estimation techniques
  - What confidence bounds are used?
  - How do configuration (and query) estimates relate to each other?
- How does **SCOPE** cope with LLM randomness?
- **More experimental results**

Future Work

- Multi-objective optimization

  (can we optimize both cost and quality at the same time?)
- Optimization on dynamic and agentic systems
- Etc.

Thanks!

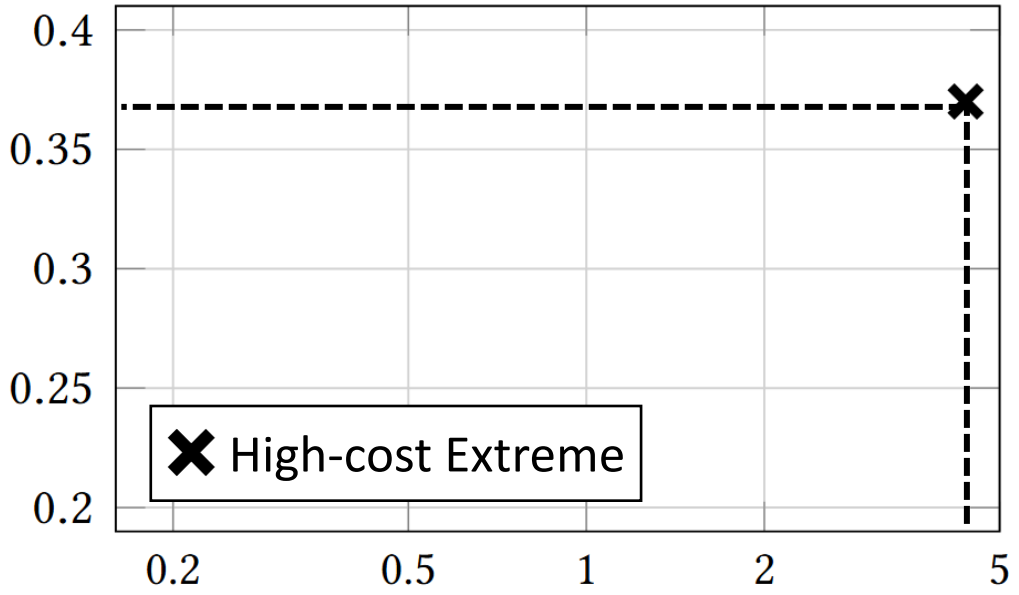
Code: **github.com/waetr/SCOPELLM-optimizer/**

Email**: yiqian@comp.nus.edu.sg**

Experiments (Generalization)

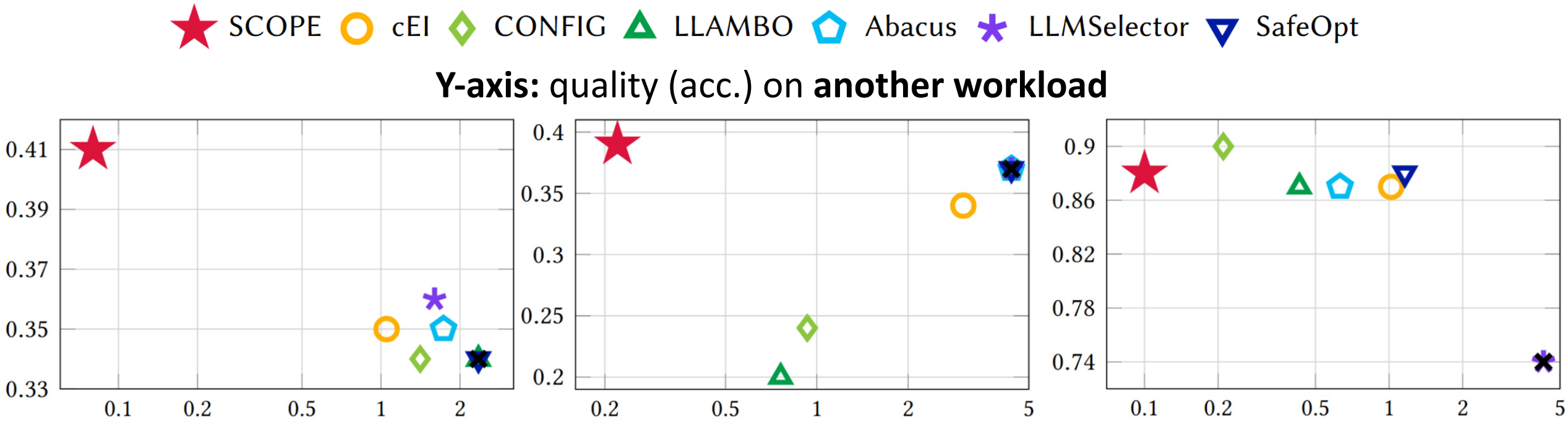
- We use the best configurations found by the methods before
- We re-run the system with them on **another workload**
- We will see something like this

X-axis: cost ($) on **another workload** (*log-scale*)
Y-axis: quality (acc.) on **another workload**



Experiments (Generalization)

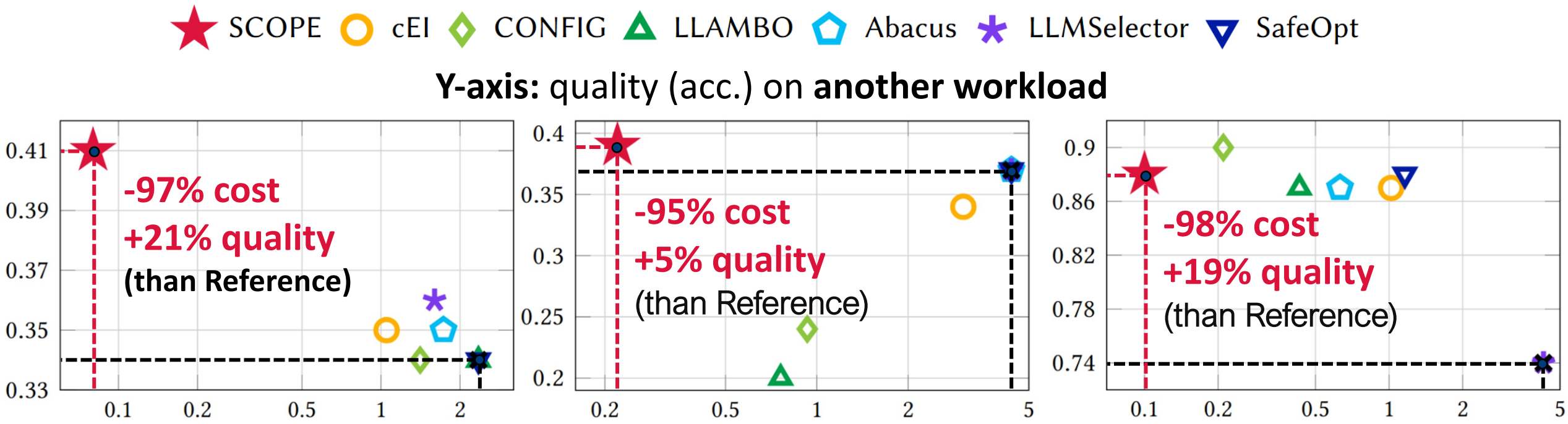
- We mainly compare with ✖ High-cost Extreme



Text-to-SQL | Data transformation | Data imputation

Experiments (Generalization)

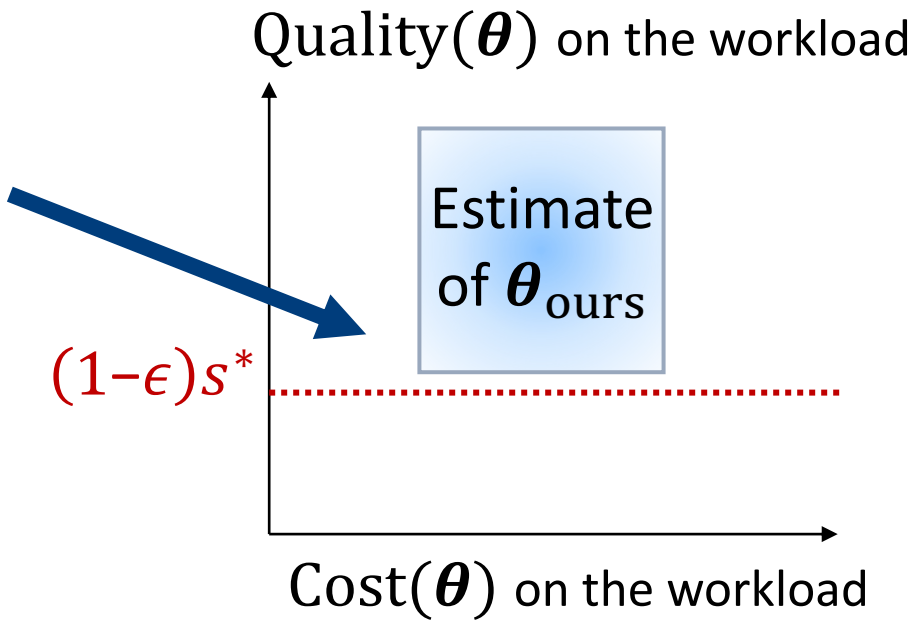
- We mainly compare with ✖ High-cost Extreme (a clearer view)



Experiments (Generalization)

- **Q:** How can SCOPE get even higher quality on another workload?
- **A:** Recall that $\text{Quality}(\boldsymbol{\theta}_{\text{ours}}) \geq (1-\epsilon)s^*$ w.h.p.

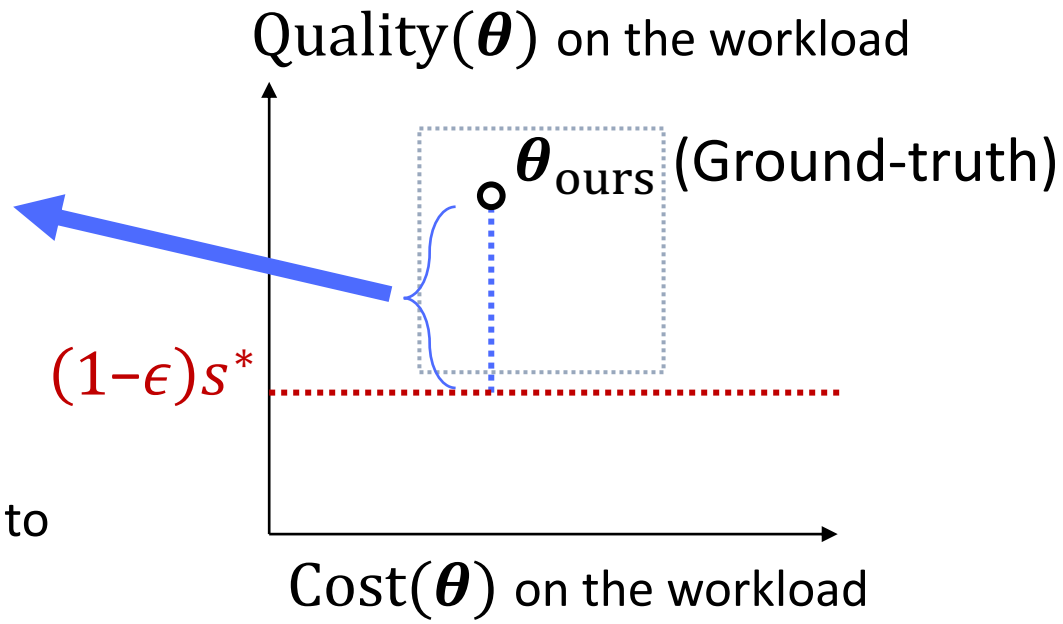
SCOPE returns $\boldsymbol{\theta}_{\text{ours}}$ whose square goes across $(1-\epsilon)s^*$



Experiments (Generalization)

- **Q:** How can SCOPE get even higher quality on another workload?
- **A:** Recall that $\text{Quality}(\boldsymbol{\theta}_{\text{ours}}) \geq (1-\epsilon)s^*$ w.h.p.

Actual $\text{Quality}(\boldsymbol{\theta}_{\text{ours}})$ can be much larger than $(1-\epsilon)s^*$



(This gap is important when we change to another workload)