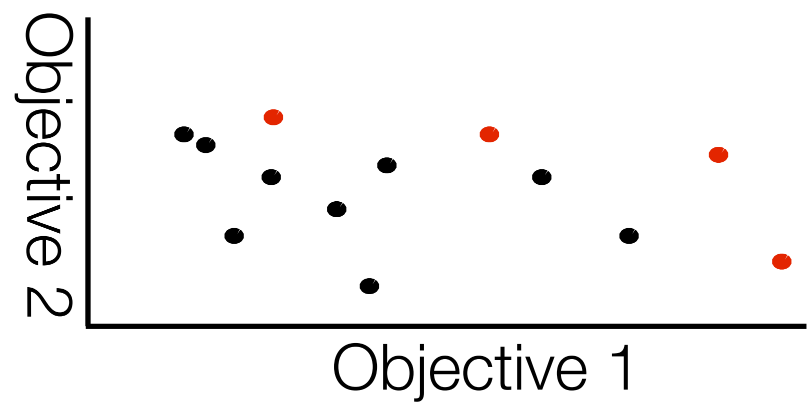
Guessing Preferences:

A New Approach to Multi-Attribute Ranking & Selection

Peter I. Frazier* and Aleksandr M. Kazachov**
* Cornell University, Operations Research and Information Engineering
** Carnegie Mellon University, PhD program in Algorithms Combinatorics and Optimization

Contributions of this Work

- We consider multi-objective (or multi-attribute) optimization, where each objective can only be evaluated through simulation.



- We contribute:

  (1) A new measure of the quality of an estimate of the Pareto frontier.

  (2) A new algorithm for allocating samples to optimize this quality measure.

Related Work

- **Multi-objective optimization via simulation:**

  - **Multi-objective OCBA**: Lee, Teng, Chew, Lye, Lendermann, Karimi, Chen, & Koh 2005; Lee, Chew, & Teng 2007; Chen & Lee 2009; Lee, Chew, Teng, & Goldsman 2010;
    R&S with stochastic constraints: Andradottir, Goldsman, & Kim 2005; Andradottir & Kim 2010; Luo & Lim 2011; and others;
    Sample average approximation: Kim & Ryu 2011;
    MO-COMPASS: Lee, Chew, Li 2011.

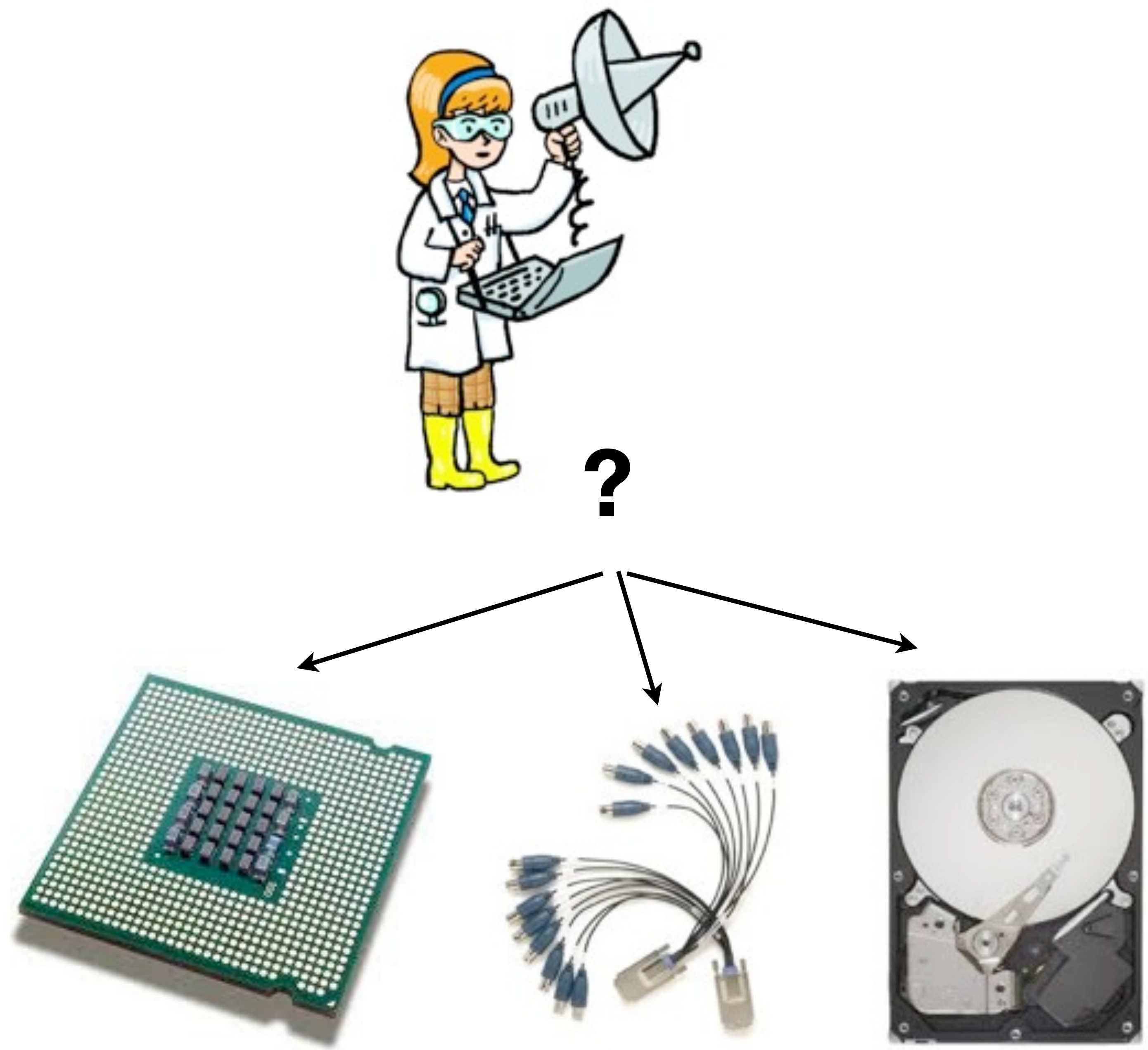
- **Multi-objective global optimization of expensive functions** (usually noise free): Keane 2006; Bautista 2009; Knowles 2006; Forrester, Sobester, & Keane 2008.

- **Measures of the quality of an estimate of the Pareto frontier**: Faulkenberg & Wiecek 2010; Sayin 2000; Zitzler, Knowles, & Thiele; Leung & Wang 2003; Meng, Zhang & Liu 2005; Zitzler, Brockhoff, & Thiele 2007; Zitzler, Thiele, Laumanns, Foneseca, & V. Fonseca 2003.

- **Putting a Bayesian prior on DM's utility function to allow efficient utility elicitation**: Chajewska, Koller, & Parr 2000; Boutilier 2002; Abbas 2004.

We Begin with an Example

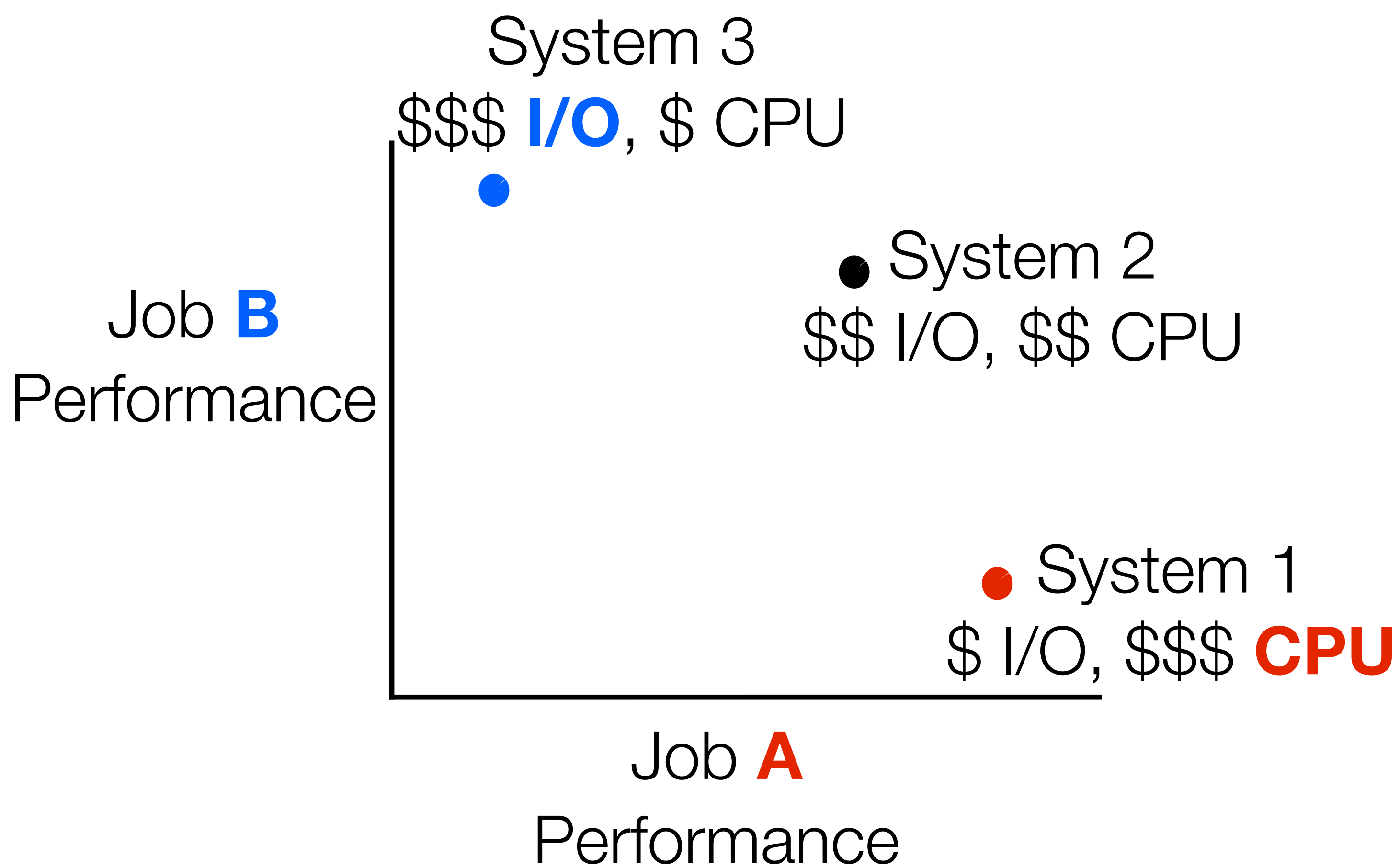
- A scientist has just been awarded a research grant, which includes a fixed budget of $25K to build a parallel high-performance computing system.
- How should she allocate this money across different types of hardware?
- How many compute nodes? What type of CPUs? How much storage and what type? What type of interconnect?



Choice of hardware depends on the compute job

- The appropriate choice of hardware depends on the type of compute job.

- The scientist DM needs to run 2 types of compute jobs.
  - Job type **A** is **CPU** intensive. It will be faster if the system has fast CPUs.
  - Job type **B** is **I/O** intensive. It will run faster if the system has fast disk and interconnect.



The scientist decision-maker (DM) asks for advice

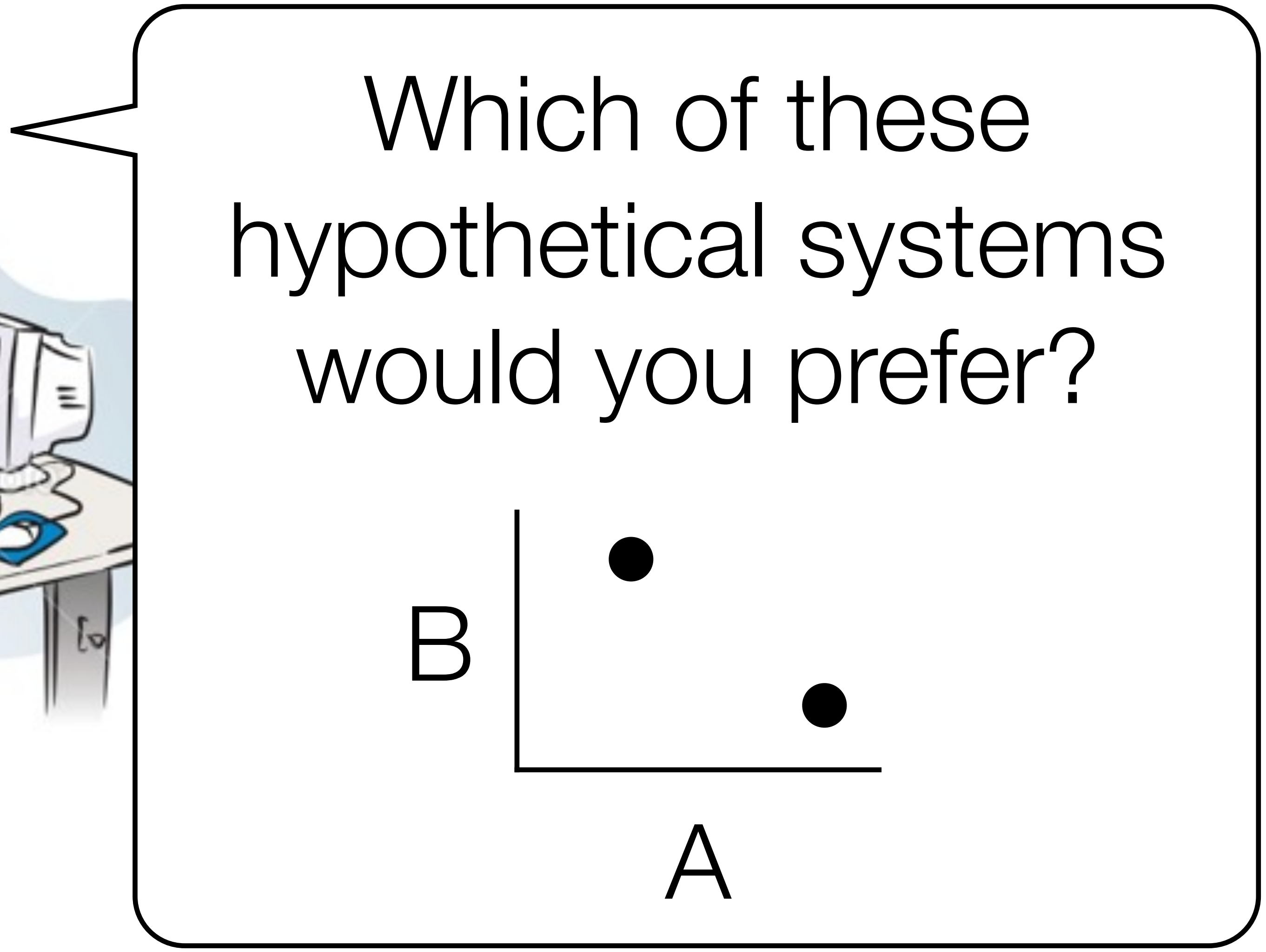
- The DM asks her contact in the university's research IT department for advice:

Please help me choose the design of my new computing system.

I need it to run the following types of compute jobs....

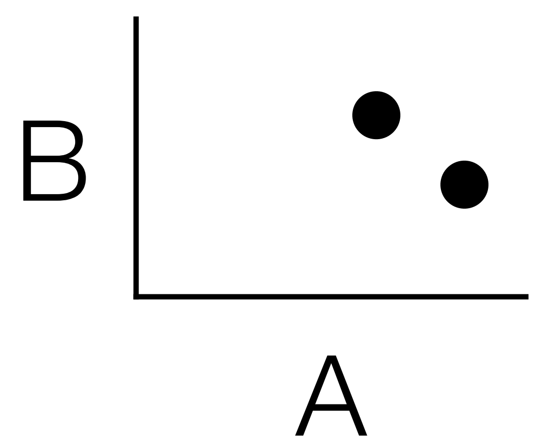
Analyst from IT department

One approach:
Elicit the DM's preferences before optimizing



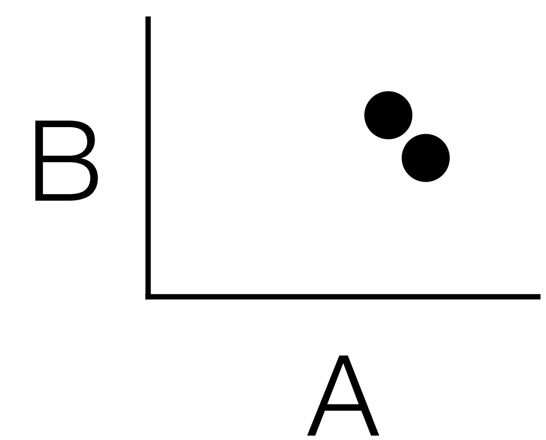
One approach:
Elicit the DM's preferences before optimizing

Now which of these of hypothetical systems?



One approach:
Elicit the DM’s preferences before optimizing

Now which of these hypothetical systems?

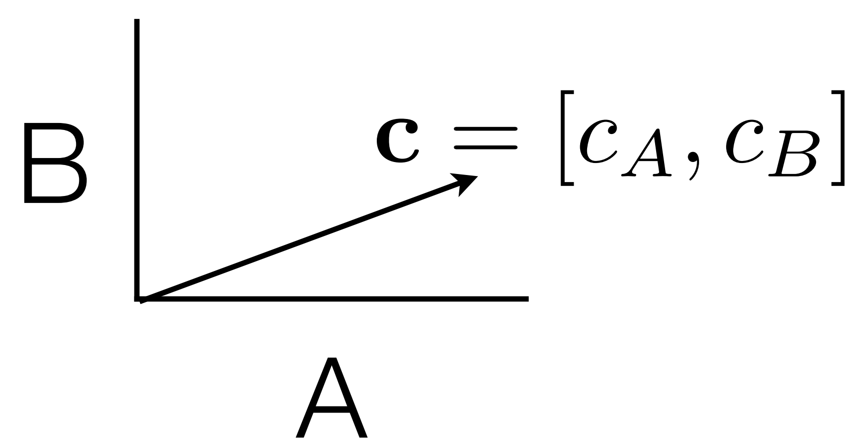


One approach:
Elicit the DM's preferences before optimizing

I now know your utility function!! It is

$$U(x) = \mathbf{c} \cdot \mathbf{v}(x)$$

where the vector c is



Excuse me while I go find a real system that optimizes that.

In some situations, we cannot elicit the DM’s preferences before optimizing.

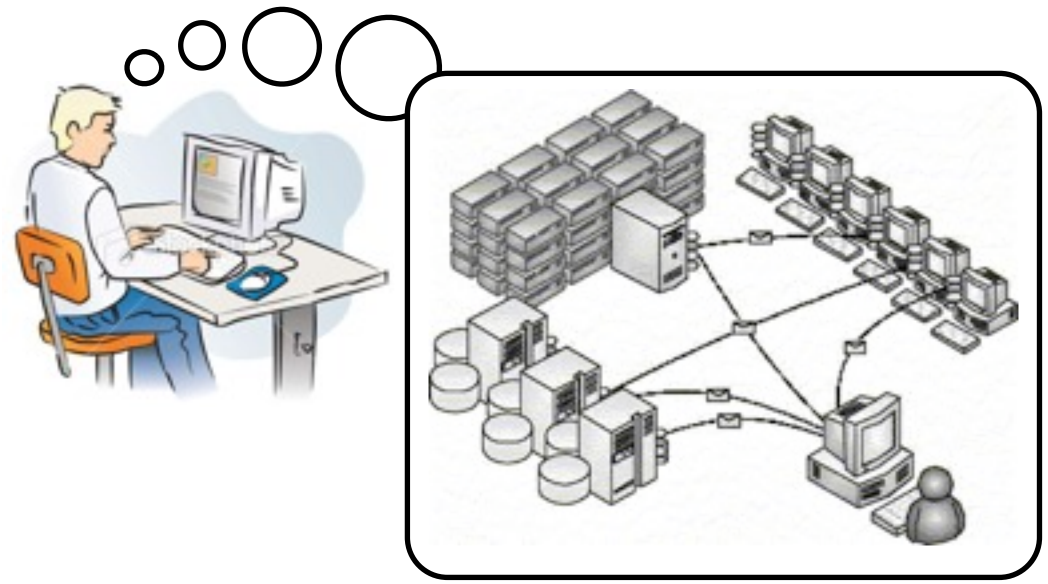
I am busy.

I do not have time to have my preferences elicited.

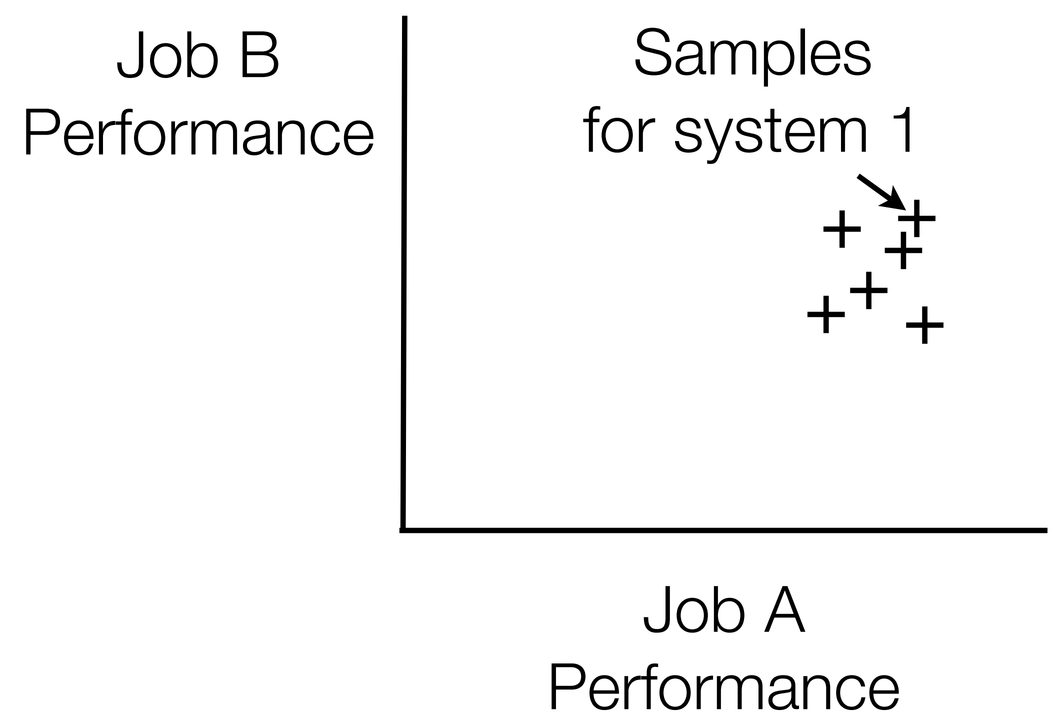
Give me an estimate of the Pareto frontier & I will use that to make my decision.

The IT analyst uses simulation to estimate the Pareto frontier.

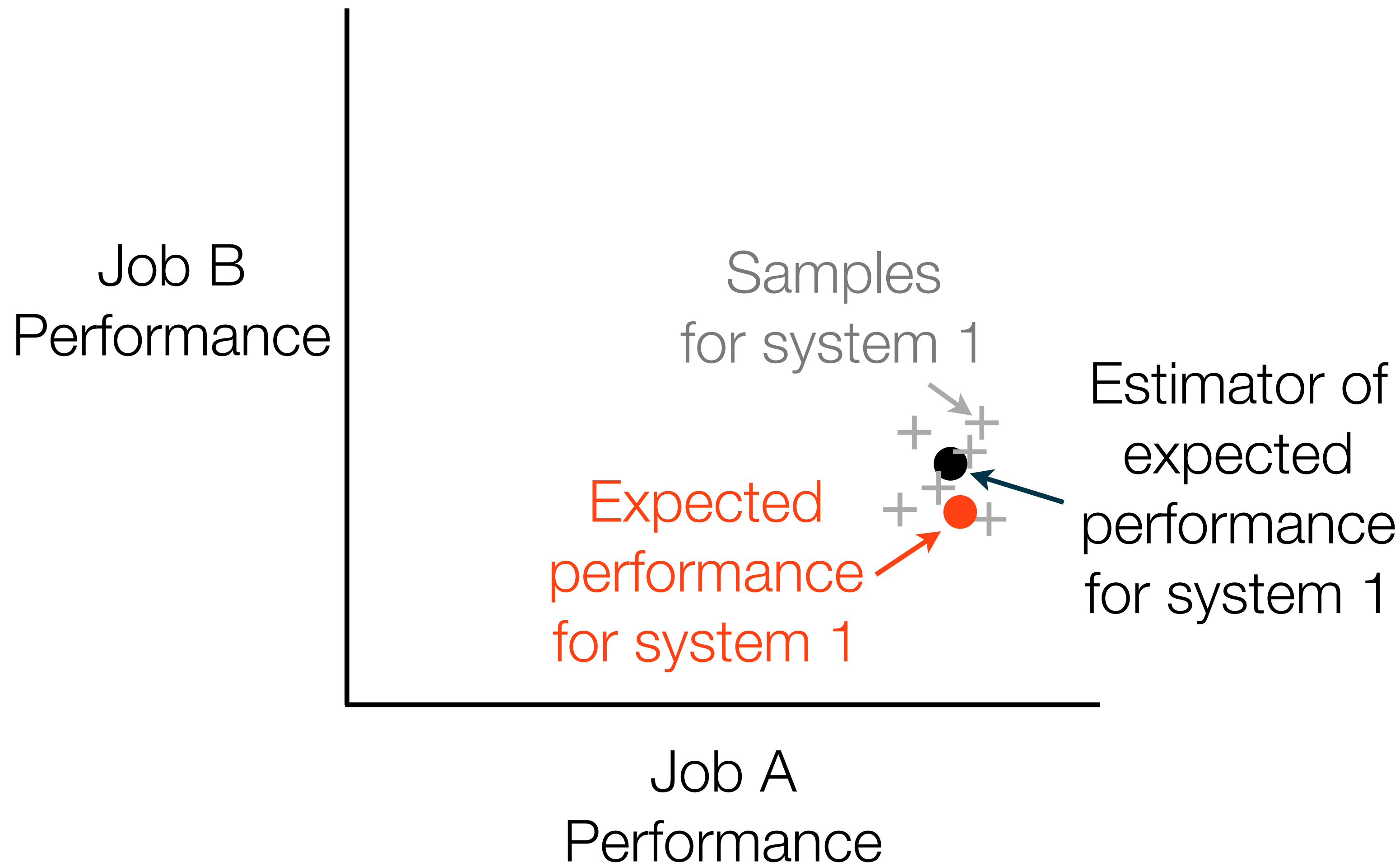
- The IT department has a stochastic simulator. Given a system configuration, this simulator can sample the system's response time for each job type.



- For simplicity, suppose we care only about a single expected performance measure for each job type, e.g., expected number of jobs of each type that can be finished in 1 hour.

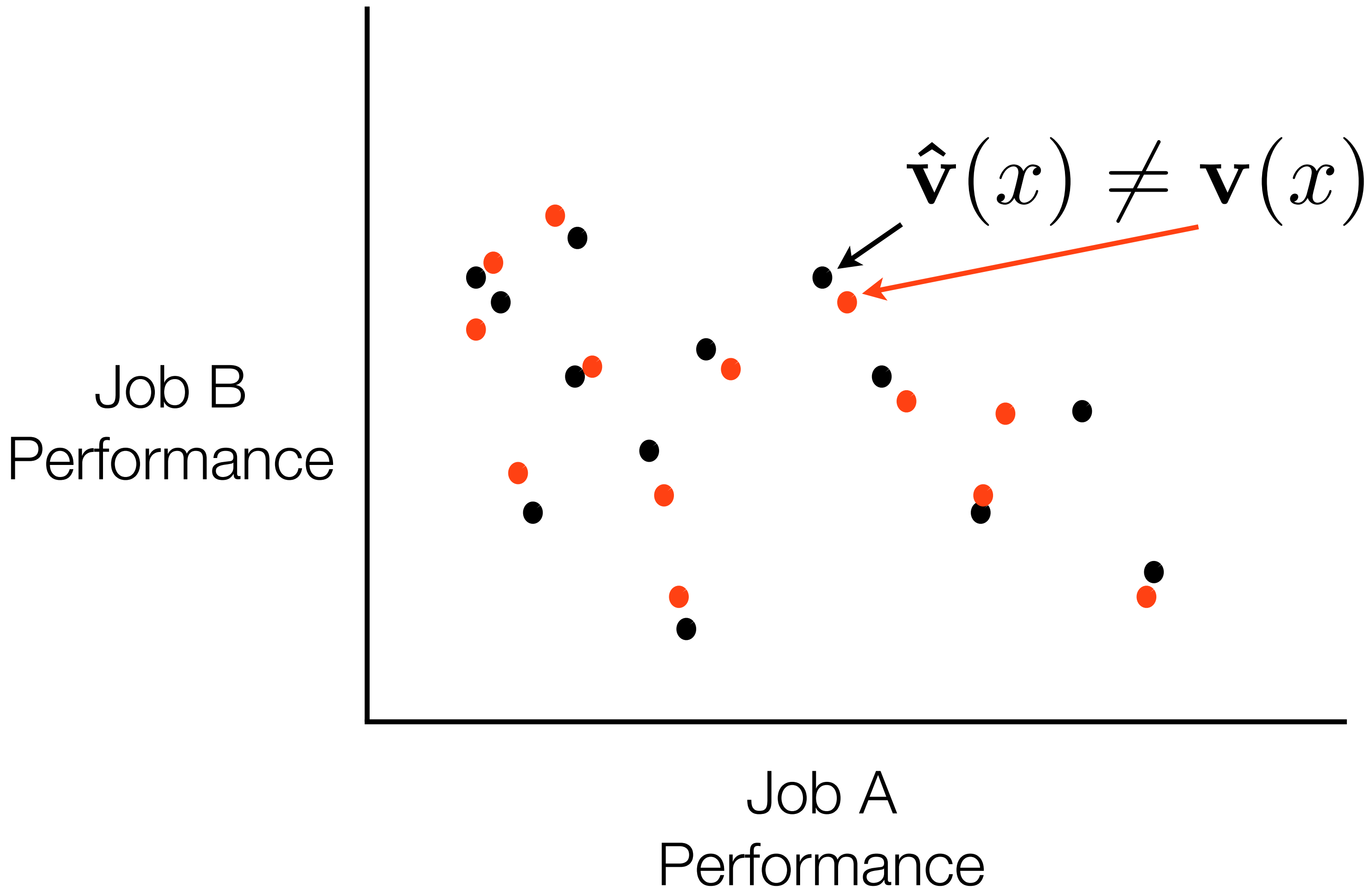


We estimate performance from simulation samples.

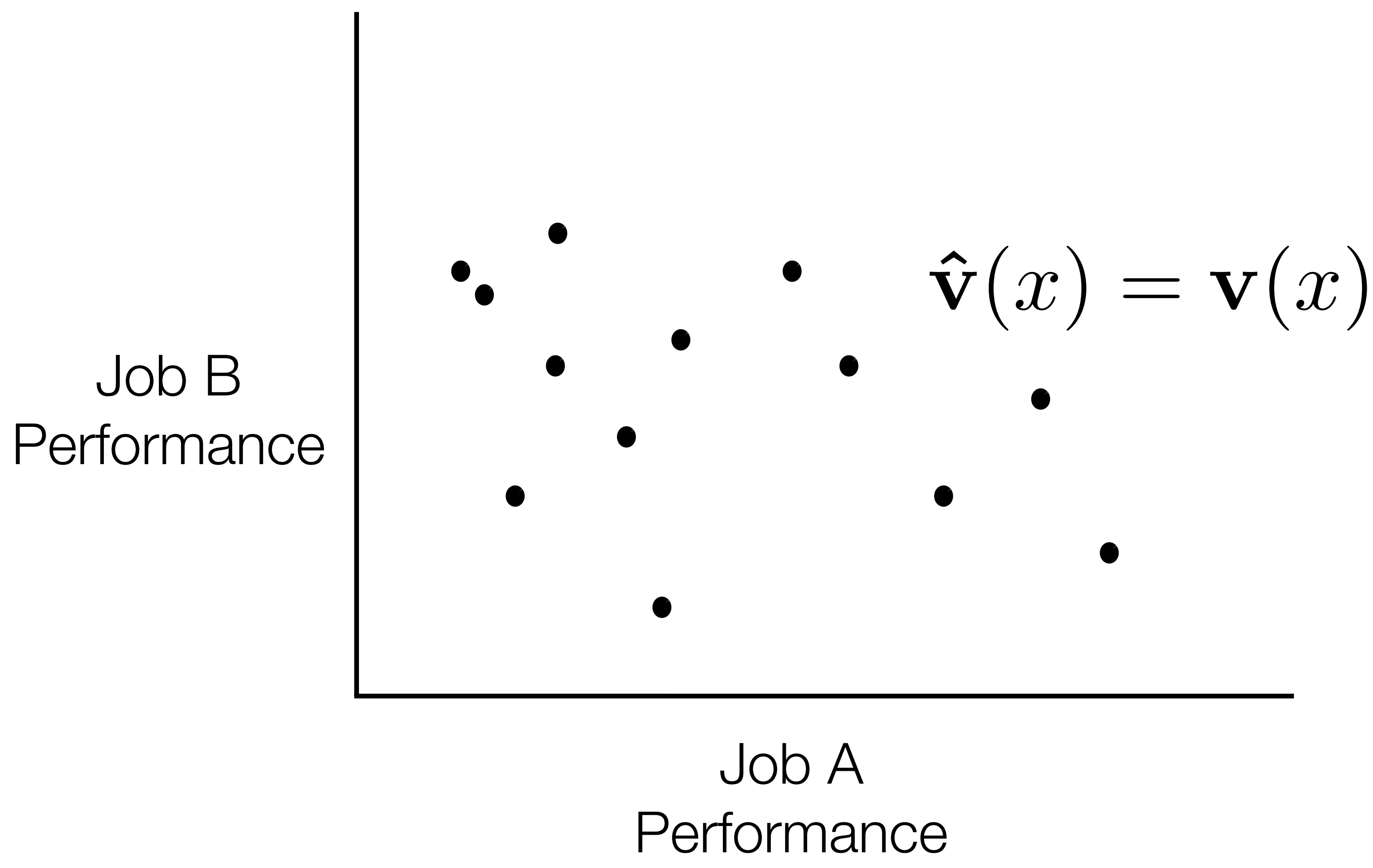


- Let $\mathbf{v}(x) = [v_A(x), v_B(x)]$ be the expected performance vector for system x.
- Let $\hat{\mathbf{v}}(x) = [\hat{v}_A(x), \hat{v}_B(x)]$ be our estimate of this vector.

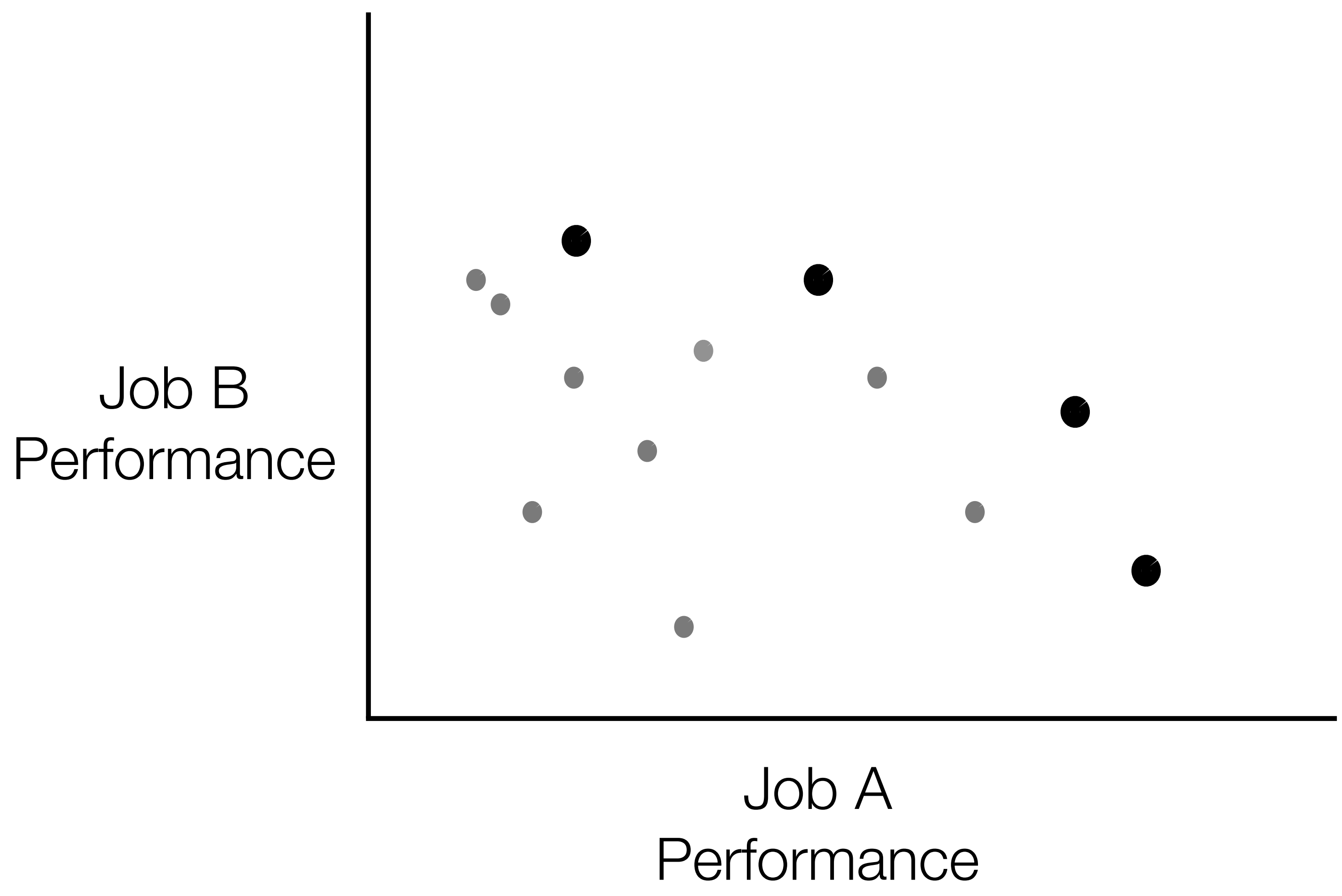
Because of sampling noise, estimates are imperfect



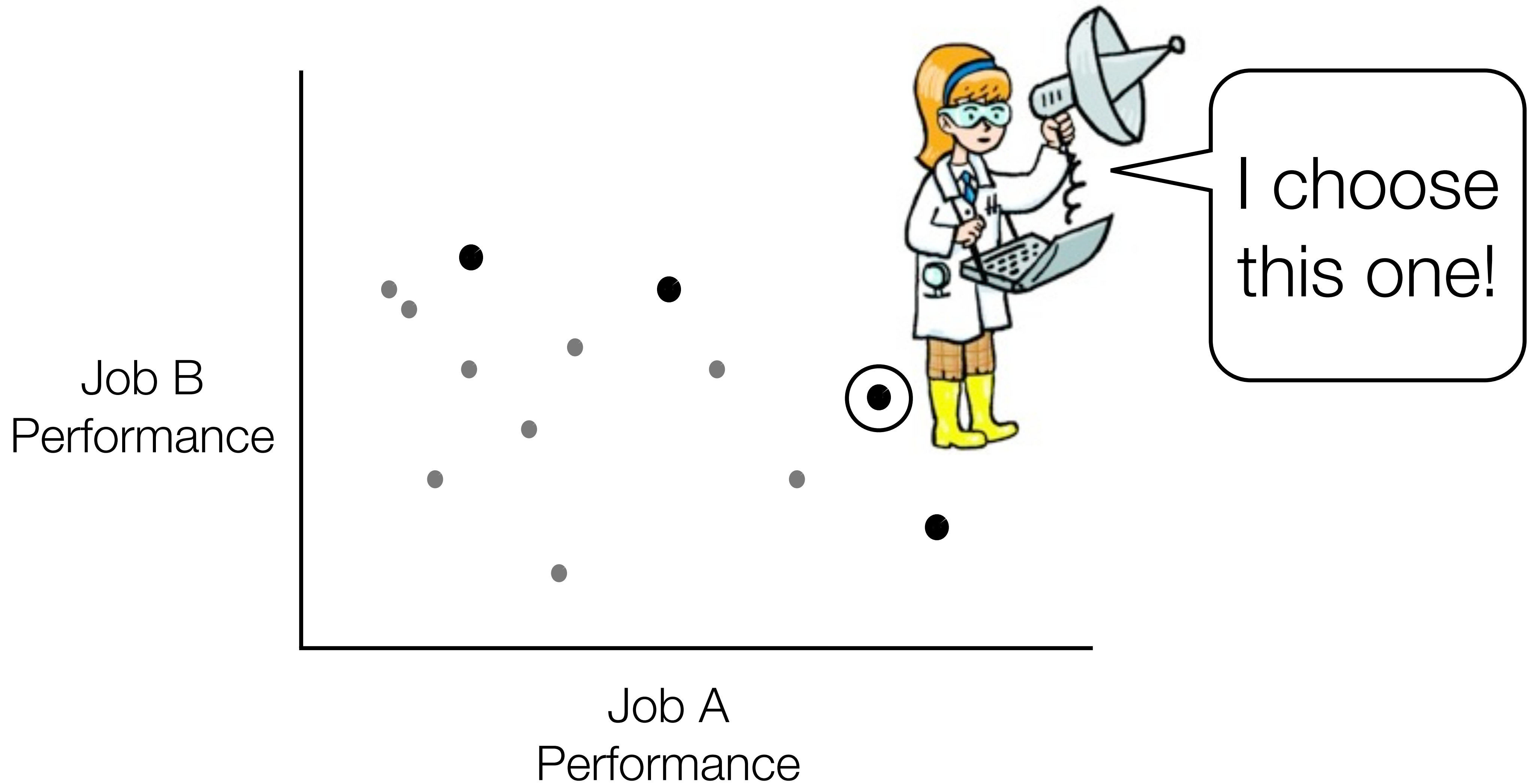
If we could take an infinite number of samples from each system, estimates would be perfect.



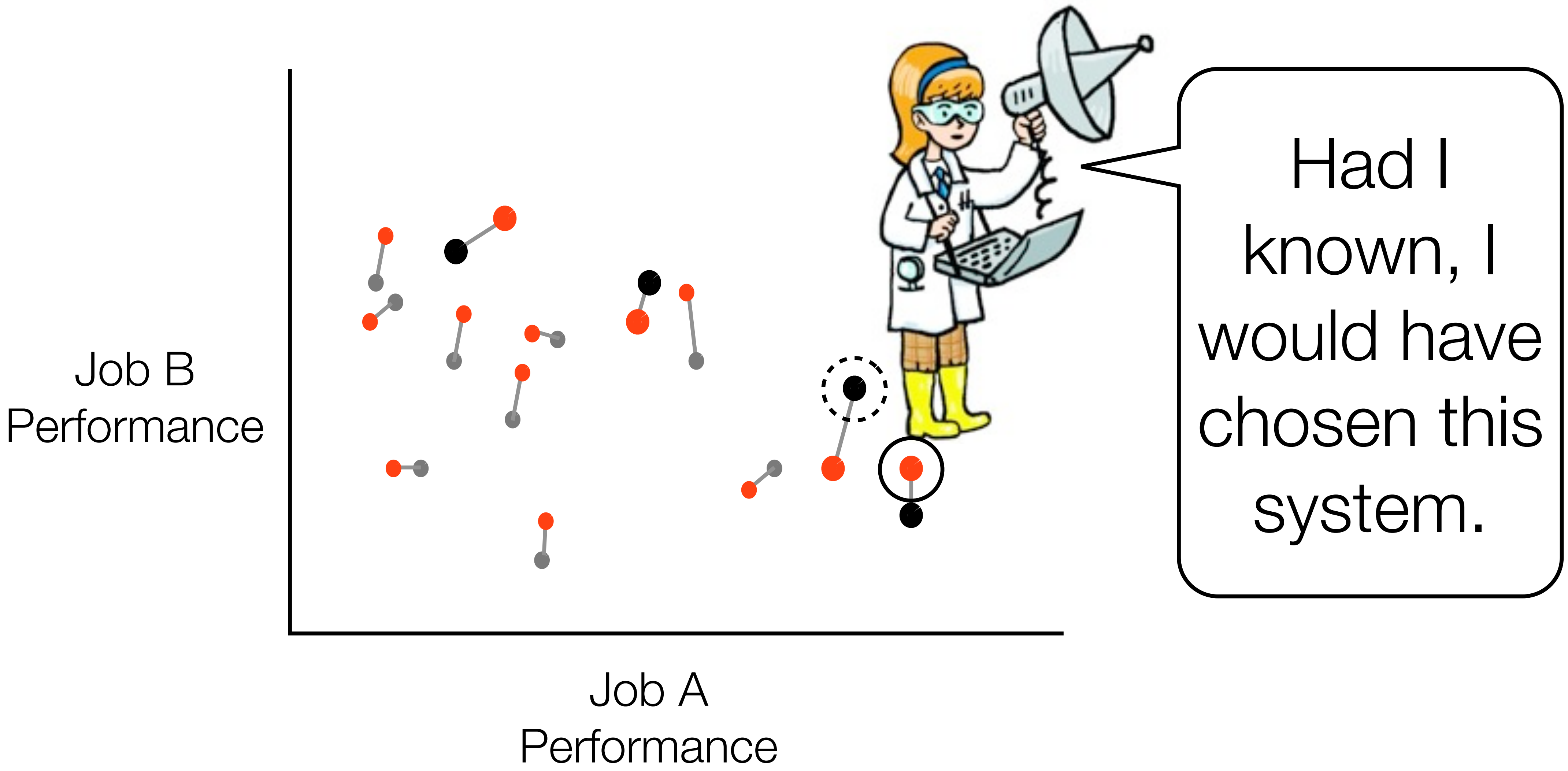
Given perfect estimates, the DM would choose among points on the Pareto frontier.



Given perfect estimates, the DM would choose among points on the Pareto frontier.



The DM suffers some loss from imperfect estimates.



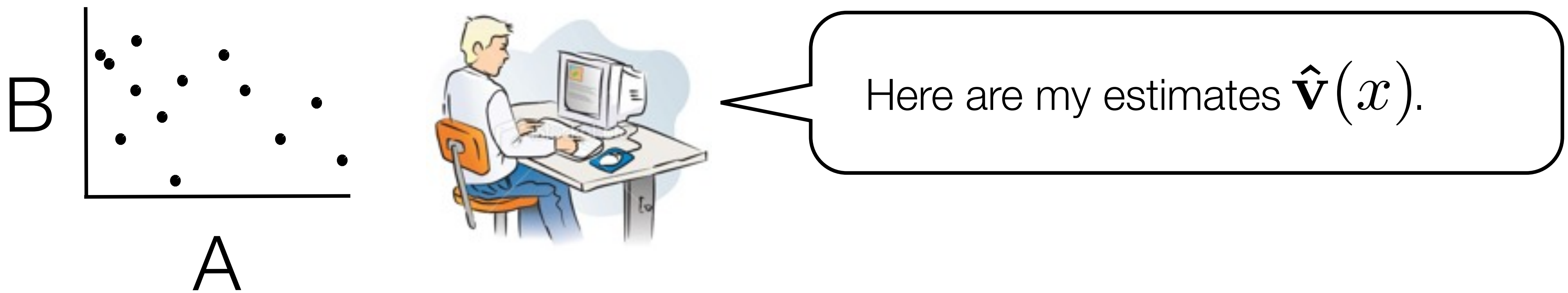
Central Questions

1. What is the loss to the DM caused by imperfect estimates?

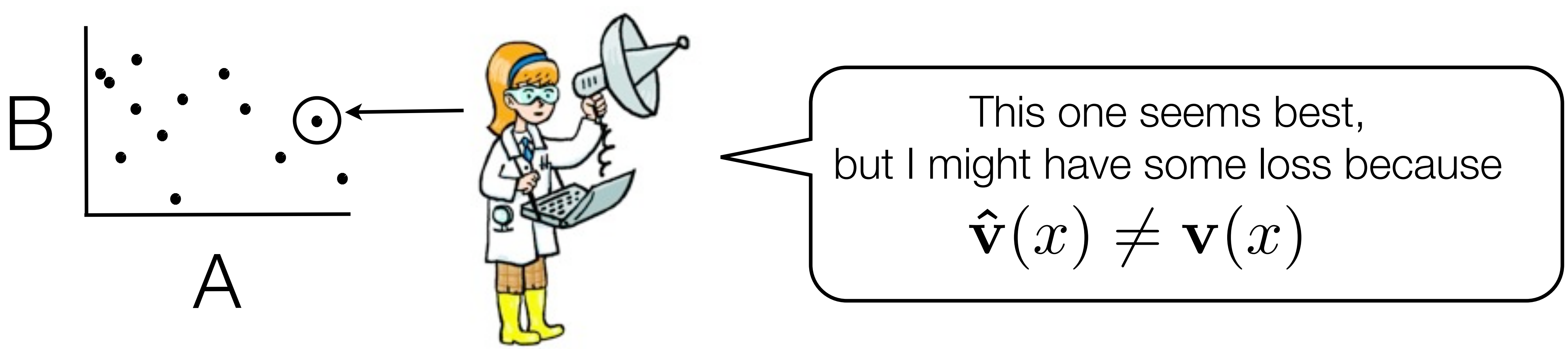
2. How should we allocate our samples across the systems to minimize this loss?

Recap: The analyst shows estimates to the DM; Then the DM chooses, suffering some loss.

1. The analyst takes some simulation samples, and uses these to estimate the system's attributes.



2. The analyst shows these estimates to the DM. She chooses the one that seems best to her internal but un-elicited utility function.



What is the DM's loss?

- We suppose the DM's utility is $U(x) = \mathbf{c} \cdot \mathbf{v}(x)$ for some vector $\mathbf{c}$. The analyst does not know $\mathbf{c}$ and does not try to elicit it. Note: this linear form disallows non-convex Pareto sets.

- Given systems with estimated performances, the DM can determine which would provide her with the most utility, if the estimates were correct. Based on the estimates provided, she chooses
$$\hat{x}(\mathbf{c}) = \arg\max_x \mathbf{c} \cdot \hat{\mathbf{v}}(\mathbf{x})$$

- Had the estimates been perfect, she would have chosen
$$x^*(\mathbf{c}) = \arg\max_x \mathbf{c} \cdot \mathbf{v}(\mathbf{x})$$

- Her loss is the **difference in utility** between the two choices
$$L(\mathbf{v}, \hat{\mathbf{v}}, \mathbf{c}) = \mathbf{c} \cdot \mathbf{v}(x^*(\mathbf{c})) - \mathbf{c} \cdot \mathbf{v}(\hat{x}(\mathbf{c}))$$

The DM's loss is unknown to the analyst.

- The DM's loss is

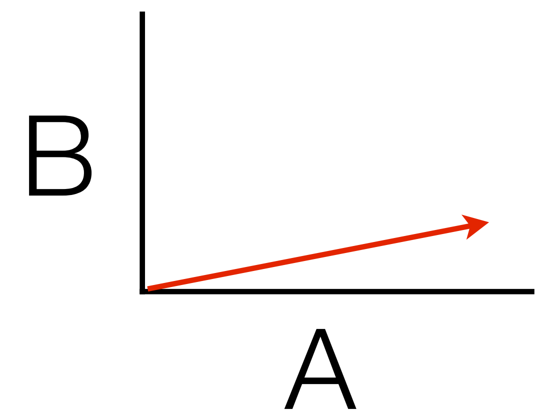
$$L(\mathbf{v}, \hat{\mathbf{v}}, \mathbf{c}) = \mathbf{c} \cdot \mathbf{v}(x^*(\mathbf{c})) - \mathbf{c} \cdot \mathbf{v}(\hat{x}(\mathbf{c}))$$

- This depends on the DM's preferences through the vector $\mathbf{c}$, which is unknown to the analyst.

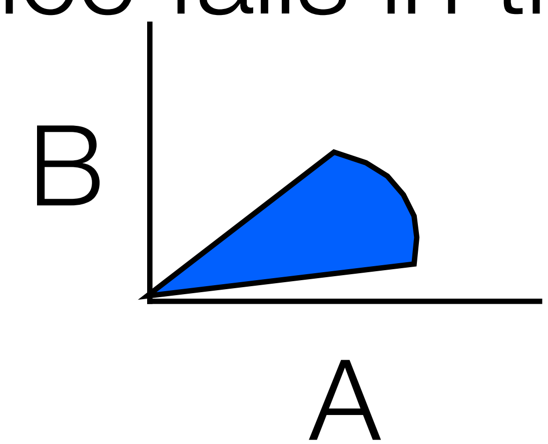
- How can the analyst minimize the loss without knowing $\mathbf{c}$?

The analyst has intuition about the DM's utility.

Based on my past experience,
and my conversation with the DM,
it's most likely her preference is this:



Also, there's a 90% chance her
preference falls in this range:



We model the analyst's intuition with a Bayesian prior distribution on the DM's utility.

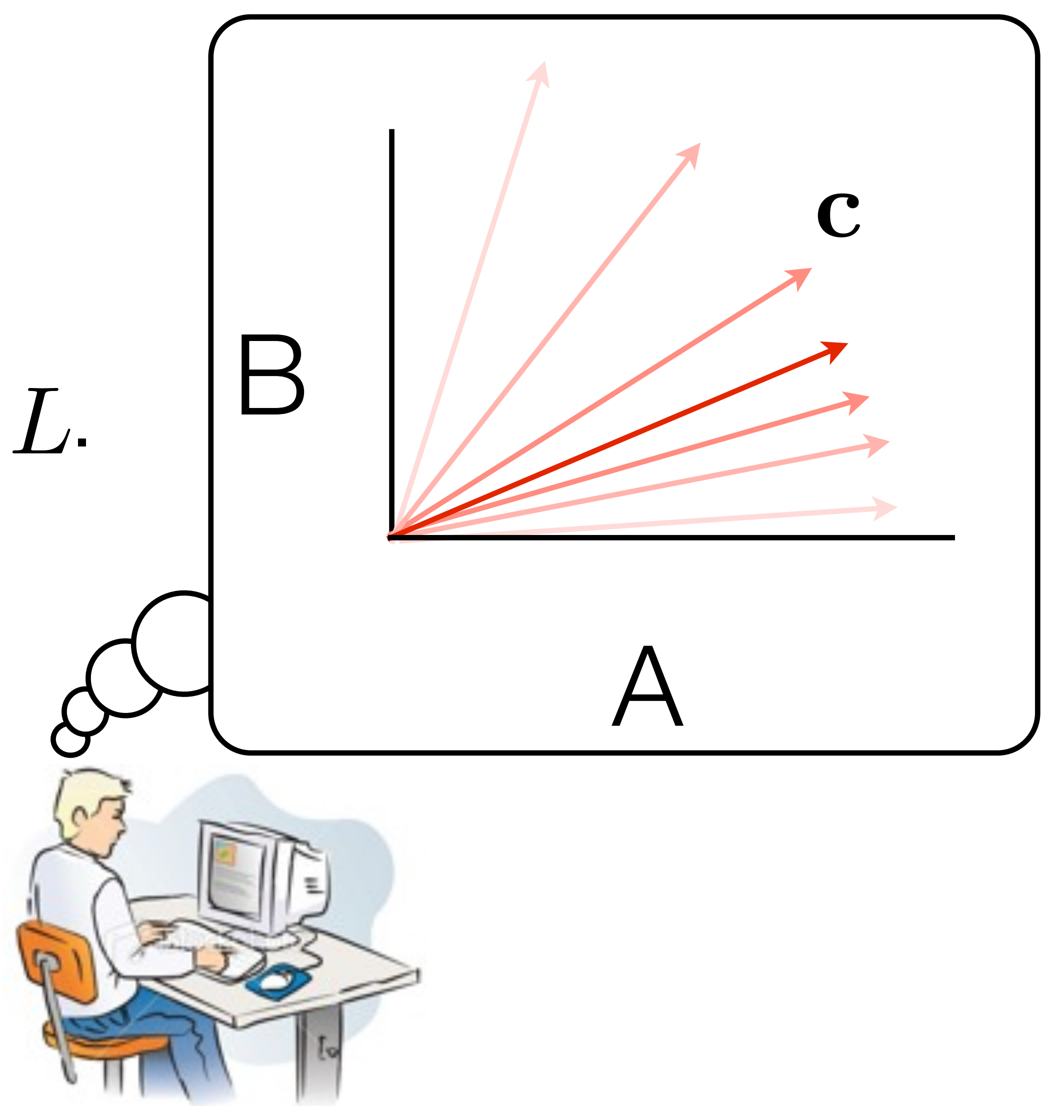
- Recall: $U(x) = \mathbf{c} \cdot \mathbf{v}(x)$ where the vector c is unknown.

- We suppose that the simulation analyst has a Bayesian prior distribution on the vector c. This prior distribution can be continuous or discrete.

- Our algorithm is nicer if this prior is discrete,

$$P(\mathbf{c} = \mathbf{c}_\ell) = p_\ell$$

where $\mathbf{c}_\ell, p_\ell$ are fixed values and $\ell = 1, \ldots, L$.



We measure the quality of our estimate with the expected loss suffered by the DM.

- Given c, the DM's loss from estimation error is

$$L(\mathbf{v}, \hat{\mathbf{v}}, \mathbf{c}) = \mathbf{c} \cdot \mathbf{v}(x^*(\mathbf{c})) - \mathbf{c} \cdot \mathbf{v}(\hat{x}(\mathbf{c}))$$

- With our prior on c, the loss from estimation error is

$$L(\mathbf{v}, \hat{\mathbf{v}}) = E[L(\mathbf{v}, \hat{\mathbf{v}}, \mathbf{c}) | \mathbf{v}, \hat{\mathbf{v}}]$$

- When our prior is discrete, this can be written,

$$L(\mathbf{v}, \hat{\mathbf{v}}) = \sum_{\ell=1}^{L} p_\ell L(\mathbf{v}, \hat{\mathbf{v}}, \mathbf{c}_\ell)$$

- **This is a general measure of the quality of an estimated Pareto frontier.**

We use a standard Bayesian framework to model uncertainty about the sampling means.

- We assume that samples are independent and normally distributed.

$$y_j(x) \sim \text{Normal}(v_j(x), \lambda_j(x))$$

- We suppose an independent Bayesian prior distribution on the sampling means and variances,

$$v_j(x)|\lambda_j(x) \sim \text{Normal}(\mu_{0jx}, \lambda_j(x)/\rho_{0jx})$$
$$1/\lambda_j(x) \sim \text{Gamma}(a_{0jx}, b_{0jx})$$

- We obtain a Bayesian posterior distribution after sampling.

$$v_j(x)|\lambda_j(x) \sim \text{Normal}(\mu_{njx}, \lambda_j(x)/\rho_{njx})$$
$$1/\lambda_j(x) \sim \text{Gamma}(a_{njx}, b_{njx})$$

The Bayesian posterior distribution implies a Bayesian expected loss

- Our Bayesian posterior distribution after sampling is

$$v_j(x)|\lambda_j(x) \sim \mathrm{Normal}(\mu_{njx}, \lambda_j(x)/\rho_{njx})$$

$$1/\lambda_j(x) \sim \mathrm{Gamma}(a_{njx}, b_{njx})$$

- The Bayes-optimal estimate is the posterior mean

$$\hat{\mathbf{v}}(x) = E_n[\mathbf{v}(x)] = \vec{\mu}_{nx}$$

where E_n is the expectation with respect to the posterior distribution.

- Our Bayesian expected loss at time n is

$$E_n[L(\mathbf{v}, \hat{\mathbf{v}})] = E_n[\max_x \mathbf{c} \cdot \mathbf{v}(x)] - \max_x \mathbf{c} \cdot \vec{\mu}_{nx}$$

We derive a knowledge-gradient allocation method for this loss function.

- We define the knowledge-gradient (KG) factor to be the expected one-step reduction in loss,

$$\mathrm{KG}_{nx} = E_n \left[\left(\max_{x'} \mathbf{c} \cdot \vec{\mu}_{n+1,x'} \right) - \left(\max_{x'} \mathbf{c} \cdot \vec{\mu}_{n,x'} \right) | x_{n+1} = x \right]$$

- The knowledge-gradient policy allocates the next sample to the one that maximizes this one-step benefit,

$$x_{n+1} \in \arg\max_x \mathrm{KG}_{nx}$$

- The KG factor can be computed analytically if the sampling variance is known. We use an adaptively updated point estimate of the variance as an approximation when the variance is unknown.

Details of the knowledge-gradient policy

- When the sampling variances are known and the prior on c is discrete, the KG factor is

$$\mathrm{KG}_{nx}(\lambda) = \sum_{\ell=1}^{L} p_\ell \tilde{\sigma}_{nx}(\mathbf{c},\lambda) f\left(\frac{-\Delta_{nx}(\mathbf{c})}{\tilde{\sigma}_{nx}(\mathbf{c},\lambda)}\right)$$

where $(\tilde{\sigma}_{nx}(\mathbf{c},\lambda))^2 = \sum_{j=1}^{m} c_j^2 \frac{\lambda_{xj}}{\rho_{nxj}(\rho_{nxj}+1)},$

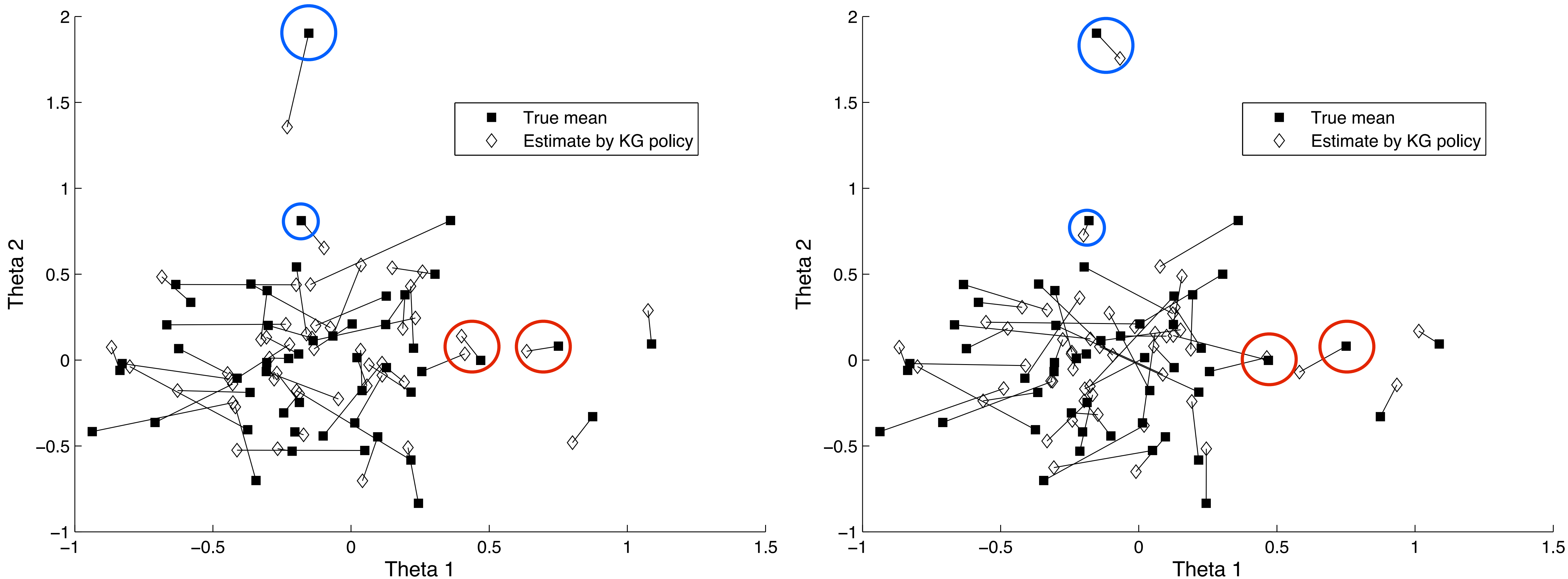
$$f(d) = d\Phi(d) + \varphi(d),$$

$$\Delta_{nx}(\mathbf{c}) = \left| \mathbf{c}\cdot\vec{\mu}_{nx} - \max_{x'\neq x} \mathbf{c}\cdot\vec{\mu}_{nx'} \right|.$$

- We then obtain a vector of point estimates $\hat{\lambda}_n$ and choose to sample next,

$$x_{n+1} \in \arg\max_x \mathrm{KG}_{nx}(\hat{\lambda}_n)$$

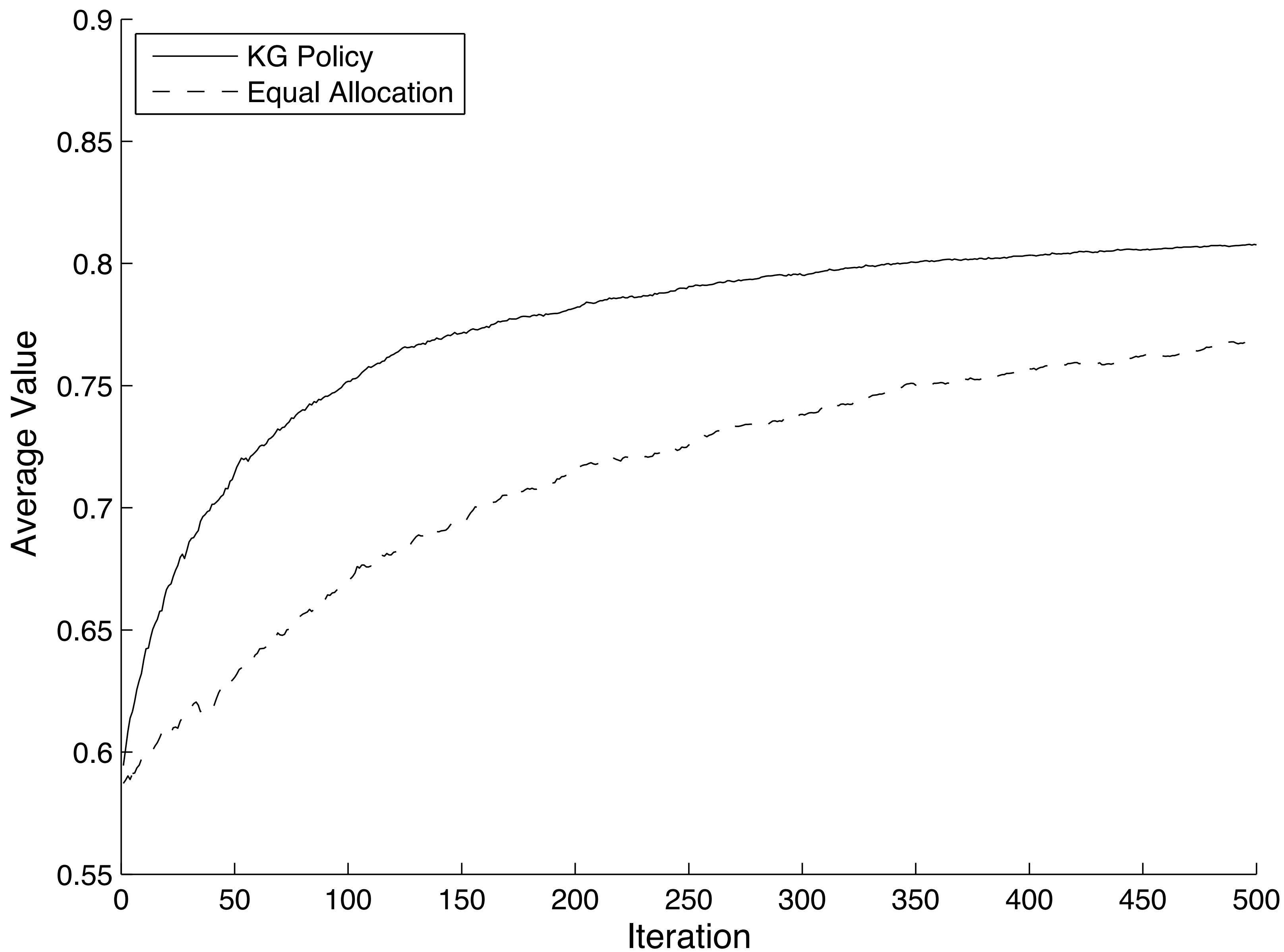
Numerical Results



Prior says **horizontal** objective is likely to be more important. More samples are allocated to alternatives with good horizontal values, giving better estimates.

Prior says **vertical** objective is likely to be more important. More samples are allocated to alternatives with good vertical values, giving better estimates.

Numerical Results



- Expected reward vs the number of samples taken (after an initial stage of 5 samples per alternative completes). Parameters: 20 systems, sampling variance of 1, homogeneous prior means, 5 equi-probable values of c. Maximum standard error is 0.011.

Conclusion

- We presented a new way to think about Pareto frontier estimation, and the quality of a Pareto frontier.

- We presented a new KG algorithm based on this measure. Other algorithmic approaches can also use this measure.

- Future work:
  - Compare this measure of the Pareto frontier to other measures.
  - Compare the KG algorithm to other multi-objective algorithms.
  - Incorporate correlated prior distributions.
  - Incorporate noisy information from DM about preferences.

Thank You!