Foundations of Application-Specific Algorithm Configuration

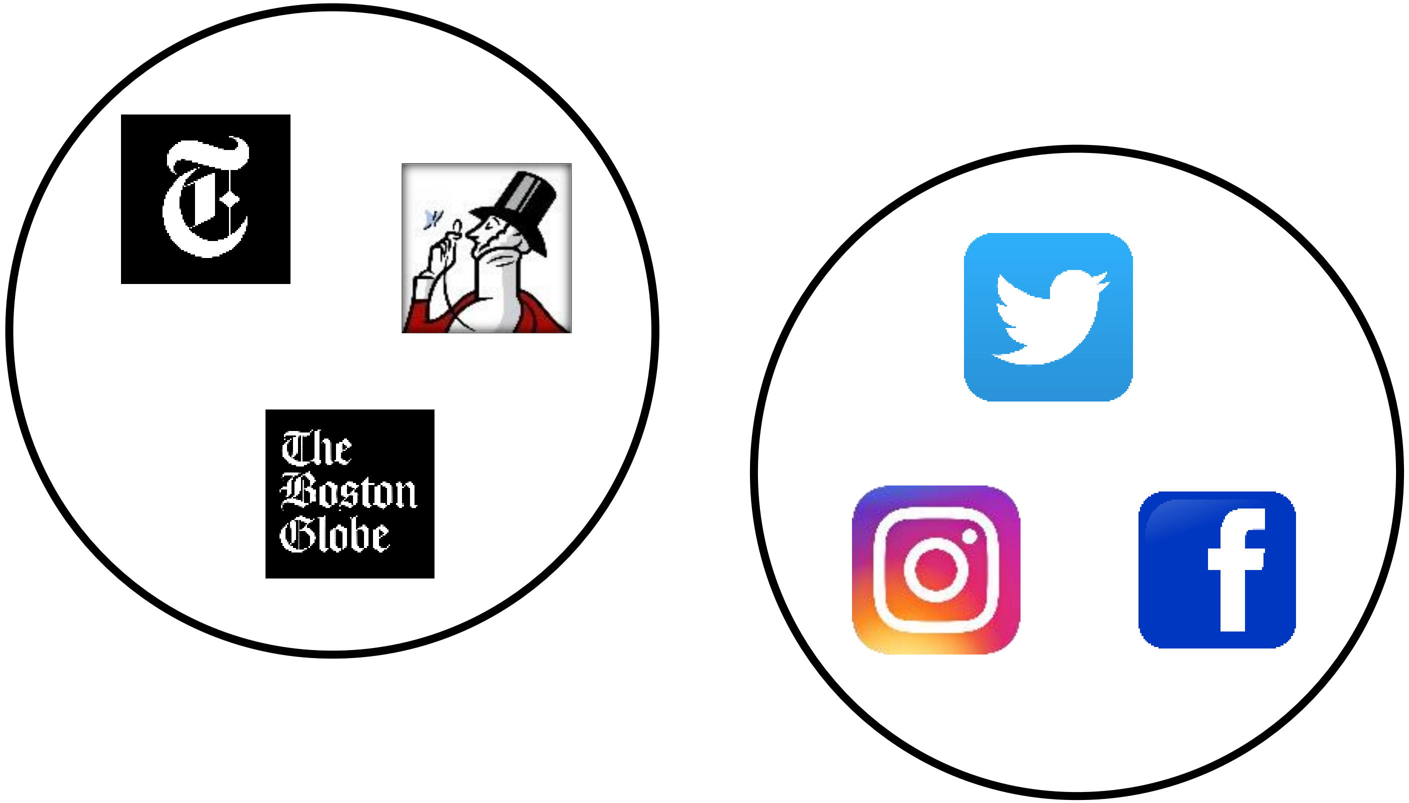
Ellen Vitercik
CMU, Computer Science Department

Microsoft Research New England, Machine Learning Lunch
November 15, 2017

Joint work with Nina Balcan, Colin White, and Vaishnavh Nagarajan
Published in COLT 2017

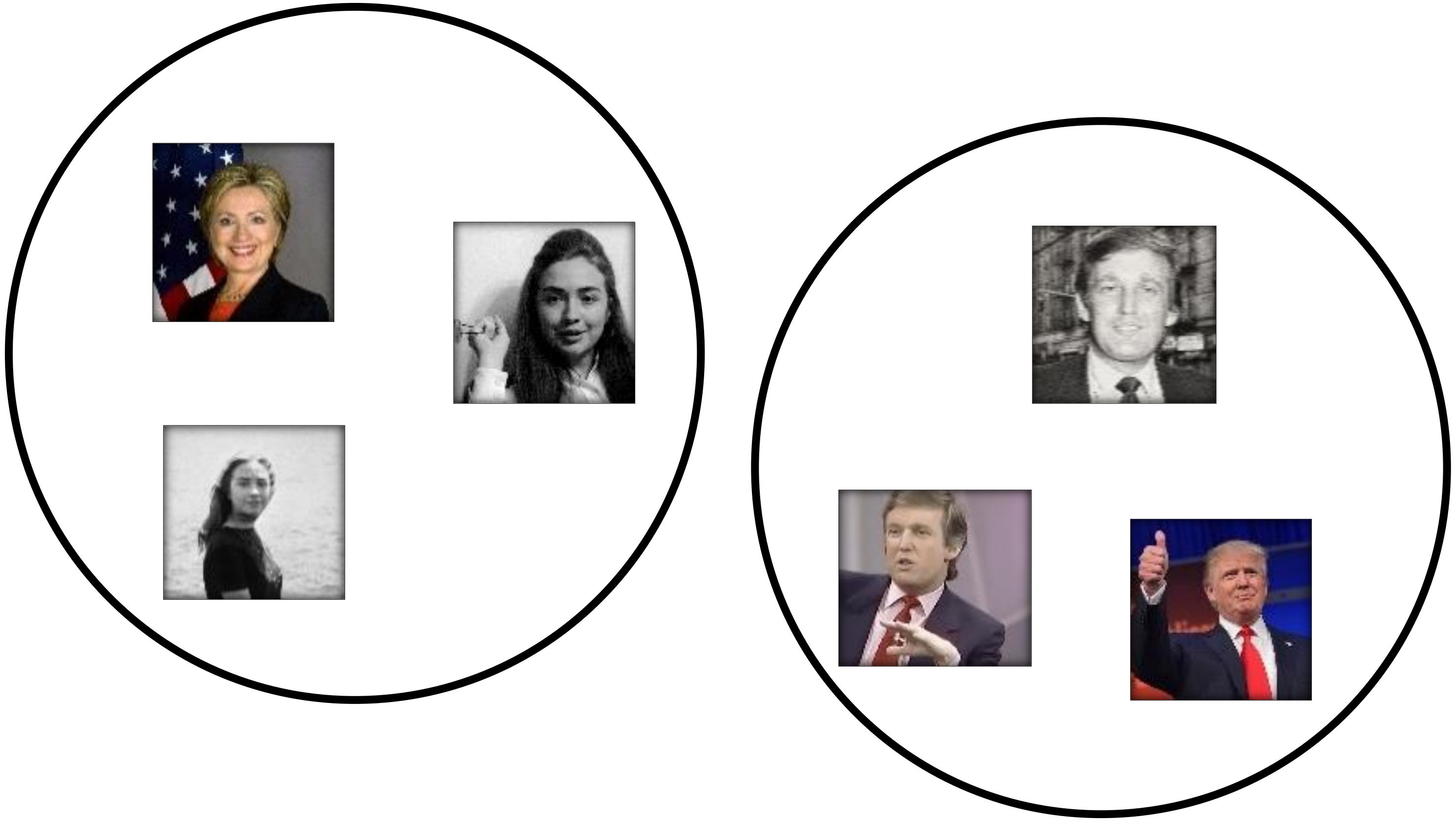
Clustering is a general technique applied in diverse settings.

We can cluster web pages by topic.



Clustering is a general technique applied in diverse settings.

We can cluster images by subject.

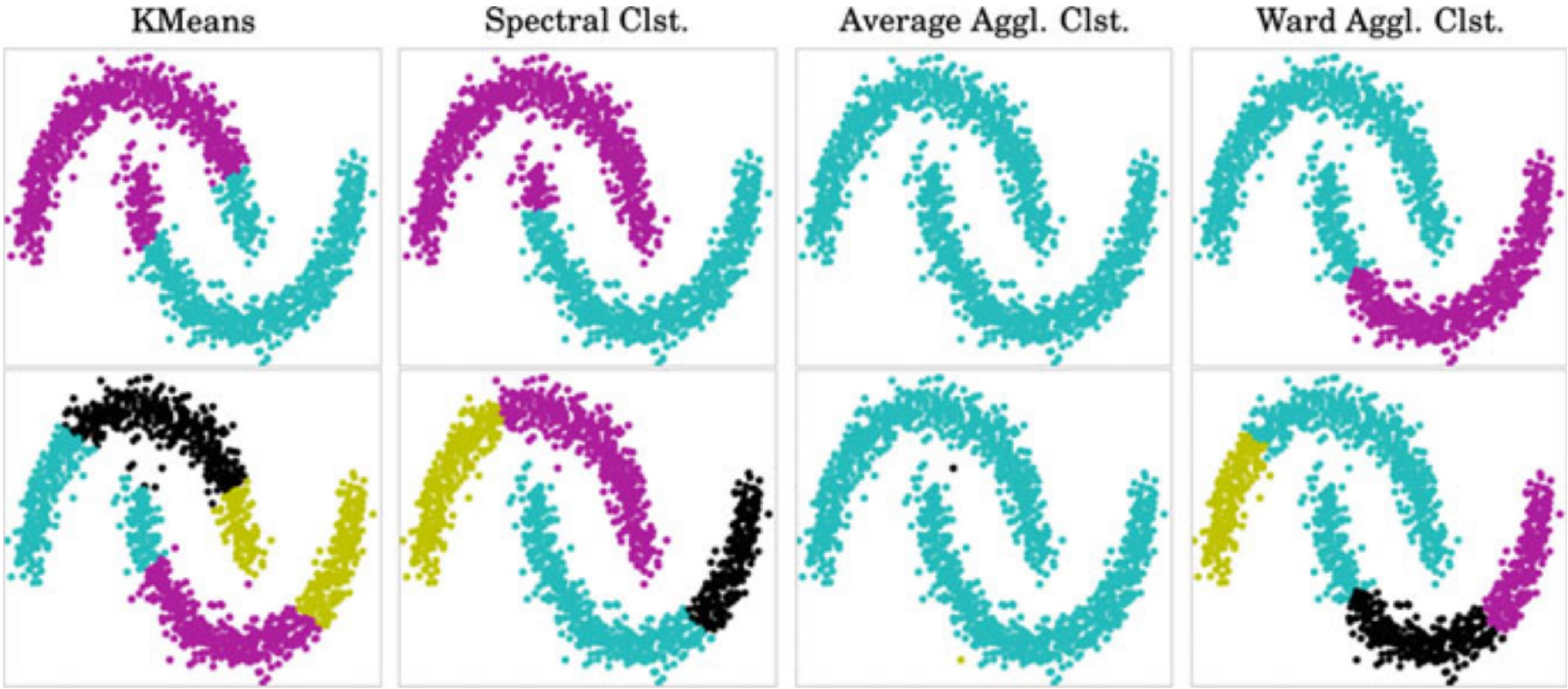


Clustering is a general technique applied in diverse settings.

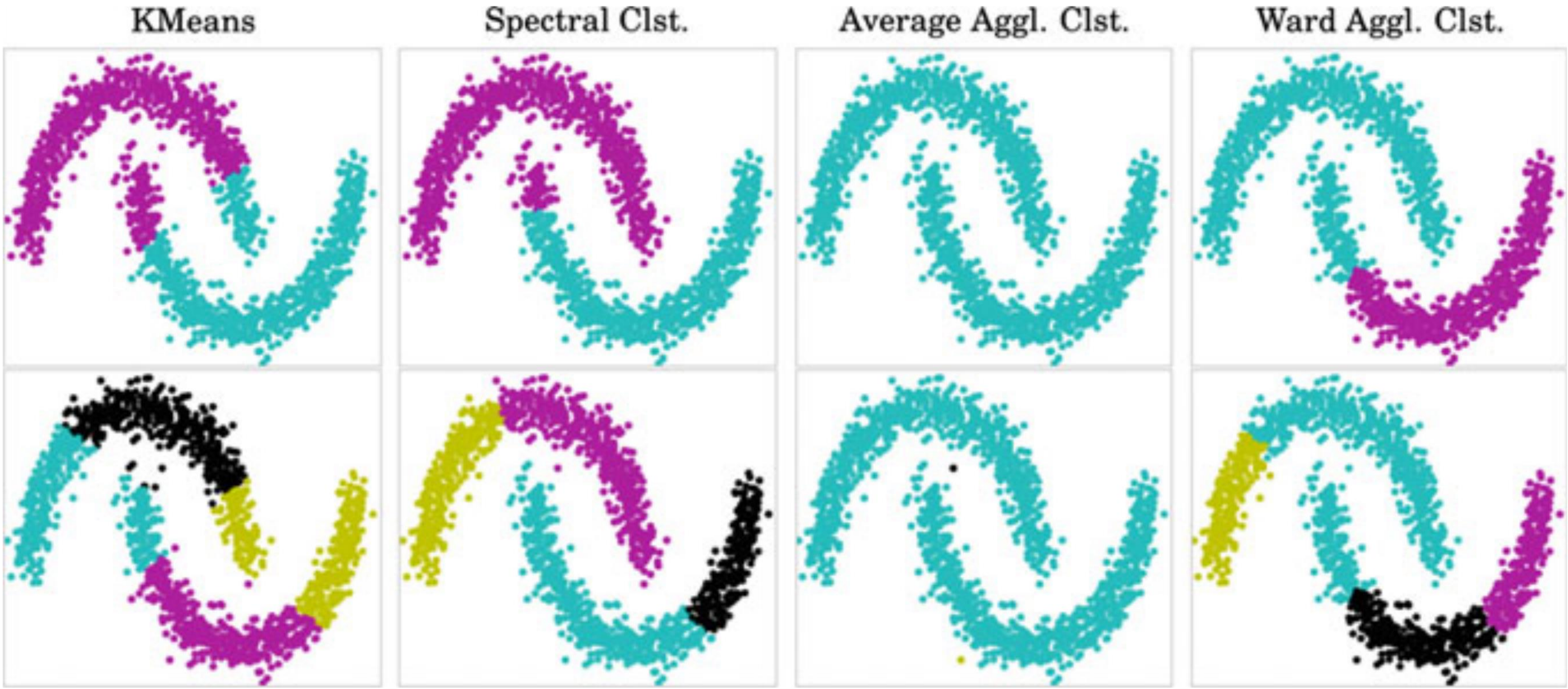
We can cluster residences in towns to determine optimal locations for fire stations.

Like many real-world problems, clustering is NP-hard.

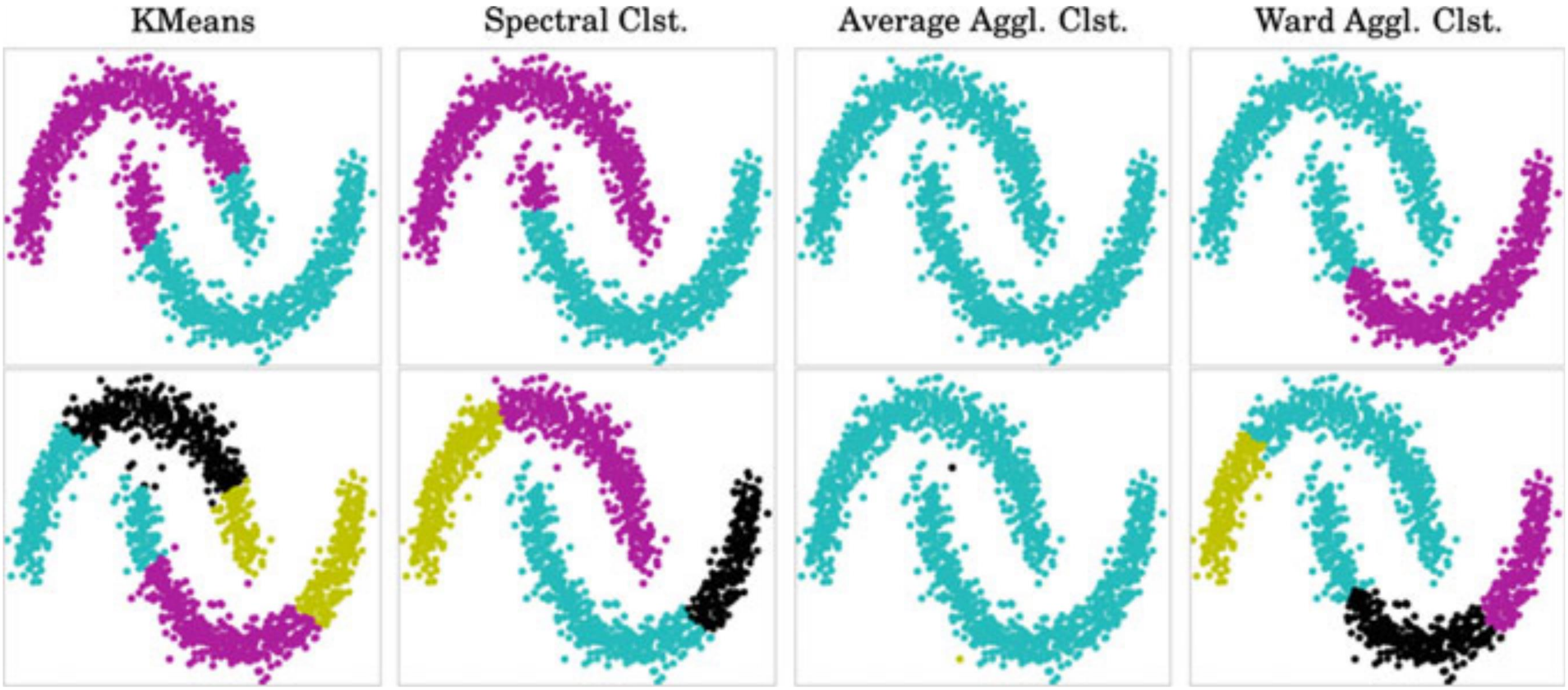
There are many approximation and heuristic algorithms that work well for some problems and poorly for others.



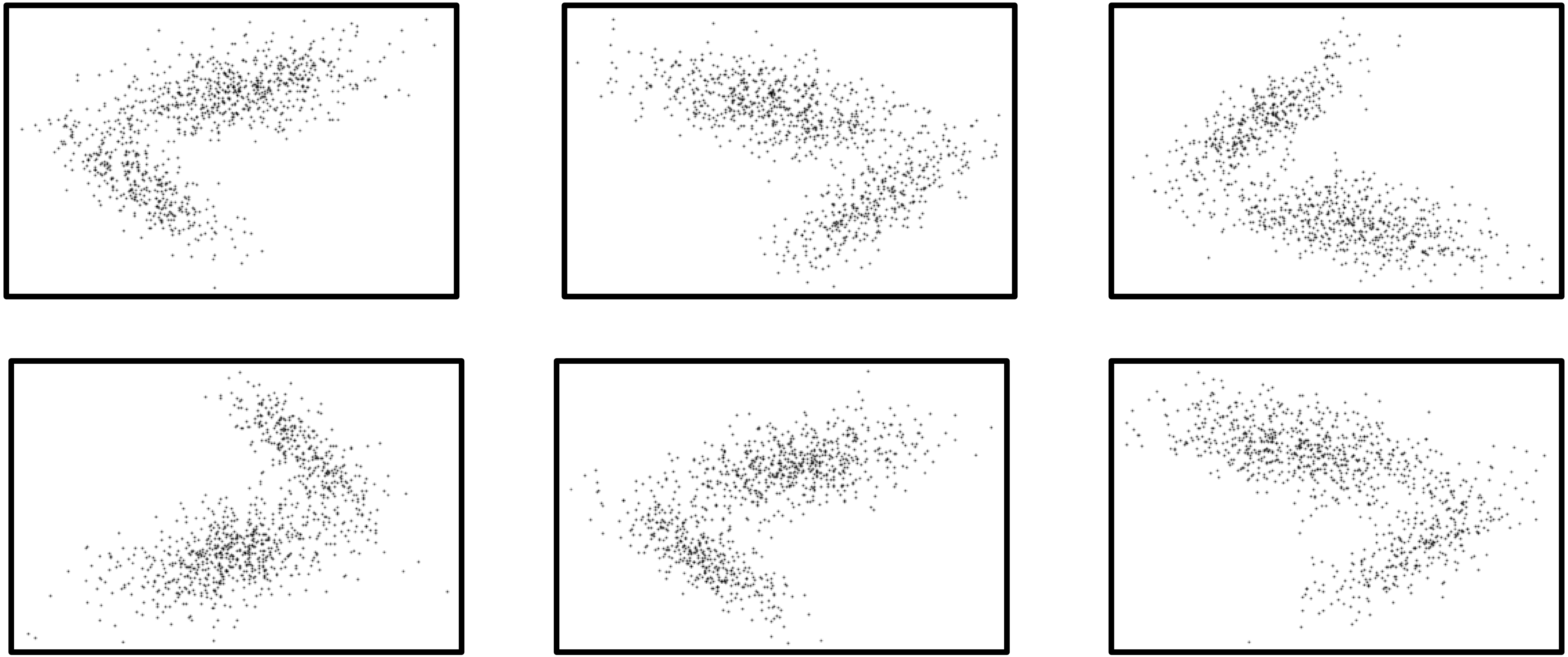
For a given application domain, how do we know which algorithm to use?

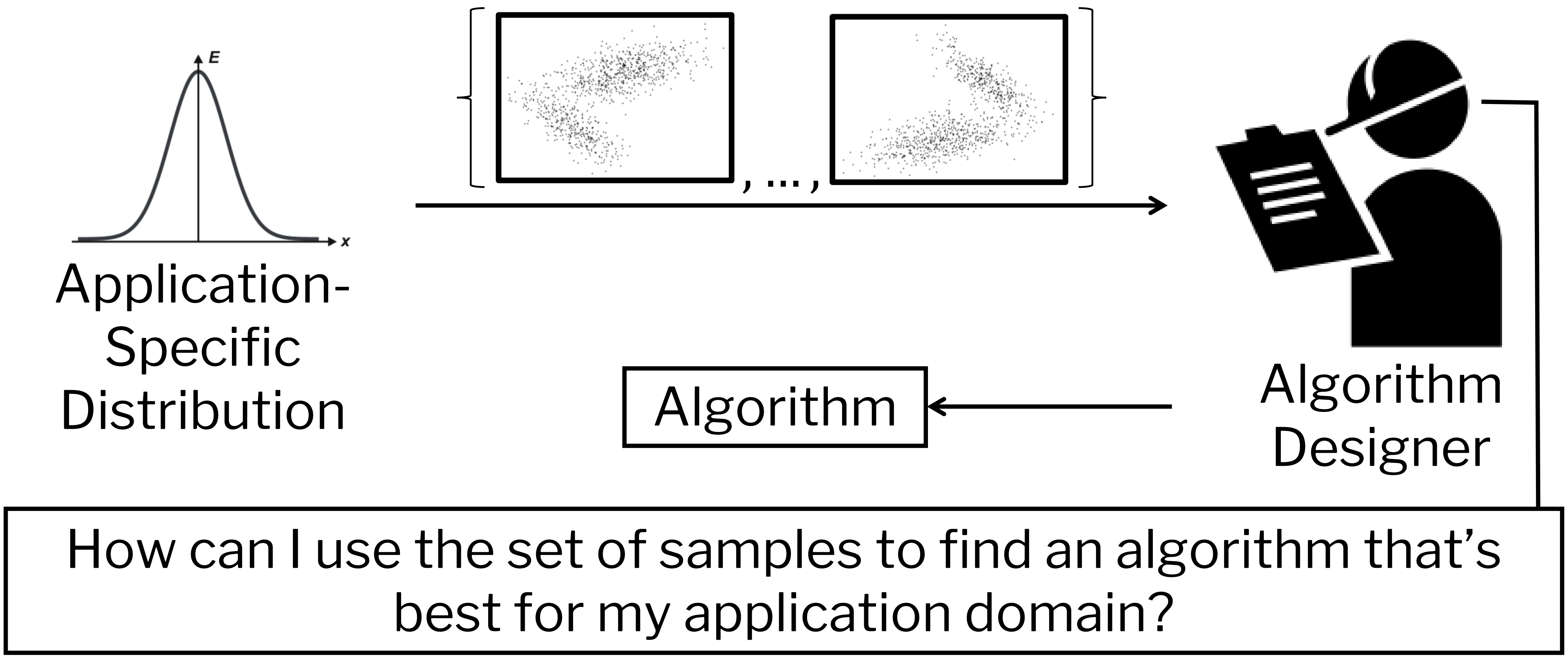


We could compare worst-case guarantees, but this won't help if worst-case instances don't appear in the application domain.

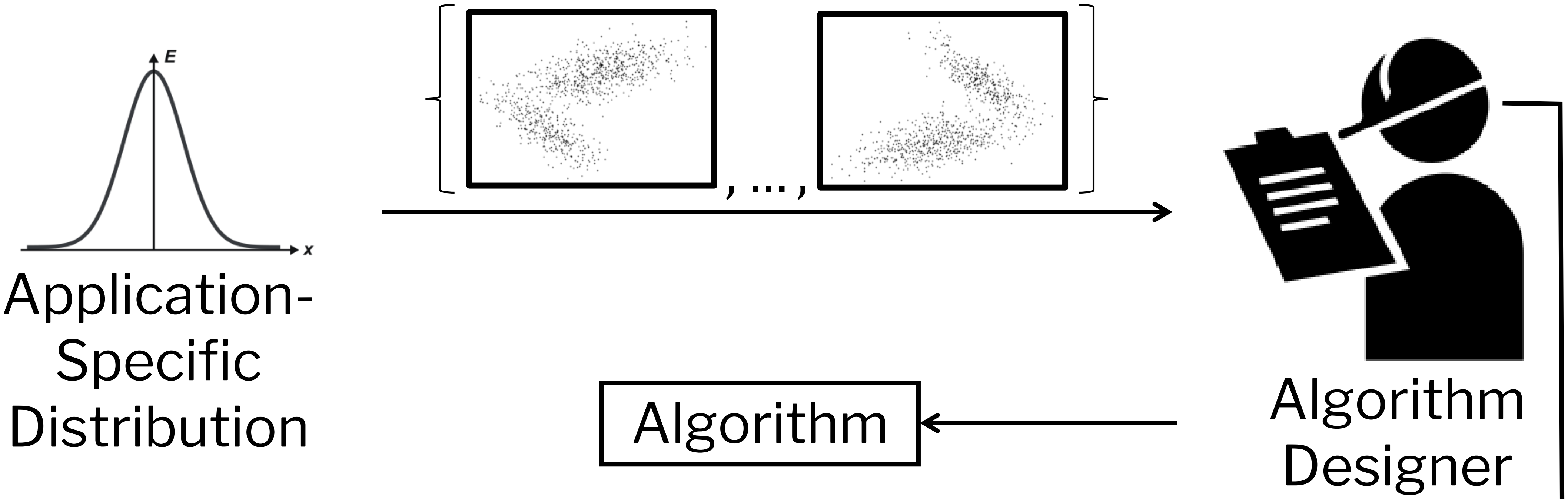


Given a set of typical problem instances from our application domain, we can **learn** the best algorithm for that domain.



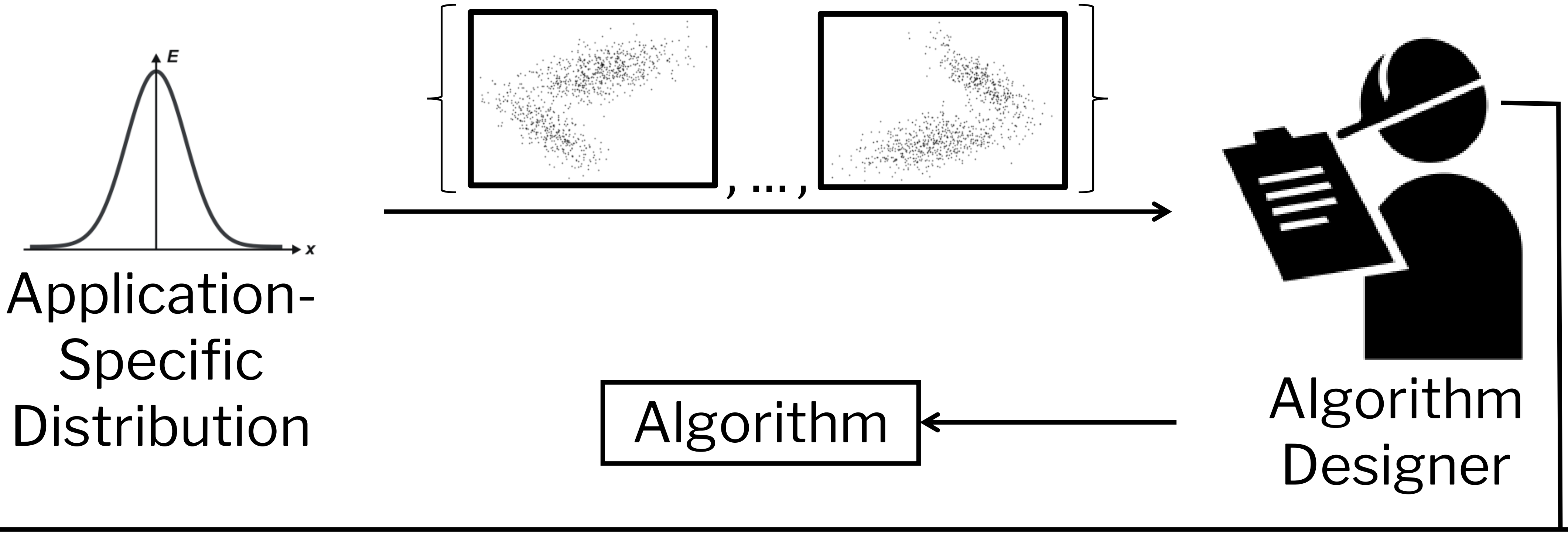

E
x
Application-Specific Distribution
, … ,
Algorithm Designer
Algorithm
How can I use the set of samples to find an algorithm that's best for my application domain?

This model has been studied in applied communities (e.g. [Hutter et al. ‘09]).



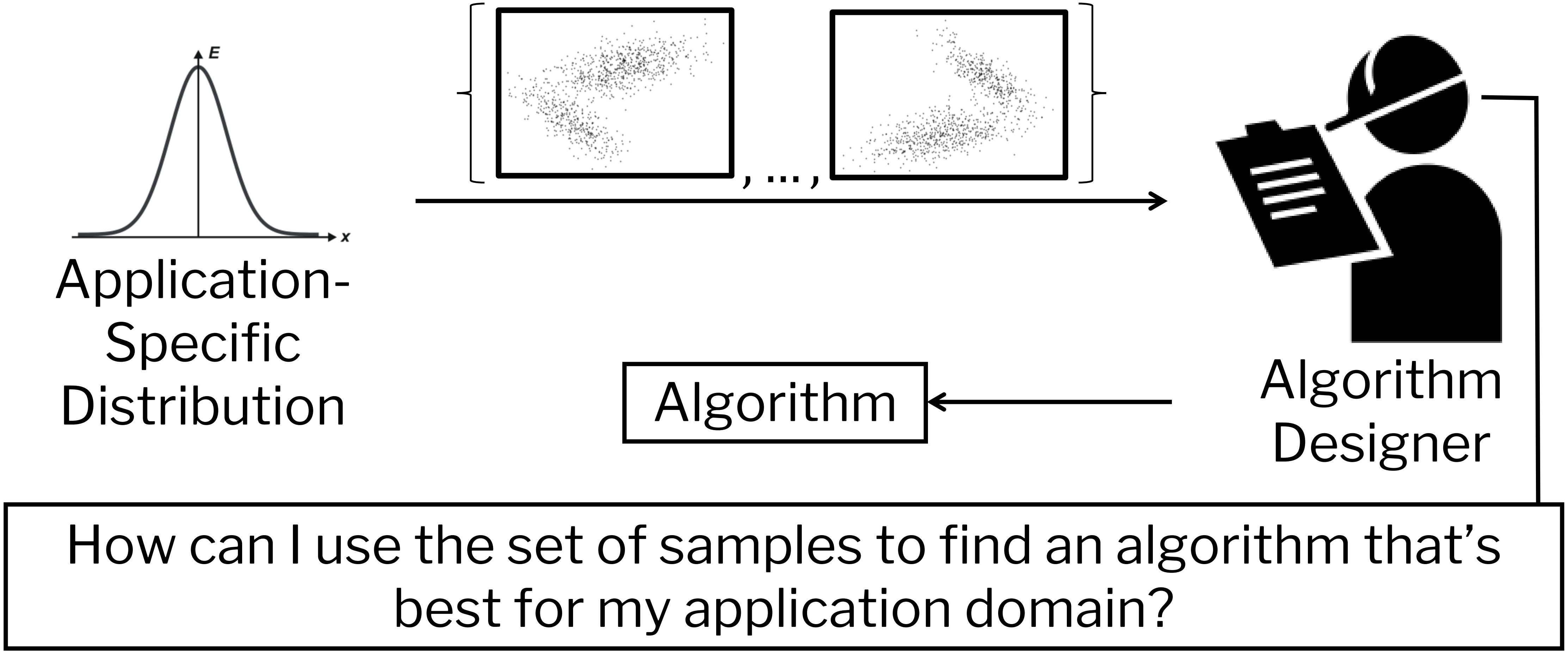
How can I use the set of samples to find an algorithm that’s best for my application domain?

This model has been studied from a theoretical perspective [Gupta and Roughgarden '16].



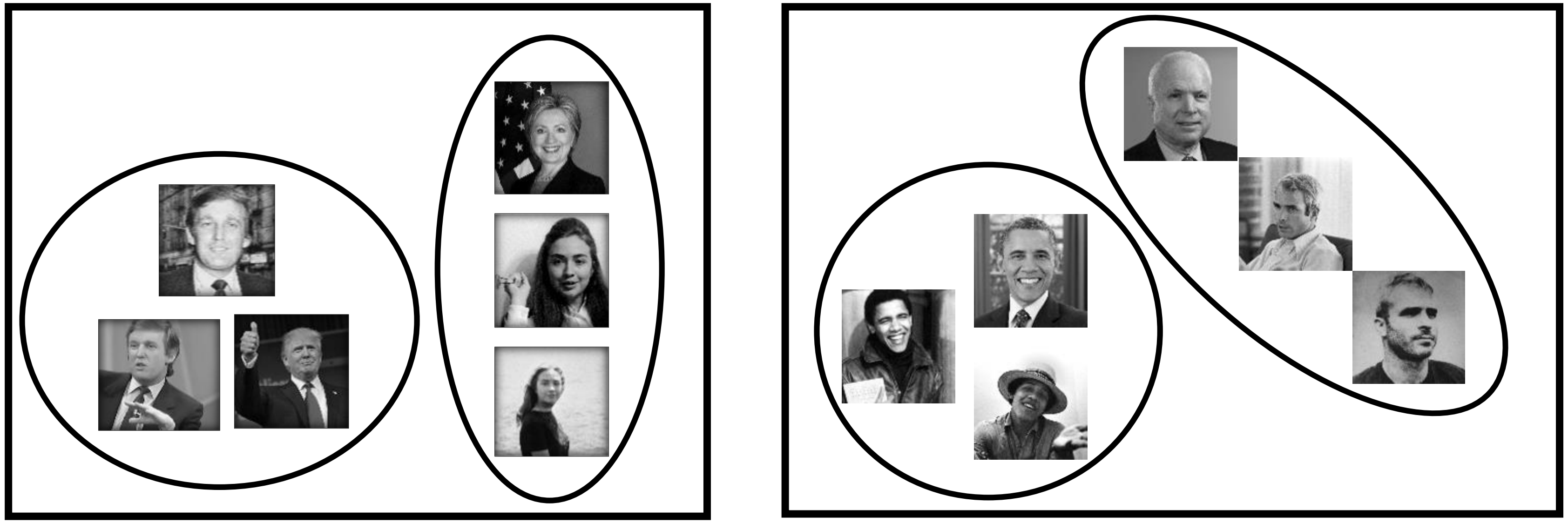
How can I use the set of samples to find an algorithm that's best for my application domain?

Several works beyond Gupta and Roughgarden's work have explored algorithm configuration with theoretical guarantees [Li et al. '16, Garg and Kalai '17, Kleinberg, Leyton-Brown, and Lucier '17, Cohen-Addad and Kanade '17]


E
x
Application-Specific Distribution
, … ,
Algorithm
Algorithm Designer
How can I use the set of samples to find an algorithm that's best for my application domain?

1. Fix a class of (clustering) algorithms $\mathcal{A}$

2. Receive a sample of (clustering) problem instances from an unknown distribution $\mathcal{D}$

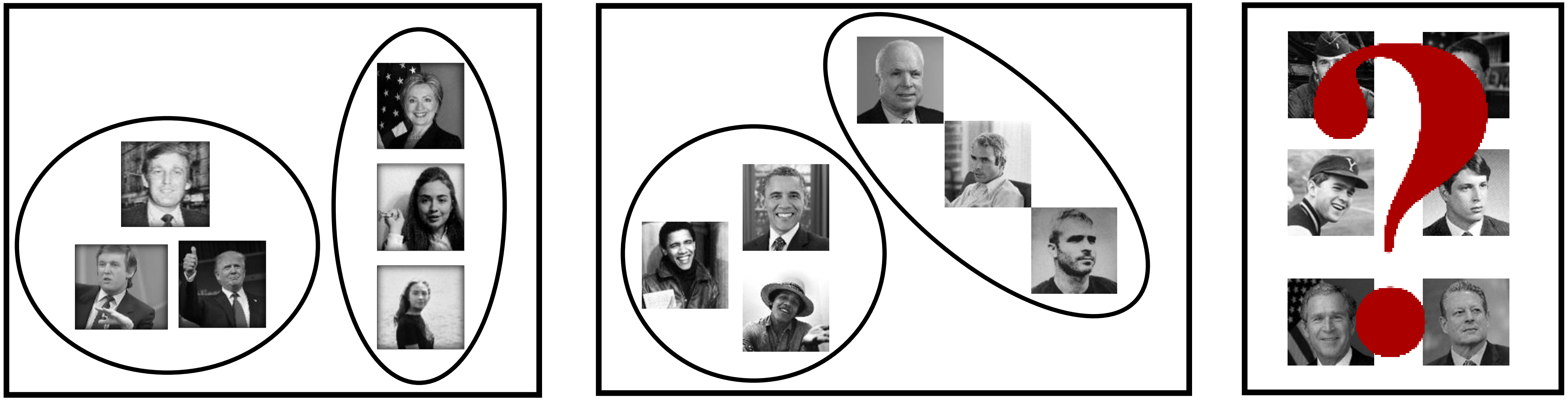


3. Find the algorithm A^* in $\mathcal{A}$ that does best on the sample

"Best" could mean closest to ground truth, smallest k-means objective, etc.

1. What should the class $\mathcal{A}$ of clustering algorithms be?

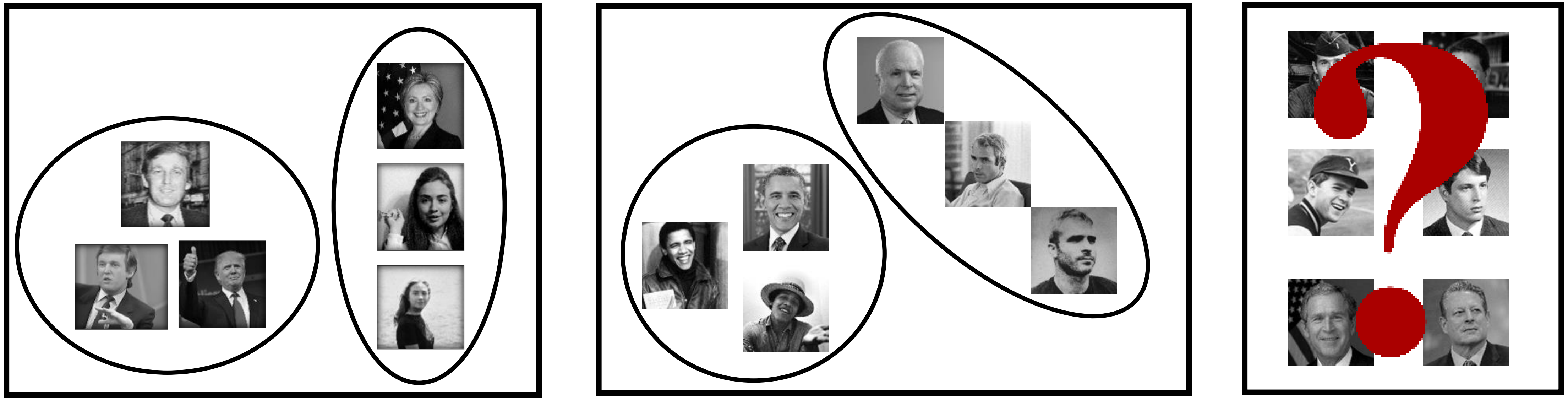
2. How do we find the empirically optimal algorithm A^* in $\mathcal{A}$?



3. Will the performance of A^* generalize to the distribution?

1. What should the class $\mathcal{A}$ of clustering algorithms be?

2. How do we find the empirically optimal algorithm A^* in $\mathcal{A}$?



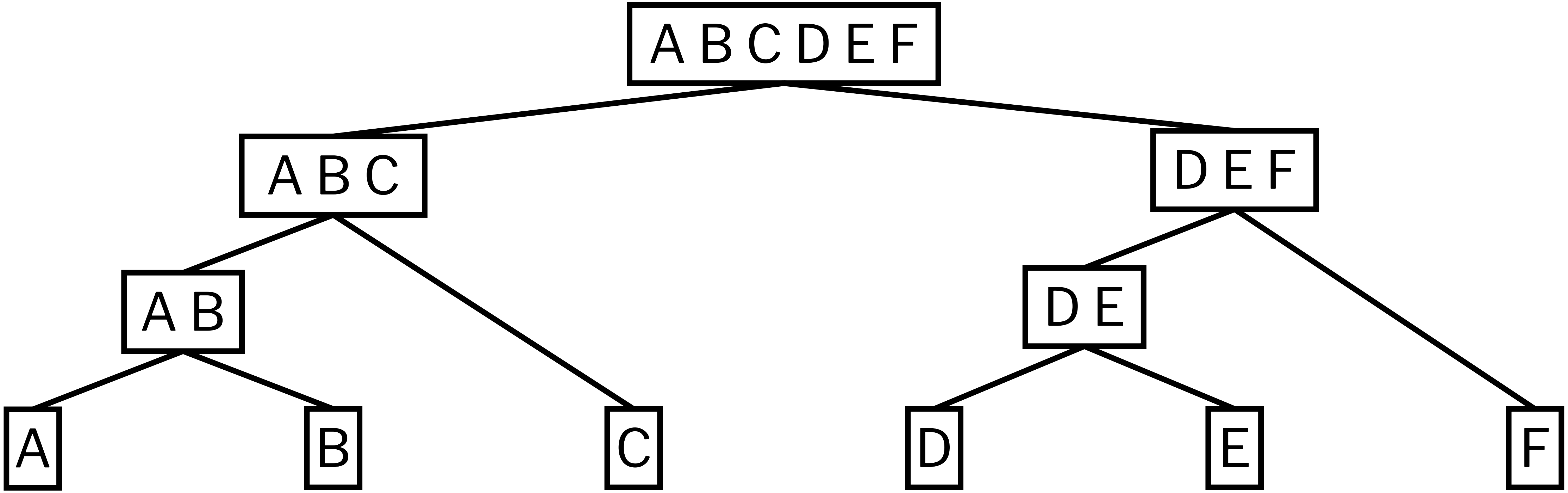
3. A^* has high performance over the sample, but will it have high performance in expectation over $\mathcal{D}$?

1. Clustering algorithm configuration
   a. What should the class of clustering algorithms be?
   b. How do we find the empirically optimal algorithm A^*?
   c. Will the performance of A^* generalize to the distribution?

2. Integer quadratic programming algorithm configuration

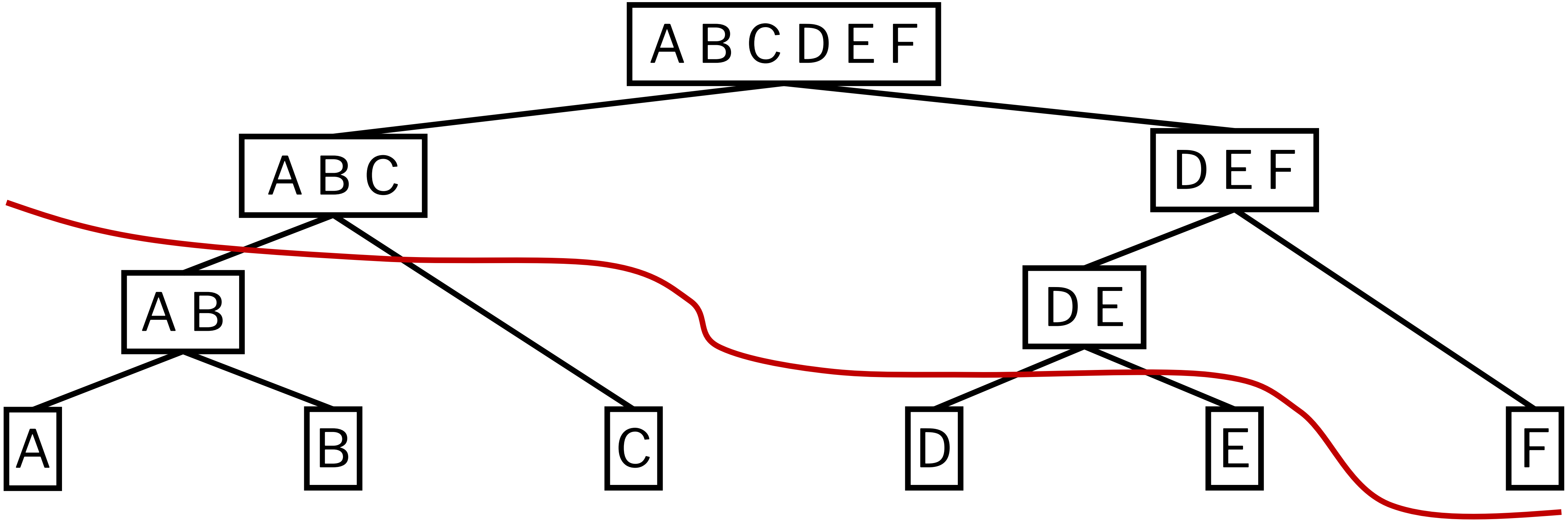
3. Ongoing work

1. Use a linkage-based algorithm to organize data into a hierarchy (tree) of clusters



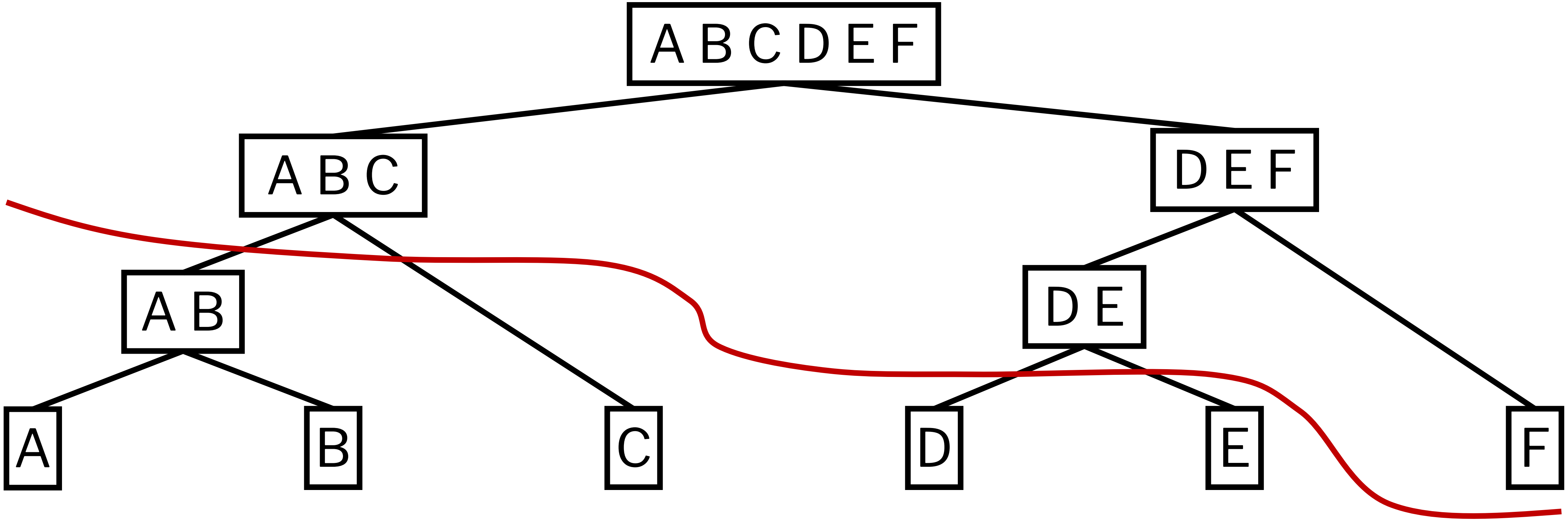
1. Use a linkage-based algorithm to organize data into a hierarchy (tree) of clusters

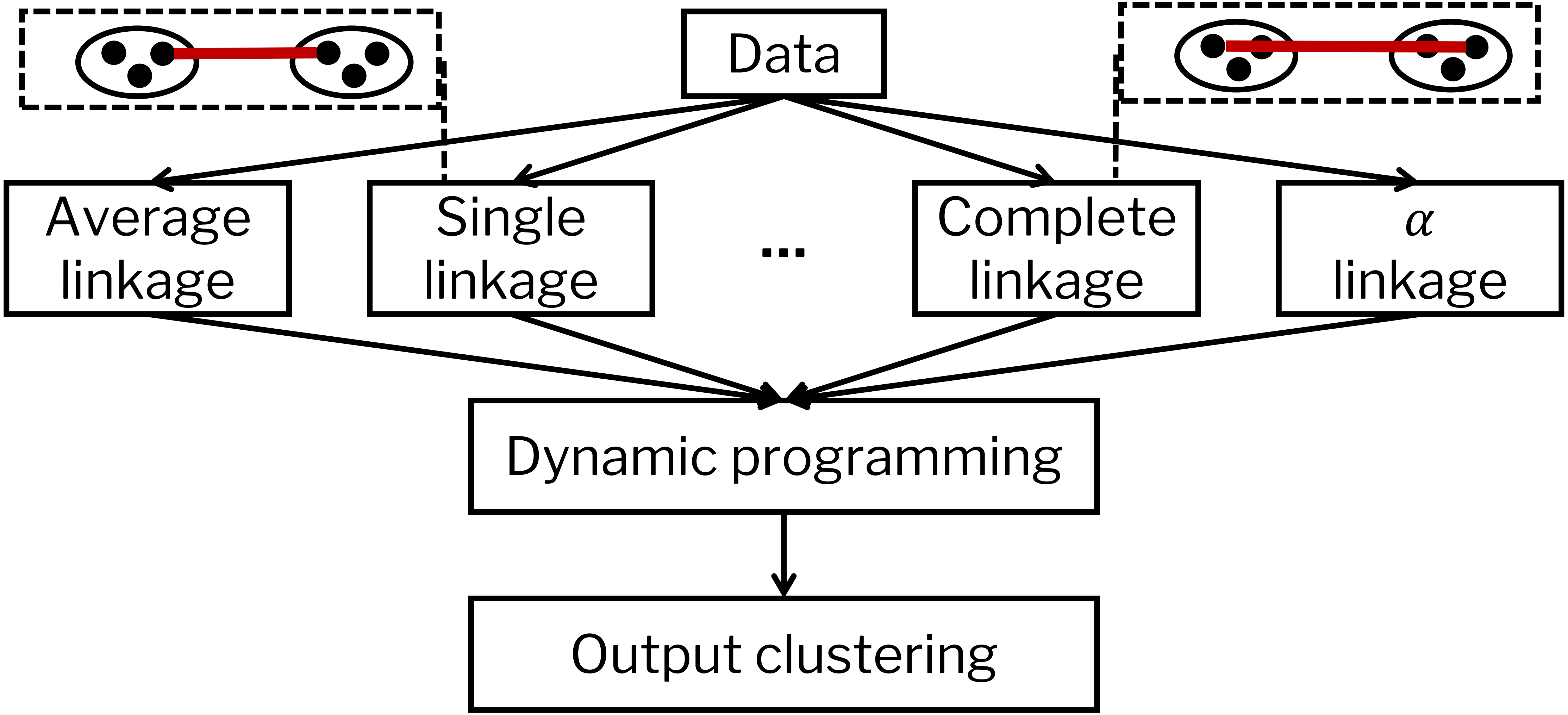
2. Perform dynamic programming over this tree to identify a pruning corresponding to the best clustering

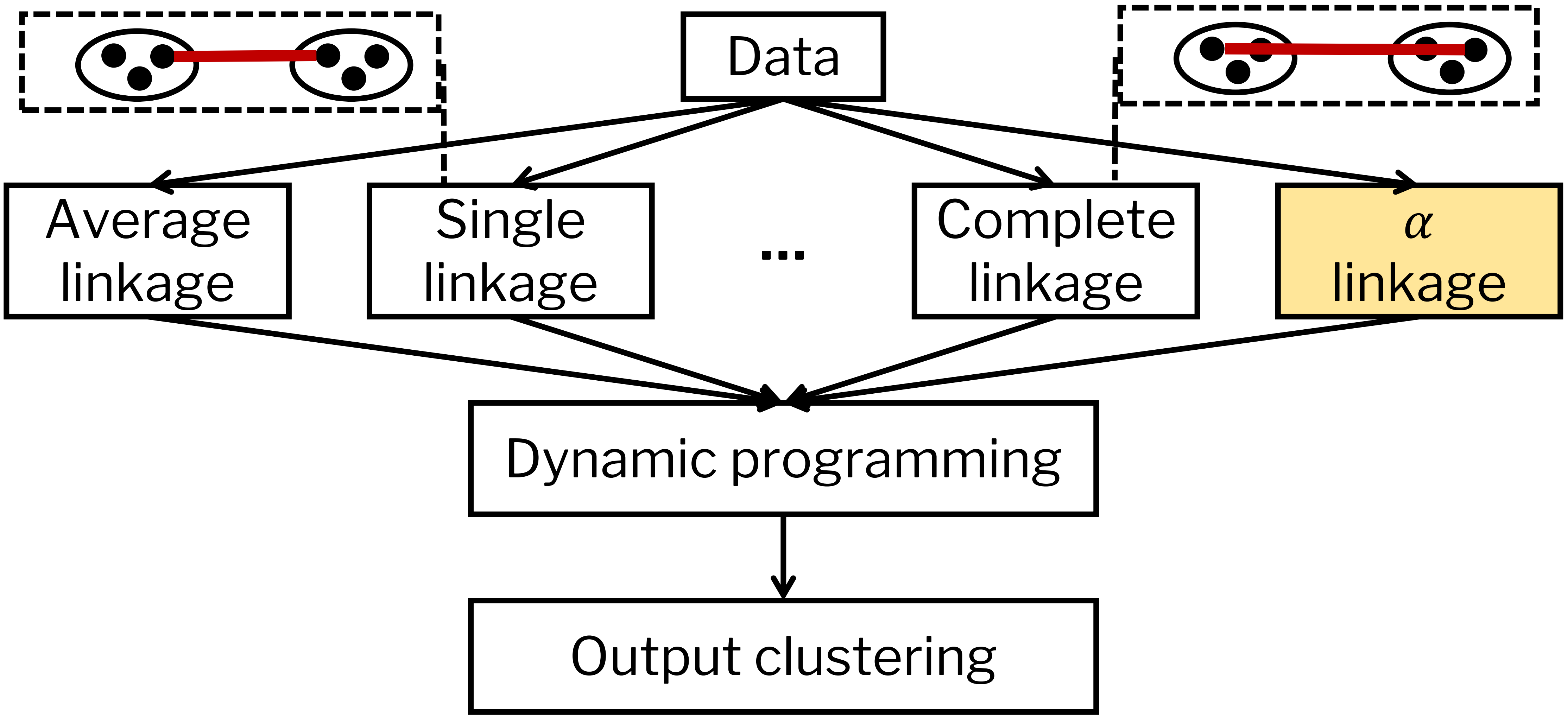


1. Use a **linkage-based algorithm** to organize data into a hierarchy (tree) of clusters

2. Perform dynamic programming over this tree to identify a pruning corresponding to the best clustering



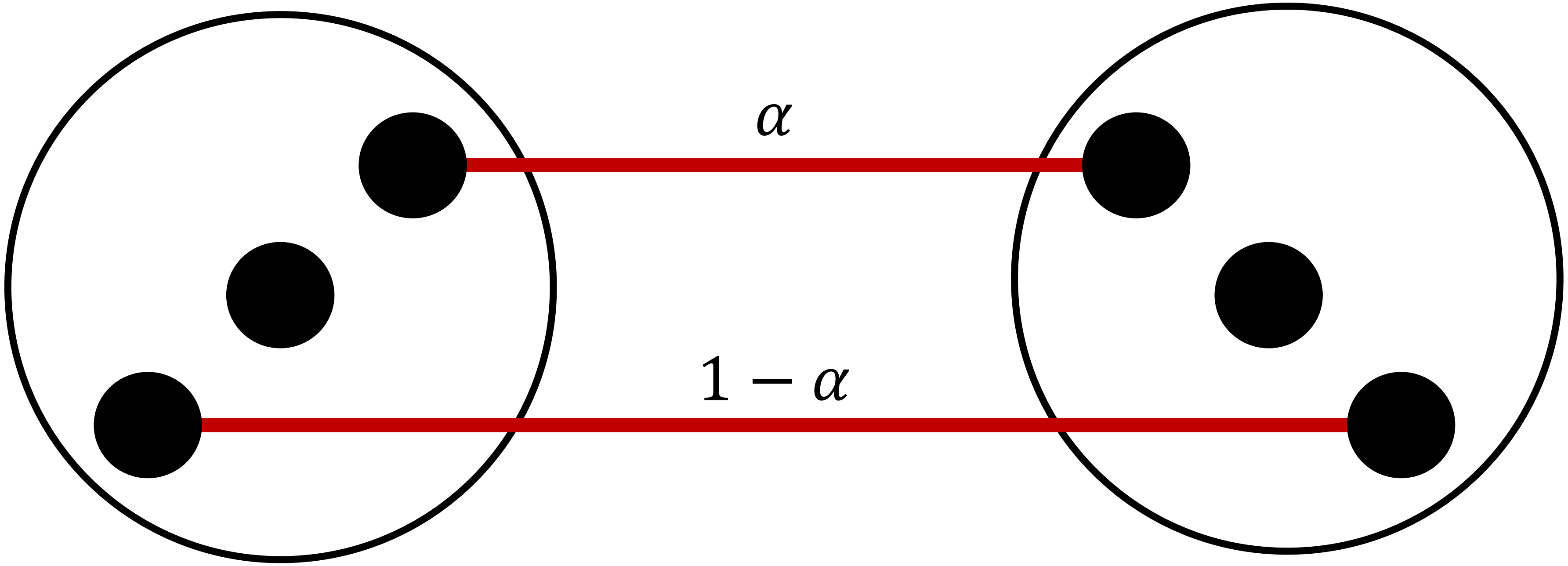

Data
Average linkage
Single linkage
...
Complete linkage
α linkage
Dynamic programming
Output clustering



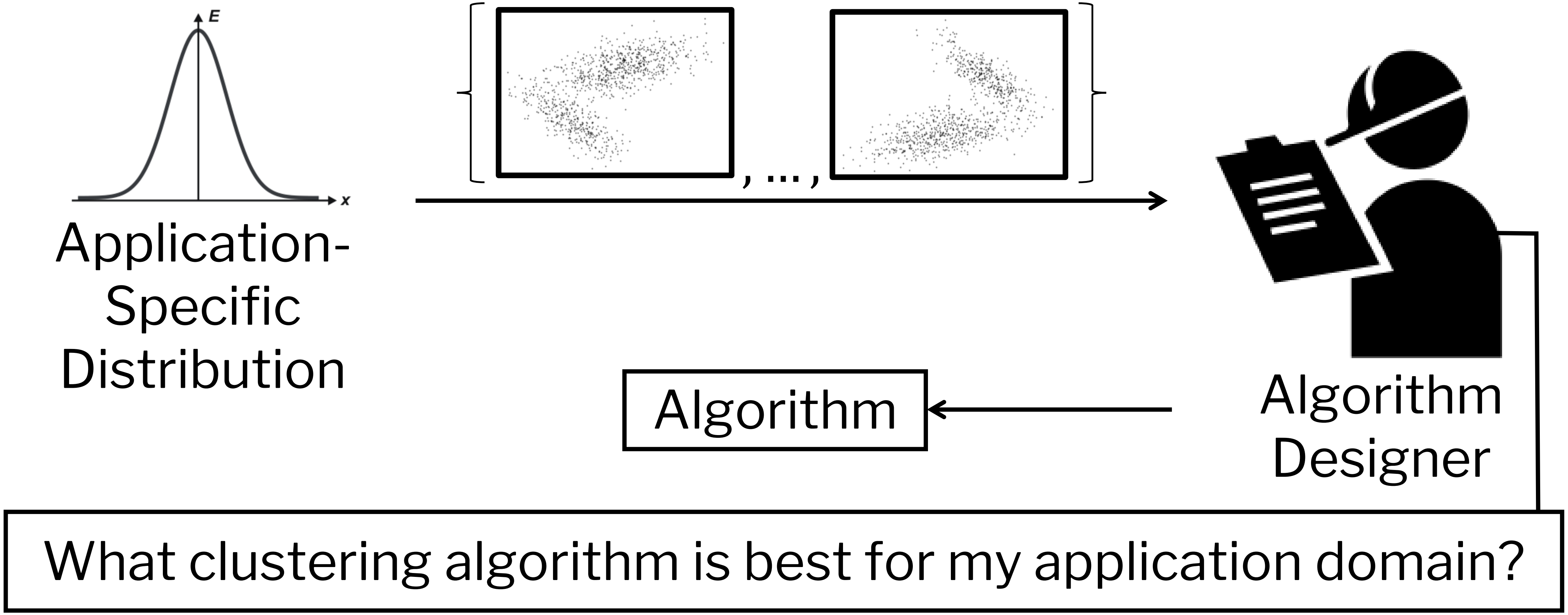
Data
Average linkage
Single linkage
...
Complete linkage
α linkage
Dynamic programming
Output clustering

α-linkage: Merge $\mathcal{N}_i$ and $\mathcal{N}_j$ if they minimize

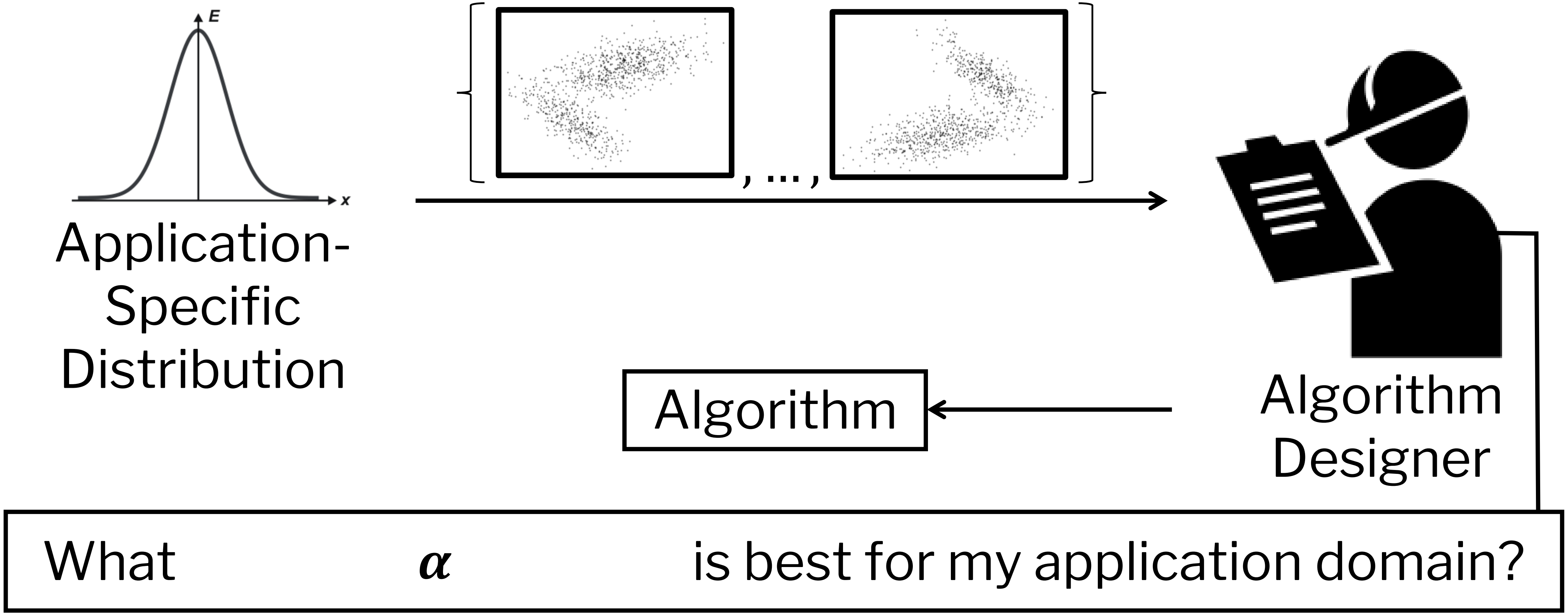
$$\min_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \left(d(p,q)\right)^{\alpha} + \max_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \left(d(p,q)\right)^{1-\alpha}$$



Varying α interpolates between complete and single linkage.

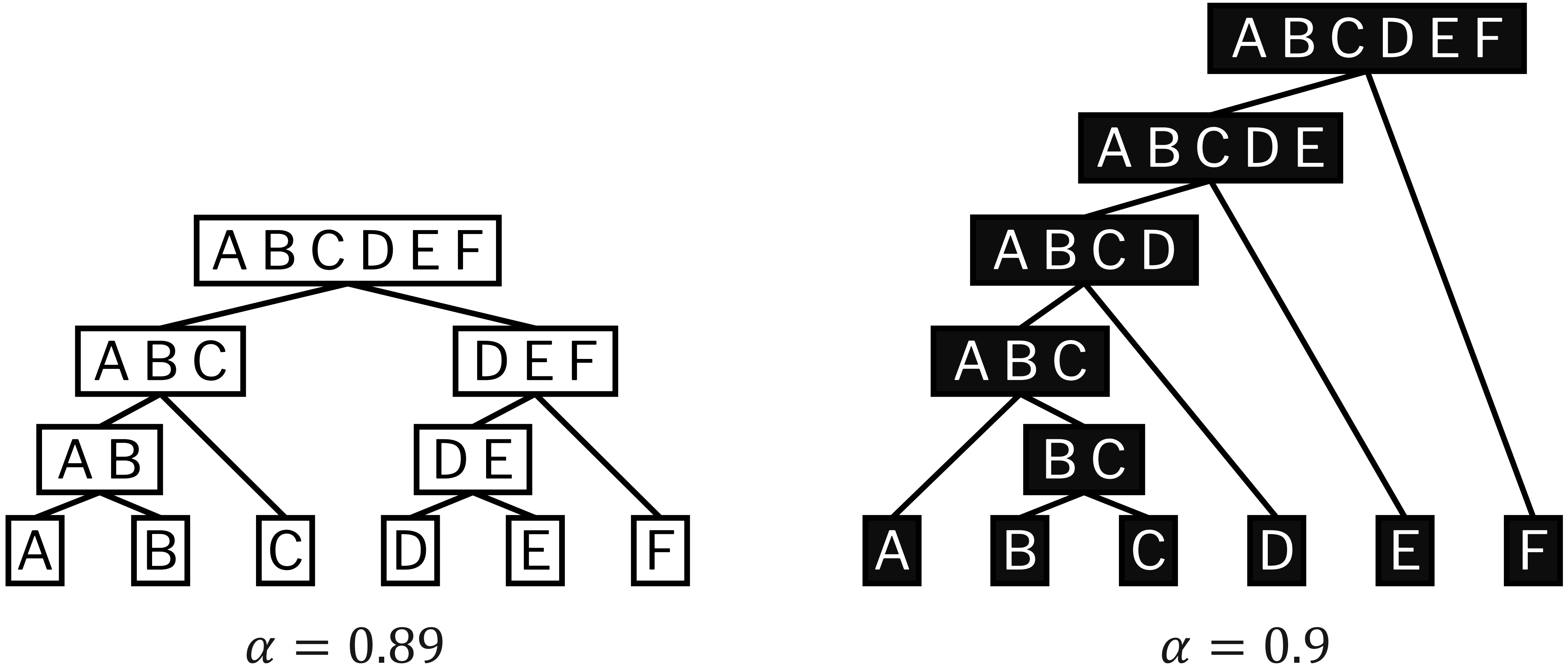


E
x
Application-Specific Distribution
, ... ,
Algorithm
Algorithm Designer
What clustering algorithm is best for my application domain?


E
x
Application-Specific Distribution
, … ,
Algorithm
Algorithm Designer
What α is best for my application domain?

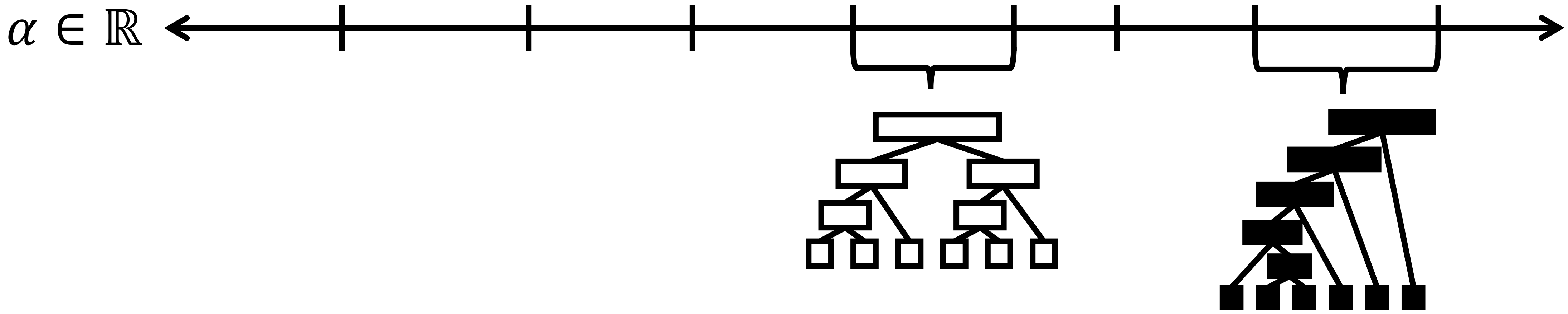
1. Clustering algorithm configuration
   a. What should the class of clustering algorithms be?
   b. How do we find the empirically optimal algorithm A^*?
   c. Will the performance of A^* generalize to the distribution?
2. Integer quadratic programming algorithm configuration
3. Ongoing work

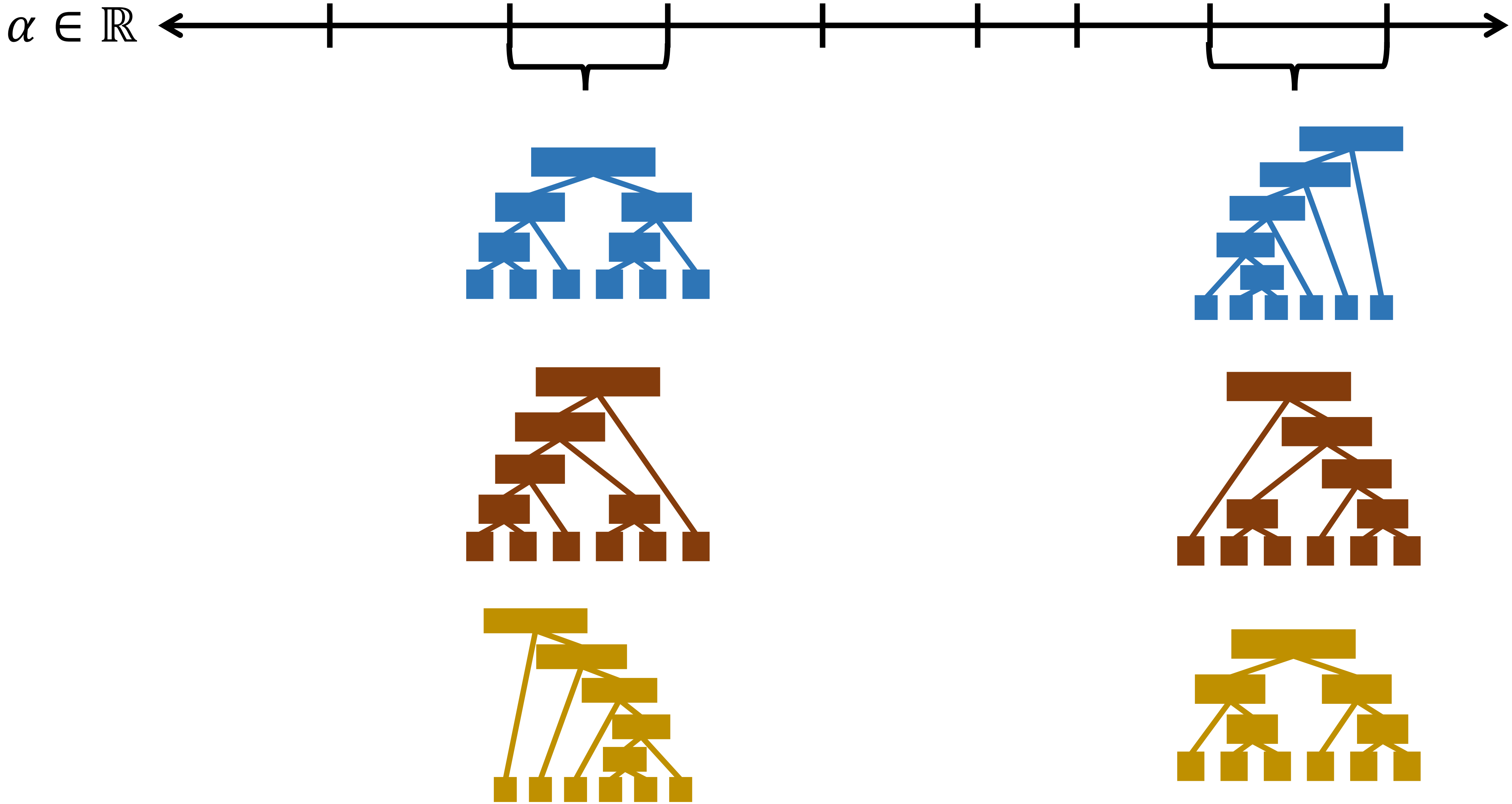
Key challenge: Neighboring α values could result in different trees, and therefore very different performance costs.



We explicitly break the real line into a small number of intervals such that **on each sample**:

Two α's from one interval result in the same tree.



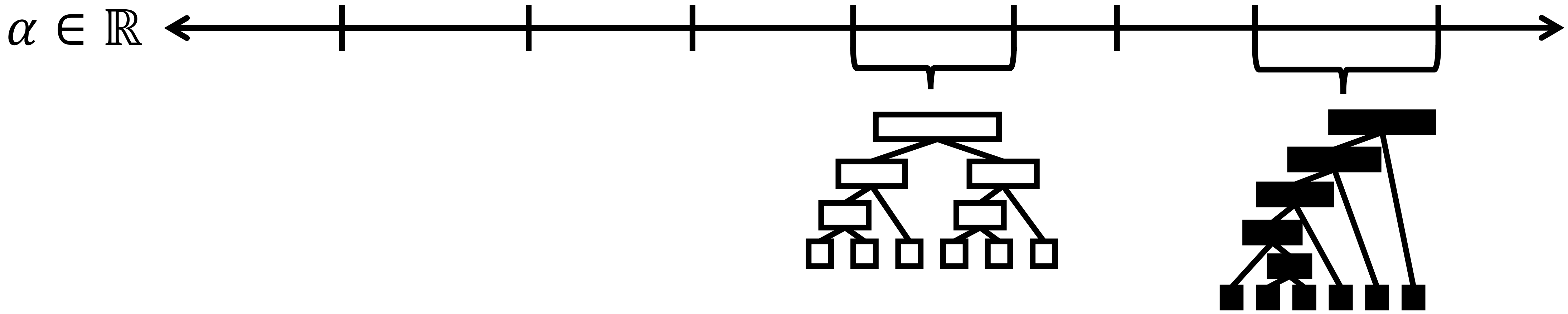

$\alpha \in \mathbb{R}$

We explicitly break the real line into a small number of intervals such that **on each sample**:

Two α's from one interval result in the same tree.

And therefore the same clustering.

And therefore the same performance cost.



We explicitly break the real line into a small number of intervals such that **on each sample**:

Two α's from one interval result in the same tree.

And therefore the same clustering.

And therefore the same performance cost.

Theorem
For a clustering instance of n points, there are $O(n^8)$ intervals such that any two α's from one interval result in the same tree.

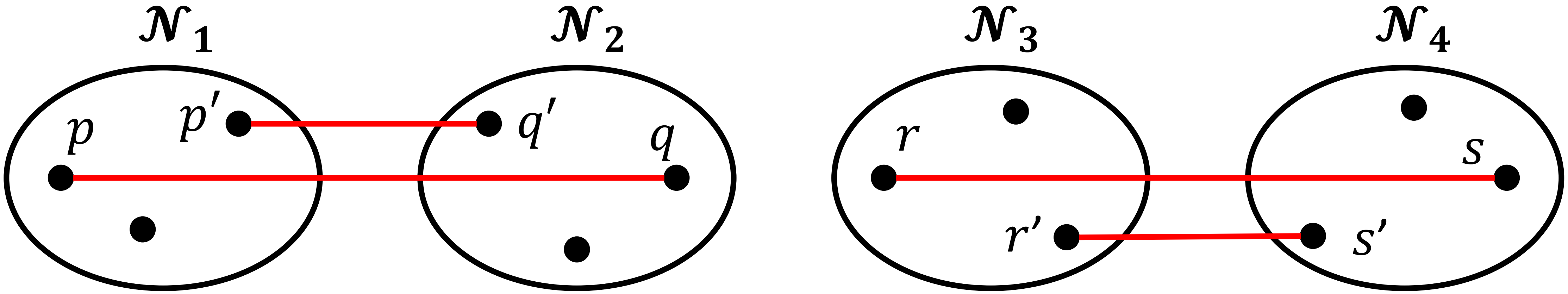
$\boldsymbol{\alpha}$-linkage: Merge $\mathcal{N}_i$ and $\mathcal{N}_j$ if they minimize

$$\min_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \big(d(p,q)\big)^{\alpha} + \max_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \big(d(p,q)\big)^{1-\alpha}$$

Main idea:
Over any α interval, so long as the order in which all pairs of nodes are merged is fixed, then the resulting tree will be invariant.

α-linkage: Merge $\mathcal{N}_i$ and $\mathcal{N}_j$ if they minimize

$$\min_{p\in\mathcal{N}_i, q\in\mathcal{N}_j}\left(d(p,q)\right)^{\alpha} + \max_{p\in\mathcal{N}_i, q\in\mathcal{N}_j}\left(d(p,q)\right)^{1-\alpha}$$



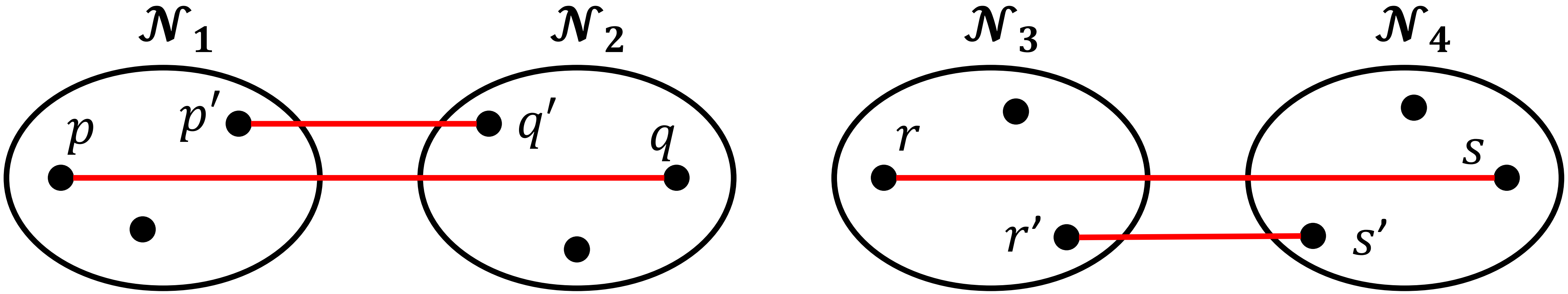
Which will merge first, $\mathcal{N}_1$ and $\mathcal{N}_2$, or $\mathcal{N}_3$ and $\mathcal{N}_4$?

Depends on whether

$$d(p,q)^{1-\alpha} + d(p',q')^{\alpha} \geq d(r,s)^{1-\alpha} + d(r',s')^{\alpha}$$

α-linkage: Merge $\mathcal{N}_i$ and $\mathcal{N}_j$ if they minimize

$$\min_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \left(d(p,q)\right)^{\alpha} + \max_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} \left(d(p,q)\right)^{1-\alpha}$$



Which will merge first, $\mathcal{N}_1$ and $\mathcal{N}_2$, or $\mathcal{N}_3$ and $\mathcal{N}_4$?

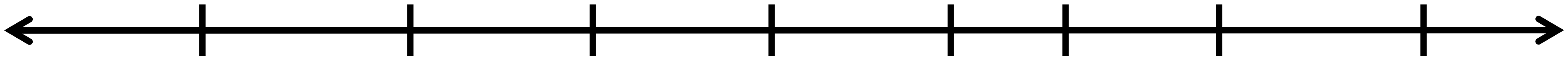
Depends on the sign of

$$d(p,q)^{1-\alpha} + d(p',q')^{\alpha} - d(r,s)^{1-\alpha} - d(r',s')^{\alpha}$$

$$d(p,q)^{1-\alpha} + d(p',q')^{\alpha} - d(r,s)^{1-\alpha} - d(r',s')^{\alpha}$$

has ≤ 4 zeros. We call these 4 zeros **critical α values**.

Sort all $4n^8$ critical α values for all n^8 8-tuples of points.

$\alpha \in \mathbb{R}$ 

For all 8-tuples, the sign of

$$d(p,q)^{1-\alpha} + d(p',q')^{\alpha} - d(r,s)^{1-\alpha} - d(r',s')^{\alpha}$$

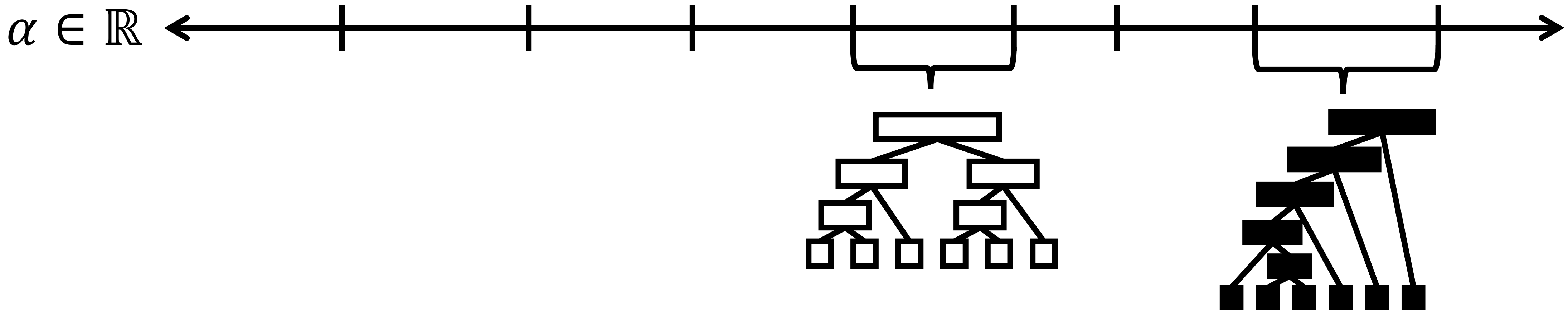
is **fixed** between any two critical α values.

Therefore, the **order of all merges is fixed** between any two critical α values.

Since on a single interval, the order of all merges is fixed, the resulting tree will also be fixed.

Theorem

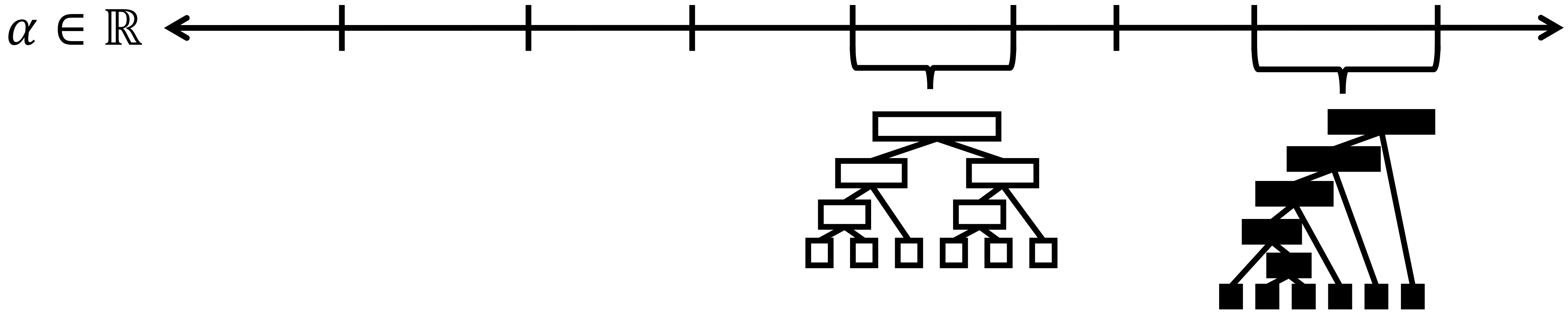
For a clustering instance of n points, there are $O(n^8)$ intervals such that any two α's from one interval result in the same tree.



Algorithm (high level)

1. Solve for all α intervals over the sample
2. Find the α interval with the smallest empirical cost

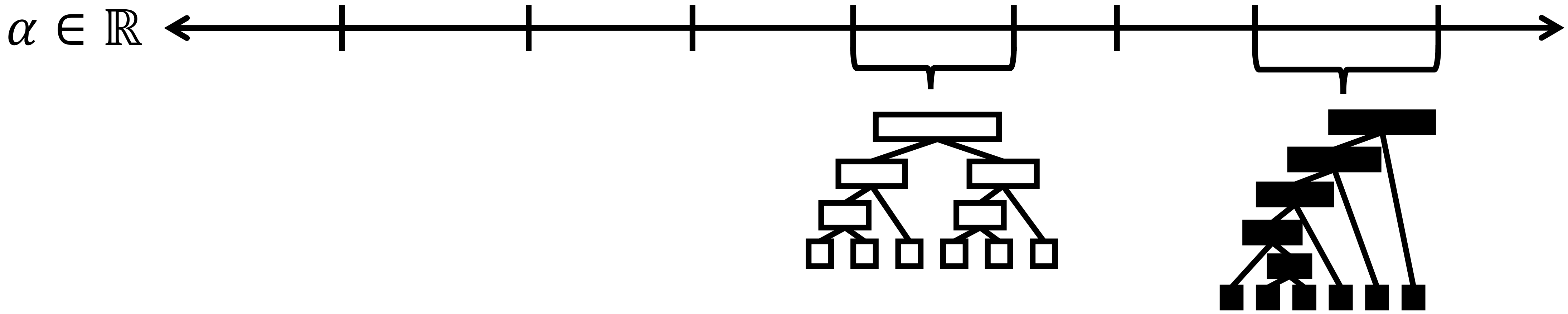
(We prove that any α from that interval will have approximately smallest expected cost)



1. Clustering algorithm configuration
    a. What should the class of clustering algorithms be?
    b. How do we find the empirically optimal algorithm A^*?
    c. Will the performance of A^* generalize to the distribution?
2. Integer quadratic programming algorithm configuration
3. Ongoing work

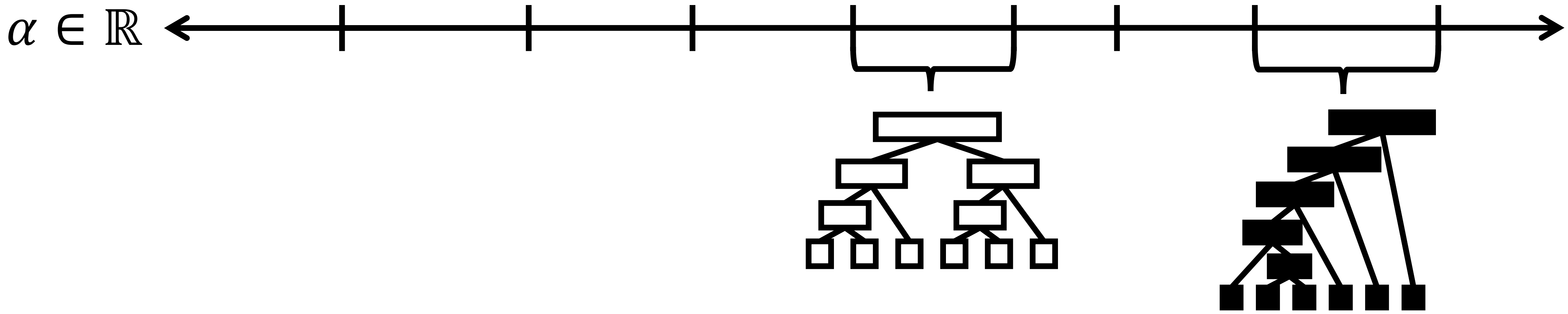
We've shown that over any set of samples, there are only a **small** number of significantly different algorithms.

This implies **low complexity** (think **VC dimension**)



Theorem

Given a sample of $\tilde{O}(1/\epsilon^2)$ clustering problems, with high probability, the expected performance cost of the best α over the sample is ϵ-close to optimal over the distribution.



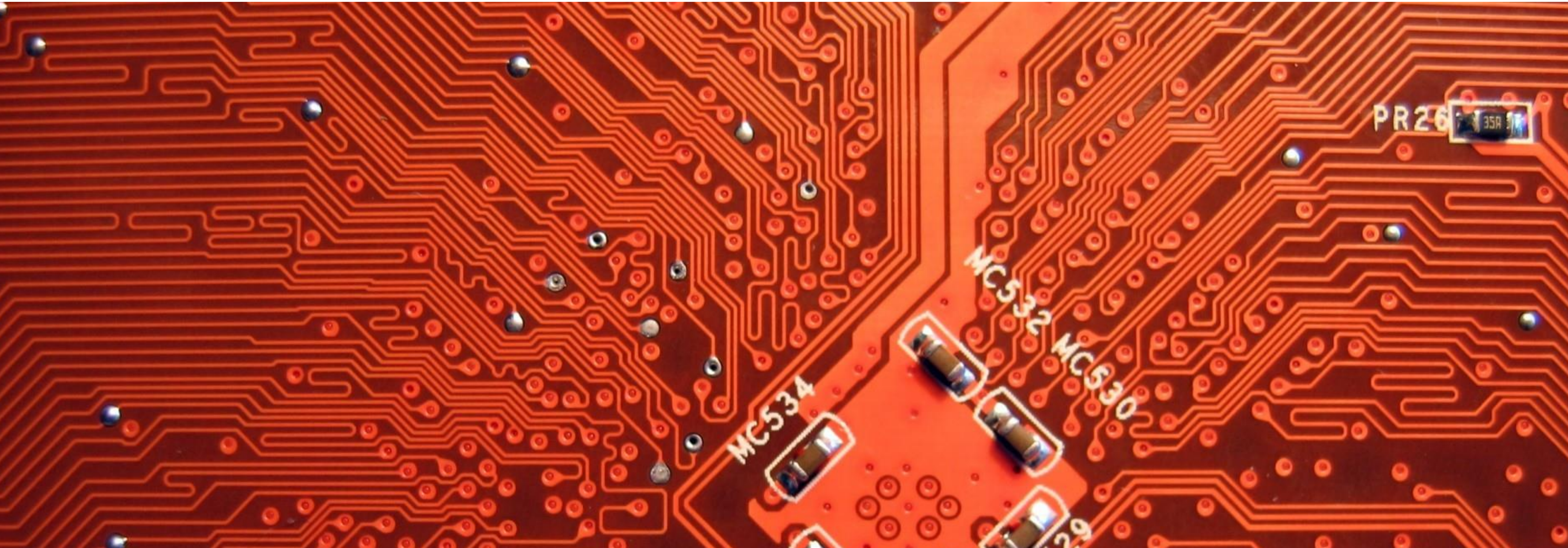
1. Clustering algorithm configuration
   a. What should the class of clustering algorithms be?
   b. How do we find the empirically optimal algorithm A^*?
   c. Will the performance of A^* generalize to the distribution?

2. Integer quadratic programming algorithm configuration

3. Ongoing work

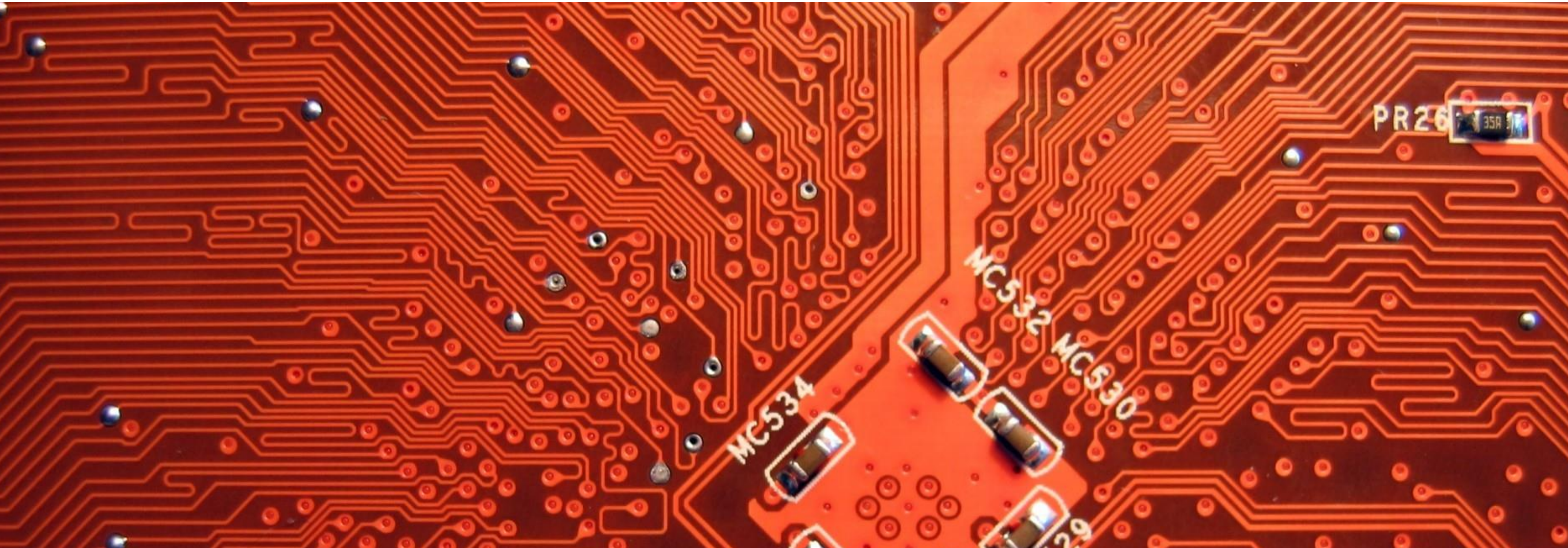
Many classic problems can be formulated as IQPs, including max-cut, max-2sat, and correlation clustering.

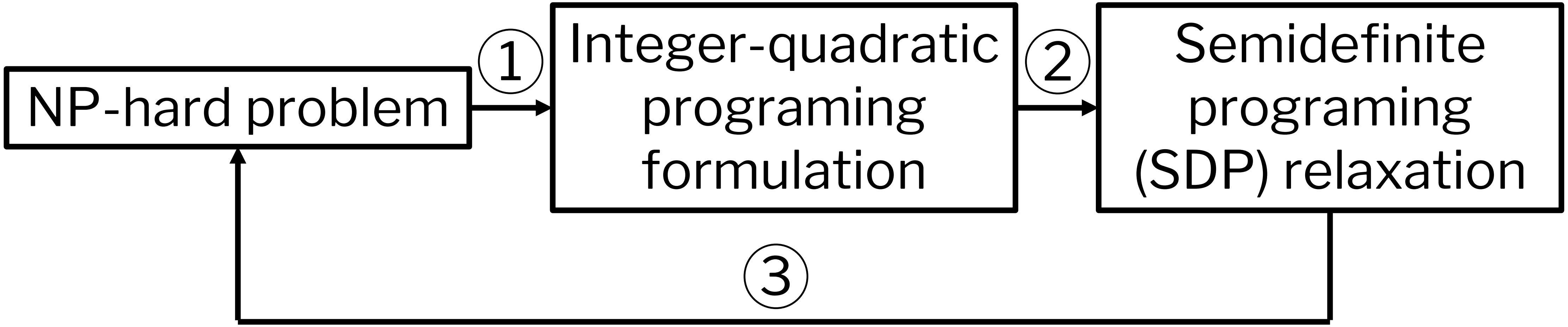
These problems have applications in computational biology, circuit design, and statistical physics.



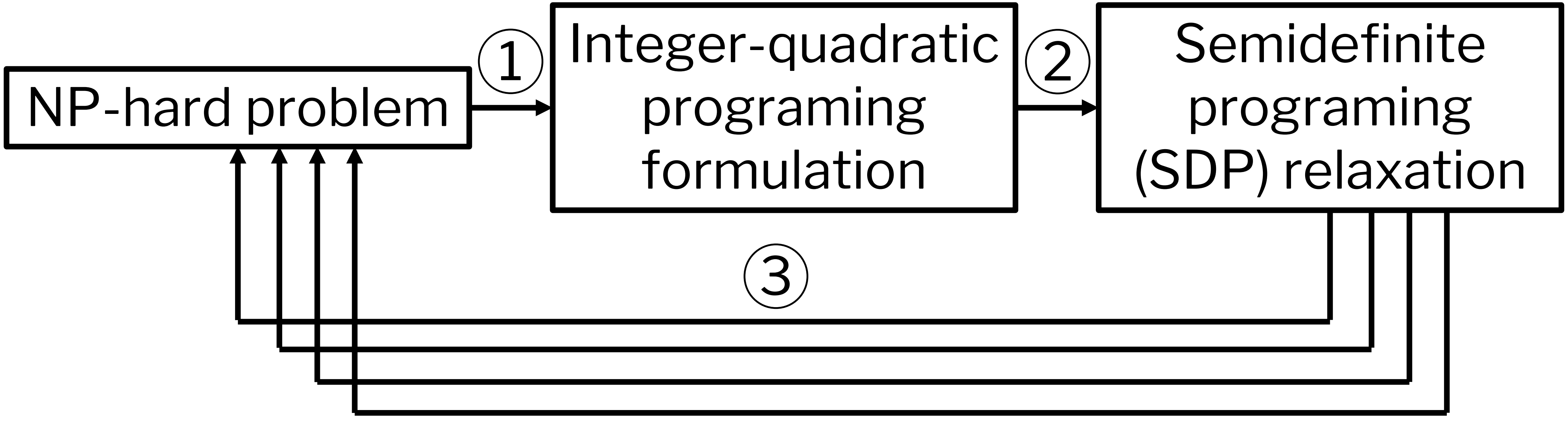
We focus on IQPs of the form:

$$\text{maximize } \boldsymbol{x}^T A \boldsymbol{x} \text{ subject to } \boldsymbol{x} \in \{-1,1\}^n$$



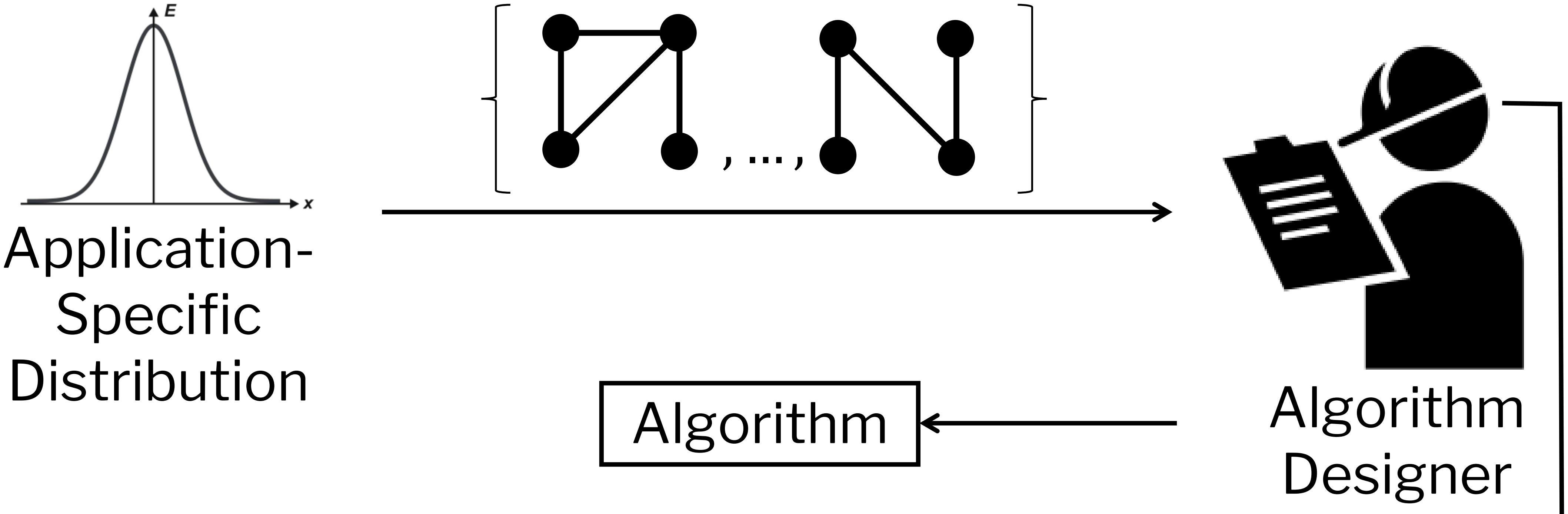


Transform SDP output to a feasible solution

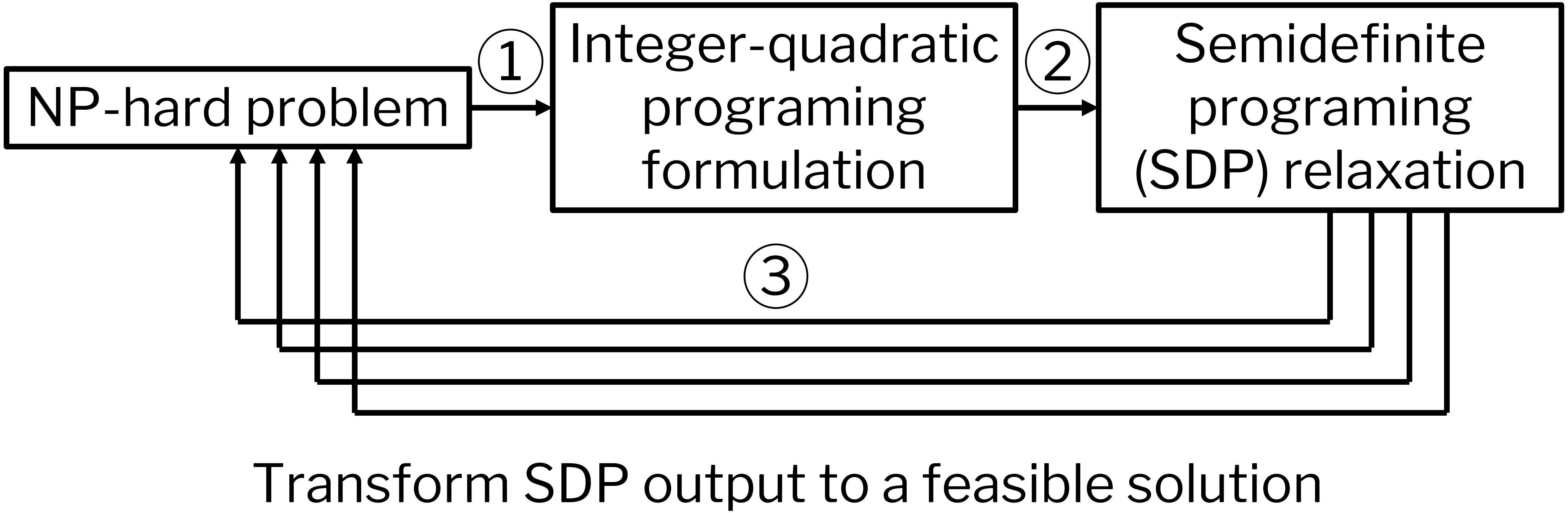


Transform SDP output to a feasible solution

Out of infinitely many ways to perform ③ how do we choose the best?



How can I use the set of samples to find an **SDP rounding algorithm** that's best for my application domain?

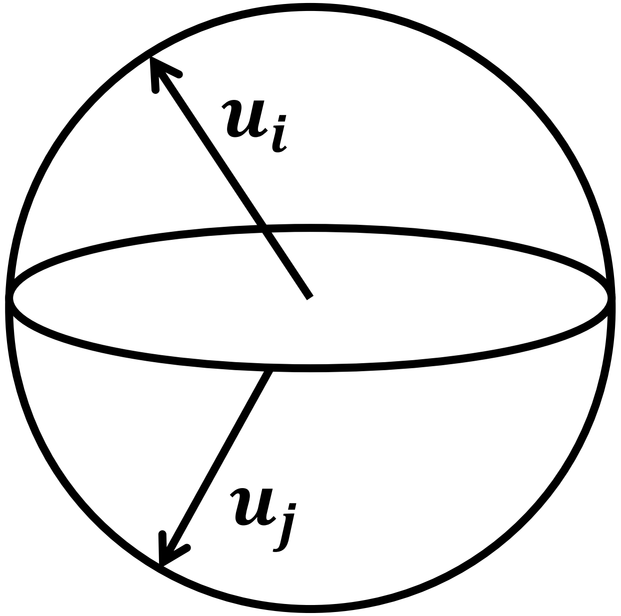

NP-hard problem
①
Integer-quadratic programing formulation
②
Semidefinite programing (SDP) relaxation
③
Transform SDP output to a feasible solution

② SDP relaxation

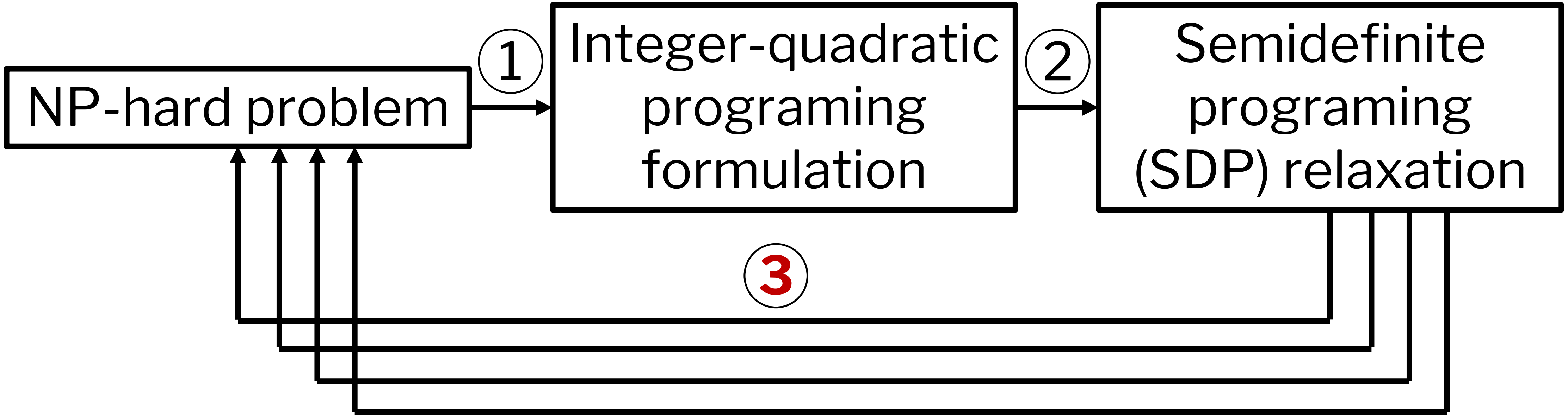
IQP formulation
maximize $\boldsymbol{x}^T A\boldsymbol{x} = \sum_{i,j} a_{i,j} x_i x_j$ subject to $\boldsymbol{x} \in \{-1,1\}^n$

Associate each binary variable x_i with a vector $\boldsymbol{u}_i$.

SDP relaxation
maximize $\sum_{i,j} a_{i,j} \langle \boldsymbol{u}_i, \boldsymbol{u}_j \rangle$ subject to $\|\boldsymbol{u}_i\| = 1$



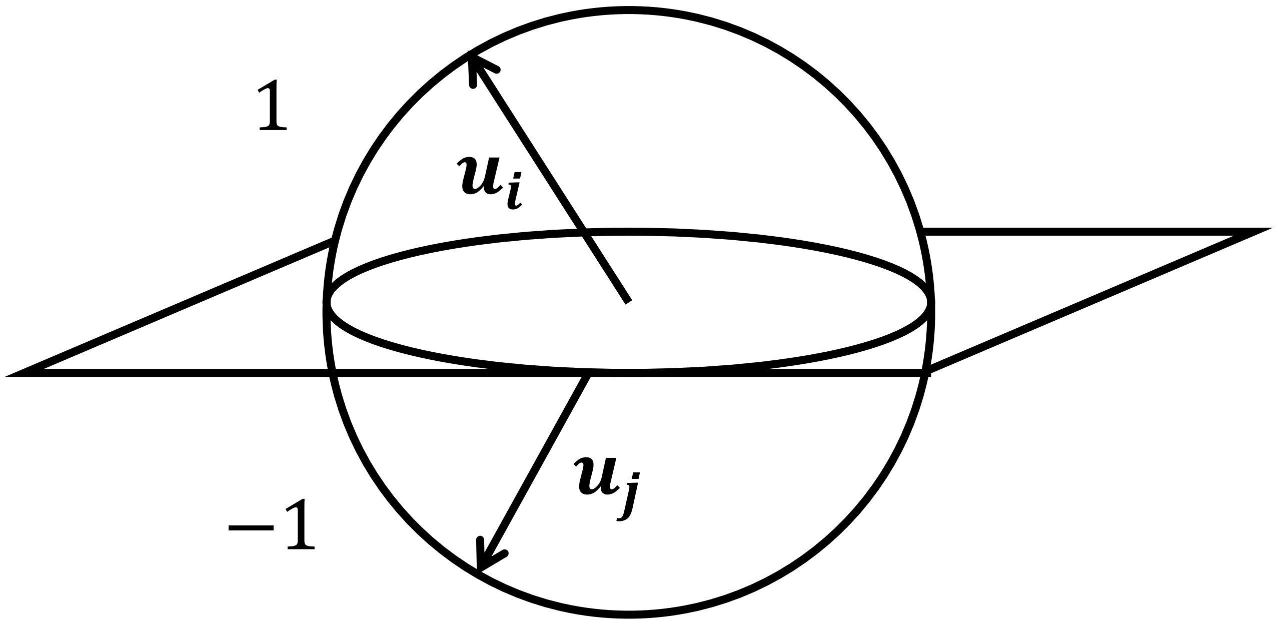
The SDP yields an optimal *embedding*…
but **how does it indicate a feasible solution x to the IQP**?



Transform SDP output to a feasible solution

Rounding procedure [Goemans and Williamson ‘95]

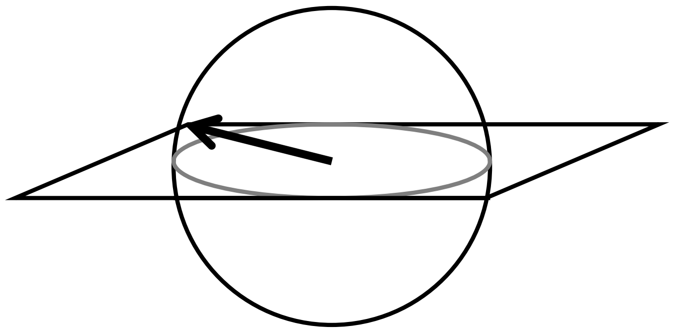
1. Choose a random hyperplane
2. (**Deterministic thresholding.**) Set x_i to -1 or 1 based on which side of the hyperplane the vector $\boldsymbol{u}_i$ falls on



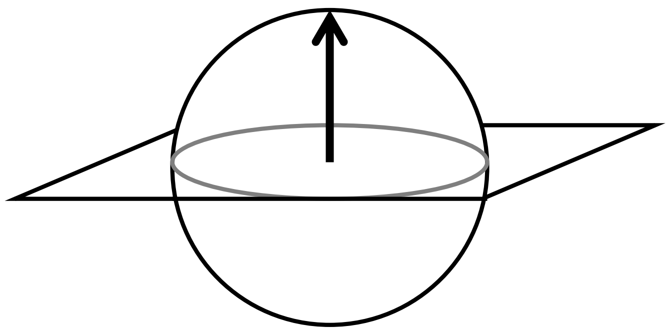
Rounding procedure [Goemans and Williamson ‘95]

1. Choose a random hyperplane
2. (**Deterministic thresholding.**) Set x_i to -1 or 1 based on which side of the hyperplane the vector $\boldsymbol{u}_i$ falls on

Zwick [‘99] and Feige and Langberg [’06] showed that **randomized thresholding** works even better in some cases.

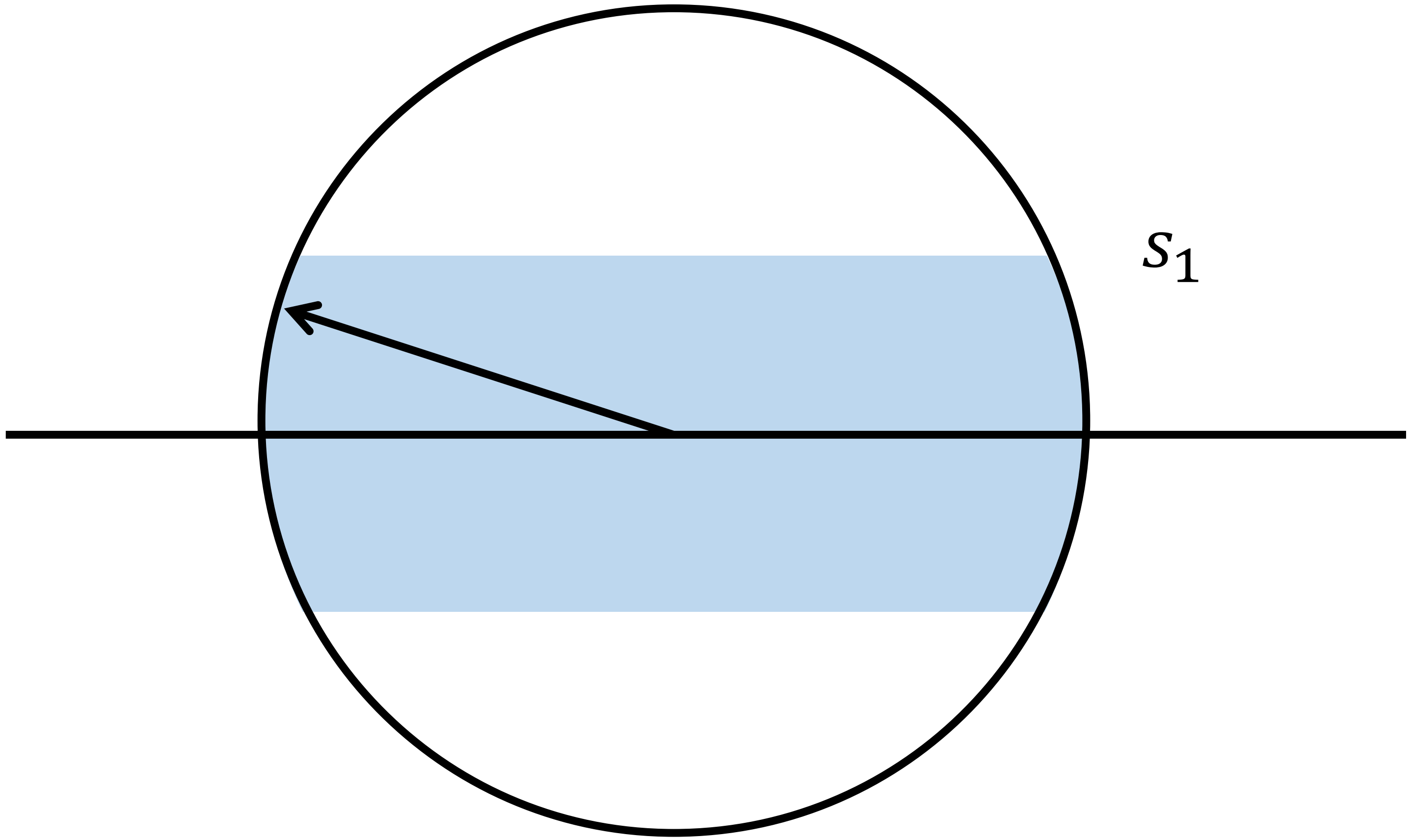


1 or −1?

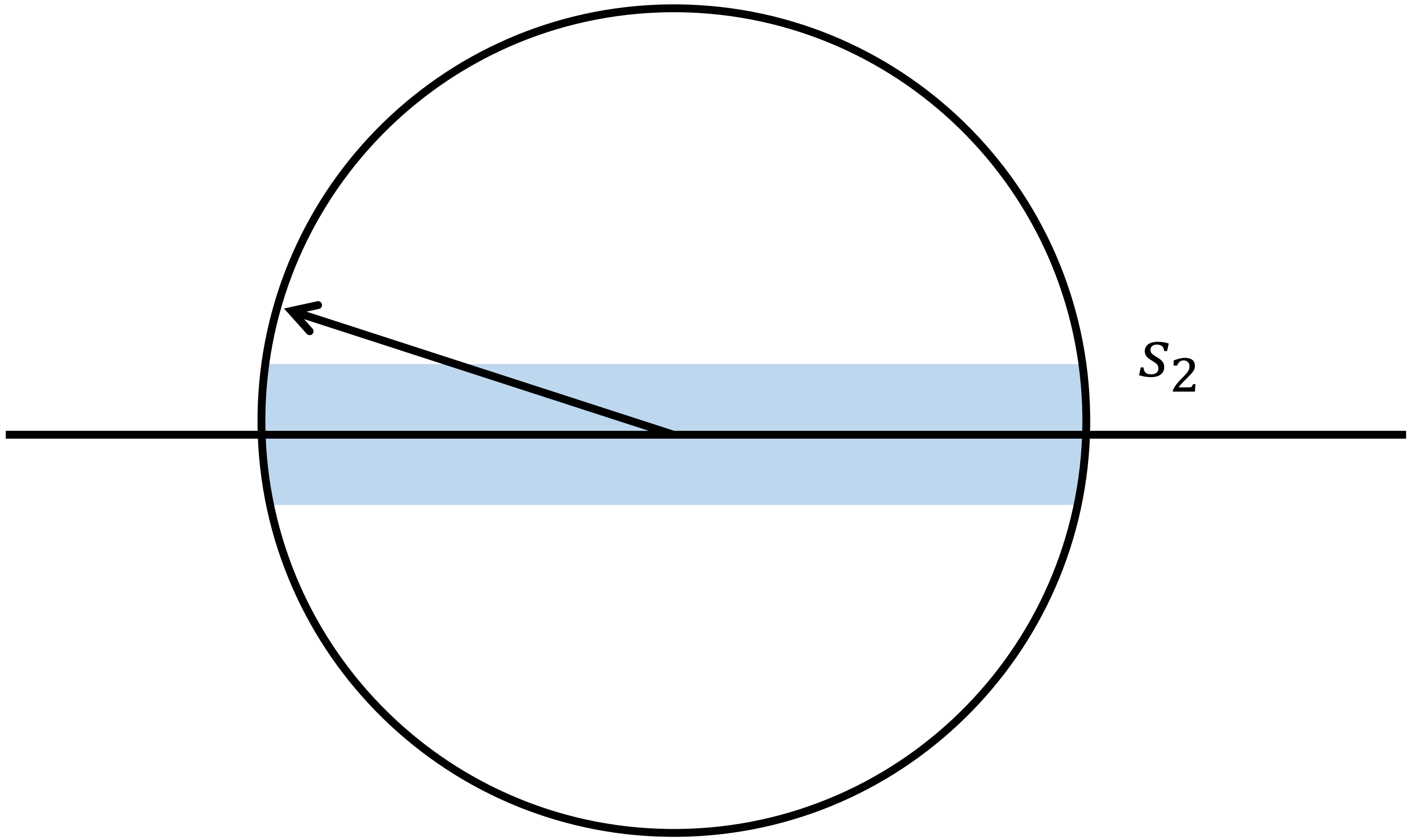


Definitely 1.

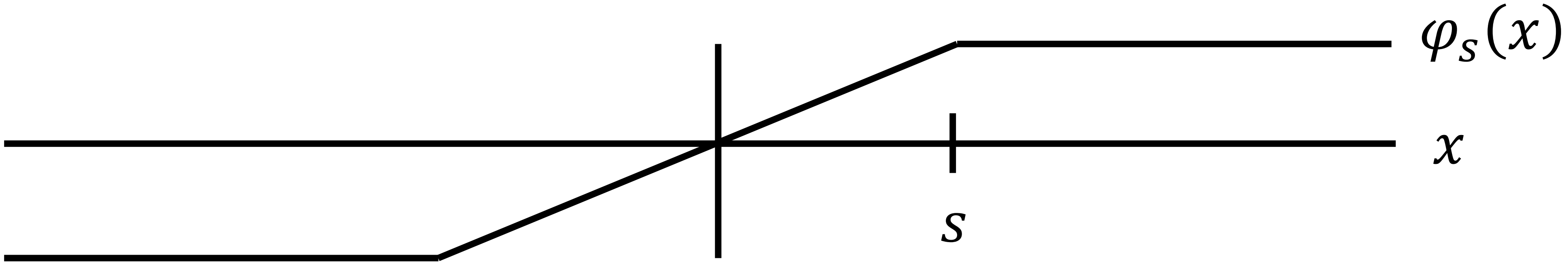
Feige and Langberg ['06] proposed a parameterized algorithm family. The parameter controls the level of randomness in the final vertex assignment.



Feige and Langberg ['06] proposed a parameterized algorithm family. The parameter controls the level of randomness in the final vertex assignment.

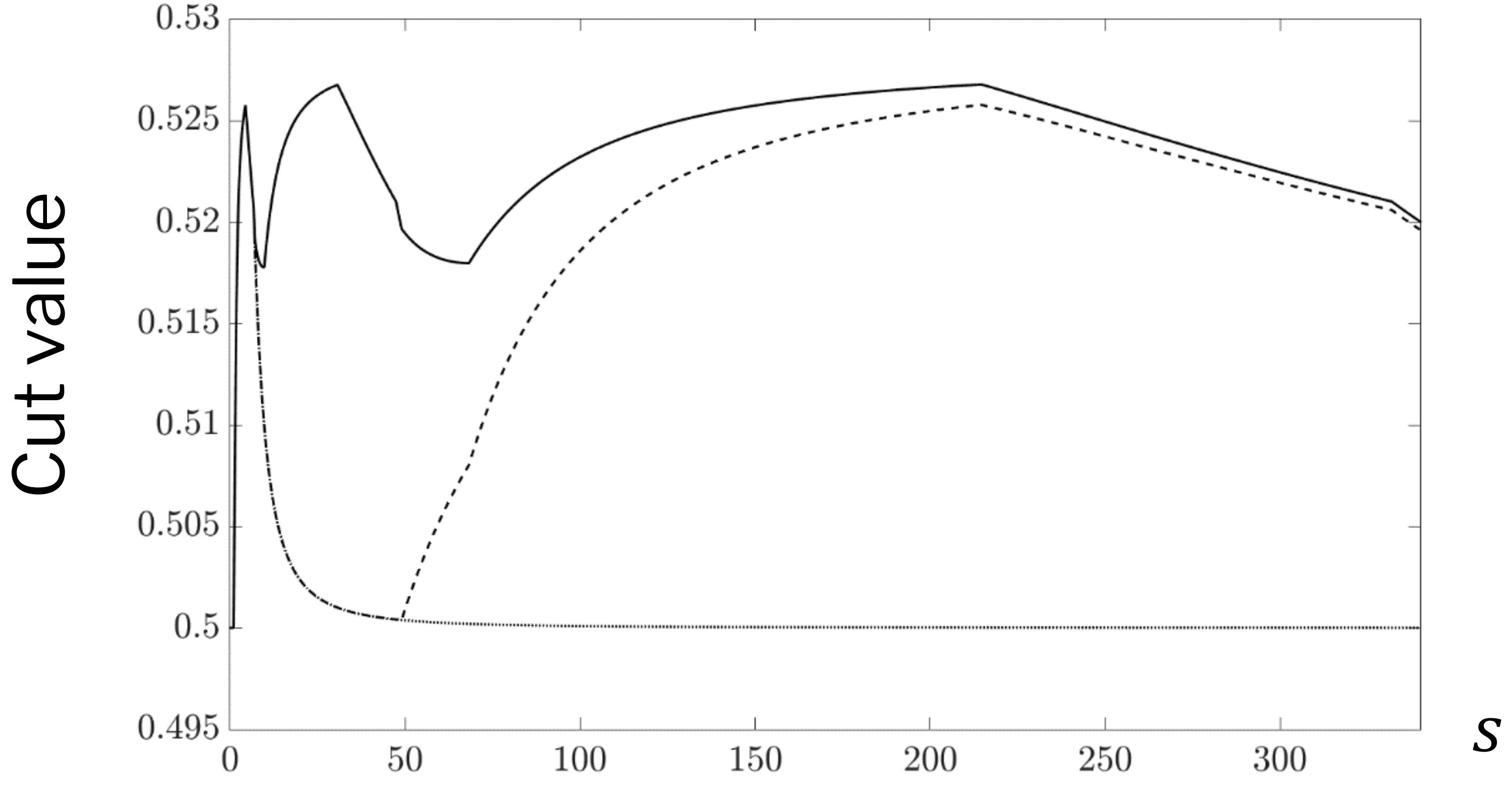


This parameterized family is defined by ***s*-linear rounding** functions: $\varphi_s(x) = -\mathbf{1}_{x<-s} + \frac{x}{s} \cdot \mathbf{1}_{x\in[-s,s]} + \mathbf{1}_{x>s}$



Rounding prodedure [Feige and Langberg ‘06]
1. Draw a random hyperplane $\boldsymbol{Z}$ 2. Set x_i to 1 with probability $\frac{1}{2} + \frac{1}{2}\varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle)$ and -1 with probability $\frac{1}{2} - \frac{1}{2}\varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle)$

The expected IQP objective value is **piecewise quadratic** in $\frac{1}{s}$ with a **small** number of pieces.



Rounding prodedure [Feige and Langberg '06]
1. Draw a random hyperplane $\boldsymbol{Z}$
2. Set x_i to 1 with probability $\frac{1}{2} + \frac{1}{2}\varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z} \rangle)$ and -1 with probability $\frac{1}{2} - \frac{1}{2}\varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z} \rangle)$

Notice that $\mathbb{E}[x_i] = \varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z} \rangle)$.

Given a hyperplane $\boldsymbol{Z}$, the expected value of the solution produced by this s-linear rounding scheme is

$$\mathbb{E}\left[\sum_{i,j} a_{i,j} x_i x_j\right] = \sum_{i,j} a_{i,j}\, \varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle)\varphi_s\big(\langle \boldsymbol{u}_j, \boldsymbol{Z}\rangle\big)$$

$\varphi_s(\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle)$ is $-1, +1$ or $\frac{\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle}{s}$
dependening on whether
$\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle < -s$, $\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle \in [-s, s]$, or $\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle > s$

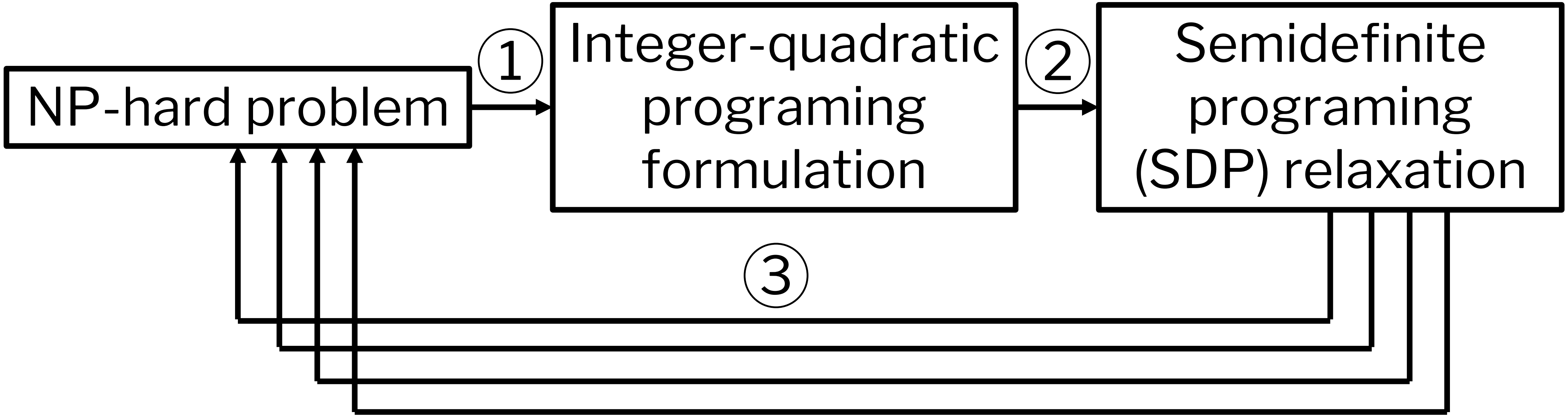
The expected IQP value is piecewise quadratic in $\frac{1}{s}$ with boundaries at the points $|\langle \boldsymbol{u}_i, \boldsymbol{Z}\rangle|$.

Algorithm (high level)

1. Solve for all intervals over the sample where the objective function is piecewise quadratic.
2. Find the best parameter over each interval.
3. Output the best parameter overall.

(We prove that this parameter is approximately optimal over the unknown distribution)

We give an **efficient** algorithm for determining a nearly **optimal randomized rounding function** from Feige and Langberg's infinite class. It requires $\tilde{O}(1/\epsilon^2)$ samples.



Transform SDP output to a feasible solution

1. Clustering algorithm configuration
    a. What should the class of clustering algorithms be?
    b. How do we find the empirically optimal algorithm A^*?
    c. Will the performance of A^* generalize to the distribution?

2. Integer quadratic programming algorithm configuration

3. Ongoing work

In recently submitted work, we extract structure shared among many configuration problems.

We show how to use this structure to design both **online** and **private** configuration algorithms.

Nina Balcan, Travis Dick, and Ellen Vitercik. Private and Online Optimization of Piecewise Lipschitz Functions. *arXiv*, 2017.

Thanks!

Questions?