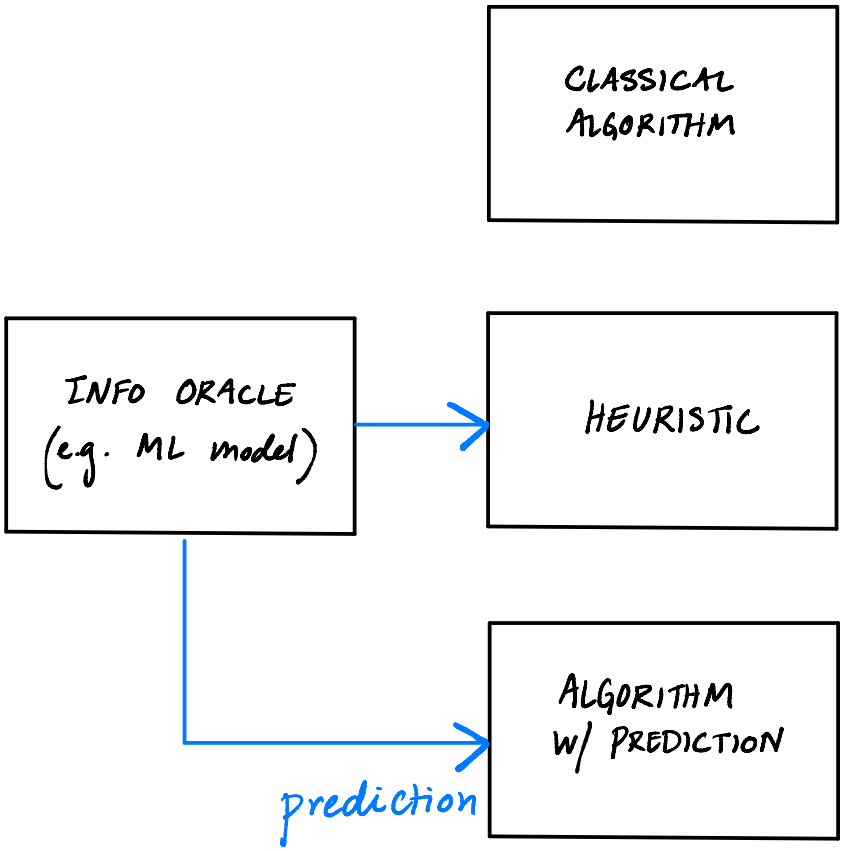
Competitive strategies to use "warm start" algorithms with predictions

Vaidehi Srinivas
Northwestern University

joint work with
Avrim Blum
Toyota Technological Institute, Chicago

ALGORITHMS WITH PREDICTIONS



Pessimistic:

Worst-case guarantee on every input

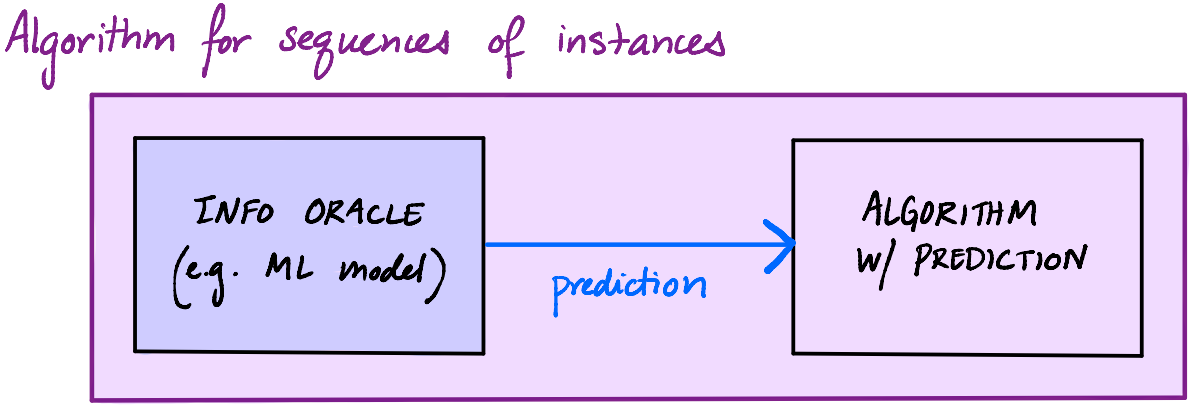
Optimistic:

Performs well when information is good

Best of both worlds:

Good performance when info is good
and worst-case guarantee

LEARNING PREDICTIONS



- Growing line of work in designing algorithms that use predictions

OUR GOAL: learn and use good predictions for algorithms with predictions

↳ "End to end" guarantees for solving sequences of related instances

WARM STARTS

Assume an instance space $\mathbb{I}$, metric solution space $\mathbb{S}$

Definition.

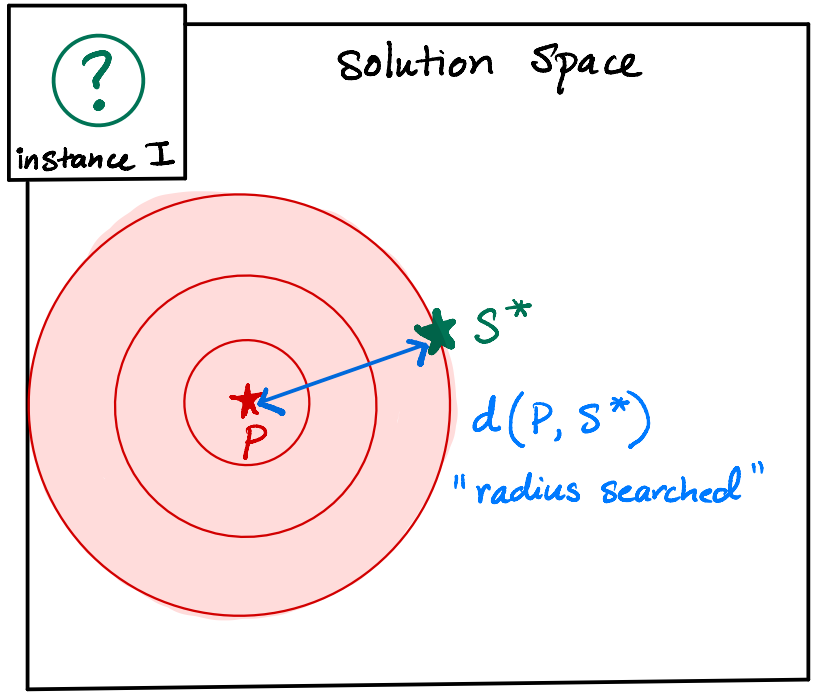
A warm start algorithm $\mathcal{A}(I, P)$ for $I \in \mathbb{I}$, $P \in \mathbb{S}$

finds S^* (true solution for instance I)

in time at most $d(P, S^*)$.

Examples.

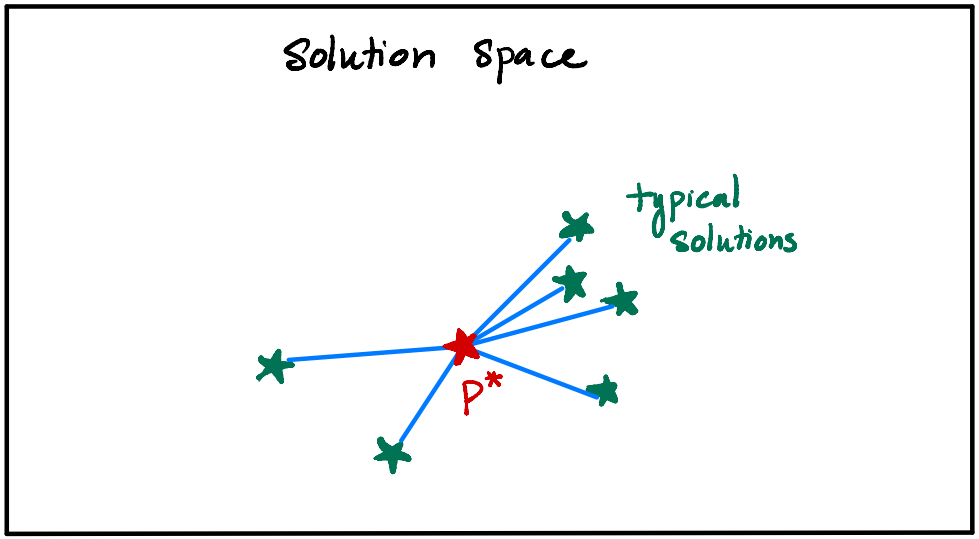
- Bipartite matching via learned duals
  [Dinitz Im Lavastida Moseley Vassilvitskii 21],
  [Chen Silwal Vakilian Zhang 22]
- Max Flow via Ford Fulkerson
  [Polak Zub 22],
  [Davies Moseley Vassilvitskii Wang 23]



HOW TO FIND PREDICTIONS: DISTRIBUTIONAL

Model: See instance-solution pairs drawn from a distribution

$$(I, S) \sim \mathcal{D} \quad \text{over} \quad \mathbb{I} \times \mathbb{S}$$



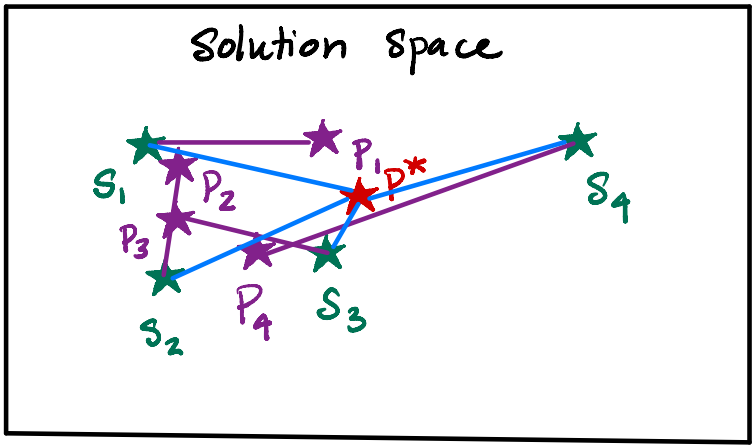
Strategy: Learn fixed prediction P^* to minimize

$$\mathop{\mathbb{E}}_{(I,S)\sim\mathcal{D}} \left[d\,(P^*, S) \right].$$ (clustering cost)

On future instances, run $\mathcal{A}(I, P^*)$.

HOW TO FIND PREDICTIONS: ONLINE

Model: A sequence of instances $I_1, \ldots, I_t$ arrive online



Strategy: Select predictions to compete with best fixed prediction in hindsight p^* via online convex optimization

[Khodak Balcan Talwalkar Vassilvitskii '22]

⚠ Distributional and online guarantees require solutions to be very close together to provide strong guarantees

? Can we get strong guarantees in more general settings?

↳ still require some structure!

WARM UP: COMPETING WITH k PREDICTIONS (DISTRIBUTIONAL)

? What if there are multiple types of instances?

Strategy: Learn predictions $P_1, \ldots, P_k$ to minimize expected distance to some prediction

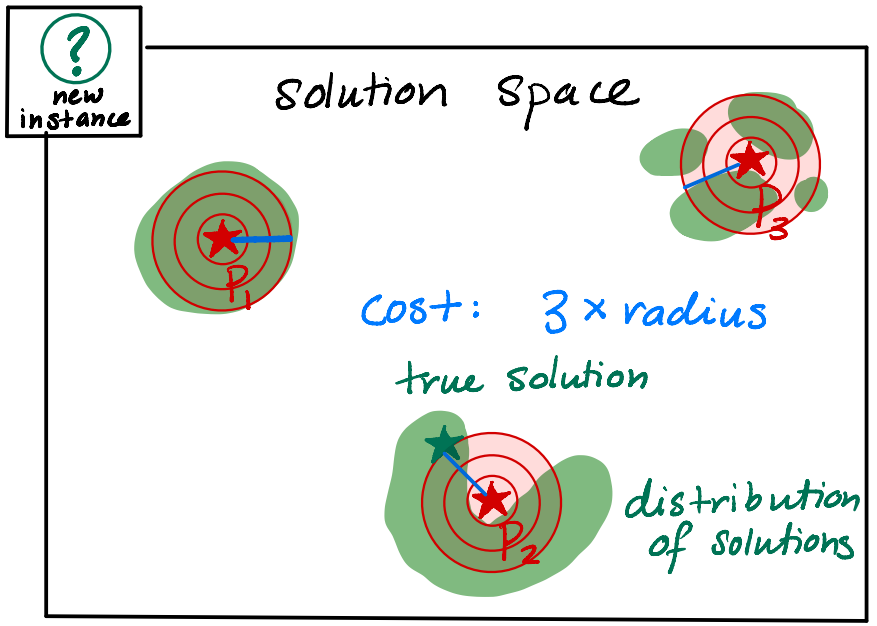
$$\text{minimize} \quad \mathop{\mathbb{E}}_{(I,S)\sim\mathcal{D}}\left[\min_i d(P_i, S)\right] \qquad \text{(k-medians clustering cost)}$$

Algorithm.

For a fresh $(I, S) \sim \mathcal{D}$,

run $\mathcal{A}(I, P_1), \ldots, \mathcal{A}(I, P_k)$ "in parallel"

output computation that completes first.



Guarantee.

Expected work of algorithm is $O(k)$ times the k-median clustering cost.

OVERVIEW OF RESULTS

? Can we get around the $O(k)$ factor by using other structure?

- Show how to learn a mapping from instances to predictions, $h : \mathbb{I} \longrightarrow \mathbb{S}$
- Get $O(1)$ approximation guarantee when h comes from a learnable class $\mathcal{H}$

? What can we do in the online setting?

FOCUS OF THIS TALK

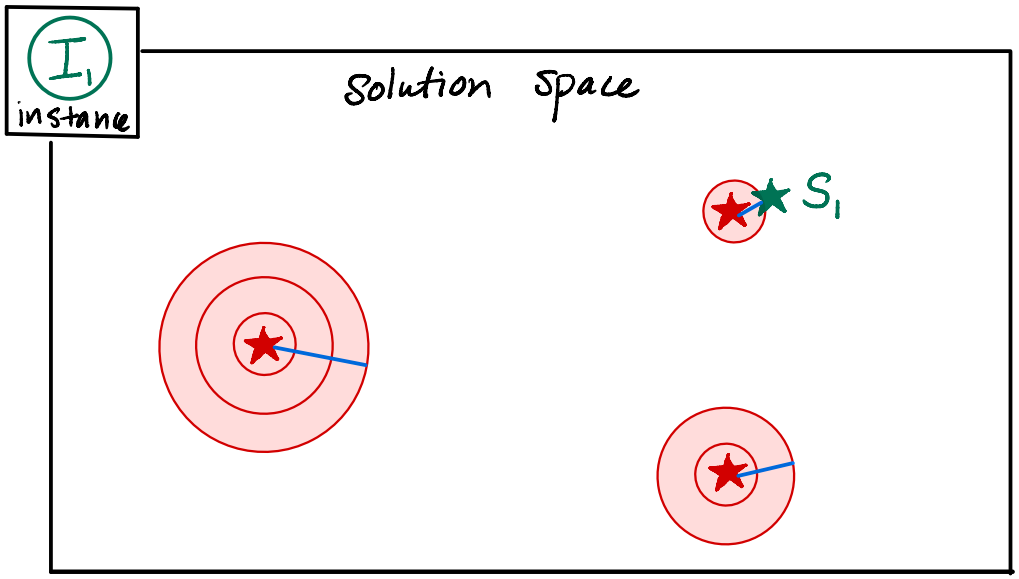
- Formulate the "online ball search" problem
- Provide an algorithm that works well when solutions are k-clusterable in hindsight.

ONLINE FORMULATION

Problem.

- Have access to warm-start algorithm $\mathcal{A}$ (search function)
- On each day $t \in [T]$, instance I_t arrives online
  - Algorithm chooses predictions and search schedule to find solution S_t of I_t

Goal: minimize total radius searched over all days



total radius searched:

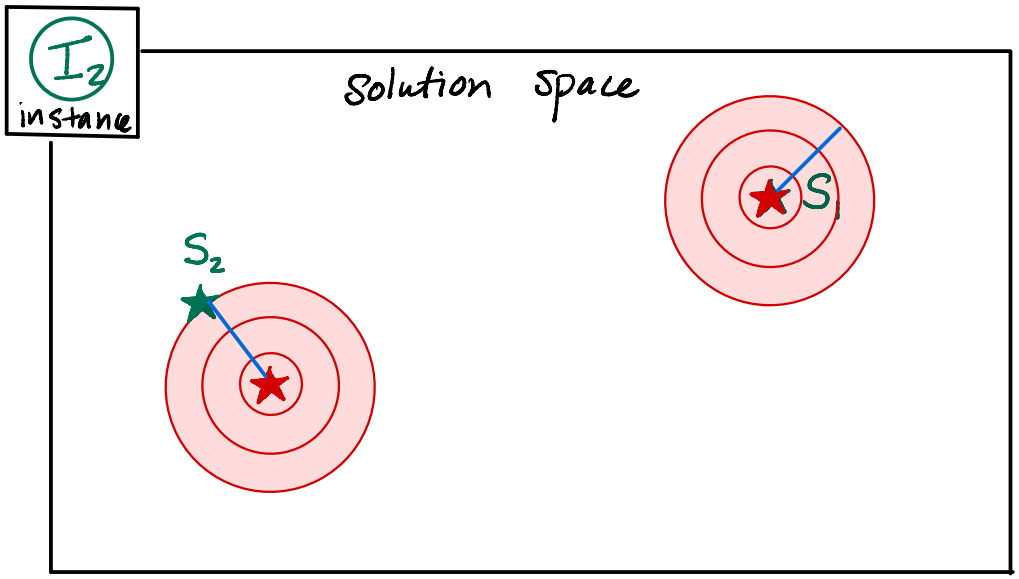
6

ONLINE FORMULATION

Problem.

- Have access to warm-start algorithm $\mathcal{A}$ (search function)
- On each day $t \in [T]$, instance I_t arrives online
  - Algorithm chooses predictions and search schedule to find solution S_t of I_t

Goal: minimize total radius searched over all days



total radius searched:

$6 + 6 = 12$

TYPICAL BASELINES

WANT: Analyze algorithms via competitive analysis

Offline optimum guesses solution perfectly every day and incurs 0 cost!

▷ Analysis does not assume enough structure in the sequence

Previous approach: [Khodak Balcan Talwalkar Vassilvitskii '22]

Competes with best fixed prediction in hindsight

↳ restricts power of the offline strategies that we compete against

Modeling contribution: What is a more powerful set of offline strategies that we can design competitive algorithms against?

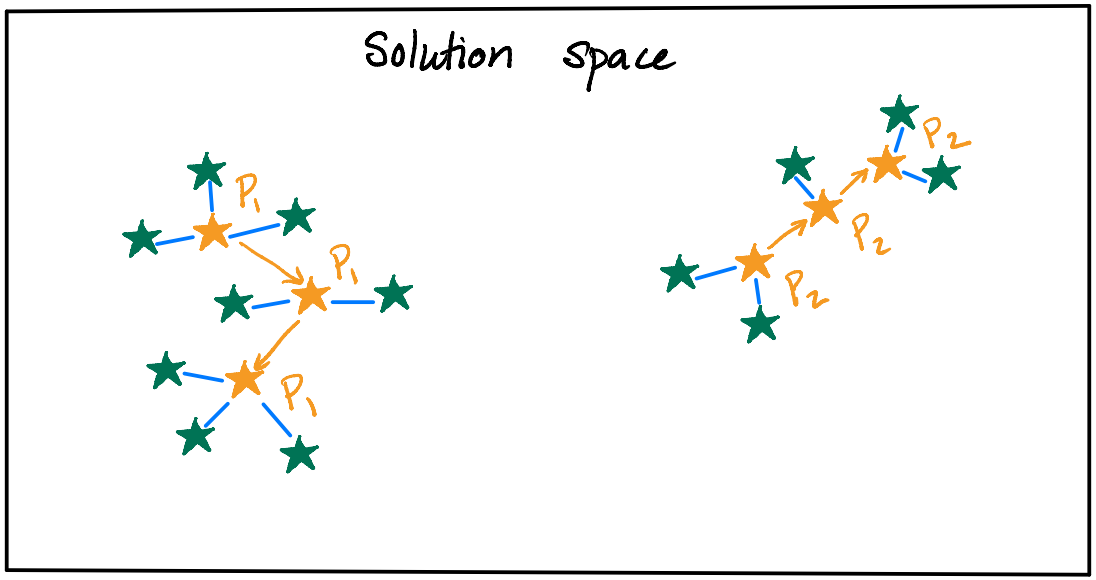
BASELINES

Definition [k offline trajectories]

Offline strategy that maintains a set of k predictions, and on each day pays

hit cost: distance from today's solution to closest prediction

movement cost: distance moved by all predictions



(Similar to offline k-server cost)

- Sequences that are "clusterable" have good baselines
- **WANT:** Design algorithm that is competitive with baselines

ONLINE RESULT

Theorem [Blum S. 25].

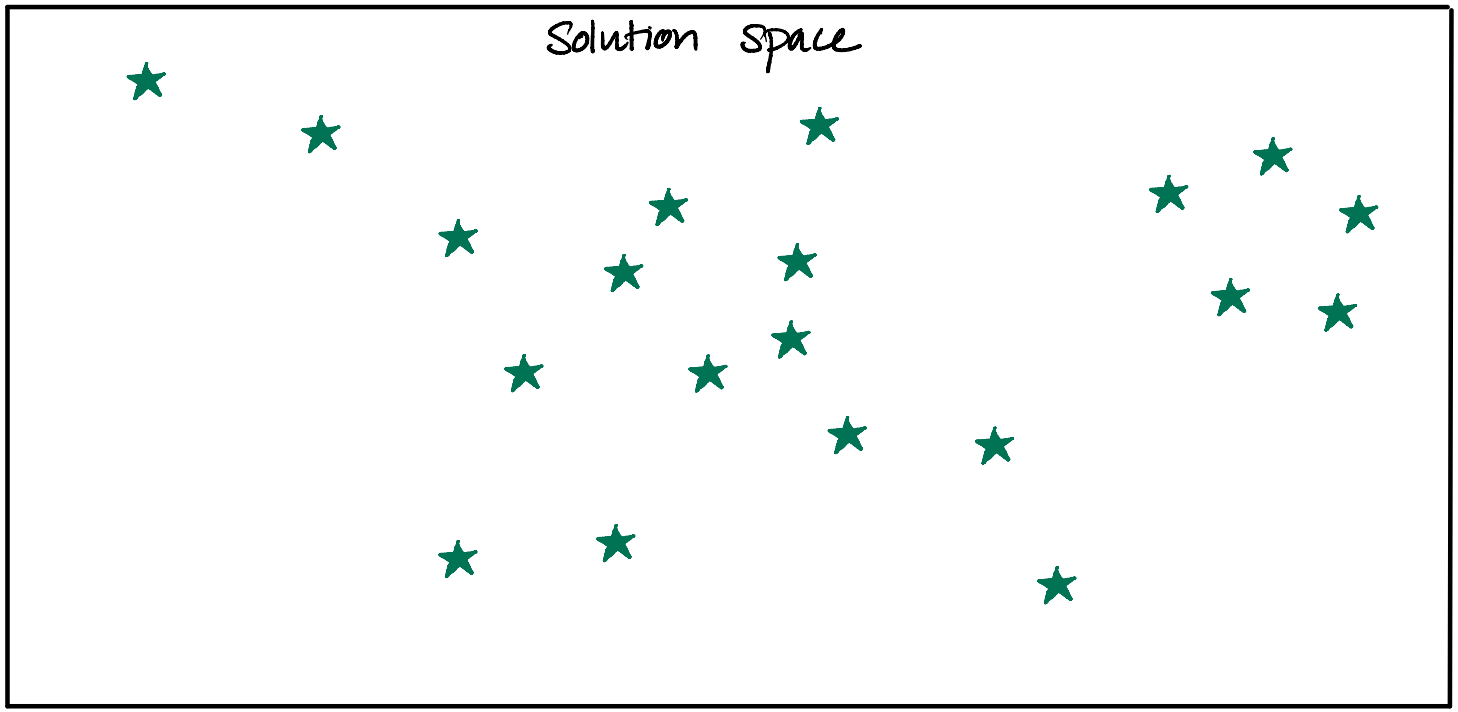
We give an algorithm that is $O(k^4 \ln^2 k)$ - competitive with any set of k offline trajectories in radius searched.

Furthermore,

- total runtime of algorithm is bounded by $O(1)$ times the radius searched
- algorithm is deterministic (robust to adaptive adversary)
- guarantee holds for all k simultaneously

- Our result + warm start algorithm $\Longrightarrow$ end-to-end guarantee for sequences of instances that display "clustering" structure

ONLINE CHALLENGES

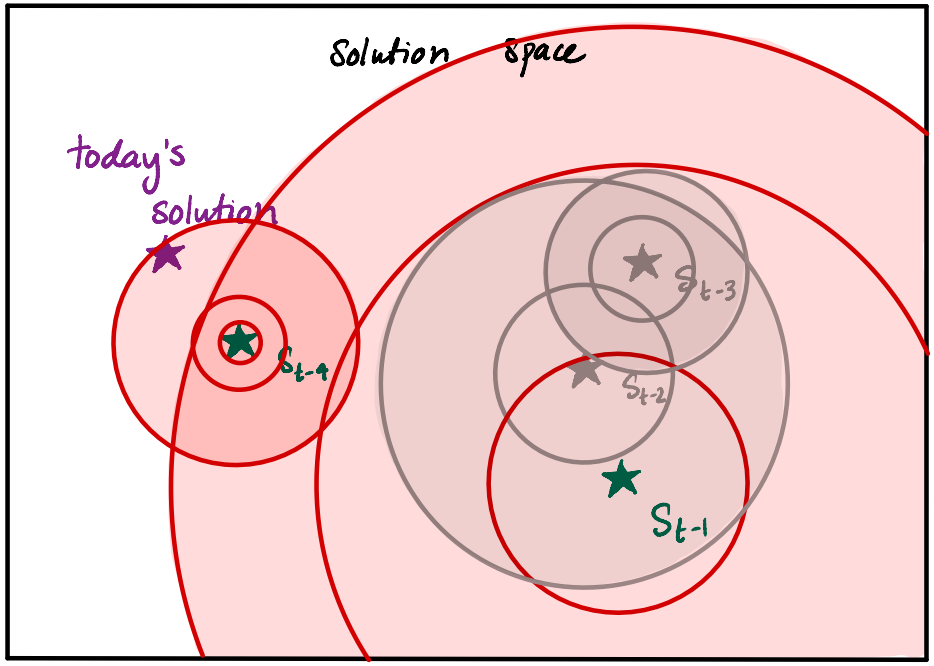


- Competing with 1 trajectory is simple!
  Use yesterday's solution as today's prediction

Issues with multiple trajectories:

1. Don't know which trajectory today belongs to
2. Don't know how to group previous solutions into trajectories

OVERVIEW OF ONLINE ALGORITHM



Algorithm. [Quadratic decay]

On day t:

- Search from i^{th} most recent solution at rate $\frac{1}{i^2 \ln^2 i}$
- if Solution i "subsumes" Solution j: kill solution j and increase rates of slower solutions

- **WANT:** Most recent Solution from Same trajectory will find ★
- If the algorithm runs long enough, other trajectories "collapse" and solution from same trajectory elevated to high rate
- Spiritually related to double coverage algorithm for k-server

CONCLUSION

We study how to learn and use predictions to solve sequences of related instances using a "warm start" algorithm.

Distributional setting:

- a simple strategy pays $O(k)$ times as much as the k-clustering cost of the distribution
- Can potentially avoid the $O(k)$-factor blow-up by learning a mapping from instances to solutions

Online setting:

- Can design an algorithm to solve sequences of instances that are k-clustered in hindsight.

THANKS! vaidehi@u.northwestern.edu