# Online Algorithms with Multiple Predictions

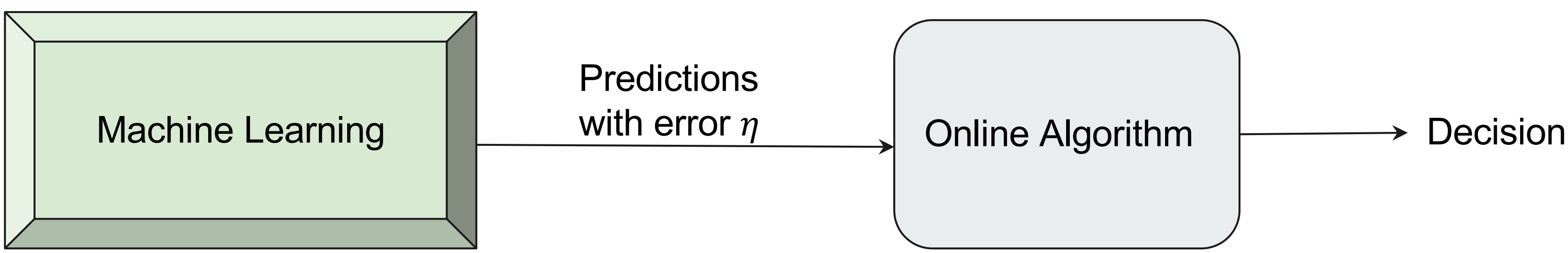
Keerti Anand[1], Rong Ge[1], Amit Kumar[2], Debmalya Panigrahi[1]

[1]Department of Computer Science, Duke University
[2]Department of Computer Science, Indian Institute of Technology Delhi

# Introduction

Online Algorithms with Predictions: An attempt to bypass the pessimistic worst-case bounds of traditional algorithm design.

Design an algorithm whose performance improves with the accuracy of the prediction; yet maintaining an inherent worst-case guarantee. Example Problems : Ski Rental, Scheduling, Caching, Matching and Secretary Problems.



## Algorithms with Multiple Predictions

Multiple Predictions Setting: Instead of one prediction, what if we get multiple sets of predictions?

Multiple ML algorithms, human experts can suggest their suitable solutions.

Some solutions can be arbitrarily bad but as long as one solution is good, our algorithm should be able to find it.

**Goal:** Algorithm whose performance is competitive against that of the *best* solution using the predictions

## Online Covering Framework

$$min \sum_{i=1}^{n} c_i \cdot x_i$$

Objective

$$\sum_{i=1}^{n} a_{ij} \cdot x_i \geq 1$$

At each step j:
New covering constraint arrives!

# Applications

Many online problems can be modelled into this framework:

1. Set Cover
2. Weighted Caching
3. Steiner Tree/Forest
4. Facility Location

We consider the fractional version of this framework

# Online Covering with Multiple Predictions

j-th covering constraint arrives! $$\sum_{i=1}^{n} a_{ij} \cdot x_i \geq 1$$

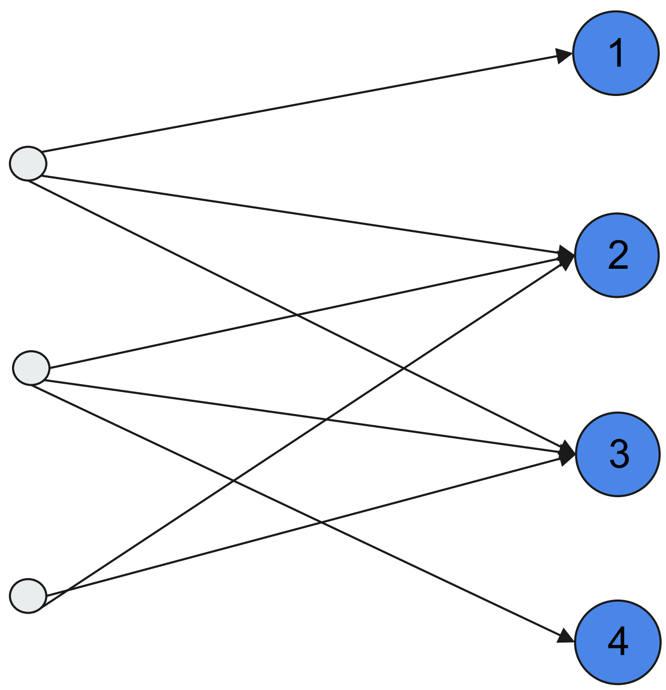
We get k sets of suggestions: $$x_i \rightarrow x_i(j, s)$$

Suggestions are feasible!

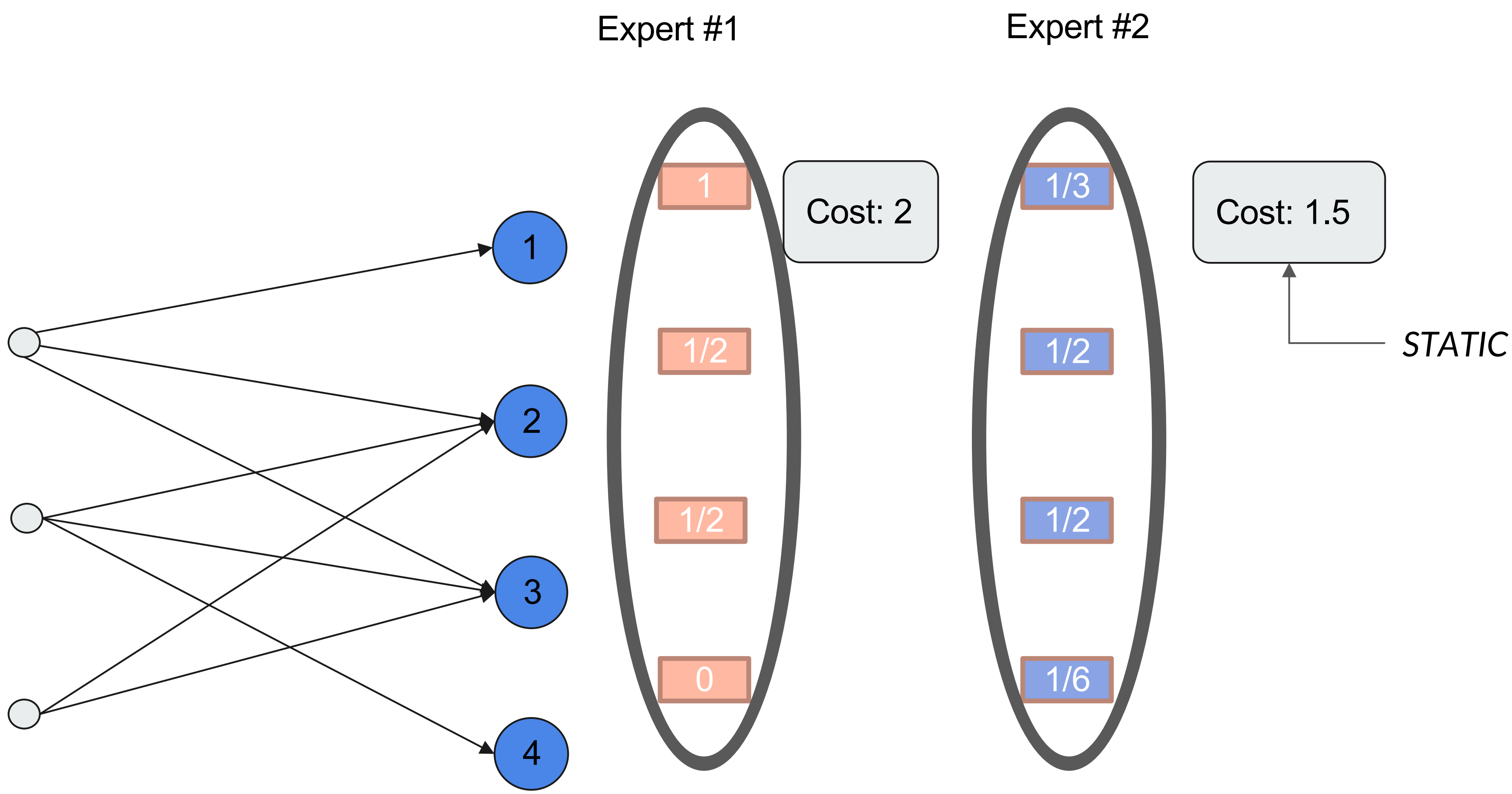
Benchmark:

- An optimal solution that chooses  the same suggestion in each time step — *STATIC*
- An optimal solution that chooses any of the suggestions in each time step — *DYNAMIC*

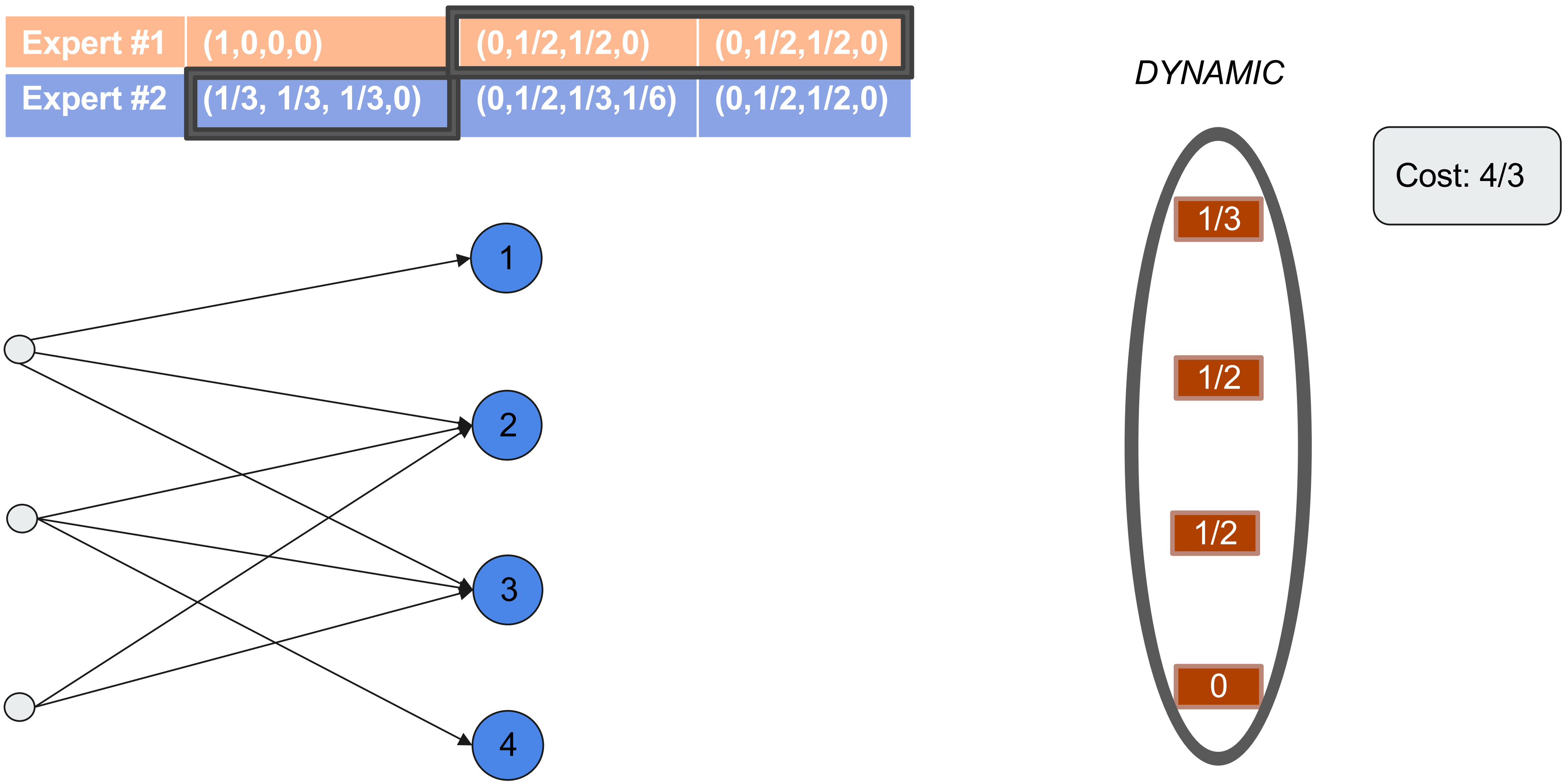
## Example : Set Cover




| | | | |
|---|---|---|---|
| Expert #1 | (1,0,0,0) | (0,1/2,1/2,0) | (0,1/2,1/2,0) |
| Expert #2 | (1/3, 1/3, 1/3,0) | (0,1/2,1/3,1/6) | (0,1/2,1/2,0) |



Expert #1
Expert #2
1
1/2
1/2
0
Cost: 2
1/3
1/2
1/2
1/6
Cost: 1.5
STATIC
1
2
3
4


Expert #1
(1,0,0,0)
(0,1/2,1/2,0)
(0,1/2,1/2,0)
Expert #2
(1/3, 1/3, 1/3,0)
(0,1/2,1/3,1/6)
(0,1/2,1/2,0)
1
2
3
4
DYNAMIC
1/3
1/2
1/2
0
Cost: 4/3

## Results

**Upper Bound**

There exists an algorithm whose cost is within an $O(\log k)$ factor of benchmark DYNAMIC

**Lower Bound**

No algorithm can do better than an $\Omega(\log k)$ factor even against the benchmark STATIC

**Algorithm 1:** Online Covering Algorithm

**Offline:** All variables $x_i$ are initialized to 0.

**Online:** On arrival of the $j$-th constraint:

    **While** $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$,

        **for** $i \in [n]$

            **if** $x_i < \frac{1}{2}$, increase $x_i$ by $\frac{dx_i}{dt} = \frac{a_{ij}}{c_i} \left( x_i + \frac{1}{k} \cdot \sum_{s=1}^{k} x_i(j, s) \right)$.

## Analysis (Sketch)

Use a potential function to budget the algorithm's cost.

- The value of the potential is at most O(log k) times the cost of DYNAMIC Benchmark.
- The rate of increase in the algorithms cost is O(1) times the decrease in potential.

# Thank You !