MedleySolver: Online SMT Algorithm Selection

Nikhil Pimpalkhare[1], Federico Mora[1],
Elizabeth Polgreen[1,2], and Sanjit A. Seshia[1]

[1] University of California, Berkeley

[2] University of Edinburgh

Sunday, June 27, 2021

24th International Conference on Theory and Applications of Satisfiability Testing

Berkeley UNIVERSITY OF CALIFORNIA

THE UNIVERSITY of EDINBURGH

Where Are SMT Solvers Used?

Example Applications

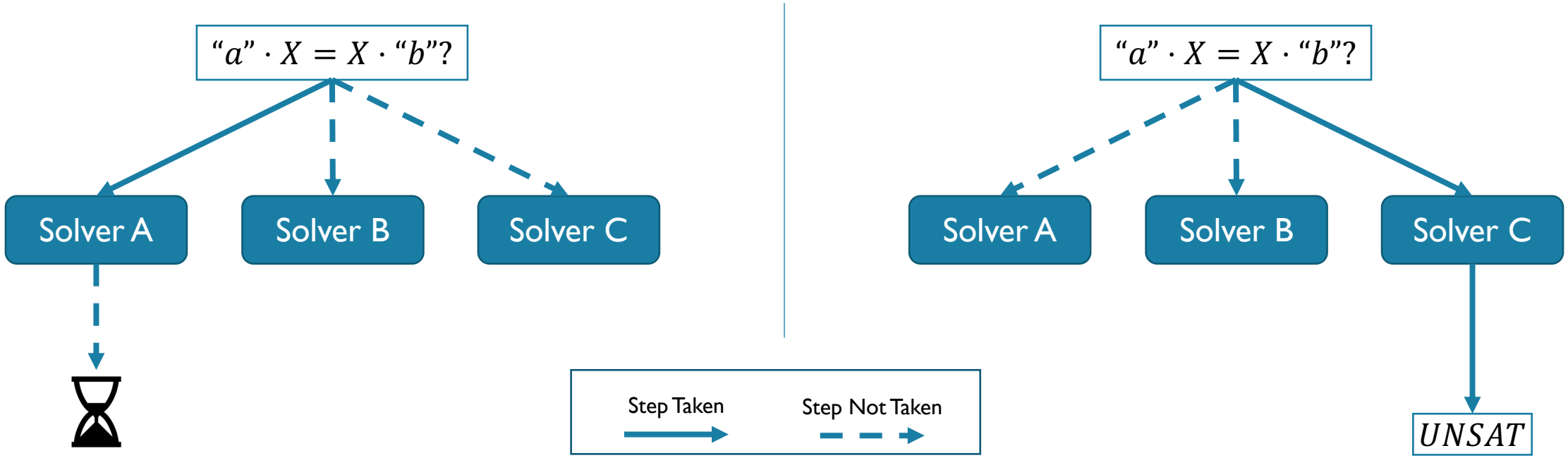
- Verification engines,
  - check many verification conditions.
- Symbolic execution engines,
  - check many path conditions.
- Program synthesis engines,
  - check many candidate programs.

Example Tools

The Rosette Language

What is SMT Algorithm Selection?

- For a given query, can we predict the solver that will perform best?



[1] J. R.Rice: The algorithm selection problem. Advances in Computers (1976).

Desired Features for Algorithm Selection

(End-User Perspective)

Feature \ Approach	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	☑	☑	☑
Minimal Upfront Costs			
No Data Requirements			
No Solver Requirements			
No Need to Repeat Upfront Costs			
Fine Grained Decisions			

Feature Support

- ☑ Strong
- ☐ Medium
- ☒ Weak

Existing Approaches

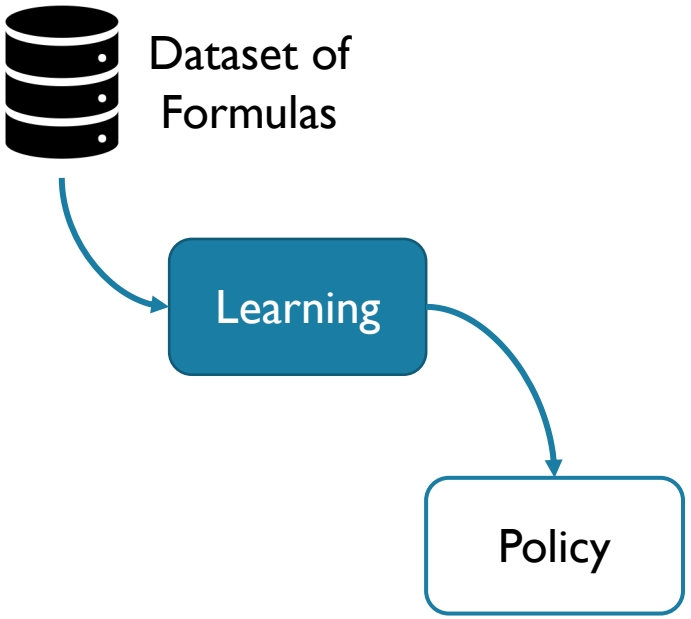
Expert Encoded

E.g., Z3's quantifier-free bit-vector solver tactic

```
(preamble_st,
 // If the user sets HI_DIV0=false, then the formula may contain uninterpreted function
 // symbols. In this case, we should not use the `sat', but instead `smt'. Alternatively,
 // the UFs can be eliminated by eager ackermannization in the preamble.
 cond(mk_is_qfbv_eq_probe(),
      and_then(mk_bv1_blaster_tactic(m),
               using_params(smt, solver_p)),
      cond(mk_is_qfbv_probe(),
           and_then(mk_bit_blaster_tactic(m),
                    when(mk_lt(mk_memory_probe(), mk_const_probe(MEMLIMIT)),
                         and_then(using_params(and_then(mk_simplify_tactic(m),
                                                        mk_solve_eqs_tactic(m)),
                                               local_ctx_p),
                                  if_no_proofs(cond(mk_produce_unsat_cores_probe(),
                                                    mk_aig_tactic(),
                                                    using_params(mk_aig_tactic(),
                                                                 big_aig_p))))),
                    sat),
           smt))));
```

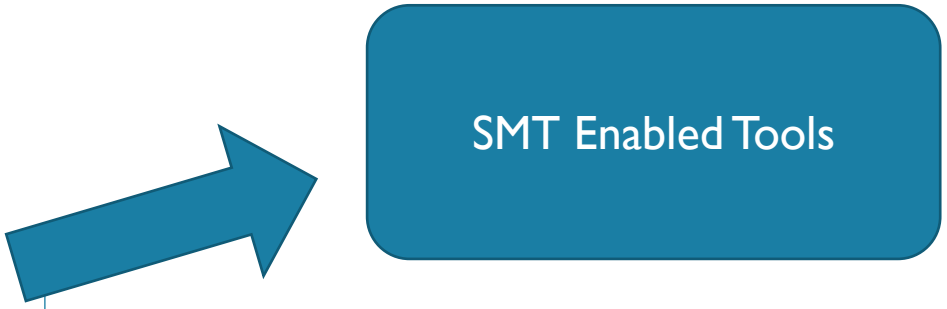
Machine Learned

E.g., SatZilla [2], MachSMT [3], FastSMT [4], …



[2] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. (2008)
[3] Scott, J., Niemetz, A., Preiner, M., Nejati, S., Ganesh, V.: MachSMT: A machine learning-based algorithm selector for SMT solvers. TACAS (2021)
[4] Balunovic, M., Bielik, P., Vechev, M.T.: Learning to solve SMT formulas. NeurIPS (2018)

Existing Approaches



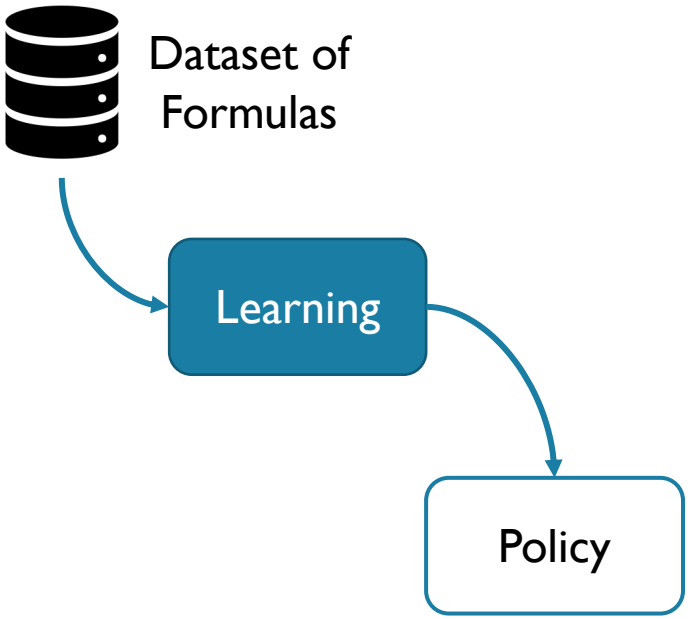
Expert Encoded

E.g., Z3's quantifier-free bit-vector solver tactic

```
(preamble_st,
 // If the user sets HI_DIV0=false, then the formula may contain uninterpreted function
 // symbols. In this case, we should not use the `sat', but instead `smt'. Alternatively,
 // the UFs can be eliminated by eager ackermannization in the preamble.
 cond(mk_is_qfbv_eq_probe(),
      and_then(mk_bv1_blaster_tactic(m),
               using_params(smt, solver_p)),
      cond(mk_is_qfbv_probe(),
           and_then(mk_bit_blaster_tactic(m),
                    when(mk_lt(mk_memory_probe(), mk_const_probe(MEMLIMIT)),
                         and_then(using_params(and_then(mk_simplify_tactic(m),
                                                        mk_solve_eqs_tactic(m)),
                                               local_ctx_p),
                                  if_no_proofs(cond(mk_produce_unsat_cores_probe(),
                                                    mk_aig_tactic(),
                                                    using_params(mk_aig_tactic(),
                                                                 big_aig_p))))),
                    sat),
           smt))));
```

Machine Learned

E.g., SatZilla [2], MachSMT [3], FastSMT [4], …



[2] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. (2008)
[3] Scott, J., Niemetz, A., Preiner, M., Nejati, S., Ganesh, V.: MachSMT: A machine learning-based algorithm selector for SMT solvers. TACAS (2021)
[4] Balunovic, M., Bielik, P., Vechev, M.T.: Learning to solve SMT formulas. NeurIPS (2018)

Existing Approaches

SMT Enabled Tools

Expert Encoded

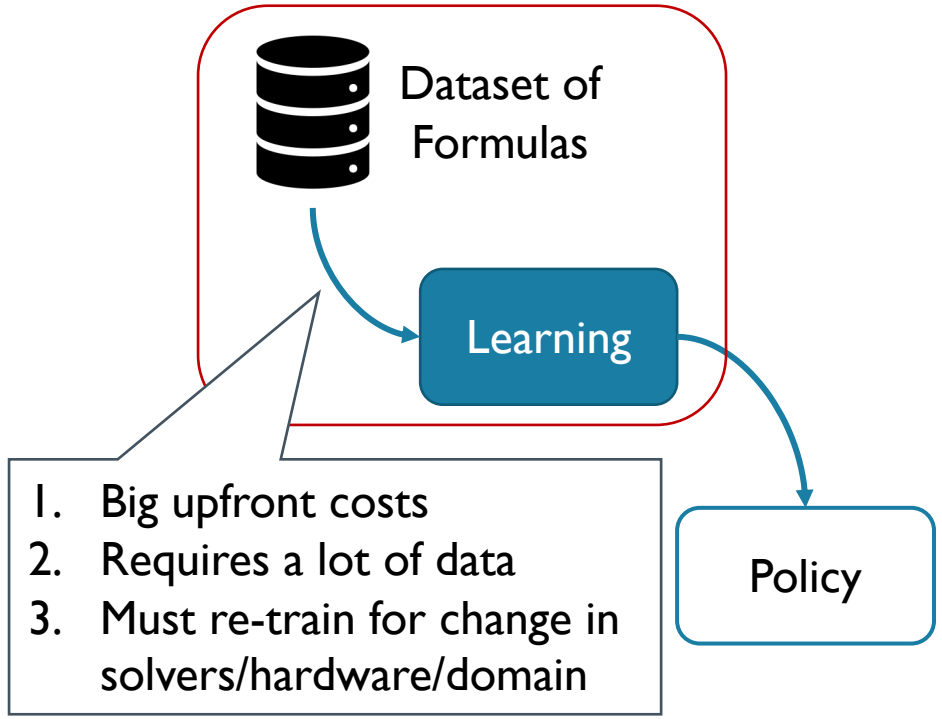
E.g., Z3's quantifier-free bit-vector solver tactic

```
(preamble_st,
 // If the user sets HI_DIV0=false, then the formula may contain uninterpreted function
 // symbols. In this case, we should not use the `sat', but instead `smt'. Alternatively,
 // the UFs can be eliminated by eager ackermannization in the preamble.
 cond(mk_is_qfbv_eq_probe(),
      and_then(mk_bv1_blaster_tactic(m),
               using_params(smt, solver_p)),
      cond(mk_is_qfbv_probe(),
           and_then(mk_bit_blaster_tactic(m)
                    when(mk_lt(mk_memory_p                (), mk_const_probe(MEMLIMIT)),
                         and_then(using_                      nd_then(mk_simplify_tactic(m),
                                                                      mk_solve_eqs_tactic(m)),
                                                                 ),
                                                                 duce_unsat_cores_probe(),
                                                                 _tactic(),
                                                                 arams(mk_aig_tactic(),
                                                                       big_aig_p))))),
                    sat),
           smt))));
```

1. Tailored to specific solvers
2. Not fine grained (takes time to engineer)

Machine Learned

E.g., SatZilla [2], MachSMT [3], FastSMT [4], …



[2] Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Satzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. (2008)
[3] Scott, J., Niemetz, A., Preiner, M., Nejati, S., Ganesh, V.: MachSMT: A machine learning-based algorithm selector for SMT solvers. TACAS (2021)
[4] Balunovic, M., Bielik, P., Vechev, M.T.: Learning to solve SMT formulas. NeurIPS (2018)

Desired Features for Algorithm Selection

(End-User Perspective)

Approach / Feature	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	
Minimal Upfront Costs	✓	✗	
No Data Requirements	✓	▢	
No Solver Requirements	✗	✓	
No Need to Repeat Upfront Costs	✓	✗	
Fine Grained Decisions	✗	✓	

Feature Support

- ✓ Strong
- ▢ Medium
- ✗ Weak

Desired Features for Algorithm Selection

(End-User Perspective)

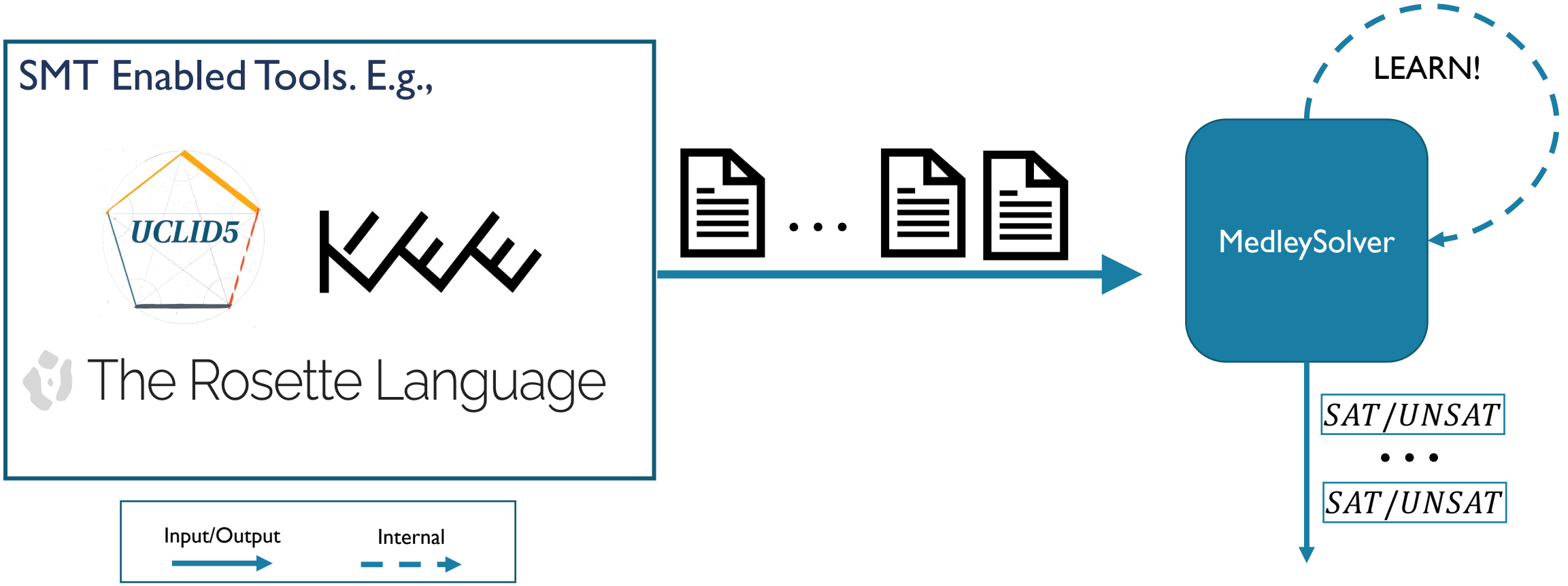
Approach / Feature	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	✓
No Data Requirements	✓	▯	✓
No Solver Requirements	✗	✓	✓
No Need to Repeat Upfront Costs	✓	✗	✓
Fine Grained Decisions	✗	✓	✓

Feature Support

- ✓ Strong
- ▯ Medium
- ✗ Weak

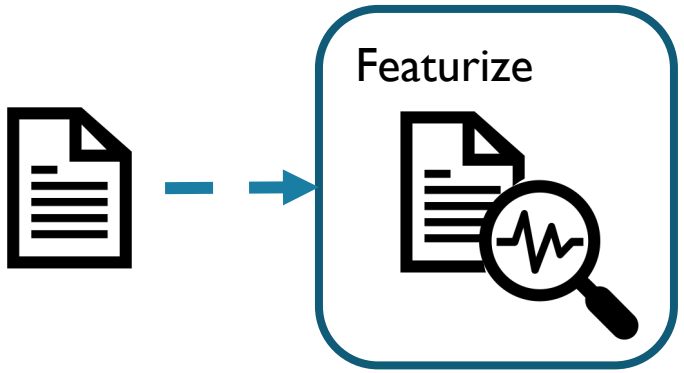
Our Proposal: Use *Online* Learning

- Learn as we go with what we have locally!

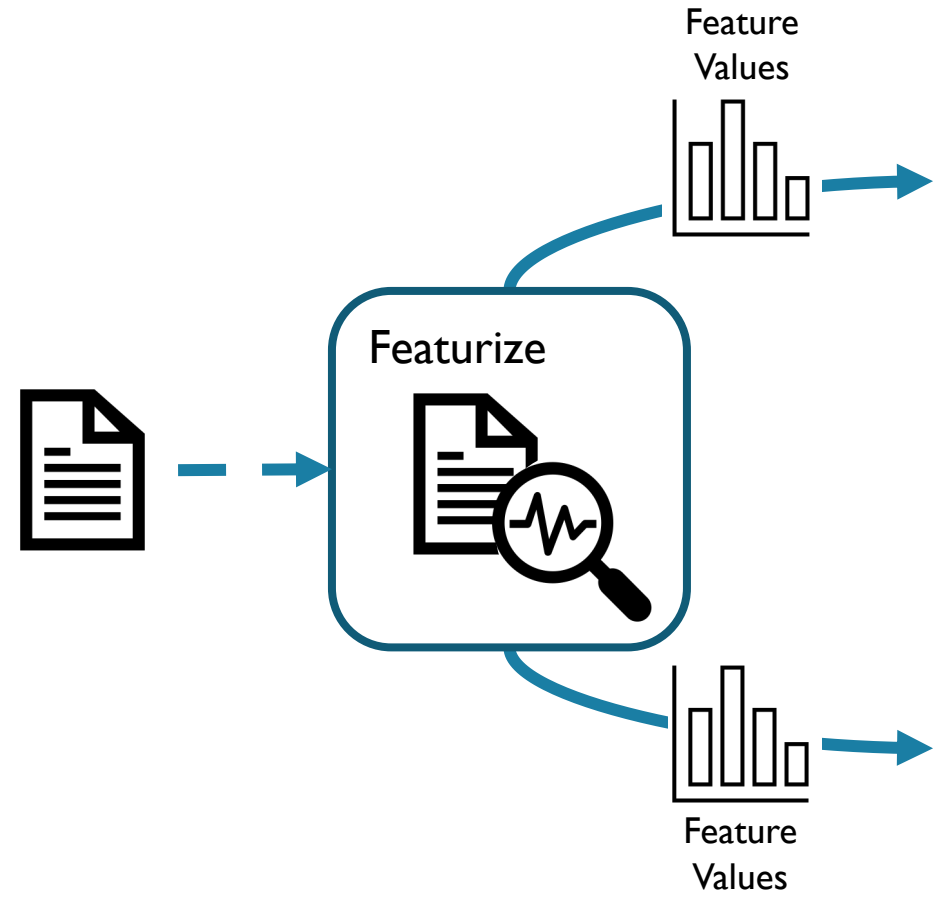


MedleySolver Overview

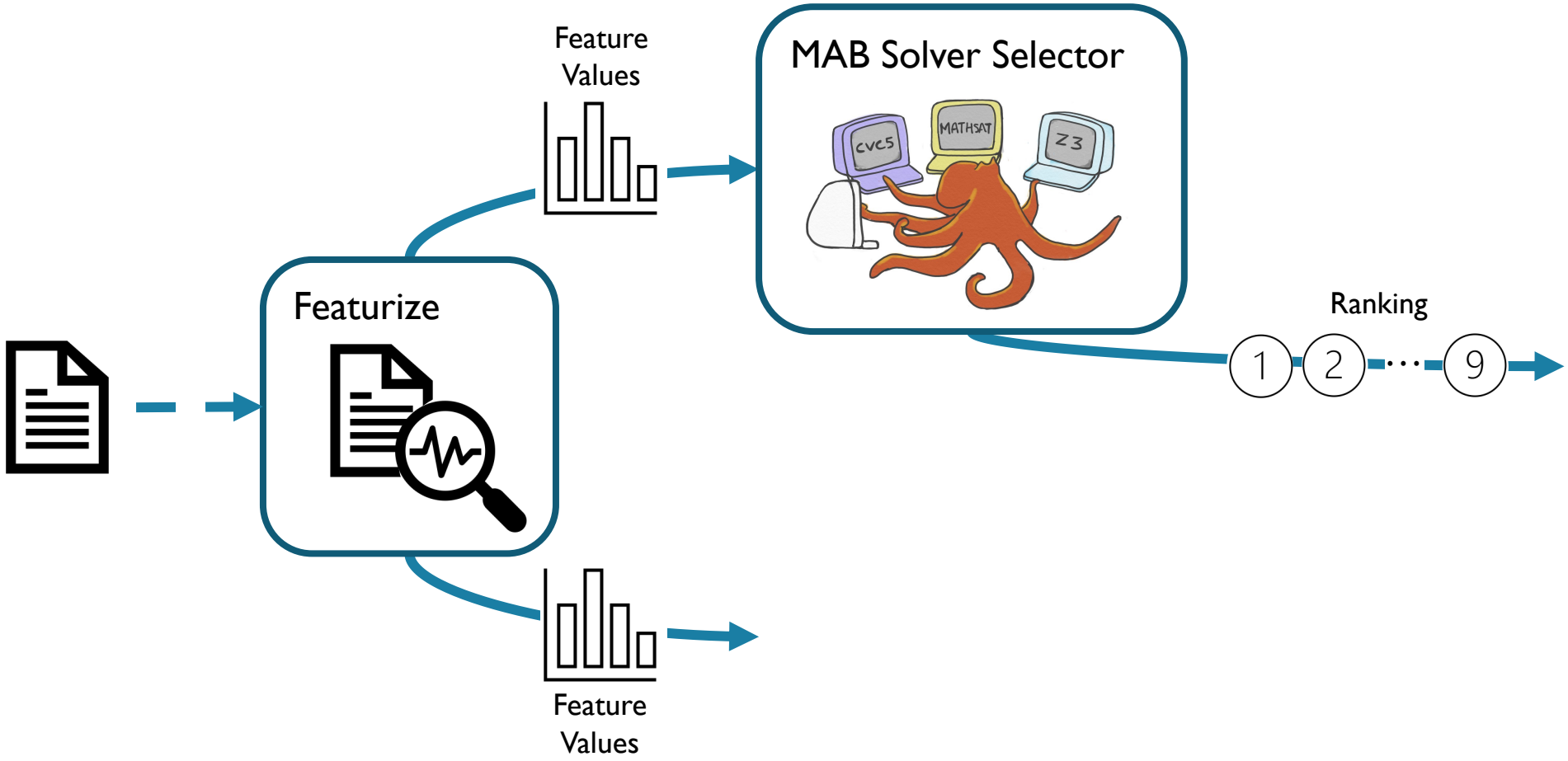
MedleySolver Overview



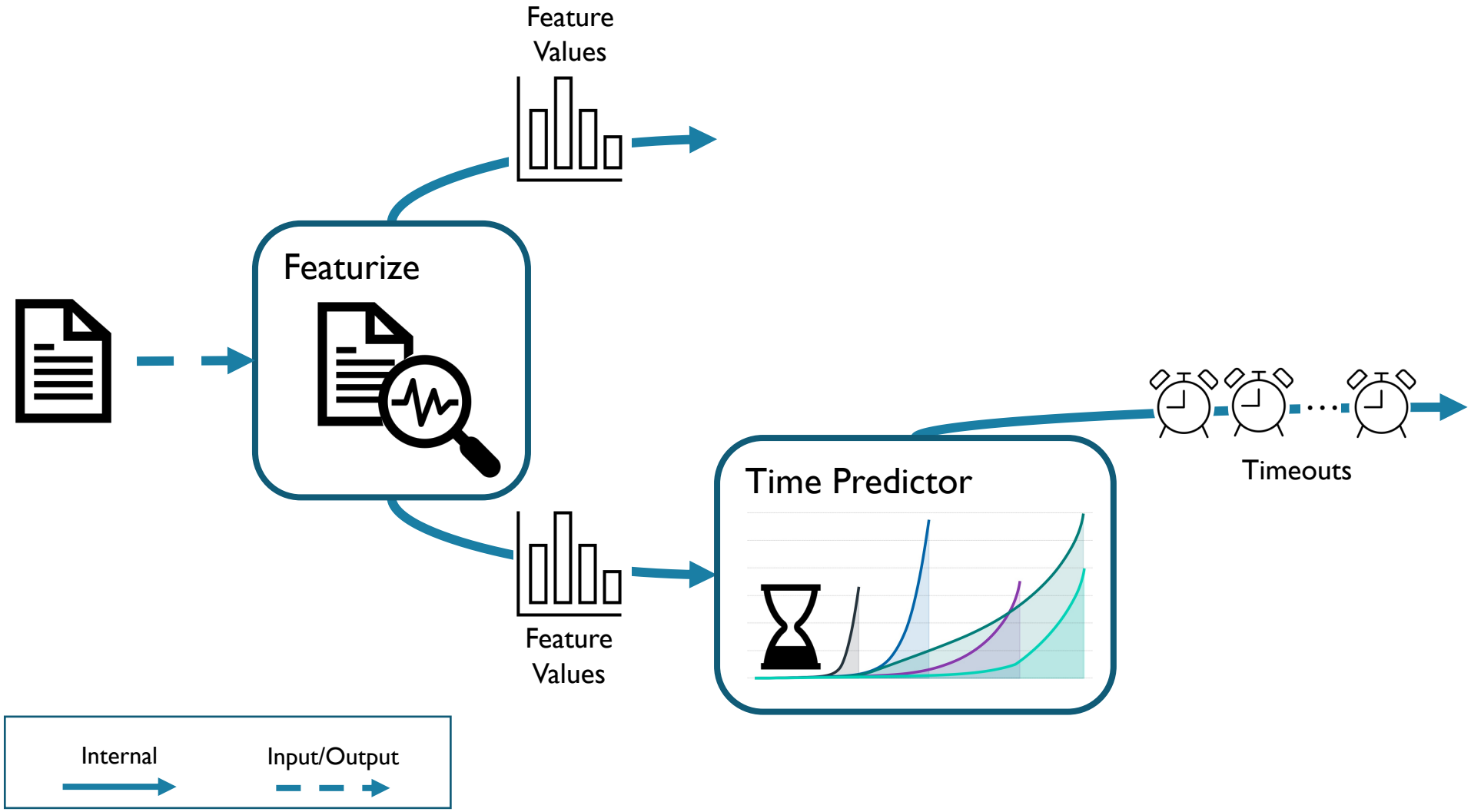
MedleySolver Overview



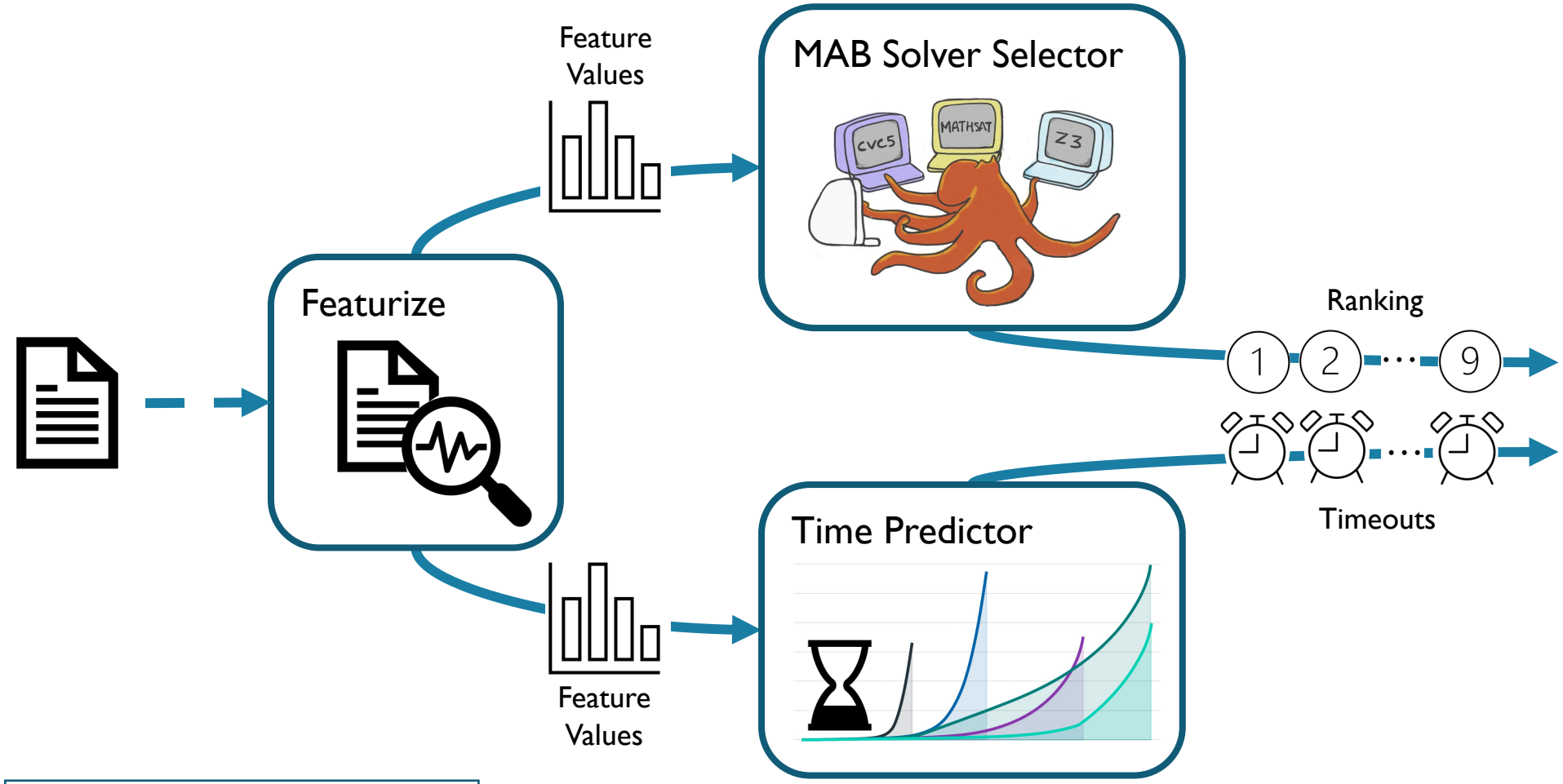
MedleySolver Overview



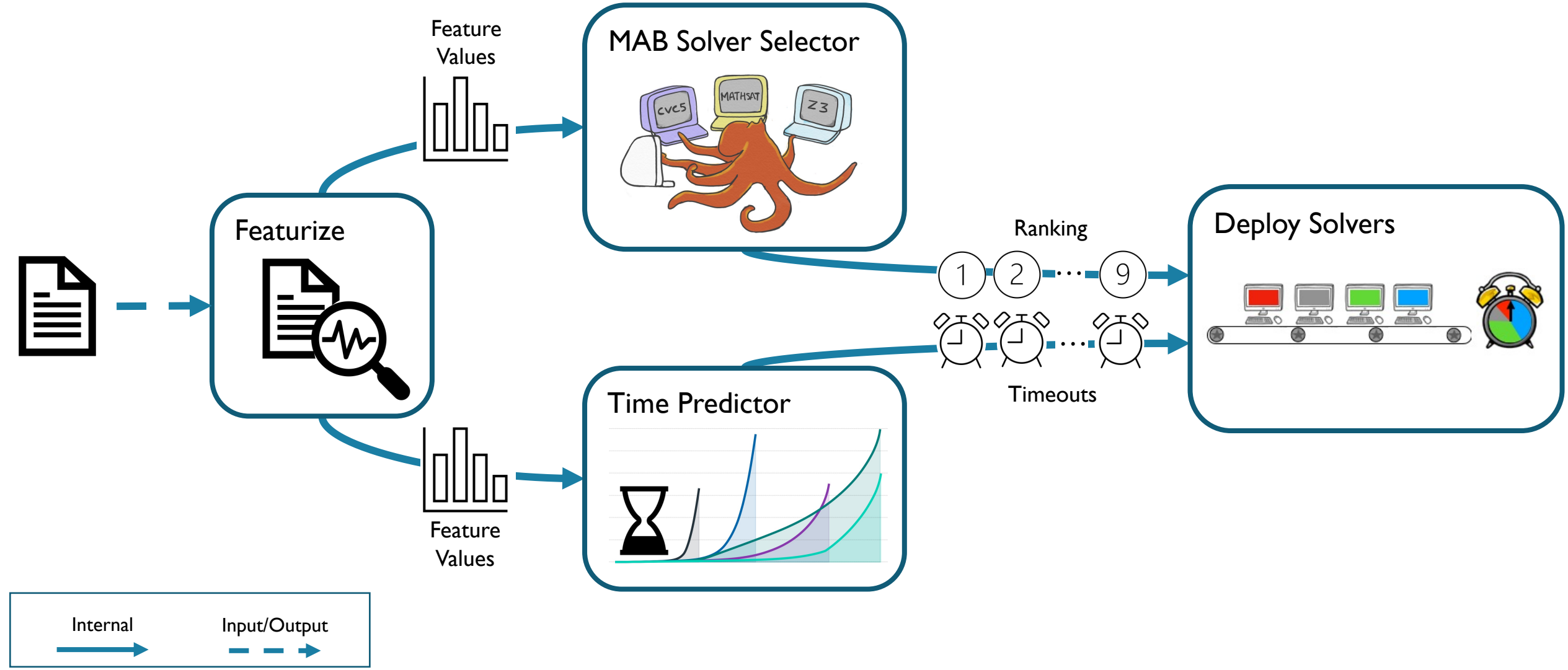
MedleySolver Overview



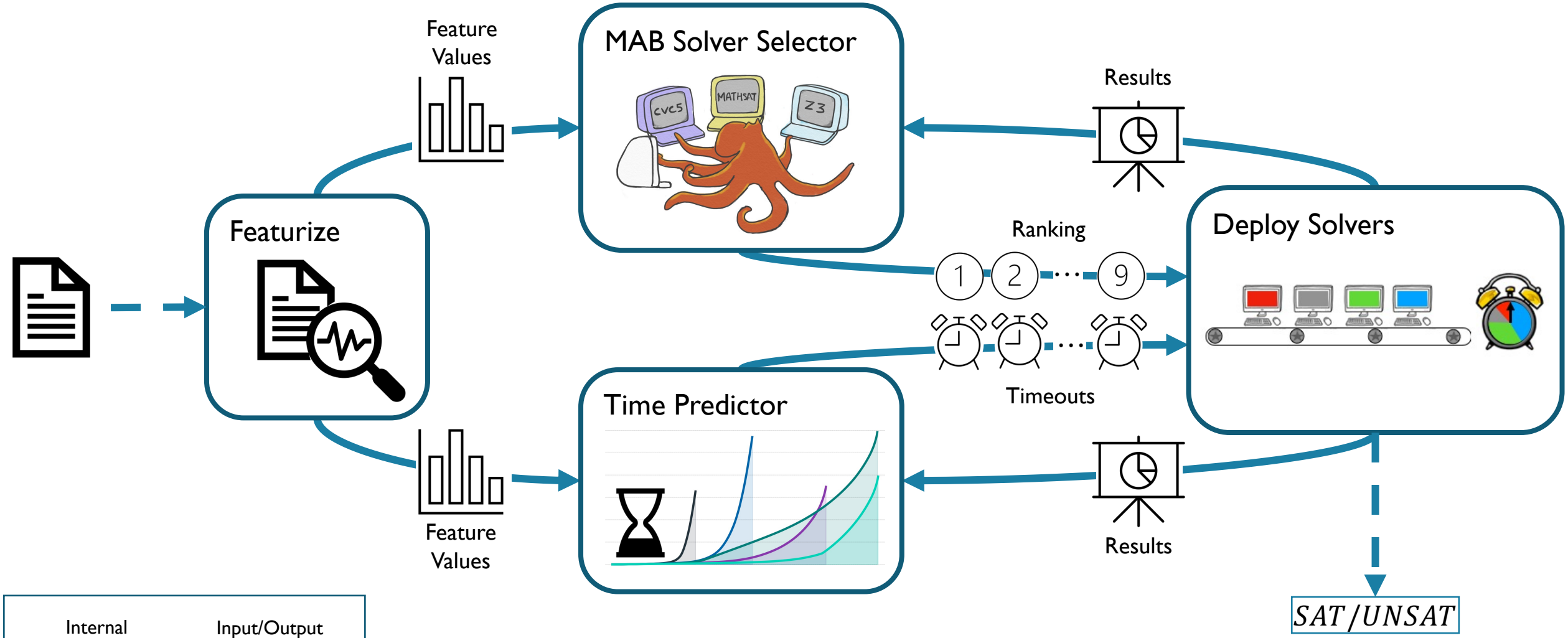
MedleySolver Overview



MedleySolver Overview



MedleySolver Overview

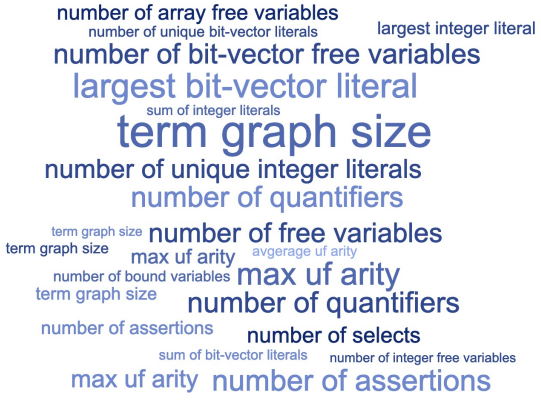


Computing Features

Feature Selection

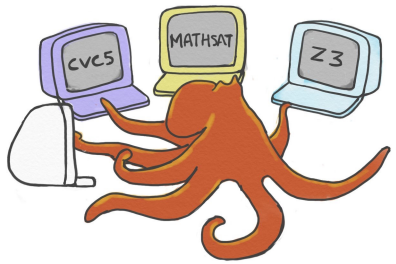
- Used a set of context-free features
  - E.g., "number of integer variables"
- Identified relevant features using Pearson R coefficient
  - Compute correlation of each feature against solving time, for each solver
  - Use 10 features with the highest average coefficient

Hypothetical feature correlation graphs (actual graphs are messier..)



Selecting Solver Order

MAB Solver Selector



Selecting Solver Order

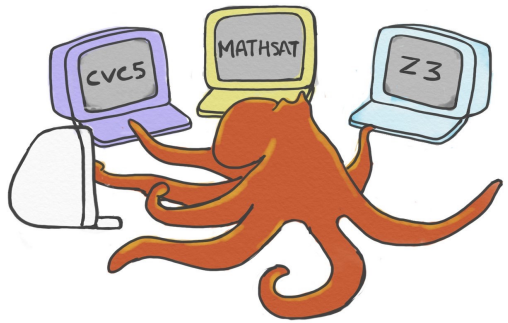
Multi-Armed Bandit Intuition

- Arms are slot machines
- Reward is payout
- Pick what slot machine to use

Our Multi-Armed Bandit Use

- Arms are solvers
- Reward is negative time taken
- Pick order of solvers to run

MAB Solver Selector



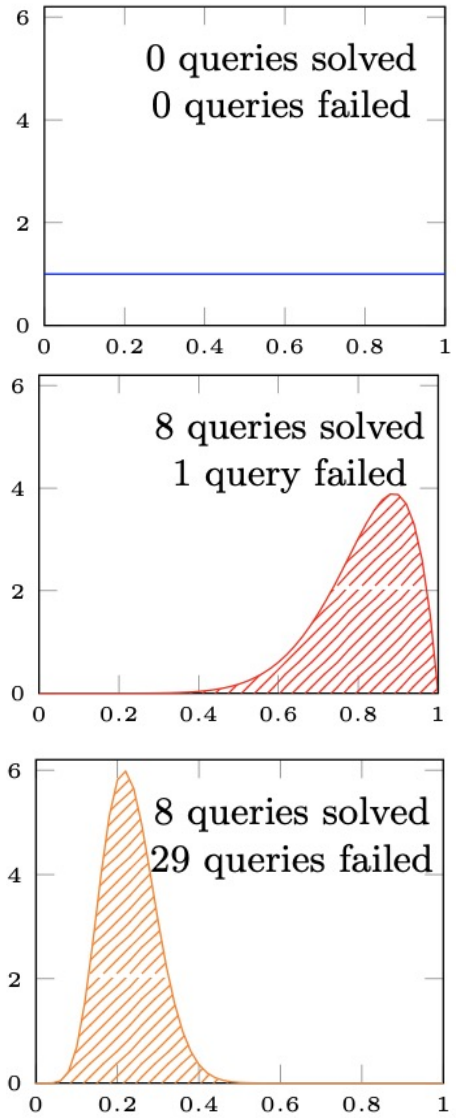
Implemented Algorithms

- Thompson Sampling
- k-Nearest-Neighbour Bandit
- Exp3
- Neural network bandit
- LinUCB

Thompson Sampling

Non-Contextual Algorithm 1

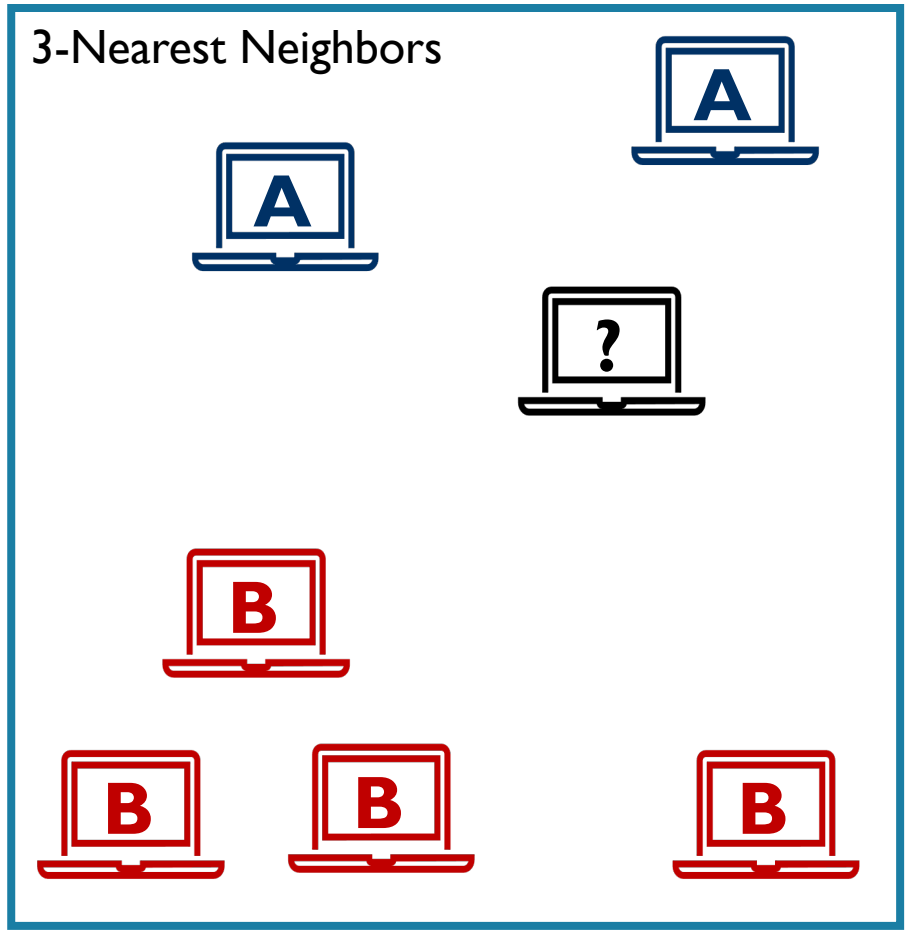
- Model rewards using probability distributions
  - (beta distribution)
- One distribution per solver
- Sample from each beta distribution
  - execute solvers in order of sample value
- Update using Bayes' rule
- Arms which are not thoroughly explored will have higher variance, making them more likely to be explored in the future



k-Nearest Neighbor

Contextual Algorithm I

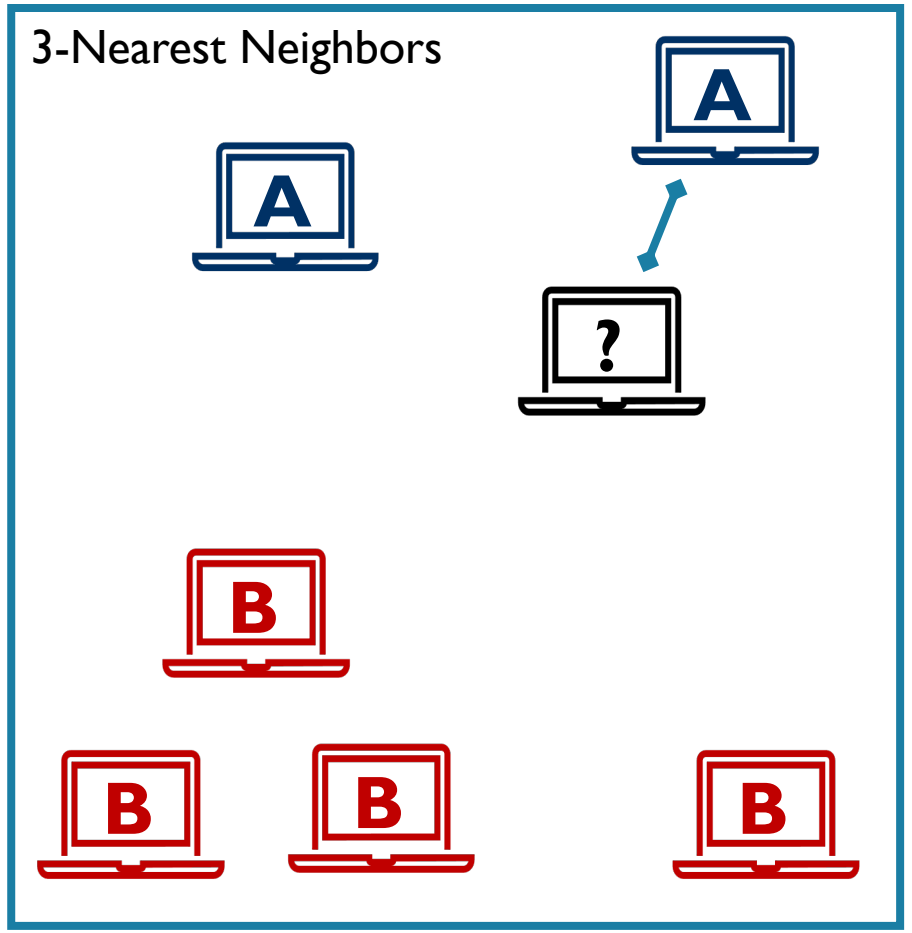
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

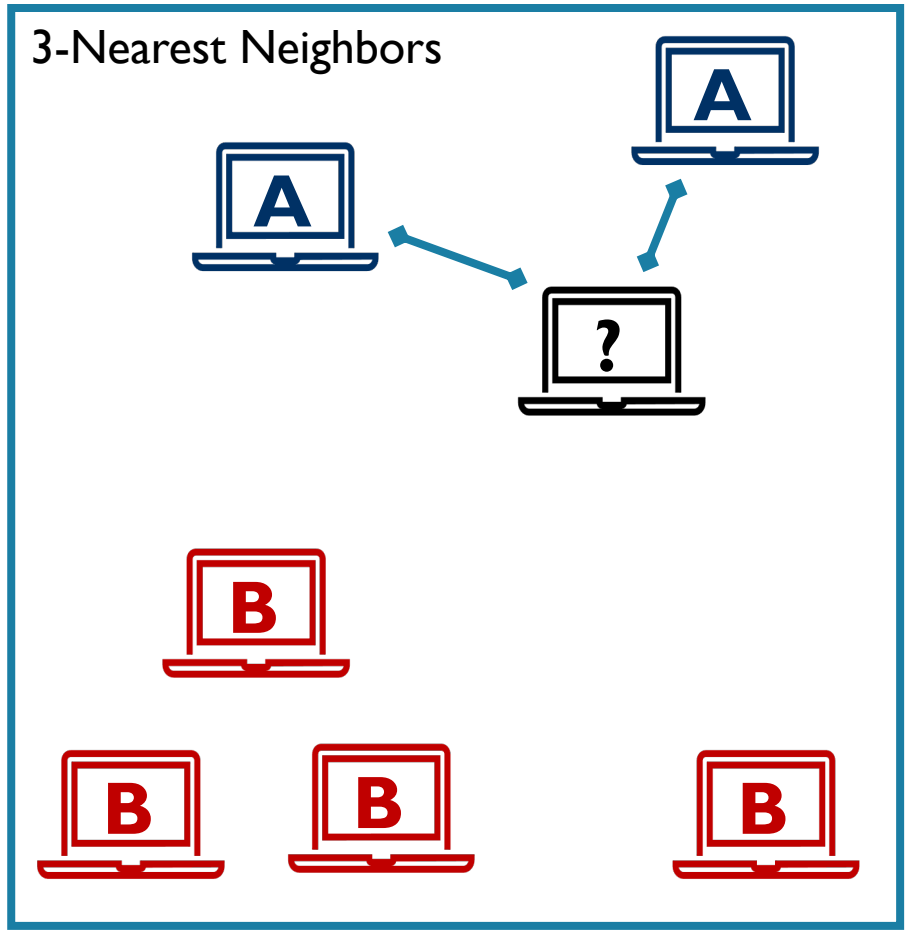
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

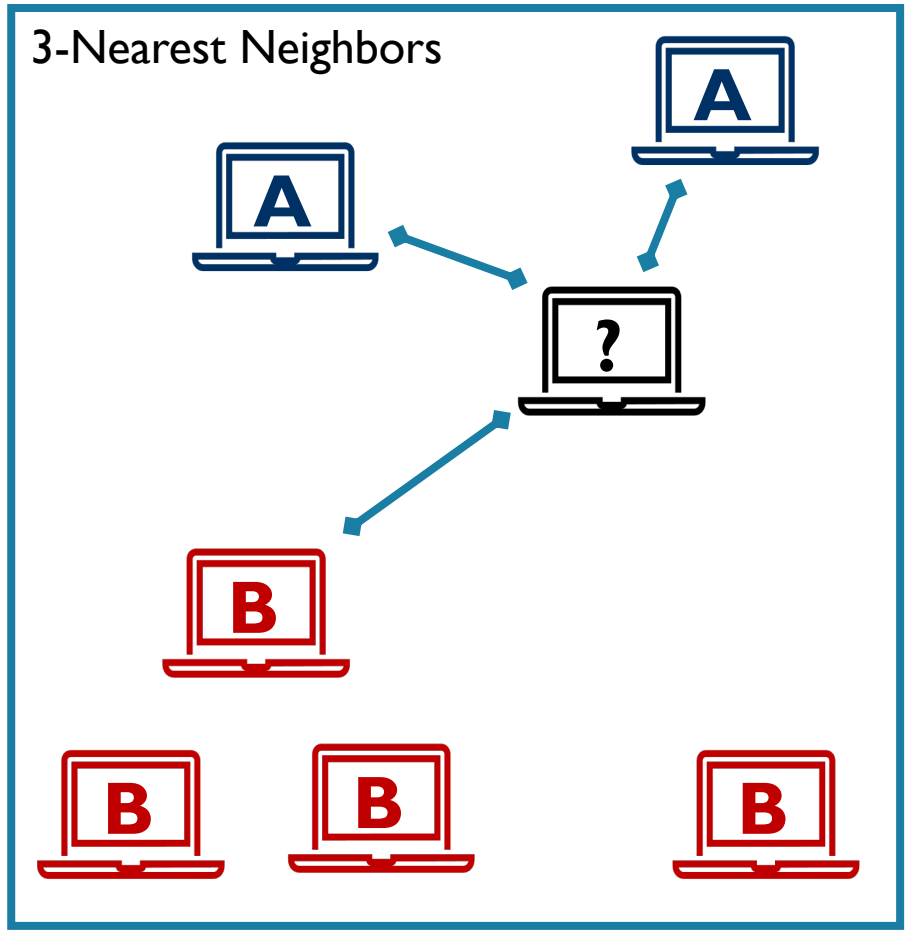
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

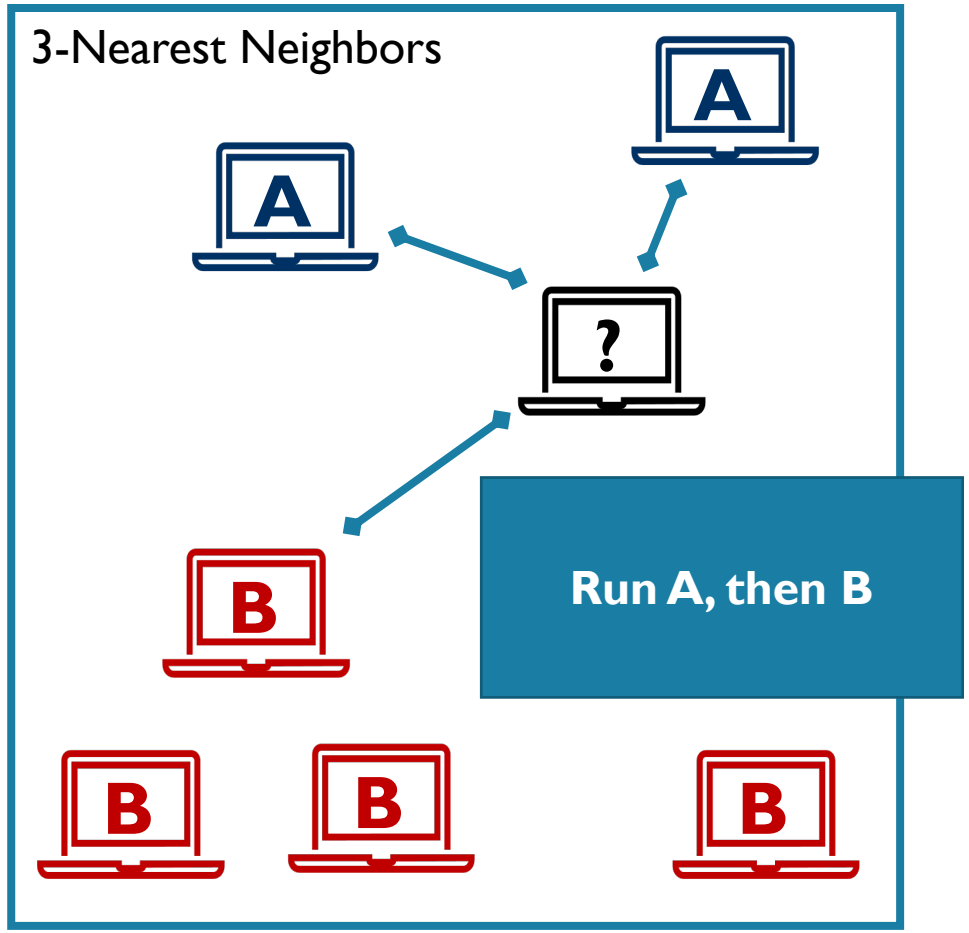
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

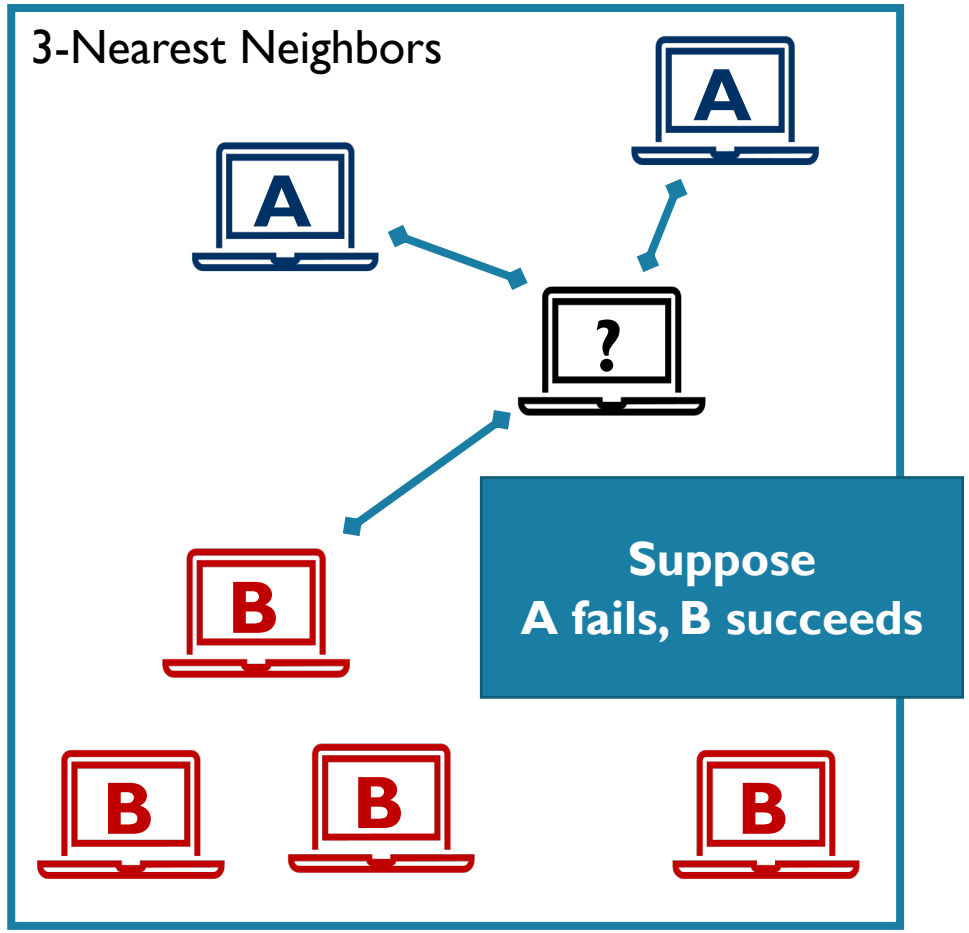
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

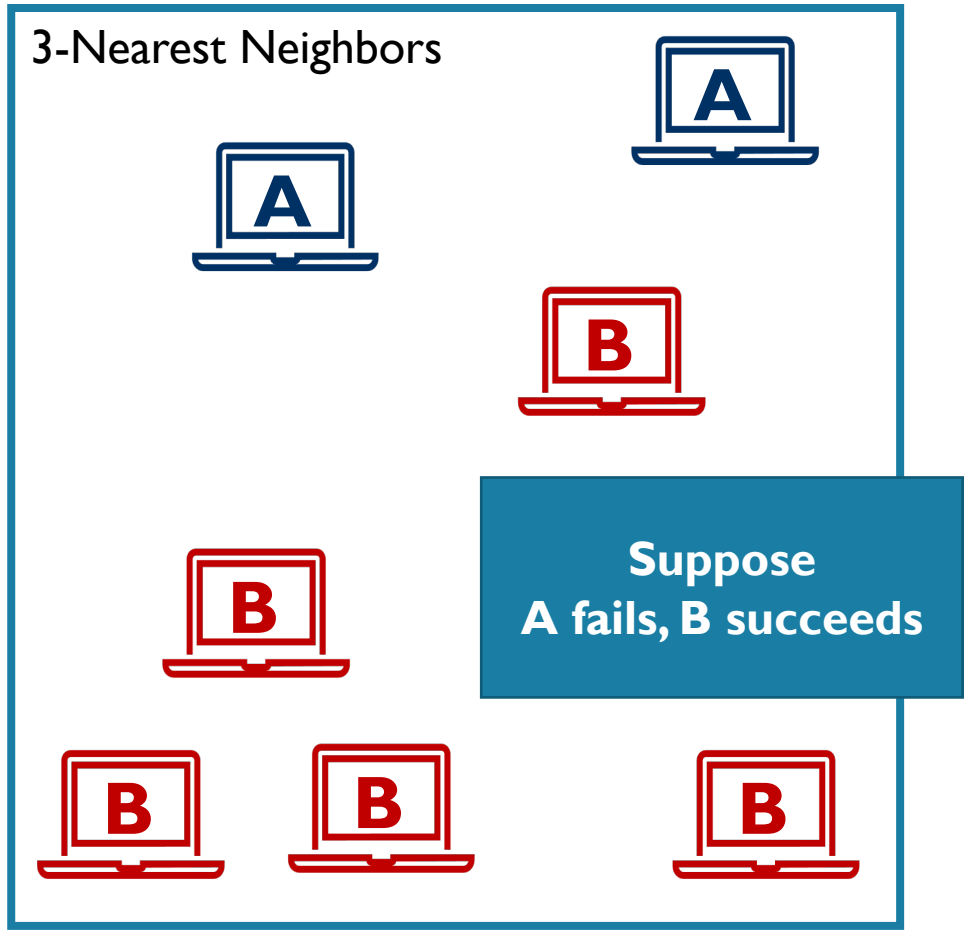
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



k-Nearest Neighbor

Contextual Algorithm I

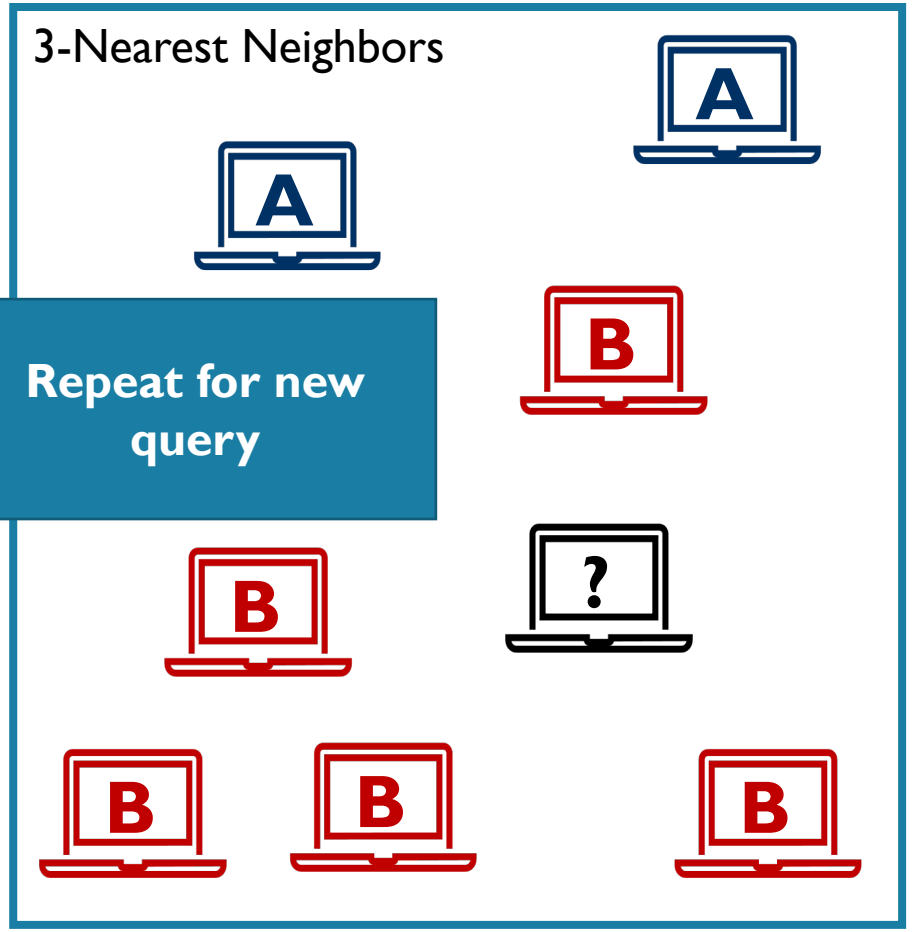
- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label



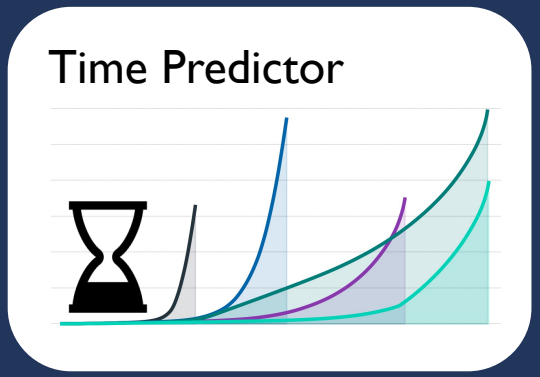
k-Nearest Neighbor

Contextual Algorithm I

- Given a query, k-NN looks at the k closest past solved queries and orders solvers by their number of appearances in these k, breaking ties randomly
- Update: if query k was solved, we put it into our list of past solved queries with the corresponding solver label

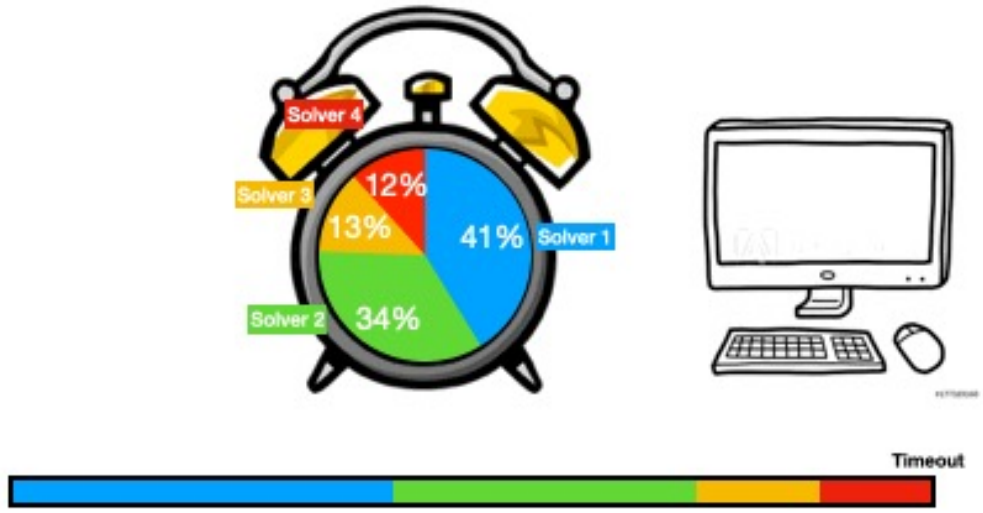


Allocating Timeouts



Allocating Timeouts

- Task: Split overall timeout T
  - allocate time to each solver in order.
- Cut off solver once we are confident it is unlikely to terminate



Modeling Execution Time as an Exponential

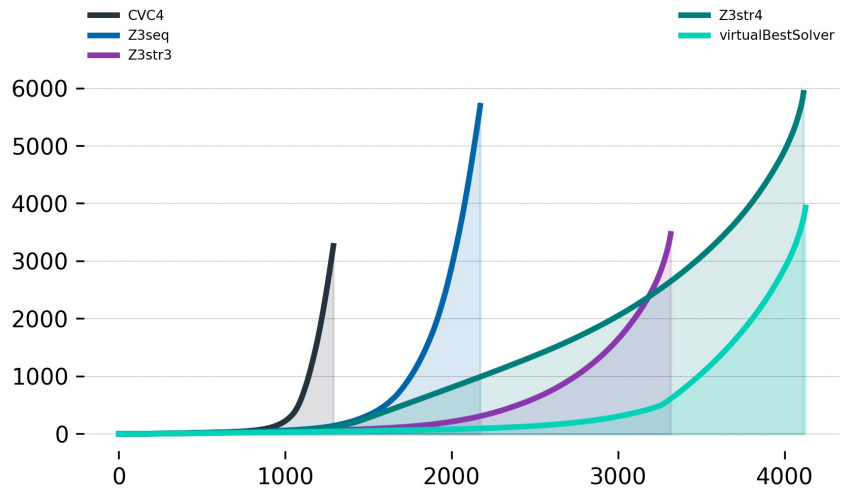
- Solver runtimes look like samples from an exponential distribution
- Estimate parameter λ using maximum likelihood estimation

$$\lambda^* = \frac{n}{\sum_i q_i}$$

- For a hyper-param δ, find t such that solver is δ% likely to terminate

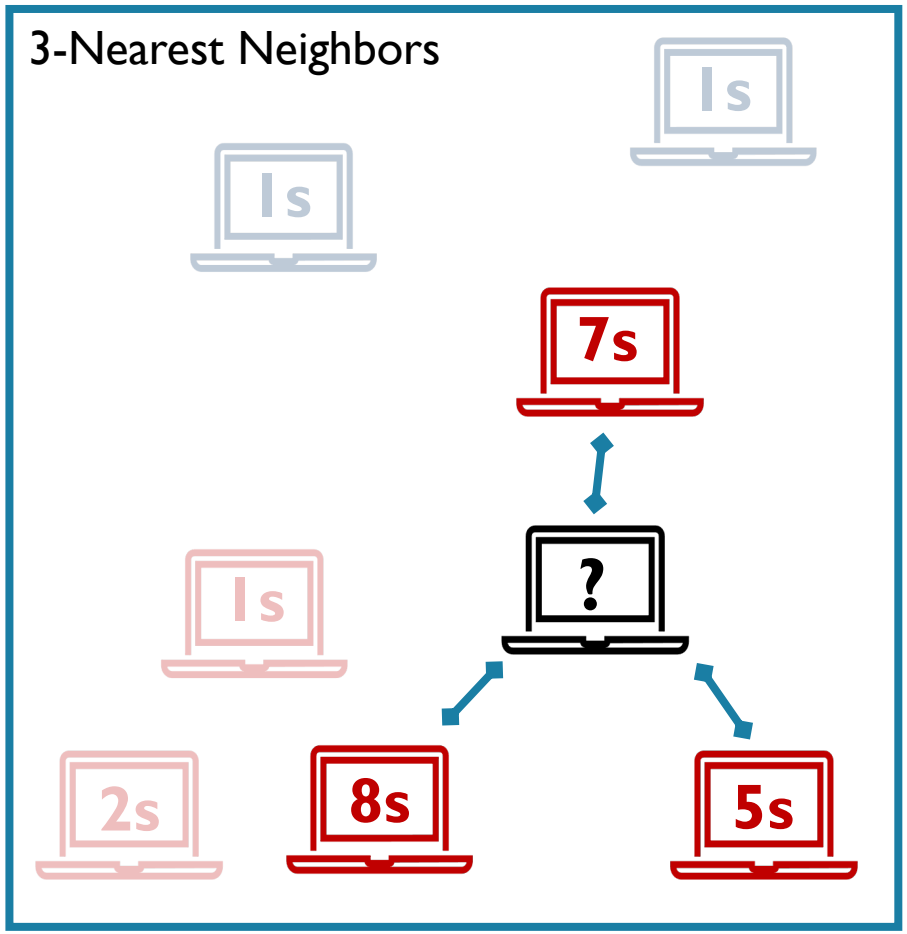
$$\mathrm{t} = \frac{-\ln(\delta + e^{-\lambda T})}{\lambda^*}$$

- After t seconds try the next solver!



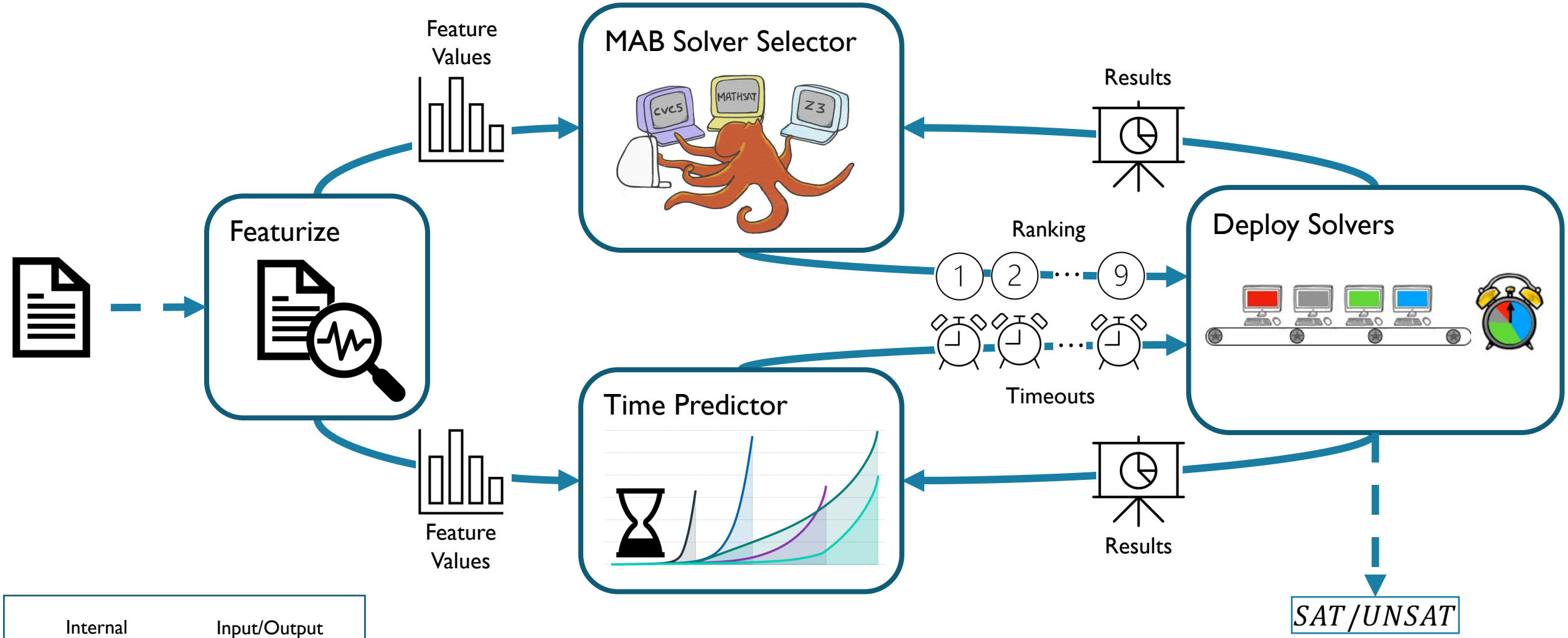
Contextual Time Allocation

- k-Nearest Neighbor Exponential:
  - Use the k-nearest queries to estimate λ
- Linear Regression:
  - Train a linear model which
    - takes in a query's feature vector as input,
    - outputs the expected runtime



Deploying Solvers and Recap

MedleySolver Overview



Desired Features for Algorithm Selection

(End-User Perspective)

Approach / Feature	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	
No Data Requirements	✓	▯	
No Solver Requirements	✗	✓	
No Need to Repeat Upfront Costs	✓	✗	
Fine Grained Decisions	✗	✓	

Feature Support

- ✓ Strong
- ▯ Medium
- ✗ Weak

Desired Features for Algorithm Selection

(End-User Perspective)

Feature \ Approach	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	✓ No pre-training
No Data Requirements	✓	☐	✓ No pre-training
No Solver Requirements	✗	✓	
No Need to Repeat Upfront Costs	✓	✗	
Fine Grained Decisions	✗	✓	

Feature Support

- ✓ Strong
- ☐ Medium
- ✗ Weak

Desired Features for Algorithm Selection

(End-User Perspective)

Feature \ Approach	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	✓ No pre-training
No Data Requirements	✓	▯	✓ No pre-training
No Solver Requirements	✗	✓	✓ No assumptions
No Need to Repeat Upfront Costs	✓	✗	
Fine Grained Decisions	✗	✓	

Feature Support

- ✓ Strong
- ▯ Medium
- ✗ Weak

Desired Features for Algorithm Selection

(End-User Perspective)

Feature \ Approach	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	✓ No pre-training
No Data Requirements	✓	▯	✓ No pre-training
No Solver Requirements	✗	✓	✓ No assumptions
No Need to Repeat Upfront Costs	✓	✗	✓ Adapt to changes as they come.
Fine Grained Decisions	✗	✓	

Feature Support

- ✓ Strong
- ▯ Medium
- ✗ Weak

Desired Features for Algorithm Selection

(End-User Perspective)

Feature \ Approach	Expert Encoded Decision Rule	Existing Methods (Offline Learning)	MedleySolver (Online Learning)
No Manual Input	✓	✓	✓
Minimal Upfront Costs	✓	✗	✓ No pre-training
No Data Requirements	✓	▯	✓ No pre-training
No Solver Requirements	✗	✓	✓ No assumptions
No Need to Repeat Upfront Costs	✓	✗	✓ Adapt to changes as they come.
Fine Grained Decisions	✗	✓	✓ Complex, flexible decision rules

Feature Support

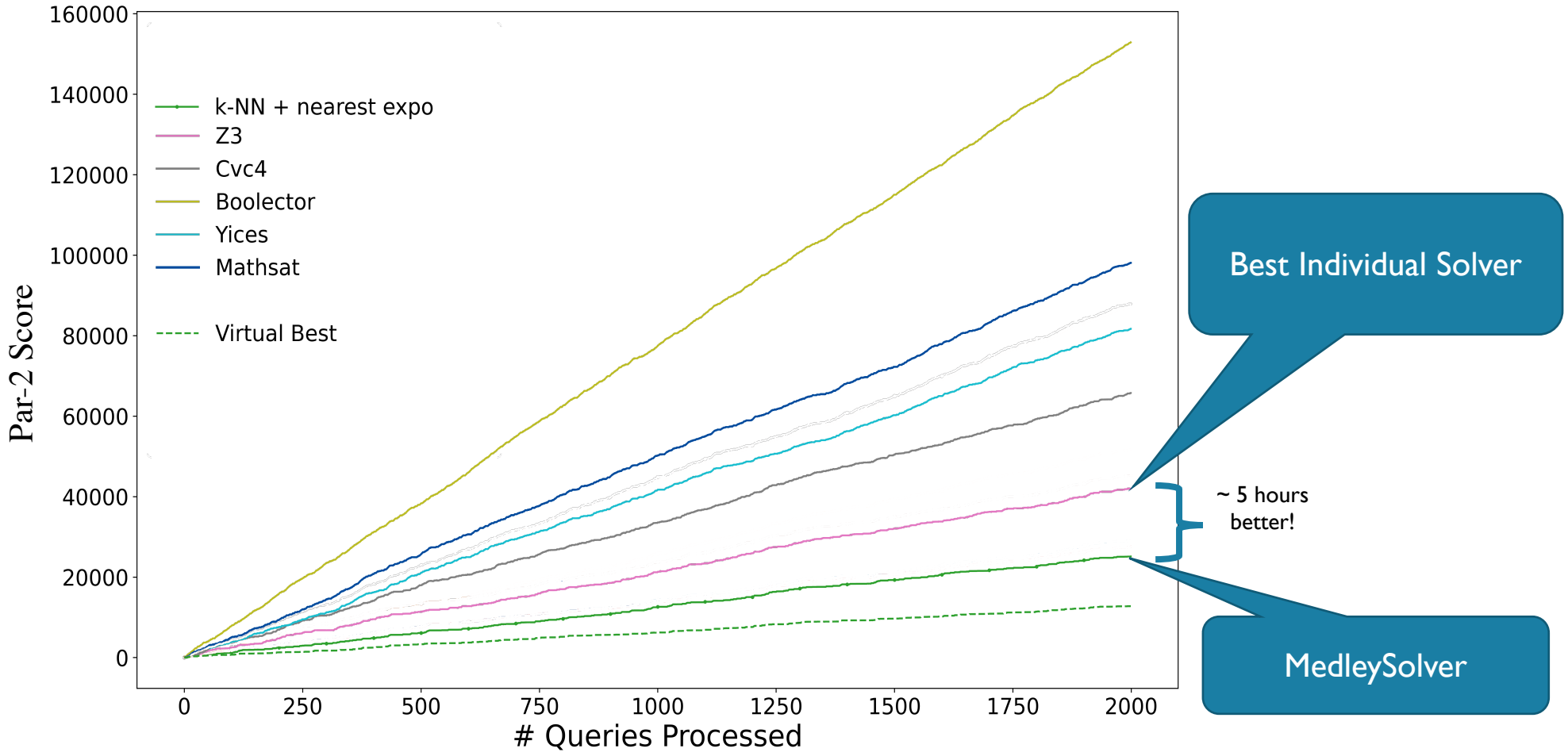
- ✓ Strong
- ▯ Medium
- ✗ Weak

Evaluation and Future Work

Experimental Setup

- 6 individual SMT solvers
  - CVC4, MathSAT, Z3, Boolector, Bitwuzla, and Yices
- 4 individual benchmark sets
  - BV, QF_ABV, Sage2, and Uclid5 (first 3 sets are from SMTCOMP)
  - 500 queries each (randomly sampled without replacement)
- 6 MedleySolver configurations in paper, 1 for presentation
  - 10-Nearest Neighbor for order selector
  - 10-Nearest Neighbor exponential distribution estimation for time predictor
- Dell PowerEdge C6220 server, 60 second timeout per query.

Exp. 1: Comparison to Individual Solvers



Exp. 2: Comparison to Pre-Trained Tools

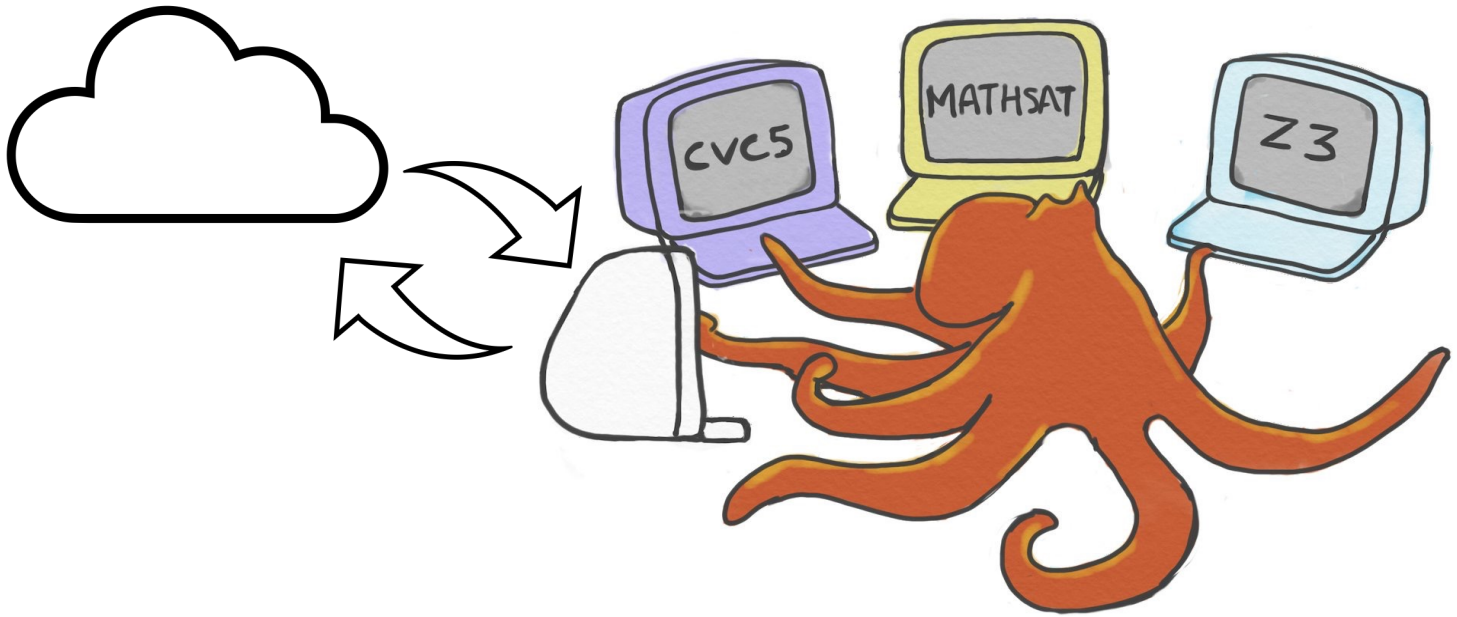
- Compare against MachSMT
  - State-of-the-art
- Report par-2 score for test
- Report time in seconds for training

- 40% of queries for training
  - Only used by MachSMT
- 60% of queries for testing
  - Used by both

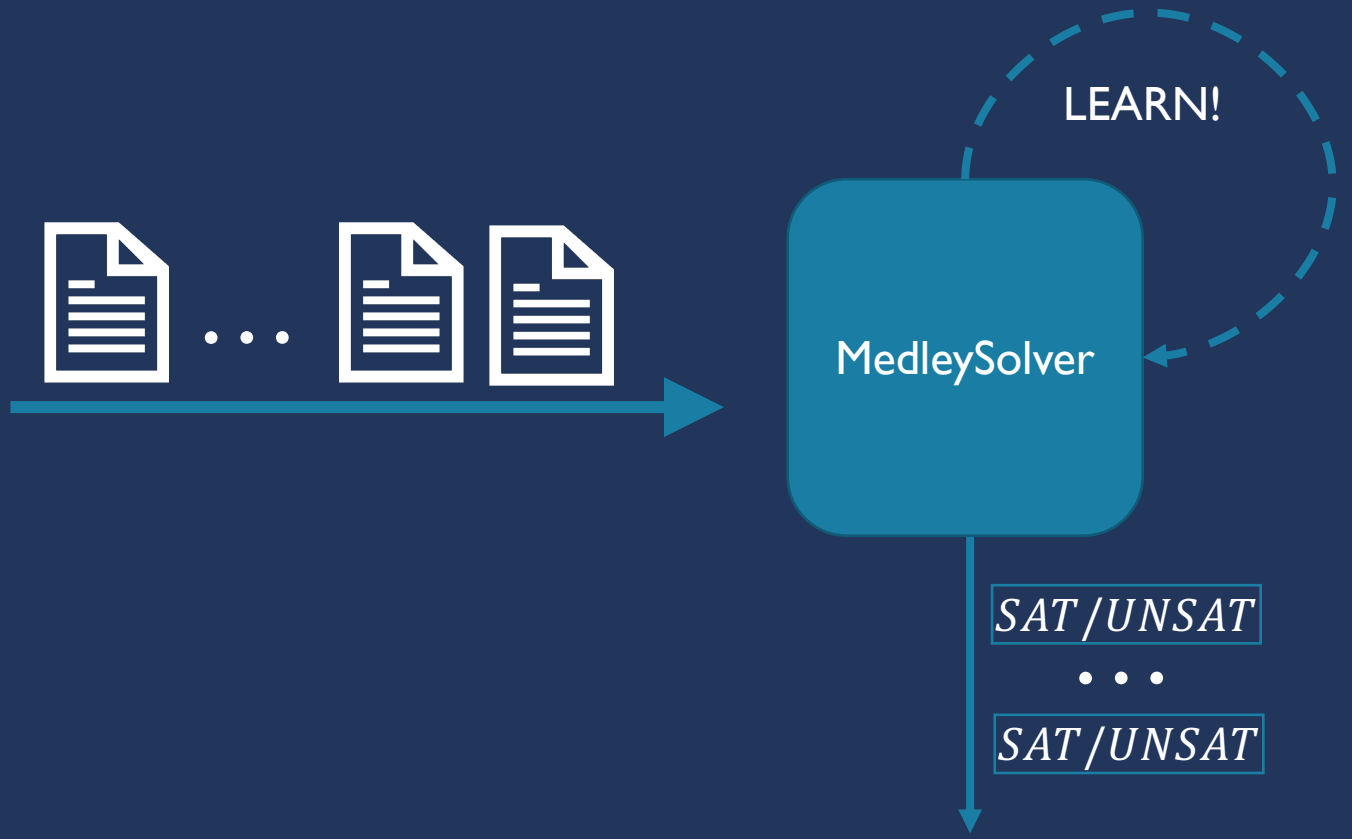
Benchmark / Tool	BV	QF_ABV	Sage2	Uclid5	Combined
MedleySolver	Test: 1638.7	Test: **310.5**	Test: 9245.3	Test: 4248.0	Test: 18565.5
MachSMT	Test: **1458.3** Train: 33895.5	Test: 919.2 Train: 4498.9	Test: **8516.1** Train: 55115.5	Test: **2430.9** Train: 276419.8	Test: **12539.1** Train: 300072.8
Virtual Best	Test: 801.7	Test: 184.3	Test: 5204.2	Test: 1464.7	Test: 6746.0

Future Work

- Whitebox monitoring techniques
  - instead of black box timeout estimation
- What if one of the solvers is a distributed solver in the cloud?
  - Front end to decide when to solve locally and when to query server



MedleySolver: Online SMT Algorithm Selection



Paper PDF

Code & Data