Frugal Algorithm Selection

Erdem Kuş, Özgür Akgün, Nguyen Dang, Ian Miguel

University of St Andrews, School of Computer Science
ek232@st-andrews.ac.uk

Algorithm Selection

- Many algorithms are proposed for many problems.
  - There are more than 30 sorting algorithms proposed
- No Free Lunch: There is no universally best algorithm

Name	Best	Average	Worst	Memory	Stable	Method
Block sort	n	$n \log n$	$n \log n$	1	Yes	Insertion & Merging
Bubble sort	n	n^2	n^2	1	Yes	Exchanging
Cocktail shaker sort	n	n^2	n^2	1	Yes	Exchanging
Comb sort	$n \log n$	n^2	n^2	1	No	Exchanging
Cubesort	n	$n \log n$	$n \log n$	n	Yes	Insertion
Cycle sort	n^2	n^2	n^2	1	No	Selection
Exchange sort	n^2	n^2	n^2	1	No	Exchanging
Gnome sort	n	n^2	n^2	1	Yes	Exchanging
Heapsort	$n \log n$	$n \log n$	$n \log n$	1	No	Selection
In-place merge sort	—	—	$n \log^2 n$	1	Yes	Merging

Source: *Sorting algorithm*. Wikipedia. Available at: https://en.wikipedia.org/wiki/Sorting_algorithm.

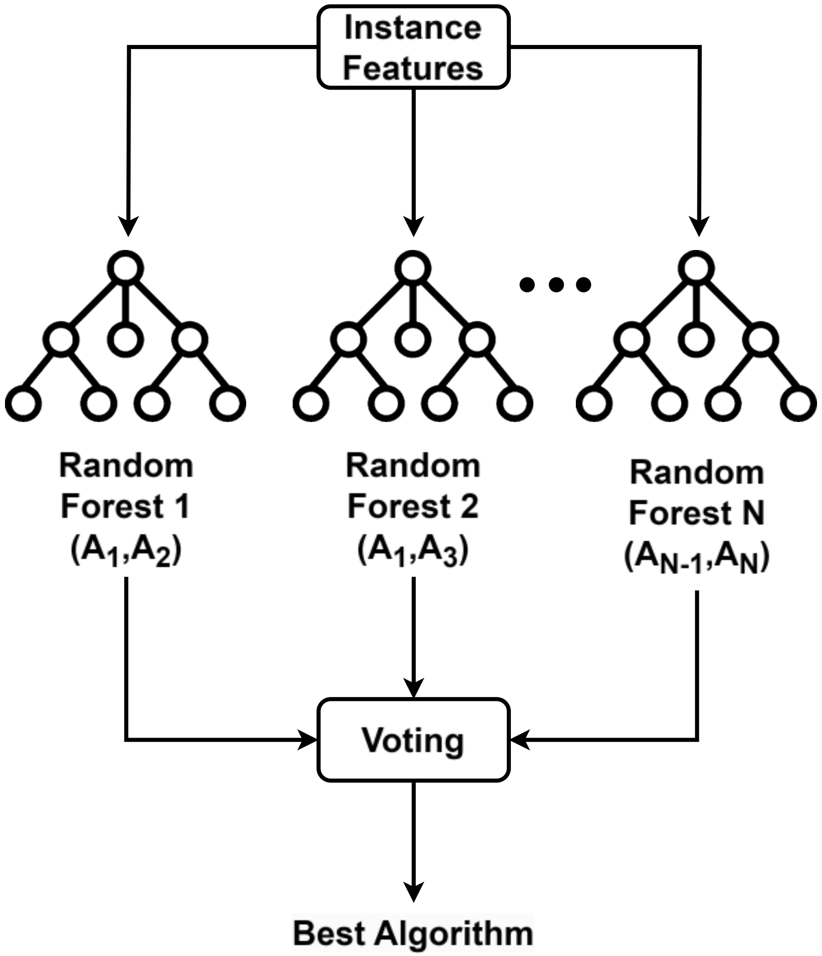
Algorithm Selection

Algorithm Selection (AS): Predict the **best algorithm** for each problem **instance**.

Training Data: Instance features

Label: Best algorithm

Classification Type: Pairwise (binary) classification



Expensive Training

Labelling cost: Time spent evaluating all algorithms across all instances to identify the best-performing ones for training.

Datasets

Dataset	Instances	Algorithms	Features	Total Time
ASP-POTASSCO	1294	11	138	87 days
CPMP-2015	527	4	22	28 days
CSP-2010	2024	2	86	18 days
MAXSAT12-PMS	876	6	37	61 days
MAXSAT19-UCMS	572	7	54	23 days
QBF-2011	1368	5	46	15 days

Expensive Training: MAXSAT12-PMS

- The top 2 algorithms perform 4 times faster than the bottom 2 algorithms.
- We pay **76%** of the cost for the bottom 3 algorithms
- Both **good** and **bad** solvers must be run

Algorithm	Runtime (Hours)
qmaxsat0.21g2comp	85h
qmaxsat0.21comp	99h
pwbo2.1	163h
DSWPM1_924	358h
akmaxsat	383h
akmaxsat_ls	383h

Motivation

1) Can we find a smaller subset that can give the same predictive performance?

- We can use Active Learning.

Active Learning

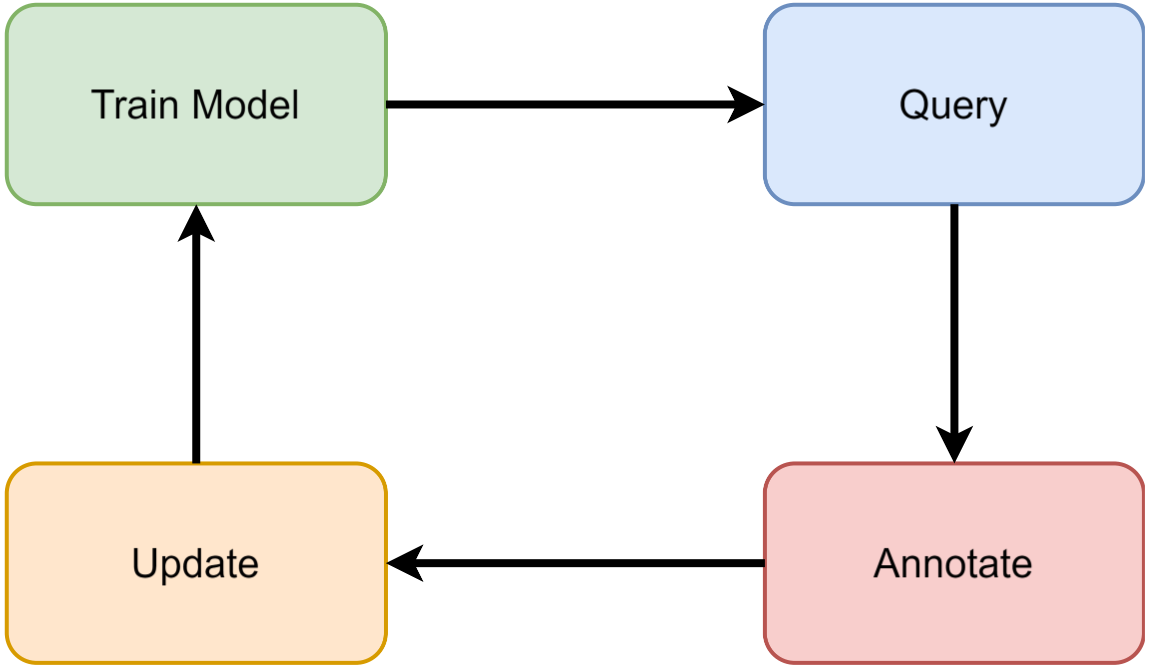
1) Train model with small labelled set

2) Select the most informative data points from the unlaballed set using the table

3) Label selected data points by an oracle

4) Update the labelled set with newly labeled data.

5) Repeat



Active Learning-An Example Query Table

Query size

Model	Instance	Uncertainty
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree505p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree610p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree525p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree430p.wcnf	0.5
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5

⋮

How many datapoints will be queried

Measurement of informativeness

Experimental Setup

Dataset: 6 datasets from AsLib.

Seed: 5 seeds, 10 splits

Instance Selection Method: Active learning (uncertainty-based) & random-based.

Query Size: 1% of the query table.

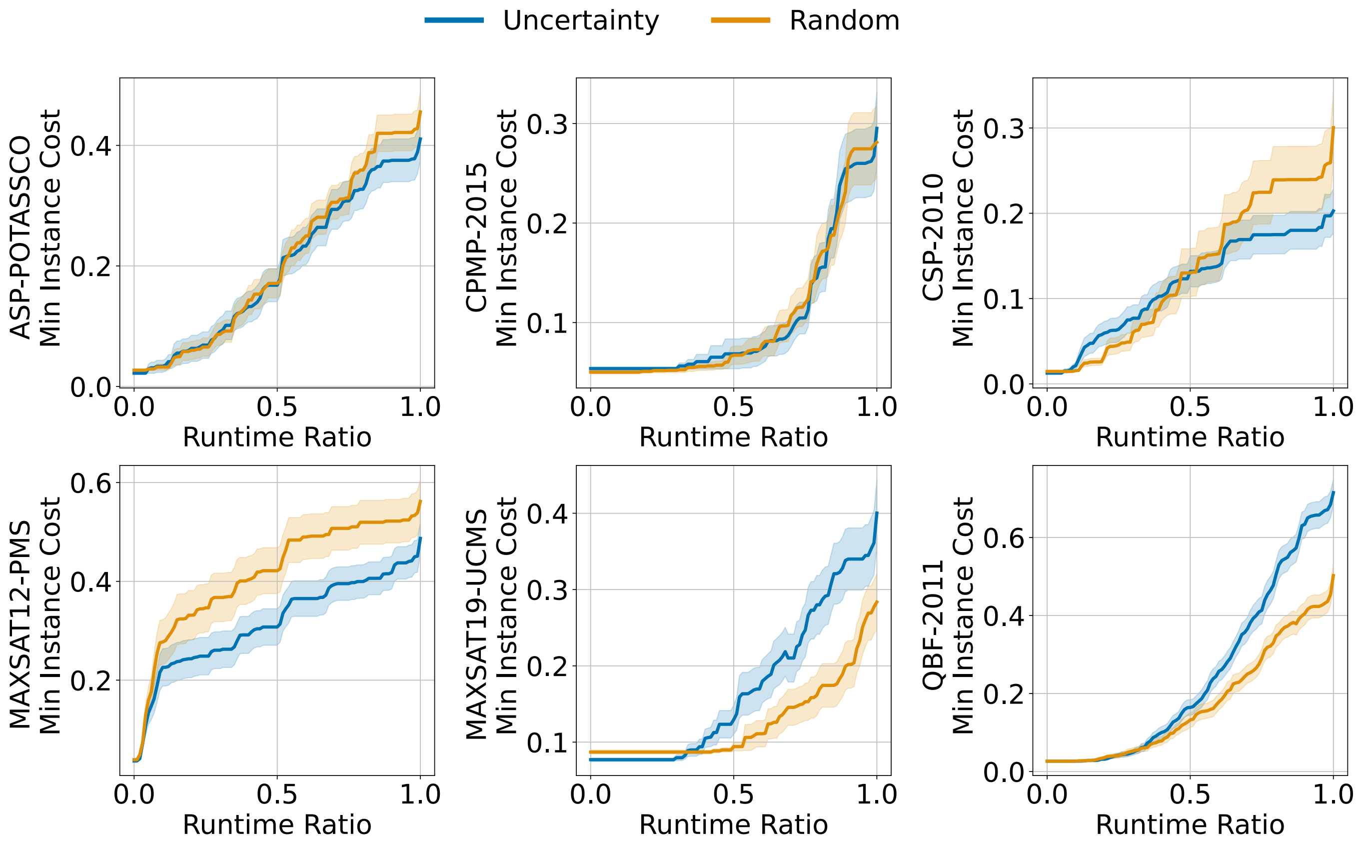
Experimental Results – Active Learning

x-axis: Performance comparison of AS model on selected subset vs. full dataset (1 = same performance).

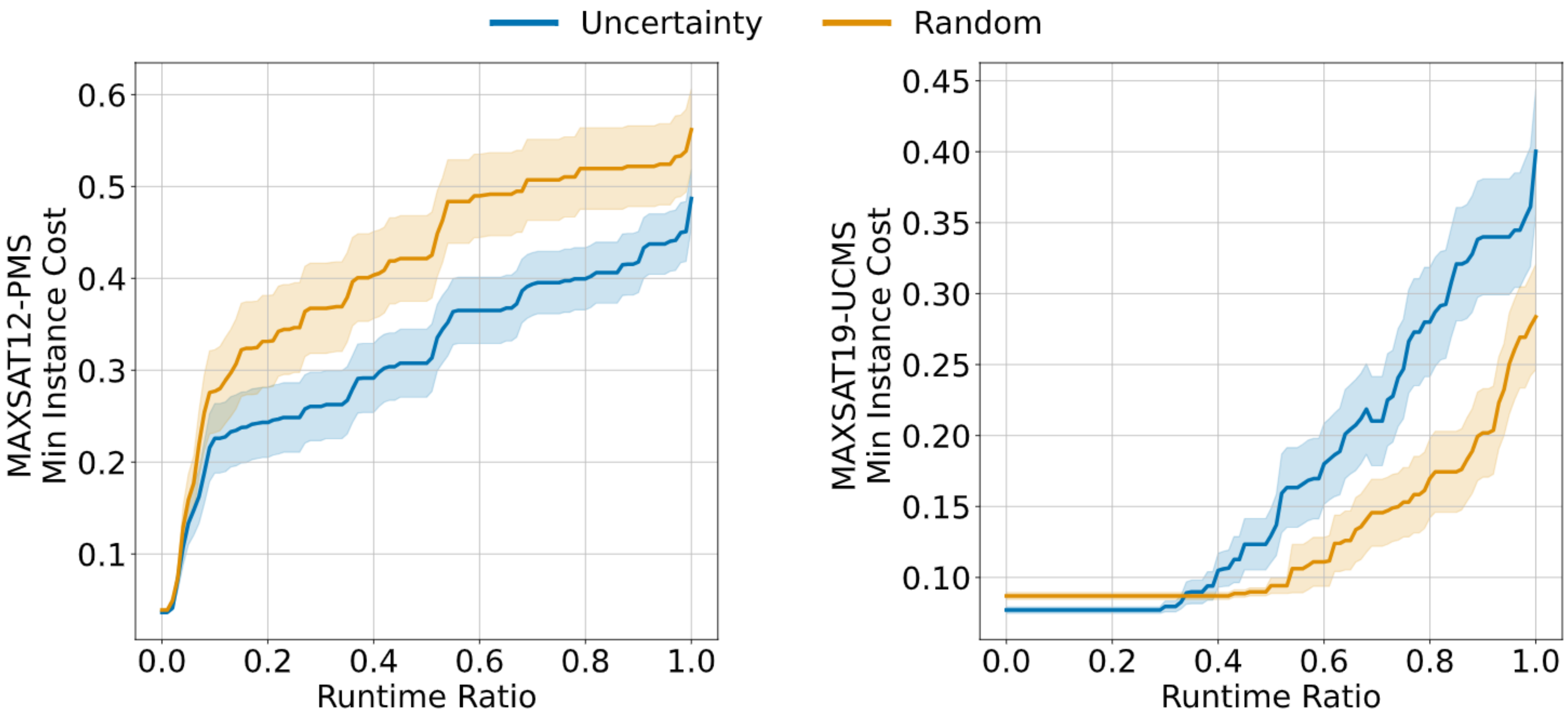
y-axis: Labelling cost of the selected subset (0 = no data, 1 = full data).

- No clear difference between random approach and active learning

Why?



Experimental Results – Active Learning



Problems in Active Learning

Uniform Cost: Ignores varying labelling costs.

Expensive Query: Top candidates can be expensive instances

Problems – Query Table

Model	Instance	Uncertainty	Algorithm1	Algorithm2
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5	**0.75**	**0.39**
('akmaxsat', 'akmaxsat_ls')	10tree505p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree610p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree525p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree430p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5	**0.87**	**410.12**

Query

Labelling

Query Table

Model	Instance	Uncertainty
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree505p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree610p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree525p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5
('akmaxsat', 'akmaxsat_ls')	10tree430p.wcnf	0.5
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5

Can we **eliminate** these instances in **query**?

Motivation

1) Can we find a smaller subset that can give the same predictive performance?

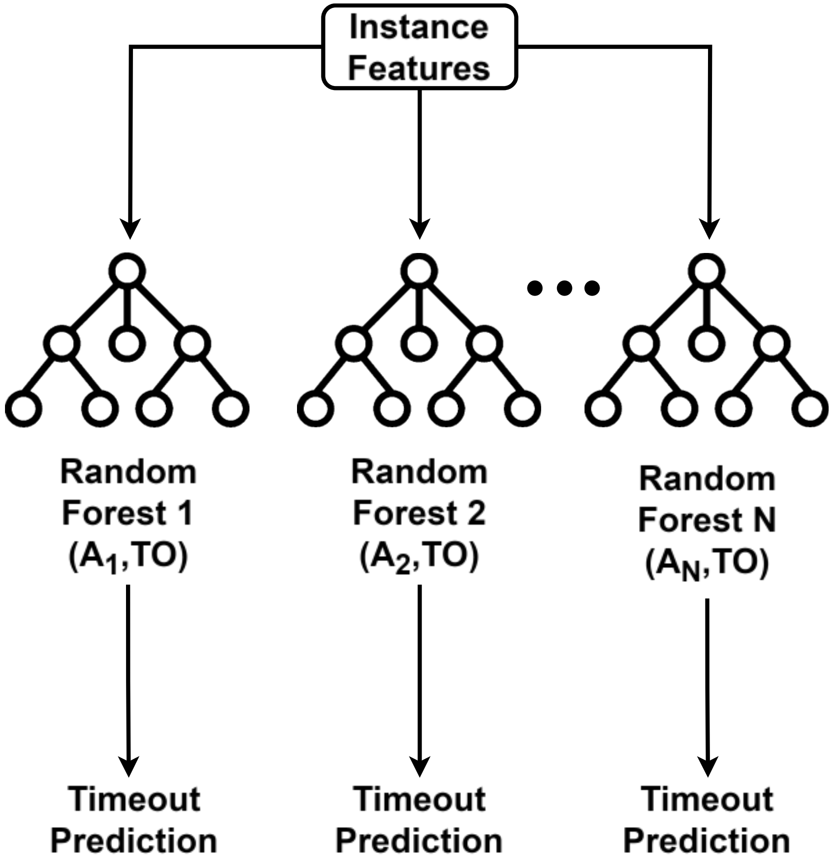
- We can use Active Learning.
- We saw that Active Learning selects uninformative and expensive instances in query.

2) How can we eliminate costly, uninformative instances in query table?

- Timeout Predictor

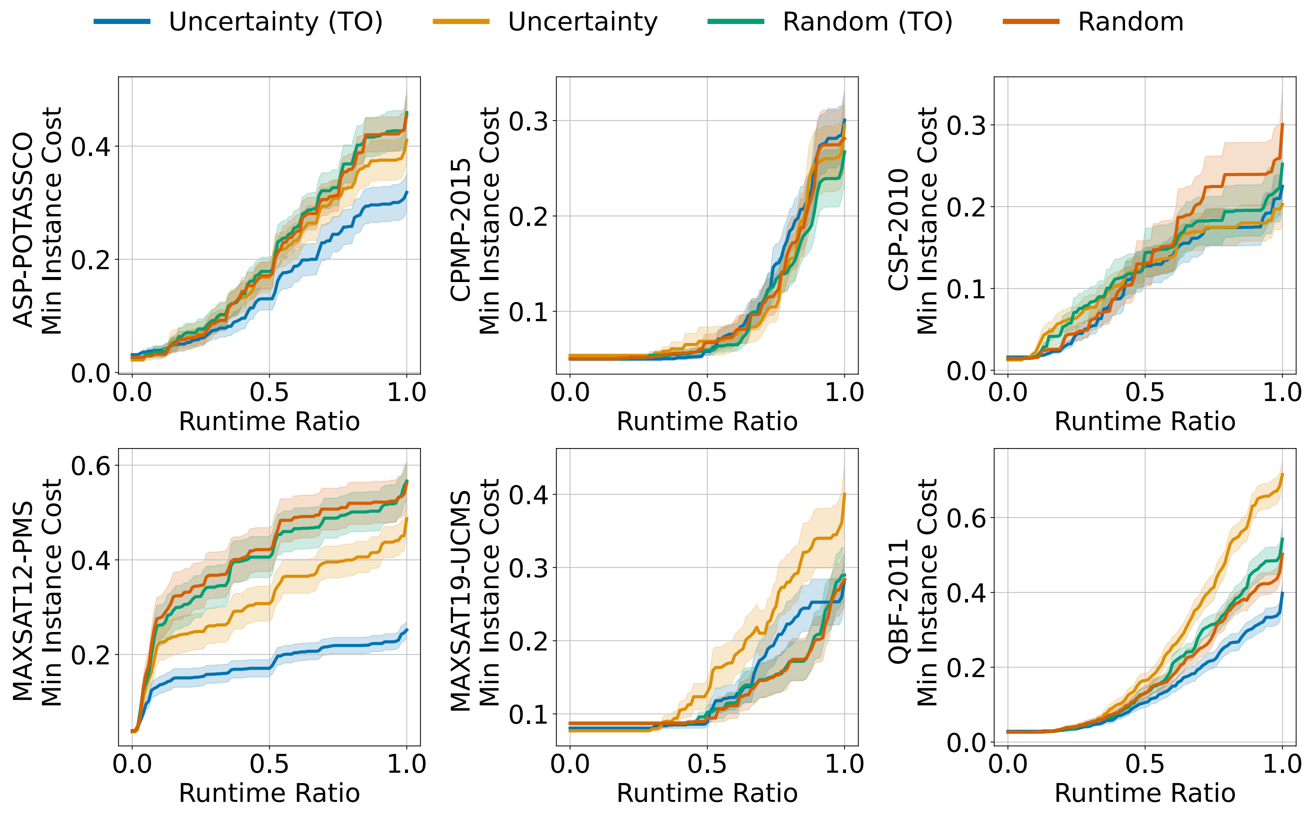
Frugal Algorithm Selection-Timeout Predictor

- Train timeout predictor for each algorithm.
- Use timeout predictions to eliminate timeout instances in **query** step.



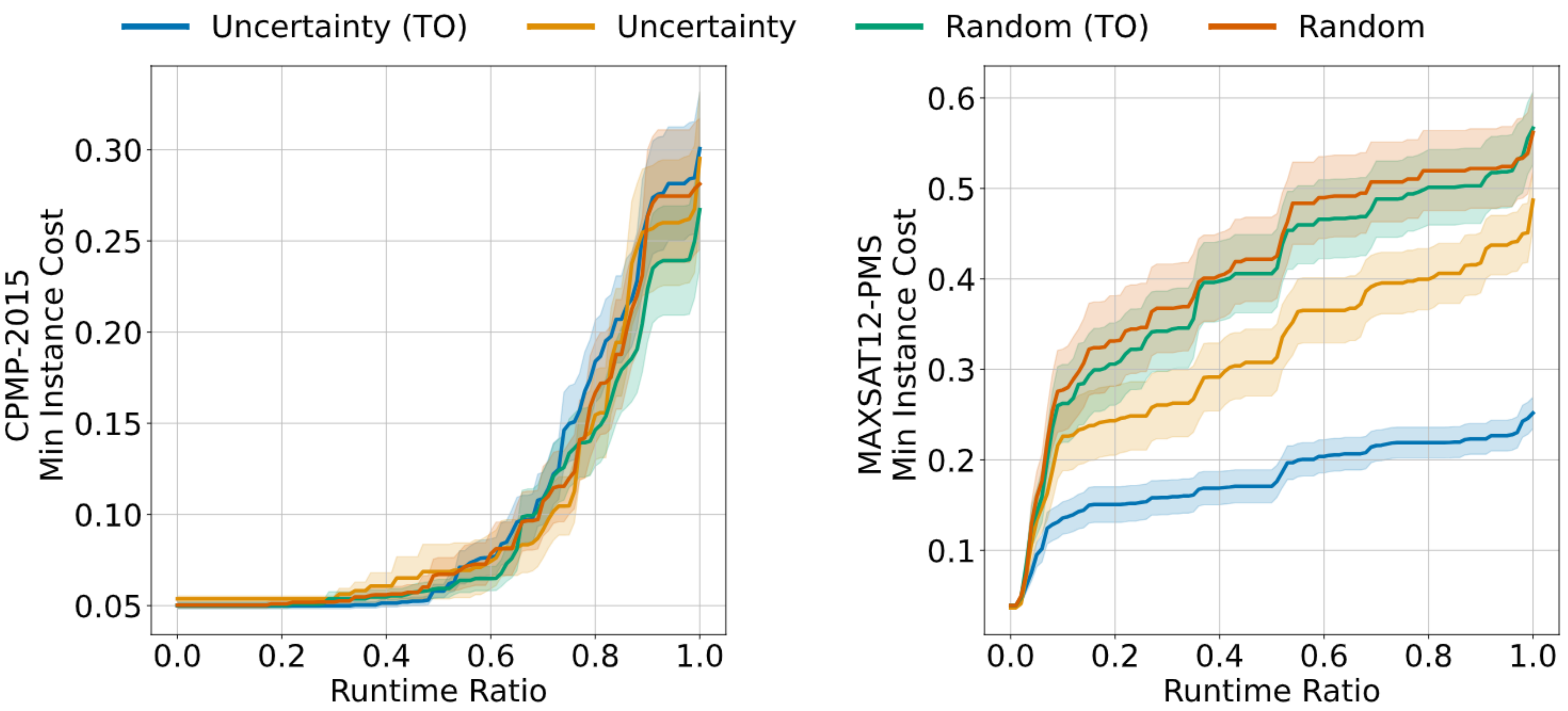
Experimental Results – Timeout Predictor

- Slightly improves performance on some datasets.
- In some cases, there is no improvement.



Why?

Experimental Results – Timeout Predictor



Problems in Timeout Predictor

Labelling Cost: We need to still pay high labelling cost to label an algorithm as timeout.

Query Table-Timeout Predictor

Model	Instance	Uncertainty	Algorithm1 TO	Algorithm2 TO
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5	NO	NO
('akmaxsat', 'akmaxsat_ls')	10tree505p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree610p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5	NO	YES
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5	YES	NO
('akmaxsat', 'akmaxsat_ls')	10tree525p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5	NO	YES
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5	YES	NO
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5	NO	NO
('akmaxsat', 'akmaxsat_ls')	10tree430p.wcnf	0.5	YES	YES
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5	NO	NO

Query Table-Timeout Predictor

Model	Instance	Uncertainty	Algorithm1	Algorithm2
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5	**0.75**	**0.39**
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5	**0.87**	**410.12**

How can we label instances cost-efficiently in **labelling**?

Motivation

1) Can we find a smaller subset that can give the same predictive performance?

- We can use Active Learning.
- We saw that Active Learning selects uninformative and expensive instances during query.

2) How can we eliminate costly, uninformative instances during query?

- Timeout Predictor.
- We saw that it improves the performance but still expensive due to timeout labelling.

3) How can we label instances cost-efficiently in labelling?

- Dynamic Timeout

Frugal Algorithm Selection-Dynamic Timeout

- Start with a small time limit.
- Run algorithms within this time during labelling. If the algorithm doesn't finish, pause it.
- If prediction performance doesn't improve, gradually extend the time limit.

Query Table-Dynamic Timeout

Model	Instance	Uncertainty	Algorithm1 TO	Algorithm2 TO
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5	NO	NO
('akmaxsat', 'akmaxsat_ls')	10tree505p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree610p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5	NO	YES
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5	YES	NO
('akmaxsat', 'akmaxsat_ls')	10tree525p.wcnf	0.5	YES	YES
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5	NO	YES
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5	YES	NO
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5	NO	NO
('akmaxsat', 'akmaxsat_ls')	10tree430p.wcnf	0.5	YES	YES
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5	NO	NO

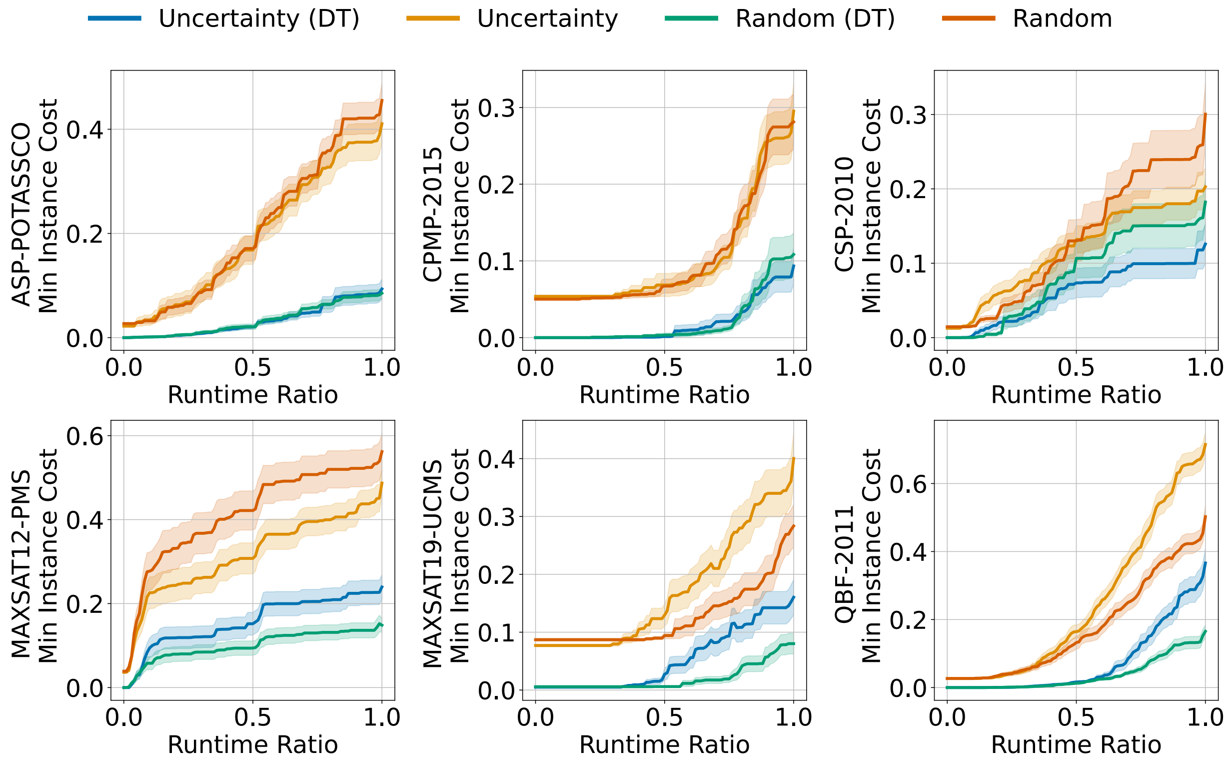
Query Table-Timeout Predictor

Model	Instance	Uncertainty	Algorithm1	Algorithm2
('akmaxsat_ls', 'qmaxsat0.21comp')	cnf3.150.600.372700.cnf.wcnf	0.5	**0.75**	**0.39**
('akmaxsat', 'akmaxsat_ls')	10tree605p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree530p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree520p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree515p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'akmaxsat_ls')	10tree510p.wcnf	0.5	**3600.0**	**3600.0**
('akmaxsat', 'qmaxsat0.21comp')	cnf3.150.450.284126.cnf.wcnf	0.5	**0.87**	**410.12**

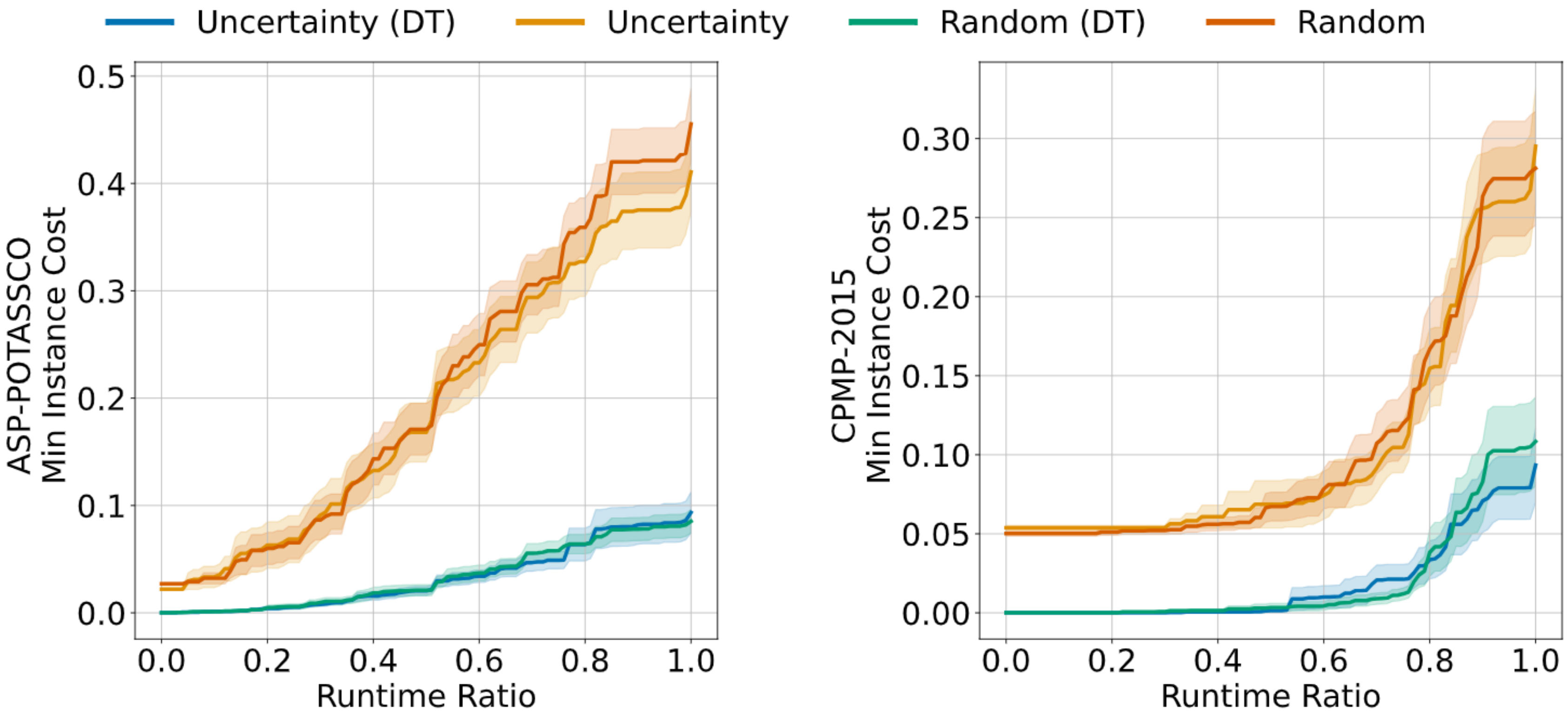
How can we label instances cost-efficiently in **labelling**?

Experimental Results – Dynamic Timeout

- DT shows consistent performance improvements across various datasets.



Experimental Results – Dynamic Timeout

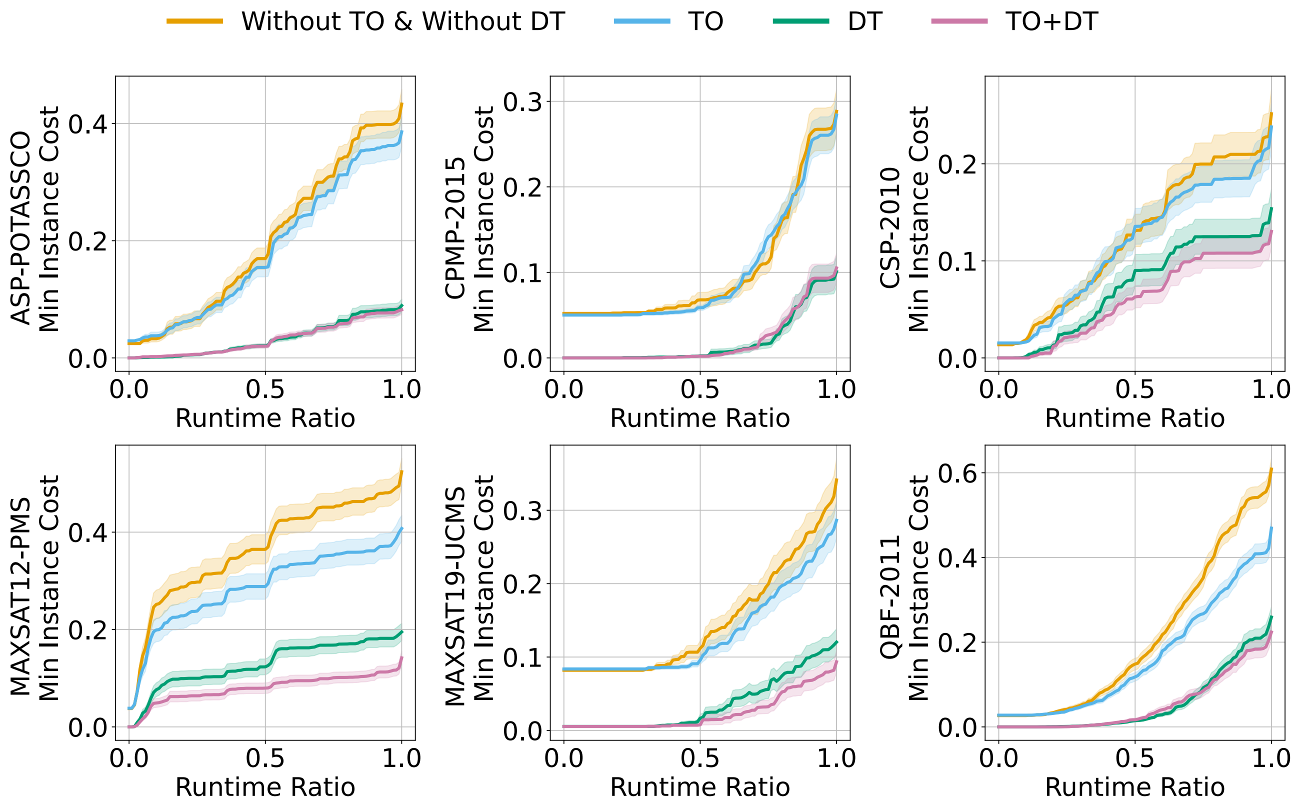


Experimental Setup - Configurations

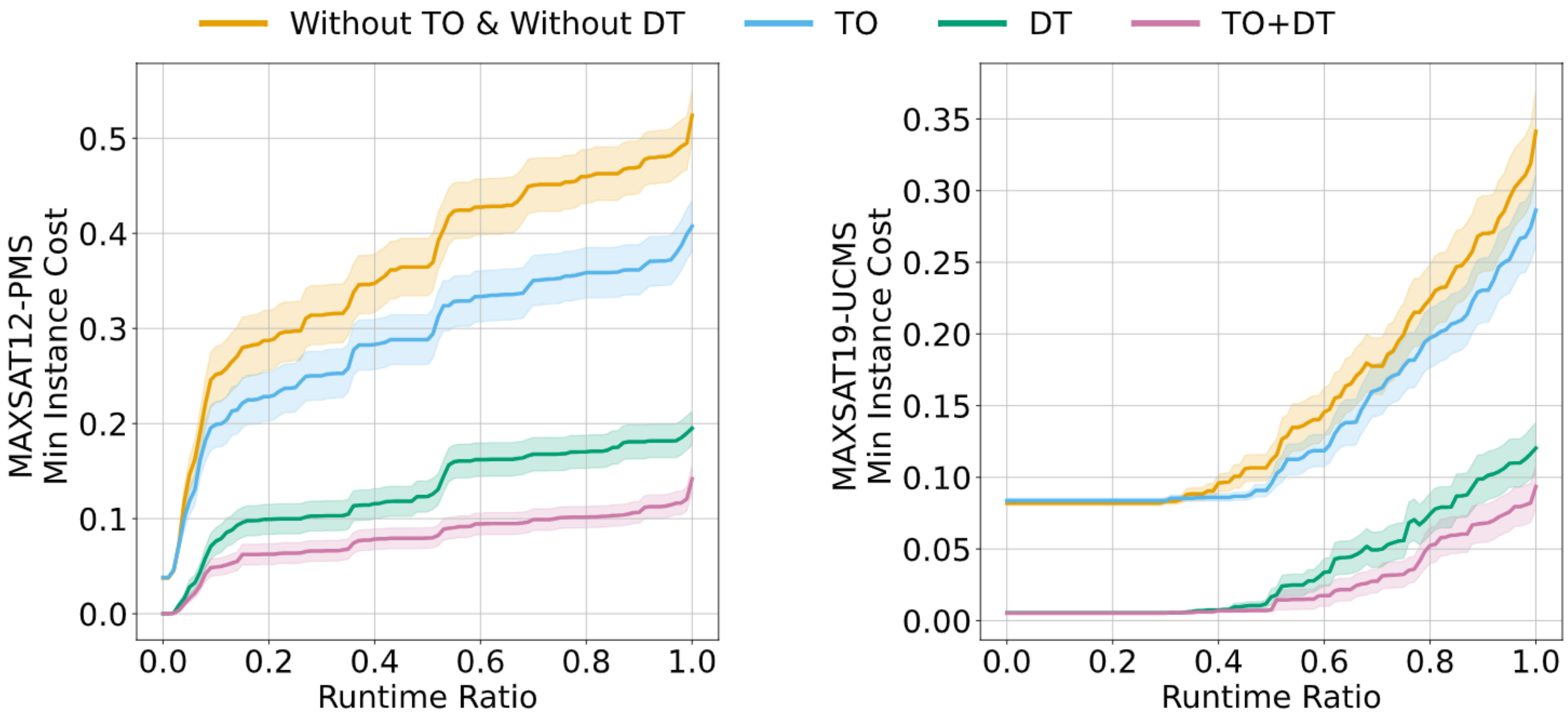
- Vanilla (No TO & No DT)
- Only Timeout Predictor (TO)
- Only Dynamic Timeout (DT)
- Dynamic Timeout with Timeout Predictor (TO+DT)

Experimental Results – All Configurations

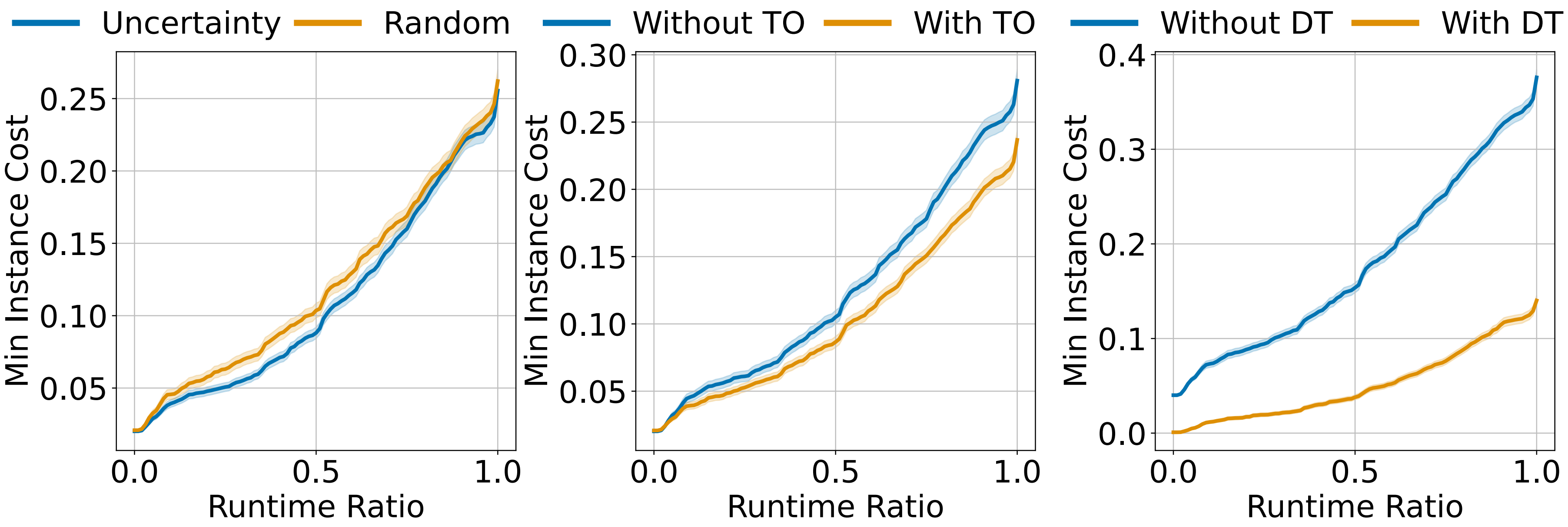
- TO+DT and dynamic timeout alone significantly outperform other configurations.



Experimental Results – All Configurations



Experimental Results – Overall Comparison



- No significant difference between random and uncertainty-based selection methods.
- Timeout Predictor (TO) slightly improves runtime ratio
- Dynamic Timeout (DT) leads to a significant improvement.

Conclusion and Future Work

- Algorithm selection is costly
- Active Learning (AL) is not cost-efficient for Algorithm Selection (AS).
- We propose timeout predictor (in query step) and dynamic timeout (in labelling step) based on AL.
- DT+TO configuration outperforms other configurations provides up to 90% cost-efficiency.

Future Work:

Enhancement Techniques: Pre-solving schedule and cost-sensitive pairwise classification.

Hyper-Parameter Tuning: Explore effects of hyper-parameter tuning.

Evaluation Scope: Expand evaluation to more problem areas.

Thank You!

https://github.com/stacs-cp/CP2024-Frugal