## A Frugal Algorithm Selection Mechanism

**Algorithm 1** Voting Mechanism for Algorithm Selection

```
Require: Set of algorithm predictors $\{P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}\}$, set of timeout predictors
    $\{P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}\}$, validation set $V$
Ensure: Runtimes of the selected algorithms on selected set
 1: Initialize a list to store runtimes: runtimes $\leftarrow$ []
 2: for each instance $x$ in selected set $S$ do
 3:     Initialize a dictionary to count votes for each algorithm: vote_count $\leftarrow$ {}
 4:     Initialize a list to store selected algorithm predictors for voting: voting_predictors $\leftarrow$ []
 5:     if Timeout predictor is used then
 6:         for each pairwise predictor $P_{A_i,A_j}$ in algorithm predictors do
 7:             if $P_{A_i,A_iTO}$ and $P_{A_j,A_jTO}$ predict no timeout for instance $x$ then
 8:                 Add $P_{A_i,A_j}$ to voting_predictors
 9:             end if
10:         end for
11:     else
12:         voting_predictors $\leftarrow$ all algorithm predictors
13:     end if
14:     for each predictor $P_{A_i,A_j}$ in voting_predictors do
15:         Predict the better algorithm $A_{ij}$ for instance $x$ using $P_{A_i,A_j}$
16:         if $A_{ij}$ is not in vote_count then
17:             Initialize vote_count[$A_{ij}$] $\leftarrow$ 0
18:         end if
19:         Increment vote_count[$A_{ij}$] by 1
20:     end for
21:     Find the algorithm $y$ with the maximum votes in vote_count
22:     Run the selected algorithm $y$ on instance $x$
23:     Append runtime to runtimes
24: end for
25: return Sum of runtimes
```

**Algorithm 2** Frugal Algorithm Selection

```
Require: Dynamic timeout flag, Timeout predictor flag, maximum timeout (3600 seconds), query
    size (1%), initial dynamic timeout, timeout increase rate
Ensure: Trained algorithm and timeout predictors, runtimes on validation set
 1: Initialize the unlabeled data pool
 2: if Dynamic timeout is used then
 3:     Select $N$ pairs of algorithm-instance combinations from the unlabeled set of instances
    within *initial dynamic timeout*
 4: else
 5:     Select $N$ pairs of algorithm-instance combinations from the unlabeled set of instances
    within *maximum timeout*
 6: end if
 7: Train algorithm predictors $P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}$ using labeled data
 8: if Timeout predictor is used then
 9:     Train timeout predictors $P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}$ using labeled data
10: end if
11: while There is data left for query do
12:     Create a table of uncertainty information for algorithm-instance combinations
13:     Sort the table by uncertainty in *descending* order
14:     Select the top *query size* instances
15:     if Timeout predictor is used then
16:         Eliminate algorithm-instance combinations that are likely to timeout using timeout
    predictors
17:     end if
18:     if Dynamic timeout is used then
19:         Label algorithm-instance combinations within the *initial dynamic timeout*
20:     else
21:         Label algorithm-instance combinations within the *maximum timeout*
22:     end if
23:     Delete algorithm-instance combinations from the pool if they do not timeout
24:     Train algorithm predictors $P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}$ using labeled data
25:     if Timeout predictor is used then
26:         Train timeout predictors $P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}$ using labeled data
27:     end if
28:     if Timeout predictor is used then
29:         validation runtime ← Call **Voting Mechanism** with validation set $V$, algorithm predic-
    tors, and timeout predictors
30:         test runtime ← Call **Voting Mechanism** with test set $T$, algorithm predictors, and
    timeout predictors
31:     else
32:         validation runtime ← Call **Voting Mechanism** with validation set $V$ and algorithm
    predictors
33:         test runtime ← Call **Voting Mechanism** with test set $T$ and algorithm predictors
34:     end if
35:     if Dynamic timeout is used then
36:         if validation runtime >= previous validation runtime then
37:             *dynamic timeout += timeout increase rate*
38:         end if
39:     end if
40: end while
```

## B Analysis of Uncertainty Measurement Behaviours in Active Learning for Binary Classification

There are three main approaches for uncertainty sampling in active learning. However, in a binary classification setting (which is what we use) these approaches perform identically to each other. We explain the different approaches here. Figure 3 shows the behaviour of these uncertainty sampling methods graphically.

We implement 'Least Confidence' in our code.

- *Least Confidence*: for a given input $x$ and an output label $\hat{y}$, we can measure the posterior probability $P(\hat{y}|x;\theta)$ of observing $\hat{y}$ given $x$ via the current model (parameterised by $\theta$). The Least Confidence method selects data points $x^*$ with the smallest maximum posterior probability across all labels:
$$x^* = \operatorname*{argmin}_x \max_{\hat{y}} P(\hat{y}|x;\theta) \tag{1}$$

- *Margin-based*: this approach takes the two highest posterior probability values for each input data point $x$ and calculates their difference. The smaller the difference, the less certain the model is about its prediction and vice versa. More formally, let $\hat{y}_1$ and $\hat{y}_2$ the output labels with the highest and second-highest posterior probabilities for a given input $x$, respectively, the queried points $x^*$ are chosen as:
$$x^* = \operatorname*{argmin}_x P(\hat{y}_1|x;\theta) - P(\hat{y}_2|x;\theta) \tag{2}$$

- *Entropy-based*: this approach takes into account the posterior probability values across *all* output classes. The idea is to select the data points $x^*$ where there is a high entropy among the predicted output labels:
$$x^* = \operatorname*{argmax}_x - \sum_i P(\hat{y}|x;\theta) \log P(\hat{y}|x;\theta) \tag{3}$$

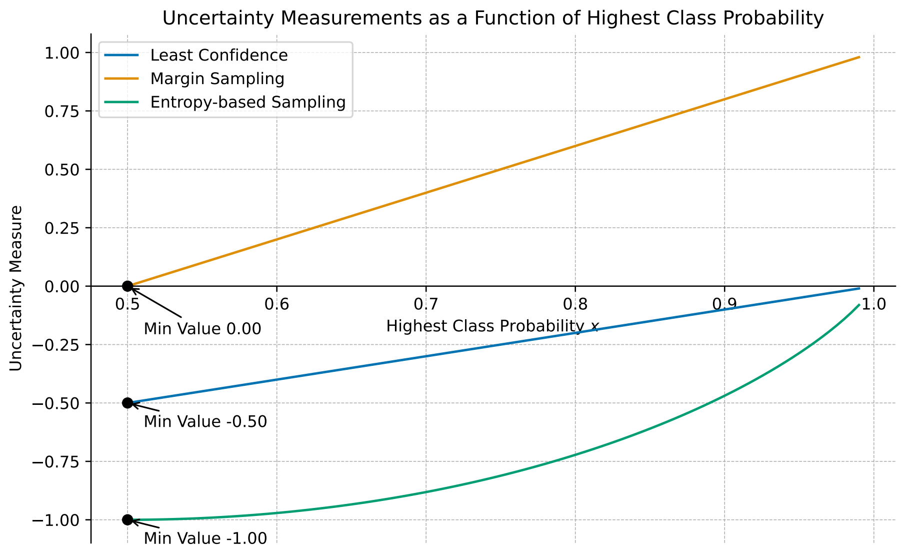

Figure 3: Uncertainty measurements as a function of the highest class probability. The red curve represents the Least Confidence uncertainty (LC) calculated as $LC = x - 1$, the green curve denotes Margin Sampling (MS) using the formula $MS = x - (1 - x)$, and the blue curve illustrates the Entropy-based method ($H(x) = -[x \log_2(x) + (1 - x) \log_2(1 - x)]$). Critical minimum values for each method are marked with black circles and annotated to emphasise the points where the uncertainty function is minimised.

## C Performance of 8 Individual Configurations and Best Configuration

Figure 4 illustrates a side-by-side comparison of the following eight active learning strategies in binary classification without aggregation across configurations:

- Uncertainty Sampling (Baseline)
- Uncertainty Sampling with Timeout Predictor (TO)
- Uncertainty Sampling with Dynamic Timeout (DT)
- Uncertainty Sampling with Timeout Predictor and Dynamic Timeout (TO+DT)
- Random Sampling (Baseline)
- Random Sampling with Timeout Predictor (TO)
- Random Sampling with Dynamic Timeout (DT)
- Random Sampling with Timeout Predictor and Dynamic Timeout (TO+DT)

Figure 5 shows the performance comparison of the best configuration in the experiment, disaggregated by approach.

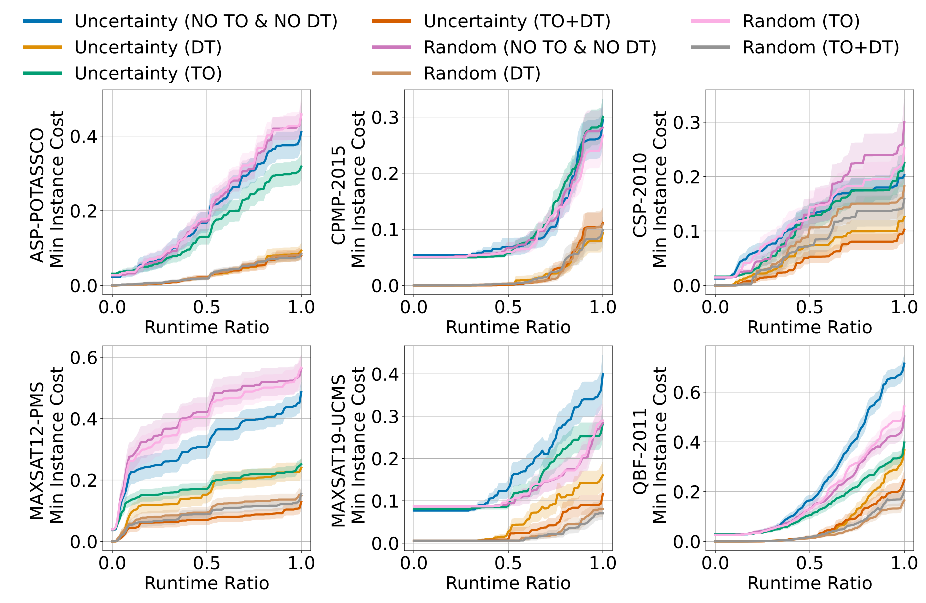

Figure 4: Comparison of performance across eight configurations as described in the paper. Each configuration was normalised according to the passive learning prediction performance ratio.

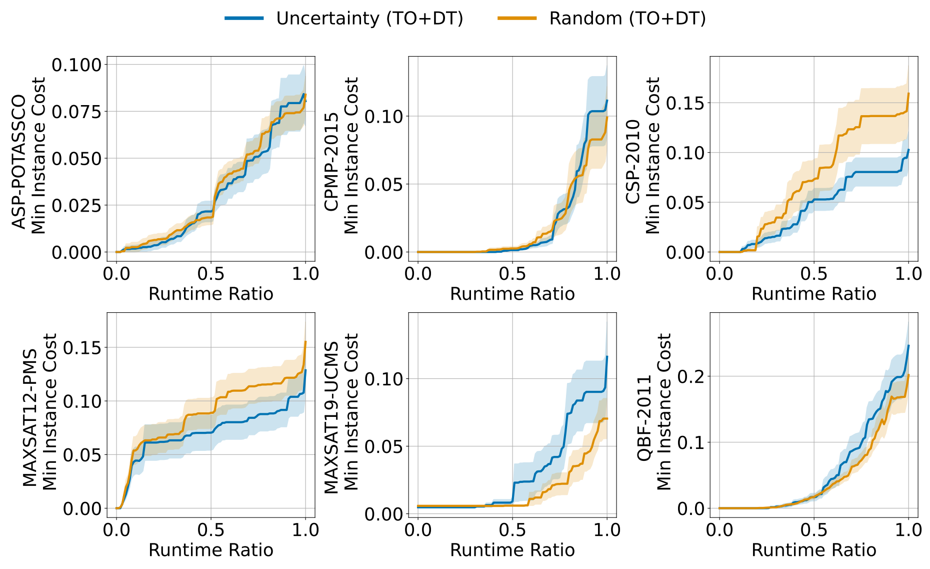

Figure 5: Comparison of TO+DT options presented in Figure 2, disaggregated by instance selection approach. No single option (uncertainty-based vs random instance selection) emerged as the clear winner.

## D Experimental Setup

This study used a Random Forest classifier configured with 100 estimators and the Gini impurity measure to determine the best splits. Each tree is limited to using up to the square root of the number of features, and the depth of the decision trees is practically unlimited (with a maximum depth set to $2^{31}$). Nodes require at least two samples before splitting, and bootstrapping is enabled for sampling data when building each decision tree. These settings were determined through experimentation in the passive learning setup and were consistently used throughout the study.

We also addressed missing data by removing features where more than 20% of the instances had missing values and applied a median imputer to fill the remaining gaps.

We employed a cross-validation approach with 10 splits to validate the robustness of our study. To ensure reproducibility, we used 5 distinct seeds (7, 42, 99, 123, 12345) across our experiments, ensuring consistent generalization across multiple runs.

To determine when to increase the timeout in configurations where dynamic timeout is used, 10% of the training set was allocated as the validation set. Throughout the experiments, timeout values were scaled by a factor of 10, following the PAR10 measure.

The performance of all eight configurations of the frugal algorithm selection methods was evaluated using six datasets from ASLib, selected for their diverse characteristics. These datasets span various problem-solving domains, including one from Answer Set Programming (ASP), two from Constraint Programming (CP), two from propositional satisfiability (SAT), and one from Quantified Boolean Formula (QBF) solving. The datasets differ significantly in complexity and size, featuring between 2 to 11 algorithm options, 22 to 138 features, and 527 to 2024 instances. We chose these six specific datasets to evaluate different domains and problem characteristics comprehensively.

**Additional Parameters and Configurations**:

**Timeout Predictor Usage**: This parameter determines whether the timeout predictor is used on the system.

**Timeout Limit**: Sets the initial time for the dynamic timeout. We used an initial timeout of 100 seconds when employing dynamic timeout, and a fixed timeout of 3600 seconds when not using dynamic timeout.

**Timeout Increase Rate**: Adjusts the dynamic timeout when there is no improvement in prediction performance on the validation set. We set this rate to increase by 100 seconds when no improvement was observed.

**Initial Train Size**: Determines the size of the initial training set for uncertainty selection. The initial training set was created by randomly selecting 20 data points from the overall training set.

**Query Size**: Refers to the percentage of the dataset queried in each iteration. We set this to 1%, meaning 1% of the total pool of candidates was queried in each iteration of our experiments.

For active learning, we utilized the modAL framework [6], which facilitated the implementation of uncertainty sampling and other active learning strategies in our experiments.

This work used the Cirrus UK National Tier-2 HPC Service at EPCC (http://www.cirrus.ac.uk) funded by the University of Edinburgh and EPSRC (EP/P020267/1). The total CPU time for both the training and evaluation of the passive learning, uncertainty selection model, and random selection, was 1291.48 hours.

## E Description Table of Selected Datasets

Table 1 shows key information about the datasets used in this study. It includes the time it took for the algorithms to run, the Virtual Best Solver(VBS) representing the best algorithm for each problem, and the Single Best Solver (SBS) as the best overall algorithm. While VBS is the hypothetical best, SBS serves as a benchmark for comparison against other algorithms.

| Dataset | Instances | Algorithms | Features | Total Time | VBS | SBS |
|---|---|---|---|---|---|---|
| ASP-POTASSCO | 1294 | 11 | 138 | 2,085h | 8h | 112h |
| CPMP-2015 | 527 | 4 | 22 | 682h | 33h | 134h |
| CSP-2010 | 2024 | 2 | 86 | 435h | 49h | 82h |
| MAXSAT12-PMS | 876 | 6 | 37 | 1,472h | 8h | 85h |
| MAXSAT19-UCMS | 572 | 7 | 54 | 545h | 20h | 52h |
| QBF-2011 | 1368 | 5 | 46 | 352h | 28h | 300h |

Table 1: Descriptive statistics of selected datasets. Times rounded to the nearest whole number.

## F Timeout (TO) Configuration Impact on Passive Learning

Figure 6 illustrates that the TO configuration in passive learning does not lead to a substantial improvement in performance on the test set. This observation was made in the context of selecting a baseline for comparing the performance of random and uncertainty selection.

Given that the TO configuration does not significantly enhance performance, and considering that it adds an extra layer of complexity to the model, the decision was made to use a simpler configuration for the passive learning model. Therefore, the passive learning model without a timeout predictor was chosen as the baseline for the comparisons.

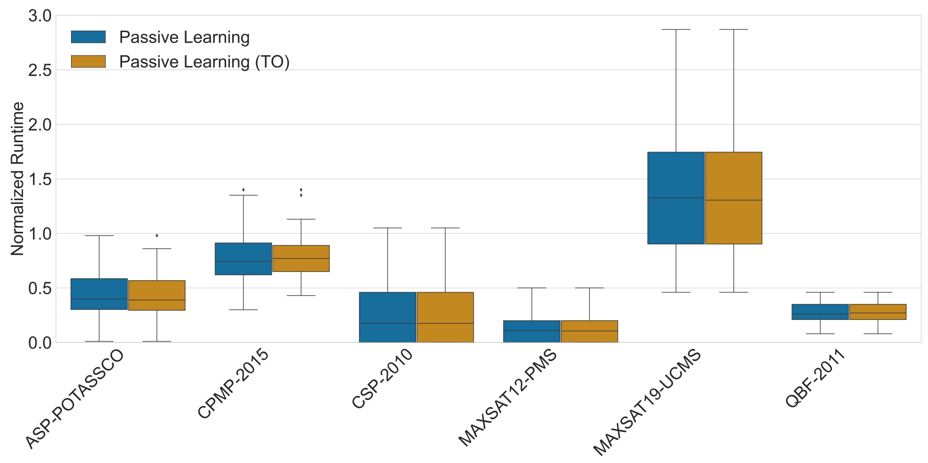

Figure 6: Comparison of Timeout (TO) Configuration Impact on Passive Learning: The graph illustrates that implementing the TO configuration in passive learning on the test set does not significantly enhance performance, yet importantly, it does not compromise prediction accuracy either.