On the Effect of Training Data Selection in Automated Algorithm Selection

Erdem Kuş
School of Computer Science, University of St Andrews, Scotland, UK

Özgür Akgün
School of Computer Science, University of St Andrews, Scotland, UK

Nguyen Dang
School of Computer Science, University of St Andrews, Scotland, UK

Lars Kotthoff
School of Computer Science, University of St Andrews, Scotland, UK

Ian Miguel
School of Computer Science, University of St Andrews, Scotland, UK

Abstract

Algorithms for solving combinatorial optimisation problems often exhibit complementary strengths, motivating automated algorithm selection by training ML models that predict the best algorithm for a given instance. However, collecting training data for such models is computationally expensive, as it requires running all provided algorithms on all training instances. Recent work on frugal algorithm selection shows that the training data collection cost can be reduced substantially through active learning, but the interaction between model choices and data efficiency remains poorly understood. In this work, we empirically investigate how different learning formulations behave under limited training data using the ASLib benchmark. Our results reveal that multiclass classification (MC), despite weak performance when trained on full training data, improves dramatically with active learning. Remarkably, active MC matches strong passive learners while using only a fraction of the training data. This highlights an unexpected efficiency gain: algorithm selectors that underperform with full training data become highly effective when training data is selected actively.

2012 ACM Subject Classification Theory of computation → Active learning; Computing methodologies → Supervised learning by classification; Theory of computation → Constraint and logic programming

Keywords and phrases Algorithm selection, Active learning, Learning strategies

Digital Object Identifier 10.4230/LIPIcs.CP.2026.38

Supplementary Material *Software (Source Code)*: `https://doi.org/10.5281/zenodo.19978961`

Acknowledgements We thank the University of St Andrews for computational support via access to its high-performance computing cluster Hypatia.

1 Introduction

Solving combinatorial problems in practice requires considerable computational resources, and different algorithms have been proposed to deliver efficient solutions. Such algorithms often exhibit complementary performance across problem instances, with no single algorithm dominating uniformly across all cases. It has been shown that selecting the most appropriate algorithm for a given instance can yield substantial efficiency gains (e.g. [20]). This is known as the *Algorithm Selection (AS)* problem [18]. Machine Learning is a well-established approach to solving Algorithm Selection in practice. By learning from features of given

training instances and algorithm performance data on those instances, ML models are able to predict the most effective algorithms for unseen problem instances, leading to improvements in computational efficiency or solution quality [13, 12].

There are multiple learning formulations for the AS problem. A relatively simple approach is to build a multiclass classifier that directly predicts the best algorithm from a problem instance's features. An alternative approach is to build a regression model for each algorithm to predict its performance from instance features and select the algorithm with the best predicted performance. There are also more sophisticated approaches, such as the cost-sensitive pairwise classification where for every pair of algorithms, a (weighted) binary classifier is built to predict the winning algorithm [15], and the algorithm with the highest number of winning votes is selected. Common to all such approaches is the high upfront *labelling cost*: constructing a suitable training dataset requires that all candidate algorithms be executed on each training instance to generate the performance data necessary for ground-truth labelling, a procedure that is often very expensive.

Active learning (AL) [4] has been widely studied by the ML community as an effective approach to reducing labelling cost. Rather than labelling all available training instances, informative data points are selected iteratively. This often results in substantially smaller labelling costs while maintaining high predictive performance. Standard active learning methods, however, implicitly assume that all labels can be obtained at a uniform cost, seeking to minimise labelling cost solely by reducing the number of labelled samples required. This assumption does not hold in the algorithm selection contexts, where labelling cost is inherently non-uniform: some algorithms may solve a given instance within seconds, while others time out after hours of execution. To address this issue, *frugal algorithm selection* [14] was introduced, incorporating two *frugality* mechanisms into active learning for AS contexts: (i) *timeout prediction ML models* to predict whether an algorithm fails to solve a given problem instance within a given time limit (without having to actually run the algorithm), and (ii) *dynamic time limit*, where the time limit is gradually increased during the training data collection process. The effectiveness of these frugality mechanisms in reducing labelling cost was demonstrated across several AS scenarios within the context of pairwise classification-based AS.

However, different AS learning formulations impose different labelling granularities. For example, the labelling of each data point for a multiclass classification-based AS approach involves running all algorithms on a given problem instance, while in pairwise classification-based AS approaches, the labelling of each data point corresponds to running only the two algorithms in the pair on the given problem instance. The implications of labelling granularity and its interactions with different learning formulations in frugal AS settings remain poorly understood. To address this gap, in this work, we conduct a systematic investigation of four commonly-used learning formulations in AS within the frugal AS framework.

The main contributions of this work are as follows.

- We present a systematic comparison of four common learning formulations for frugal algorithm selection; Pairwise Classification (PC), Multiclass Classification (MC), Pairwise Regression (PR), and Single Regression (SR), across 16 scenarios from ASLib, the standard benchmarking library for algorithm selection [2].
- We show that MC, despite poor algorithm selection performance with passive learning, becomes the most frugal formulation under active learning, matching strong passive learners while using only a fraction of the labelling cost.
- We analyse the effect of labelling granularity in frugal algorithm selection and show that coarse-grained labelling improves frugality for other formulations, although MC remains the strongest overall.

2 Background

2.1 Algorithm Selection Problem

The AS problem was first formally introduced by Rice [18], who defined it as the problem of selecting the best algorithm from a set of candidates for a given problem instance. The notion of algorithm portfolios was subsequently introduced by [10] and was also studied by [8], whereby a set of complementary algorithms is maintained, and a selector is learned to predict the best algorithm for any given instance. Given an algorithm portfolio $\mathcal{A} = \{a_1, \ldots, a_n\}$ of n algorithms and a problem instance distribution $\mathcal{D}_\mathcal{I}$, where each instance $i \sim \mathcal{D}_\mathcal{I}$ is represented by a d-dimensional feature vector $v(i) \in \mathbb{R}^d$, the goal is to learn an algorithm selector that can predict the best algorithm in $\mathcal{A}$ for any given problem instance based on its features. Formally, following [2], assuming we are given a cost function c where $c(i, a)$ represents the cost of solving an instance i using the algorithm a, the objective is to find a selector $s^* : \mathbb{R}^d \to \mathcal{A}$ that minimises the expected cost over the distribution of instances $\mathcal{D}_\mathcal{I}$:

$$s^* = \underset{s}{\operatorname{argmin}} \quad \mathbb{E}_{i \sim \mathcal{D}_\mathcal{I}} \left[c(i, s(v(i))) \right] \tag{1}$$

2.2 Frugal Algorithm Selection

Over the last two decades, several ML-based approaches have been proposed for solving the AS problem [13, 12]. The training of an ML-based AS selector involves collecting the performance data of all algorithms in the portfolio $\mathcal{A}$ on a given set of training problem instances $\mathcal{I}$. The cost of this process, namely *labelling cost* and formally defined as $\sum_{i \in \mathcal{I}, a \in \mathcal{A}} c(i, a)$, is often computationally expensive (as demonstrated in Table 2). This motivates the study of *Frugal Algorithm Selection (Frugal AS)* [14], where active learning was used to actively select a subset of the training data for the labelling process. The aim was to achieve competitive AS performance with the "passive leaner", the AS selector trained on the full training dataset, while using only a fraction of the training dataset.

In this sense, *frugality* refers to the ability of an AL strategy to achieve comparable AS performance to passive learning whilst incurring a substantially lower labelling cost; a more frugal strategy is one that reaches comparable passive learning AS performance by spending less of the available labelling budget. Frugal AS aims to maximise this frugality by reducing the labelling cost incurred during the AL process without sacrificing AS performance.

Previous work has empirically shown the limitation of naively applying standard AL in Frugal AS contexts [14]. Concretely, standard AL methods implicitly assume that all labels can be obtained at a *uniform* cost, an assumption that does not hold in AS, where some algorithms may solve a given instance within seconds whilst others time out after hours of execution. To address this, two *frugality mechanisms* were introduced [14]:

- **Dynamic Time Limit (DTL):** The DTL mechanism adaptively reduces the time allocated to each algorithm run during the AL process, thereby avoiding unnecessary use of the labelling budget on algorithm runs that are unlikely to complete within the time limit. Following the setup of Kuş et al. [14], the initial time limit is set to 100 seconds and is increased by 100 seconds for each iteration in which no improvement in AS performance is observed on the validation set.
- **Timeout Prediction (TO):** The TO mechanism employs a set of surrogate models to predict whether a given algorithm run will time out on a selected instance, and eliminates predicted timeout runs before they are executed, thereby avoiding the unnecessary consumption of the labelling budget on runs that are known to be uninformative.

Together, these mechanisms substantially reduce the labelling cost incurred during the AL process while maintaining competitive AS performance, allowing for an overall reduction of 40% of labelling cost in most cases (this reduction even rises to 90% in some cases).

2.3 The Algorithm Selection Library

The Algorithm Selection Library (ASLib) [2] is a standardised benchmark suite for the empirical evaluation of AS approaches. ASLib provides a collection of scenarios, each consisting of a set of problem instances, a portfolio of algorithms, pre-computed algorithm performance data, and instance features extracted from the problem instances. The scenarios cover a diverse range of problem classes, including Answer Set Programming (ASP), Boolean Satisfiability (SAT), Quantified Boolean Formula (QBF), Container Pre-Marshalling Problem (CPMP), and Maximum Satisfiability (MaxSAT), among others. ASLib has been widely adopted as the standard benchmark for evaluating AS approaches [16, 13].

We measure AS performance using *PAR10*, a penalised runtime metric that assigns the runtime to runs that succeed and a penalty of ten times the time limit to any algorithm run that fails to solve an instance within the time limit. To account for the different scales of PAR10 values among different AS scenarios, AS performance is often normalised relative to two baselines. The *Virtual Best Solver* (VBS) represents an oracle that always selects the best-performing algorithm for each instance, serving as a lower bound on the achievable normalised runtime and corresponding to a normalised score of 0. The *Single Best Solver* (SBS) represents the single algorithm with the best average performance across all instances in the scenario, serving as a simple (non-ML) baseline and corresponding to a normalised score of 1. A normalised score below 1, therefore, indicates that the AS system outperforms the SBS, with lower values indicating better performance.

3 Methodology

The original frugal AS framework was developed exclusively within the context of a pairwise classification-based AS approach. However, there are other ML-based approaches in the AS literature [13, 12]. Moreover, the definition of a training data point can differ between learning formulations, which in turn impacts the *query granularity*, i.e., the granularity of how data is sampled for labelling during the AL process (more details in Section 3.2). The interactions between AS learning formulations, query granularity, and frugality mechanism in the context of frugal AS remain an open question. In this work, we addresses this gap by extending the frugal AS framework to four commonly-used AS learning formulations and conducting a systematic investigation of the interaction between these formulations and the components of the AL process (i.e., query granularity and frugality mechanisms). We first describe the AS learning formulations used in our study in Section 3.1, following with an explanation of query granularity when applying AL to frugal AS contexts (Section 3.2).

3.1 Learning Formulations for Algorithm Selection

Several learning formulations have been proposed for AS. We consider four commonly-used formulations from the AS literature [15]:

- **Pairwise Classification (PC):** PC trains $\binom{n}{2}$ binary classifiers, one for each pair of algorithms (a_j, a_k) with $j < k$, each predicting which of the two algorithms performs better on a given instance. Formally, the training label for instance i and the algorithm pair (a_j, a_k) is defined as:

$$y_{jk}(v(i)) = \begin{cases} a_j & \text{if } c(i,a_j) < c(i,a_k) \\ a_k & \text{if } c(i,a_j) > c(i,a_k) \end{cases} \tag{2}$$

If the two algorithms perform equally well on the given instance, the label is chosen randomly among the two.
At test time, each classifier votes for its preferred algorithm, and the algorithm with the highest number of votes is selected:

$$s(v(i)) = \arg\max_{a_j \in \mathcal{A}} \sum_{a_k \in \mathcal{A} \setminus \{a_j\}} \mathbf{1}\left[\hat{y}_{jk}(v(i)) = a_j\right] \tag{3}$$

- **Pairwise Regression (PR):** PR follows the same pairwise structure as PC, but trains a regression model to predict the performance difference between the two algorithms in the pair instead:

$$y_{jk}(v(i)) = c(i,a_j) - c(i,a_k) \tag{4}$$

where $y_{jk}(v(i)) > 0$ indicates that a_j is worse than a_k on instance i. This produces $\binom{n}{2}$ regression models in total. At test time, the algorithm with the lowest aggregated predicted difference is selected:

$$s(v(i)) = \arg\min_{a_j \in \mathcal{A}} \sum_{a_k \in \mathcal{A} \setminus \{a_j\}} \hat{y}_{jk}(v(i)) \tag{5}$$

- **Single Regression (SR):** SR trains a separate regression model for each algorithm $a \in \mathcal{A}$ to directly predict its cost on a given instance:

$$y_a(v(i)) = c(i,a) \tag{6}$$

This approach builds n regression models in total. At test time, the algorithm with the lowest predicted cost is selected:

$$s(v(i)) = \arg\min_{a \in \mathcal{A}} \hat{y}_a(v(i)) \tag{7}$$

- **Multiclass Classification (MC):** MC trains a single multiclass classifier to directly predict the best-performing algorithm for a given instance. The training label for a problem instance i is defined as:

$$y((v(i)) = \arg\min_{a \in \mathcal{A}} c(i,a) \tag{8}$$

At test time, the classifier outputs the predicted best algorithm:

$$s(v(i)) = \hat{y}(v(i)) \tag{9}$$

3.2 Query Granularity in Frugal Algorithm Selection

As summarised in Table 1, AS learning formulations differs not only in the number of ML models constructed during the training process, but also on how a training data point is defined. Given a problem instance, SR requires collecting performance of only one algorithm per training data point. PC and PR share the same structure and require collecting performance of two algorithms in the pair per training data point. Finally, each data point

Table 1 For each AS learning formulation: the number of ML models constructed and the number of algorithm runs required per training data point, where n is the number of algorithms in the portfolio.

Formulation	#ML Models	#Algorithm runs per data point
PC	$\binom{n}{2}$	2
PR	$\binom{n}{2}$	2
SR	n	1
MC	1	n

in MC requires running all algorithms in the portfolio on the given problem instance. These differences have important implications for the data querying process during active learning in the context of Frugal AS.

Concretely, at each iteration of the active learning process, a number of unlabelled training data points are "queried" based on their informativeness. The higher the prediction uncertainty of the current models on a data point, the more informative the data point is (and the more likely the data point is selected for labelling). In Frugal AS contexts, this querying process can be conducted within two levels of granularity. Under *fine-grained* querying, each ML model independently selects its own most informative instance for labelling. This mechanism natively applies to formulations with multiple models (PC, PR, and SR). Under *coarse-grained* querying, all models collectively agree on a single instance, and all algorithms in the portfolio are run on the selected instance. This is the native querying strategy for MC, as it trains a single classifier over all algorithms. For PC, PR, and SR, which natively operate under fine-grained querying, coarse-grained querying can still applied by aggregating the uncertainty measurement across all models and selecting the instance with the highest mean uncertainty as the single shared query. In this study, we investigate all possible combinations of query granularities and learning formulations and their effectiveness in frugal AS contexts.

4 Experimental setup

We conduct experiments on 16 scenarios from ASLib [2], as described in Section 2. The scenarios cover a diverse range of combinatorial optimisation domains, including Answer Set Programming (ASP), Bayesian Network Structure Learning (BNSL), Constraint Programming (CP), Boolean Satisfiability (SAT), and Quantified Boolean Formula (QBF). As shown in Table 2, the scenarios exhibit substantial heterogeneity: portfolio sizes range from 2 to 31 algorithms, instance features range from 22 to 155, and problem instances range from 296 to 4642.

Model and evaluation. Random forests are used as the predictive model across all four formulations, as they have been shown to achieve stronger performance on AS benchmarks [2], consistent with the model choice of prior work [14], and naturally provide uncertainty estimates from the ensemble of decision trees without any additional calibration, making them particularly well-suited to active learning. All formulations are evaluated via 10-fold cross-validation. Results are averaged over five random seeds and ten folds to account for variability in training and data partitioning. For classification-based formulations (MC and PC), the random forest is configured with 100 estimators, Gini impurity as the splitting criterion, and the square root of the number of features as the maximum number of features considered at each split, no maximum depth constraint, a minimum of 2 samples required to

Table 2 Summary of ASLib benchmark scenarios used in this study, reporting the total computational cost (in CPU hours) required to run all algorithms on all instances.

Scenario	Instances	Algorithms	Features	Total Labelling Cost
ASP-POTASSCO	1294	11	138	2,085h
BNSL-2016	1179	8	86	3,272h
CPMP-2015	527	4	22	682h
CSP-2010	2024	2	86	435h
CSP-MZN-2013	4642	11	155	26,263h
MAXSAT12-PMS	876	6	37	1,472h
MAXSAT15-PMS-INDU	601	29	37	7,582h
MAXSAT19-UCMS	572	7	54	545h
MAXSAT-WPMS-2016	630	18	37	5,113h
QBF-2011	1368	5	46	2,352h
QBF-2014	1254	14	46	6,510h
QBF-2016	825	24	46	5,832h
SAT11-HAND	296	15	115	1,851h
SAT11-RAND	600	9	115	1,829h
SAT12-HAND	767	31	115	9,069h
SAT12-INDU	1167	31	115	12,272h

split an internal node, and bootstrap sampling is enabled. For regression-based formulations (PR and SR), the same configuration is used with the exception that the splitting criterion is omitted, as random forest regressors minimise mean squared error by default. AS performance is measured using PAR10 and normalised by the SBS-VBS gap as described in Section 2, following [16]:

$$\widehat{\text{PAR10}}_{AS} = \frac{\text{PAR10}_{AS} - \text{PAR10}_{VBS}}{\text{PAR10}_{SBS} - \text{PAR10}_{VBS}}$$

where a normalised score of 0 corresponds to VBS-level performance, and 1 corresponds to SBS-level performance.

Active Learning Setup. Following the frugal AS procedure in previous work [14], the AL loop is initialised with 20 randomly selected instances for labelling and training the initial AS models. We then proceed with alternating between expanding the labelled training data and and re-training the models until the entire training set is labelled. At each iteration, 1% of the remaining unlabelled algorithm runs is selected for labelling. Runs associated with the highest prediction uncertainty are selected first. PC, PR, and SR are evaluated under both fine-grained querying (their native setting) and coarse-grained querying, while MC is evaluated under coarse-grained querying only, as it is the only valid query mode under this setting. Finally, one of the frugality mechanisms studied – Dynamic Time Limit – requires a separate validation set to determine when to increase the time limit during the AL process. We take 10% of the training data and use it as the validation set, following the setup of [14].

Statistical Tests. To assess whether the observed performance differences between strategies are statistically significant, the Friedman test [7] is used. This is a widely-adopted non-parametric test for comparing multiple strategies across multiple scenarios [5]. Upon finding significant differences ($p < 0.05$), the Nemenyi post-hoc test [17] is applied to identify which

pairs of strategies differ significantly in their average ranks. The results are visualised using Critical Difference (CD) diagrams, produced using the `Orange3` toolkit [6]. In these diagrams, strategies are ranked from left to right in order of increasing rank, where a lower rank reflects better frugality, and strategies connected by a horizontal bar are not statistically distinguishable from one another.

All source code, data, and experimental results are available at `https://doi.org/10.5281/zenodo.19978961`.

5 Experimental Results

The evaluation is structured as follows. The passive learning performance of all learning formulations is first assessed to establish a baseline for comparison. The behaviour of each formulation under the active learning setting is then examined with respect to frugality. Finally the effect of query granularity is investigated by comparing coarse-grained and fine-grained querying strategies across all formulations and frugality mechanism combinations.

5.1 Passive Learning Performance

Before investigating the behaviour of each formulation within an AL framework, the passive learning (PL) performance of all four AS formulations is examined to establish a baseline for subsequent comparison. As PL represents the upper bound of labelling cost and information, this baseline serves as a reference point for evaluating the frugality achieved through AL.

Figure 1 presents the PL performance of all four AS learning formulations across 16 ASLib benchmark scenarios. The results reveal a clear distinction between classification-based and regression-based formulations. Regression-based formulations, namely PR and SR, consistently achieve more competitive performance across scenarios, with their median normalised runtime closer to the VBS baseline. PC and MC demonstrate broadly similar PL performance, with PC achieving marginally more competitive results on a number of scenarios. MC, however, exhibits the weakest PL performance overall, with the highest variance across scenarios and a median normalised runtime considerably further from the VBS baseline than the other formulations. These results establish a clear PL baseline and motivate the central investigation of this work: whether a weak passive learner such as MC can achieve competitive AS performance and strong frugality through AL at a substantially reduced labelling cost.

5.2 Learning Formulations Under Active Learning

With the PL baseline established, the AL behaviour of each AS learning formulation is now examined. The key question is which formulation achieves the most favourable trade-off between labelling cost and AS performance as the labelling budget is progressively increased.

Figure 2 presents the trade-off between AS performance and labelling cost for all four formulations under a baseline AL setting across 16 ASLib benchmark scenarios. AS performance, for both PL and AL, is normalised relative to a single shared reference point, defined as the best average PL performance achieved across all formulations, seeds, and cross-validation splits for each scenario, and set to 100%. This is an important distinction: without a shared reference, a formulation with weak PL performance could appear frugal simply by reaching its own lower performance ceiling at a reduced labelling cost, rather than genuinely matching the best performance achievable by any formulation. The results reveal a clear distinction between classification-based and regression-based formulations in terms of

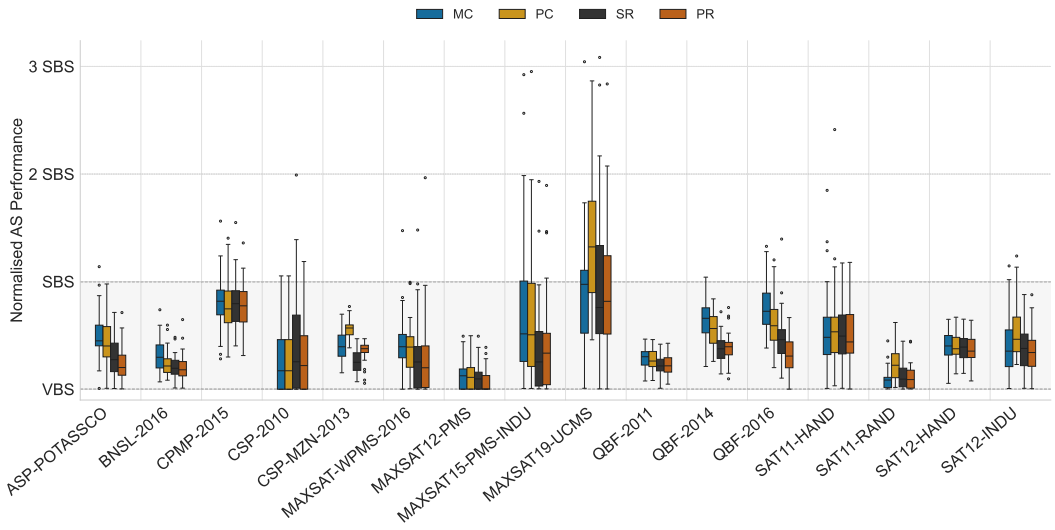

Figure 1 PL performance of all four AS learning formulations, namely MC, PC, SR, and PR, across 16 ASLib benchmark scenarios. Normalised AS performance is expressed relative to the VBS and SBS baselines, with lower values indicating better performance. Regression-based formulations generally achieve more competitive performance, while MC shows the weakest PL performance with the highest variance across scenarios.

frugality. MC consistently achieves comparable performance to the best PL at a substantially lower labelling cost than the other formulations, demonstrating the strongest frugality among all formulations investigated. PC emerges as the second most frugal formulation, achieving competitive AS performance at a moderate labelling cost, and demonstrating comparable frugality to MC on several scenarios.

This is consistent with the known sensitivity of AL to distributional shift, whereby actively queried instances are not drawn i.i.d. from the underlying data distribution [19], and the additional challenges of uncertainty estimation in regression [1]. Unlike classification models, which naturally produce a probability distribution over labels, regression models must first be equipped with a reliable uncertainty estimator by approximating output variance, making them more susceptible to the effects of a reduced labelling budget and causing greater fluctuations in performance as the number of labelled instances decreases.

We also note that AL curves can exceed the PL performance level of individual formulations and converge to the shared 100% reference. Active learning acts as an implicit noise and outlier filter [9] by selecting only informative instances, naturally excluding noisy or outlier instances that would otherwise be included in passive learning and distort model predictions, with proven effect in both regression [3] and classification [9] settings.

Furthermore, PC involves a collection of binary models, each of which queries instances independently, introducing the risk that the query process becomes dominated by timeout runs, a phenomenon that is particularly prevalent in pairwise classification under AL. MC, by contrast, involves only one model and concentrates the labelling effort on a smaller set of instances within each batch. As a result, cheaper instances, which tend to exhibit higher uncertainty early in training, are more likely to be selected, suggesting that MC's strong frugality is driven by a combination of its coarse-grained query strategy and the implicit bias introduced by concentrating the labelling effort on a shared set of instances across all algorithms.

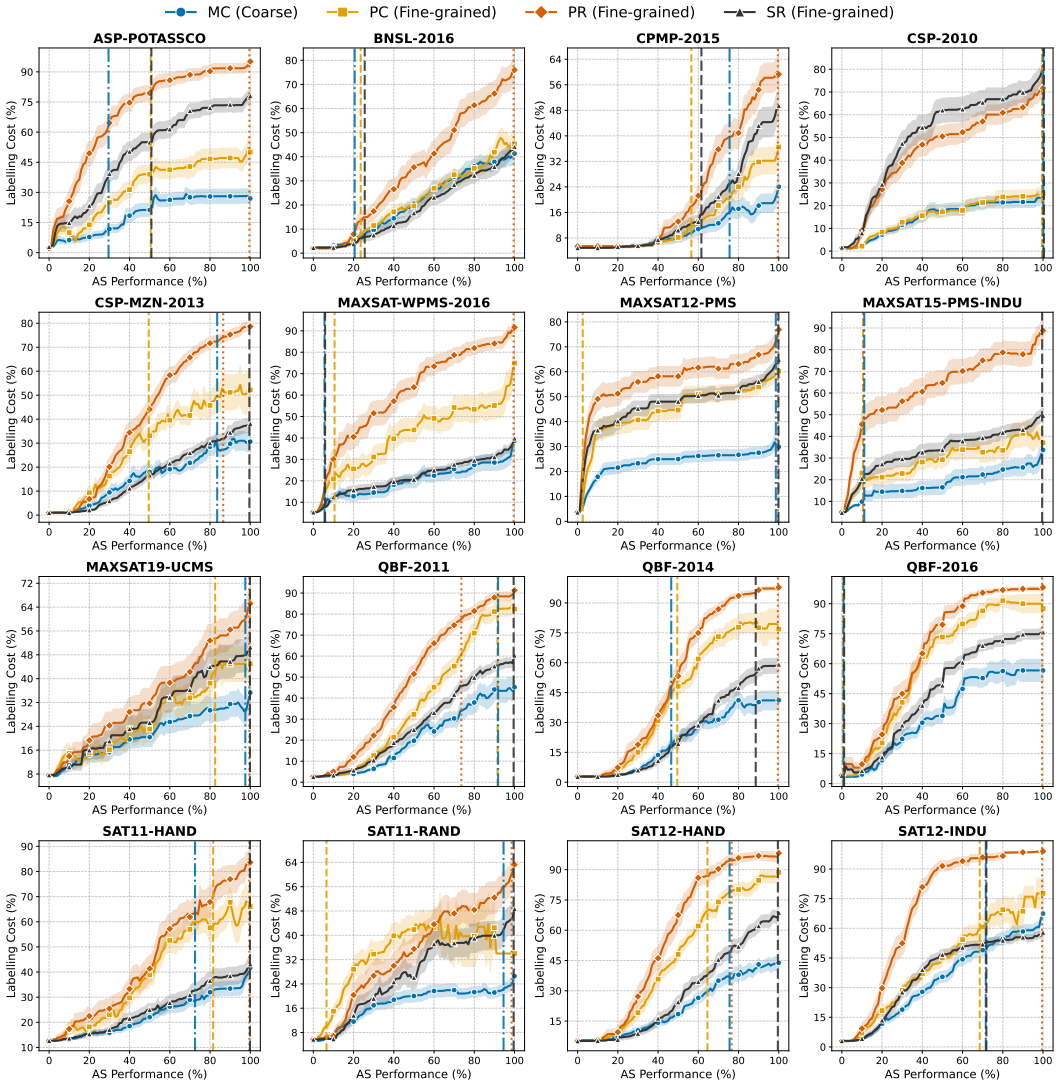

Figure 2 Trade-off between AS performance and labelling cost for MC, PC, PR, and SR under a baseline AL setting across 16 ASLib benchmark scenarios. AS performance, for both PL and AL, is expressed as a percentage of the best average PL performance achieved across all formulations, seeds, and splits for each scenario, set to 100% as a shared reference point common to all formulations. Vertical lines indicate the PL performance level of each individual formulation, expressed relative to the shared 100% reference. MC consistently reaches the shared PL reference at a substantially lower labelling cost than the other formulations.

This behaviour is illustrated concretely in Tables 3 and 4, which presents example query batches from the QBF-2014 scenario [11] for MC and PC, respectively. For MC, the total labelling costs vary from as little as 0.708 to 8.632 seconds, yet all queried instances remain relatively cheap to label, providing evidence of this implicit bias towards lower-cost instances. For PC, by contrast, the queried instances are dominated by timeout runs, leading to substantially higher labelling costs. While these observations are in line with the frugality advantage of MC, they do not constitute a full explanation, as other factors, such as the learning formulation itself are also likely to play a role.

Table 3 An example query table generated during AL of MC on the QBF-2014 scenario. The **Instance** column lists the training instances selected for labelling, and the **Unc.** column shows the model uncertainty measured by the MC model for each training instance. The remaining columns (**bcghostq**, **brareqs**, **cbdepqbf**, **cghostq**, **depqbf**, **dual_ooq13**, **ghostq**, **hiqqer1**, **hiqqer3**, **ooq13**, **pre_dual_ooq13**, **rareqs**,**sqube**, **xbdepqbf**) report the runtime in seconds of each of the 14 algorithms on the queried instances, with the **Cost** colum reporting the total labelling cost incurred during the labelling process. All cost values are rounded to three decimal places. The queried instances indicate that MC tends to query cheaper instances, a behaviour observed across most of the scenarios that directly contributes to its strong frugality.

Instance	Unc.	bcghostq	brareqs	cbdepqbf	cghostq	depqbf	dual_ ooq13	ghostq	hiqqer 1	hiqqer 3	ooq13	pre_dual_ ooq13	rareqs	sqube	xbdepqbf	Cost
p10-1 planlen=5	**0.740**	0.195	0.014	0.019	0.178	0.016	0.012	0.179	0.031	0.037	0.011	0.034	0.012	0.036	0.018	**0.792**
p10-1 planlen=4	**0.734**	0.161	0.014	0.026	0.153	0.011	0.008	0.161	0.037	0.030	0.008	0.035	0.013	0.034	0.017	**0.708**
p10-1 planlen=6	**0.734**	0.225	0.015	0.028	0.214	0.012	0.011	0.206	0.036	0.036	0.008	0.035	0.016	0.037	0.020	**0.899**
p10-1 planlen=12	**0.728**	0.665	0.073	1.227	0.651	0.984	0.695	0.833	0.235	0.234	0.551	0.720	0.054	0.690	1.008	**8.620**
p10-1 planlen=9	**0.728**	0.397	0.019	0.032	0.383	0.029	0.012	0.343	0.045	0.049	0.017	0.045	0.025	0.064	0.030	**1.490**
p10-1 planlen=11	**0.728**	0.677	0.037	0.404	0.672	0.293	0.196	0.626	0.107	0.115	0.147	0.234	0.065	0.414	0.296	**4.283**
p20-1 planlen=9	**0.728**	0.930	0.036	0.068	0.902	0.023	0.027	0.821	0.056	0.056	0.020	0.059	0.037	0.090	0.071	**3.196**
p20-1 planlen=11	**0.728**	2.016	0.053	0.478	1.969	0.368	0.214	1.463	0.115	0.105	0.270	0.262	0.097	0.863	0.359	**8.632**

Table 4 An example query table generated during AL of PC on the QBF-2014 scenario. The **Instance** column lists the training instances selected for labelling, and the **Pair** column reports the algorithm pair queried by the PC model for each training instance. The **Uncertainty** column reports the model uncertainty assigned by PC models to each instance. The **Algorithm 1 Cost** and **Algorithm 2 Cost** columns report the runtime in seconds of each algorithm in the pair on the queried instance, where **Timeout** indicates that the algorithm timed out, with the **Total Cost** column reporting the total labelling cost incurred during the labelling process. All cost values are rounded to three decimal places. The queried instances indicate that PC tends to select instances involving timeout runs, resulting in a substantially higher labelling cost compared to MC, a key limitation of PC under AL that directly contributes to its weaker frugality.

Instance	Pair	Uncertainty	Algorithm 1 Cost	Algorithm 2 Cost	Total Cost
p20-5.pddl_planlen=43	cghostq, rareqs	0.5	61.853	Timeout	3661.853
p20-5.pddl_planlen=41	cghostq, rareqs	0.5	61.943	Timeout	3661.943
p20-5.pddl_planlen=40	cghostq, rareqs	0.5	57.482	Timeout	3657.482
ii8a4-00	cghostq, pre_dual_ooq13	0.5	6.672	93.567	100.239
pipesnotankage06_4	hiqqer3, ooq13	0.5	4.002	137.016	141.018
pipesnotankage07_6	hiqqer3, ooq13	0.5	Timeout	Timeout	7200.0
p20-1.pddl_planlen=43	ooq13, sqube	0.5	Timeout	Timeout	7200.0
p20-1.pddl_planlen=44	ooq13, sqube	0.5	Timeout	Timeout	7200.0
p20-10.pddl_planlen=13	brareqs, pre_dual_ooq13	0.5	Timeout	Timeout	7200.0
dungeon_i10-m5-u10-v0.pddl_planlen=190	ghostq, hiqqer3	0.5	148.291	10.512	158.803

Taken together, these results suggest that classification-based formulations, and MC in particular, are considerably more suited to AL than their regression-based counterparts. Convergence plots for all formulations under the DTL, TO, and DTL+TO frugality mechanisms are provided in the Appendix A for further reference, where consistent patterns are observed across all frugality mechanism combinations.

Figure 3 presents a CD diagram ranking all formulations by their frugality across all 16 scenarios. MC (DTL) emerges as the most frugal formulation overall, with PC-based formulations occupying the mid-range rankings and regression-based formulations consistently ranking lowest regardless of the frugality mechanism applied, confirming their limited suitability for frugal AL. Formulations without any frugality mechanism consistently rank

lower than their mechanism-enhanced counterparts. Of particular note is the observation that MC (DTL) and MC (DTL+TO) are not statistically distinguishable, as the strict consensus requirement of the TO mechanism is rarely satisfied for MC, leaving it largely ineffective in this setting.

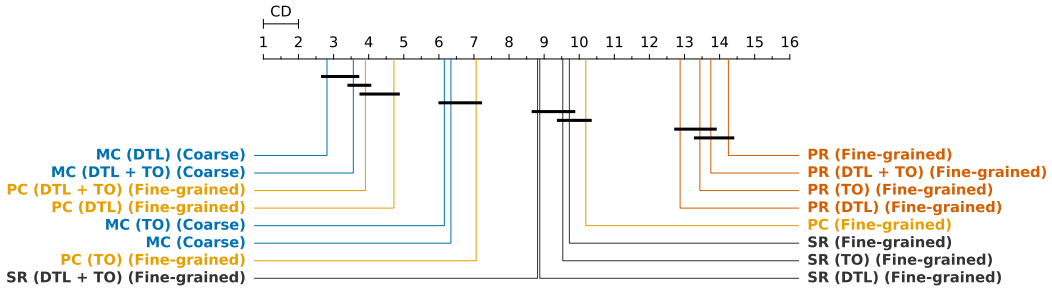

Figure 3 CD diagram comparing the frugality of all learning formulations with frugality mechanism combinations, where frugality is measured as the fraction of labelling cost required to reach 100% of the best average PL performance of the strongest scenario-specific passive learner across all formulations, seeds, and splits. MC-based formulations, particularly MC (DTL) and MC (DTL + TO), rank consistently higher than all other formulations, demonstrating the effectiveness of MC within AL.

These results raise a natural question: is the strong frugality of MC driven by its learning formulation, or is it a consequence of its coarse-grained query granularity? The following section investigates this question by examining whether applying the coarse-grained labelling strategy of MC to other AS formulations produces comparable frugality gains.

5.3 Effect of Coarse-Grained Granularity in Active Learning

The results presented in the previous section demonstrate that MC achieves strong frugality within an AL framework. A natural question that arises is whether this efficiency is driven by the learning formulation itself or by the coarse-grained nature of its query strategy. To investigate this, the coarse-grained query strategy of MC is applied to the other AS formulations, whereby all models jointly select instances based on their mean uncertainty, in contrast to the standard setting in which each model queries instances independently.

Figure 4 presents the trade-off between AS performance and labelling cost for all formulations under both standard and coarse-grained settings across all 16 ASLib benchmark scenarios. MC continues to achieve the most favourable trade-off between labelling cost and AS performance, consistent with the results reported in the previous section. Crucially, coarse-grained variants of all other formulations consistently reach comparable AS performance at a lower labelling cost than their standard fine-grained counterparts, providing clear evidence that the coarse-grained query strategy contributes positively to frugality in its own right, independently of the underlying learning formulation. This improvement varies across classification and regression-based formulations. For PC, the coarse-grained strategy mitigates the risk of the query process being dominated by timeout runs, as all binary models are required to collectively agree on the same instance rather than querying independently. For regression-based formulations, by contrast, the primary benefit lies in the ability of the coarse-grained strategy to regularise the query process by constraining high-variance models to agree on the same instance, thereby reducing the negative effect of output variance on frugality. Convergence plots for all formulations under both standard

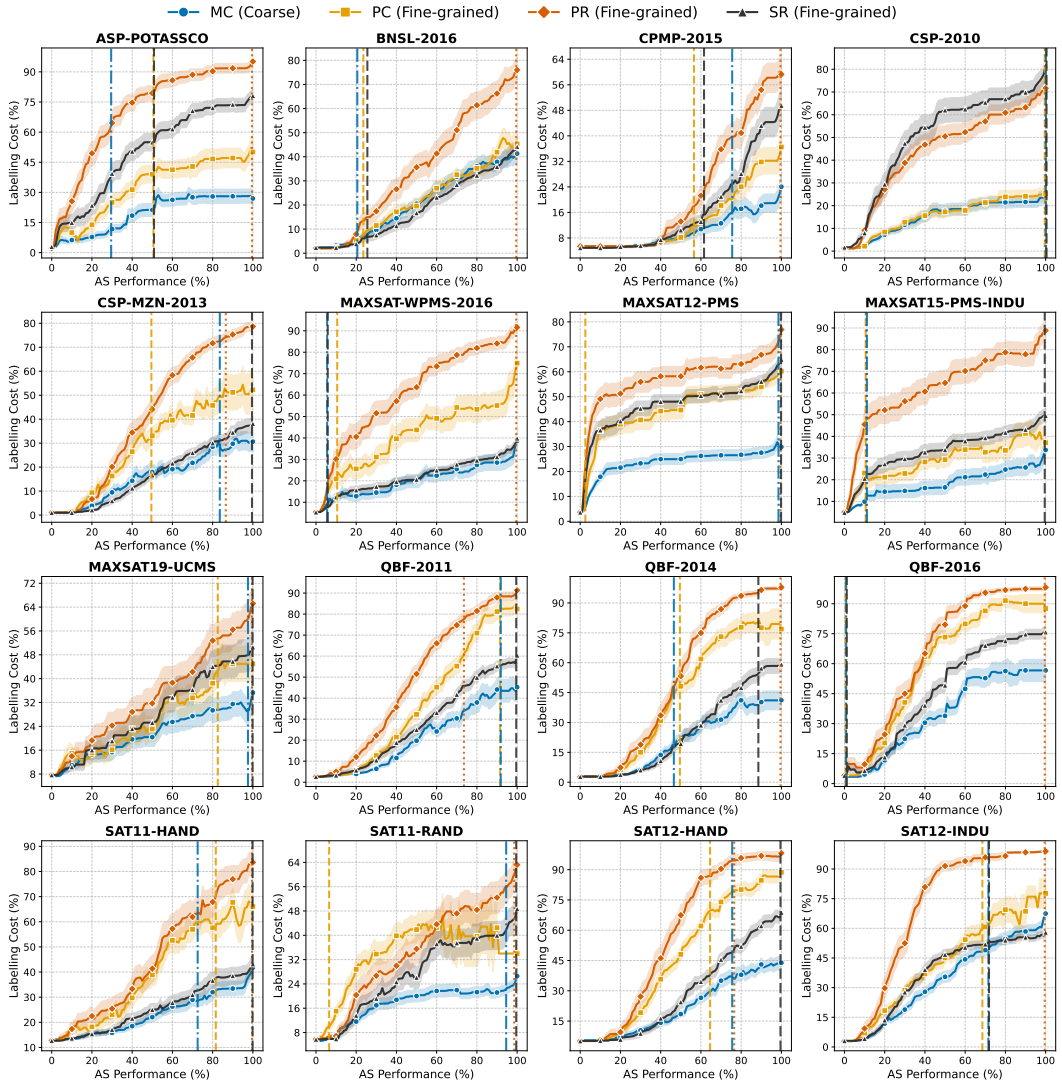

Figure 4 Trade-off between AS performance and labelling cost for MC, PC, PR, and SR under both standard and coarse-grained querying strategies across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Dashed lines represent coarse-grained variants of each formulation, while solid lines represent their standard fine-grained counterparts. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC consistently achieves the most favourable trade-off between labelling cost and AS performance, while coarse-grained variants consistently require a lower labelling cost than their standard fine-grained counterparts across all formulations. Although SR (Coarse) appears competitive on several scenarios, its high variance and failure to reach multiple PL performance levels on some scenarios suggest it is an unreliable formulation in this setting.

and coarse-grained settings across the DTL, TO, and DTL+TO frugality mechanisms are provided in the Appendix B for further reference, extending the previous comparison to include coarse-grained variants of all formulations.

To provide statistical evidence for the observed differences between formulations, Figure 5 presents a CD diagram comparing the frugality rankings of all formulations under both standard and coarse-grained settings across all 16 scenarios. The results are consistent

with those of the previous section, with MC-based formulations retaining their top frugality rankings and further confirming their strong frugality within an AL framework. Crucially, under the standard AL setting without any frugality mechanisms, coarse-grained variants consistently achieve higher frugality rankings than their fine-grained counterparts across all other learning formulations, providing statistical evidence that the coarse-grained query strategy itself contributes positively to frugality, independently of the underlying learning formulation.

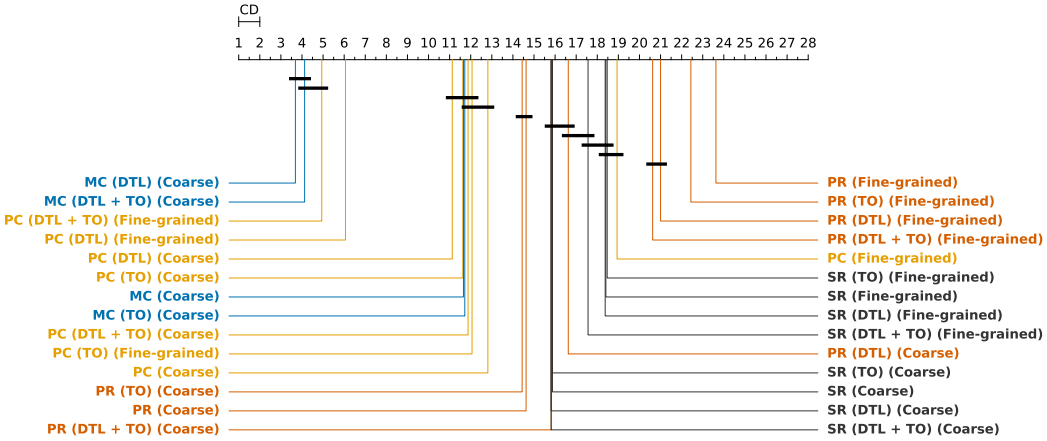

Figure 5 CD diagram comparing the frugality (fraction of labelling cost to reach 100% performance) rankings of all AS formulations with both fine-grained and coarse-grained variants across 16 ASLib benchmark scenarios. MC-based formulations continue to occupy the top frugality rankings, while coarse-grained variants of other learning formulations consistently rank higher than their standard fine-grained counterparts under the standard AL setting.

However, when frugality mechanisms are applied, PC fine-grained variants already achieve competitive frugality rankings, with their coarse-grained counterparts providing no consistent additional benefit. For regression-based formulations, coarse-grained variants consistently achieve higher frugality rankings than their fine-grained counterparts across all frugality mechanism combinations, further supporting the observation that constraining high-variance models to agree on the same instance effectively regularises the query process. Nevertheless, regression-based formulations continue to rank considerably lower than classification-based formulations regardless of the strategy applied, confirming that while coarse-grained granularity positively influences frugality, the learning formulation itself remains a key determinant of frugality in AL.

To summarise, the key findings of this study are as follows:

- **Passive learning performance**: Regression-based formulations achieve stronger PL performance and are closer to the SBS-VBS gap, while MC generally performs poorly compared to other formulations (Section 5.1).
- **MC as a weak passive but strong active learner**: Despite its weak PL performance, MC achieves the strongest frugality under AL, consistently matching the best PL performance at a substantially lower labelling cost, followed by PC as the second most frugal formulation, while regression-based formulations require a substantially higher labelling cost to reach comparable AS performance (Section 5.2 and Section 5.3).
- **Effect of coarse-grained granularity**: Coarse-grained querying consistently improves frugality across all learning formulations. For PC, this benefit diminishes when frugality mechanisms are applied, while for regression-based formulations it persists across all frugality mechanism combinations (Section 5.3).

6 Conclusion

In this work, we presented a systematic empirical investigation of how the choice of learning formulation and query granularity affects frugality in algorithm selection. Four AS learning formulations, namely PC, MC, PR, and SR, were evaluated within a frugal AL framework across 16 ASLib benchmark scenarios to identify the most frugal formulation and examine the effect of coarse-grained querying on frugality across different learning paradigms.

The results revealed a striking and unexpected finding: MC, despite being the weakest passive learner among all formulations, consistently achieves the strongest frugality under AL, matching the best PL performance at a substantially lower labelling cost. This contrasts sharply with regression-based formulations, which, despite their strong PL performance, prove considerably less suited to AL due to their greater sensitivity to training data distribution. PC emerges as the second most frugal formulation, while regression-based formulations require a substantially higher labelling cost to reach comparable AS performance regardless of the frugality mechanism applied.

The investigation of query granularity revealed that applying the coarse-grained query strategy of MC to other formulations consistently improves their frugality under the standard AL setting, providing evidence that coarse-grained querying is itself a positive contributor to frugality, independently of the learning formulation. However, MC retains its top frugality ranking overall, suggesting that its strong frugality is driven by a combination of its learning formulation and coarse-grained query strategy. For regression-based formulations, coarse-grained variants consistently improve frugality across all frugality mechanism combinations, whilst for PC, this benefit diminishes when frugality mechanisms are applied.

These findings have important implications for the design of frugal AS systems. In settings where labelling cost is a primary concern, MC with DTL emerges as the recommended formulation, offering the most favourable trade-off between labelling cost and AS performance. More broadly, the results highlight the importance of considering both learning formulation and query granularity when designing frugal AL strategies for AS, and suggest that simple models that underperform under passive learning can become highly effective when training data is selected actively.

Future work could investigate the effect of alternative acquisition functions in active learning for frugal algorithm selection and their interactions with the studied mechanisms in this work. We also plan to expand the frugal AS framework to other combinatorial optimisation domains beyond those covered by ASLib.

References

1 Jinghou Bi, Yuanhao Xu, Felix Conrad, Hajo Wiemer, and Steffen Ihlenfeldt. A comprehensive benchmark of active learning strategies with AutoML for small-sample regression in materials science. *Scientific Reports*, 15(1):37167, 2025. doi:10.1038/s41598-025-24613-4.

2 Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. Aslib: A benchmark library for algorithm selection. *Artif. Intell.*, 237:41–58, 2016. doi:10.1016/j.artint.2016.04.003.

3 Rui M. Castro, Rebecca Willett, and Robert D. Nowak. Faster rates in regression via active learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, volume 18, pages 179–186. MIT Press, 2005. URL: https://proceedings.neurips.cc/paper/2005/hash/4ea6a546c19499318091a9df40a13181-Abstract.html.

4 David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, May 1994. doi:10.1007/BF00993277.

5 Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL: http://jmlr.org/papers/v7/demsar06a.html.

6 Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, Miha Štajdohar, Lan Umek, Lan Žagar, Jure Žbontar, Marinka Žitnik, and Blaž Zupan. Orange: Data mining toolbox in python. *Journal of Machine Learning Research*, 14:2349–2353, 2013. doi:10.5555/2567709.2567736.

7 Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. doi:10.1080/01621459.1937.10503522.

8 Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artif. Intell.*, 126(1-2):43–62, 2001. doi:10.1016/S0004-3702(00)00081-3.

9 Steve Hanneke. Activized learning: Transforming passive to active with improved label complexity. *Journal of Machine Learning Research*, 13(49):1469–1587, 2012. doi:10.5555/2503308.2343693.

10 Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997. doi:10.1126/science.275.5296.51.

11 Mikolas Janota, Charles Jordan, Will Klieber, Florian Lonsing, Martina Seidl, and Allen Van Gelder. The qbfgallery 2014: The QBF competition at the floc olympic games. *J. Satisf. Boolean Model. Comput.*, 9(1):187–206, 2014. doi:10.3233/sat190108.

12 Pascal Kerschke, Holger H Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019. doi:10.1162/evco_a_00242.

13 Lars Kotthoff. *Algorithm Selection for Combinatorial Search Problems: A Survey*, pages 149–190. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-50137-6_7.

14 Erdem Kus, Özgür Akgün, Nguyen Dang, and Ian Miguel. Frugal algorithm selection (short paper). In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming, CP 2024, Girona, Spain, September 2-6, 2024*, volume 307 of *LIPIcs*, pages 38:1–38:16, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.CP.2024.38.

15 Marius Lindauer, Holger H Hoos, Frank Hutter, and Torsten Schaub. Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015. doi:10.1613/jair.4726.

16 Marius Lindauer, Jan N. van Rijn, and Lars Kotthoff. The algorithm selection competitions 2015 and 2017. *Artif. Intell.*, 272:86–100, 2019. doi:10.1016/j.artint.2018.10.004.

17 Peter B. Nemenyi. *Distribution-free multiple comparisons.* PhD thesis, Princeton University, 1963.

18 John R. Rice. The algorithm selection problem. *Adv. Comput.*, 15:65–118, 1976. doi:10.1016/S0065-2458(08)60520-3.

19 Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL: http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf.

20 Lin Xu, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. SATzilla: portfolio-based algorithm selection for SAT. *Journal of artificial intelligence research*, 32:565–606, 2008. doi:10.1613/jair.2490.

A Appendix A: Learning Formulations of Active Learning with Frugality Mechanisms

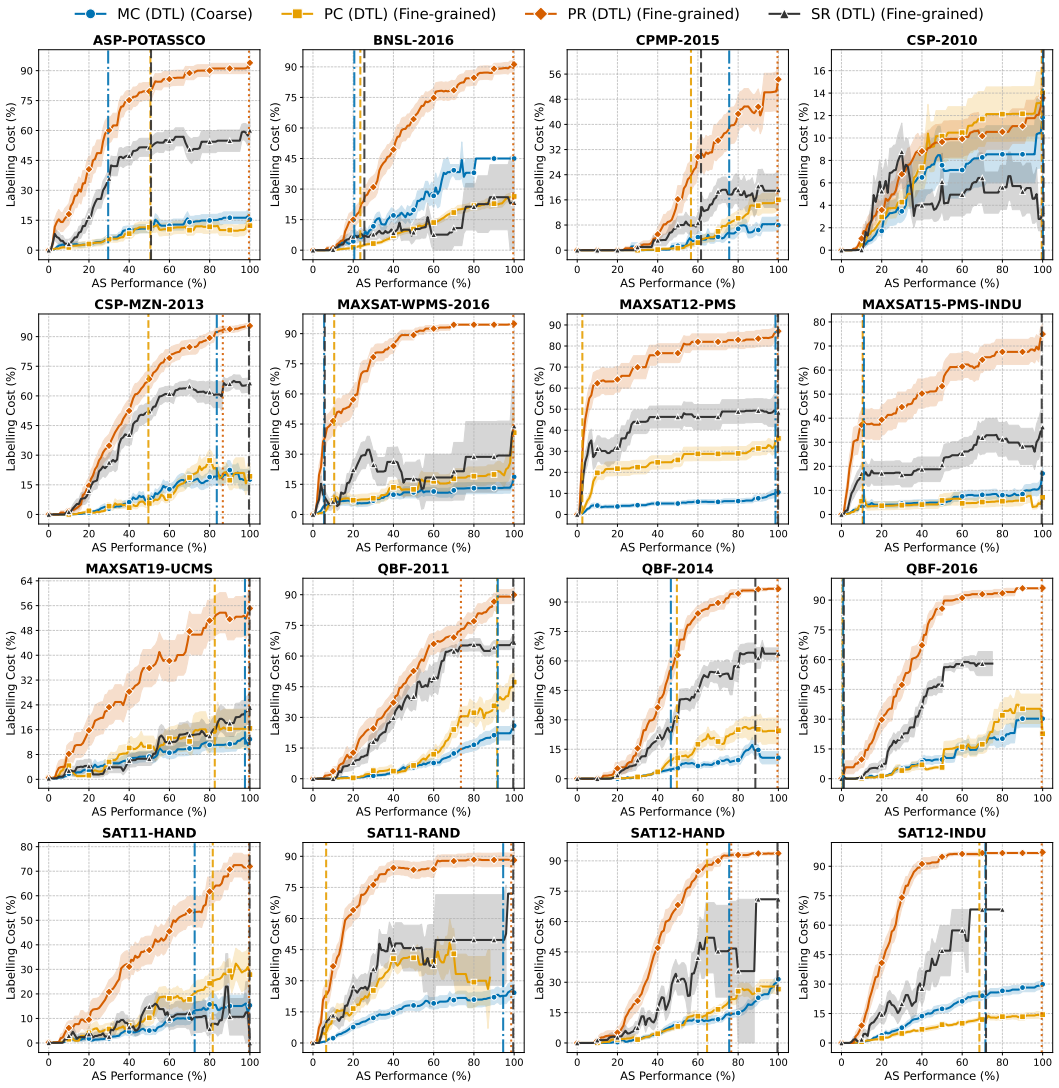

Figure 6 Trade-off between AS performance and labelling cost for MC (DTL) (Coarse), PC (DTL), SR (DTL), and PR (DTL) under fine-grained querying across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (DTL) (Coarse) consistently achieves the most favourable trade-off between labelling cost and AS performance. PC (DTL) (Fine-grained) also demonstrates strong frugality across scenarios, achieving competitive AS performance at a comparable labelling cost to MC (DTL) (Coarse) on several scenarios, whilst PR and SR require a considerably higher labelling cost to reach comparable AS performance.

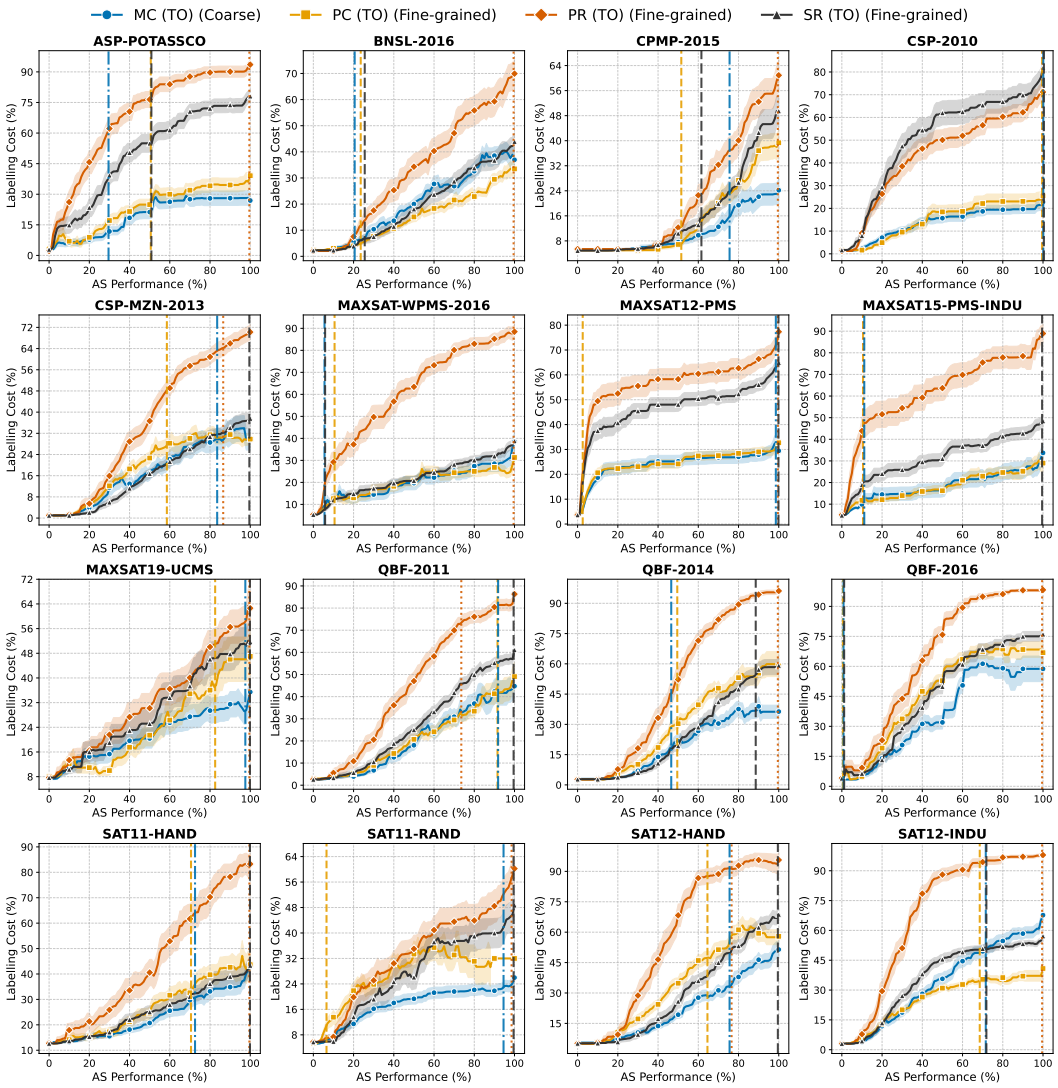

Figure 7 Trade-off between AS performance and labelling cost for MC (TO) (Coarse), PC (TO), SR (TO), and PR (TO) under fine-grained querying across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (TO) (Coarse) consistently achieves the most favourable trade-off between labelling cost and AS performance, with PC (TO) fine-grained emerging as the second most frugal formulation, whilst PR and SR require a considerably higher labelling cost to reach comparable AS performance.

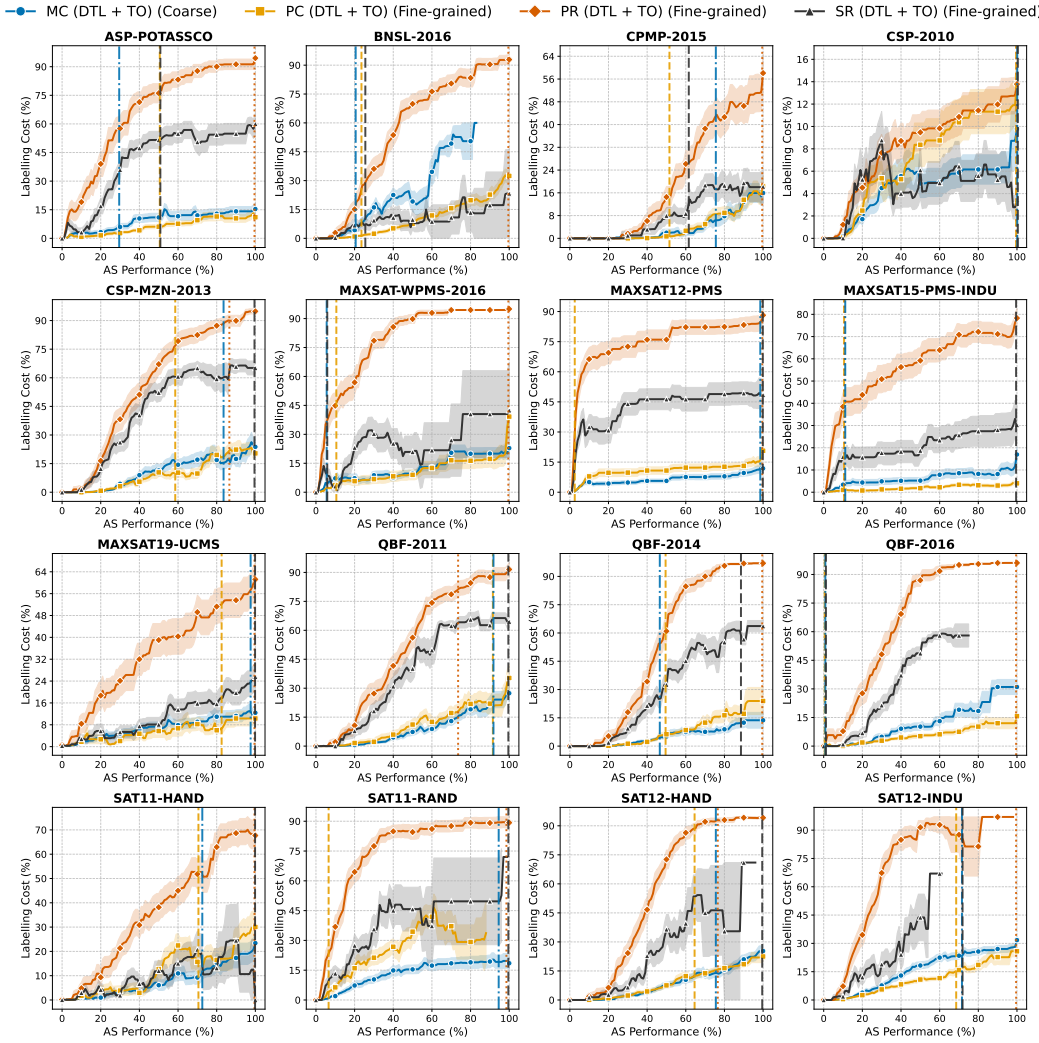

Figure 8 Trade-off between AS performance and labelling cost for MC (DTL+TO) (Coarse), PC (DTL+TO), SR (DTL+TO), and PR (DTL+TO) under fine-grained querying across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (DTL+TO) (Coarse) and PC (DTL+TO) fine-grained achieve comparable frugality across scenarios, with both formulations consistently reaching the best average PL performance at a substantially lower labelling cost than SR and PR, consistent with the observation that the combined application of DTL and TO mechanisms reduces the benefit of coarse-grained querying for PC.

B Appendix B: Trade-off Between AS Performance and Labelling Cost Under Standard and Coarse-Grained Settings

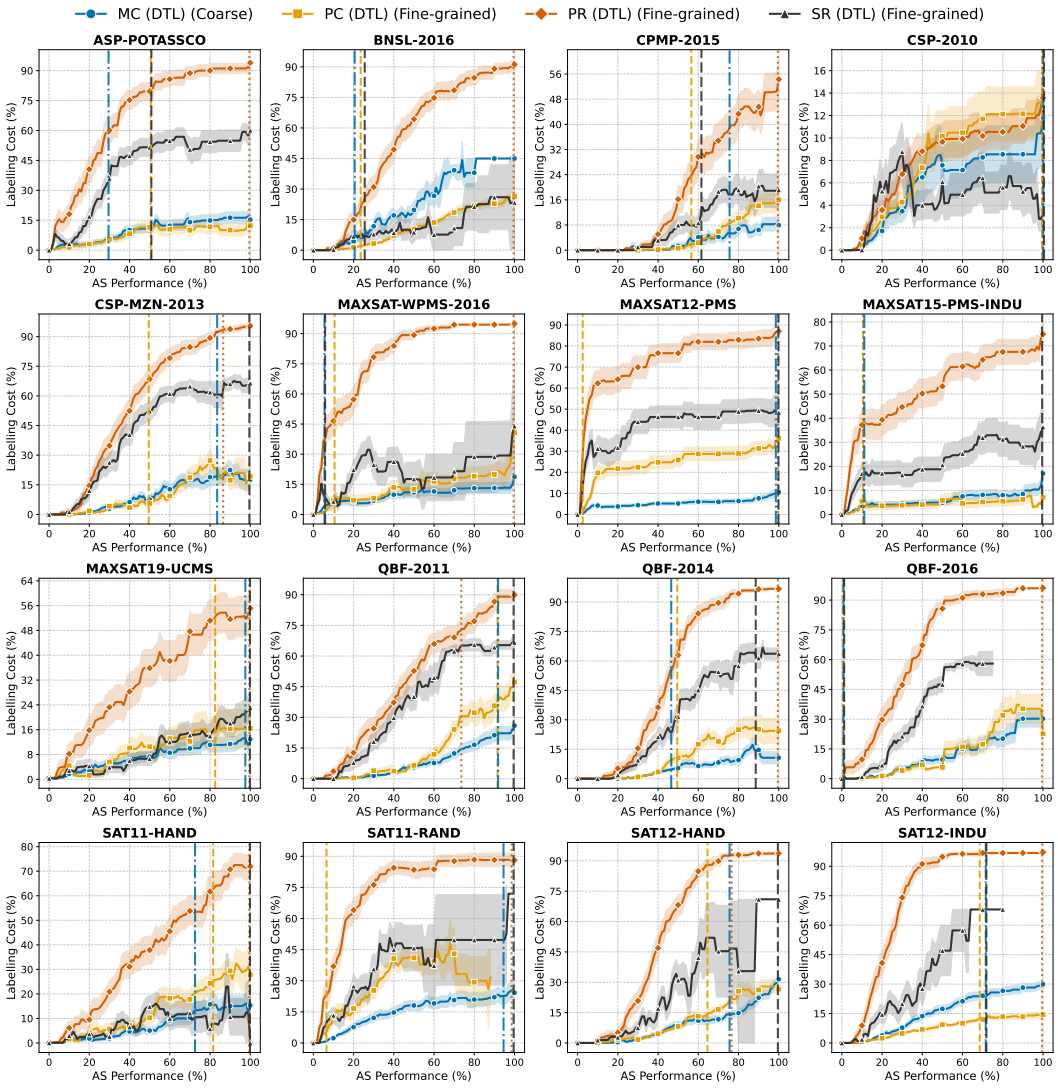

Figure 9 Trade-off between AS performance and labelling cost for all learning formulations under the DTL frugality mechanism, comparing fine-grained and coarse-grained querying strategies across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Dashed lines represent coarse-grained variants of each formulation, while solid lines represent their standard fine-grained counterparts. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (DTL) consistently achieves the most favourable trade-off between labelling cost and AS performance, while coarse-grained variants consistently require a lower labelling cost than their fine-grained counterparts across all formulations.

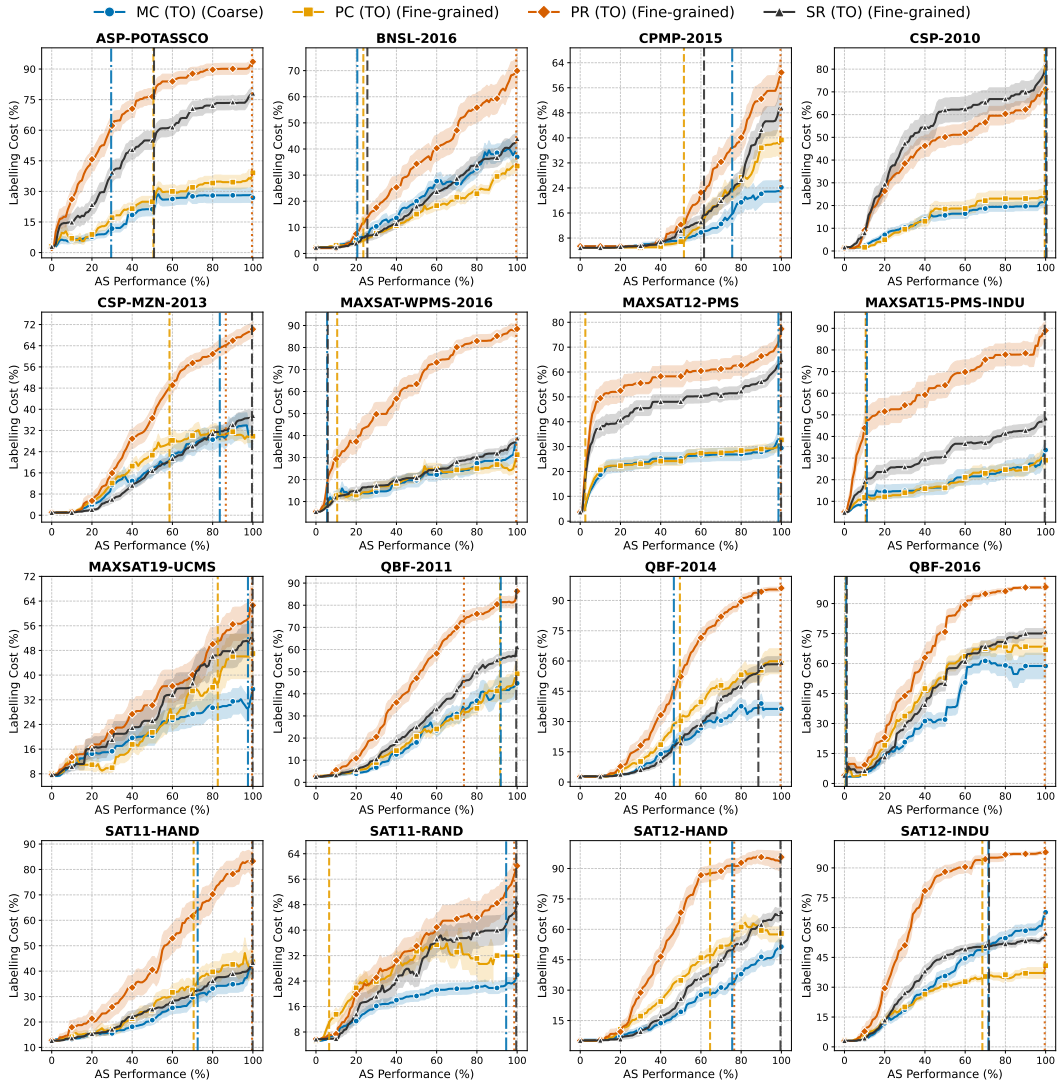

Figure 10 Trade-off between AS performance and labelling cost for all learning formulations under the TO frugality mechanism, comparing fine-grained and coarse-grained querying strategies across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Dashed lines represent coarse-grained variants of each formulation, while solid lines represent their standard fine-grained counterparts. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (TO) consistently achieves the most favourable trade-off between labelling cost and AS performance, while coarse-grained variants consistently require a lower labelling cost than their fine-grained counterparts across all formulations. Although SR (TO) (Coarse) appears competitive on several scenarios, its inconsistent performance across scenarios suggests it is an unreliable formulation in this setting.

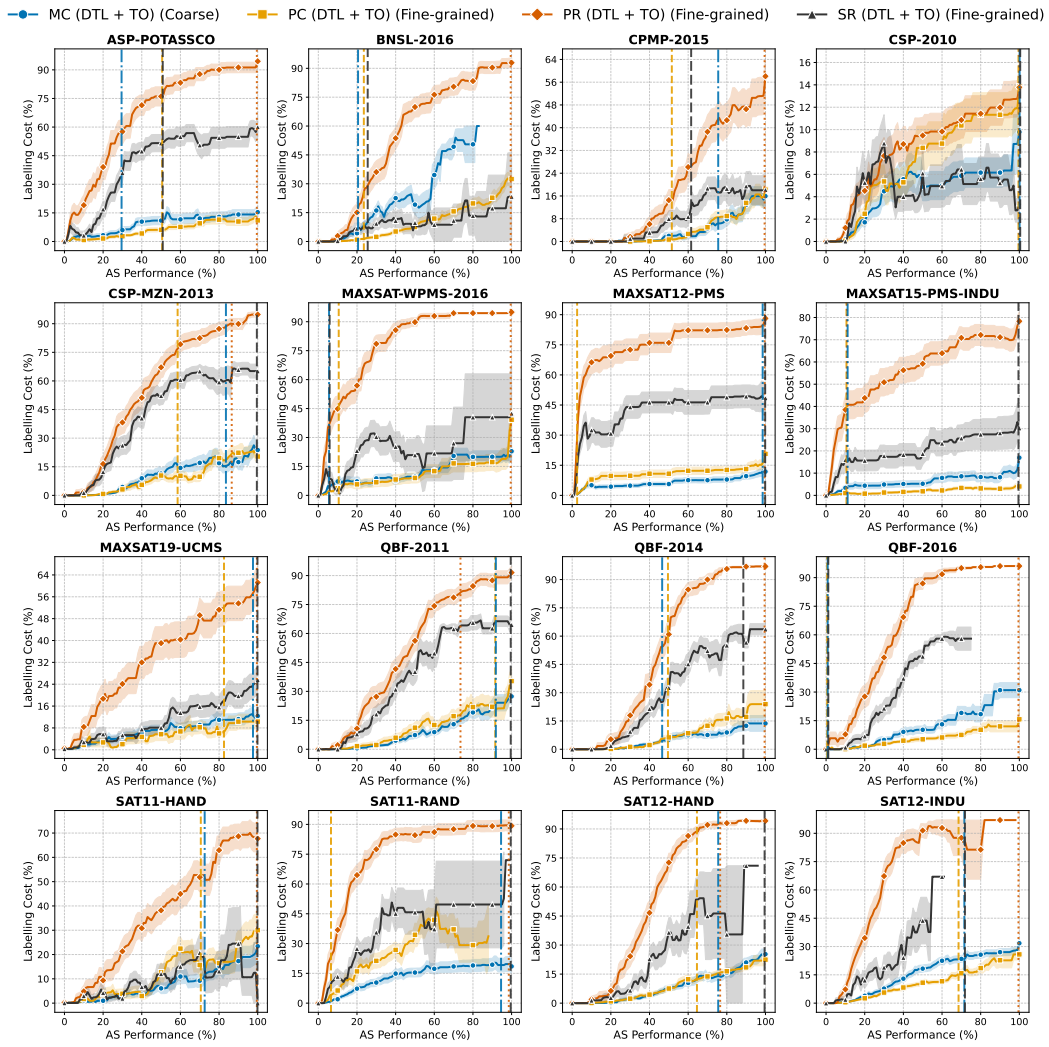

Figure 11 Trade-off between AS performance and labelling cost for all learning formulations under the combined DTL and TO frugality mechanisms, comparing fine-grained and coarse-grained querying strategies across 16 ASLib benchmark scenarios, averaged over multiple seeds and splits. AS performance is expressed as a percentage of the best average PL performance across seeds and splits, set to 100%. Dashed lines represent coarse-grained variants of each formulation, while solid lines represent their standard fine-grained counterparts. Vertical lines indicate the labelling cost at which each formulation reaches the best average PL performance. MC (DTL+TO) consistently achieves the most favourable trade-off between labelling cost and AS performance, with PC (DTL+TO) fine-grained emerging as the second most frugal formulation, consistent with the observation that frugality mechanisms reduce the benefit of coarse-grained querying for PC. Although SR (DTL+TO) (Coarse) appears competitive on several scenarios, its inconsistent performance across scenarios suggests it is an unreliable formulation in this setting.