Frugal Algorithm Selection for Combinatorial Search

ERDEM KUŞ*, ÖZGÜR AKGÜN, NGUYEN DANG, IAN MIGUEL, and LARS KOTTHOFF, University of St Andrews, United Kingdom

Background: Algorithms to solve combinatorial search and optimisation problems typically exhibit complementary performance – on problem instances where one fails, another shines. Selecting the best algorithm for each problem instance, rather than relying on a single algorithm, is crucial for optimal performance. This is known as the Algorithm Selection Problem, which is usually solved by training Machine Learning models that predict the performance of available algorithms. However, the *labelling cost*, i.e., the cost of collecting the training data to build such models, can be extremely high, as it involves running algorithms on all training instances.
Objectives: Rather than following the traditional approach of exhaustively running all algorithms on all training instances to train algorithm selection models, we propose a more *frugal* approach using active learning to intelligently decide which algorithms to run on which problem instances, thus reducing the labelling cost to build algorithm selection systems.
Methods: We find that standard active learning strategies perform poorly in algorithm selection settings, where labelling costs are inherently non-uniform due to the highly variable runtimes of algorithms across problem instances. Therefore, we propose novel active learning strategies for algorithm selection that leverage a dynamic time limit mechanism, together with auxiliary models for predicting timeouts and algorithm runtimes, to decide which algorithm runs strike the best balance between cost and informativeness.
Results: The proposed active learning strategies significantly outperform traditional approaches in terms of frugality, achieving competitive performance at substantially lower labelling cost. Our results demonstrate that we can save up to 90% of the labelling cost associated with generating training data for algorithm selection models without loss in algorithm selection performance.
Conclusions: Our Frugal Algorithm Selection framework, which leverages a dynamic time limit mechanism, prediction uncertainty, and the auxiliary models for timeout and runtime prediction, can dramatically reduce labelling cost without compromising predictive performance. This allows the training of algorithm selection systems to be more efficient for combinatorial optimisation problems.

JAIR Track: Constraint Programming and Machine Learning

JAIR Associate Editor: Chu-Min LI

JAIR Reference Format:
Erdem Kuş, Özgür Akgün, Nguyen Dang, Ian Miguel, and Lars Kotthoff. 2026. Frugal Algorithm Selection for Combinatorial Search. *Journal of Artificial Intelligence Research* 86, Article 27 (July 2026), 31 pages. DOI: 10.1613/jair.1.21266

1 Introduction

Solving combinatorial optimisation problems often requires large computational resources, with different approaches competing to deliver the most efficient solution. Different *algorithms*, in which we include both modelling

*Corresponding Author.

Authors' Contact Information: Erdem Kuş, ORCID: 0000-0001-7775-5610, ek232@st-andrews.ac.uk; Özgür Akgün, ORCID: 0000-0001-9519-938X, ozgur.akgun@st-andrews.ac.uk; Nguyen Dang, ORCID: 0000-0002-2693-6953, nttd@st-andrews.ac.uk; Ian Miguel, ORCID: 0000-0002-6930-2686, ijm@st-andrews.ac.uk; Lars Kotthoff, ORCID: 0000-0003-4635-6873, lk223@st-andrews.ac.uk, University of St Andrews, St Andrews, United Kingdom.

DOI: 10.1613/jair.1.21266

and solver choices, tend to exhibit complementary performance across different problem instances. Selecting the most suitable algorithm for a given instance has been shown to yield substantial efficiency gains (Xu, Hutter, H. H. Hoos, et al. 2008).

Choosing the most suitable algorithm for a given problem instance is known as the Algorithm Selection Problem (AS) (Rice 1976). Machine Learning (ML) has proven to be a powerful approach for automating this process. By examining instance features and algorithm performance, ML models can be trained to recommend suitable algorithms, improving computational efficiency and overall problem-solving performance (see, e.g., Kerschke, H. H. Hoos, et al. 2019; Kotthoff 2016). A key limitation, however, is the *labelling cost*, as constructing the training set for the ML models requires running candidate algorithms on each training instance to identify the best-performing one, which is often computationally expensive.

In this work, we propose a *Frugal Algorithm Selection (FAS)* framework, where *frugality* refers to minimising the computational cost of constructing the training set by selecting only the most informative and inexpensive runs. It is an iterative approach that uses Active Learning (AL) (Cohn et al. 1994) to determine which runs (i.e., the execution of a single algorithm on a single problem instance) to select to construct the training set. AL is a subfield of ML that is particularly useful for problems where obtaining labels is expensive, and it iteratively selects data points to label. This often achieves performance comparable to using the full training set with only a small fraction of labelled samples.

Standard AL methods minimise labelling cost by reducing the number of labelled samples, assuming that all labels can be obtained at the same cost. However, in algorithm selection, labelling cost is not uniform, as different algorithms take different amounts of time on the same instance. Further, many runs are censored due to predefined time limits, and their true runtime and the label for the problem instance remain unknown even after incurring a cost. Following the AS literature (Lindauer, van Rijn, et al. 2019), we use pairwise classification as the core AS approach, comparing pairs of runs to predict which algorithm performs better, and build frugal algorithm selection strategies on top of it.

As the first technical contribution, we introduce two frugality mechanisms that address the limitations of standard AL in AS: *timeout prediction (TO)* and *dynamic time limits (DTL)*. TO is used to improve frugality in the query by filtering out pairs of runs that are likely to time out, as they are costly and uninformative. DTL enhances frugality in labelling by starting with a low time limit and gradually increasing it, prioritising the labelling of inexpensive pairs of runs and reducing time spent on those likely to time out. Further details of these two mechanisms and how they are combined are provided in Section 4.2.

As the second technical contribution, we introduce cost-awareness into the selection process to further enhance frugality, using two new metrics: *predicted cost (PC)* and *predicted runtime difference (PD)*. For a pair of runs, PC corresponds to the total estimated computational cost, while PD corresponds to the estimated absolute runtime difference between the two algorithms. These metrics become particularly important when both algorithms time out for a particular instance, as in this case, we obtain no new information yet pay the maximum possible cost. To evaluate the effectiveness of these cost-aware metrics and their combination with the uncertainty metric used in standard AL in a multi-objective setting, we use both *lexicographic ordering*, which prioritises metrics sequentially based on a predefined order, and Pareto optimisation, where metrics are treated as equally important and runs are selected from the Pareto front. Further details of these cost-aware metrics and multi-objective selection strategies are provided in Section 4.3 and Section 4.4.

We evaluate the proposed selection strategies on 16 ASLib (Bischl et al. 2016) scenarios to assess their frugality. Our results show that standard AL alone has limited effectiveness. Although it reduces the number of runs used for training, it does not effectively reduce labelling cost, as it frequently selects expensive runs. Adding TO and DTL makes standard AL substantially more frugal, with DTL consistently improving all strategies. Amongst all methods, our proposed AL framework with Pareto-based selection achieves the highest frugality, reducing labelling cost by up to 90% while maintaining comparable AS performance to training on the full data.

To summarise, our contributions are as follows:

- A cost-aware active learning framework for algorithm selection that reduces labelling cost while maintaining strong predictive performance.
- Two mechanisms that improve frugality in querying and steer the process towards informative and inexpensive runs: timeout prediction (TO) and dynamic time limits (DTL).
- New metrics for cost-aware selection, predicted cost (PC) and predicted runtime difference (PD), together with lexicographic and Pareto-based strategies that integrate these metrics with uncertainty.
- A comprehensive empirical study on 16 ASlib scenarios, demonstrating the gains in frugality and performance achieved by the proposed methods.

2 Background

This section introduces the key concepts used in our study. We begin with automated AS, which chooses the most suitable algorithm for a given problem instance using instance features and performance data. We then define FAS, which extends the AS framework by reducing labelling cost while preserving AS performance. For our empirical evaluation, we use ASLib, a well-established collection of AS scenarios from different problem domains. Finally, we discuss AL, a machine learning framework that iteratively selects the most informative data points for labelling.

2.1 Automated Algorithm Selection

The AS problem was introduced by Rice (Rice 1976). It involves identifying the most suitable algorithm for solving a given problem instance, usually from a predefined algorithm portfolio (Gomes and Selman 2001) and based on instance features. The objective is to optimise a chosen performance metric across a distribution of problem instances. In this work, we focus on runtime-based AS scenarios, where we minimise the time required to solve each instance rather than other criteria, such as solution quality.

Formally, let $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$ denote a portfolio of n algorithms, each exhibiting complementary strengths across different instances. Let $\mathcal{D}_I$ be a problem instance distribution, where each instance $i \sim \mathcal{D}_I$ is represented by a d-dimensional feature vector $v(i)$. The feature representation may include continuous, categorical, ordinal, or integer features, depending on the problem domain. Let $c(i, a)$ denote the cost of solving instance i using algorithm $a \in \mathcal{A}$. The objective of AS is to learn a selector function f that maps $v(i)$ of an instance i to the algorithm $a \in \mathcal{A}$ expected to perform best on i. More precisely, we want to find f that minimises the expected cost:

$$\underset{f}{\text{minimise}} \quad \mathbb{E}_{i \sim \mathcal{D}_I} \left[c(i, f(v(i))) \right].$$

In practice, we are often given a set of problem instances sampled from $\mathcal{D}_I$. This set is split into a training set and a test set. A machine learning-based AS approach would then use the training set to build an algorithm selector, and the performance of this selector is evaluated on the test set.

Passive learning. In practice, most AS approaches rely on full performance data – each algorithm in the portfolio is run on each training problem instance, and the resulting performance data are used to train models to select the best algorithm for each problem instance. We term this approach *passive learning*. However, obtaining full training data is expensive, as many algorithm runs can take a long time.

To address this limitation, we introduce the concept of *frugal algorithm selection*, which reduces the labelling cost of data collection while maintaining AS performance.

2.2 Frugal Algorithm Selection

FAS achieves frugality by selecting a subset of runs instead of running every algorithm on every instance. It focuses on runs that are most informative and inexpensive. Formally, we define *full labelling cost* as:

$$C(\mathcal{I} \times \mathcal{A}) = \sum_{i \in \mathcal{I}} \sum_{a \in \mathcal{A}} c(i, a),$$

where $\mathcal{I}$ is the training instance set. Our objective is to select the subset of runs $\mathcal{L} \subset \mathcal{I} \times \mathcal{A}$ with minimal labelling cost such that the expected cost of solving a problem instance with the selector function $f_{\mathcal{L}}$ trained on $\mathcal{L}$ is at most as big as the expected cost for the selector function $f_{\mathcal{I} \times \mathcal{A}}$ trained on the full set of runs:

$$\underset{\mathcal{L} \subset \mathcal{I} \times \mathcal{A}}{\operatorname{argmin}} \quad C(\mathcal{L}) \quad \text{s.t.} \quad \mathbb{E}_{i \sim \mathcal{D}_{\mathcal{I}}} \left[c(i, f_{\mathcal{L}}(v(i))) \right] \leq \mathbb{E}_{i \sim \mathcal{D}_{\mathcal{I}}} \left[c(i, f_{\mathcal{I} \times \mathcal{A}}(v(i))) \right].$$

In particular, we expect $C(\mathcal{L}) \ll C(\mathcal{I} \times \mathcal{A})$.

In this context, different AS learning formulations define the structure of the prediction task and the associated labelling requirements. More specifically, there are three types of AS approaches that are commonly used (Lindauer, H. H. Hoos, et al. 2015): (i) multiclass classification-based AS, which directly predicts the best algorithm for a given problem instance; (ii) regression-based AS, which predicts the performance of each algorithm on a given instance and selects the algorithm with the best predicted performance; and (iii) pairwise classification-based AS, which builds a binary classifier for each pair of algorithms to predict the better algorithm on a given instance and selects the algorithm with the highest number of votes. These formulations differ in their labelling requirements. For (i), labelling each training data point requires running all algorithms on a given problem instance. For (ii), each label involves running one algorithm on an instance. Finally, (iii) requires running a pair of algorithms on a given instance. In this work, we adopt pairwise classification as the foundational AS formulation. This choice is motivated by its strong empirical performance in prior AS studies and competitions (Lindauer, van Rijn, et al. 2019). Because each label in this formulation is derived from an algorithm pair's performance data, FAS identifies pairs of runs that are both informative and inexpensive to obtain.

Throughout the remainder of this paper, the term uncertainty-based selection refers to the strategy that applies a standard AL using an uncertainty acquisition function within the FAS framework (see Section 2.3). The term AL is used more broadly as the general formulation of the underlying ML paradigm.

2.3 Active Learning

Active learning is an ML paradigm that achieves high predictive performance with less labelled data by iteratively selecting the most informative data points for labelling (Cohn et al. 1994; Huang et al. 2014; Settles 2009). Rather than relying on a large, fully labelled training set, AL begins with a small set of labelled data and incrementally expands it by obtaining the ground truth labels for selected unlabelled data points. A typical AL loop consists of the following four stages:

Initial Training A predictive model is trained on a small dataset of initial labels.

Query The trained model identifies data points and selects those whose labelling is expected to most reduce uncertainty or improve predictive performance, based on a predefined acquisition strategy.

Labelling The selected data points are labelled with ground truth values. In pairwise-classification AS, this involves comparing pairs of runs on the same instance to identify the better algorithm.

Update The model is retrained using the updated labelled data.

These core steps remain consistent across AL frameworks, but their implementation may vary depending on assumptions about data access and the acquisition function used for determining what data points to obtain ground truth labels for. These variations give rise to distinct *AL variants* (Cuong et al. 2013; Park and Pillow 2012;

Wang et al. 2015; Zhu et al. 2010), each reflecting different constraints and opportunities in real-world settings. The three most widely studied variants are:

Pool-based sampling The learner has access to a large pool of unlabelled data and selects queries from this pool based on an acquisition function (Lewis and Gale 1994).

Stream-based sampling Unlabelled data points arrive sequentially, typically one at a time, from a data source. For each data point, the learner must immediately decide whether to query its label or discard it, often under the assumption that labelling is relatively inexpensive (Atlas et al. 1989; Cohn et al. 1994).

Membership query synthesis The learner is allowed to generate new data points synthetically and query their labels, potentially exploring parts of the data space not present in the original data distribution (Angluin 1988).

As AS considers predefined problem instances rather than a continuous data stream, we adopt a pool-based active learning setting, in which the learner selectively queries from a large pool of unlabelled data.

Regardless of the variant, a key component of AL is the *acquisition strategy* used to determine which data points to label. Among these, uncertainty-based strategies are the most commonly used, as they select data points where the model is least confident, thereby maximising the information gain from each query. The three most widely used uncertainty measures are:

Least Confidence (LC) Selects data points for which the model's most confident prediction has the lowest posterior probability (Lewis and Catlett 1994).

Margin Sampling (MS) Chooses data points where the difference between the top two predicted probabilities is smallest (Scheffer et al. 2001).

Entropy-Based Sampling (ES) Selects data points with the highest entropy over the full posterior distribution (Shannon 1948).

In FAS, the three uncertainty measures (LC, MS, and ES) lead to identical query selections across iterations. Although defined differently, they identify the same maximally uncertain data points in our pairwise classification learning formulation, where we have only two classes. Further details of these uncertainty measures are provided in Section A.

2.4 The Algorithm Selection Library

The Algorithm Selection Library (ASLib) (Bischl et al. 2016) is a widely used benchmark repository for developing and evaluating AS methods. It provides a standardised format for representing AS scenarios across diverse problem domains and currently contains 45 datasets covering areas such as Boolean Satisfiability (SAT), Quantified Boolean Formula (QBF), Maximum Satisfiability (MAX-SAT), Constraint Satisfaction Problems (CSP), Answer Set Programming (ASP), Bayesian Network Structure Learning (BNSL), and other combinatorial optimisation problems. ASLib includes both runtime-based and solution-quality-based performance measures; our study focuses on runtime scenarios, where performance is determined by the algorithm's runtime on a given instance. As is standard in the AS literature, we rely on ASLib's pre-recorded runtimes rather than re-running the algorithms. ASLib was specifically designed to address the lack of a standard format for sharing and comparing approaches across the community, and using its pre-recorded data is the intended and established use of this benchmark.

ASlib uses the following performance metrics and reference baselines.

PAR10 A penalised runtime metric that measures the algorithm's runtime on each instance; if the algorithm fails to solve an instance within the time limit, a penalty of ten times the time limit is applied.

Virtual Best Solver (VBS) An oracle baseline that, for every instance, selects the algorithm with the best performance. It represents an upper bound on the performance achievable by any AS system.

Single Best Solver (SBS) The SBS serves as a natural baseline, representing the algorithm with the best average performance across all training instances.

Several ASLib scenarios require more than 200 CPU days for full data collection, with some demanding up to 3 CPU years. This labelling cost is a major limitation in AS systems, particularly when scaling to new domains or expanding existing portfolios, and motivates the need for more efficient labelling strategies, as addressed by the FAS framework.

3 Related Work

This section provides an overview of the related work on Algorithm Selection, Dynamic Time Limits, and Timeout Prediction.

3.1 Algorithm Selection Techniques and Applications

Over the past two decades, AS has become a cornerstone of performance improvement across computational domains, including Constraint Programming (CP) (O'Mahony et al. 2008; Spracklen et al. 2023; Zhang et al. 2024), SAT (Xu, Hutter, Shen, et al. 2012; Xu, Hutter, H. H. Hoos, et al. 2008), AI planning (Rizzini et al. 2016; Vallati et al. 2014), and combinatorial optimisation (Kerschke, Kotthoff, et al. 2018; Müller et al. 2022). The fundamental premise of AS is to predict, for each problem instance, the algorithm expected to perform best based on measurable instance features (Rice 1976). Its effectiveness arises from performance complementarity, the observation that no single algorithm consistently dominates across all instances (Huberman et al. 1997).

Modern AS systems rely heavily on ML trained on problem instance features, portfolios of algorithms, and performance metrics (Vanschoren 2019). Pioneering systems such as SATzilla (Xu, Hutter, H. H. Hoos, et al. 2008) employed empirical performance models with handcrafted features to predict runtime or success probability. Subsequent studies introduced many other approaches, including k-nearest neighbours (Collautti et al. 2013), clustering-based selection (Ansótegui, Gabas, et al. 2016; Kadioglu et al. 2010), cost-sensitive pairwise classification (Xu, Hutter, Shen, et al. 2012) and more recently Large Language Model (LLM)-based selection (Wu et al. 2024). Several practical enhancements have improved the robustness and generalisation of AS systems. Pre-solving schedules (H. Hoos et al. 2014; Lindauer, H. H. Hoos, et al. 2015; Xu, Hutter, Shen, et al. 2012) execute short runs before feature computation to efficiently handle trivial instances, while algorithm scheduling (Amadini et al. 2014; Kashgarani and Kotthoff 2023; Liu et al. 2021; Margraf et al. 2025) constructs per-instance sequences of algorithms to minimise the impact of prediction errors.

Research on AS has explored different execution paradigms, particularly sequential and parallel portfolios. Early work introduced parallel portfolios, where multiple algorithms run concurrently to enhance robustness and overall performance (Gomes and Selman 2001). Later studies demonstrated how effective parallel SAT solvers can be automatically constructed from sequential ones (Lindauer, H. Hoos, et al. 2017) and examined whether AS remains beneficial compared to executing algorithms in parallel, highlighting that parallel execution is often less suitable due to performance trade-offs and resource contention (Kashgarani and Kotthoff 2021). In contrast, the sequential setting, popularised by systems like SATzilla, selects a single algorithm per instance to reduce redundant computation and improve efficiency. The sequential setting where a single algorithm is selected remains the primary focus of most studies, including ours, owing to its efficiency and simplicity. For comprehensive surveys of AS techniques and their applications, we refer the reader to (Kerschke, H. H. Hoos, et al. 2019; Kotthoff 2016).

A major challenge in AS arises from incomplete or censored training data. Evaluating all runs to completion is infeasible, which is why time limits are typically applied. Running all algorithms under these limits, while feasible, remains computationally expensive. Several strategies have been proposed to mitigate this challenge. ALORS (Mısır and Sebag 2017) formulates AS as a collaborative filtering problem, leveraging latent representations of instances and algorithms to predict performance in sparse evaluation matrices. Unlike traditional AS approaches that require exhaustive evaluations, ALORS operates effectively on partially observed data by inferring missing

performance values and recommending promising algorithms. Arbelaez and Hamadi Arbelaez et al. (2010) proposed *Continuous Search*, a lifelong learning framework that alternates between solving and exploration phases to improve heuristics as new instances are encountered iteratively, eliminating the need for predefined training data. For censored runtimes, survival analysis provides a principled framework for handling incomplete observations. Tornede et al. (Tornede et al. 2020) introduced *Run2Survive* for survival-based AS, and Margraf et al. (Margraf et al. 2025) extended this idea in *Run2ScheduleAndSurvive* for algorithm scheduling, facilitating risk-aware performance prediction under censored data.

Building on this foundation, our work takes a complementary perspective. While previous studies have mainly focused on improving AS performance or handling missing information, our approach emphasises *frugality* by selecting a small but informative subset of runs and minimising their labelling cost through cost-awareness and frugality mechanisms, while still maintaining AS performance. The most closely related work is by Volpato and Song (Volpato and Song 2019), who use AL to reduce the number of labelled samples required to train a model for AS. However, their approach focuses solely on reducing the number of data points and does not account for labelling cost. In algorithm selection, this distinction is important because it is a non-uniform-cost problem: reducing the number of runs does not guarantee the same reduction in labelling cost, as runs vary in cost. Therefore, selecting runs that are both informative and inexpensive is essential and the focus of this work, rather than simply selecting fewer runs.

3.2 Dynamic Time Limit and Timeout Prediction

In AS, managing computational resources efficiently is essential for achieving robust performance across diverse problem domains. Timeout prediction estimates whether a given algorithm will solve an instance within a predefined time limit, thereby preventing unnecessary computation on runs likely to time out. In addition, a dynamic time limit can be adjusted according to various criteria, such as observed algorithm performance or predefined total computational budgets. Such adaptive mechanisms enable more informed resource allocation, reduce wasted computation, and improve overall system reliability in real-world settings.

Several related approaches in hyperparameter optimisation (HPO) and automated algorithm configuration have explored adaptive resource allocation to improve computational efficiency. *Hyperband* (Li et al. 2017), a HPO method that builds on the principle of *successive halving* (Jamieson and Talwalkar 2016), employs a multi-fidelity optimisation strategy that progressively allocates larger computational budgets to promising configurations based on their early performance. By evaluating many configurations with small budgets and iteratively promoting the best ones to receive more resources, Hyperband achieves an effective balance between exploration and exploitation, avoiding full evaluations of unpromising candidates. Similarly, *adaptive capping*, implemented in frameworks such as *ParamILS* (Hutter, H. H. Hoos, et al. 2009), *irace* (López-Ibáñez et al. 2016), *SMAC* (Lindauer, Eggensperger, et al. 2022), and *GGA* (Ansótegui, Sellmann, et al. 2009), dynamically terminates runs when intermediate results indicate that a configuration is unlikely to outperform the current best. This early termination prevents unnecessary computation and allows the system to focus on more competitive configurations. Both approaches manage computational budgets effectively by prioritising promising evaluations, reducing cost without harming model quality. However, our approach differs in how resource adaptation is applied. Rather than dynamically reallocating budgets to explore promising configurations, we focus on selecting runs that are both informative and inexpensive, and systematically minimise the evaluation of costly and uninformative runs. This is achieved through our dynamic time limit mechanism, which maintains a low time limit unless AS performance degrades, thereby aligning with the practical constraints of active learning in AS.

A similar motivation underlies the analysis and prediction of *timeouts*, which proactively identifies runs unlikely to complete within a given time limit, thus avoiding wasteful computation. Kaulen et al. (Kaulen et al. 2025) applied *timeout prediction* within a dynamic termination framework for branch-and-bound-based neural

network verification, using classification models to estimate whether an instance can be solved within a specified time limit and terminating runs expected to exceed it. While their approach makes termination decisions *during* execution, our method applies TO *before* execution, within the query phase of AL, to proactively exclude costly and uninformative runs.

4 The Frugal Algorithm Selection Framework

Building on the foundations of AS, we introduce the Frugal Algorithm Selection (FAS) framework, which provides a unified methodology for reducing labelling cost while preserving AS performance. The framework comprises a set of strategies designed to improve labelling efficiency and analyse their trade-offs. We begin with random selection and uncertainty-based selection as baseline cost-unaware approaches. We further propose selection metrics that explicitly account for cost. Finally, we combine these metrics using multi-objective selection that optimise both informativeness and cost, achieving frugality by balancing performance with labelling efficiency. The parameter settings used in FAS are detailed in Section 5.2.

4.1 Cost-Unaware Run Selection

Pool-based sampling AL operates on a pool of unexecuted runs, from which informative runs are selected for labelling.

4.1.1 Random Selection. As a baseline, random selection samples run uniformly from the pool without relying on model predictions, uncertainty estimates, or any additional metrics. Its simplicity makes it a useful baseline for comparing more informed selection strategies.

4.1.2 Uncertainty-Based Selection. A widely used strategy in AL is uncertainty-based selection, which maximises the informativeness of each query by selecting runs for which the ML models show the lowest confidence. The underlying assumption is that highly uncertain runs are more informative for refining the decision boundary, and therefore improve model performance more effectively than random sampling.

In pairwise classification for AS, runs are queried in pairs, comparing each algorithm pair on a given instance. For any unexecuted pair of runs, the classifier estimates the probability of which algorithm is likely to perform better. Pairs with probabilities close to 0.5 reflect high uncertainty and are considered most informative; therefore, they are prioritised in the query phase.

Algorithm 1 presents a schematic overview of our uncertainty-based selection strategy, which follows the standard uncertainty-based selection implementation (Cohn et al. 1994; Nguyen et al. 2022; Settles 2009) adapted to the AS setting. We begin by constructing the initial labelled set of runs $\mathcal{L} \subseteq \mathcal{I} \times \mathcal{A}$. To construct this, we randomly selected a subset of problem instances and executed all algorithms on them to obtain complete run information. The remaining runs are added to the pool of unexecuted runs $\mathcal{U} = (\mathcal{I} \times \mathcal{A}) \setminus \mathcal{L}$. At each iteration, the uncertainty score of every unexecuted pair of runs (i, a_j, a_k) in the pool are calculated using the acquisition function $\phi(M_{j,k}, i)$, where $M_{j,k} \in \mathcal{M}$ is the pairwise classifier between algorithms a_j and a_k, and ϕ returns the uncertainty of $M_{j,k}$ on instance i based on least confidence (LC). The pairs are then ranked based on their uncertainty scores, and the top q most informative ones are selected for labelling, where q is the query size. For each selected pair, the evaluation function EvaluateRuns is used to execute the algorithms on instance i up to time limit t, only if their runtimes have not been obtained from any previously executed pair, in which case the recorded results are reused to avoid redundant execution. After labelling, the selected pairs are removed from $\mathcal{U}$. If a pair is not a double-timeout, the observed performance is used to generate labels, which are added to $\mathcal{L}$. The pairwise models are retrained on the expanded labelled set, and the process continues until the stopping criterion is reached. The stopping criterion can be defined in various ways, such as reaching a target performance or when the labelling budget is exhausted.

Algorithm 1 Uncertainty-Based Selection

Input: Algorithm portfolio $\mathcal{A} = \{a_1, \ldots, a_n\}$, set of pairwise AS models $\mathcal{M} = \{M_{j,k} \mid 1 \leq j < k \leq n\}$, initial subset of runs $\mathcal{L}$, pool of unexecuted runs $\mathcal{U}$, uncertainty measure $\phi(M_{j,k}, i)$ returns the uncertainty of model $M_{j,k}$ on instance i, query size q, time limit t
while stopping criterion not satisfied **do**
    compute uncertainty $\phi(M_{j,k}, i)$ for all (i, a_j) and $(i, a_k) \in \mathcal{U}$
    select top q most uncertain pairs of runs
    for each selected (i, a_j, a_k) **do**
        EvaluateRuns(i, a_j, a_k, t)
        if not both (i, a_j) and (i, a_k) time out **then**
            extend $\mathcal{L}$ with (i, a_j) and (i, a_k)
        end if
        remove (i, a_j) and (i, a_k) from $\mathcal{U}$
    end for
end while

function EvaluateRuns(i, a_j, a_k, t)
    execute (i, a_j) and (i, a_k) up to t
end function
Output: $\mathcal{L}$

4.2 Dynamic Time Limit and Timeout Prediction as Frugality Mechanisms

To improve the frugality of selection, we propose two complementary frugality mechanisms: *DTL* and *TO*. To clearly discuss labelling costs in the pairwise classification setting, we define the following terminology: a *double-timeout pair of runs* refers to a case where both algorithms in the pair time out on the same instance, while a *single-timeout pair of runs* is when only one of them times out. These distinctions help clarify the sources of labelling cost and support the development of more effective frugal selection strategies.

4.2.1 Timeout Prediction. The first frugality mechanism, *TO*, enhances frugality in the query phase of active learning. It identifies runs that are potentially expensive and uninformative, as they provide no useful information for AS model training, while incurring a high cost. In pairwise classification, a clear example of such cases is a double-timeout pair of runs. These runs are highly expensive, as neither algorithm finishes within the time limit, and they provide no useful information for model training, since no performance comparison between the two algorithms can be established on that instance. In contrast, *single-timeout pair of runs*, where only one algorithm exceeds the time limit, remain valuable for training because they provide a label. In such cases, a clear performance difference can be established, as the algorithm that finishes is considered to perform better than the one that times out. These runs are particularly useful under penalty-based performance metrics such as PAR10, where the distinction between a completed run and a timeout run is explicitly reflected in the score, making them important for AS.

To obtain timeout information, we train additional binary classifiers that predict whether an algorithm will time out on a given instance. These timeout predictors are trained using the same subset of runs used to train the AS models, for consistency and avoiding additional labelling cost. During the query phase, the queried pair of runs is then filtered using these predictions: if both algorithms in a pair are predicted to time out, the pair is discarded before labelling, as it is likely to be a double-timeout pair of runs.

4.2.2 Dynamic Time Limit. The DTL strategy introduces an adaptive time limit mechanism to control how long queried runs are allowed to execute. If a run does not finish within the current time limit, its execution is paused. When at least one algorithm in a queried pair finishes within the limit, the relative performance of the algorithms can still be determined. Otherwise, both runs in the pair are returned to the pool, where they may be reconsidered in future iterations with an increased time limit if they remains among the most uncertain or informative candidates. The time limit is increased when no further improvement is observed in the AS performance on the validation set, showing that the current limit is too restrictive. This avoids costly uninformative runs by potentially stopping them before having used a large amount of resources, especially in single-timeout pair of runs where one run finishes quickly.

TO can be effectively integrated into the DTL strategy by training the predictor to estimate whether a given algorithm will time out for a given time limit, retraining it as the time limit changes. Although this procedure incurs a small computational cost, it is negligible relative to the savings obtained by avoiding expensive runs. As shown in Algorithm 2, the main difference from Algorithm 1 is that the evaluation function uses both DTL and TO: TO prevents the execution of potential double-timeout pairs of runs, while DTL limits the execution of the remaining runs to the current time limit; all other steps follow Algorithm 1 unchanged.

Algorithm 2 Frugality Mechanisms DTL & TO

Input: All inputs from Algorithm 1, timeout predictor $\Psi = \{\psi_j \mid 1 \leq j \leq n\}$, initial time limit τ, and the time limit increase $\Delta\tau$.
while stopping criterion not satisfied **do**
  compute uncertainty $\phi(M_{j,k}, i)$ for all (i, a_j) and $(i, a_k) \in \mathcal{U}$
  select top q most uncertain pairs of runs
  for each selected (i, a_j, a_k) **do**
    EvaluateRunsWithDTLTO($i, a_j, a_k, \tau, \psi_j, \psi_k$) ▷ Evaluation function with DTL and TO
    if not both (i, a_j) and (i, a_k) time out **then**
      extend $\mathcal{L}$ with (i, a_j) and (i, a_k)
    end if
    remove (i, a_j) and (i, a_k) from $\mathcal{U}$
  end for
  if AS validation performance does not improve **then**
    $\tau \leftarrow \tau + \Delta\tau$
  end if
end while

function EvaluateRunsWithDTLTO($i, a_j, a_k, \tau, \psi_j, \psi_k$)
  if $\psi_j(i) =$ timeout $\wedge\ \psi_k(i) =$ timeout **then**
    return ▷ skip execution for this pair
  end if
  execute (i, a_j) and (i, a_k) up to τ
end function
Output: $\mathcal{L}$

4.3 Cost-Aware Selection Metrics

To further improve frugality, we integrate cost-awareness into the selection process through two metrics: predicted cost (PC) and predicted runtime difference (PD). For a given pair of runs, PC estimates the total computational

cost of executing both algorithms on a given instance. PD measures the expected performance gap between the two algorithms, with an emphasis on instances where an incorrect choice would incur a significant penalty.

To derive these metrics, we use runtime regression models to predict the runtime of an algorithm on a given instance. Similar to the timeout predictors, the regressors are trained using the same subset of runs that are used for the AS models. Note that while the runtime predictions could also be used to predict timeouts, we keep our separate timeout predictors as predicting runtime is more challenging (Hutter, Xu, et al. 2014) than simply whether a run will time out or not.

4.3.1 Predicted Cost. PC allows to prioritise pairs of runs which are inexpensive to label, but also informative. Under a static time limit, the PC is the full expected runtime for both algorithms. Under a dynamic time limit, where runs may be paused and reselected across iterations, PC accounts for partial computation that has already been performed. Instead of considering the full expected cost, it only captures the additional predicted cost required to complete the run, where smaller PC values correspond to less expensive and thus more desirable pairs of runs.

Formally, let $\hat{R}(a, i)$ denote the predicted runtime of algorithm a on instance i, and let $R_{\text{el}}(a, i)$ represent the elapsed runtime from a previously-paused run. The predicted cost for a single algorithm is defined as:

$$\text{PC}(a, i) = \max\left(\hat{R}(a, i) - R_{\text{el}}(a, i),\ 0\right)$$

We lower bound the cost at zero to account for cases where the observed runtime exceeds the predicted value due to noise or estimation error. The total predicted cost for a pair of runs is their sum.

4.3.2 Predicted Runtime Difference. In AS, we care most about the problem instances where algorithms exhibit very different performances – one might solve the instance in fractions of a second, while another times out after hours. The PD metric captures this behaviour by prioritising runs that help identify such cases and ensuring that they are considered during training. It is defined as:

$$\text{PD}(a_i, a_j, i) = \left|\hat{R}(a_i, i) - \hat{R}(a_j, i)\right|$$

Pairs of runs with large PD values denote where the most consequential decisions have to be made, while low PD values indicate that the decision does not matter much – even if the worse algorithm is selected, there is only a small performance penalty.

4.4 Cost-Aware Multi-Objective Run Selection

Individual metrics such as uncertainty, PC, and PD each provide valuable insights for guiding run selection. To leverage their complementary strengths, we propose two multi-objective selection strategies that combine all three metrics: a lexicographic ordering approach and a Pareto-based selection strategy. These strategies allow multiple forms of information to be jointly considered without manually specifying arbitrary trade-offs between the metrics.

4.4.1 Lexicographic Ordering. This strategy ranks candidate runs using all three metrics according to a predefined priority order. We explore all six possible orders of uncertainty, PC, and PD. Runs are first ranked by the primary metric, with ties resolved using the secondary metric, and further ties resolved using the tertiary metric. Similar to single-metric selection, a fixed number of top-ranked runs are selected. Among these metrics, we expect uncertainty to have low variation, mainly due to double timeout pairs of runs frequently producing uncertainty values of 0.5.

4.4.2 Pareto Based Selection. In contrast, the Pareto-based selection strategy identifies runs that lie on the *Pareto set*, which represents the set of non-dominated runs across multiple metrics. A run is considered non-dominated if no other run performs better in all metrics simultaneously, and it is better than all other runs according to at least one metric. Similar to the other selection strategies, we select a fixed number of non-dominated runs, which are obtained by randomly sampling from the Pareto set. If the size of the Pareto set is smaller than the fixed number of required runs, the process is repeated iteratively by removing already selected runs and recomputing the Pareto set until the desired number of runs is obtained.

To explore the trade-offs between cost and informativeness from different perspectives, we apply this strategy in both two-objective and three-objective settings. Specifically, we compute the full Pareto set using all three metrics, as well as all Pareto sets formed by all possible combinations of two metrics.

5 Experimental Setup

Building on the methodology described in the previous sections, this section presents the experimental setup used to evaluate the effectiveness and efficiency of the proposed run selection strategies within the FAS framework in the pairwise classification-based AS setting.

5.1 Datasets

To evaluate the effectiveness of the proposed strategies, we conduct experiments on 16 scenarios from ASLib. These benchmarks cover a diverse range of combinatorial problem-solving domains, including Answer Set Programming (ASP), Bayesian Network Structure Learning (BNSL), Constraint Programming (CP), Boolean Satisfiability (SAT), and Quantified Boolean Formula (QBF). The datasets exhibit substantial heterogeneity: algorithm portfolio sizes range from 2 to 31 algorithms with 22 to 155 instance features and between 296 and 4642 problem instances.

Table 1 summarises the datasets used in this study, including the total computational cost (in CPU hours) for running all algorithms in the portfolio on all problem instances.

5.2 Algorithm Selection Setup

We use random forests for all predictive models, and their performance is evaluated using 10-fold cross-validation. We assess performance using PAR10, and average the results over five random seeds and ten folds to reduce variability from randomness in training and partitioning. Within each training fold, we split 10% of the run in the train set into a validation set to determine whether the DTL should be increased. To make results comparable across scenarios we normalise the PAR10 scores by the performance gap between the Single Best Solver and the Virtual Best Solver, following (Lindauer, van Rijn, et al. 2019):

$$\widehat{PAR10}_{AS} = \frac{PAR10_{AS} - PAR10_{VBS}}{PAR10_{SBS} - PAR10_{VBS}}$$

Our run selection strategies begin with an initial subset of runs, obtained by selecting 20 problem instances and running all algorithms on these instances to collect complete runs. In each iteration, we select 1% of the total number of runs in the ASlib scenario. The initial time limit used for DTL is set to 100 seconds and is increased by 100 seconds whenever no improvement in AS performance is observed on the validation set. A preliminary sensitivity analysis on the DTL parameters revealed that lower time limits yield better results; however, they also require finer-grained execution steps, resulting in a larger number of iterations. We therefore adopt the 100*s* initial limit with 100*s* increments as a practical and stable baseline.

We evaluate four families of selection strategies: cost-unaware strategies, including random selection and uncertainty-based selection, and cost-aware strategies, including lexicographical ordering and Pareto-based selection. For uncertainty-based selection, we use the pairwise classification variant adapted to the AS setting.

Table 1. Summary of ASlib benchmark scenarios used in our experiments. Since the proportion of double timeouts is high in most of these datasets, this observation motivates the use of frugality mechanisms and cost-aware strategies to mitigate their impact on labelling cost.

Scenario	Instances	Algorithms	Features	Total Labelling Cost	Double Timeouts
ASP-POTASSCO	1294	11	138	2,085h	9.42%
BNSL-2016	1179	8	86	3,272h	11.44%
CPMP-2015	527	4	22	682h	11.67%
CSP-2010	2024	2	86	435h	0.00%
CSP-MZN-2013	4642	11	155	26,263h	44.99%
MAXSAT12-PMS	876	6	37	1,472h	11.34%
MAXSAT15-PMS-INDU	601	29	37	7,582h	22.01%
MAXSAT19-UCMS	572	7	54	545h	7.97%
MAXSAT-WPMS-2016	630	18	37	5,113h	30.45%
QBF-2011	1368	5	46	2,352h	22.44%
QBF-2014	1254	14	46	6,510h	30.17%
QBF-2016	825	24	46	5,832h	19.77%
SAT11-HAND	296	15	115	1,851h	32.73%
SAT11-RAND	600	9	115	1,829h	19.35%
SAT12-HAND	767	31	115	9,069h	37.76%
SAT12-INDU	1167	31	115	12,272h	20.75%

This corresponds to the AL framework proposed by Volpato and Song (Volpato and Song 2019), as their approach applies standard active learning without any frugality mechanisms, making it equivalent to our cost-unaware uncertainty-based baseline. Our work, therefore, directly extends this setting by introducing cost-aware selection strategies.

To evaluate the efficiency of the proposed strategies, we compare their performance against a passive learning baseline. In the context of frugality, we assess labelling cost by analysing when each selection strategy reaches the AS performance of passive learning. To determine whether the performance differences between strategies are statistically significant, we follow the literature (Demšar 2006) and apply the Friedman test (Friedman 1937) to compare multiple strategies across multiple datasets. When significant differences are detected ($p < 0.05$), we perform the Nemenyi post-hoc test (Nemenyi 1963) to identify which pairs of strategies differ significantly based on their average ranks. We visualise the statistical comparisons using critical difference (CD) diagrams, generated with the `Orange3` toolkit (Demšar et al. 2013). In the CD diagrams, methods are ranked based on frugality, where a lower rank (positioned further to the left) indicates greater efficiency in achieving the desired AS performance with lower labelling cost. Strategies connected by horizontal bars are not significantly different, showing that their performance cannot be statistically distinguished under the Nemenyi test.

All source code, data, and experimental results are available at https://doi.org/10.5281/zenodo.20095993.

6 Evaluation and Analysis of Selection Strategies

We focus on labelling cost as the primary evaluation metric rather than the number of runs, due to the non-uniform cost nature of algorithm selection. For example, in the ASP-POTASSCO scenario, *PARETO (U - PC) (DTL)* executes 8,698 runs compared to 2,371 runs for uncertainty-based selection, but this corresponds to about 100 CPU hours of labelling cost for Pareto versus 626 CPU hours for uncertainty-based selection. This shows that

the number of runs alone is a misleading measure of frugality. We focus on labelling cost and give details of the reductions in the number of runs in Section F.

The cost of training and retraining the AS models, runtime prediction models, and timeout prediction models represents only about 0.024% of the total labelling cost, and we therefore do not explicitly consider it here.

Our key findings are as follows:

- PARETO (U - PC) (DTL) is the most frugal method overall, followed by the cost-unaware UNCERTAINTY (DTL + TO).
- Uncertainty-based selection is not effective in a setting like ours where labelling costs are highly non-uniform.
- TO enhances uncertainty-based selection by filtering out expensive and uninformative runs. DTL further improves frugality by limiting costly evaluations; the combination (DTL + TO) provides the largest cost savings.
- Multi-objective selection methods, particularly PARETO (U - PC), are the most frugal strategies.
- The RANDOM (DTL) strategy emerges as a surprisingly strong baseline.
- Lexicographic strategies that prioritise PC perform reasonably well, but show limited frugality improvements under DTL compared to Pareto-based methods.
- Strategies without DTL, particularly those relying on PD, perform the worst.
- PC does most to reduce labelling cost in Pareto-based selection, followed by U, while PD increases labelling cost on average.

6.1 Frugality-Based Ranking of Selection Strategies

We compare all strategies in terms of labelling cost when the AS performance matches that of passive learning. The Friedman test indicates statistically significant differences across all comparisons. We show critical difference (CD) diagrams based on Nemenyi post-hoc tests to compare all strategies.

The overall comparison in Figure 1 shows that strategies without frugality mechanisms are not competitive in terms of labelling cost. Adding TO and DTL improves performance, with *UNCERTAINTY (DTL + TO)* emerging as the most frugal cost-unaware strategy. TO hurts random selection however, with *RANDOM (TO)* being the worst-ranked variant in this group. DTL substantially improves the frugality of random selection, making *RANDOM (DTL)* the strongest variant within the random family.

Among the cost-aware strategies, the Pareto-based variants achieve the highest level of frugality across all evaluated methods. In particular, *PARETO (U – PC) (DTL)*, *PARETO (U – PD – PC) (DTL)*, and *PARETO (U – PD) (DTL)* obtain top ranks. Moreover, *PARETO (U – PC)* is the best-performing approach among all strategies without TO or DTL. Interestingly, the lexicographical configurations *LEXICAL (PC → U → PD)* and *LEXICAL (PC → PD → U)* rank directly after *PARETO (U – PC)*, demonstrating the competitiveness of lexicographical ordering in the absence of frugality mechanisms. However, when DTL is applied, their ranks improve less than those of most other strategies, indicating that lexicographical ordering benefits only marginally from the frugality mechanism; notably, even *RANDOM (DTL)* outperforms all lexicographical configurations under DTL.

Based on these results, we select *PARETO (U - PC)* as the representative Pareto-based strategy and *LEXICAL (PC → PD → U)* as the representative lexicographical strategy for further analysis, to avoid overwhelming the reader with detailed results for all 28 strategies. Note that *LEXICAL (PC → U → PD)* achieves identical ranks with no statistically significant difference, and therefore either strategy can be used as a representative. Additional CD diagrams demonstrating performance trends at 75%, 50%, and 25% of passive learning performance are provided in Section B for comprehensive analysis.

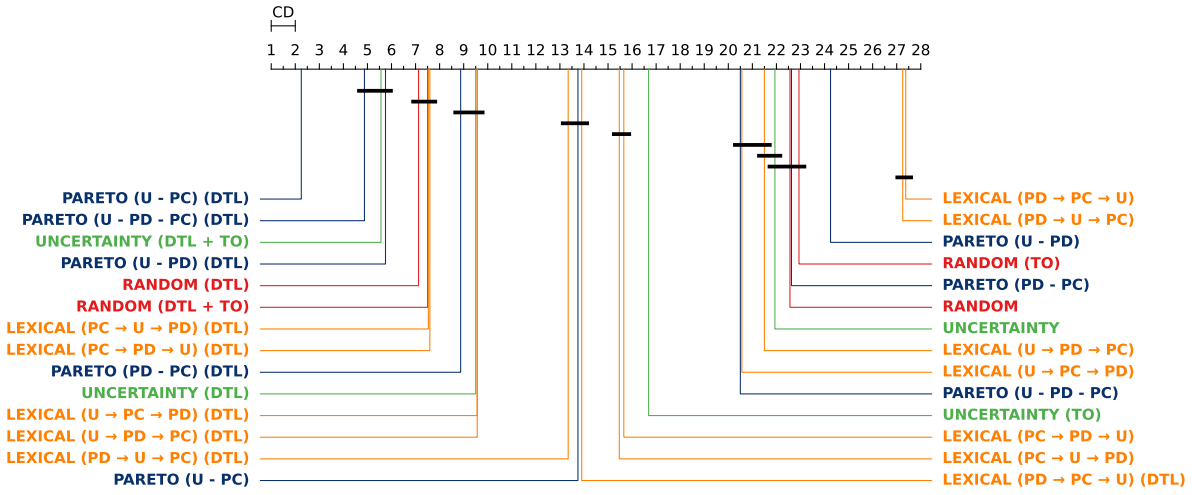

Fig. 1. CD diagram presenting the average ranks of all evaluated selection strategies across 16 benchmark datasets. Lower ranks (further to the left) indicate better frugality performance. Strategies that are not statistically significantly different, according to the Nemenyi test, are connected by horizontal bars. Colours distinguish strategy families: blue for Pareto-based methods, orange for lexicographic methods, green for uncertainty-based methods, and red for random baselines. Strategies using frugality mechanisms, such as DTL and TO, are annotated accordingly. The best overall frugality performance, with statistical significance, is achieved by *PARETO (U – PC) (DTL)*. DTL consistently improves frugality across all strategy families. Conversely, strategies without DTL or TO and those relying primarily on PD generally exhibit poorer frugality.

6.2 Detailed Quantitative Analysis of Frugality

To further evaluate the effectiveness of the selected methods, we now quantitatively analyse their frugality by examining labelling cost across datasets in Figure 2. Among all evaluated methods, the cost-aware strategy *PARETO (U – PC) (DTL)* demonstrates the highest frugality, requiring only 2.7% to 19.1% of the full labelling cost to achieve passive learning performance.

For cost-unaware strategies, both random and uncertainty-based selection require approximately 50% of the labelling cost to match passive learning performance without TO or DTL; substantially less with those techniques. Specifically, uncertainty-based selection incurs a labelling cost ranging from 20.3% to 82.7% across datasets and introducing TO moderately reduces this to 22.2%–65.8%. Applying DTL alone achieves better frugality, lowering the labelling cost to 6.8%–39.3%. The most frugal configuration within the uncertainty-based family is the combined DTL + TO variant, which further reduces the labelling cost to just 3.1%–33.1%. For random selection, the labelling cost ranges from 25.5% to 87.1%. When TO is applied, the average labelling cost changes to between 26.7% and 86.8%, indicating that TO is not effective in improving frugality for random selection. The most frugal configuration within the random selection family is *RANDOM (DTL)*, which substantially reduces the labelling cost to between 7.1% and 30.2%.

For lexicographical ordering, the strategy *LEXICAL (PC → PD → U)* incurs a labelling cost ranging from 14.3% to 63.8%. When DTL is applied, the cost reduces to 3.1%–28.6%. However, the gains are comparatively smaller than in other families, suggesting that DTL offers limited additional benefit for lexicographical ordering. The detailed labelling costs for each strategy are provided in Section C, and the actual labelling time (in hours) can be

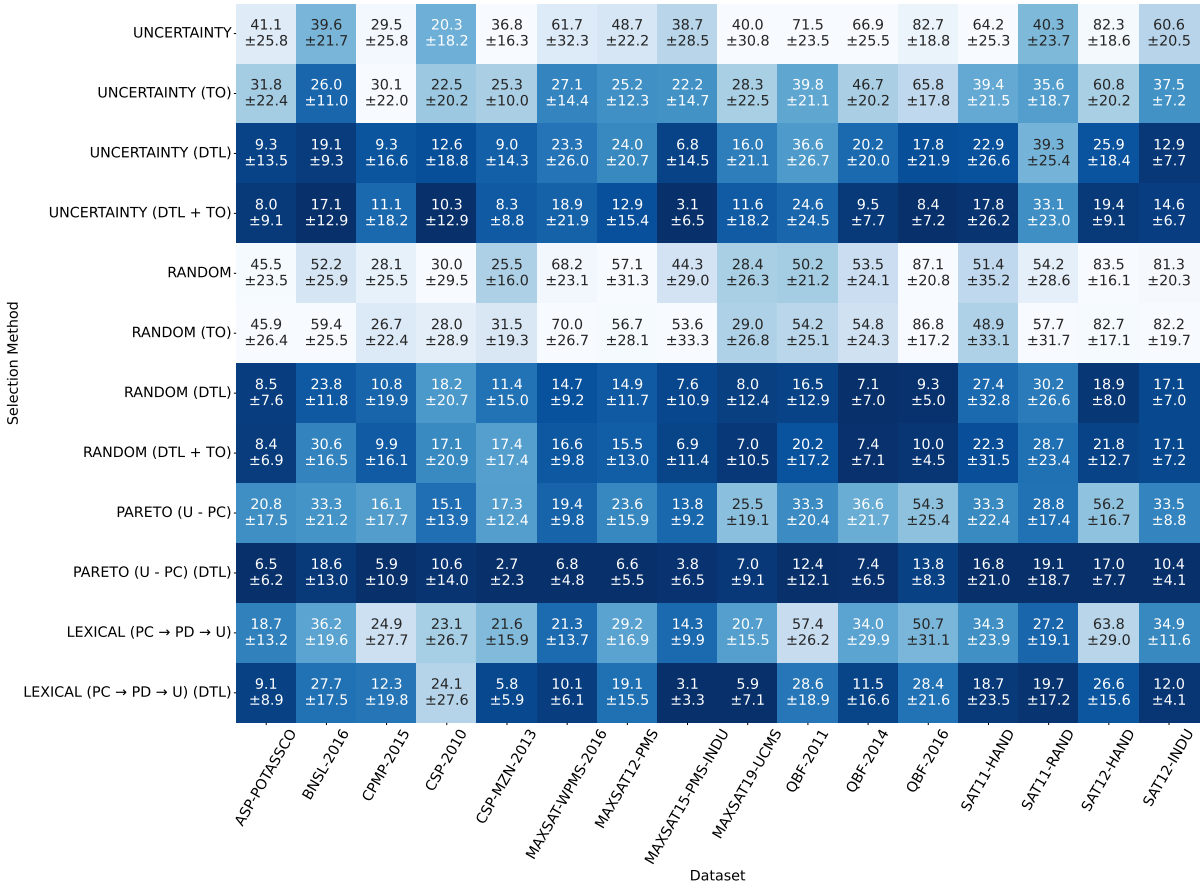

Fig. 2. Average labelling cost (in %) and standard deviation across 16 ASlib benchmark datasets for a selected subset of representative strategies from each strategy family. Each row corresponds to a specific selection method, including the representative variants of Pareto-based, Lexicographic, Uncertainty-based, and Random strategies, with and without frugality mechanisms such as DTL and TO. Darker blue regions indicate lower labelling costs, while lighter blue regions correspond to higher labelling costs. PARETO (U – PC) (DTL) achieves the lowest overall labelling cost, while UNCERTAINTY (DTL + TO) is the most effective cost-unaware strategy.

found in Section D. The actual runtime incurred by executing the selected algorithms on the test instances is reported in Section E to assess the practical applicability of each selection strategy.

6.3 Performance for Different Labelling Budgets

Figure 3 illustrates how different selection strategies achieve AS performance relative to passive learning as labelling cost increases. In this figure, 100% performance corresponds to the level achieved by passive learning. To highlight the impact of cost-awareness and frugality, we compare the best cost-aware strategy against the standard cost-unaware strategies. Across most benchmarks, *PARETO (U - PC) (DTL)* demonstrates strong early gains and reaches high AS performance with relatively low labelling cost, reflecting its ability to consistently prioritise informative and inexpensive runs.

In comparison to the Pareto-based strategy, both uncertainty-based selection and random selection show slower performance gains as the labelling budget increases. Uncertainty-based and random selection show similar overall trends, with neither consistently outperforming the other. In some datasets, uncertainty-based selection improves more with higher budgets, while in others random selection performs better, or both strategies produce comparable results. For a direct comparison of the best-performing variant within each family, see Section G.

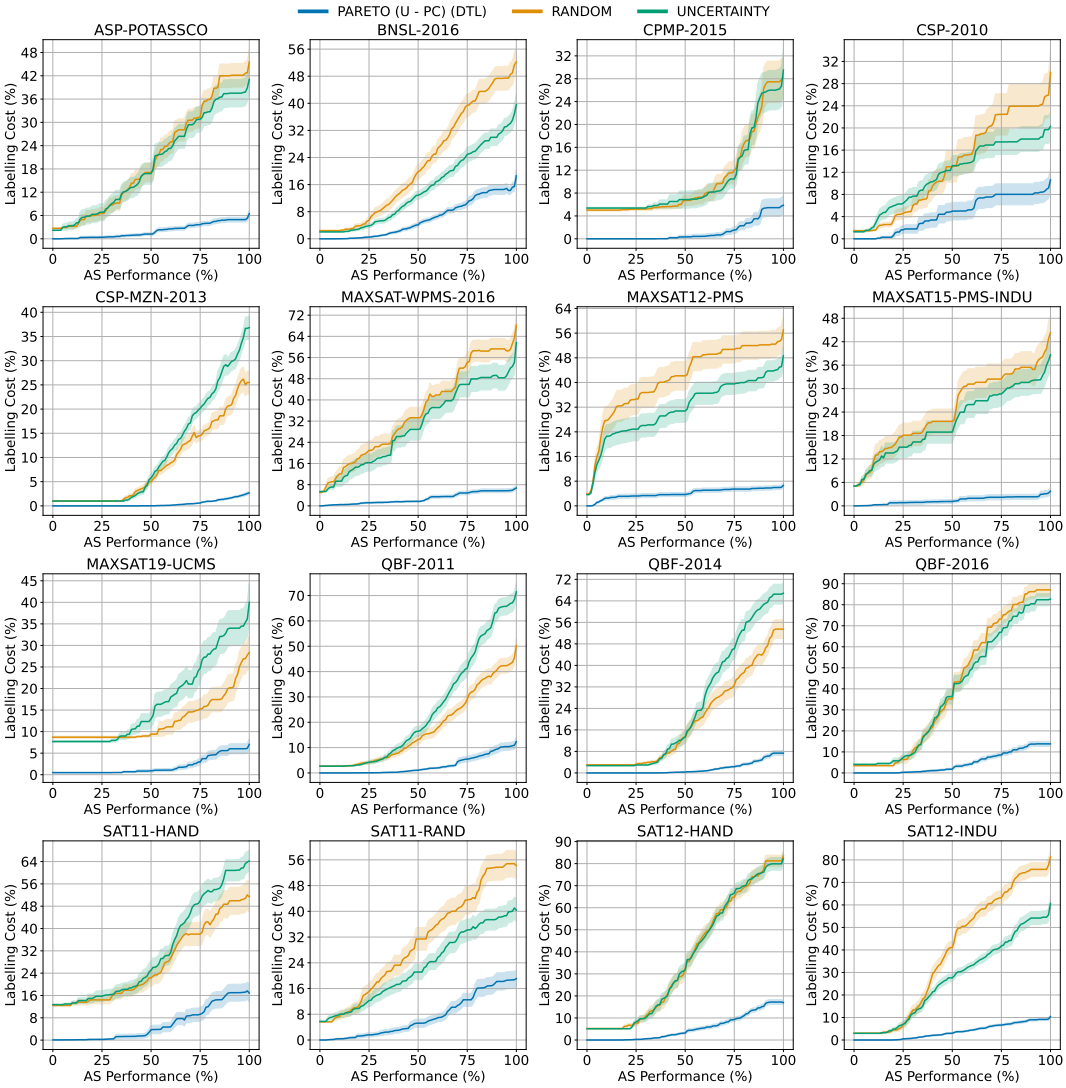

Fig. 3. Labelling cost over AS performance across benchmarks for three representative strategies. Each line shows the labelling cost required to achieve different levels of AS performance, where 100% corresponds to the performance of passive learning. PARETO (U - PC) (DTL) consistently achieves robust AS performance with minimal labelling effort and represents the top-performing strategy across all datasets. In contrast, there is no consistent winner between uncertainty-based selection and random; their relative performance varies across datasets, with neither demonstrating a clear overall advantage.

6.4 Ablation Analysis of Dynamic Time Limits and Selection Metrics

To analyse the individual effects of DTL and the selection metrics on labelling cost, we perform an ablation study that isolates their contributions.

Figure 4 presents the ablation analysis for DTL. We show the mean rank across strategies belonging to the same family. The results show that DTL substantially improves the frugality of all selection strategy families, with a clear rank separation compared to their non-DTL counterparts. *PARETO (DTL)* achieves the best overall rank, confirming it as the most frugal option. Interestingly, *RANDOM (DTL)* ranks second despite *RANDOM* ranking last, though there is no statistically significant difference between the two, suggesting that DTL alone is a strong driver of frugality regardless of the selection strategy. In contrast, the lexicographical ordering strategies, which are second best in the non-DTL setting, become the worst-performing family under DTL. Further details of this analysis are provided in Section H.

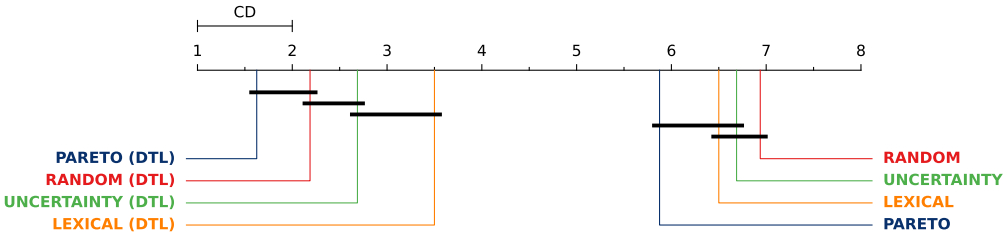

Fig. 4. CD diagram for DTL comparing mean ranks for cost-aware strategy families (Pareto and Lexicographical ordering) with random and uncertainty-based selection, ranked by overall frugality. All DTL-enhanced strategies significantly outperform their non-DTL counterparts, with *PARETO (DTL)* achieving the best overall rank.

Figure 5 shows the effect of different metric combinations on the frugality of Pareto-based strategies. Among all Pareto variants, *PARETO (U - PC) (DTL)*, which combines uncertainty and predicted cost, achieves the best average rank. Adding PD to this results in the second-best configuration U - PD - PC, while removing PC leads to the third-ranked variant U - PD. Among the DTL-enhanced strategies, the PD - PC combination performs worst. A similar trend is observed for the non-DTL variants, where the combination of uncertainty and predicted cost again yields the most frugal behaviour, while configurations relying primarily on PD perform the worst. Overall, these results confirm that configurations based on PC have a stronger positive impact on frugality, and that combining PC with uncertainty results in the most frugal selection strategy. Additional details on the ablation study can be found in Section I.

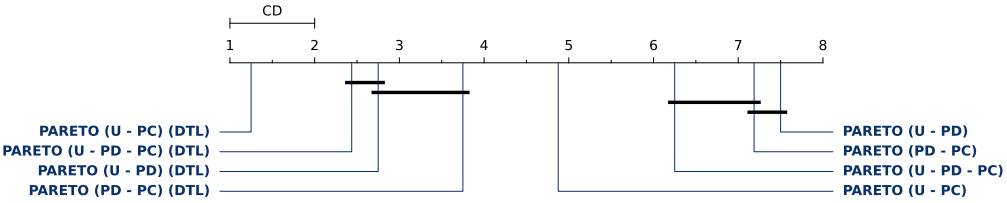

Fig. 5. CD diagram for selection metrics for Pareto-based strategies with and without DTL, ranked by frugality. The U-PC configuration is the most frugal among both DTL and non-DTL variants.

7 Conclusions and Future Work

In this work, we introduced and evaluated a range of strategies to reduce labelling cost in algorithm selection for combinatorial search and optimisation. These strategies form the foundation of *Frugal Algorithm Selection*, an active learning framework designed to minimise labelling effort while preserving algorithm selection performance. Our approach extends traditional active learning by incorporating frugality mechanisms to mitigate bias in uncertainty-based selection, and by integrating selection metrics that prioritise runs that are both informative and inexpensive.

We assessed the effectiveness of these strategies across 16 ASlib scenarios. Among all evaluated methods, the cost-aware strategy *PARETO (U – PC) (DTL)* achieved the highest level of frugality, reducing labelling cost by approximately 90% on average. This highlights the advantage of integrating cost-awareness in multi-objective selection. DTL consistently improved frugality across all strategies, particularly when combined with cost-aware metrics. Even cost-unaware methods, such as uncertainty-based selection, benefited substantially when combined with DTL and TO.

Overall, our findings demonstrate that integrating DTL and cost awareness into active learning substantially reduces labelling cost without compromising algorithm selection performance. In resource-intensive domains such as combinatorial search, this enables the training of state-of-the-art algorithm selection systems at significantly lower cost, making this powerful methodology more widely accessible. From a practical perspective, we suggest that Frugal Algorithm Selection can be deployed in real-world settings by explicitly defining a labelling budget; a small percentage of the total labelling cost, for which an upper bound can be obtained easily by assuming that all algorithm runs time out, is sufficient to train well-performing algorithm selection systems.

As future work, we plan to investigate algorithm pre-selection mechanisms to further reduce labelling cost, along with more advanced cost-aware strategies that incorporate algorithm features. We also aim to improve the predictive components by developing unified runtime models for both estimation and timeout prediction, and by exploring early stopping strategies to avoid unnecessary labelling. Furthermore, we intend to explore the effectiveness of the proposed Frugal Algorithm Selection framework in other learning paradigms, such as AS based on multi-class classification. Since the labelling of each training data point in such a setting requires executing *all* algorithms on a given instance, we expect a decrease in the flexibility and effectiveness of the frugality mechanisms. Finally, we plan to extend the framework to a wider range of AS scenarios, particularly in combinatorial optimisation, to assess its generalisability and scalability.

Acknowledgments

This research was supported by computing resources provided by the Cirrus UK National Tier-2 HPC Service at EPCC[1] and the Advanced Research Computing Center (ARCC)[2] at the University of Wyoming. We gratefully acknowledge their support in enabling large-scale experimentation essential to this study. This work was funded by the Engineering and Physical Sciences Research Council (EPSRC, EP/V027182/1), the Royal Society University Research Fellowship. Ian Miguel is funded by EPSRC grant EP/V027182/1.

References

R. Amadini, M. Gabbrielli, and J. Mauro. 2014. "SUNNY: a lazy portfolio approach for constraint solving." *Theory and Practice of Logic Programming*, 14, 4-5, 509–524.

D. Angluin. Apr. 1988. "Queries and Concept Learning." *Machine Learning*, 2, 4, (Apr. 1988), 319–342. doi:10.1023/A:1022821128753.

C. Ansótegui, J. Gabas, Y. Malitsky, and M. Sellmann. 2016. "MaxSAT by improved instance-specific algorithm configuration." *Artificial Intelligence*, 235, 26–39.

[1]https://www.cirrus.ac.uk

[2]https://www.uwyo.edu/arcc/

C. Ansótegui, M. Sellmann, and K. Tierney. 2009. "A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms." In: *Principles and Practice of Constraint Programming - CP 2009*. Ed. by I. P. Gent. Springer Berlin Heidelberg, Berlin, Heidelberg, 142–157. ISBN: 978-3-642-04244-7.

A. Arbelaez, Y. Hamadi, and M. Sebag. Oct. 2010. " Continuous Search in Constraint Programming." In: *2010 22nd International Conference on Tools with Artificial Intelligence (ICTAI 2010)*. Vol. 1. IEEE Computer Society, Los Alamitos, CA, USA, (Oct. 2010), 53–60. doi:10.1109/ICTAI.2010.17.

L. Atlas, D. Cohn, R. Ladner, M. A. El-Sharkawi, R. J. Marks, M. E. Aggoune, and D. C. Park. 1989. "Training connectionist networks with queries and selective sampling." In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems* (NIPS'89). MIT Press, Cambridge, MA, USA, 566–573.

B. Bischl et al.. 2016. "ASlib: A benchmark library for algorithm selection." *Artificial Intelligence*, 237, 41–58. doi:https://doi.org/10.1016/j.artint.2016.04.003.

D. Cohn, L. Atlas, and R. Ladner. May 1994. "Improving generalization with active learning." *Machine Learning*, 15, 2, (May 1994), 201–221.

M. Collautti, Y. Malitsky, D. Mehta, and B. O'Sullivan. 2013. "SNNAP: Solver-Based Nearest Neighbor for Algorithm Portfolios." In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Železný. Springer Berlin Heidelberg, Berlin, Heidelberg, 435–450. ISBN: 978-3-642-40994-3.

N. V. Cuong, W. S. Lee, N. Ye, K. M. A. Chai, and H. L. Chieu. 2013. "Active learning for probabilistic hypotheses using the maximum Gibbs error criterion." In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1* (NIPS'13). Curran Associates Inc., Lake Tahoe, Nevada, 1457–1465.

J. Demšar. 2006. "Statistical Comparisons of Classifiers over Multiple Data Sets." *Journal of Machine Learning Research*, 7, 1, 1–30. http://jmlr.org/papers/v7/demsar06a.html.

J. Demšar et al.. 2013. "Orange: Data Mining Toolbox in Python." *Journal of Machine Learning Research*, 14, 2349–2353. http://jmlr.org/papers/v14/demsar13a.html.

M. Friedman. 1937. "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. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1937.10503522. doi:10.1080/01621459.1937.10503522.

C. P. Gomes and B. Selman. 2001. "Algorithm Portfolios." *Artificial Intelligence*, 126, 1-2, 43–62. doi:http://dx.doi.org/10.1016/S0004-3702(00)00081-3.

H. Hoos, M. Lindauer, and T. Schaub. 2014. "claspfolio 2: Advances in algorithm selection for answer set programming." *Theory and Practice of Logic Programming*, 14, 4-5, 569–585.

S.-J. Huang, R. Jin, and Z.-H. Zhou. 2014. "Active Learning by Querying Informative and Representative Examples." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36, 10, 1936–1949. doi:10.1109/TPAMI.2014.2307881.

B. A. Huberman, R. M. Lukose, and T. Hogg. 1997. "An Economics Approach to Hard Computational Problems." *Science*, 275, 5296, 51–54. doi:10.1126/science.275.5296.51.

F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Sept. 2009. "ParamILS: an automatic algorithm configuration framework." *J. Artif. Int. Res.*, 36, 1, (Sept. 2009), 267–306.

F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. 2014. "Algorithm runtime prediction: Methods & evaluation." *Artificial Intelligence*, 206, 79–111. doi:https://doi.org/10.1016/j.artint.2013.10.003.

K. Jamieson and A. Talwalkar. May 2016. "Non-stochastic Best Arm Identification and Hyperparameter Optimization." In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (Proceedings of Machine Learning Research). Ed. by A. Gretton and C. C. Robert. Vol. 51. PMLR, Cadiz, Spain, (May 2016), 240–248. https://proceedings.mlr.press/v51/jamieson16.html.

S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. 2010. "ISAC –Instance-Specific Algorithm Configuration." In: *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. IOS Press, NLD, 751–756. ISBN: 9781607506058.

H. Kashgarani and L. Kotthoff. 2023. "Automatic Parallel Portfolio Selection." In: *ECAI* (Frontiers in Artificial Intelligence and Applications). Ed. by K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, and R. Radulescu. Vol. 372. IOS Press, Kraków, Poland, 1215–1222. ISBN: 978-1-64368-437-6. http://dblp.uni-trier.de/db/conf/ecai/ecai2023.html#KashgaraniK23.

H. Kashgarani and L. Kotthoff. Feb. 2021. "Is Algorithm Selection Worth It? Comparing Selecting Single Algorithms and Parallel Execution." In: *AAAI Workshop on Meta-Learning and MetaDL Challenge* (Proceedings of Machine Learning Research). Ed. by I. Guyon, J. N. van Rijn, S. Treguer, and J. Vanschoren. Vol. 140. PMLR, Virtual Conference (AAAI 2021, originally planned for Vancouver, Canada), (Feb. 2021), 58–64. https://proceedings.mlr.press/v140/kashgarani21a.html.

K. Kaulen, M. König, and H. H. Hoos. 2025. "Dynamic algorithm termination for branch-and-bound-based neural network verification." In: *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence* (AAAI'25/IAAI'25/EAAI'25) Article 3048. AAAI Press, Philadelphia, Pennsylvania, 9 pages. ISBN: 978-1-57735-897-8. doi:10.1609/aaai.v39i26.34946.

P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. 2019. "Automated algorithm selection: Survey and perspectives." *Evolutionary computation*, 27, 1, 3–45.

P. Kerschke, L. Kotthoff, J. Bossek, H. H. Hoos, and H. Trautmann. 2018. "Leveraging TSP solver complementarity through machine learning." *Evolutionary computation*, 26, 4, 597–620.

L. Kotthoff. 2016. "Algorithm Selection for Combinatorial Search Problems: A Survey." In: *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach*. Springer International Publishing, Cham, 149–190. ISBN: 978-3-319-50137-6. doi:10.1007/978-3-319-50137-6_7.

D. D. Lewis and J. Catlett. 1994. "Heterogeneous Uncertainty Sampling for Supervised Learning." In: *Machine Learning Proceedings 1994*. Ed. by W. W. Cohen and H. Hirsh. Morgan Kaufmann, San Francisco (CA), 148–156. ISBN: 978-1-55860-335-6. doi:https://doi.org/10.1016/B978-1-55860-335-6.50026-X.

D. D. Lewis and W. A. Gale. 1994. "A Sequential Algorithm for Training Text Classifiers." In: *SIGIR '94*. Ed. by B. W. Croft and C. J. van Rijsbergen. Springer London, London, 3–12. ISBN: 978-1-4471-2099-5.

L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar. Jan. 2017. "Hyperband: a novel bandit-based approach to hyperparameter optimization." *J. Mach. Learn. Res.*, 18, 1, (Jan. 2017), 6765–6816.

M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. Jan. 2022. "SMAC3: a versatile Bayesian optimization package for hyperparameter optimization." *J. Mach. Learn. Res.*, 23, 1, Article 54, (Jan. 2022), 9 pages.

M. Lindauer, H. Hoos, K. Leyton-Brown, and T. Schaub. 2017. "Automatic construction of parallel portfolios via algorithm configuration." *Artificial Intelligence*, 244, 272–290. Combining Constraint Solving with Mining and Learning. doi:https://doi.org/10.1016/j.artint.2016.05.004.

M. Lindauer, H. H. Hoos, F. Hutter, and T. Schaub. 2015. "Autofolio: An automatically configured algorithm selector." *Journal of Artificial Intelligence Research*, 53, 745–778.

M. Lindauer, J. N. van Rijn, and L. Kotthoff. 2019. "The algorithm selection competitions 2015 and 2017." *Artificial Intelligence*, 272, 86–100. doi:https://doi.org/10.1016/j.artint.2018.10.004.

T. Liu, R. Amadini, M. Gabbrielli, and J. Mauro. 2021. "sunny-as2: Enhancing SUNNY for Algorithm Selection." *Journal of Artificial Intelligence Research*, 72, 329–376.

M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. 2016. "The irace package: Iterated racing for automatic algorithm configuration." *Operations Research Perspectives*, 3, 43–58. doi:https://doi.org/10.1016/j.orp.2016.09.002.

V. Margraf, T. Koerner, A. Tornede, and M. Wever. Aug. 2025. "RunAndSchedule2Survive: Algorithm Scheduling Based on Run2Survive." *ACM Trans. Evol. Learn. Optim.*, 5, 3, Article 21, (Aug. 2025), 17 pages. doi:10.1145/3737705.

M. Mısır and M. Sebag. 2017. "Alors: An algorithm recommender system." *Artificial Intelligence*, 244, 291–314. Combining Constraint Solving with Mining and Learning. doi:https://doi.org/10.1016/j.artint.2016.12.001.

D. Müller, M. G. Müller, D. Kress, and E. Pesch. 2022. "An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning." *European Journal of Operational Research*, 302, 3, 874–891. doi:https://doi.org/10.1016/j.ejor.2022.01.034.

P. B. Nemenyi. 1963. "Distribution-free multiple comparisons." Ph.D. Dissertation. Princeton University.

V.-L. Nguyen, M. H. Shaker, and E. Hüllermeier. 2022. "How to measure uncertainty in uncertainty sampling for active learning." *Machine Learning*, 111, 1, 89–122. doi:10.1007/s10994-021-06003-9.

E. O'Mahony, E. Hebrard, A. Holland, C. Nugent, and B. O'Sullivan. 2008. "Using case-based reasoning in an algorithm portfolio for constraint solving." In: *Irish conference on artificial intelligence and cognitive science*. University of Cork, Ireland, 210–216.

M. Park and J. W. Pillow. 2012. "Bayesian active learning with localized priors for fast receptive field characterization." In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2* (NIPS'12). Curran Associates Inc., Lake Tahoe, Nevada, 2348–2356.

J. R. Rice. 1976. "The Algorithm Selection Problem**This work was partially supported by the National Science Foundation through Grant GP-32940X. This chapter was presented as the George E. Forsythe Memorial Lecture at the Computer Science Conference, February 19, 1975, Washington, D. C." In: *Advances in computers*. Advances in Computers. Vol. 15. Ed. by M. Rubinoff and M. C. Yovits. Elsevier, United States, 65–118. doi:https://doi.org/10.1016/S0065-2458(08)60520-3.

M. Rizzini, C. Fawcett, M. Vallati, A. Gerevini, and H. Hoos. Jan. 2016. "Portfolio Methods for Optimal Planning: An Empirical Analysis." English. In: *2015 IEEE 27th International Conference on Tools with Artificial Intelligence*. 2015 IEEE 27th International Conference on Tools with Artificial Intelligence, ICTAI ; Conference date: 09-11-2015 Through 11-11-2015. IEEE, United States, (Jan. 2016), 494–501. doi:10.1109/ICTAI.2015.79.

T. Scheffer, C. Decomain, and S. Wrobel. 2001. "Active Hidden Markov Models for Information Extraction." In: *Advances in Intelligent Data Analysis*. Ed. by F. Hoffmann, D. J. Hand, N. Adams, D. Fisher, and G. Guimaraes. Springer Berlin Heidelberg, Berlin, Heidelberg, 309–318. ISBN: 978-3-540-44816-7.

B. Settles. 2009. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison. http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf.

C. E. Shannon. 1948. "A mathematical theory of communication." *The Bell System Technical Journal*, 27, 3, 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x.

P. Spracklen, N. Dang, Ö. Akgün, and I. Miguel. 2023. "Automated streamliner portfolios for constraint satisfaction problems." *Artificial Intelligence*, 319, 103915.

A. Tornede, M. Wever, S. Werner, F. Mohr, and E. Hüllermeier. Nov. 2020. "Run2Survive: A Decision-theoretic Approach to Algorithm Selection based on Survival Analysis." In: *Proceedings of The 12th Asian Conference on Machine Learning* (Proceedings of Machine Learning Research). Ed. by S. J. Pan and M. Sugiyama. Vol. 129. PMLR, Virtual Conference 12th Asian Conference on Machine Learning, originally planned for Bangkok, Thailand), (Nov. 2020), 737–752. https://proceedings.mlr.press/v129/tornede20a.html.

M. Vallati, L. Chrpa, and D. Kitchin. 2014. "ASAP: an automatic algorithm selection approach for planning." *International Journal on Artificial Intelligence Tools*, 23, 06, 1460032.

J. Vanschoren. 2019. "Meta-Learning." In: *Automated Machine Learning: Methods, Systems, Challenges*. Springer International Publishing, Cham, 35–61. ISBN: 978-3-030-05318-5. doi:10.1007/978-3-030-05318-5_2.

R. Volpato and G. Song. 2019. *Active learning to optimise time-expensive algorithm selection*. (2019). http://arxiv.org/abs/1909.03261 arXiv: 1909.03261.

L. Wang, X. Hu, B. Yuan, and J. Lu. 2015. "Active learning via query synthesis and nearest neighbour search." *Neurocomputing*, 147, 426–434. Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012). doi:https://doi.org/10.1016/j.neucom.2014.06.042.

X. Wu, Y. Zhong, J. Wu, B. Jiang, and K. C. Tan. 2024. "Large language model-enhanced algorithm selection: towards comprehensive algorithm representation." In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence* (IJCAI '24) Article 579. International Joint Conferences on Artificial Intelligence Organization, Jeju, Korea, 10 pages. ISBN: 978-1-956792-04-1. doi:10.24963/ijcai.2024/579.

L. Xu, F. Hutter, J. Shen, H. Hoos, and K. Leyton-Brown. Jan. 2012. "SATzilla2012: Improved algorithm selection based on cost-sensitive classification models." In: *Proceedings of SAT Challenge 2012: Solver and Benchmark Descriptions*. University of Helsinki, Finland, (Jan. 2012), 55–58.

L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2008. "SATzilla: portfolio-based algorithm selection for SAT." *Journal of artificial intelligence research*, 32, 565–606.

Z. Zhang, C. Xiao, S. Wang, W. Yu, and Y. Bai. 2024. "Automatic constraint programming solver selection method based on machine learning for the cable tree wiring problem." *The Journal of Engineering*, 2024, 3, e12368. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/tje2.12368. doi:https://doi.org/10.1049/tje2.12368.

J. Zhu, H. Wang, B. K. Tsou, and M. Ma. 2010. "Active Learning With Sampling by Uncertainty and Density for Data Annotations." *IEEE Transactions on Audio, Speech, and Language Processing*, 18, 6, 1323–1331. doi:10.1109/TASL.2009.2033421.

A Analysis of Uncertainty Measures in Pairwise Classification

In AL, uncertainty measures are frequently used as acquisition functions. While several uncertainty measures exist, the most common ones (LC, MS, and ES) result in identical ordering in pairwise classification. Their definitions are as follows:

- *Least Confidence*: For a given input x and predicted label $\hat{y}$, the model provides a posterior probability $P(\hat{y} \mid x)$. Least Confidence ranks and selects data point x^* for which the maximum posterior probability across all classes is the smallest:

$$x^* = \underset{x}{\operatorname{argmin}} \max_{\hat{y}} P(\hat{y}|x) \tag{1}$$

- *Margin Sampling*: This method considers the difference between the two highest posterior probabilities for a given input x. Let $\hat{y}_1$ and $\hat{y}_2$ denote the labels with the highest and second-highest posterior probabilities, respectively. A smaller margin indicates higher uncertainty, and the queried point x^* is selected as:

$$x^* = \underset{x}{\operatorname{argmin}}\, P(\hat{y_1}|x) - P(\hat{y_2}|x) \tag{2}$$

- *Entropy-Based Sampling*: This method considers the full posterior distribution across all posterior probabilities. It selects data points x^* with the highest predictive entropy, indicating maximum uncertainty:

$$x^* = \underset{x}{\operatorname{argmax}} - \sum_i P(\hat{y}|x) \log P(\hat{y}|x) \tag{3}$$

In pairwise classification, these three uncertainty strategies always rank the data points in the same way. This is because all three measures are monotonic functions of the same underlying probability distribution. For prediction probabilities p_1 and p_2 for a given input x, with $p_1 \geq p_2$, uncertainty increases when:

$$\mathrm{LC}(x) = 1 - p_1 \quad \text{and} \quad \mathrm{ES}(x) = -\left[p_1 \log(p_1) + p_2 \log(p_2)\right]$$

have higher values, whereas

$$\mathrm{MS}(x) = p_1 - p_2$$

has lower values. Despite their opposite monotonic behaviours, they give the same ranking of data points:

$$\mathrm{LC}(x_1) > \mathrm{LC}(x_2) \Leftrightarrow \mathrm{MS}(x_1) < \mathrm{MS}(x_2) \Leftrightarrow \mathrm{ES}(x_1) > \mathrm{ES}(x_2).$$

All three measures reach *maximum uncertainty* when the probability distribution is uniform, and *minimum uncertainty* when it is deterministic.

B Comparative Ranks of Selection Strategies Across Performance Levels

This appendix presents CD diagrams comparing the average ranks of selection strategies across different performance thresholds (75%, 50%, and 25% of passive learning performance). The results highlight the effectiveness of cost-aware strategies, particularly *PARETO (U–PC) (DTL)*, in achieving strong labelling efficiency.

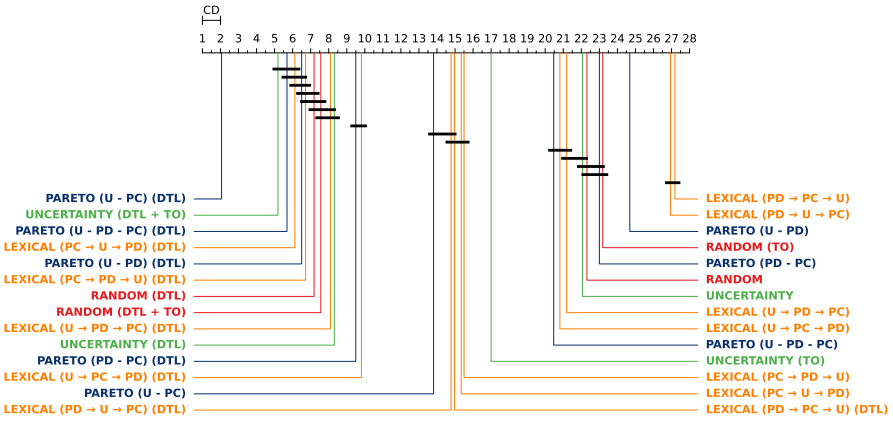

Fig. 6. CD diagram displaying the average ranks of selection strategies based on the labelling cost required to reach 75% of passive learning performance. PARETO (U – PC) (DTL) remains top-performing, confirming its effectiveness in minimising labelling cost under high-performance demands. On average, cost-aware strategies rank better than cost-unaware ones across the overall results.

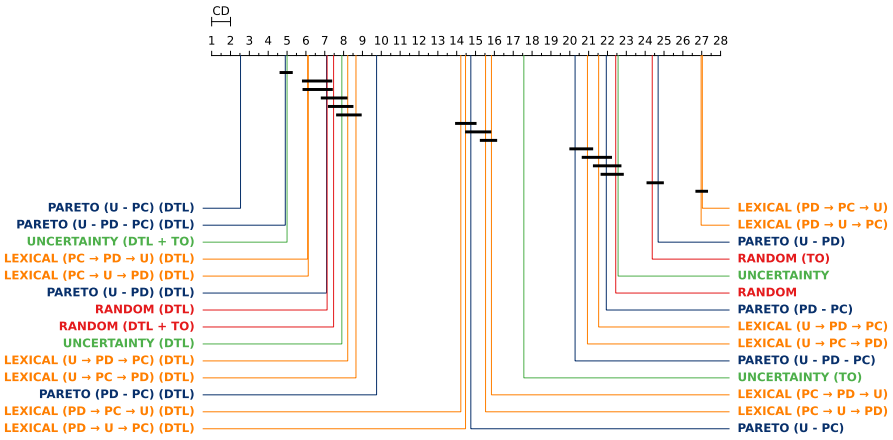

Fig. 7. CD diagram showing the average labelling cost ranks of selection strategies achieving 50% of passive learning performance. As the performance threshold increases, the relative ranking of strategies remains largely consistent, with only minor variations. Pareto-based methods incorporating DTL continue to hold top positions, while unconfigured strategies consistently rank lower in labelling efficiency.

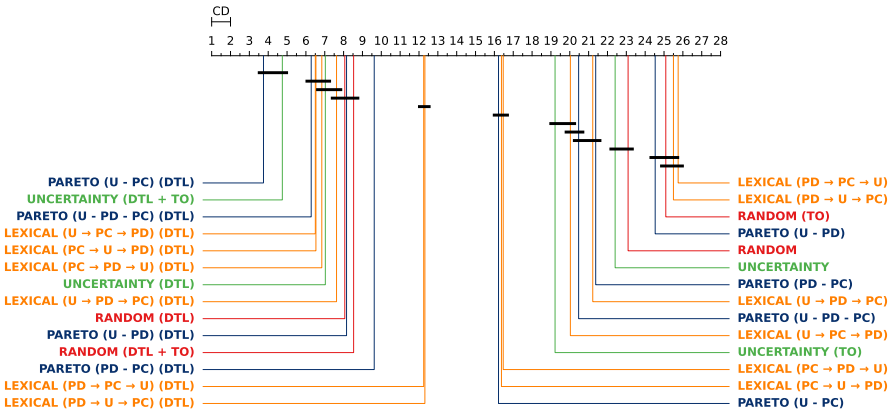

Fig. 8. CD diagram showing the average ranks of all evaluated selection strategies across 16 ASlib benchmark datasets, based on labelling cost required to reach 25% of passive learning performance. PARETO (U – PC) (DTL) and UNCERTAINTY (DTL + TO) rank highest, indicating strong cost-efficiency early in the early stages of learning.

C Labelling Cost Comparison of All Selection Strategies

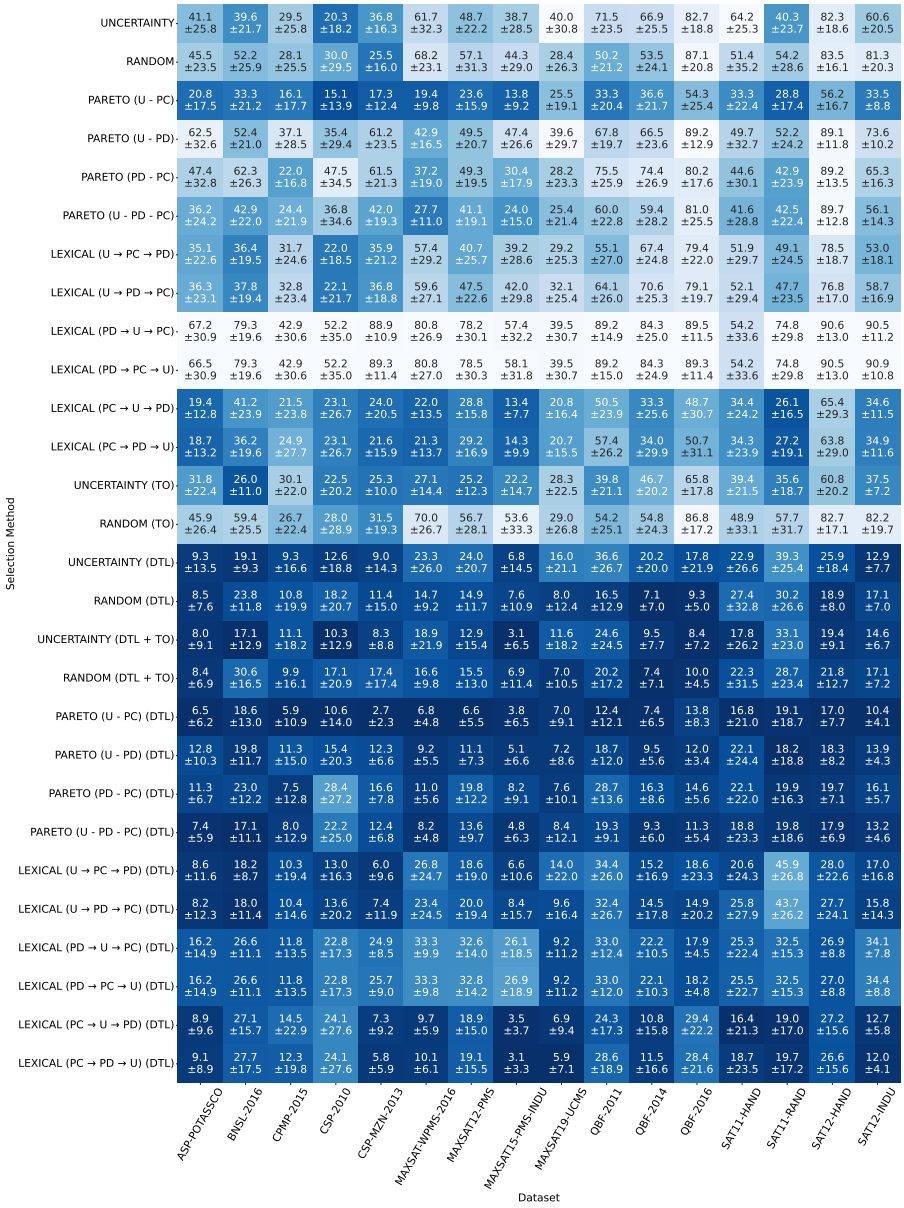

Fig. 9. The heatmap presents all proposed selection methods, including those with DTL and TO, aggregated over all splits and seeds. Each cell reports the average labelling cost along with its standard deviation. Darker blue regions indicate lower labelling costs and thus higher cost-efficiency on the corresponding datasets, whereas lighter blue regions denote comparatively less cost-efficient methods.

D Total Labelling Hours for Selection Strategies

Table 2. Labelling cost (in hours) of different selection and cost-efficient strategies compared to passive learning across all scenarios. Lower values indicate more efficient labelling cost while maintaining comparable performance to passive learning. The *PASSIVE LEARNING* row represents the baseline trained on the full train set. The *TEST SET COST* row denotes the cost of running all algorithms on all test instances for each scenario, serving as an additional reference point; notably, PARETO (U - PC) (DTL) achieves a labelling cost lower than the full test set cost across most scenarios.

Dataset	*ASP-POTASSCO*	*BNSL-2016*	*CPMP-2015*	*CSP-2010*	*CSP-MZN-2013*	*MAXSAT12-PMS*	*MAXSAT15-PMS-INDU*	*MAXSAT19-UCMS*	*MAXSAT-WPMS-2016*	*QBF-2011*	*QBF-2014*	*QBF-2016*	*SAT11-HAND*	*SAT11-RAND*	*SAT12-HAND*	*SAT12-INDU*
LEXICAL (PC → PD → U)	286h	847h	108h	42h	3865h	284h	781h	88h	806h	978h	1569h	2167h	500h	372h	4290h	3157h
LEXICAL (PC → PD → U) (DTL)	140h	648h	54h	43h	1029h	185h	173h	25h	381h	486h	530h	1208h	274h	268h	1783h	1082h
LEXICAL (PC → U → PD)	297h	962h	94h	42h	4285h	280h	735h	89h	833h	860h	1539h	2081h	502h	358h	4400h	3131h
LEXICAL (PC → U → PD) (DTL)	136h	634h	64h	43h	1305h	183h	194h	29h	368h	413h	503h	1253h	239h	259h	1824h	1160h
LEXICAL (PD → PC → U)	1015h	1854h	187h	94h	15893h	760h	3187h	168h	3071h	1520h	3894h	3806h	788h	1021h	6086h	8235h
LEXICAL (PD → PC → U) (DTL)	247h	623h	52h	41h	4579h	318h	1477h	39h	1261h	562h	1023h	776h	371h	445h	1814h	3114h
LEXICAL (PD → U → PC)	1025h	1854h	187h	94h	15821h	758h	3146h	168h	3071h	1519h	3898h	3815h	788h	1021h	6088h	8195h
LEXICAL (PD → U → PC) (DTL)	247h	623h	52h	41h	4420h	316h	1432h	39h	1268h	562h	1024h	762h	368h	444h	1813h	3088h
LEXICAL (U → PC → PD)	537h	854h	138h	40h	6390h	394h	2142h	125h	2178h	942h	3113h	3390h	756h	670h	5275h	4804h
LEXICAL (U → PC → PD) (DTL)	131h	426h	45h	23h	1065h	180h	354h	59h	1016h	586h	705h	793h	300h	628h	1890h	1545h
LEXICAL (U → PD → PC)	554h	881h	143h	40h	6543h	460h	2297h	137h	2261h	1092h	3261h	3381h	759h	651h	5165h	5319h
LEXICAL (U → PD → PC) (DTL)	125h	419h	46h	25h	1299h	195h	460h	41h	890h	552h	670h	636h	377h	595h	1864h	1431h
PARETO (PD - PC)	724h	1455h	96h	86h	10949h	477h	1669h	120h	1413h	1285h	3442h	3422h	652h	587h	5992h	5919h
PARETO (PD - PC) (DTL)	171h	536h	33h	51h	2942h	192h	453h	32h	420h	489h	756h	626h	321h	271h	1321h	1463h
PARETO (U - PC)	318h	776h	70h	27h	3086h	229h	758h	109h	736h	568h	1689h	2312h	487h	391h	3779h	3038h
PARETO (U - PC) (DTL)	100h	433h	26h	19h	470h	65h	205h	29h	257h	211h	341h	591h	246h	260h	1138h	947h
PARETO (U - PD - PC)	552h	1001h	106h	66h	7491h	398h	1316h	109h	1052h	1023h	2747h	3458h	606h	579h	6032h	5080h
PARETO (U - PD - PC) (DTL)	113h	402h	35h	40h	2185h	132h	265h	35h	314h	329h	431h	477h	273h	270h	1198h	1200h
PARETO (U - PD)	954h	1225h	161h	64h	10893h	480h	2597h	169h	1629h	1155h	3076h	3807h	722h	712h	5989h	6673h
PARETO (U - PD) (DTL)	196h	462h	49h	28h	2186h	108h	278h	31h	347h	319h	442h	510h	322h	250h	1233h	1259h
RANDOM	694h	1221h	122h	54h	4535h	553h	2431h	121h	2596h	858h	2472h	3713h	747h	741h	5611h	7368h
RANDOM (DTL + TO)	128h	716h	43h	31h	3099h	150h	377h	30h	626h	343h	335h	425h	324h	391h	1470h	1556h
RANDOM (DTL)	131h	557h	47h	33h	2031h	144h	411h	33h	562h	281h	328h	397h	396h	413h	1264h	1544h
RANDOM (TO)	701h	1389h	116h	50h	5616h	549h	2934h	123h	2657h	925h	2534h	3695h	712h	790h	5562h	7456h
UNCERTAINTY	626h	925h	128h	36h	6556h	471h	2119h	171h	2345h	1218h	3091h	3530h	940h	551h	5533h	5499h
UNCERTAINTY (DTL + TO)	123h	398h	49h	18h	1479h	125h	169h	49h	717h	419h	438h	361h	257h	451h	1304h	1324h
UNCERTAINTY (DTL)	143h	447h	41h	23h	1601h	232h	373h	68h	890h	624h	932h	759h	334h	538h	1738h	1172h
UNCERTAINTY (TO)	485h	607h	131h	40h	4496h	243h	1214h	121h	1029h	678h	2164h	2809h	575h	486h	4083h	3400h
PASSIVE LEARNING	1528h	2341h	435h	181h	17805h	969h	5483h	428h	3799h	1705h	4624h	4269h	1462h	1368h	6728h	9065h
TEST SET COST	209h	327h	68h	44h	2626h	147h	758h	54h	511h	235h	651h	583h	185h	183h	907h	1227h

E Algorithm Runtime in Hours for Selection Strategies

Table 3. Average AS performance among seeds and splits measured as test set runtime (in hours) for different selection strategies compared to passive learning across all scenarios. Instead of PAR10, the reported values reflect the actual runtime incurred by the algorithm selected for each test instance. If the selected algorithm fails to solve an instance within the time limits, all candidate algorithms in the portfolio are subsequently run on that instance to identify the best-performing algorithm. The *TEST COST* row denotes the cumulative runtime of executing all candidate algorithms on all test instances for each dataset, and serves as an additional reference point.

Dataset	*ASP-POTASSCO*	*BNSL-2016*	*CPMP-2015*	*CSP-2010*	*CSP-MZN-2013*	*MAXSAT12-PMS*	*MAXSAT15-PMS-INDU*	*MAXSAT19-UCMS*	*MAXSAT-WPMS-2016*	*QBF-2011*	*QBF-2014*	*QBF-2016*	*SAT11-HAND*	*SAT11-RAND*	*SAT12-HAND*	*SAT12-INDU*
LEXICAL (PC → PD → U)	27h	38h	20h	7h	525h	4h	28h	15h	25h	29h	142h	123h	53h	24h	162h	69h
LEXICAL (PC → PD → U) (DTL)	28h	39h	21h	7h	521h	4h	31h	15h	25h	30h	134h	126h	54h	23h	168h	76h
LEXICAL (PC → U → PD)	27h	39h	21h	7h	550h	4h	31h	15h	25h	29h	136h	127h	53h	23h	166h	71h
LEXICAL (PC → U → PD) (DTL)	28h	39h	21h	7h	547h	4h	30h	15h	25h	30h	134h	128h	54h	24h	169h	71h
LEXICAL (PD → PC → U)	28h	42h	20h	7h	583h	4h	31h	15h	24h	29h	134h	118h	52h	22h	168h	73h
LEXICAL (PD → PC → U) (DTL)	28h	39h	20h	7h	551h	4h	31h	15h	26h	29h	127h	116h	51h	23h	162h	76h
LEXICAL (PD → U → PC)	28h	42h	20h	7h	581h	4h	31h	15h	23h	29h	134h	119h	52h	22h	168h	73h
LEXICAL (PD → U → PC) (DTL)	28h	39h	20h	7h	547h	4h	32h	16h	28h	30h	128h	117h	52h	22h	162h	77h
LEXICAL (U → PC → PD)	26h	39h	20h	7h	518h	4h	28h	15h	25h	29h	130h	122h	50h	23h	172h	78h
LEXICAL (U → PC → PD) (DTL)	28h	39h	20h	7h	540h	4h	30h	16h	26h	29h	129h	118h	54h	24h	167h	75h
LEXICAL (U → PD → PC)	28h	38h	21h	7h	518h	4h	27h	15h	25h	29h	132h	122h	49h	23h	157h	72h
LEXICAL (U → PD → PC) (DTL)	28h	39h	21h	7h	545h	4h	30h	15h	28h	28h	128h	123h	53h	24h	169h	76h
PARETO (PD - PC)	27h	39h	21h	7h	533h	4h	29h	15h	25h	29h	133h	120h	51h	24h	164h	71h
PARETO (PD - PC) (DTL)	27h	39h	20h	7h	550h	4h	31h	15h	25h	30h	129h	123h	51h	23h	163h	78h
PARETO (U - PC)	27h	40h	21h	7h	532h	4h	28h	16h	24h	28h	129h	121h	52h	23h	166h	76h
PARETO (U - PC) (DTL)	27h	38h	21h	7h	538h	4h	30h	15h	26h	29h	131h	118h	53h	24h	169h	74h
PARETO (U - PD - PC)	26h	41h	21h	7h	555h	4h	27h	15h	24h	29h	135h	123h	52h	24h	157h	73h
PARETO (U - PD - PC) (DTL)	27h	38h	20h	7h	558h	4h	31h	16h	27h	29h	132h	123h	51h	23h	164h	77h
PARETO (U - PD)	28h	41h	20h	7h	582h	4h	28h	15h	25h	29h	126h	116h	52h	22h	165h	76h
PARETO (U - PD) (DTL)	28h	39h	20h	7h	579h	4h	29h	15h	28h	29h	127h	115h	53h	24h	161h	74h
RANDOM	26h	40h	20h	7h	498h	4h	28h	15h	24h	29h	131h	121h	51h	24h	165h	77h
RANDOM (DTL + TO)	28h	38h	20h	7h	559h	4h	30h	15h	26h	30h	133h	118h	50h	24h	163h	79h
RANDOM (DTL)	29h	39h	20h	7h	551h	4h	30h	15h	26h	30h	131h	120h	49h	24h	160h	78h
RANDOM (TO)	27h	40h	20h	7h	505h	4h	27h	15h	24h	30h	133h	124h	52h	24h	163h	72h
UNCERTAINTY	25h	40h	21h	7h	524h	4h	26h	15h	24h	29h	129h	125h	50h	23h	160h	70h
UNCERTAINTY (DTL + TO)	28h	38h	20h	7h	531h	4h	30h	15h	26h	30h	132h	114h	49h	24h	165h	77h
UNCERTAINTY (DTL)	29h	39h	21h	7h	566h	4h	28h	15h	29h	29h	133h	118h	52h	24h	172h	77h
UNCERTAINTY (TO)	29h	39h	20h	7h	495h	4h	27h	15h	24h	29h	132h	120h	51h	22h	169h	70h
PASSIVE LEARNING	30h	42h	22h	7h	624h	4h	35h	16h	28h	30h	139h	127h	55h	25h	161h	79h
TEST SET COST	209h	327h	68h	44h	2626h	147h	758h	54h	511h	235h	651h	583h	185h	183h	907h	1227h

F Analysis of the Fraction of Runs Used Across Selection Strategies

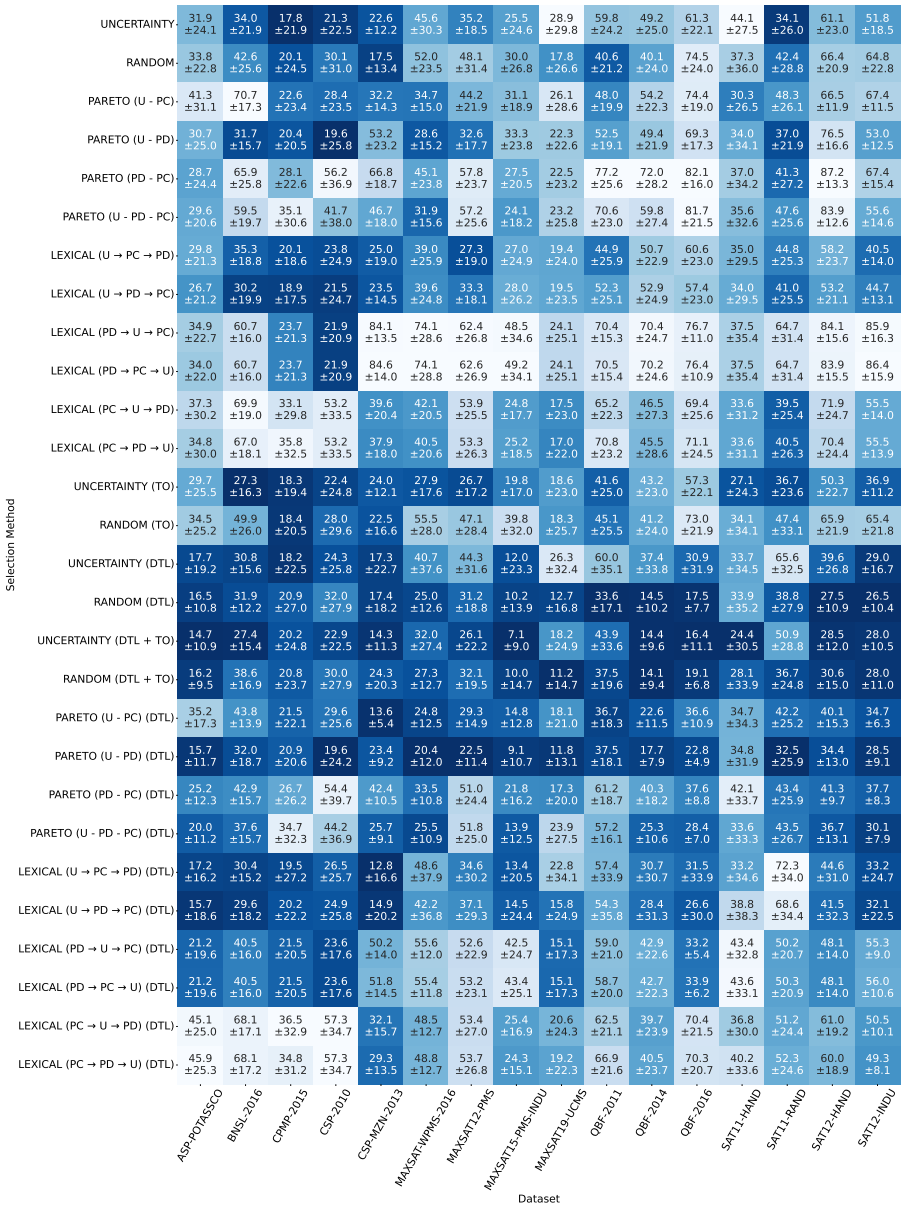

Fig. 10. Fraction of runs (in %) and standard deviation across 16 ASlib benchmark datasets for all selection strategies

G Additional Analysis for Different Budgets

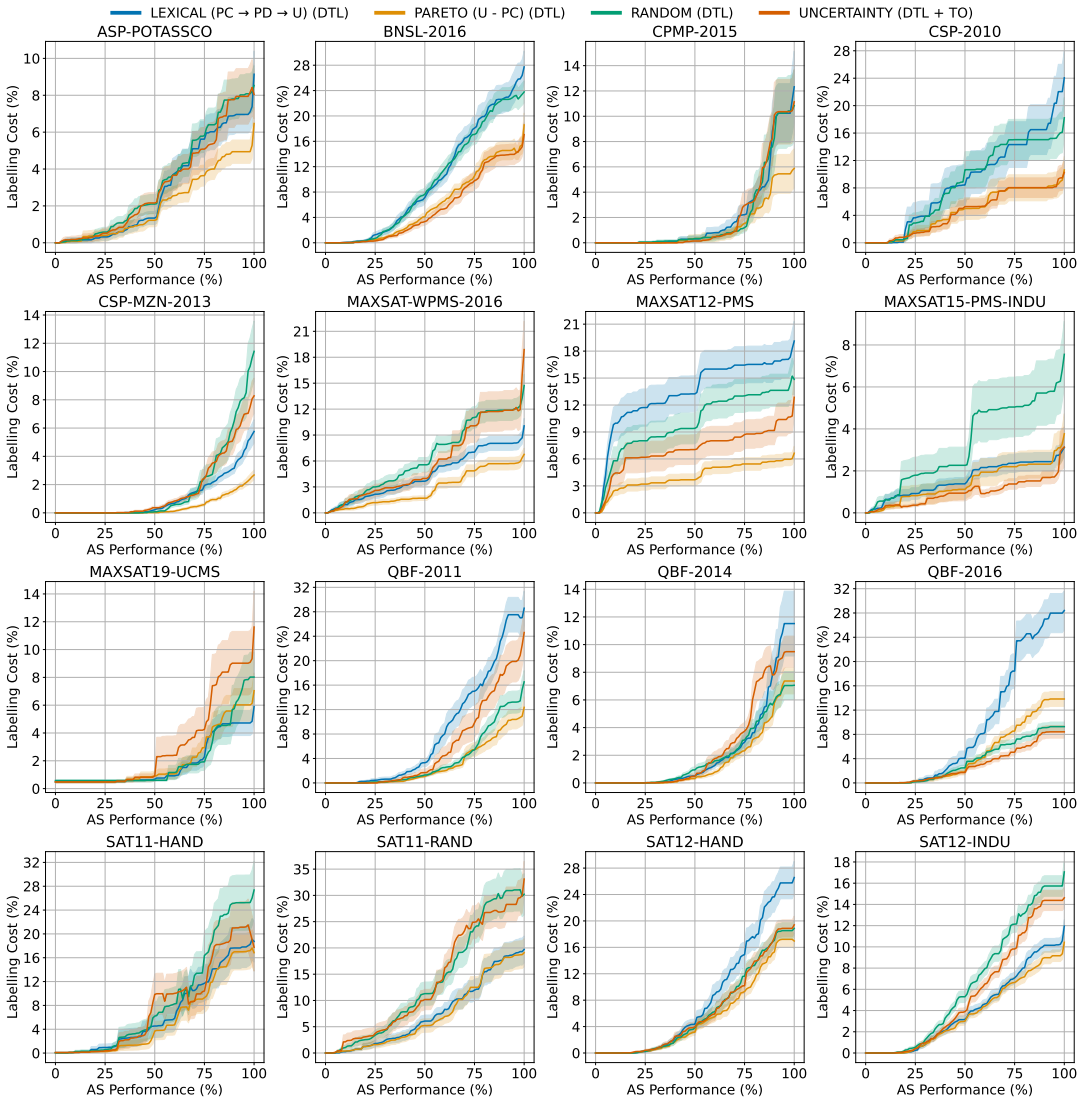

Fig. 11. Labelling cost versus algorithm selection performance across benchmark datasets for four representative strategies. Each line shows how much labelling cost is required to reach different levels of algorithm selection performance, where 100% corresponds to the performance of passive learning trained with the full labelled dataset. PARETO (U - PC) (DTL) consistently achieves high performance with minimal labelling effort. While the lexicographic strategy performs worse than the Pareto and uncertainty-based approaches, its efficiency is comparable to the RANDOM (DTL) baseline, suggesting limited benefit from lexicographic ordering.

H Ablation Analysis of the Impact of DTL on Aggregated Selection Strategies

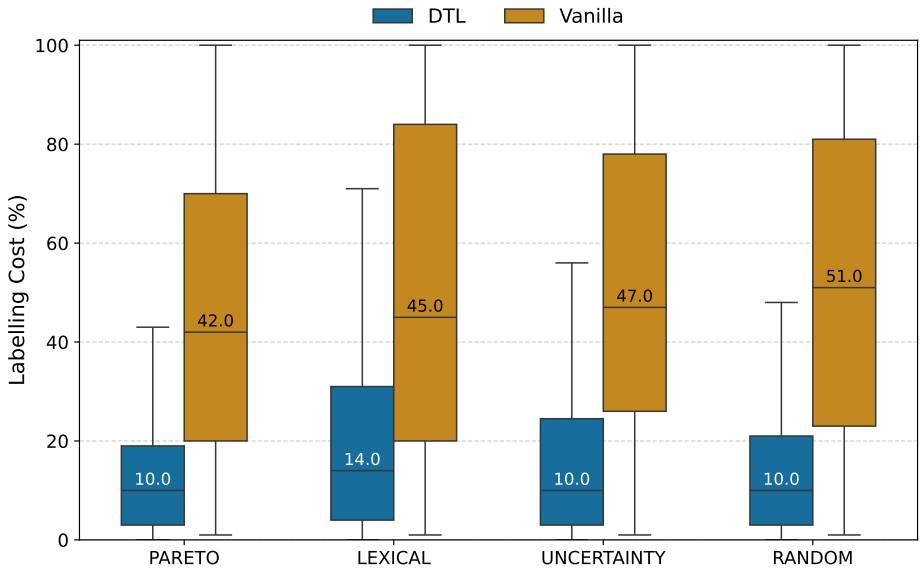

Fig. 12. Labelling cost (%) across all lexicographic and Pareto-based strategies, compared to representative uncertainty-based and random baselines. Each box shows the interquartile range (25th–75th percentiles) of performances across the different strategies of the respective strategy family, with the median marked by a horizontal line and annotated, and whiskers indicating the full observed range. Pareto-based strategies consistently achieve the lowest median labelling costs across both setups, confirming their superior overall performance. Under non-DTL conditions, lexicographic strategies perform similarly to the random and uncertainty-based baselines, while Pareto strategies remain clearly superior. Lexicographic strategies benefit less from DTL than random and uncertainty-based approaches.

I Ablation Study of Selection Metrics in Pareto-Based Selection

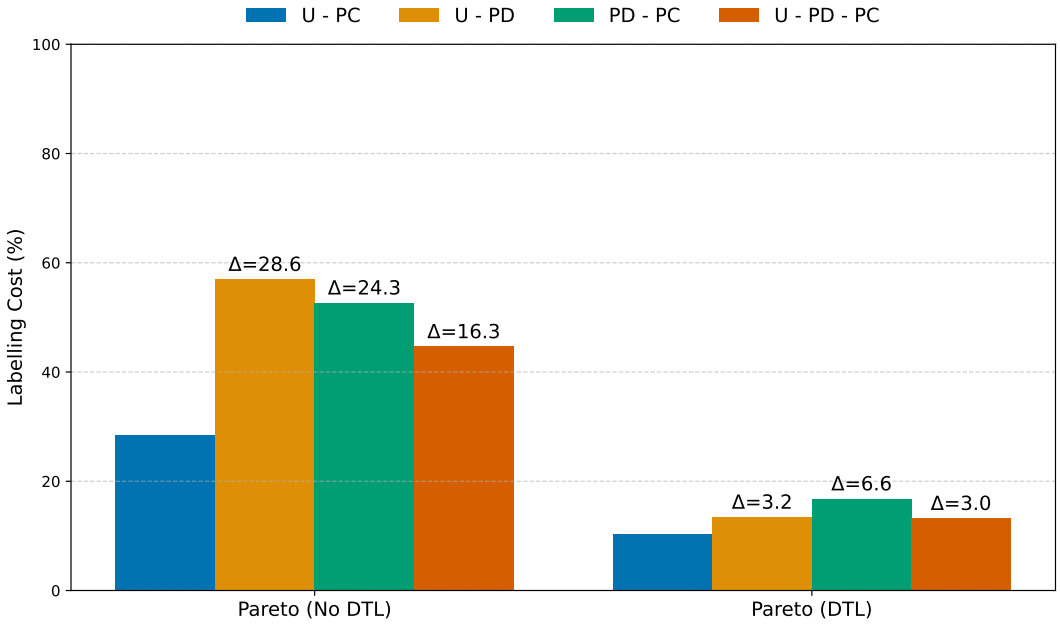

Fig. 13. Impact of different selection metrics used in Pareto-based selection strategies, both with and without Dynamic Time Limits (DTL). Each bar shows the average labelling cost (%) for a specific metric configuration: U - PC, U - PD, PD - PC, and U - PD - PC. Within each group, Δ denotes the labelling cost difference relative to the best-performing strategy (U - PC).

J Online Resources

All source code, data, and experimental results are available at https://doi.org/10.5281/zenodo.20095993.

Received 06 December 2025; accepted 12 December 2025