MODis: Multi-Objective Dataset Recommendation for Data Science Models

Abstract

Data discovery has been studied to identify relevant data to boost the performance of data science models in data-intensive tasks. In practice, there often comes the need to collect the data that can simultaneously optimize or trade among multiple performance metrics (such as training cost, accuracy, or inference cost) for specific models. Existing data augmentation techniques that apply table-wise data integration or feature selection are constrained to a single utility goal and meanwhile not sufficiently fine-grained to scale to multiple performance metrics. This paper introduces **MODis**, a novel paradigm to automatically create data from multiple sources with a goal to improve expected performance of a specific data science model in terms of multiple user-defined performance metrics. We formulate the problem as a multi-objective data discovery, and establish a formal computation system in terms of finite state transducer with finer grained operators enhanced by selection conditions. We analyze its expressiveness, and establish the hardness and approximability results. We then provide feasible algorithms that can create and maintain multiple datasets, from approximation scheme to fast heuristics that maintains the solution, all with guarantees in terms of Pareto optimality. We experimentally verify the efficiency and effectiveness of our algorithms in data creation. We also showcase their applications in accelerating the design of scientific data analytical pipelines.

1 Introduction

Data-driven analytical pipelines with data science models are routinely processed in a wide range of applications. Such pipelines rely on high-quality data science (machine learning) models. Among the challenges is the effective selection and creation of datasets that lead to high-quality models. In other words, *how to create new data to improve the overall (expected) performance of a model?*

Given a data science model M and a set of tables $\mathcal{D} = \{D_1, \ldots D_n\}$, data discovery aims to properly select *and* refine the data from $\mathcal{D}$ to obtain a new data table D', such that the expected performance of M over D' is improved. Moreover, the model M is evaluated by *multiple* (user-specified) performance measures such as training cost, processing cost, accuracy, inference cost, memory consumption, etc. It is desirable to suggest a dataset, over which the model has desirable performances in terms of *all* the measures.

Crowdsourced data platforms such as HuggingFace [13] provide portals to make datasets and models available. Data augmentation [30] and feature selection [22] have been also separately studied to improve machine learning, by carefully choosing useful data sources and feature space, respectively. Data integration has been adopted as an enabling technique to create new data for data augmentation [5]. Nevertheless, these approaches are not optimized where data science models are the first-class citizens. There is still a lack of effective solutions that (1) can suggest data that directly responds to a model as a "query" (*i.e.,* "search data with a model"); and (2) in particular, improve the expected performances of the model in the presence of *multiple* performance measures.

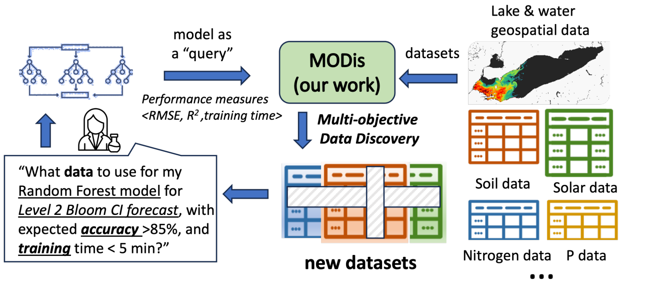

model as
a "query"
Performance measures
<RMSE, R^2 ,training time>
MODis
(our work)
datasets
Lake & water
geospatial data
Multi-objective
Data Discovery
"What **data** to use for my Random Forest model for *Level 2 Bloom CI forecast*, with expected ***accuracy*** >85%, and ***training*** time < 5 min?"
new datasets
Soil data
Solar data
Nitrogen data
P data
...

Figure 1: Discover and integrate features from multiple data sources improves the performance of a trained random forest (RF) for predicting CI index in both accuracy and training cost, as required by users. Manual tuning with cross-validation and feature selection tools find 10 features from more than 50 after several weeks. Our framework automatically suggested a new dataset with 8 features with comparable model accuracy and training cost, in a few minutes.

We consider the following real-world scenario.

Example 1.1. To understand the impact of harmful algal bloom (HABs) of a lake and its reason, a research team needs a model that can forecast the chlorophyll-a index (CI-index), a critical algal bloom quantification index, in the short term. The team has collected a set of tables that involve more than 50 factors (*e.g.,* fertilizer, water quality, weather) involving several upstream rivers and watershed systems of the lake. The team has already initialized a random forest (RF) model from small-scale tests, and wish to (1) improve its performance in terms of *multiple* metrics, including two accuracy measures: root mean square error (*RMSE*), and R^2 test, for "Level 2 bloom" CI-index, and its training cost; (2) understand *what* are new important factors and their values to be considered to achieve such improvement; and (3) track *where* the values are from and *how* they are integrated from the original data.

A typical process involves an excessive number of trials that repeats a pipeline of (manual) hyper-parameter tuning, feature selection, data integration, cross-validation, and inference tests. It turns out that only 8 key features with certain ranges of values are needed to achieve a regression with desirable accuracy at 89.6% (quantified by RMSE and R^2 test). This process took the team 8 weeks to get the desired model and the following observations.

(1) Not all the water data tables from every upstream river system are needed, but only two - a basin area and an upstream river.

(2) A single physicochemical attribute "`Phosphorus`" alone does not determine C1-index well for recent years, but "`Nitrogen`" and "`Phosphorus`" jointly determines CI-index, as an emerging pattern.

(3) The original data contains, for both columns "`Nitrogen`" and "`Phosphorus`", the values over the past 11 years. However, for an accurate short-term prediction in 2023, the fraction of the seasonal

"*Spring*" data and in particular years "*2013*" and "*2015*" are important, due to high nutrition loads.

The above example calls for an automated data discovery process that can (1) suggest data from available ones to improve data science models for *multiple measures* at scale, and (2) provide an integrative solution to suggest "what" data to choose and "how" to integrate the selected data. Moreover, the overall process should avoid expensive retraining and inference cost of the model.

One may exploit established data integration approaches to identify "joinable" tables and test the model. This requires an exhaustive enumeration of all the plans, leaving alone the overhead for actual evaluation. Such evaluation may already be expensive especially when M has high model complexity or with expensive user-defined functions [24], or involves nontrivial in-lab experiments or simulations as seen in scientific workflows [27].

Example 1.2. The research team may issue a query that asks: "*What are proper datasets for which our random forest model can have an improved performance in forecasting 'Level 2 bloom' CI-index, with expected accuracy in terms of RMSE at most* 0.3*,* R^2 *score at least* 0.7*, and training cost less than 5 minutes?*"

At first glance, this seems "contradicting" due to the trade-off between training cost and model accuracy for the measures on the CI index. Nevertheless, a data discovery process can still compromise to suggest one or more datasets, over which the model can "showcase" of desired accuracy for at least one metric (*e.g.,* a $\geq$ 86% accuracy for forecasting Level 2 bloom) and demonstrate comparable performance to the rest two metrics. This encourages us to seek for data discovery that can (1) *combine* data integration and multi-objective model evaluation, (2) pursue *multi-objective* optimization. Both are *new* challenges that are not well-addressed by existing data discovery methods for improving data-driven pipelines.

Moreover, the computation should not be limited to data augmentation by joining columns, but also to explore proper *selection conditions* and prune irrelevant rows. This reduces unnecessary overhead for downstream fine-tuning effort of the model.

Contribution. This paper proposes a novel framework, from a formal characterization of the data discovery process to feasible algorithms with provable guarantees, to address the above two challenges. Our main technical contributions are as follows.

(1) In response to the need of suggesting data with both columns and rows, We introduce a formal characterization of data discovery computation in terms of a *finite state transducer*, enhanced with fine-grained reduction and augmentation operators with selection conditions (Section 3). We study the expressiveness and property of the system. We show that the system resembles of data integration and feature selection. Better still, its computation demonstrate confluence property and a Church Russel property.

(2) Based on our formulation of a data discovery system, we address the second challenge and introduce a novel problem of multi-objective data discovery problem (MODis) (Section 4). We model the problem as a multi-objective optimization process of the running of the data discovery system, which performs sequential operators to update an initial dataset. The running aims to outputs a *Pareto set* of datasets, over which a model is expected to have optimized performance over multiple evaluations metrics.

(3) While the problem is in general intractable, we show that there exists a fully polynomial time approximation scheme (FPTAS) for a constrained version of the problem (Section **??**) under a fixed parameter specification. Our algorithm adopts a performance-driven search process, which interleaves multiple rounds of data integration and cost-effective model evaluation over historically observed tests and their performances.

(4) Moreover, we introduce two variants of the algorithm to address the practical needs. The first exploits known correlations of the performance metrics to perform early termination and pruning, via a bi-directional search scheme (Section 6). The second introduce a diversification process to improve the generality of the generated datasets (Section 7). Both preserve the quality guarantee.

(5) We finally introduce an efficient maintenance algorithm to maintain the datasets upon the updates of the underlying source dataset.

Using real benchmark datasets, models and analytical tasks, we experimentally verify the effectiveness of our data discovery scheme in improving the model performance and the efficiency of data generation. For example, our algorithms take 30 seconds to generate new data, that can improve pre-trained regression or classification models by 1.5-2 times in accuracy and makes their training cost 1.7 times faster. It outperforms other approaches that separately performs data augmentation or feature selection alone; and is feasible for large dataset. Our case study verifies its application in generating data for fine tuning data science models or benchmarking.

Related works. We categorize related works as follows. We remark that these work involve a common goal of improving the performance of data analysis models.

Feature Selection. Feature selection is a cornerstone task to remove irrelevant and redundant attributes and identify important ones for model training. Common strategies include filtering, wrapping, and embedding [22]. Filtering methods rank features in terms of correlation or mutual information [26, 29] and choose the top ones. They typically assume linear correlation among features, omitting collective effects from feature sets and hence are often limited to support directly optimizing model performance. Wrappers [16] choose features based on model performance estimated by a predictive model. The search cost is often extensive. Moreover, they risk overfitting due to optimizing a single criteria of model accuracy. Embedded methods directly learn feature selection models such as Lasso Regression, by penalizing (layer-wise) feature weights to improve model accuracy [36]. While they can be more efficient than wrapping, their performance are more sensitive to model architectures and task types, hyper-parameter tuning and additional domain-specific knowledge from datasets and models.

Our method differs in that (1) it creates new data with both data augmentation and masking of less useful values, rather than simply dropping the entire feature columns; (2) it finds data that improves the model in terms of multiple performance measures, beyond a single (accuracy) measure; and (3) the computation does not require internal knowledge of the models but simply data integration and masking operators, without incurring learning overhead. The process resembles SPJ (select, project, join) operations and readily benefit from well-established query optimization techniques.

Data Augmentation. Data augmentation aim to create data from multiple data sources towards a unified view [5, 38]. It is often specified to improve data completeness and richness [30] and may be sensitive to the quality of schema that in turn requires entity matching [21] and schema matching [15]. Our method has a goal to create a dataset to improve the expected performance of given data-driven models and analytical pipelines. This is different from the goal of conventional data integration which mostly focuses on improving the completeness of data and query answers alone.

Closer to our work is goal-driven data discovery [8, 9]. The methods perform data augmentation queries (joins) to enrich input data to improve the performance of given tasks. The process discovers and augments datasets as path plans, and achieves the desired performance quality with a bounded number of queries. Our approach differs from these works as follows. (1) We formalize data discovery with a more expressive model with finer-grained column augmentation and cell-level operators with selection conditions. This allows us to create data with more expressiveness, and to better mitigate over-fitting or under-fitting issues. (2) We target multi-objective goals beyond a single performance measure. We provide algorithms with quality guarantees in terms of Pareto optimality, as well as optimization techniques. These are not addressed in prior methods.

Crowdsourced Data Services. Several platforms are available to allow users to share and search datasets, such as Dataset search [3], Data.gov [1], Kaggle [14], Hugging Face [13], and Zenodo [7]. These services exploit user-defined tags to retrieve relevant datasets, yet lack the necessary capability to provide datasets for improving the expected performance for specific models or tasks. Our approach is among the the first effort with the enhanced capability to directly create datasets that improve the expected performance of models as "queries", and in terms of multiple performance measures.

2 Models and Performance Evaluation

Datasets. A dataset $D(A_1, \dots A_m)$ is a structured table instance that conforms to a relational schema $R_D(A_1, \dots A_m)$. Each tuple $t \in D$ is a m-ary vector, where $t.A_i = a$ ($i \in [1, m]$) means the ith attribute A_i of t is assigned a value a. A dataset may have missing values at some attribute A (*i.e.,* $t.A = \emptyset$).

We denote as $\mathcal{D} = \{D_1, \dots D_n\}$ a set of given datasets. Each dataset D_i confirms to a schema R_i. We denote as $\mathcal{A}$ the set of all the attributes from the datasets in $\mathcal{D}$. Given an attribute $A \in \mathcal{A}$, the *active domain* of A, denoted as $\text{adom}(A)$, refers to the finite set of all the values of A occurred in $\mathcal{D}$.

Models. We characterize a data science model as a function in the general form of $M : D \rightarrow \mathbb{R}^d$, which takes as input a dataset D, and outputs a result embedding in $\mathbb{R}^d$ for some $d \in \mathbb{N}$. Here $\mathbb{R}$ and $\mathbb{N}$ are real and integer sets. In practice, M can be a pre-trained machine learning model, a statistical model, or a simulator. The input D may represent a feature matrix (a set of numerical feature vectors), or a tensor (from real-world physical systems), to be used for a data science model M as training or testing data. The output embedding can be conveniently converted to task-dependent output.

We say a model M is *fixed*, if its computation process does not change for fixed input. For example, a regression model M is fixed if any factors that determines its inference (*e.g.,* number of layers, learned model weights) remain fixed. The model M is *deterministic* if it always outputs the same result for the same input. We consider fixed, deterministic data science models for the needs of consistency and accuracy required in data science pipelines.

Symbol	Notation
$\mathcal{D}, D, R_D$	a set of datasets, dataset, and schema
$\mathcal{A}, A, \text{adom}(A)$	attribute set, attribute, and active domain
M	a data science model $D \rightarrow \mathbb{R}^d$
$\mathcal{P}, p, (p_l, p_u)$	performance measures, a measure, and its range
$s.L$	bitvector encodes attributes A and $\text{adom}(A)$ in s
$T, t = (M, D, \mathcal{P}), t.\mathcal{P}$	a test set; a single test, and its performance vector
$\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$	a data discovery system
$\mathcal{E}$	a performance estimation model
$C = (s_M, \mathcal{O}, M, T, \mathcal{E})$	a configuration of data discovery system
$G_{\mathcal{T}} = (\mathcal{V}, \delta)$	running graph
$s \prec s', D \prec D'$	state dominance, dataset dominance
$\mathcal{D}_F$	output (ϵ)-Pareto set

Table 1: Table of notations

Model Evaluation. Given a model M and an input dataset D, a *test* t is a triple $(M, D, \mathcal{P})$ that specifies a test dataset D, a pre-trained model M, and a set of (user-defined) *performance measures* $\mathcal{P} = \{p_1, \dots p_l\}$. A *measure* p is an indicator to be *maximized* such as the model accuracy *e.g.,* precision, recall, F1 (for classification); or minimized *e.g.,* mean average error for regression tasks, training time, inference time, or memory consumption.

A performance measure $p \in \mathcal{P}$ can be always verified by an actual application of M over D, yet this can be expensive, especially when it involves in-lab tests or human inspection. We assume the measure p can be efficiently *approximated* by an estimation model $\mathcal{E}$ (or simply "estimator") [12, 35]. An estimator $\mathcal{E}$ makes use of a set of historically observed performance vectors of M (denoted as T) to infer its performance over a new dataset. By default, we use a multi-output Gradient Boosting Model [28] that allows us to obtain the entire performance vector by a single call (see Section 8).

We use the following notations. (1) We unify $\mathcal{P}$ as a set of normalized measures to be *minimized*, within a range $(0, 1]$. For a measure to be maximized (*e.g.,* accuracy), one can easily convert it to an inversed counterpart (*e.g.,* relative error). (2) Each measure $p \in \mathcal{P}$ has a user-specified pair of lower and upper bounds p_l, p_u to be normalized in $(0, 1]$, *i.e.,* $0 < p_l \leq p_u \leq 1$. The range $[p_l, p_u]$ may specify tolerable training costs, memory consumption, or error ranges. (3) A test tuple $t = (M, D, \mathcal{P})$ is *valuated* by an estimator $\mathcal{E}$, which assigns a (estimated) value to each measure $p \in \mathcal{P}$ in polynomial time. The *performance vector* $t.\mathcal{P}$ is a valuated tuple where each entry has a constant $t.p$ of the corresponding metric $p \in \mathcal{P}$, estimated by $\mathcal{E}$.

Example 2.1. Consider Example 1.1. A pre-trained random forest (RF) model M that predicts CI-index is evaluated by three measures $\mathcal{P} = \{\text{RMSE}, \text{R}^2, \text{T}_{\text{train}}\}$, which specifies the root mean square error, the R^2 score, and the training cost. A user specifies a desired normalized range of RMSE to be within $(0, 0.6]$, R^2 in $[0, 0.35]$ *w.r.t.* a "inversed" lower bound $1 - 0.65$, and T_{train} in $(0, 0.5]$ *w.r.t.* an upper bound of "3600 seconds" (*i.e.,* no more than 1800 seconds).

We summarize the main notations in Table 1.

3 Data Discovery: A Formalization

Given datasets $\mathcal{D}$ and a fixed deterministic model M, we formalize a data discovery system in terms of finite state transducers. This formalization not only provides us the design principles of feasible data discovery algorithms, but also verifies necessary properties that lead to their provable correctness and optimality guarantees.

Data discovery system. We characterize a *data discovery system* as a finite state transducer, denoted as $\mathcal{T} = (s_M, \mathcal{S}, \mathcal{O}, \mathcal{S}_F, \delta)$, where

- $\mathcal{S}$ is a set of states,
- $s_M \in \mathcal{S}$ is a designated start state,
- $\mathcal{O}$ is a set of operators of types $\{\oplus, \ominus\}$;
- $\mathcal{S}_F$ is a set of output states; and
- δ refers to a set of transitions.

We next specify its components.

States. A *state* s specifies a table D_s that conforms to schema R_s and active domains adom_s. For each attribute $A \in R_s$, $\mathsf{adom}_s(A) \subseteq \mathsf{adom}(A)$ refers to a fraction of values A can take at state s. $\mathsf{adom}_s(A)$ can be set as empty set $\emptyset$, which indicates that the attribute A is not involved for training or testing M; or a wildcard '_' ('don't care'), which indicates that A can take any value in $\mathsf{adom}(A)$.

Operators. We consider two classes of "shorthanded" operators to generate new data from an input dataset. These operators can be expressed and processed as SPJ (select, project, join) queries, enhanced with data augmentation or feature selection semantics.

(1) *Augmentation*: denoted as $\oplus_c(D_M, D)$, which augments a dataset D_M with dataset $D \in \mathcal{D}$ subject to a selection condition c. Here c is a single literal in the form of $A = a$ (an equality condition).

An augmentation operator $\oplus_c(D_M, D)$ performs the following queries: (a) augment the schema R_M of D_M with an attribute A from the schema R of D, *if* $A \notin R_M$; and (b) integrate D_M with tuples from D with values of $R.A$ satisfy value constraint c; and (c) fill in the rest cells of the augmented table with "null" whenever their attribute values are not known.

(2) *Reduction* $\ominus_c(D_M)$: mask the cells of the tuples in D_M with the values of an attribute $R_M.A$ satisfy a literal constraint c with "null". Here c is a single literal in the form of $A = a$, with $a \in \mathsf{adom}(A)$.

Transitions. Given a state s and a set of operators $\mathcal{O}$ of the two classes (simply denoted as $\{\oplus, \ominus\}$), applying an operator op $\in \mathcal{O}$ over s creates a new *result* state s'. We represent this as a *transition*, denoted as $r = (s, op, s')$, where s' is the result of applying op to s.

Running. A *configuration* of $\mathcal{T}$, denoted as $C = (s_M, \mathcal{O}, M, T, \mathcal{E})$, initializes a start state s_M with a dataset D_M, a finite set of operators $\mathcal{O}$ of augmentation and reductions, a fixed deterministic model M, an estimator $\mathcal{E}$, and a test set T. Both D_M and T can be empty set $\emptyset$. A *running* of $\mathcal{T}$ *w.r.t.* a configuration $C = (s_M, M, T, \mathcal{E})$ follows a general, deterministic process below.

- Starting from s_M, and at each state s, $\mathcal{T}$ iteratively applies operators from $\mathcal{O}$, whenever applicable, to either augment a state with new attributes and tuples, or mask tuple values from the state. This spawns a set of children of a state, each specifies a new result dataset.
- For each transition $r = (s, \text{op}, s')$ that is spawned with a result s' by applying operator op $\in \mathcal{O}$, $\mathcal{T}$ (1) constructs a test tuple $t(M, s'.D, \mathcal{P})$, (2) verifies if t is already in T; if so, directly load $t.\mathcal{P}$; otherwise, it invokes estimator $\mathcal{E}$ *at runtime* to valuate t, and adds t to T.
- The above process *terminates* at a set of output states $\mathcal{S}_F$, under the validation of an external termination condition, or no new transition can be spawned (no new datasets can be generated with $\mathcal{O}$).

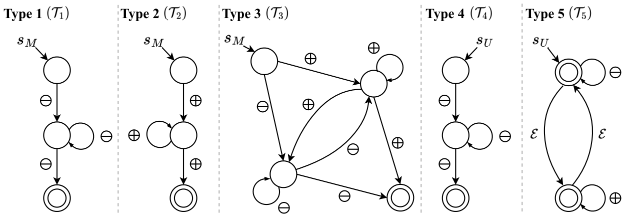


Figure 2: Data Discovery Systems: Specifications

The *result* of a running of $\mathcal{T}$ refers to the set of corresponding datasets $\mathcal{D}_F$ induced from the output states $\mathcal{S}_F$. As each output state $s \in \mathcal{S}$ uniquely determines a corresponding output dataset $s.D_s$, for simplicity, we shall use a single general term "output", denoted as $\mathcal{D}_F$, to refer to output states or datasets as needed.

Example 3.1. Fig. 2 illustrates several classes of data discovery systems, categorized by the type of computation they perform.

(1) **Type 1** (resp. **Type 2**) system, denoted as $\mathcal{T}_1$ (resp. $\mathcal{T}_2$) starts from an initial dataset D_M in its start state s_M, and only applies reduction (resp. augmentation operators) to update D_M.

(3) **Type 3** system, denoted as $\mathcal{T}_3$ is a general form (that generalizes Types 1-2 systems) and perform either augmentation or reduction in any step to create datasets.

(4) **Type 4** system is a variant of Type 1 under a configuration that starts with a designated universal state s_U, which is associated with a "universal" dataset D_U. The dataset is obtained by concatenating all schema in $\mathcal{D}$ as a "universal schema" R_U, and by integrating (joining) all tuples from $\mathcal{D}$. It can be specified to use a *"reduce-from-universal"* strategy and only performs reduction operators to remove cell values or tuples from D_U.

(2) **Type 5** system, denoted as $\mathcal{T}_5$, extends transitions δ with an ϵ-transition ("do-nothing and jump"), so allows a *"bi-directional"* computation. It can perform (a) a reduction to remove values from an initial dataset followed by an option to (b) jump and start to augmentation to another dataset with new attributes and their values. For example, it may start with a universal dataset D_U as aforementioned and perform reduction; and choose to "jump" to augment a set of (small) datasets, or vice versa, until the computations converge to some same datasets.

Running graph. A running of $\mathcal{T}$ can be naturally represented as the dynamic generation of a *running graph* $G_{\mathcal{T}} = (\mathcal{V}, \delta)$, which is a directed acyclic graph (DAG) with a set of state nodes $\mathcal{V}$, and a set of transition edges $r = (s, op, s')$. A *path* of length k is a sequence of k transitions $\rho = \{r_1, \ldots r_k\}$ such that for any $r_i = (s_i, op, s_{i+1})$, $r_{i+1} = (s_{i+1}, op, s_{i+2})$; *i.e.,* it depicts a consecutive application of operators that converts an initial state s_1 with dataset D_1 to a new counterpart s_k with D_k. Consistently, we say a state node $s \in \mathcal{V}$ is *valuated*, if a corresponding test $t(M, D_s, \mathcal{P})$ is valuated by $\mathcal{E}$.

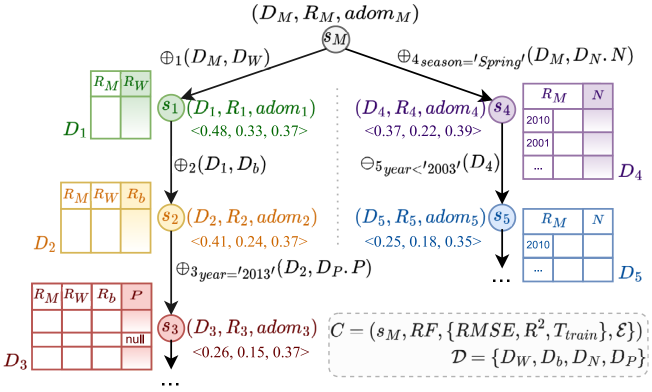


Figure 3: The running of a data discovery system, corresponding transition graphs, and result datasets.

Example 3.2. Revisiting Example 1.1, the observations specify a small fraction of data that is sufficient for CI-index forecasting using a random forest (RF) M at desired accuracy. Fig 3 illustrates a fraction of a running graph of a Type 4 system, creates such a dataset with $\mathcal{D}$ contains four datasets $\{D_w, D_b, D_N, D_P\}$, for a water table, a basin table, a Nitrogen table, and a Phosphorus table. The augmentation $\oplus$ is implemented with a library of spatial joins [31], a common query that join tables with tuple-level spatial similarity.

With a configuration $C = (s_M, \text{RF}, \{RMSE, R^2, T_{train}\}, \mathcal{E})$, where $\mathcal{E}$ is an MO-GBM estimator $\mathcal{E}$, a running starts by joining D_w and D_b, the basin table and the upstream river data to get D_2. It then augments D_2 with one more attribute "Phosphorus" under a selection condition "year = "*2013*", and leads to result D_3, via a path with edges labeled $\{\oplus_1, \ldots, \oplus_3\}$. In each step, a test t is initialized (if not already in test set T); the estimator $\mathcal{E}$ is consulted to valuate the performance vector of t and enrich T.

Similarly, another path is spawned by the first converting D_M to D_4 with an augmentation, followed by a reduction that removes historical tuples older then "2003", that are estimated to be less relevant. Fig 3 depicts a fraction of the transition graph of the above running, containing the two paths.

3.1 Expressiveness and Properties

We next discuss fundamental properties of a data discovery system. These properties help us justify the generality as well as practical implementations of data discovery algorithms (see Section **??**).

Expressiveness. We study the expressiveness power of the data discovery system in terms of all the paths with operators as "labels" it can produce in all its possible transition graphs. This gives us a set of strings, forming its language $L(\mathcal{T})$. We show that a data discovery system and its runnings resembles, and can be specified in practice for data integration [20], or feature selection [25], We provide the following result.

Proposition 3.3. *A data discovery system $\mathcal{T}$ can be configured to express (1) a data augmentation system, and (2) feature selection.*

Proof sketch: We show the above cases by an analysis on the languages generated by the specifications of type 4 system $\mathcal{T}_4$. To see this, it suffices to show that (1) $L(\mathcal{T}_1) \subseteq L(\mathcal{T}_4)$, and $\mathcal{T}_1$ can express feature selection; and (2) $L(\mathcal{T}_2) \subseteq L(\mathcal{T}_4)$, and $\mathcal{T}_2$ can simulate data augmentation process. One can readily infer (1) and (2) by treating $\mathcal{T}_1$ and $\mathcal{T}_2$ as special cases of $\mathcal{T}_4$, with proper configurations. For (1), one removes the model M and constrain $\mathcal{O}$ as data reduction operators only without selection conditions. For (2), one constraint $\mathcal{O}$ as data augmentation ("join") operators. □

Non-blocking. We also verify a desirable property of data discovery computation. A data operator (query) op is *non-blocking* [19], if for any path ρ with a transition that applies op and a result D in the running of a transducer $\mathcal{T}$, it generates the same result D regardless of at which step it applies op in ρ (whenever applicable). We have the following claim.

Lemma 3.4. *The operators of type $\oplus$ and $\ominus$ are non-blocking.*

We observe that in most applications, (a) $\oplus$ is commutative and associative, and (b) any consecutive application of an $\oplus$ operator followed by an operator in $\ominus$ are commutative. This ensures that the operators $\oplus$ and $\ominus$ are non-blocking. Moreover, this suggests that any paths in $\mathcal{T}$ are order-independent.

Following the above analysis, we have the result below.

Proposition 3.5. *Given a set of operators $\mathcal{O}$ with non-blocking operators $\oplus$ and $\ominus$, Type 3, Type 4, and Type 5 systems have the same expressiveness.* i.e., $L(\mathcal{T}_3) = L(\mathcal{T}_4) = L(\mathcal{T}_5)$.

In other words, any dataset that can be created by a running of a general form $\mathcal{T}_4$ can be simulated by a running of T_3 or T_5 under a proper configuration that leads to the same result. This allows us to choose a proper design of $\mathcal{T}$ from any of $\mathcal{T}_3$, $\mathcal{T}_4$ or $\mathcal{T}_5$ to perform the search with a completeness guarantee.

The above properties further indicate effective asynchronous implementation of data discovery in distributed environment for non-blocking operators. We defer such discussion in future work.

Church-Russel Property. With the above analysis, we verify that the computation of a data discovery system, from the perspective of a rewriting system, is both terminating and confluent, *i.e.,* demonstrate a "Church-Russel" property.

Proposition 3.6. *Given a configuration C with non-blocking operators of types $\{\oplus, \ominus\}$, and a data discovery system $\mathcal{T}$, the running of $\mathcal{T}$ is terminating and confluent.*

We show this by verifying that: (1) any computation of $\mathcal{T}$ leads to a "fixpoint" dataset D_f that has a universal schema R_U with all the tuples having 'null' values only, given possible and applicable operators in $\oplus$ or $\ominus$. This is because augmentation does not repeatedly add new attributes from R_U, and each cell's value, once set to be 'null' by a reduction, will not be changed again. (2) The sequences of operations are locally confluent (Lemma 3.4).

Due to limited space, we provide all the detailed proofs in [2].

4 Multi-Objective Data Discovery

Given a configuration, we want to find a running of $\mathcal{T}$ that leads to a "global" optimal dataset, over which M is expected to achieve the best performance for all measures. Nevertheless, such an optimal solution may not always exist. (1) Even normalized to be minimized, the measures in $\mathcal{P}$ may in nature conflict as trade-offs, such as training cost versus accuracy, or precision versus recall. (2) Given the "no free lunch" theorem [32], it is likely that no data-driven

(ML) model performs better than others. We thus instead consider a scheme that pursues *Pareto optimality* for $\mathcal{D}_F$.

To this end, we introduce a state dominance relation below.

Dominance. Given $\mathcal{P}$ and a data discovery system $\mathcal{T}$, a state $s = (D_s, R_s, \mathsf{adom}_s)$ is *dominated* by another state $s' = (D'_s, R'_s, \mathsf{adom}'_s)$ (resp. s' dominates s), denoted as $s \prec s'$ (resp. $s' \succ s$), if there are two valuated tests $t = (M, D_s)$ and $t' = (M, D'_s)$ in T, such that

- for each measure $p \in \mathcal{P}$, $t.p \geq t'.p$; and
- there exists a measure $p^* \in \mathcal{P}$, such that $t.p^* > t'.p^*$.

Consistently, we say a dataset D_s is dominated by another D'_s, denoted as $D_s \prec D'_s$, if for their states s and s', $s \prec s'$.

Given $\mathcal{T}$ and a configuration C, let $\mathcal{D}_F$ be the set of all the possible output datasets that can be generated by a running of $\mathcal{T}$, a set of dataset $\mathcal{D}^*_F \subseteq \mathcal{D}_F$ is a *Pareto set w.r.t.* $\mathcal{T}$ and C, if

- any pair D_1 and D_2 in $\mathcal{D}^*_F$ cannot dominate each other; and
- for any other $D \in \mathcal{D}_F \setminus \mathcal{D}^*_F$, and any $D' \in \mathcal{D}^*_F$, $D \prec D'$.

We now formulate the multi-objective data discovery problem.

Problem Statement. Given a configuration $C = (s_M, \mathcal{O}, M, T, \mathcal{E})$, and a data discovery system $\mathcal{T}$ that specifies non-blocking operators $\mathcal{O}$ and δ, the *multi-objective data discovery* problem, denoted as MODis, is to compute a Pareto set $\mathcal{D}_F$ in terms of $\mathcal{T}$ and C.

Example 4.1. Revisiting Example 3.2 and consider the temporal results $\mathcal{D}_F = \{D_1, \ldots, D_5\}$ with the following performance vectors valuated by the estimator $\mathcal{E}$ so far:

T: (D, M, $\mathcal{P}$, $\mathcal{E}$)	RMSE	$\hat{R^2}$	T_{train}
$t_1 : (D_1, \mathsf{RF}, \mathcal{P}, MO - GBM)$	0.48	0.33	0.37
$t_2 : (D_2, \mathsf{RF}, \mathcal{P}, MO - GBM)$	0.41	0.24	0.37
$t_3 : (D_3, \mathsf{RF}, \mathcal{P}, MO - GBM)$	0.26	$\underline{0.15}$	0.37
$t_4 : (D_4, \mathsf{RF}, \mathcal{P}, MO - GBM)$	0.37	0.22	0.39
$t_5 : (D_5, \mathsf{RF}, \mathcal{P}, MO - GBM)$	$\underline{0.25}$	0.18	$\underline{0.35}$

Here $\hat{R^2}$ is "inversed" as 1-R^2: the smaller, the better. All the measurement values are normalized in $(0, 1]$ *w.r.t.* user-specified upper and lower bounds, and the optimal values are underlined. One can verify the following dominance relation among the datasets: (1) $D_1 \prec D_2 \prec D_3$, and $D_4 \prec D_5$; (2) $D_3 \not\prec D_5$ and vice versa. Hence a Pareto set $\mathcal{D}_F$ currently contains $\{D_3, D_5\}$, over the valuated cases.

The problem, in nature a multi-objective optimization problem, is in general intractable [10]. We next investigate feasible algorithms that implements the running of data discovery systems, with quality guarantees. We also investigate how different search strategies may impact the efficiency. Our main results are summarized in Table ??.

5 Computing Pareto Optimal Datasets

To compute a Pareto set, a straightforward solution performs complete runnings of the system $\mathcal{T}$, to verify M's performance over all the N states (datasets) and invoke established multi-objective optimization methods such as Kung's algorithm [18]. This will incur $O(N \log N)^{|\mathcal{P}|-2}$ number of valuation (for $|\mathcal{P}| \geq 4$), or $O(N(\log N))$ if $|\mathcal{P}| < 4$. Such valuation cost is infeasible in practice, even when enlisting N states as a "once-for-all" cost is affordable. Moreover, a Pareto set may already contain an excessive number of datasets that are too expensive to be inspected.

We next present an algorithm that approximates Pareto set with a size-bounded solution, and reduces unnecessary valuations.

5.1 Approximating Pareto Optimality

To characterize a feasible algorithm for data discovery with size-bounded solutions and quality guarantee, we introduce the notion of ϵ-Pareto set, followed by an optimality measure.

ϵ**-Pareto set**. Consider a data discovery system $\mathcal{T}$ with a configuration $(s_M, \mathcal{O}, M)$. Let $\mathcal{D}_{\mathcal{S}}$ be a set of datasets generated by a running of $\mathcal{T}$. Given two datasets D and D' in $\mathcal{D}_{\mathcal{S}}$, and a constant $\epsilon > 0$, we say D' *ϵ-dominates* D, denoted as $D' \succeq_\epsilon D$, if the corresponding tests t=(M, D) and t'=(M, D'), $t'.p \leq (1 + \epsilon)t.p$ for each $p \in \mathcal{P}$.

A set of datasets $\mathcal{D}_\epsilon \subseteq \mathcal{D}_{\mathcal{S}}$ is an *ϵ-Pareto set* of $\mathcal{D}_{\mathcal{S}}$, if for every dataset $D' \in \mathcal{D}_{\mathcal{S}}$, there exists a dataset $D \in \mathcal{D}_\epsilon$ such that $D \succeq_\epsilon D'$. This balances a trade-off between computational efficiency and solution quality by allowing a state to be within an ϵ-threshold instead of strictly dominating another. Moreover, applied in our layer-wise Pareto set, it enables the inclusion of initially suboptimal states that may lead to competitive solutions in the future.

(N, ϵ)**-approximation**. We say an algorithm is an (N, ϵ)-*approximation* for MODis, if it satisfies the following:

- for arbitrarily given, and fixed small constant $\epsilon > 0$, it correctly outputs an ϵ-Pareto set *w.r.t.* N valuated states; and
- the time cost is a polynomial determined by $|\mathcal{D}|$, N, and $\frac{1}{\epsilon}$.

Below we present our main result.

Theorem 5.1. *Given datasets $\mathcal{D}$, configuration C, and a number N, there exists an (N, ϵ)-approximation for* MODis *in* $O\left(\min(N_u^{|R_u|}, N) \cdot \left(\left(\frac{\log(p_m)}{\epsilon}\right)^{|\mathcal{P}|-1} + I\right)\right)$ *time, where $|R_u|$ is number of attributes in $\mathcal{D}$, $N_u = |R_u| + |\mathsf{adom}_m|$ with* adom_m *the largest active domain, $p_m = \max \frac{p_u}{p_l}$; and I the valuation cost per test.*

Remarks. The scheme characterizes a relative optimality *w.r.t.* ϵ and input size of states N. When $\mathcal{T}$ is exhaustively run and N is the total number of possible states, it *ensures* to output a ϵ-Pareto set of a Pareto set $\mathcal{D}^*_F$. Although the algorithm is *not* in polynomial time *w.r.t.* dataset size $|\mathcal{D}|$, as N can be exponential, it is still desirable. As one can balance the 'closeness' of the output to a Pareto set, and the actual time cost, by explicitly tuning ϵ and a managable size N.

5.2 Approximation Algorithm

As a constructive proof of Theorem 5.1, we next present an (N, ϵ)-approximation algorithm, denoted as ApxMODis.

"Reduce-from-Universal". Algorithm ApxMODis implements a Type-4 system (Section **??**) and starts with a node as a "universal" dataset D_U (with a universal schema). In a nutshell, it "transforms" state dominance to a "path dominance" counterpart, which aggregate valuated measures from nodes to the end of the paths at runtime. This strategy has the following desirable properties. (1) As we uniformly minimize all the measurements, this allows us to grow "shortest" paths with prioritized reductions and valuation to refine D_U towards any of the user provided lower bounds, encouraging *early termination*; (2) starting from more features and more tuples allow the smaller sacrifice of model accuracy and generality.

Auxiliary structure. Algorithm ApxMODis dynamically spawns and maintains a running graph $G_{\mathcal{T}}$. In addition, it enhances $G_{\mathcal{T}}$ to a weighted graph, with the following global structure:

(1) Each state node s has a Boolean vector L(a bit-vector) to encode

Algorithm 1 :ApxMODis

```
1:  Input: Configuration $C = (s_U, \mathcal{O}, M, T, \mathcal{E})$, a constant $\epsilon > 0$,
        thresholds $\boldsymbol{t}$;
2:  Output: $\epsilon$-Pareto set $\mathcal{D}_F$.
3:  Queue $Q := \emptyset$; $Q$.enqueue($(s_U, 0)$); ${\mathcal{D}_F}^0[pos(\rho_U)] = \mathcal{P}_{s_U}$
4:  while $Q \neq \emptyset$ do
5:      (s, d) = $Q$.dequeue(), ${\mathcal{D}_F}^{d+1} = {\mathcal{D}_F}^{d}$;
6:      for all $s' \in$ OpGen $(s)$ do
7:          $Q$.enqueue($(s'$, d + 1));
8:          ${\mathcal{D}_F}^{d+1}$ = UPareto ($s'$, ${\mathcal{D}_F}^{d+1}$, ${\mathcal{D}_F}^{d}$, $\epsilon$);
9:  return ${\mathcal{D}_F}^{d+1}$
10: procedure OpGen(s)
11:     set $Q := \emptyset$
12:     for each $l \in s.L$ do
13:         if $l = 1$ then
14:             $s'.L = s.L$ with $l$ set to 0
15:             $Q$.append($s'$)
16:     return $Q$
17: procedure UPareto($s$, $R$, $Q$, $\epsilon$)
18:     for each $\rho' \in Q$ do
19:         compute $\mathcal{P}_s$ by invoking $\mathcal{E}$; set $\rho$ with $\mathcal{P}_s$;
20:         for $i \leftarrow 1$ to $|\mathcal{P}_s|$ do
21:             if $\mathcal{P}_s[i] > \boldsymbol{t}[i]$ then continue;
22:         compute pos($\rho$) with Equation (1);
23:         set $p$ as deterministic metric
24:         if $R$[pos($\rho$)] = $null$ or $R$[pos($\rho$)].$p$ < $\rho.p$ then
25:             $R$[pos($\rho$)] = $\rho$;
26:     return $R$
```

Figure 4: ApxMODis: **Approximating Pareto Sets**

if its local schema $s.R_s$ contains an attribute A from the universal schema and its active domain adom(A). This guards the applicability of reduction and avoids generating redundant datasets.

(2) Each transition (edge) $r = (s, \mathrm{op}, s')$ has a vector of weights $\mathbf{c}(r)$ of length $|\mathcal{P}|$, where each entry $\mathbf{c}(r)_p = t.p$ - $t'.p$ records the difference of valuated measure as p ranges over $\mathcal{P}$.

(3) As ApxMODis spawns paths, it also maintains at runtime for each path $\rho = \{r_1, \ldots r_k\}$ a *path label* as a pair $(\mathcal{P}(\rho), \mathrm{pos}(\rho))$, where (a) $\mathcal{P}(\rho)$ is a $|\mathcal{P}|$-ary vector with each entry $\mathcal{P}(\rho).p$ aggregates, for each measure $p \in \mathcal{P}$ as $\sum_{r \in \rho} \mathbf{c}(r).p$, and (b) pos($\rho$) "coordinates" ρ to a *discretized* $|\mathcal{P}|$-ary space as

$$\mathrm{pos}(\rho) = \left[\left\lfloor \log_{1+\epsilon} \frac{\mathcal{P}(\rho).p_1}{p_{l1}} \right\rfloor, \ldots, \left\lfloor \log_{1+\epsilon} \frac{\mathcal{P}(\rho).p_{|\mathcal{P}|}}{p_{l|\mathcal{P}|}} \right\rfloor\right] \quad (1)$$

which captures a relaxed "envelope" of valuated performance, that allows us to decide ϵ-dominance by a simple comparison.

Algorithm. The algorithm ApxMODis is illustrated in Fig. 4, which induces a Pareto Set $\mathcal{D}_F$ from a data discovery system $\mathcal{T}$. We first initialize queue Q with the initial state s_U, and set ${\mathcal{D}_F}^0$ at position ρ_U to s_U in line 3. In line 4 to 8, we update the Pareto set $\mathcal{D}_F$ for each depth iteratively. When we reach a certain depth, for each state $s \in Q$, we carry over the Pareto Set from the previous round. Procedure OpGen (line 10 to 16) explores all one-flip transitions from s and returns a set with the corresponding targets. After that, we iterate over all states s' in the output of OpGen, enqueue a new tuple with s' and incremented depth, and update the Pareto set ${\mathcal{D}_F}^{d+1}$ accordingly by Procedure UPareto (line 17 to 26).

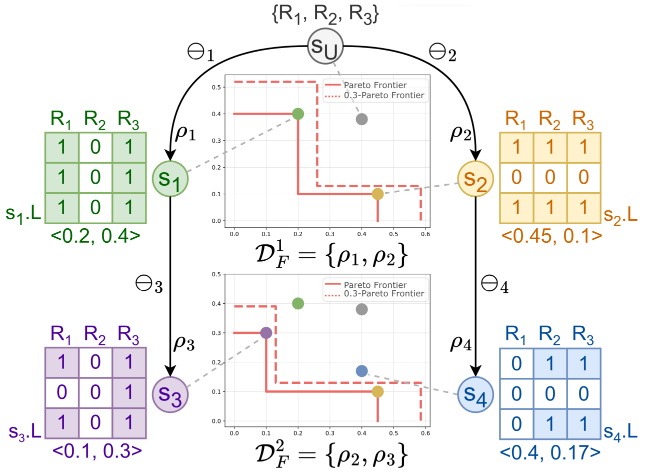


Figure 5: "Reduct-from-Universal": an illustration of two-level computation. The algorithm performs multiple level-wise spawns and updates the ϵ-Pareto set.

In Procedure UPareto, each potential path ρ' leading to the state s is extended to $\rho = \rho' \cup \{s\}$, and $\mathcal{P}_s$ is estimated by surrogate model $\mathcal{E}$. ρ will be discarded if any measures in $\mathcal{P}_s$ exceed **ts** thresholds. By Equation 1, we compute the ρ's position in $\mathcal{D}_F$, which will be a $(d-1)$-vector without considering the deterministic metric. In line 24, we check the dominance and maintain the superiority in the Pareto set, so that it retains only one optimal path in each position of $\mathcal{D}_F$, keeping its size polynomially bounded. After iterating through all depths, ApxMODis returns the refined ϵ-Pareto set $\mathcal{D}_F$ in line 9.

Example 5.2. Fig. 5 depicts a two rounds running of ApxMODis to create a (5, 0.3)-Pareto set, for a given dataset $\mathcal{D} = \{D_1, \ldots, D_3\}$ and a set of two measures $\mathcal{P} = < p_1, p_2 >$. The operator set $\mathcal{O}$ contains four avaibale reduction $\{\ominus_1, \ldots \ominus_4\}$. Start state s_U specifies a universal dataset D_U with a universal schema R_U, which can be efficiently computed by optimized multi-way join.

In the first round, it spawns a set of children from s_U that contains two states s_1 and s_2, with two reductions $\ominus_1$ and $\ominus_2$, respectively, as generated by the procedure OpGen by "fliping" the attributes and selection condition. It then consults the estimator $\mathcal{E}$ to valuate their performances, and identified that $D_1 \not\geq_{0.3} D_2$ and vice versa. Thus it sets the current 0.3-Pareto set as $\{D_1, D_2\}$.

In the second round, it further spawns with applicable reductions $\ominus_3$ and $\ominus_4$, extends ρ_1 that leads to s_1 and ρ_2 that leads to s_2, obtains new paths ρ_3 and ρ_4 with results D_3 and D_4, respectively. The coordination finds that $\mathcal{D}_3 \geq_{0.3} D_1$, but $D_2 \geq_{0.3} D_4$, and $D_2 \not\geq_{0.3} D_3$ and vice versa. This yields a new 0.3-Pareto set $\{D_2, D_3\}$. As the total valuated nodes reach 5, it outputs $\{D_2, D_3\}$ as the result.

Correctness and Approximability. Algorithm ApxMODis terminates as it spawns over N nodes with at most $|R_U||\mathrm{adom}_m|$ distinct reduction operators, where $|\mathrm{adom}_m|$ refers to the size of the largest active domain. Following Proposition 3.5, it also correctly processes over a complete solution space that can be generated by the general Type-3 system with a Type-4 implementation.

We next verify the ϵ-approximability. For any constant ϵ, we show that our computation simulates an iterative process that solves a multi-objective shortest path problem (MOSP) [33]. Given an edge-weighted graph G_w, where each edge e_w has a vector of utility values $e_w.\mathcal{P}'$, it is to find a Pareto set Π of optimal paths from a start node u. The dominance relation between two paths is determined by their utility vectors $\mathcal{P}'$, which are the cumulative utilities of their edges. It has been shown that MOSP is fully polynomial time approximable (FPTAS) for computing ϵ-dominance. ApxMODis in nature simulates the FPTAS scheme with multiple rounds of "replacement" strategy in terms of path dominance relation.

Time cost. Let $|R_u|$ be the total number of attributes in the universal schema R_u of D_u, and $|\mathrm{adom}_m|$ the size of the largest active domain. For time cost, ApxMODis performs $|R_u|$ levels of spawning, and at each node, spawns at most $|R_u|$ + $|\mathrm{adom}_m|$ children, given that it "flip" at most one attribute to reduce, and for each attribute, at most one domain value to mask. Let N_u be $|R_u|$+$|\mathrm{adom}_m|$. Thus it valuates at most $\min(N_u^{|R_u|}, N)$ nodes (datasets), in $I \cdot \min(N_u^{|R_u|}, N)$ time, where I refers to a polynomial time valuation cost of $\mathcal{E}$ per test. For each node, it then takes at most $\prod_{i=1}^{|\mathcal{P}|-1}\left(\left\lfloor \log_{1+\epsilon}\frac{p_{u_i}}{p_{l_i}}\right\rfloor + 1\right)$ time to update the ϵ-Pareto set. Given ϵ is small, $\log(1+\epsilon) \approx \epsilon$, and the total cost is in $O\left(\min(N_u^{|R_u|}, N)\cdot\left(\left(\frac{\log(p_m)}{\epsilon}\right)^{|\mathcal{P}|-1}+I\right)\right)$ time.

Given the above analysis, and considering $|R_u|$ and $|\mathcal{P}|$ are small constants, the cost is in a polynomial of input size $|D_u|$, N and $\frac{1}{\epsilon}$. Theorem 5.1 thus follows. The analysis confirms that ApxMODis provides a stronger guarantee when N is polynomially bounded by the universal dataset size $|D_U|$. Given $\mathcal{T}$ and configuration C, let $\mathcal{D}_S$ be the finite set of all the datasets that can be generated by $\mathcal{T}$.

Lemma 5.3. *If $|\mathcal{D}_S|$ is contained within the order of $O(f(|D_U|))$, where f is a polynomial, then the algorithm* ApxMODis *is a fully polynomial time approximation scheme (FPTAS) for* MODis.

We present the detailed analysis in [2].

6 "Bi-Directional" Data Discovery

Algorithm ApxMODis tends to exhibit enhanced efficiency when users specify higher expectations of the model performance, indicated by smaller thresholds $\boldsymbol{t}$, allowing for earlier termination. Yet, in cases involving extensive data discovery with a large $|\mathcal{D}|$, its "reduction-only" approach can lead to an enormous number of valuating. To mitigate this, we introduce BiMODis, which implements a bi-directional search strategy, aligning with Type-5 systems in Fig 2, incorporating a *Correlation-Based Pruning* strategy with early detection of dominance. It covers the space more rapidly and can offer a more balanced view of the solution, especially in high-dimensional datasets where many features may have a minimal impact.

Auxiliary structure. Algorithm BiMODis builds a dynamic spawning graph $G_{\mathcal{T}}$ akin to ApxMODis. For forward search, it maintains a forward frontier Q_f, which initiates from a universal schema s_U and progresses by reduction. Meanwhile, a backward frontier Q_b is introduced for backward search, from a state s_b, which includes a minimal subset of s_U, *e.g.,* ensures no classes will be lost if M is a classifier, and advancing step-by-step through augmentations.

Algorithm 2 :BiMODis

```
1:  Input: Configuration $C = (s_U, O, M, T, \mathcal{E})$, a constant $\epsilon > 0$;
2:  Output: $\epsilon$-Pareto set $\mathcal{D}_F$.
3:  set $s_b$ = BackSt($s_U$); queue $Q_f := \{(s_U, 0)\}$, $Q_b := \{(s_b, 0)\}$;
4:  while $Q_f \neq \emptyset$, $Q_b \neq \emptyset$ and $Q_f \cap Q_b = \emptyset$ do
5:      $(s', d) = Q_f$.dequeue();                    ▷ Forward Serach
6:      $(s'', d) = Q_b$.dequeue();                   ▷ Backward Serach
7:      ${\mathcal{D}_F}^{d+1} = {\mathcal{D}_F}^{d}$;
8:      for all $s^f \in$ OpGen ($s'$) and $s^b \in$ OpGen ($s''$) do
9:          ${\mathcal{D}_F}^{d+1}$ = UPareto ($s^f$, ${\mathcal{D}_F}^{d+1}$, ${\mathcal{D}_F}^{d}$, $\epsilon$);
10:         ${\mathcal{D}_F}^{d+1}$ = UPareto ($s^b$, ${\mathcal{D}_F}^{d+1}$, ${\mathcal{D}_F}^{d}$, $\epsilon$);
11:         if canPrune($s^f$, $s^b$) then
12:             prune ($C, s^f, s^b$);
13:         $Q_f$.enqueue(($s^f$, $d+1$)), $Q_b$.enqueue(($s^b$, $d+1$));
14: return $\mathcal{D}_F$
```

Figure 6: BiMODis**: Bi-directional Search**

Algorithm. Algorithm BiMODis, as depicted in Fig. 6, aims to get an ϵ-Pareto set $\mathcal{D}_F$ given a configuration C and a constant ϵ.

Initialization (lines 3). We first invoke the procedure BackSt to initialize a back-end start state node s_b. Two queues Q_f and Q_b are then established, seeded with start nodes s_U and s_b, and serve as the forward and backward search frontiers, respectively.

Bi-directional Search (lines 4-13). BiMODis conducts simultaneous exploration from two directions, controlled by Q_f for forward and Q_b for backward searches. Similar to ApxMODis, a layerwise Pareto set $\mathcal{D}_F$ is maintained and the search space is expanded incrementally by invoking the procedure OpGen, which employs reduction for the forward and augmentation for the backward search. The search process terminates when both Q_f and Q_b are empty, or when the two directions meet, at which point the final $\mathcal{D}_F$ is returned.

Procedure BackSt. This procedure initializes a backend dataset D_b for augmentation, tailored to the specific task. For example, for a classifier M with input features and a target attribute A to be classified, we sample a small (minimal) set of tuples in D_U to D_b that covers all the values of adom of A to ensure that no classes will be "missed". Other strategies can be applied to initialize D_b.

Correlation-Based Pruning. The bi-directional approach, though effective, still faces the challenge of valuating an overwhelming number of states. To address this, we proposed this pruning strategy that makes use of the simultaneous forward and backward search, also leveraging statistical correlations among performance measures, to enable the early detection and elimination of suboptimal states, which is not applicable in the reduction-only approach.

Correlation Analysis. For each pair of measures $p_i, p_j \in \mathcal{P}$, *Spearman correlation coefficient* is calculated based on dynamically updated records R, denoted as $corr(p_i, p_j)$. A perfect Spearman correlation of ±1 occurs when both variables are perfect monotone functions of each other [37]. p_i and p_j are considered strongly correlated if $|corr(p_i, p_j)|$ exceeds a user-defined threshold θ. A correlation graph G_C is maintained with measures in $\mathcal{P}$ as nodes and updated correlations between two measures as edge weights.

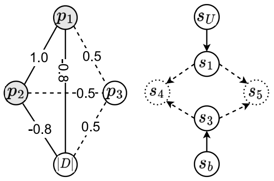
Parameterized Dominance. Given that measures $p \in \mathcal{P}$ includes ranges, and considering two valuated tests $t = (M, D_s)$ and $t' = (M, D_{s'})$, a state s is said to *parameterized ϵ-dominates* another state s', denoted as $s \gtrapprox_\epsilon s'$, if for every $p \in \mathcal{P}$:

- For single values: $t.p \le (1+\epsilon)t'.p$.
- For ranges with $t.p = [pl, pu]$ and $t'.p = [pl', pu']$: $pl \le (1+\epsilon)pl'$ and $pu \le (1+\epsilon)pu'$.
- For mixed cases: if $t.p = [pl, pu]$ then $pu \le (1+\epsilon)t'.p$, and if $t'.p = [pl', pu']$ then $t.p \le (1+\epsilon)pl'$.

Pruning Strategy. Here we focus on key measures, denoted as $\mathcal{P}^k \subseteq \mathcal{P}$, which exhibit monotonic behaviors throughout the search. For any feasible states s and s' from opposite directions, when either $\mathcal{P}^k_s$ or $\mathcal{P}^k_{s'}$ has missing values, we first estimate potential bounds for unrecorded measures using the test records R, based on strongest correlated measures that were recorded. If missing values persist, we will infer $\mathcal{E}$ to fill in the gap and update R accordingly. Intermediate states between s and s are pruned when $s \gtrapprox_\epsilon s'$ or $s' \gtrapprox_\epsilon s$.

Example 6.1. This example illustrates the operation of pruning. From left to right, the diagram presents the test records R, the correlation graph G_C, and part of the running graph $G_\mathcal{T}$. For each $s_n \in G_\mathcal{T}$, the associated $p \in \mathcal{P}_{s_n}$ is obtained by test $t_{s_n} = (M, D_{s_n})$.

	Label	p_1	p_2	p_3
s_U	(1, 1, 1, 1)	$\underline{0.42}$	$\underline{0.18}$	0.9
s_1	(1, 1, 1, 0)	0.4	0.17	0.1
s_2	(1, 0, 0, 1)	$\underline{0.5}$	$\underline{0.22}$	/
s_3	(0, 1, 0, 0)	**0.45**	/	/
s_b;	(0, 0, 0, 0)	0.6	0.4	0.3



At $\theta = 0.8$, p_1 and p_2 are positively correlated with each other, and negatively correlated with $|D|$, so $\mathcal{P}^k = \{p_1, p_2\}$. From s_U and s_b, forward and backward frontiers derive states s_1 and s_3, respectively. To determine $t_{s_3}.p_2$, we observe that $t_{s_3}.p_1 = 0.45$, which lies between $t_{s_U}.p_1 = 0.42$ and $t_{s_2}.p_1 = 0.5$ in R. As p_1 and p_2 are strong correlated, $t_{s_3}.p_2$ can be inferred to fall within the interval $[t_{s_U}.p_3, t_{s_2}.p_3]$, which is range $[0.18, 0.22]$. Set $\epsilon = 0.3$, since $0.45 \le (1+0.3) \cdot 0.4$ and $0.22 \le (1+0.3) \cdot 0.17$, it follows that $s_3 \gtrapprox_\epsilon s_1$. Therefore, intermediate states s_4 with label $(1, 1, 0, 0)$ and s_5 with label $(0, 1, 1, 0)$ will be pruned without further exploration.

Lemma 6.2. REVISE *For two feasible states s and s' from opposite directions, if any measures $p \in \mathcal{P}$ that is monotonically related to $|D|$ is missing in updated records R, infer p's the range from the measures most strongly correlated with it in R. Based on these inferred ranges and available measures, if $s \gtrapprox_\epsilon s'$, then an intermediate state s'' where $D_{s'} \subseteq D_{s''} \subseteq D_s$ (or $D_s \subseteq D_{s''} \subseteq D'_s$) can be prune, as $D_{s''} \notin \mathcal{D}_F$.*

Proof sketch: (1) If $s \gtrapprox_\epsilon s'$ holds, then for all measures $p \in \mathcal{P}$, and for any possible values within the defined ranges and available records, s is superior to s'. (2) Given $D_{s''}$ is an intermediate dataset between D_s and $D_{s'}$, and considering the monotonic relationship of the measures *w.r.t.* dataset size $|D_{s''}|$, the performance of s'' fall within the bounds established by s and s', so s'' cannot outperform s on any measure p, leading to the conclusion that $D_{s''} \notin \mathcal{D}_F$. □

Time Cost. The algorithm BiMODis takes the same time cost as ApxMODis. The (N, ϵ)-approximation holds for BiMODis given that it correctly updates the ϵ-Pareto set by definition, and simulates the computation of the Type 5 system. Also, our experimental study verifies that it is in practice much faster, and in particular suitable when creating new data from small datasets (see Section 8).

7 Diversified Data Discovery

In real-world scenarios, it is essential to diversify the discovered datasets to ensure a robust Pareto set. This diversification enables rapid adjustments in response to biases or inadequacies in the initial selection. Take medical diagnosis: if D_1 shows male bias and poor female diagnosis, an alternative dataset D_2 may offer a gender-balanced solution for accurate and inclusive outcomes. Moreover, diverse results can provide a holistic view of the exploration, which aids in optimizing experimental design (OED) [17, 23]. For instance, in developing eco-friendly cleaners, one dataset focuses on chemical formulations, while another one assesses effectiveness and user experience. Integrating them in development ensures the product is both environmentally friendly and consumer-effective.

From this, we introduce DivMODis, a variant of MODis that considers diversification: given a configuration C, a set of datasets $\mathcal{D}$, a number k, and constant ϵ, the task is to compute an ϵ-Pareto set $\mathcal{D}_F$ of at most k tables, maximizing diversification as follows:

$$\max_{D, D' \in \mathcal{D}_F} \sum_{i=1}^{k} \sum_{j=i+1}^{k} d(D_i, D_j)$$

The distance between two tables D_i and D_j is computed by a function d, encompassing data characteristics and performance vectors:

$$d(D_i, D_j) = \alpha \frac{1 - cos(s_i.L, s_j.L)}{2} + (1-\alpha) \frac{euclid(t_i.\mathcal{P}, t_j.\mathcal{P})}{euclid_{max}}$$

We show that both algorithms ApxMODis and BiMODis can be readily extended to return a k-set of diversified ϵ-Pareteo set, which remains to be (N, ϵ)-approximations. We outline such an algorithm, denoted as DivMODis is a variant of BiMODis.

Algorithm. Algorithm DivMODis addresses the max-sum diversification problem by enhancing BiMODis with on-the-fly diversification as it valuates tables. It maintains a diversified subset $\mathcal{D}^P_F \subseteq \mathcal{D}_F$ of size k, using a greedy strategy to select tables from $\mathcal{D}_F \setminus \mathcal{D}^P_F$ for inclusion. Starting with $\mathcal{D}^P_F$ as empty, DivMODis incrementally updates this subset with each execution of OpGen (s'), adding tables that maximize diversification by two cases:

(1) Adding tables to $\mathcal{D}^P_F$ until it contains k tables, choosing those with the greatest marginal gain.

(2) Once $|\mathcal{D}^P_F| = k$, it employs a sub-modular maximization solver $\mathcal{S}_\mathcal{A}$ [4] to potentially replace an existing table in $\mathcal{D}^P_F$ with a new table D' that offers higher diversification.

It is known that this greedy streaming selection strategy yields a $\frac{1}{4}$-approximation for stream-based Max-Sum Diversification [6]. Differs from BiMODis, DivMODis updates $\mathcal{D}_F$ with the k-set $\mathcal{D}^P_F$ to ensure it contains at least k diversified datasets, instead of feeding all spawned tables in *OpGen*(s') to the next level of bi-directional spawning. It thus returns a diversified ϵ-Pareto set $\mathcal{D}_F$ *w.r.t.* N valuated.

Time cost. DivMODis also performs $|R_u|$ levels of spawning, and at each node, spawns and selects at most k children at each level. DivMODis incurs additional overhead at each level to update the diversified k-set. The cost of updating $\mathcal{D}_F^P$ for a single level is in $O(k\cdot(|R_u|+|\mathsf{adom}_m|)\cdot T_{\mathcal{S}})$ time, where $T_{\mathcal{S}}$ is a unit cost of invoking the solver $\mathcal{S}_{\mathcal{A}}$ to process a single table, which is in polynomial time. The total additional overhead for diversification of DivMODis is thus in $O(k(|R_u|^2 + |\mathsf{adom}_m||R_u|T_{\mathcal{S}}))$.

As k, R_u, adom_m and $T_{\mathcal{S}}$ are all small costs, the overhead for diversification is in practice small. As verified by our tests, DivMODis has comparable time cost with BiMODis in almost all cases (see Section 8). This verifies the feasibility of diversified data discovery.

LEMMA 7.1. DivMODis *achieves a* $\frac{1}{4}$*-approximation for* MODis *diversification variant.*

Proof: □

8 Experiment Study

We next experimentally verify our algorithms. We aim to answer three questions: **RQ1**: How well can our algorithms improve the performance of models in multiple measures? **RQ2**: How fast are they in response to finding data for a model as "queries"? **RQ3**: What is the impact of configuration options? We also illustrate the applications of our approaches with case studies.

Datasets. We use the following datasets summarized below:

(1) Kaggle [14], a set of tables involving movie information;

(2) OpenData: a fraction of a large open public dataset [1]. We sampled $2K$ tables, involing schools recording, school evaluations and school types, houses recording, housing price in New York and Chicago, geographical location, among others;

(3) HF: a set of tables involving Avocado prices and relevant information, sampled from Hugging Face [13].

Dataset	Total #tables	Total #Columns	Total #Rows	Size
Kaggle	1943	33573	7317773	704.6MB
OpenData	2457	71416	33296998	8.58GB
HF	255	1395	10207261	3.5GB

Tasks and Models. We have trained the following models: (1) a random forest models RFhouse for classifying house price (Task T_1), using OpenData and the settings consistently in [8, 9]; (2) a Gradient Boosting Model (GBmovie) for predicting the movies' worldwide gross sale, using Kaggle (Task T_2); and (3) a regression model for predicting Avocado price, using HF (Task T_3).

We trained all these models with scikit-learn [28]. For a fair comparison, we use the original training scripts provided by the baseline methods and validated that the reproduced models have consistent performance as reported.

Estimator $\mathcal{E}$. We trained a multi-output Gradient Boosting Model (MO-GBM) in scikit-learn [28] as estimator. This type of regressor outputs predicted values for multiple variables, thus allows us to valuate the entire performance vector for each test with one call.

Algorithms. We implemented the following algorithms in Python.

(1) **MODis**: Our multi-objective data discovery algorithms, including ApxMODis, BiMODis, and DivMODis. We also implemented NOBiMODis, a counterpart of BiMODis without the optimization.

(2) METAM [9]: A goal-oriented data augmentation algorithm that perform consecutive joins of tables to optimize a single specific performance measure for a downstream task.

(3) Starmie [8]: A data discovery algorithm with table union search as the main use case. Given a table, it discovers union-able tables by learning feature semantic similarity with contrastive learning.

(4) SkSFM [28]: an automated feature selection method in scikit-learn's SelectFromModel, which recommend important features with a built-in estimator;

(5) H2O [11]: an AutoML platform used to automatically optimize machine learning pipelines. We utilized its feature selection module to fit features and predictors into a generalized linear model and then obtained a subset of selected features.

Evaluation. We evaluate data discovery algorithms in terms of tasks, performance measures and test models, as summarized below.

Tasks	Type	Dataset	Model	Perf.
T_1: House Price (C)	Classification	OpenData	RFhouse	$\mathcal{P}_1$
T_2 : Movie Gross (C)	Classification	Kaggle	GBmovie	$\mathcal{P}_2$
T_3 : Avocado Price (R)	Regression	HF	LRavocado	$\mathcal{P}_3$

Performance Measures. For each task, we adopted a group of measures to guide the data discovery, as summarized below.

Notation	Measures	Used In
p_{Acc}	Model Accuracy	$\mathcal{P}_1$-$\mathcal{P}_3$
p_{Tr}	Training Time Cost	$\mathcal{P}_1$-$\mathcal{P}_3$
p_{F1}	F_1 score	$\mathcal{P}_2$
p_{MAE}	Mean Absolute Error	$\mathcal{P}_3$
p_{MSE}	Mean Squared Error	$\mathcal{P}_3$
p_{Fsc}	Fisher Score [22]	$\mathcal{P}_1$
p_{MI}	Mutual Information [9, 22]	$\mathcal{P}_1$
p_{VIF}	Variance Influence Factor [22]	$\mathcal{P}_1$
p_{DSize}	Size of Created Dataset	None

(1) The first four directly quantify a model's performance in terms of accuracy (p_{Acc} for classification and regression, and both p_{MAE} and p_{MSE} for regression) and training cost (p_{Tr}). (2) The latter three (p_{Fsc}, p_{MI} and p_{VIF}), generally used in feature selection [22] quantify the statistical relationship between a set of input variables (features) and a "target" feature (*e.g.,* 'House Price' to be classified, or 'Avocado Price' to be predicted); the larger, the better. Among these, p_{MI} is also adopted by [9] as an optimization goal for data discovery. (3) To evaluate the amount of result, we also report the size of the data (p_{DSize}), in terms of (total # of rows, total # of features). As all baselines only report a single table, and MODis report a set of tables, we report total size in favor of baselines. Here if a column has all cell masked, we consider the column reduced and remove it from the output table.

For each tasks in T_1-T_3, we initialized our MODis methods consistently with a configuration that specifies an original dataset, the matching trained model, and the corresponding measures $\mathcal{P}_1$-$\mathcal{P}_3$.

Evaluation metrics. We adopt the following metrics to quantify the effectiveness of data discovery approaches. Denote as D_M an initial dataset, and $\mathcal{D}_o$ a set of output datasets from a data discovery algorithm. (1) We define the *relative improvement* $\mathsf{rImp}(p)$ for a given measure p achieved by a method as $\frac{M(D_M).p}{M(D_o).p}$. As all metrics are normalized to be minimized, the larger $\mathsf{rImp}(p)$ is, the better D_p is in improving M *w.r.t.* p. Here $M(D_M).p$ and $M(D_p).p$ are obtained by actual model inference test. This allows us to fairly compare all methods in terms of the quality of data suggestion.

For efficiency, we compare the total time cost of the data discovery process upon receiving a given model or task as a "query".

Exp-1: Effectiveness with Single Measure. Our first experiment evaluates all algorithms in evaluating how well the model's performance can be improved over the dataset(s) they created. As p_{Acc} is the single measure considered by METAM, and all baseline produce a single table, we (1) compare MODis algorithms by selecting the table in the Pareto set with best estimated p_{Acc}, and (2) apply model inference to all the datasets, to report the actual measurement values. We show the results for T_3 in Table 2 ("Original" refers to the measures over the input dataset). We find the following.

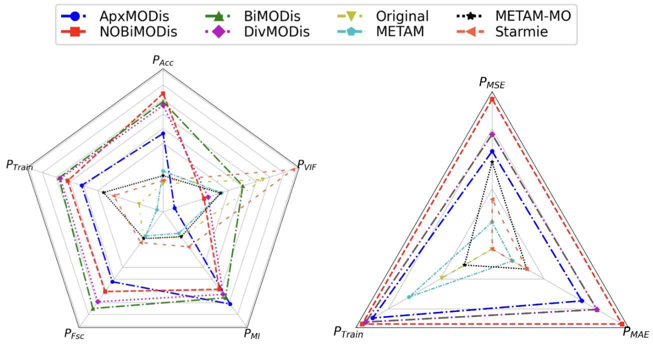
(1) MODis algorithms outperform all the baselines in creating a dataset to improve the performance in terms of p_{Acc}. The one with best p_{Acc} and second best is obtained by BiMODis and NOBiMODis, respectively, and all MODis methods finds data for which p_{Acc} achieves 0.94.

(2) Over the same dataset and for other measures, MODis algorithms still outperforms the baselines in most cases. For example, the result datasets that optimize p_{Fsc}, p_{MI} and p_{VIF} are obtained by ApxMODis, NOBiMODis and DivMODis, respectively; and BiMODis finds a dataset that achieves three second best results in p_{Train}, p_{Acc} and p_{Fsc}. This verifies their ability in optimize data discovery towards multiple measures simultaneously.

(3) All baseline methods perform data augmentation or feature selection that leads to a single table. The data augmentation methods (METAM, Starmie) mainly include more features to improve accuracy; and feature selection (SkSFM and H2O) reduce them at a cost of accuracy but improved training cost. MODis methods are able to balance these trade-offs better by *explicitly* performing multi-objective optimization. Consider p_{Acc} and p_{Train}. The best result for training cost is contributed from SkSFM, yet at a cost of lowest model accuracy. As MODis methods are able to optimize both measures (among others), by making flexible decision to augment with new features or reduce cells and tuples to make the data smaller, they are able to find data with improved accuracy as well as smaller training cost, compared with baselines.

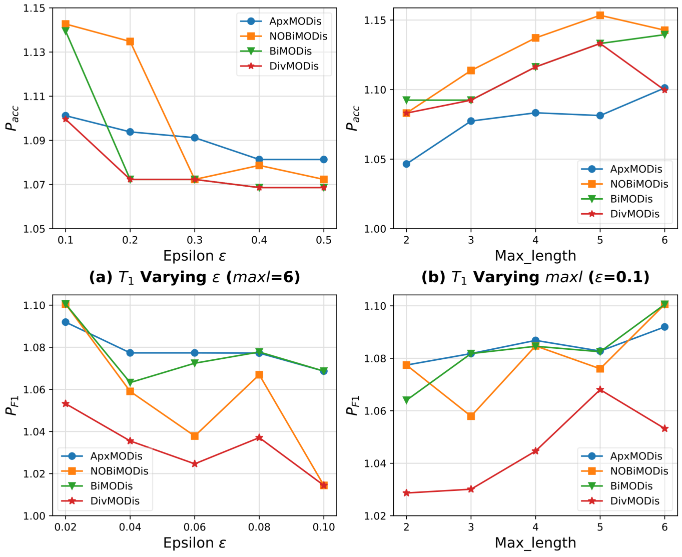
(4) Despite p_{Acc} is a first-class citizen in this comparison, not all baselines improve it (given its value over "Origin") significantly, except Starmie. Yet Starmie improves accuracy at a cost of including the most number of features (13 new ones). Feature selection methods (SkSFM and H2O) achieved better result on accuracy with much less number of features, and consistently showing better results in feature correlation measures in terms of p_{Fsc}, p_{MI} and p_{VIF}. On the other hand, MODis methods *explicitly* included these into optimization scope with a multi-objective estimator, and are able to improve accuracy without introducing many new attributes.

Exp-2: Effectiveness with multiple measures. We next evaluate MODis algorithms, METAM and Starmie, using multiple measures in T_1 and T_3. For each measure p and an algorithm, we choose the dataset D with the best estimated measure of p it generates. We then retrain the model using D to get the true measurement. We normalize all the values into a same range. The results are illustrated as radar graphs in Fig. 7. The lines "Original" mark the values of the measures in the original data.



T1: Classification T3: Regression

Figure 7: Effectiveness: Multiple Measures



(a) T_1 Varying ϵ (*maxl*=6) **(b) T_1 Varying *maxl* (ϵ=0.1)**

(c) T_2 Varying ϵ (*maxl*=6) **(d) T_2 Varying *maxl* (ϵ=0.02)**

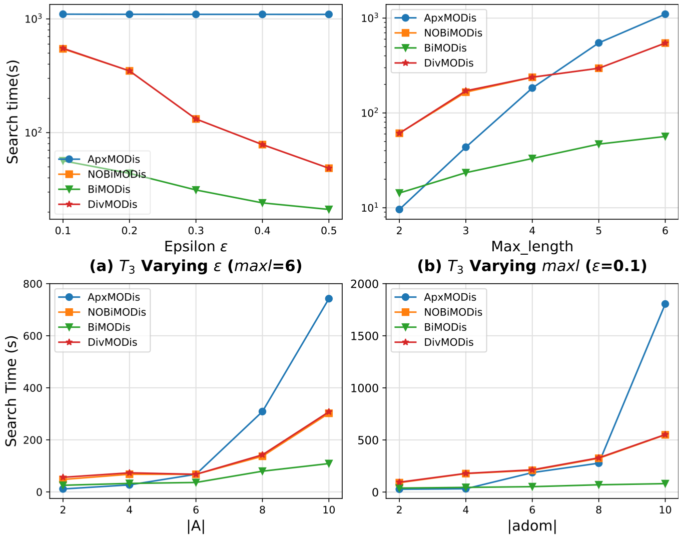
Figure 8: Effectiveness with Varying Factors

In general, MODis algorithms are able to create datasets that generally improve a model in a balanced performance. In particular, BiMODis, DivMODis and NOBiMODis provide top results for multiple measures. METAM is optimized to provide good results for a single measure, such as accuracy in T_1. Starmie is not specifically optimized for optimizing measures, and provides a balanced performance in T_2. ApxMODis provides in particular better results over p_{VIF}, a measure for feature correlation, with a possible reason that it performs more localized reduction only operations that is closer to feature selection process.

Exp-3: Effectiveness and Impact factors. We next investigate the performance of MODis methods under the impact of factors, in particular ϵ, the approximation factor; and the maximum path length (maxl) to bound the number of valuated states N.

T_3: Movie Gross (C)	Original	METAM	Starmie	SkSFM	H2O	ApxMODis	NOBiMODis	BiMODis	DivMODis
p_{Acc}	0.8560	0.8515	0.8606	0.8285	0.8545	0.9257	**0.9782**	*0.9755*	0.9413
p_{Train}	1.4775	1.5905	1.2643	**0.6028**	0.9692	1.0575	0.9107	*0.8027*	1.0486
p_{Fsc}	0.0824	0.0949	0.1286	0.7392	0.3110	0.6013	**0.9257**	*0.9240*	0.6618
p_{MI}	0.0538	0.0580	0.1072	0.3921	0.1759	**0.4108**	*0.3944*	0.3839	0.3644
p_{VIF}	1.5831	1.5831	*1.2980*	1.6742	1.5096	2.2023	1.9743	1.7688	**1.2923**
Output Data Size	(3264, 10)	(3264, 12)	(3264, 23)	(3264, 3)	(3264, 8)	(2958, 9)	(1935, 11)	(1835, 11)	(2894, 9)

Table 2: Comparison of Data Discovery Algorithms in Multi-Objective Setting



(a) T_3 Varying ϵ ($maxl$=6)

(b) T_3 Varying $maxl$ (ϵ=0.1)

(c) T_1 Varying $|adom|$ ($|A|$=11)

(d) T_1 Varying $|A|$ ($|adom|$=11)

Figure 9: Efficiency with Varying Factors

Accuracy vs. ϵ. Fixing maxl = 6, we varied ϵ from 0.5 to 0.1. As shown in Fig. **??**, MODis algorithms are able to improve the model in p_{acc} better with smaller ϵ, as they all ensure to output a ϵ-Pareto set that better approximate a Pareto set when ϵ is set to be smaller. In all cases, they achieve a relative improvement rImp(p_{Acc}) at least 1.07. BiMODis and NOBiMODis perform better in recognizing better solutions from both ends in reduction and augmentation as smaller ϵ is enforced. ApxMODis, with reduction only, is less sensitive to the change of ϵ due to that larger ϵ may "trap" it to local optimal sets from one end. Adding diversification (DivMODis) is able to strike a balance between ApxMODis and BiMODis by enforcing to choose difference datasets out of local optimal sets, thus improving ApxMODis for smaller ϵ.

Accuracy vs. maxl. Fixing ϵ = 0.1, we varied maxl from 2 to 6. Fig. **??** tells us that all MODis algorithms improves the accuracy of classification better for more rounds of path extension, as expected. Specifically, BiMODis and NOBiMODis benefit most as bi-directional search allow both to find better solution from wider search space as maxl become larger. ApxMODis and DivMODis are less sensitive, yet may due to different reason. ApxMODis is able to be robust in reporting dataset with similar quality due to localized search with reduction; and DivMODis, while *can* find the dataset with best result (*e.g.,* when maxl = 5), may "lose chance" to keep the one by replacing it with less optimal but more different counterparts when its search enlarges (when maxl = 6).

Using HF data, we report the impact of ϵ and maxl for Task 4 for regression analysis in Figs. **??** and **??**. The results are consistent with their counterparts in Figs. **??** and **??**. As ϵ varies from 0.1 to 0.5 (with maxl fixed as 6), BiMODis is able to generate datasets that improve p_{mse} of the regression by 1.5 times.

Exp-3: Efficiency. We next report the the efficiency of the MODis on generating datasets for task T_1 and task T_3 over OpenData and HF, respectively, and the impact of two major factors ϵ and maxl.

T_1: Varying ϵ. Fixing maxl = 4 and varying ϵ from 0.1 to 0.5, Fig. **??** verify the following. (1) BiMODis, NOBiMODis and DivMODis take less time to create datasets as ϵ becomes larger. As ϵ is larger, it provides more chance for bidirectional search to prune unnecessary valuation. DivMODis has a comparable performance with BiMODis, as it mainly benefits from the bi-directional implementation as BiMODis, hence exploits early pruning and incurs a small overhead in performing a stream-style placement strategy. (2) BiMODis, NOBiMODis and DivMODis is 2.5, 2 and 2.5 times faster than ApxMODis on average, respectively. ApxMODis takes most time in exploring a large universal table with a reduncton only strategy. It is not sensitive to ϵ, as its localized search from "data rich" end may be stablized at a local optimal ϵ-Pareto set that are less frequently updated.

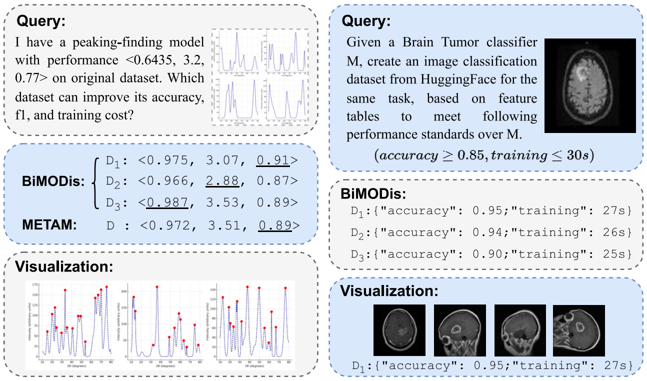
T_1: Varying maxl. Fixing ϵ = 0.2 and varying maxl from 2 to 6, Fig. **??** tells us that all MODis algorithms take longer time as maxl becomes larger. As expected, there are more states with non-ϵ-dominance relation to existing solution to be resolved, and more state nodes to be valuated. On the other hand, ApxMODis is the most sensitive to maxl due to rapid growth of search space. BiMODis is much less sensitive, as it mitigates the impact better due to bi-directional strategy and pruning.

T_3: Varying ϵ and maxl. We report the efficiency of our algorithms for regression task. Our observation is consistent with their counterparts for T_1. This verifies that the efficiency of our approach is not very sensitive to the type of learning tasks or models.

We have also investigated the impact of input data size. Due to limited space, we report the details in [2].

Exp-3: Case study. We next perform two real-world case studies to illustrate the application scenarios of our MODis methods. We anonymized the cases for this submission.

"Find data with models". A material science team has a trained random forest-based classifier to recognize peaks in a 2-dimensional angle-intensive X-ray diffraction data. They hope to find good additional datasets for which the model has improved performance



Query:

I have a peaking-finding model with performance <0.6435, 3.2, 0.77> on original dataset. Which dataset can improve its accuracy, f1, and training cost?

```
           D_1: <0.975, 3.07, 0.91>
BiMODis:   D_2: <0.966, 2.88, 0.87>
           D_3: <0.987, 3.53, 0.89>
METAM:     D : <0.972, 3.51, 0.89>
```

Visualization:

Query:

Given a Brain Tumor classifier M, create an image classification dataset from HuggingFace for the same task, based on feature tables to meet following performance standards over M.

$(accuracy \geq 0.85, training \leq 30s)$

BiMODis:

```
D_1:{"accuracy": 0.95;"training": 27s}
D_2:{"accuracy": 0.94;"training": 26s}
D_3:{"accuracy": 0.90;"training": 25s}
```

Visualization:

```
D_1:{"accuracy": 0.95;"training": 27s}
```

Figure 10: Case 1 (right): Recommending Datasets for Materials Peak Classification Model. Case 2 (left): Test Data Generation for Model Performance Benchmarking

in terms of F_1 score, training cost, and accuracy for downstream fine-tuning. The team has uploaded their original X-ray datasets and models to a crowd-sourced X-ray data collection platform we have deployed, with a performance vector $< 0.6435, 3.2, 0.77 >$. Over a set of shared X-ray diffraction datasets from other facilities, BiMODis created three new datasets $\{D_1, D_2, D_3\}$, which optimizes the model in each measure to 0.975, 3.07, and 0.89, respectively. A test of the model over the datasets is illustrated in Fig. **??**, which is manually validated to be accurate with ground-truth given by a database provided by a third-party international institution.

Generating Test data for Model Evaluation. In the second case study, we show that our configurable MODis paradigm readily fits the need for generating data (from scratch) for model benchmarking with required performances [34]. Given a trained scientific image classification model from Kaggle, we processed a pool of image features $\mathcal{D}$ from HF with 75 tables and in total 768 columns and more than 1000 rows as data sources. We set the ranges of training time requirement to be "accuracy > 0.85", "training cost" <30s, By setting these as *hard constraints* for BiMODis, we found that it outputs a set of 5 datasets within 30 seconds, over which the model's record on each measure is 0.95 in accuracy and 27s in training costs. This maps to a set of test images that serve as better training or testing data, with an example illustrated in Fig. **??**.

9 Conclusion

We have introduced a novel problem of multi-objective data discovery (MODis). We formalized a class of data discovery system and their computation in terms of finite state transducers, and verified their properties in terms of terminating, confluence and expressiveness. We then introduced three feasible algorithms to approximate Pareto set of datasets with different search strategies, and show that they preserve the provable sub-optimality in terms of ϵ-Parato set. Our experimental study has verified their feasibility in creating new datasets to improve multiple measurements for model performance. One future topic is to study the incremental maintenance of the datasets upon changes of the source datasets. Another topic is to develop effective parallel algorithms for distributed data discovery.

References

[1] U.S. General Services Administration. 2023. Data.gov. https://www.data.gov/

[2] Anonymous. 2023. Full Version. https://anonymous.4open.science/w/modis_full/full.pdf

[3] Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *The World Wide Web Conference*. 1365–1375.

[4] Amit Chakrabarti and Sagar Kale. 2015. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming* (2015).

[5] AnHai Doan, Alon Halevy, and Zachary Ives. 2012. *Principles of data integration.* Elsevier.

[6] Marwa El Halabi, Slobodan Mitrović, Ashkan Norouzi-Fard, Jakab Tardos, and Jakub Tarnawski. 2020. Fairness in Streaming Submodular Maximization: Algorithms and Hardness. In *Proceedings of the 34th International Conference on Neural Information Processing Systems.*

[7] European Organization For Nuclear Research and OpenAIRE. 2013. Zenodo. https://doi.org/10.25495/7GXK-RD71

[8] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée Miller. 2022. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *arXiv preprint arXiv:2210.01922* (2022).

[9] Sainyam Galhotra, Yue Gong, and Raul Castro Fernandez. 2023. METAM: Goal-Oriented Data Discovery. *arXiv preprint arXiv:2304.09068* (2023).

[10] Christian Glaßer, Christian Reitwießner, Heinz Schmitz, and Maximilian Witek. 2010. Approximability and hardness in multi-objective optimization. In *Programs, Proofs, Processes: 6th Conference on Computability in Europe, CiE 2010, Ponta Delgada, Azores, Portugal, June 30–July 4, 2010. Proceedings 6.* 180–189.

[11] H2O.ai. 2022. *H2O: Scalable Machine Learning Platform.* https://github.com/h2oai/h2o-3 version 3.42.0.2.

[12] José Hernández-Orallo, Wout Schellaert, and Fernando Martínez-Plumed. 2022. Training on the test set: Mapping the system-problem space in AI. In *AAAI*, Vol. 36. 12256–12261.

[13] Hugging Face AI 2023. Hugging Face – The AI Community Building the Future. https://huggingface.co/

[14] Kaggle. 2023. Kaggle: Your Home for Data Science. https://www.kaggle.com/

[15] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J Miller. 2022. Integrating Data Lake Tables. *Proceedings of the VLDB Endowment* 16, 4 (2022), 932–945.

[16] Ron Kohavi and George H John. 1997. Wrappers for feature subset selection. *Artificial intelligence* 97, 1-2 (1997), 273–324.

[17] Mina Konakovic Lukovic, Yunsheng Tian, and Wojciech Matusik. 2020. Diversity-guided multi-objective bayesian optimization with batch evaluations. *Advances in Neural Information Processing Systems* 33 (2020), 17708–17720.

[18] Hsiang-Tsung Kung, Fabrizio Luccio, and Franco P Preparata. 1975. On finding the maxima of a set of vectors. *J. ACM* 22, 4 (1975), 469–476.

[19] Yan-Nei Law, Haixun Wang, and Carlo Zaniolo. 2004. Query languages and data models for database sequences and data streams. In *VLDB*. 492–503.

[20] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *PODS*.

[21] Guoliang Li. 2017. Human-in-the-loop data integration. *Proceedings of the VLDB Endowment* 10, 12 (2017), 2006–2017.

[22] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. 2017. Feature selection: A data perspective. *ACM computing surveys (CSUR)* 50, 6 (2017), 1–45.

[23] Andre KY Low, Flore Mekki-Berrada, Aleksandr Ostudin, Jiaxun Xie, Eleonore Vissol-Gaudin, Yee-Fun Lim, Abhishek Gupta, Qianxiao Li, Yew Soon Ong, Saif A Khan, et al. 2023. Evolution-guided Bayesian optimization for constrained multi-objective optimization in self-driving labs. (2023).

[24] Yao Lu, Aakanksha Chowdhery, Srikanth Kandula, and Surajit Chaudhuri. 2018. Accelerating machine learning inference with probabilistic predicates. In *Proceedings of the 2018 International Conference on Management of Data.* 1493–1508.

[25] Jianyu Miao and Lingfeng Niu. 2016. A survey on feature selection. *Procedia computer science* 91 (2016), 919–926.

[26] Xuan Vinh Nguyen, Jeffrey Chan, Simone Romano, and James Bailey. 2014. Effective global approaches for mutual information based feature selection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* 512–521.

[27] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. Challenges in deploying machine learning: a survey of case studies. *Comput. Surveys* 55, 6 (2022), 1–29.

[28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[29] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27, 8 (2005), 1226–1238.

[30] Yuji Roh, Geon Heo, and Steven Euijong Whang. 2019. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering* 33, 4 (2019), 1328–1347.

[31] Darius Šidlauskas and Christian S Jensen. 2014. Spatial joins in main memory: Implementation matters! *Proceedings of the VLDB Endowment* 8, 1 (2014), 97–100.

[32] Tom F Sterkenburg and Peter D Grünwald. 2021. The no-free-lunch theorems of supervised learning. *Synthese* 199, 3-4 (2021), 9979–10015.

[33] George Tsaggouris and Christos Zaroliagis. 2009. Multiobjective optimization: Improved FPTAS for shortest paths and non-linear objectives with applications. *Theory of Computing Systems* 45, 1 (2009), 162–186.

[34] Margarida LC Vicente, José FO Granjo, Ruomu Tan, and Franz D Bähner. 2022. A Benchmark Model to Generate Batch Process Data for Machine Learning Testing and Comparison. In *Computer Aided Chemical Engineering*. Vol. 51. Elsevier, 217–222.

[35] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*. 12133–12143.

[36] Huaqing Zhang, Jian Wang, Zhanquan Sun, Jacek M Zurada, and Nikhil R Pal. 2019. Feature selection for neural networks using group lasso regularization. *IEEE Transactions on Knowledge and Data Engineering* 32, 4 (2019), 659–673.

[37] JH Zheng, YN Kou, ZX Jing, and QH Wu. 2019. Towards many-objective optimization: Objective analysis, multi-objective optimization and decision-making. *IEEE Access* 7 (2019), 93742–93751.

[38] Patrick Ziegler and Klaus R Dittrich. 2007. Data integration—problems, approaches, and perspectives. In *Conceptual modelling in information systems engineering*. Springer, 39–58.

Appendix A: Proofs and Algorithms

Proof of Proposition 3.3 *A data discovery system $\mathcal{T}$ can be configured to express (1) a data augmentation system, and (2) feature selection.*

Proof: □

Proof of Lemma 3.4 *The operators of type* $\oplus$ *and* $\ominus$ *are non-blocking.*

Proof: □

Proof of Proposition 3.5 Given a set of operators $\mathcal{O}$ with non-blocking operators $\oplus$ and $\ominus$, Type 3, Type 4, and Type 5 systems have the same expressiveness. *i.e.,* $L(\mathcal{T}_3) = L(\mathcal{T}_4) = L(\mathcal{T}_5)$.

Proof: □

Proof of Proposition 3.6 Given a configuration C with non-blocking operators of types $\{\oplus, \ominus\}$, and a data discovery system $\mathcal{T}$, the running of $\mathcal{T}$ is terminating and confluent.

Proof: □

Proof of Theorem 5.1 Given datasets $\mathcal{D}$, configuration C, and a number N, there exists an (N, ϵ)-approximation for MODis in $O\left(\min(N_u^{|R_u|}, N) \cdot \left(\left(\frac{\log(p_m)}{\epsilon}\right)^{|\mathcal{P}|-1} + I\right)\right)$ time, where $|R_u|$ is the total number of attributes from $\mathcal{D}$, $N_u = |R_u|+|\text{adom}_m|$ with adom_m the largest active domain, $p_m = \max \frac{p_u}{p_l}$ as p ranges over $\mathcal{P}$; and I the valuation cost per test.

Proof: □

Proof of Lemma 5.3 *If $|\mathcal{D}_S|$ is in $O(f(|D_U|))$ with f a polynomial, then the algorithm* ApxMODis *is a fully polynomial time approximation scheme (FPTAS) for* MODis.

Proof: □

Proof of Lemma 6.2 *For any feasible states $s \in \mathcal{Q}_f$, $s' \in \mathcal{Q}_b$, if (1) $s \succeq_\epsilon s'$ or $s' \succeq_\epsilon s$, (2) $D_{s'} \subseteq D_{s''} \subseteq D_s$, then $D_{s''} \notin \mathcal{D}_F$.*

Proof: □

Proof of Lemma 7.1 DivMODis *achieves a $\frac{1}{4}$-approximation for $Div(\cdot)$ of* MODis *problem with diversification data discovery.*

Proof: We here prove that DivMODis achieves "anytime" $\frac{1}{4}$-approximation for the following problem dataset selection problem. Considering a stream of datasets, where a dataset $\mathcal{D} \in \mathcal{D}^F$ arrives at any time t. At any time t,

□

Complete version of BiMODis

Algorithm 3 Complete BiMODis

```
1:  Input: Configuration $C = (s_U, \mathcal{O}, M, T, \mathcal{E})$, Records $Rec$, a constant $\epsilon > 0$;
2:  Output: $\epsilon$-Pareto set $\mathcal{D}_F$.
3:  Set $\mathcal{D}_F := \emptyset$, BIB $:= \emptyset$, PrunS $:= \emptyset$; $s_b$ = BackSt$(s_E, s_U)$;
4:  queue $\mathcal{Q}_f := \{(s_U, 0)\}$, queue $\mathcal{Q}_b := \{(s_b, 0)\}$;
5:  $\mathcal{D}_F^0[pos(\rho_{s_U})]$ = CorrFP$(s_U, Rec, \mathcal{E})$;
6:  $\mathcal{D}_F^0[pos(\rho_{s_b})]$ = CorrFP$(s_b, Rec, \mathcal{E})$;
7:  while $\mathcal{Q}_f \neq \emptyset$, $\mathcal{Q}_b \neq \emptyset$ and $\mathcal{Q}_f \cap \mathcal{Q}_b = \emptyset$ do
8:      $(s', d) = \mathcal{Q}_f$.dequeue(), ${\mathcal{D}_F}^{d+1} = {\mathcal{D}_F}^d$;  ▷ Forward Serach
9:      for all $s \in$ OpGen ($s'$, ‘F’) do
10:         $\mathcal{P}_s$ = CorrFP$(s, Rec, \mathcal{E})$; set $\rho_s$ with $\mathcal{P}_s$;
11:         if $pos[\rho_s] \in$ PrunS then continue;
12:         pruned = False
13:         for bound in SandwBs do
14:             pruned = SandwPrun($\rho_s$, bound, SandwBs)
15:             if pruned then break;
16:         pruned = UPareto ($\mathcal{D}_F^{d+1}$, PrunS, pos[$\rho_s$], $\epsilon$)
17:         if not pruned then $\mathcal{Q}_f$.enqueue((s, d+1))
18:     $(s', d) = \mathcal{Q}_b$.dequeue(), ${\mathcal{D}_F}^{d+1} = {\mathcal{D}_F}^d$; ▷ Backward Serach
19:     for all $s \in$ OpGen ($s'$, ‘B’) do
20:         same with line 10 to 16 in Forward Search
21:         if not pruned then $\mathcal{Q}_b$.enqueue((s, d+1))
22: procedure CorrFP($s$, $Rec$, $\mathcal{E}$)
23:     Build $G_c$ for measures recorded in $Rec$;
24:     StrongRs = GetSR($G_c$)
25:     if s in $Rec$.keys() then                                   ▷ Case 1: By $Rec$
26:         $\mathcal{P}_s = Rec[s]$;
27:         if $|valid(\mathcal{P}_s)| \geq 0.8|\mathcal{P}_s|$ then return $\mathcal{P}_s$;
28:     for all missing $p_i^s$ in $\mathcal{P}_s$ do                       ▷ Case 2: By $G_c$
29:         if $(p_i, p_j) \in$ StrongRs and $p_j^s \in Rec[s]$ then
30:             find closed $p_j^l$ and $p_j^u$ with $p_j^s$ in $Rec$;
31:             $p_i^s = (p_i^l + p_i^u)/2$
32:     if $|valid(\mathcal{P}_s)| < 0.8|\mathcal{P}_s|$ then                    ▷ Case 3: By $\mathcal{E}$
33:         fill missing $p_s \in \mathcal{P}_s$ by invoking $\mathcal{E}$; update $Rec[s] = \mathcal{P}_s$
34:     return $\mathcal{P}_s$;
```

Appendix B: Additional Experimental Results

Update in revision.