Computational Estimation by Scientific Data Mining with Classical Methods to Automate Learning Strategies of Scientists

APARNA S. VARDE, Department of Computer Science, Montclair State University, Visiting Researcher, Max Planck Institute for Informatics

Experimental results are often plotted as 2-dimensional graphical plots (aka graphs) in scientific domains depicting dependent versus independent variables to aid visual analysis of processes. Repeatedly performing laboratory experiments consumes significant time and resources, motivating the need for computational estimation. The goals are to estimate the graph obtained in an experiment given its input conditions, and to estimate the conditions that would lead to a desired graph. Existing estimation approaches often do not meet accuracy and efficiency needs of targeted applications. We develop a computational estimation approach called AutoDomainMine that integrates clustering and classification over complex scientific data in a framework so as to automate classical learning methods of scientists. Knowledge discovered thereby from a database of existing experiments serves as the basis for estimation. Challenges include preserving domain semantics in clustering, finding matching strategies in classification, striking a good balance between elaboration and conciseness while displaying estimation results based on needs of targeted users, and deriving objective measures to capture subjective user interests. These and other challenges are addressed in this work. The AutoDomainMine approach is used to build a computational estimation system, rigorously evaluated with real data in Materials Science. Our evaluation confirms that AutoDomainMine provides desired accuracy and efficiency in computational estimation. It is extendable to other science and engineering domains as proved by adaptation of its sub-processes within fields such as Bioinformatics and Nanotechnology.

CCS Concepts: • **Information systems → Data management systems**; • **Computing methodologies → Knowledge representation and reasoning**; **Supervised learning by classification**; **Cluster analysis**; **Feature selection**; • **Applied computing → Computers in other domains**; **Physical sciences and engineering**;

Additional Key Words and Phrases: Applied research, clustering, classification, domain knowledge, estimation, graphical data mining, machine learning, predictive analytics, scientific applications

The author Dr. Aparna Varde at Montclair State University is currently supported by grants from NSF "MRI: Acquisition of a High-Performance GPU Cluster for Research and Education" Award Number 2018575, and "MRI: Acquisition of a Multimodal Collaborative Robot System (MCROS) to Support Cross-Disciplinary Human-Centered Research and Education at Montclair State University", Award Number 2117308. She is a visiting researcher in the Databases and Information Systems group of Dr. Gerhard Weikum at the Max Planck Institute for Informatics, Saarbrücken, Germany, during her sabbatical from Montclair State University. The valuable contributions of Prof. Elke Rundensteiner, Prof. David Brown and Prof. Richard D. Sisson Jr. from Worcester Polytechnic Institute, Massachusetts, are gratefully acknowledged on significant work related to this journal article. A part of this work has been supported by an esrtwhile grant from the Department of Energy – Industrial Technology Program (DOE-ITP) Award DE-FC-07-01ID14197. The author thanks the funding sources.

Author's address: A. S. Varde, Department of Computer Science, Montclair State University, NJ, 07043, Visiting Researcher, Max Planck Institute for Informatics, 66123 Saarbrücken, Germany; email: vardea@montclair.edu.

1556-4681/2022/03-ART86 $15.00
https://doi.org/10.1145/3502736

ACM Reference format:
Aparna S. Varde. 2022. Computational Estimation by Scientific Data Mining with Classical Methods to Automate Learning Strategies of Scientists. *ACM Trans. Knowl. Discov. Data* 16, 5, Article 86 (March 2022), 52 pages.
https://doi.org/10.1145/3502736

1 INTRODUCTION

Background and Motivation: Experimental results of processes in scientific domains such as Materials Science, Physics, Chemistry, Biology, and Geoscience can be depicted as graphical plots [2, 13, 33, 55, 68]. These, also known as *graphs*, are 2-dimensional plots of dependent versus independent variables often modeling the behavior of processes. They serve as good visual tools for analysis. However, performing laboratory experiments to plot such graphs consumes significant time and resources.

We present a motivating example from the domain of Heat Treating of Materials that inspired this research. Heat Treating is a field in Materials Science that involves the controlled heating and rapid cooling of a material in a liquid or gas medium to achieve desired mechanical and thermal properties [37, 55, 57]. Consider the laboratory experiment shown in Figure 1. This depicts experimental data, i.e., input conditions and the resulting graph during a process called *quenching*, namely, the rapid cooling step in heat treatment. The input conditions in this experiment such the Quenchant Name (cooling medium) and Part Material are the parameters of the experimental setup used in quenching. The result of the experiment is plotted as a graph called a heat transfer curve, namely, the heat transfer coefficient h on the y-axis versus temperature T on the x-axis. The heat transfer coefficient, a parameter measured in Watt/m^{2}K, characterizes the experiment by representing how the material reacts to rapid cooling. Scientists are interested in analyzing this graph to assist decision-making for various activities such as designing the setup in the corresponding real processes. For instance, in the material ST4140, a type of steel, heat transfer coefficient curves with steep slopes imply fast heat extraction capacity. The corresponding input conditions could be used to treat this steel in an industrial application that requires such a capacity. Performing such an experiment in the laboratory takes approximately 5 hours and the involved resources incur a capital investment of thousands of dollars with recurring costs worth hundreds of dollars. This is clearly a huge expense especially since experiments are performed frequently to conduct studies.

Problem Definition: It is desirable to estimate the graph in an experiment given its input conditions. Conversely, given a desired graph, it is useful to estimate the conditions to achieve it. This inspires the development of a computational estimation approach with the following goals.

(1) Given some or all of the input conditions of an experiment, display the most likely resulting graph.
(2) Given the desired graph in an experiment or ranges describing its features, determine the most appropriate input conditions to achieve it.

The estimation is to be performed under the assumption that the input conditions and graphs of prior experiments are stored in a database. Approaches such as mathematical modeling of process parameters and similarity search over existing data are not found satisfactory, due to insufficient formulae available and not enough use of domain knowledge [19, 39, 43]. Neural models based on the original **artificial neural networks** (**ANN**) [39, 52] and deep learning advances such as **convolutional neural networks** (**CNN**) [17, 29] are not recommended here since it important to decipher the actual learning strategies of scientists, e.g., the flow from input conditions to the

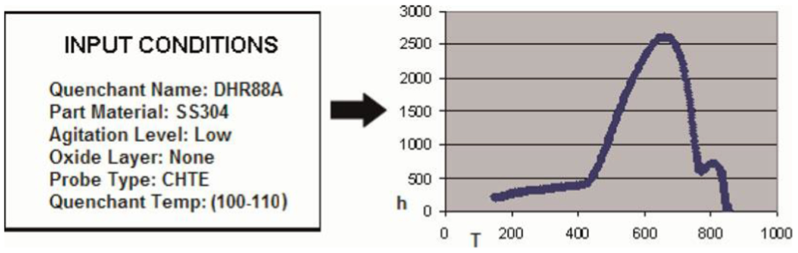


Fig. 1. Example of scientific experimental data (input conditions and graphical plots).

experimental results. Since a neural network is basically a black box, such reasoning is not facilitated via its deployment due to lack of clear interpretability. Hence, other methods are needed. All this background is evident from a study of the literature and discussion with domain experts. There is a need to perform the estimation with: accuracy acceptable for targeted applications; distinctly less time than a laboratory experiment; minimal manual intervention; and clear decipherable, comprehensible results for explicability.

Solution Approach: We develop an approach called AutoDomainMine as a solution to the given computational estimation problem. AutoDomainMine is based on a graphical data mining framework that constitutes mining stored graphs from existing experiments to discover knowledge for estimation. This framework integrates the classical data mining methods of clustering and classification into a unique framework to perform knowledge discovery over existing graphical data as follows. The data from existing experiments has been stored in a database in the form of input conditions and the corresponding graph obtained. As a first step in AutoDomainMine, existing experiments are clustered based on their graphs. Decision tree classification is then used to learn the clustering criteria and the learned criteria are applied to build representative pairs of input conditions and graph per cluster. These representatives along with the clustering criteria learned through decision trees serve as the domain knowledge discovered from existing experiments. This discovered knowledge then forms the basis for estimation as follows. Given a new set of input conditions, the closest matching decision tree path is found to estimate the cluster of the experiment. The representative graph of that cluster serves as the estimated graph for the new experiment. Given a desired graph, the closest matching representative graph is found by comparison with graphs. The corresponding representative conditions are offered as the estimated input conditions that would obtain the desired graph. This approach is based on automating typical learning methods of scientists from the targeted domains. A significant challenge in AutoDomainMine is learning a domain-specific notion of distance that adequately captures semantics for clustering graphs. This is addressed through our proposed technique called LearnMet that will be elaborated in Section 3. Another major challenge is designing semantics-preserving representative pairs of input conditions and graphs as the output of classification, to be used during estimation. This is addressed by our proposed methodology called DesRept that is outlined in detail in Section 4. Likewise, other issues encountered along with their proposed solutions are described in the respective sections.

Evaluation: The AutoDomainMine approach has been used to develop a software system that serves as a computational estimation tool. This is evaluated rigorously with real data in the Heat Treating domain that motivated this work. Targeted applications such as parameter selection and expert systems are used for evaluation. The estimation provided by AutoDomainMine is compared with the real laboratory experiments not used for training the technique. If the real output completely matches the estimation, then the estimation is considered to be accurate. Also, it is required that the estimation should take distinctly less time than a laboratory experiment. This evaluation is conducted by targeted users through surveys and via the holdout strategy. AutoDomainMine provides extremely high accuracy and efficiency ensuring user satisfaction. It does not require domain expert intervention during estimation and has comprehensible, explainable results.

Contributions: The contributions of this research fall in 3 broad categories: (1) AutoDomainMine framework of integrating clustering and classification; (2) LearnMet technique to learn domain-specific distance metrics for graphs; (3) DesRept methodology to design semantics-preserving representatives. The AutoDomainMine approach is published as a short poster paper in AAAI [63] and a demo paper in SIGMOD [67]. However, these papers present a panoramic view of the proposed technique and its system demonstration respectively.

In this journal article, we delve deeper into the AutoDomainMine approach, describing all its steps in the knowledge discovery and estimation processes. We explain how this approach automates classical learning methods of scientists by integrating clustering and classification for knowledge discovery. Furthermore, we focus on the details of the clustering and classification steps, along with their respective sub-tasks. We address issues such as clustering over curves instead of points, and finding approximate matches in decision tree classification for the purpose of estimation. The LearnMet and DesRept techniques, as contributions of this overall effort, have received attention as in the data mining community via papers [64–66]. However, these papers focus on the individual proposal of the respective techniques in distance metric learning and semantics-preserving representative design. In this article, as we describe the details of LearnMet and DesRept, we also explain how these two techniques fit into the big picture of AutoDomainMine, thus emphasizing their use in the overall problem of computational estimation. We explain in detail how exactly AutoDomainMine uses the knowledge discovered through the integrated framework of clustering and classification, after incorporating the outcomes of LearnMet and DesRept, for the purpose of conducting the estimation. We present the stepwise approach involved in estimating a graph that would result from an experiment given its input conditions, and estimating the conditions that would produce a desired graph.

It is to be noted that this work is scalable to various domains as evident from the adaptation of its sub-processes in fields such as Bioinformatics, Nanotechnology, and others [10, 62] with interesting and useful results from the accuracy, efficiency, and interpretability standpoints.

Overall, the contributions of this work can be emphasized in terms of its specific advantages as well as the differences between existing methods as follows.

- This work automates a typical learning strategy of scientists via an integrated framework of data mining techniques and is among the first ones to achieve that, to the best of our knowledge.
- The computational estimation in our proposed approach AutoDomainMine obtains the desired levels of accuracy acceptable for targeted applications.
- The estimation is performed in distinctly less time than a laboratory experiment, thereby saving significant time and resources.
- There is minimal manual intervention required in the implementation of AutoDomainMine and its sub-processes of LearnMet and DesRept.
- The approaches in this work can proceed without requiring the usage of prior mathematical models with explicit formulas for conducting simulations.
- Domain knowledge is incorporated within the processes in order to make them more meaningful with respect to the computational estimation.
- The results are clearly decipherable and easily comprehensible, therefore catering to the needs of interpretability as well as explicability for the users.

Hence, this work has advantages over lengthy simulations that can be very time-consuming; it is better than similarity search which does not provide sufficient accuracy; it has an edge over mathematical modeling for which the required formulas are not available; and it is more beneficial than neural networks since the latter constitute a black box with lack of interpretability. These advantages are highlighted further in the evaluation. Furthermore, this work harnesses domain knowledge, yet requiring minimal manual intervention and also automates a typical learning strategy of scientists, providing added advantages from pedagogical as well as application standpoints, e.g., with respect to usefulness in developing intelligent tutors and expert systems.

Layout of the Article: Section 2 of this article describes the overall approach, AutoDomainMine, namely its knowledge discovery process that integrates clustering and classification, and its estimation process that uses the discovered knowledge to perform the required estimation. Sections 3 and 4 give details of clustering and classification along with LearnMet and DesRept, respectively. Section 5 describes the steps of the process of estimation in AutoDomainMine using the discovered knowledge. Section 6 presents the performance evaluation. Section 7 overviews related work while Section 8 gives the conclusions.

2 COMPUTATIONAL ESTIMATION APPROACH: AUTODOMAINMINE

The approach developed for computational estimation, called AutoDomainMine, constitutes a unique framework of employing both clustering and classification in a collaborative fashion over graphical data in order to learn from the results of existing experiments. This is analogous to the manner in which scientists often reason.

2.1 Learning Methods of Scientists

From a detailed study of the relevant literature [37, 55, 57], and discussions with domain experts, it has been noticed that researchers in scientific domains such as Materials Science often use the following learning methods to discover certain facts from experiments to propound hypotheses empirically. They group experiments based on similarity of results and then reason about causes of similarity between groups based on input conditions of experiments. For example, *a hypothesis propounded empirically was that a thin oxide layer on the surface of a part causes fast cooling while a thick oxide layer causes slow cooling*. This hypothesis was learned by conducting experiments with thin and thick oxide layers as input conditions, respectively, all other criteria being the same. The results showed that for all experiments with thin oxide layer the cooling was fast while for those with thick oxide layer it was slow. Experts then reasoned further on the basis of existing domain knowledge that the thin oxide probably caused the vapor blanket around a part to break, thus resulting in fast cooling; while thick oxide acted as an insulator, resulting in slower cooling. In short, this scientific understanding was gained by grouping experiments based on similarity of their results and reasoning based on their input conditions in order to put forth the concerned hypothesis empirically [37].

In this work, we postulate that such empirical learning by *grouping* and *reasoning* is analogous to the data mining techniques of *clustering* and *classification,* respectively. This is because *clustering* involves *grouping* objects based on their similarity while *classification* serves to identify the target class of an object, which is often done by *reasoning* about its attributes using a symbolic method such as a decision tree. We thus set out to automate these learning methods of scientists by integrating clustering and decision tree classification into a learning strategy for knowledge discovery in AutoDomainMine.

Accordingly, the AutoDomainMine approach involves a one-time process of knowledge discovery from existing data by integrating clustering and decision tree classification, and a recurrent process of using the discovered knowledge for performing each estimation. The overall AutoDomainMine approach is illustrated in Figure 2, and is explained further next.

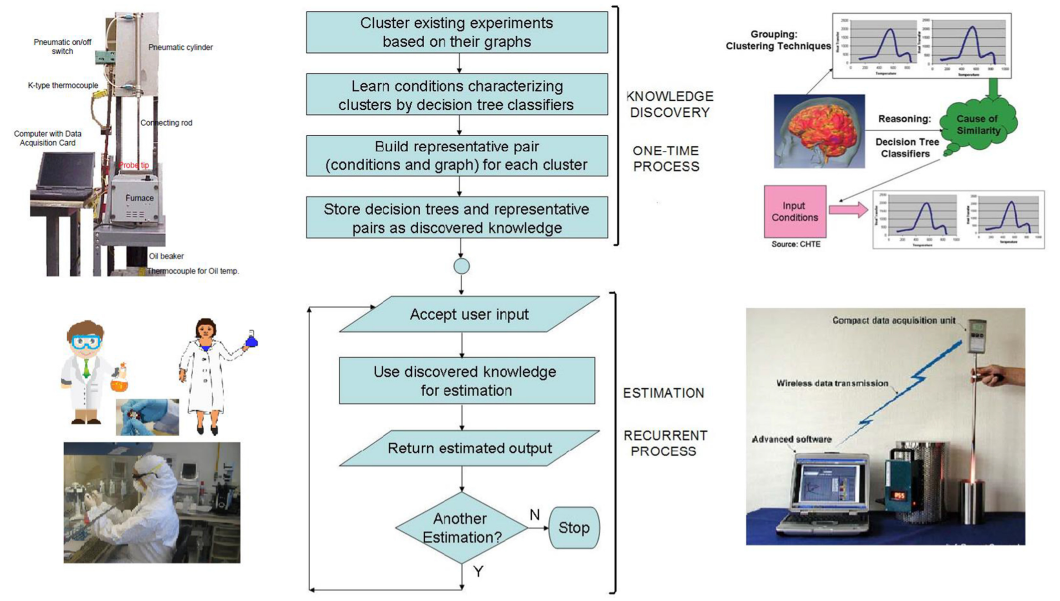

Pneumatic on/off switch
Pneumatic cylinder
K-type thermocouple
Connecting rod
Computer with Data Acquisition Card
Probe tip
Furnace
Oil beaker
Thermocouple for Oil temp.
Cluster existing experiments based on their graphs
Learn conditions characterizing clusters by decision tree classifiers
Build representative pair (conditions and graph) for each cluster
Store decision trees and representative pairs as discovered knowledge
KNOWLEDGE DISCOVERY
ONE-TIME PROCESS
Grouping: Clustering Techniques
Reasoning:
Cause of Similarity
Decision Tree Classifiers
Input Conditions
Source: CHTE
Accept user input
Use discovered knowledge for estimation
Return estimated output
Another Estimation?
N
Stop
Y
ESTIMATION
RECURRENT PROCESS
Compact data acquisition unit
Wireless data transmission
Advanced software

Fig. 2. AutoDomainMine approach in a nutshell—Simulating the learning strategies of scientists.

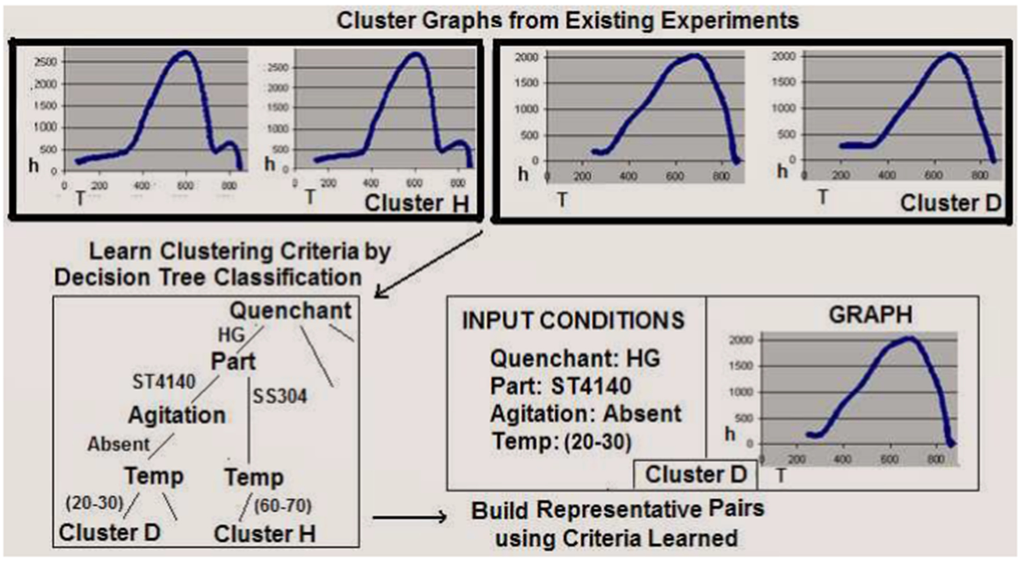


Fig. 3. Knowledge discovery process in AutoDomainMine.

2.2 Knowledge Discovery in AutoDomainMine

The process of knowledge discovery is a one-time operation that is performed over data from existing experiments stored in a database. This process is depicted in Figure 3.

Clustering is first performed over the graphical results of the existing experiments. Since clustering techniques originally developed for points [19], a mapping is proposed that converts a 2-dimensional graph into an n-dimensional point. A suitable notion of distance for clustering is

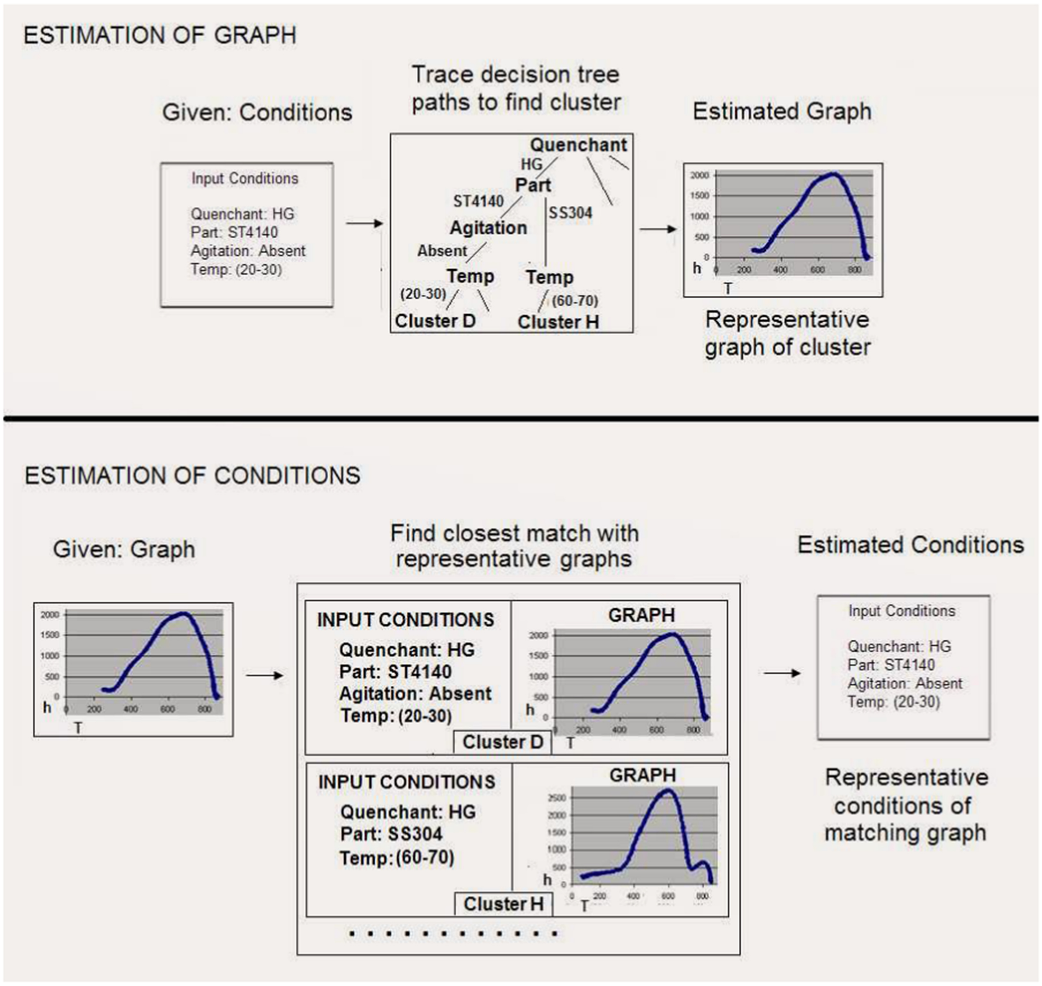


Fig. 4. Estimation process in AutoDomainMine.

learned by incorporating domain knowledge as using our proposed approach called LearnMet (elaborated in Section 3). Once the clusters of experiments are identified by grouping their graphs, the clustering criteria, i.e., input conditions that characterize each cluster are learned by decision tree classification. This helps understand the relative importance of the conditions in clustering. The paths of each decision tree are then traced to build a representative pair of input conditions and graph per cluster. These cluster representatives are designed so as to capture semantics using our proposed approach DesRept (see Section 4). The decision trees and representative pairs form the discovered knowledge which is then used for estimation as follows.

2.3 Estimation in AutoDomainMine

The process of estimation is a recurrent operation that is performed using the discovered knowledge, each time the user submits input conditions to estimate the graph (or vice versa). The estimation process illustrated in Figure 4 with its 2 parts, namely, estimating the graph given the conditions, and estimating the conditions given the graph. These are explained as follows.

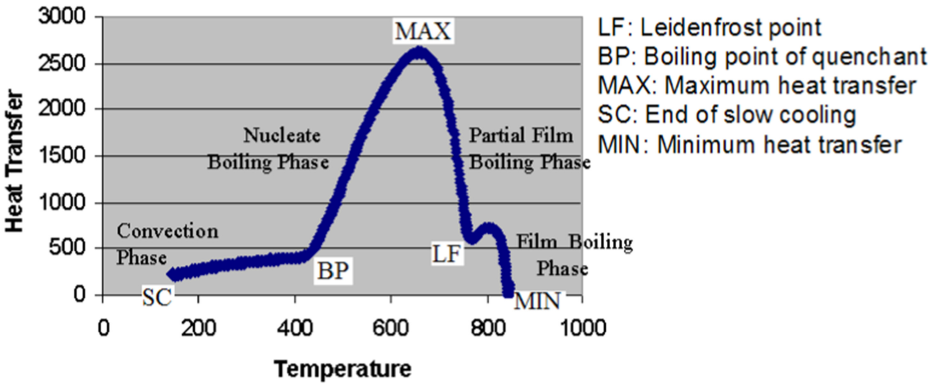


Fig. 5. Points to define distances on a type of graph in Heat Treating of Materials.

In order to estimate a graph, given a new set of input conditions, the decision tree is searched to find the closest matching cluster. The representative graph of that cluster is displayed as the estimated graph for the given set of conditions. To estimate input conditions, given a desired graph in an experiment, the representative graphs are searched to find the closest match using the learned notion of distance for the graphs from LearnMet (Section 3). The representative conditions designed using DesRept (Section 4) corresponding to the match are offered as the estimated input conditions that would likely produce the desired graph. Note that this estimation incorporates the relative importance of the conditions as identified from the decision tree.

3 DETAILS OF CLUSTERING IN AUTODOMAINMINE

The first step of the knowledge discovery process in AutoDomainMine is to cluster graphs obtained from existing experiments. Clustering is a technique that groups objects into classes such that objects within a class have high similarity but are different from items in other classes [19]. In AutoDomainMine, we can use any clustering algorithm, for instance the classical *k*-means [19]. Various issues involved in clustering are addressed next.

3.1 Mapping Graphs (Curves) to Points

Clustering techniques were originally developed for points [19]. In order to apply them to graphs that are curves, we propose a mapping from a 2-dimensional curve to a multi-dimensional point. Consider a 2-dimensional curve consisting of n points each having an x-coordinate and a y-coordinate. The n x-coordinates on the curve are mapped to n *dimensions*. The n y-coordinates on the curve are mapped to *values* along these n dimensions, respectively. The mapping is conducted such that the order is preserved thereby maintaining the sequential features on the 2-dimensional graph while converting it to a multi-dimensional point. For example, consider the graph shown in Figure 5 that represents a heat transfer coefficient curve. The n-dimensions corresponding to the temperature values (on the x-axis) would be mapped sequentially (due to which the SC feature would appear before the BP feature) and hence the corresponding heat transfer coefficient values (on the original y-axis) along those respective dimensions would also be preserved in order. Mathematically, this can be represented as $(x, y) \mapsto (dimension, value)$ where $\mapsto$ denotes "maps to". For example, a point $(200,\ 400)$ on the original curve can be mapped to value $(y = 400)$ on the $(x_{200})^{th}$ dimension. Likewise, each point on the curve would be mapped to a dimension and a value along that dimension. Likewise, after mapping each (x, y) point on the original 2-dimensional curve to a $(dimension, value)$ pair we effectively get an n-dimensional point. However, in theory, there are infinite points on any curve, thus posing the curse of dimensionality, the solution for which is addressed next.

3.2 Dimensionality Reduction

Since a curve has infinitely many points it is not feasible to convert each x-coordinate into a separate dimension. Hence, dimensionality reduction technology is needed to make our proposed mapping applicable. There are various methods of dimensionality reduction among which we found selective sampling and Fourier Transforms relevant to our problem, though other methods could also be chosen. We briefly describe these.

3.2.1 Selective Sampling. The method of Sampling involves considering a part, i.e., a sample, of the whole object (in this case, points on a graph) and using that for the purpose of analysis (in this case, clustering). In random sampling, points would be picked, or sampled, at arbitrary intervals. In selective sampling [19], points are chosen based on certain criteria. In our problem, we sample the points at regular intervals, and in addition sample critical points that correspond to significant features of the graph, thus making the sampling *selective*. Knowledge of the domain gathered from literature surveys and discussions with domain experts helps in determining the significant features on the graphs [37, 55, 57].

3.2.2 Fourier Transforms. A Fourier Transform decomposes a given waveform into sinusoids of different frequencies such that they sum back to the original waveform, e.g., as used in [1]. By retaining some of these sinusoids and discarding the rest, we can thus reduce the dimensionality of the original curve. In our AutoDomainMine approach, Fourier Transforms are used as follows. We first apply the equation below as found in suitable contexts in the literature [1] in order to map the n dimensions of the curve (n-dimensional point) into n Fourier Coefficients.

$$F(s) = \frac{1}{\sqrt{N}} \sum_{t=0}^{N-1} e^{-j2\pi f(t)/n}. \tag{1}$$

In this equation $F(s)$ refers to the frequency domain while $f(t)$ refers to the time domain. Each Fourier Coefficient corresponds to a different frequency. We then retain the Fourier Coefficients that are considered useful with respect to the domain. For example, in Heat Treating, the first 16 coefficients are considered to be useful as per domain knowledge. This is because the graphs (heat transfer curves) [57] are such that these coefficients representing lower frequency values contain useful information. The remaining coefficients representing higher frequency values are regarded as noise.

3.3 Notion of Distance

Clustering techniques group objects based on their similarity or distance from each other. Hence, in order to cluster the graphs, it is first important to define the notion of distance between them. This notion of distance should be such that it adequately preserves the semantics of the domain. It has been found from our preliminary analysis and discussions with experts that using the default Euclidean distance alone is not sufficient to capture domain knowledge. For example, some features on a graph could represent critical occurrences in the corresponding experimental results. Likewise, there can be statistical observations that are significant on the graphs. In terms of the absolute positions of the points, there could be some that are relatively more important than others. There are subtle factors that an expert would consider subjectively in an intuitive manner. These need to be captured through an objective distance function for use in computational processes such as clustering. Although a variety of distance metrics exist in the literature, it is seldom known which of them work best in terms of incorporating domain knowledge. It is thus desirable to conduct distance metric learning in order to define a domain-specific notion of distance for adequately clustering the graphs. This learning forms an important sub-problem of our research and is discussed next.

Before proceeding with the discussion of the distance metric learning strategy, we first state the following terminology on distance types in the literature.

Position-based Distances: These distances in the literature refer to distances based on the absolute position of the objects. Examples of position-based distances are Euclidean and Manhattan distances [19].

Statistical Distances: These refer to distances based on statistical observations [43]. Examples include Mean distance, Maximum distance, and Minimum distance, i.e., absolute differences between the mean values, maximum values, and minimum values of the objects, in this case, corresponding observations on the graph.

Critical Distances: In addition to standard distance types in the literature, we consider the concept of domain-specific *critical distances* as follows. Given two graphical plots A and B, we define a critical distance as the distance between critical regions of A and B where a critical region represents the occurrence of a significant physical phenomenon. Each such distance is calculated in a domain-specific manner. As examples of critical distances, we refer to the critical points on the graph in Heat Treating (heat transfer curve) in Figure 5. These are the Leidenfrost point LF, the Boiling Point BP, and the Slow Cooling Point SC.

Accordingly, the *Leidenfrost distance* is defined as the distance between the Leidenfrost points [57] on two heat transfer curves denoted as Δ_{LF}. For curves A and B this is calculated as follows.

$$\Delta_{LF}\,(A,B) = \sqrt{(A_{TLF} - B_{TLF}\,)^2 + (A_{hLF} - B_{hLF})^2} \tag{2}$$

Here, T_{LF} is the temperature at Leidenfrost Point and h_{LF} is the heat transfer coefficient at that point. The explanation for the calculation in Equation (2) is that the Leidenfrost Points on curves A and B are defined in terms of their heat transfer coefficients and temperatures, respectively. Hence, the distance between them is based on the distances between their respective heat transfer coefficients on the y-axis and the temperatures on the x-axis as plotted on the curves. Note that the Leidenfrost point is crucial in the domain of Heat Treating since it is the point at which the vapor blanket around a part breaks during *quenching* or rapid cooling, thus leading to the next stage of the overall quenching process [57]. Boiling Point and Slow Cooling distances are defined and calculated in a similar manner. Given this discussion, we now proceed to describe our proposed technique called LearnMet to learn a domain-specific notion of distance for graphs in scientific domains.

3.4 LearnMet: Domain-Specific Distance Metric Learning

A technique called LearnMet is designed as a solution for domain-specific distance metric learning. The input to LearnMet is a training set with actual (true, i.e., correct) clusters of graphs provided by domain experts. Its output is the learned distance metric for graphical plots incorporating domain semantics, and is called the *LearnMet distance*. The 5 basic steps of the LearnMet technique are stated below with details explained next.

(1) Assign an initial metric Δ_L as a weighted sum of distance metrics applicable to the domain.
(2) Use that metric Δ_L for clustering with an arbitrary but fixed clustering algorithm to get predicted clusters.
(3) Evaluate clustering accuracy by comparing predicted clusters with actual clusters.

(4) Adjust the metric Δ_L based on the error between the predicted and actual clusters, and re-execute the clustering and evaluation until error is below a threshold τ or maximum number of epochs ε is reached.
(5) Output the final metric Δ_L giving lowest error as the learned metric, i.e., LearnMet Distance.

Note that an epoch refers to a run of all the 5 steps of LearnMet. This learning is analogous to the fundamental principle of regression with gradient descent as typically used in a method such as backpropagation. It is adapted to learn domain-specific notions of distance. The details of each of these steps are explained below.

3.4.1 LearnMet Initial Metric Step. An initial distance metric Δ_L is selected as a weighted sum of components, where each component is an individual distance function such as Euclidean distance (of type position-based distance), Mean distance (of type statistical distance), Leidenfrost distance, and Boling Point distance (of type critical distance) and so on and the weight of each component is a numerical value denoting its relative importance. Thus, we have the following.

$$\Delta_L = \sum_{i=1}^{C} \omega_i \, \Delta_i \tag{3}$$

Here, each Δ_i is a component, ω_i is its weight, and C is number of components. The explanation for this Equation (3) is concerned with the relative importance of the components in the domain. Since there are multiple components, some of them are more significant than others, and this domain-specific significance is denoted by their respective weights. Thus, we propose this equation to capture domain knowledge. Domain experts are asked to identify components (i.e., distance metrics) applicable to the graphical plots that will serve as building blocks for the learning of a new metric. (Note that this is only a one-time process since we aim to have minimal manual intervention in our approach). If the experts have subjective notions about the relative importance of the components, this information is used to assign initial weights. If this relative importance is unknown then random weights are assigned to all components. Initial weights by default are assigned on a scale of 0 to 10.

3.4.2 LearnMet Clustering Step. The domain experts provide a set of actual clusters over a training set of graphs. To perform clustering, an arbitrary but fixed clustering algorithm such as k-means [19] is selected. The value of k is an experimental parameter that can be obtained using well-known approaches such as the elbow method often used in k-means. Using the equation above (Equation (3)) as the distance metric, k clusters are constructed using the selected algorithm, where k is the number of actual clusters in the training set. These clusters obtained using metric Δ_L are the predicted clusters.

3.4.3 LearnMet Cluster Evaluation Step. The cluster evaluation involves comparing the predicted and actual clusters with each other. Examples of predicted and actual clusters of graphs are shown in Figure 6.

Ideally, the predicted clusters should match the actual clusters perfectly. Any difference between them is considered an error. To compute this error, we consider pairs of graphical plots and introduce the following notation to depict the notion of correctness in the domain.

Notion of Correctness: Given a pair of graphs g_α and g_β we postulate the following.

- $(g_\alpha,\ g_\beta)$ is a True Positive (TP) pair if g_α and g_β are in the same actual cluster and in the same predicted cluster

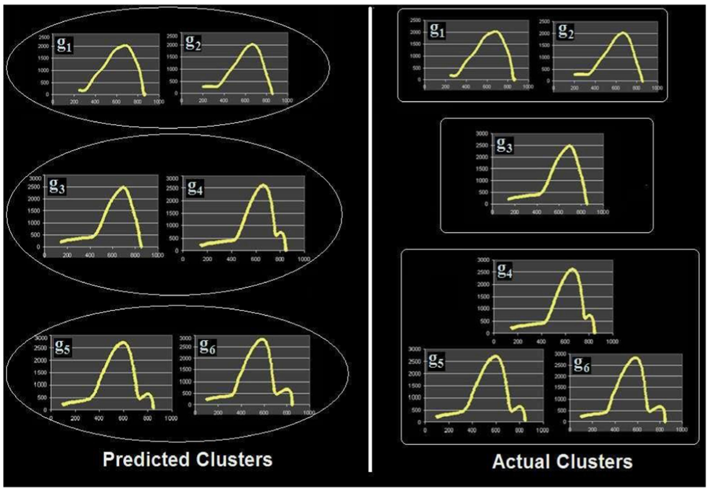


Fig. 6. Examples of predicted and actual clusters.

- (g_α, g_β) is a True Negative (TN) pair if g_α and g_β are in different actual clusters and in different predicted clusters
- (g_α, g_β) is a False Positive (FP) pair if g_α and g_β are in different actual clusters but in the same predicted cluster
- (g_α, g_β) is a False Negative (FN) pair if g_α and g_β are in the same actual cluster but in different predicted clusters

Figure 6 includes examples of such pairs: (g_1, g_2) is a true positive pair; (g_2, g_3) is a true negative pair; (g_3, g_4) is a false positive pair; and (g_4, g_6) is a false negative pair. The error measure of interest to us is called Failure Rate; it is adapted from the literature [39], and defined in our context below.

Success and Failure Rates: Let TP, TN, FP, and FN denote the number of true positive, true negative, false positive, and false negative pairs, respectively. Also let SR denote the Success Rate and FR denote the Failure Rate. Then, we have the following.

$$SR = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4}$$

$$FR = 1 - SR = \frac{FP + FN}{TP + TN + FP + FN} \tag{5}$$

In this problem, false positives and false negatives are equally undesirable. Hence, our calculation above weighs them equally. Based on this, the explanation for Equations (4) and (5) is somewhat analogous to the concept of the standard precision metric in data mining and machine learning [19, 39]. While precision considers the ratio of the true positives to the total number of positives, i.e., Precision = TP/(TP + FP), in our context we need to incorporate false negatives in exactly the same manner, because both of these constitute erroneous clustering. Hence, we propose one combined formula that measures the correctness in the clustering as defined by Success Rate. Conversely, Failure Rate is calculated as: 1 − Success Rate.

Overfitting: To avoid overfitting in LearnMet, we proffer an approach based on the realm of combinatorics in mathematics, often useful in machine learning [39]. Instead of using all pairs of graphs for evaluation, we propose that a subset of pairs is used called *ppe* or pairs per epoch. In each epoch, a randomly selected subset of pairs is used for evaluation and weight adjustment. Thus, there is significant randomization in every epoch due to multiple possible *combinations*. If there are γ number of graph-pairs, we have λ number of distinct learning-pairs, calculated as follows using the mathematical formula for combinations.

$$\lambda = \binom{\gamma}{ppe} = \frac{\gamma!}{ppe!\,(\gamma - ppe)!} \tag{6}$$

The explanation for Equation (6) follows from the combinatorics realm in mathematics. The term *combination* (as well known in mathematics) refers to the number of possible arrangements in a collection such that the order of the selection does not matter. In this case, the collection is total number of graph pairs γ from which we need to make ppe number of arrangements, i.e., we select ppe number of pairs from the total γ number of pairs which constitutes a combination. This is calculated using the standard formula for combinations, which gives Equation (6) in this context. For example, if $ppe = 15$, then for $\gamma = 50$ graphs we get a total of $\lambda = \binom{50}{15}$ which gives $\frac{50!}{15!(50-15)!}$ distinct pairs for learning [43]. This is clearly a very high number. Thus, in each epoch, if as few as 15 randomly selected pairs are used, it still gives a large number of epochs with many distinct pairs for learning. This incremental approach helps avoid overfitting. Additionally, the random seed is altered in the clustering algorithm in different epochs as an additional method to avoid overfitting.

Error Threshold: A domain-specific error threshold τ is the extent of error allowed per epoch, where error is measured by Failure Rate FR as calculated above (Equation (5)). If the error is below threshold τ then the final distance metric is returned. However, if the error is not below threshold in a given epoch, then the metric is adjusted based on this error as explained below.

3.4.4 *LearnMet Weight Adjustment Step.*

In order to proceed with the details of weight adjustment the following terminology related to distances is first explained. This is because the cause of the error can be traced to certain distances between pairs of graphs in the predicted and actual clusters.

Distance between a Pair of Graphical Plots: The distance $\Delta_L(g_\alpha, g_\beta)$ between a pair of graphs g_α and g_β is the weighted sum of components in the plots using metric Δ_L (Equation (3)). Given this, the concept of average distance between false positive and false negative pairs is introduced next.

Average False Negative and False Positive Distances: The distances ΔFN and ΔFP are defined as the average distance using the metric Δ_L of the false negative pairs and of the false positive pairs, respectively. These are as follows.

$$\Delta FN = \sum_{j=1}^{FN} \Delta_L(g_{\alpha FN}, g_{\beta FN}). \tag{7}$$

Here, $(g_{\alpha FN}, g_{\beta FN})$ denotes each false negative pair and FN denotes the total number of false negatives.

$$\Delta FP = \sum_{j=1}^{FP} \Delta_L(g_{\alpha FP}, g_{\beta FP}). \tag{8}$$

Here, $(g_{\alpha FP}, g_{\beta FP})$ denotes each false negative pair and FP denotes the total number of false positives.

Equations (7) and (8) can be explained with respect to the example in Figure 7. Consider the pair (g_4, g_6) which is an example of a false negative. The distance between this pair of graphs is calculated using the metric Δ_L, which leads to $\Delta_L(g_4, g_6)$. Likewise, we get $\Delta_L(g_4, g_5)$ for the pair (g_4, g_5). Considering every such false negative pair in general, we get the summation of the distances as denoted by the total ΔFN for all false negative pairs. This is captured by our proposed

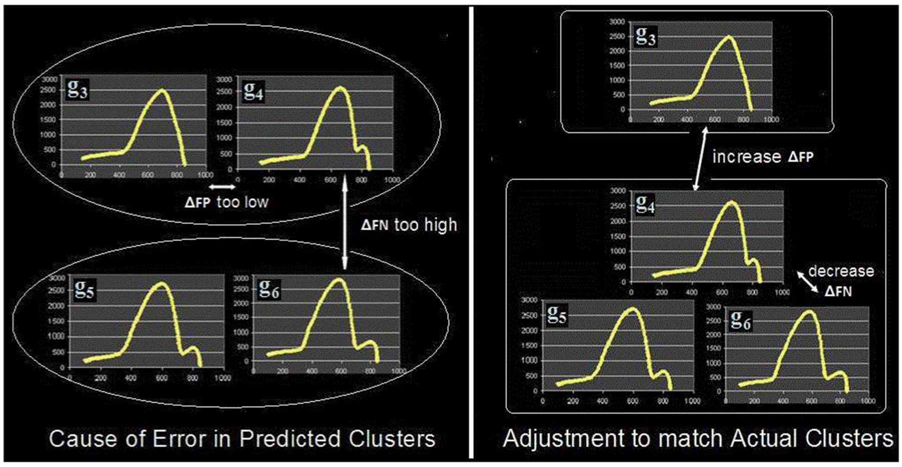


Fig. 7. Illustration of distances used in weight adjustment.

Equation (7). Analogous to this, we explain our proposed Equation (8). Refer to the pair (g_3, g_4) which is a false positive. Here, $\Delta_L(g_3, g_4)$ denotes the distance between this pair of graphs as calculated by the metric Δ_L, and the summation of all such distances between the false negative pairs in general gives the total distance ΔFP as stated in Equation (8).

Given this, consider the false negative pairs shown in Figure 7, e.g., (g_4, g_5) and (g_4, g_6). These pairs are in the same actual cluster. However, they are predicted to be in different clusters. Since predicted clusters are obtained with the metric Δ_L, the cause of the error is that the (average) distance ΔFN between these pairs with the given metric is greater than it should be. Hence, these pairs are incorrectly pushed far apart to be in different predicted clusters although they in reality they should have been closely placed in the same actual cluster. Conversely, for false positive pairs in different actual clusters but in the same predicted clusters, e.g., (g_3, g_4) in Figure 7, the cause of the error is that the (average) distance ΔFP is smaller than it should be. These distances are now used in altering weights using heuristics as follows.

Heuristics in Weight Adjustment: Consider error due to false negatives. To reduce this error, it is desirable to decrease the distance ΔFN. In order to reduce ΔFN, weights of one or more components in the metric used to calculate the distance in the present epoch is decreased. For this purpose, we propose the *FN Heuristic.*

FN Heuristic: Decrease the weights of the components in the metric Δ_L in proportion to their contributions to the distance ΔFN. That is, for each component, new weight $\omega_i`$ is defined using the term ΔFN_i which is the average false negative distance due to any individual component Δ_i alone, e.g., only Leidenfrost distance. Hence, we have the following.

$$\omega_i` = \omega_i - \frac{\Delta FN_i}{\Delta FN} \tag{9}$$

The justification for this heuristic in Equation (9) is as follows. Consider examples of false negative pairs such as (g_4, g_5) and (g_4, g_6) in Figure 7. These pairs are supposed to be in the same actual cluster as per the domain-specific notion of correctness. However, they are incorrectly predicted to be in different clusters. Since the predicted clusters are obtained with the given distance metric, it can be inferred that the error is caused due to the distance ΔFN between these pairs being greater than it should be. In other words, these pairs are incorrectly pushed far apart to be in different predicted clusters although in reality they should have been closely placed in the same actual cluster. Therefore, to rectify this error it is desirable to reduce the distance ΔFN. In order to do that, the weights of one or more components in the metric used to calculate the distance should be reduced, and that is precisely the contribution of the FN heuristic. This is the rationale behind the heuristic.

Conversely, consider the false positive pairs. To reduce their error, we increase ΔFP, by increasing weights of one or more components in the metric using the *FP Heuristic* as defined in an analogous manner next.

FP Heuristic: Increase the weights of the components in the metric Δ_L in proportion to their contributions to the distance ΔFP. Thus, for each component, new weight $\omega_i{}^{\backslash\backslash}$ is given as follows.

$$\omega_i{}^{\backslash\backslash} = \omega_i + \frac{\Delta FP_i}{\Delta FP} \tag{10}$$

Here, ΔFP_i = average false positive distance due to component Δ_L alone.

The justification for the FP heuristic in Equation (10) is analogous to that for the FN heuristic. Consider examples of false positive pairs in different actual but same predicted clusters, e.g., $(g_3,\ g_4)$ in Figure 7. Based on a similar reasoning as above, we can infer that the cause of the error is that the distance ΔFP is smaller than it should be. It follows therefore that in order to rectify error due to the FP pairs, we should increase ΔFP by increasing the weights of one or more components in distance metric, which is achieved by using the FP Heuristic. Combining both these heuristics, we get the final combined Weight Adjustment Heuristic below. Since this heuristic in Equation (11) is a combination of the FN and FP heuristics, its rationale follows from the justification of these individual heuristics.

Weight Adjustment Heuristic: For each component, its new weight is denoted as $\omega_i{}^{\backslash\backslash\backslash}$ and is calculated using the following formula, noting that its weight might drop down to zero. Obviously, there are no negative weights here since the distance components can either have a significant impact on the overall distance metric Δ_L to a certain extent (denoted by positive weights) or be absent/negligible in the metric (i.e., zero weights).

$$\omega_i{}^{\backslash\backslash\backslash} = \max\left(0,\ \omega_i - \frac{\Delta FN_i}{\Delta FN} + \frac{\Delta FP_i}{\Delta FP}\right). \tag{11}$$

These new weights obtained after adjustments is likely to minimize the error due to the both false positive and false negative type pairs. Clustering in the next training epoch is performed with these adjusted weights.

The training epochs continue with weight adjustments as needed each time. The final metric step is reached if the error drops below the threshold τ or if the maximum number of epochs ε is reached. Both τ and ε are experimental parameters.

3.4.5 LearnMet Final Metric Step. If the learning terminates because the error is below the threshold then the metric in the last epoch is output as the final metric. However, if termination occurs because the maximum number of epochs is reached then the most reasonable metric to be output is the one corresponding to the epoch with the minimum error among all epochs. The output of LearnMet serves to provide a notion of distance for clustering the graphical plots in the knowledge discovery process of AutoDomainMine. The distance metric output by LearnMet after going through all its steps is called the *LearnMet Distance* **.** This is the notion of distance for graphical plots and is used for within AutoDomainMine. Having described the LearnMet approach to solve the sub-problem of learning domain-specific distance metrics, we outline the LearnMet algorithm, and the clustering steps in AutoDomainMine.

ALGORITHM 1: LearnMet Algorithm.

```
Input: $G$ = number of graphs, $k$ = number of clusters (actual clusters of graphs given as notion of correctness), $\tau$ = error threshold, $\varepsilon$ = max number of epochs, $c$ = number of components, $\Delta_i$ is each distance component
Set metric $\Delta_L = \sum_{i=1}^{C} \omega_i \Delta_i$ with initial components and weights
Select arbitrary but fixed clustering algorithm
Set $k$ = number of clusters
Clustering Step:
        Cluster graphs using $\Delta_L = \sum_{i=1}^{C} \omega_i \Delta_i$
        Set random $ppe$ = number of pairs per epoch
        Randomly select $ppe$ pairs of graphs
        Using correct actual clusters, calculate TP, TN, FP, FN values
        Calculate Failure Rate FR = (FP + FN)/(TP + TN + FP + FN)
        If (FR < $\tau$) or (#epochs > $\varepsilon$)
        Return $\Delta_L$ as learned metric and End
        Else
            Calculate distances ΔFP and ΔFN
            Apply Weight Adjustment Heuristic to get new metric $\Delta_L = \sum_{i=1}^{C} \omega_i''' \Delta_i$
            Go to Clustering Step
End
Output: Final Learned Metric $\Delta_L$
```

3.5 LearnMet Algorithm and Clustering in AutoDomainMine

On the basis of the detailed explanation herewith, the LearnMet Algorithm is now synopsized as Algorithm 1.

Here, k, G, τ, ε, c are among the experimental input parameters that vary as per the execution of the algorithm. This algorithm can be executed with different values of these for greater robustness. In order to avoid overfitting, the *ppe* values can be adjusted here, and the seeds in the algorithm used for clustering can be altered. The final metric obtained in LearnMet serves as the notion of similarity for graphs and can be used while clustering as well as searching for the closest matching graph during the estimation step in AutoDomainMine. Note that this learning is a one-time process and the learned metric can be used recurrently to capture domain semantics during clustering and searching for graphs.

Clustering is performed in AutoDomainMine by sending the graphs, i.e., the curves mapped to multi-dimensional points after dimensionality reduction, to a clustering technique such as k-means. The notion of distance used for clustering is the domain-specific distance metric learned from LearnMet. Once the clusters are obtained for each graph, the output of the clustering needs to be sent to the classifier. Therefore, each experiment corresponding to a particular graph that results from it, is then stored in terms of its input conditions and cluster label. Note that these labels only serve to identify the cluster of a particular graph and hence the respective experiment in order to serve as the basis for learning the clustering criteria through classification. Based on this discussion and the LearnMet algorithm, steps of clustering in AutoDomainMine are stated below.

(1) Map each 2-dimensional graph into a multi-dimensional point.
(2) Perform dimensionality reduction as needed.

(3) Learn the domain-specific notion of distance for graphs i.e., the LearnMet distance Δ_L.
(4) Cluster the graphs, i.e., multi-dimensional points using the LearnMet distance with any suitable algorithm such as k-means.
(5) Store the output of clustering as the input conditions and cluster label corresponding to each graph

Thereafter, classification is performed to learn the clustering criteria, as a part of knowledge discovery.

4 DETAILS OF CLASSIFICATION IN AUTODOMAINMINE

The method proposed in AutoDomainMine for learning the clustering criteria is classification using decision trees as we justify below. As well-known in data mining, a decision tree [19] is a structure consisting of nodes, arcs, and leaves where each internal node denotes a test on an attribute, each arc leaving a node represents an outcome of the test, and leaf nodes represent classes or class distributions.

4.1 Justification for Deployment of Decision Trees

The reasons for utilizing decision trees as classifiers in AutoDomainMine are stated below.

(1) Decision tree classification is an eager learning approach, i.e., it learns in advance based on existing data as opposed to lazy learners that wait for a new instance to be classified. This is useful because our knowledge discovery step in AutoDomainMine is a one-time process executed over existing data to build a suitable hypothesis before classifying new instances.
(2) Decision tree paths help identify relative importance of criteria used in classification. In our context, this is helpful in determining relative importance of input conditions leading to clusters. This is important while tracing the cluster of a new experiment to find the most suitable match for accurate estimation.
(3) As opposed to some classifiers such as neural models with basic ANN and deep learning with CNN that could be a black box, decision trees are more interpretable, assisting in providing reasons for the decisions. Hence, they are useful to model the reasoning processes of domain experts.

In AutoDomainMine, any suitable decision tree algorithm such as J4.8 can be used [39]. Once decision tree classification is executed, it serves to identify clustering criteria, i.e., conditions leading to a given cluster. This sets the stage for building representative pairs of input conditions and graph per cluster, as the output of the classification step. Instead of randomly selecting representative pairs, there is a need for designing them.

4.2 Need for Designing Representatives

A randomly selected representative pair for each cluster does may not adequately represent the information. Distinct combinations of conditions could lead to a cluster. Graphs in a cluster could have variations. Different applications may need different levels of detail and may have conflicting requirements. We present a motivating example here. Consider Example 1 with sets of conditions ψ_1 to ψ_9 leading to a given cluster.

Example 1. Sets of Experimental Input Conditions in a Cluster

- ψ_1: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (70-80), Probe Type = CHTE

- ψ_2: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (80-90), Probe Type = CHTE
- ψ_3: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = High, Oxide Layer = Any, Quenchant Temperature = (50-60), Probe Type = CHTE
- ψ_4: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Low, Oxide Layer = None, Quenchant Temperature = (60-70), Probe Type = CHTE
- ψ_5: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (20-30), Probe Type = CHTE
- ψ_6: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Any, Oxide Layer = Thin, Quenchant Temperature = (60-70), Probe Type = CHTE
- ψ_7: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (60-70), Probe Type = CHTE
- ψ_8: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (30-40) C, Probe Type = CHTE
- ψ_9: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (90-100), Probe Type = CHTE

All these sets of conditions in Example 1 above lead to a similar experimental output. Hence, they have been assigned to the same cluster. Now, consider an application such as simulation tools [36]. Users often run simulations of real experiments with a given set of input conditions. These simulations are typically as time-consuming as a real experiment (about 6 hours). They are preferred over a real experiment mainly because they save resources. Imagine that a randomly selected set of input conditions is displayed to the user as the output of estimation. If the user runs a simulation using this representative, the ranges of information in the cluster are not captured, thus reducing the sample space of simulations. On the other hand, if the user runs a simulation using a representative that conveys all the information in the cluster, it would take very long to run. Since each simulation takes approximately 6 hours with one set of input conditions, running it with 9 sets of conditions would take 54 hours, which is often not practical. Thus, there is a need for a trade-off between these extremes in such applications. There are other applications where information loss is more critical while efficiency is not an issue, and vice versa.

Likewise, for graphs in each cluster, randomly selected representatives are not always sufficient in incorporating semantics. For example, in a particular cluster the highest heat transfer coefficient could range from 2,000 to 2,300 Watt/m^{2}K, the slow cooling could occur between 200 °C to 250 °C and so forth. It is useful to know that the corresponding graphs still get placed in the same cluster and that these variations do not separate the respective experiments. A randomly selected representative does not convey this information. Also, it is important to avoid visual clutter in displaying information, and address the interests of various users. Thus, there is a need to design semantics-preserving representative pairs consisting of a set of input conditions and graph per cluster in order to aid the classification of new experiments for estimating their results.

4.3 DesRept: Designing Semantics-Preserving Representatives

We propose a methodology called DesRept to design semantics-preserving cluster representatives as the output of the classification step in AutoDomainMine. Each cluster representative is a pair consisting of 2 parts, namely, a set of input conditions and a graph. Designing a representative of conditions involves several issues regarding semantics. Designing a representative graph is concerned with another set of issues regarding domain-specific aspects. Hence, these parts are dealt with separately. However, the general principles behind them are similar as summarized in the steps below.

(1) Input the clusters of graphs, and the decision tree paths of the conditions leading to the clusters.
(2) Define the notion of distance for the sets of conditions and for the graphs.
(3) Build candidate representatives for each cluster by using various design strategies so as to capture the different levels of detail found in the cluster.
(4) Compare the candidate representatives with an encoding for effectiveness based on the **Minimum Description Length** (**MDL**) principle.
(5) For each cluster return the winner, i.e., representative of conditions/graph with the lowest encoding as the designed representative.
(6) Output the representative pair of conditions and graph for each cluster.

Based on this general description, we further divide the DesRept approach into 2 parts. The approach of designing representatives for sets of input conditions is referred to as DesCond while that for designing representatives for graphs is called DesGraph. The details of DesCond and DesGraph are discussed next.

4.3.1 DesCond: Designing Representatives for Conditions. DesCond is a component of DesRept that designs a representative set of input conditions for each cluster. The main steps in DesCond for the design of domain-specific cluster representatives for conditions, are mentioned below and then elaborated further.

(1) Define a notion of distance between individual attributes (conditions).
(2) Derive the distance function for sets of conditions from decision tree paths.
(3) Obtain candidate cluster representatives showing different levels of detail.
(4) Propose an encoding to compare the candidates and find a suitable winner.

Notion of Distance for Attributes: The attributes describing the input conditions are of different types such as numeric, categorical, and ordinal [19]. We use the sets of conditions shown in Example 1 in order to explain the calculation of distance for each type of attribute.

Categorical Attributes: In categorical attributes, depicting character data, the distance is considered to be 0 if the attribute values are identical and 1 if they are not identical. Formally, the distance Δ_{cat} between categorical attributes among sets of conditions ψ_i and ψ_j is calculated as follows where and v_i and v_j the respective values of the given categorical attribute.

$$\Delta_{cat}(\psi_i,\ \psi_j) = \begin{cases} 0 & v_i = v_j \\ 1 & v_i \neq v_j \end{cases} \tag{12}$$

For instance, considering the categorical attribute *Part Material* and referring to Example 1, we calculate distance between the Part Material values as $\Delta_{cat}(\psi_1,\ \psi_3) = 1$ and $\Delta_{cat}(\psi_1,\ \psi_2) = 0$, since Part Material values are not equal in the sets of conditions ψ_1 and ψ_3, while they are equal in ψ_1 and ψ_2.

Numeric Attributes: In numeric attributes, distance is calculated as the absolute difference of their attribute values. If the values are grouped into ranges as a data preprocessing step, then we consider the difference between the mean values of the respective ranges. A suitable scaling factor σ based on domain knowledge can be applied to maintain parity with other attributes. Thus, distance Δ_{num} for numeric attributes in sets of conditions ψ_i and $\psi_j$is calculated as follows where and v_i and v_j the respective actual values (or mean values of ranges) of the given numeric attribute.

$$\Delta_{num}(\psi_i,\ \psi_j) = \ \sigma \times\ |v_i - v_j|. \tag{13}$$

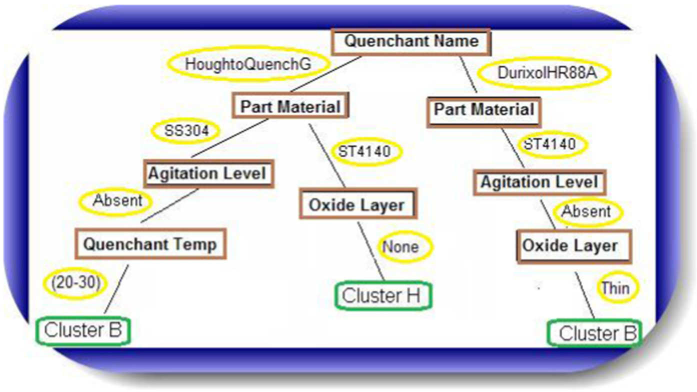


Fig. 8. Partial snapshot of decision tree in experiments from the heat treatment of materials.

Applying this in Example 1, for the numeric attribute *Quenchant Temperature* with scaling factor $\sigma = 1/10$ (given in the domain) we get distances between *Quenchant Temperature* values as $\Delta_{num}(\psi_1, \psi_3) = 2$ and $\Delta_{num}(\psi_1, \psi_2) = 1$, respectively.

Ordinal Attributes: In ordinal attributes, that represent order-based data, the distance is calculated as the absolute difference between their values after they are mapped to numeric based on their order using domain knowledge. For example, Agitation values of "*High*", "*Low*", and "*Absent*" are mapped to 3, 2, and 1, respectively. The value "*Any*" implies that the attribute can take any value. Hence, its distance is considered to be 0 from all other values. The distance Δ_{ord} for ordinal attributes in sets of conditions ψ_i and ψ_j is then given as follows where $v_i`$ and $v_j`$ are the numeric values to which the respective ordinal values are mapped.

$$\Delta_{ord}(\psi_i, \psi_j) = |v_i` - v_j`|. \tag{14}$$

In Example 1, consider ordinal attribute *Agitation Level*. Here, the distance can be calculated as $\Delta_{ord}(\psi_1, \psi_2)$ where both values are "High" thus giving $3 - 3 = 0$; and $\Delta_{ord}(\psi_1, \psi_4)$ where the values are "High" and "Low", respectively thus giving $3 - 2 = 1$. This is as per the mapping based on order explained above, i.e., *High = 3, Medium = 2,* and *Low = 1.*

Distance Function for Sets of Conditions: In order to define the distance between sets of conditions (i.e., sets of attributes), we first propose to assign weights to attributes using reasoning based on decision tree paths. A partial snapshot of an example decision tree appears in Figure 8. We use the following reasoning.

(1) An attribute is considered to have a higher weight than other attributes if it is at a higher level in the decision tree. This is because the root of the tree represents the most significant input condition while the lower levels represent less significant conditions. Also, attributes not identified in the decision tree represent insignificant conditions for the given data sample.
(2) The shorter the path in which an attribute appears, the higher is the significance of that attribute. This is because a shorter path with fewer attributes is more definite in classifying the data than a longer path. An extreme of this would be one particular value of the root leading directly to a given cluster. For example, if all data pertaining to "*QuenchantName = T7A*" belongs to *Cluster C*, irrespective of other attributes, then in this path "*QuenchantName*" should get a higher weight than in other paths.

(3) The greater the number of experiments in the cluster corresponding to a path, the more important is that path and hence an attribute appearing in that path. This is because the given path then classifies a greater amount of data.

We draw an analogy with fundamental decision tree induction algorithms such as ID3 and J4.8 [39] in this reasoning. Accordingly, a heuristic for the weights of the attributes in the decision tree is defined next.

Decision Tree Weight Heuristic: In the DesCond approach, we apply domain knowledge and fundamental decision tree concepts to define a decision tree weight heuristic Ω as calculated next.

$$\Omega_a = \frac{1}{N} \sum_{p=1}^{N} \frac{\eta_{a,p}}{\eta_p} \times \Gamma_p. \tag{15}$$

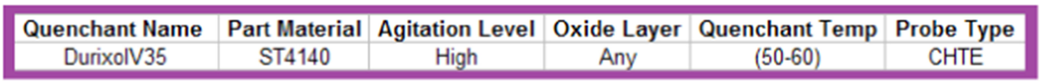
Here,
Ω_a = weight of attribute a;
N = total number of paths in the decision tree;
Γ_p = number of graphs in the cluster corresponding to the leaf of path p;
$\eta_{a,p}$ = height of node for attribute a in path p; and
η_p= height of path p.
Note that height is defined as the number of nodes away from the leaf. Hence, in a given path the leaf has a height of 0, the node immediately above the leaf has a height of 1 and so on. Thus, the height of a path is the height of its root node.

The justification for this heuristic is based on the reasoning provided above before defining the heuristic. Since the root of the tree represents the most significant attribute, the attributes closer to the root are the more significant ones, i.e., those having a greater height in the tree. Hence, the greater the height of the node for a given attribute, the greater is the weight of that attribute, as depicted by the term $\eta_{a,p}$ in the numerator of Equation (15). Likewise, the shorter the path in which the attribute appears, the greater the significance of the attribute since it is more definite in classifying the data. This is indicated by the term η_p appearing in the denominator of the equation, i.e., the higher (longer) the path, the less significant the attribute and vice versa. Finally, the more the number of experiments (and corresponding graphs) in the cluster of a respective path, the greater is the importance of that path and hence an attribute within the path, due to classifying a larger amount of data. This is portrayed by the term Γ_p in the numerator of the equation, i.e., the more the data supported by the given path and its attributes, the greater their significance. This is the rationale behind the heuristic.

This decision tree weight heuristic serves as the basis to define the notion of distance between sets of conditions in the DesCond approach. This distance Δ_{cond} is defined as follows, where a is an attribute such that Ω_a is its weight, and M is the total number of attributes. (Note that this is analogous to Equation (3) defined for distances in graphical plots). This distance function incorporates domain semantics and can be used for the design of candidate cluster representatives.

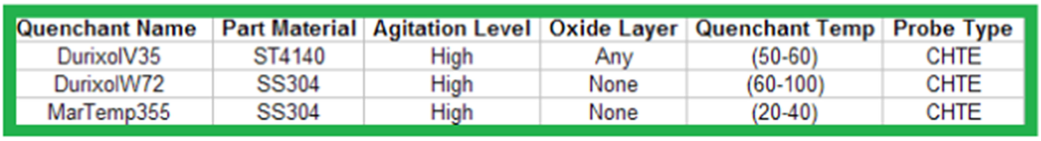
$$\Delta_{cond} = \sum_{a=1}^{M} \Omega_a \, \Delta_a. \tag{16}$$

Levels of Detail: As presented in the motivating example to emphasize the need for designing representatives, different targeted applications require different levels of detail in terms of the sets of conditions that are estimated to achieve a desired graph. Accordingly, we consider the following levels of detail in designing the candidate representatives that serve as the basis for estimation.



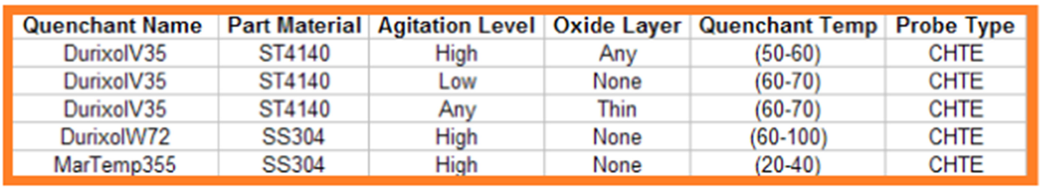
Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE

Fig. 9. Example of "Single Conditions Representative".



Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE
DurixolW72	SS304	High	None	(60-100)	CHTE
MarTemp355	SS304	High	None	(20-40)	CHTE

Fig. 10. Example of "Multiple Conditions Representative".



Quenchant Name	Part Material	Agitation Level	Oxide Layer	Quenchant Temp	Probe Type
DurixolV35	ST4140	High	Any	(50-60)	CHTE
DurixolV35	ST4140	Low	None	(60-70)	CHTE
DurixolV35	ST4140	Any	Thin	(60-70)	CHTE
DurixolW72	SS304	High	None	(60-100)	CHTE
MarTemp355	SS304	High	None	(20-40)	CHTE

Fig. 11. Example of "All Conditions Representative".

Level 1: Single Conditions Representative ψ_{SCR}. This representative is chosen from the original cluster as a single set of conditions closest to all others in the cluster. The notion of similarity to determine this closeness is the distance function Δ_{cond} for sets of conditions (stated above). The ψ_{SCR} for the cluster in Example 1 is shown in Figure 9. This representative is useful in applications where the user is interested in finding the most likely set of input conditions that would give a desired nature of output.

Level 2: Multiple Conditions Representative ψ_{MCR}. This summarizes the information in the cluster. It is designed by using information from the original cluster as follows. The set of conditions in each cluster are grouped into sub-clusters based on the similarity of the conditions. The notion of similarity for sub-clustering is the same distance function Δ_{cond} for sets of conditions. For each sub-cluster, a representative is selected as the set of conditions closest to all the others in the sub-cluster. Likewise, representatives are obtained for each sub-cluster. The ψ_{MCR} is an aggregation of all sub-cluster representatives displayed in a tabular form. The ψ_{MCR} for Example 1 is shown in Figure 10. The MCR depicts a trade-off between the amount of detail displayed to the user and the amount of information captured within the cluster.

Level 3: All Conditions Representative ψ_{ACR}. This captures all data in the cluster with no information loss. It is constructed by retaining all the original sets of conditions and displaying them by sorting from the most to least significant attribute. The significance of the attributes is determined based on the distance function Δ_{cond} for sets of conditions. The values of each set of conditions are abstracted using domain knowledge wherever possible. For example, in Heat Treating of Materials, if 3 sets of conditions are identical except that the value of *Agitation Level* is "*Absent*" for one, "*Low*" for another, and "*High*" for the third, then this is abstracted as *Agitation* = "*Any*", where "*Any*" refers to any possible value of agitation applicable to the domain. This avoids visual clutter, while still displaying all the information. The ψ_{ACR} for Example 1 is shown in Figure 11.

Comparison of Candidates: The candidate representatives are compared to select a winner for each cluster to be used as the designed representative for the purpose of estimation. This winner to some extent depends on the nature of the targeted application for which the estimation will be performed. We propose an encoding as a measure to compare the candidates drawing an analogy with the MDL principle. The MDL principle proposed by Rissanen [51] aims to minimize the sum of encoding the theory and the examples using the theory. Our proposed measure based on this is called the *DesCond Encoding (*or the *DesRept Encoding for Conditions).* It is described below.

The DesCond Encoding: This is designed to capture domain semantics while adhering to the concept of measuring the goodness of a cluster in terms of its representative set of conditions. In this context, we propose that the theory (with respect to MDL) refers to the cluster representative while the examples refer to all the other sets of conditions in the cluster. We consider the complexity of each representative and the information loss due to it. Complexity refers to the ease of interpretation, which is measured as the amount of data stored for the representative. Information loss, the capacity of the representative in capturing information within the cluster, is measured as distance of the representative from all objects (sets of conditions) in the cluster. The relative importance attached to complexity and distance (information loss) is embedded in the encoding, based on the interests of targeted users. This measure ϕ_{cond} is calculated as follows.

$$\phi_{cond} = \zeta_{complexity} \times \log_2 (AV) \; + \; \zeta_{distance} \times \log_2 \frac{1}{S} \sum_{i=1}^{S} \Delta_{cond}(\psi_{rep},\, \psi_i)\;. \tag{17}$$

Here,

ϕ_{cond} = measure of encoding for conditions;
ψ_{rep} = cluster representative as a set of conditions;
A = number of attributes in the representative;
V = number of values for each attribute, i.e. number of instances in representative;
ψ_i = each set of conditions in the cluster;
$\Delta_{cond}(\rho, \psi_i)$ = distance between representative and every set of conditions;
S = total number of sets of conditions in cluster;
$\zeta_{complexity}$ = percentage weight giving user bias for complexity;
$\zeta_{distance}$ = percentage weight giving user bias for distance.

The first term in this encoding $\log_2(AV)$ designates the complexity of the representative. This is calculated as the number of attributes and values that need to be stored for that representative. The second term, i.e., the distance term $\log_2 \frac{1}{S} \sum_{i=1}^{S} \Delta_{cond}(\psi_{rep}, \psi_i)$ denotes information loss due to the representative. It is calculated as the average distance of the representative from all the other sets of conditions in the cluster. The terms $\zeta_{complexity}$ and $\zeta_{distance}$ are the percentage weights assigned to complexity and distance, respectively, in order to express the user bias for those terms. Unless otherwise specified, equal weights are assigned to complexity and distance, i.e., 50% each.

Candidate cluster representatives are evaluated using the DesCond Encoding. The representative with the lowest value of the encoding measure ϕ_{cond} for the given cluster is considered the best and is returned as its designed representative for the cluster. Hence, the DesCond approach serves as a method to design and evaluate cluster representatives for conditions. In addition, there is the DesGraph approach for graphs.

4.3.2 *DesGraph: Designing Representatives for Graphs.*

DesGraph is a component within DesRept that designs cluster representatives of graphs. The main steps in DesGraph are listed below and discussed in the subsequent subsections.

(1) Specify a notion of distance for the graphs.
(2) Design candidate cluster representatives by design strategies: guided selection and construction.
(3) Define an effectiveness measure to compare candidates and return the best one.

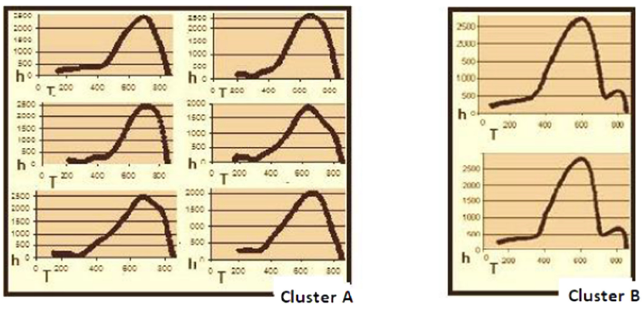


Fig. 12. Sample clusters of graphs.

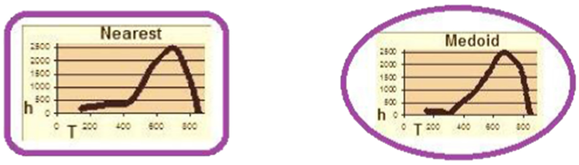


Fig. 13. Examples of "Nearest" (left) and "Medoid" (right) Representatives (for Cluster A in Figure 12).

Notion of Distance: In DesGraph, the distance metric from the LearnMet technique is used as the notion of distance for graphs. This is the *LearnMet Distance* $\Delta_L = \sum_{i=1}^{C} \omega_i \, \Delta_i$ as stated earlier.

Building Candidate Representatives: Consider the example of sample clusters of graphical plots as shown in Figure 12. We explain design of candidate representatives based on this example.

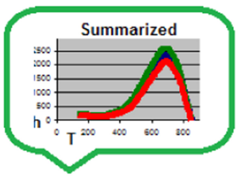
Design by Guided Selection. In guided selection, the representative is derived as one of the objects existing in the cluster. Two candidate representatives, *Nearest Representative* and *Medoid Representative*, are selected as follows. Examples of these are illustrated in Figure 13.

The nearest representative is derived based on the concept of nearest neighbors using pairwise distances, as explained below in Code Snippet 1 where Γ_i and Γ_j refer to individual graphs in the cluster, G is the total number of graphs in the cluster, $\Gamma_{nearest}$ is the nearest representative graph and Δ_L is the distance between graphs using the LearnMet distance metric.

```
CODE SNIPPET 1: NEAREST REPRESENTATIVE GRAPH
    FOR   i = 1 to G
           COMPUTE Sum (i) = $\sum_{i=1}^{G} \Delta_L(\Gamma_i, \Gamma_j)$
    ENDFOR
    RETURN $\Gamma_{nearest}$ = Argmin (Sum (i))
```

Note that we use sum here and not sum of squares because the assumption is that squared distances are already incorporated in the metric. This representative, the nearest graph, depicts the member of the cluster that is nearest to the others using the given distance metric. Since the metric incorporates domain semantics this representative conveys nearness with respect to relative importance of regions on graphs.

The medoid representative is derived as an existing graph in the cluster closest to the value of its computed centroid or mean, as mentioned below in Code Snippet 2 where Γ_i refers to each graph, G is the number of graphs in the cluster, μ is the cluster centroid (mean value), and Δ_L is the distance metric, i.e., LearnMet distance. The assumption is that the x-coordinates for all graphs are the same. Hence, in computing the centroid, we take the mean of the y-coordinates only. This



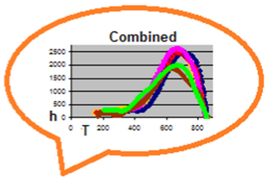


Fig. 14. Examples of "Summarized" (left) and "Combined" (right) Representatives (Cluster A in Figure 12).

medoid representative graph i.e., Γ_{medoid}, helps to visualize the object in the cluster closest to the average behavior of the dependent variable on the graphs.

CODE SNIPPET 2: MEDOID REPRESENTATIVE GRAPH
```
COMPUTE $\mu = \frac{1}{G}\sum_{i=1}^{G} \Gamma_i$
FOR  i = 1toG
    COMPUTE $\Delta(i) = \sum_{i=1}^{G} \Delta_L(\mu, \Gamma_i)$
ENDFOR
RETURN$\Gamma_{medoid}$ = Argmin (($\Delta$(i)))
```

Design by Construction. In construction, the representative is a new object derived using data in the cluster. We derive two such representatives, the *Summarized Representative* and *Combined Representative*. Examples of these are depicted in Figure 14. Note that the callouts around the representatives in the figure symbolize *construction* (as opposed to regular shapes depicting *selection* for the Nearest and Medoid representatives).

The summarized representative proffers a summary of information in the cluster. It is derived by constructing 3 (potentially new) graphs: an average of graphs in the cluster and graphs depicting domain-specific upper and lower prediction limits. The average graph is computed as the cluster centroid or mean value (which may not be any graph in the original cluster) while graphs for prediction limits are computed using percentage values of upper and lower domain-specific thresholds added and subtracted from the average, respectively. This is derived as follows in Code Snippet 3. Here, Γ_i refers to each graph, $\Gamma_i(y)$ is its y-coordinate at a given x-coordinate, n is the total number of x-coordinates, G is the number of graphs in the cluster, Γ_{av}, Γ_{up}, Γ_{low} are the constructed average graph, upper limit graph, and lower limit graph, respectively, $\Gamma_{av}(y)$, $\Gamma_{up}(y)$, $\Gamma_{low}(y)$ are their respective y-coordinates at the given x-coordinates, τ_U and τ_L are percentage thresholds for upper and lower limits, respectively, and $\Gamma_{summarized}$ denotes the summarized representative graph derived here. Since x-coordinates are identical (see Figure 14), we compute values only for the y-coordinates of the respective graphs. This representative, namely, the average graph with prediction limits, is a complex object consisting of 3 curves.

CODE SNIPPET 3: SUMMARIZED REPRESENTATIVE GRAPH
```
FOR y = 1 to n
    COMPUTE $\Gamma_{av}$ (y) = $\frac{1}{G}\sum_{i=1}^{G} \Gamma_i$ (j)
    COMPUTE $\Gamma_{up}$ (y) = $\Gamma_{av}$ (j) + $\tau_U \times \Gamma_{av}$ (j)
    COMPUTE $\Gamma_{low}$ (y) = $\Gamma_{av}$ (j) − $\tau_L \times \Gamma_{av}$ (j)
ENDFOR
RETUEN $\Gamma_{summarized}$ = $\Gamma_{av}$, $\Gamma_{up}$, $\Gamma_{low}$ with x coordinates
```

The combined representative is constructed by superimposing all the graphs in a given cluster on each other. It is derived as follows in Code Snippet 4 where Γ_i refers to each graph, $\Gamma_i(y)$ is its y-coordinate at a given x-coordinate, n is the number of x-coordinates, G is the number of graphs in the cluster, and $\Gamma_{combined}$ denotes the combined representative graph. This representative is a complex object composed of curves. It illustrates the whole cluster with no information loss and depicts possible subtleties, e.g., the combined representative in Figure 14 shows that maximum heat transfer occurs at around the same temperature for all graphs in the cluster. This is also called the *superimposed graph* since it is constructed by superimposing all graphs in the cluster.

```
CODE SNIPPET 4: COMBINED REPRESENTATIVE GRAPH
  FOR y = 1 to n
      FOR i = 1 to G
            COMPUTE $var_i(y) = \Gamma_i(y)$
      ENDFOR
  ENDFOR
  RETURN $\Gamma_{combined} = var_i$ {i = 1 to G} with x coordinates
```

Effectiveness Measure for Representative Graphs: We propose an effectiveness measure called the *DesGraph Encoding* or *DesRept Encoding for Graphs* for evaluating the representative graphs, analogous to the DesCond Encoding for Conditions. The DesGraph Encoding is also based on the MDL principle [51], and is given below.

The DesGraph Encoding: This encoding is designed to assess the effectiveness of cluster representative graphs to preserve the semantics of the given domain. Analogous to the DesCond encoding for conditions, this follows the MDL principle of minimizing the sum of coding of the theory itself and the examples using the theory. Here we state that the theory refers to the cluster representative graphs while the examples refer to all the other objects (graphs) in the cluster. Likewise, we incorporate the complexity of each representative and the information loss due to it. The complexity is measured as the total amount of data stored for the coding the representative graph while the information loss is measured as the distance of the representative graph from all the other graphs in the cluster. The relative importance attached to the terms of complexity and distance (information loss) is accommodated in the encoding, based on the interests of targeted users. This measure ϕ_{graph} is calculated as follows.

$$\phi_{graph} = \zeta_{complexity} \times \log_2(N_{rep}) \ + \ \zeta_{distance} \times \log_2 \frac{1}{G} \sum_{i=1}^{G} \Delta_L(\Gamma_{rep}, \Gamma_i). \tag{18}$$

Here,
ϕ_{graph} = measure of encoding for graphs;
Γ_{rep} = cluster representative graph;
N_{rep} = number of data points to store the representative graph;
Γ_i = each individual graph in cluster;
$\Delta_L(\rho, \psi_i)$ = distance between representative and every other graph;
G = total number of graphs in the cluster;
$\zeta_{complexity}$ = percentage weight giving user bias for complexity;
$\zeta_{distance}$ = percentage weight giving user bias for distance.

The first term in this encoding $\log_2(N_{rep})$ pertains to the complexity of the representative. This is calculated as the number of data points that need to be stored for that representative. The second term, i.e., the distance term $\log_2 \frac{1}{G} \sum_{i=1}^{G} \Delta_L(\Gamma_{rep}, \Gamma_i)$ is the average distance of each graph in the cluster from the representative. This distance gives the information loss with respect to domain semantics because it is computed using the given distance metric. The terms $\zeta_{complexity}$ and $\zeta_{distance}$ are the percentage weights assigned to the complexity and distance terms respectively in order to express the user bias for those terms. Unless otherwise specified, equal weights are assigned to complexity and distance, i.e., 50% each. In some situations, users are interested in capturing more information in the cluster and do not attach much significance to the complexity of the representative care about how complex the representative is. Thus, complexity gets a lower weight. Some categories of users give high importance to complexity for reasons such as storage and ease of display. Hence, complexity gets a higher weight. Using this encoding ϕ_{graph}, candidates are compared. The candidate with the lowest encoding is the winner for that cluster, serving as its representative. Having thus explained the design of representatives, we now summarize this process via the DesRept algorithm to design representatives for conditions and graphs capturing domain semantics. These designed representatives constitute form the output of the classification in AutoDomainMine.

4.4 DesRept Algorithm to Design Representatives as the Output of Classification

The DesRept algorithm encompasses the DesCond and DesGraph approaches to design representatives for conditions and graphs, respectively. Using the corresponding encodings, the best candidate representatives, one each for conditions and graphs, are selected for each cluster to form its representative pair. The DesRept Algorithm is synopsized herewith as Algorithm 2 on the basis of the explanation above.

While there are many parameters in the overall framework here, it is to be noted that there is no specific "optimal" choice. The type of representative depends on the needs of the targeted applications. In some applications, it is more important to capture intricate levels of detail; in other situations, brevity may be highly desirable. The relative importance of the parameters $\zeta_{complexity}$ and $\zeta_{distance}$ is user-defined based on the significance users attach to the complexity of the representative and the information loss (as measured by the distance parameter), respectively. In a default scenario, both these parameters would be given equal weight. Regarding the needs of targeted applications, there is more information in the evaluation section. Much of this is based on the respective domain and users.

We have hereby presented the of the overall DesRept methodology that is meant for designing cluster representatives of conditions and graphs in order to serve as the classification outputs in the knowledge discovery step of the AutoDomainMine approach. These designed representatives as the output of classification are used for estimation in AutoDomainMine.

5 DETAILS OF ESTIMATION IN AUTODOMAINMINE

5.1 Estimating the Graph, Given the Conditions

In order to estimate the graph for a new experiment, some or all of the conditions of the experiment are entered by the user as the input to AutoDomainMine. The estimation process of AutoDomainMine compares these conditions with the decision tree paths to find the closest matching path leading to the cluster in which the new experiment would best be placed. The *Decision Tree Weight Heuristic*, which is learned in the DesCond component of DesRept, is used while making this comparison in order to preserve semantics. The designed representative graph of the closest matching cluster is output as the estimated graph. If there is a total match of decision tree paths, the potential cluster of the new experiment is obvious, hence that representative graph is clearly offered as the estimated output. We now describe what occurs in the event of approximate matches.

The relative importance of the input conditions learned through decision tree classification helps to make an educated guess about the closest matching cluster for the user-submitted input conditions, if there is no total match. Hence, if more important conditions as identified by the higher levels of the tree do not match, this is considered insufficient to provide an estimate. However, if

ALGORITHM 2: DesRept Algorithm

```
I. The DesCond Procedure
Input: clusters of conditions, distance metric $\Delta_{cond}$, user bias for complexity, distance as $\zeta_{complexity}$, $\zeta_{distance}$
1. Design candidate cluster representatives for conditions as:
        a. Single Conditions Representative $\psi_{SCR}$
        b. Multiple Conditions Representative $\psi_{MCR}$
        c. All Conditions Representative $\psi_{ACR}$
2. Use DesCond encoding measure $\phi_{cond}$ to compare effectiveness of candidates
3. Return $\psi_{rep} = Argmin(\phi_{cond}\ \{\psi_{SCR}, \psi_{MCR}, \psi_{ACR}\})$
Output: Designed representative set of conditions as $\psi_{rep}$
II. The DesGraph Procedure
Input: clusters of graphs, distance metric $\Delta_L$, user bias for complexity, distance as $\zeta_{complexity}$, $\zeta_{distance}$
1. Design candidate cluster representatives for graphs by:
        a. Guided Selection as:
                i. Nearest Representative Graph $\Gamma_{nearest}$
                ii. Medoid Representative Graph $\Gamma_{medoid}$
        b. Construction as:
                iii. Summarized Representative Graph $\Gamma_{summarized}$
                iv. Combined Representative Graph $\Gamma_{combined}$
2. Use DesGraph encoding measure $\phi_{graph}$ to compare effectiveness of candidates
3. Return $\Gamma_{rep} = Argmin(\phi_{graph}\{\Gamma_{nearest}, \Gamma_{medoid}, \Gamma_{summarized}, \Gamma_{combined}\})$
Output: Designed representative graph as $\Gamma_{rep}$
```

no complete match is found due to lower level discrepancies, then it is considered acceptable to give an estimate based on an approximate match. The distinction between high and low levels is made depending on the height of the tree and the requirements of accuracy. Accordingly, using the *Decision Tree Weight Heuristic*, the levels at or above half the depth of the tree are considered as high, and those below are half the depth as low. If multiple paths match partially, to the same extent, then the cluster with the greatest number of graphs is considered as the matching cluster, and its representative graph is offered as the estimated output. This concept of selecting the cluster with the greatest number of graphs in a partial match is analogous to using the majority class in classifiers [39]. Based on this, the process of estimating the graph given the conditions in AutoDomainMine, has the following steps.

(1) Accept the given input conditions from the user.
(2) Compare each path of the decision tree with the given input conditions.
(3) If a complete path matches, the designed representative graph of that cluster is the estimated graph.
(4) If one or more partial paths match greater than half the height of the tree, the designed representative graph of the cluster with the maximum number of experiments is the estimated graph.
(5) If one or more partial paths match up to less than or equal to half the height of the tree, convey that the graph cannot be estimated.

5.2 Estimating the Conditions, Given the Graph

The process of estimating the input conditions, given a desired graph is as follows. A sample graph or ranges of its features are entered by the user as a desired result, for AutoDomainMine to estimate the experimental conditions to achieve such a result. The estimation process in AutoDomainMine compares this graph with the designed representative graphs in the clusters using the domain-specific notion of distance from LearnMet. Accordingly, the graph with the closest match within a threshold (discussed next) is found. The designed representative conditions of the cluster of the closest matching graph, are conveyed as the estimated conditions.

We define a similarity threshold for graphs in a domain-specific manner. If no match is found within the given threshold, then the desired graph cannot be obtained based on knowledge discovered from existing experimental data. Thus, it is conveyed to the user that the conditions to obtain this graph cannot be estimated. This is in order to adhere to the requirements of accuracy. If only one representative graph matches the desired graph within the threshold, then the corresponding representative conditions are clearly the estimated conditions. If several representative graphs match within the given threshold, we obviously select the closest match. However, if multiple representative graphs match to a similar extent then we offer the representative conditions of the cluster with the greatest number of experiments as the estimated conditions. This is again analogous to the concept of majority class in classifiers [39]. Considering the explanation above, the process of estimating the conditions, given the graph in AutoDomainMine has the following steps.

(1) Accept the given graph from the user.
(2) Compare the graph with designed representative graphs of each cluster (as per tree paths).
(3) If no match is found within the threshold then convey that the conditions cannot be estimated.
(4) If only one representative graph matches within the threshold, then the representative conditions of that cluster are the estimated conditions.
(5) If more than one graph matches within the threshold, then representative conditions of the cluster with the closest matching graph are the estimated conditions.
(6) If multiple graphs match to a similar extent, then the representative conditions of the cluster with the greatest number of experiments are the estimated conditions.

5.3 The AutoDomainMine Algorithm for Computational Estimation

In this subsection, we have outlined the AutoDomainMine algorithm that combines all its parts. This encompasses the one-time process of knowledge discovery through distance metric learning, clustering, classification, and representative design; followed by the recurrent process of using the discovered knowledge for estimation. This AutoDomainMine algorithm is presented as Algorithm 3 herewith.

This describes the AutoDomainMine approach on the whole. Note that in the one-time process of knowledge discovery, parameters such as clustering seeds, number of clusters, stopping criteria for decision trees, and so on are altered for greater robustness. We have addressed this in our implementation. Evaluation is described next.

6 EVALUATION OF THE AUTODOMAINMINE SYSTEM

The AutoDomainMine approach has been implemented as a software system and has been subjected to rigorous experimental evaluation, a synopsis of which is presented next. The LearnMet technique, the DesRept methodology and the overall AutoDomainMine system incorporating these, are evaluated separately. Comparative studies are attempted with other possible approaches

ALGORITHM 3: AutoDomainMine Algorithm

```
I. Knowle dge Discovery Step (One-Time Process)
Input: experiments with sets of conditions $\psi_i$ and graphs $\Gamma_i$,
1. Learn domain-specific distance for graphs $\Delta_L$ via LearnMet
2. Cluster experiments based on graphs using $\Delta_L$ with any algorithm such as k-means
3. Build decision tree classifiers to learn causes of similarities and differences between clusters
4. Develop representative pairs $(\psi_{rep}, \Gamma_{rep})$ for each cluster via DesRept
5. Store decision tree paths $P_i$ and representative pairs $P_i(\psi_{rep}, \Gamma_{rep})$ as discovered knowledge
II. Estimation of Graph (Recurrent Process)
Input: experimental set of conditions $\psi_E$
Given: decision tree paths $P_i$ with cluster representatives, number of experiments #E in clusters of each path, distance metric $\Delta_{cond}$
Define: $H_i = Height(P_i)$; partial-match as $match(\psi_E, P_i) > \frac{H_i}{2}$; no-match as $match(\psi_E, P_i) \leq \frac{H_i}{2}$
1. Compare $\psi_E$ with each decision tree path $P_i$
2. If $\exists P_i$ where full-match $(\psi_E, P_i)$ then $\Gamma_E = P_i(\Gamma_{rep})$        \\ complete tree path matches given
                                                                                                 conditions
3. If $\exists P_i$ where partial-match $(\psi_E, P_i)$ then $\Gamma_E = P_i(\Gamma_{rep})$ where $P_i \mapsto Argmax(P_i\{\#E\})$
4. If $\forall P_i$ no-match $(\psi_E, P_i)$ then $\Gamma_E$ = null        \\ no estimation based on existing data
Output: estimated graph $\Gamma_E$
III. Estimation of Conditions (Recurrent Process)
Input: experimental graph $\Gamma_E$
Given: decision tree paths $P_i$ with cluster representatives, number of experiments #E in clusters of each path, graph matching distance threshold $\theta$, distance metric $\Delta_L$
1. Compare $\Gamma_E$ with representative graphs $P_i(\Gamma_{rep})$ for cluster of each tree path $P_i$
2. If $\forall P_i$ $\Delta_L(\Gamma_E, P_i(\Gamma_{rep})) > \theta$ then $\psi_E$ = null        \\ no estimation based on existing data
3. If $\exists P_i$ $\Delta_L(\Gamma_E, P_i(\Gamma_{rep})) \leq \theta$ then
            If number-of-matches = 1 then $\psi_E = P_i(\psi_{rep})$
            If number-of-matches > 1 then $\psi_E = P_i(\psi_{rep})$
                                 where $P_i \mapsto (Argmax(P_i\{\Delta_L(\Gamma_E, \Gamma_{rep})\}) \wedge Argmax(P_i\{\#E\}))$
Output: estimated set of conditions $\psi_E$
```

that could potentially useful for computational estimation. Extension to other domains is discussed besides the example domain of Materials Science that motivated this work. Impacts of our work are discussed with respect to usefulness in other research.

6.1 Evaluation of LearnMet and DesRept Techniques

6.1.1 LearnMet Evaluation. We conduct evaluation with the LearnMet technique using experimental conditions and graphs from the Heat Treating of Materials. A few excerpts from evaluation are presented.

We first consider the impact of the components in learning the distance metric Δ_L. Approximately 100 executions are conducted, excerpts from which appear here. Number of graphs in the training set is $G = 25$ from which 300 pairs of graphs are obtained. We use k-means for clustering with $k = 5$ as the value obtained from actual clusters (given by experts) in the training set. We alter clustering seeds for randomization. Error threshold is $\tau = 0.01$ and maximum number of epochs is $\varepsilon = 1{,}000$. The Number of components is altered in each execution. Possible components

Table 1. Various Components in Executions of Distance Metric Learning

Execution Number	Distance Components
Ex1	Δ_{sc}
Ex2	$\Delta_{Euclidean}$
Ex3	Δ_{Min}
Ex4	Δ_{Mean}
Ex5	Δ_{Max}
Ex6	Δ_{LF}
Ex7	Δ_{BP}
Ex8	$\Delta_{Euclidean}, \Delta_{Max}$
Ex9	$\Delta_{Euclidean}, \Delta_{Max}, \Delta_{BP}$
Ex10	$\Delta_{Euclidean}, \Delta_{Max}, \Delta_{BP}, \Delta_{LF}$
Ex11	$\Delta_{Euclidean}, \Delta_{Max}, \Delta_{BP}, \Delta_{LF}, \Delta_{Mean}$
Ex12	$\Delta_{Euclidean}, \Delta_{Max}, \Delta_{BP}, \Delta_{LF}, \Delta_{Mean}, \Delta_{sc}$
Ex13	$\Delta_{Euclidean}, \Delta_{Max}, \Delta_{BP}, \Delta_{LF}, \Delta_{Mean}, \Delta_{sc}, \Delta_{Min}$

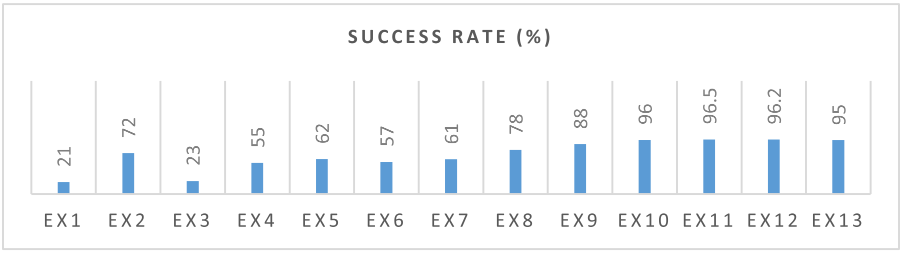


Fig. 15. Accuracy for executions with various distance components (from Table 1).

are $\Delta_{Euclidean}$, Δ_{Max}, Δ_{Mean}, Δ_{Min}, Δ_{LF}, Δ_{BP}, and Δ_{sc} that depict position-based Euclidean distance, statistical distances of Maximum, Mean, and Minimum distance, and critical distances of Leidenfrost distance, Boiling Point distance, and Slow Cooling distance, respectively. These are obtained from the literature. Our executions with different components are listed in Table 1 while accuracy (Success Rate as per Equation (4)) is plotted in Figure 15. Note that accuracy is measured over a distinct test set. It is observed herewith that the highest is accuracy is in the range of 95%, approximately. The simplest metric giving this accuracy has 4 components: Euclidean distance, Maximum distance, Leidenfrost distance, and Boiling Point distance (Ex10). Hence, if we have a preference for prefer simpler metrics, the learned metric has these 4 components. They are used in subsequent executions to learn the effect of weights, *ppe* values, and so on.

We consider the effect of weights next. We consider 5 different combinations of initial weights, including 2 different combinations given by 2 domain experts, a combination with all components having the same initial weight, and 2 random combinations. The final learned weights corresponding to these initial metrics are presented in Table 2 where DE1, DE2 are the executions with learned metrics obtained based on the initial weights given by the 2 experts, EQU represents the learned metric with random initial weights, and RND1, RND2 pertain to the learned metrics with random initial weights. The clustering accuracy (Success Rate) obtained with these respective learned metrics over a test set appears in Figure 16 while the learning efficiency from initial to final weights is depicted in Figure 17 for all these combinations.

Table 2. Final Learned Metrics for Different Combinations

Execution Number	Learned Distance Metric
DE1	$4.1382\ \Delta_{Euclidean} + 2.7315\ \Delta_{Max} + 1.8739\ \Delta_{LF} + 3.1023\ \Delta_{BP}$
DE2	$4.2099\ \Delta_{Euclidean} + 2.8784\ \Delta_{Max} + 2.1665\ \Delta_{LF} + 2.8995\ \Delta_{BP}$
EQU	$4.0983\ \Delta_{Euclidean} + 2.6997\ \Delta_{Max} + 2.2321\ \Delta_{LF} + 2.9589\ \Delta_{BP}$
RND1	$4.0898\ \Delta_{Euclidean} + 2.7228\ \Delta_{Max} + 2.0093\ \Delta_{LF} + 3.0955\ \Delta_{BP}$
RND2	$4.2008\ \Delta_{Euclidean} + 2.6981\ \Delta_{Max} + 1.9976\ \Delta_{LF} + 3.1328\ \Delta_{BP}$

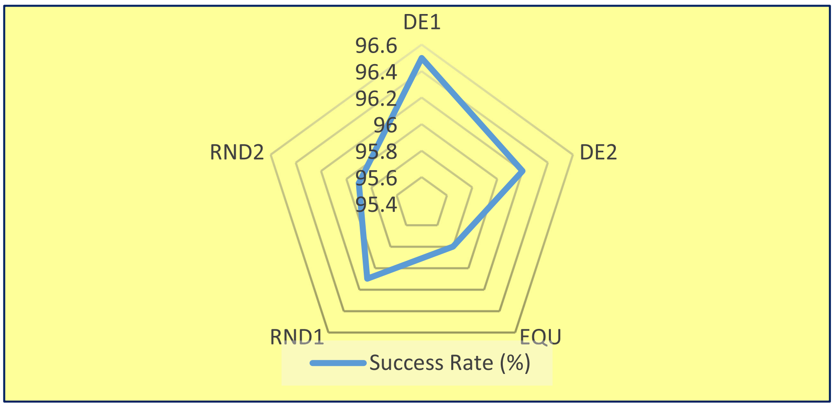


Fig. 16. Accuracy for executions with different combinations of learned metrics (from Table 2).

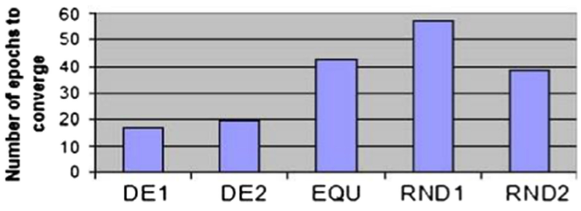


Fig. 17. Learning efficiency to obtain different combinations of learned metrics (from Table 2).

The observations from these different combinations reveal that convergence to error below threshold occurs in all the executions, however, the experiments with initial metrics provided by experts converge faster. The accuracy obtained from all the learned metrics is high, i.e., in the 95% range. An important fact to note is that all the executions converge to approximately the same learned metrics, thus proving the effectiveness of the LearnMet technique in learning a good distance metric, regardless of the initial combination selected.

Among other executions conducted, we present those for *ppe* values, i.e., effect of number of *pairs* (of graphs) *per epoch* on the learning. We dwell on these since they are significant in the context of big data. It is interesting to note the selection of pairs per epoch for randomness and learning efficiency. There are 200 executions conducted on *ppe* values with different data sets and clustering seeds. The training set has $G = 40$ graphical plots with number of clusters $k = 7$. The error threshold is 0.01. The clustering accuracy (Success Rate) on the test set and behavior during convergence on the training set are illustrated in Figures 18 and 19, respectively.

Based on these executions, it is noticed that low *ppe* values, e.g., $ppe < G$ may converge faster but the learned metrics give relatively lower accuracy over the test set. Moreover, some executions

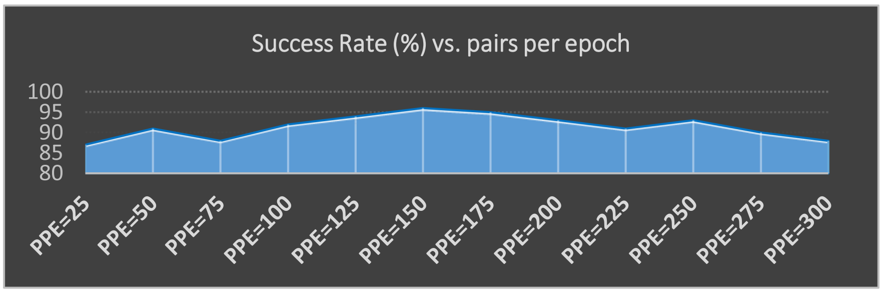


Fig. 18. Accuracy with different values of “ppe”.

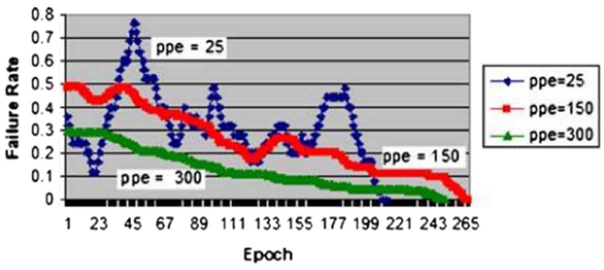


Fig. 19. Training behavior with different ranges of “ppe”.

with low *ppe* values may take as long to converge as *ppe* gets close to *G*. This probably depends on which *ppe* pairs get selected in each epoch. Middle range *ppe* values take longer to converge but give the best accuracy over the test set. High *ppe* values close take still longer to converge and give accuracy over test set less than middle range values. Another observation is that the Failure Rate (as per Equation (5)) decreases monotonously for high *ppe* values but oscillates for lower *ppe* values. This could potentially be because for low *ppe* values, a distinctly different set of pairs get used in each epoch for learning, so the metric is learned over a different set of pairs each time. For higher *ppe* values, almost the same pairs get selected in each epoch, thus causing a uniform decrease in Failure Rate. As data gets bigger, this might be a useful takeaway in other contexts as well.

6.1.2 DesRept Evaluation. The effectiveness of designed representatives in DesRept is assessed with respect to their usefulness in providing the AutoDomainMine estimation. Since the representatives are designed such that they should appeal to users, the assessment of DesRept is achieved through formal surveys conducted by the users of this system. Users execute tests comparing the AutoDomainMine estimation with the results of real laboratory experiments (from data not used for training, i.e., test data). In each test executed by users, the designed representatives are compared in terms of how effective they are in displaying information in various targeted applications of AutoDomainMine. The applications include parameter selection, simulation tools, intelligent tutors, and expert systems. In order to perform this assessment, the estimated output of AutoDomainMine is displayed to the users in different levels of detail, as the Single Conditions Representative, Multiple Conditions Representative, and All Conditions Representative, respectively, for conditions; and the Nearest Representative, Medoid Representative, Summarized Representative, and Combined Representative, respectively, for graphs. Note that the names of the representatives are obviously not shown to the users, only the corresponding visual displays are presented. Different categories of users are asked to choose which display (designed representative) best meets their needs, i.e., “wins” with respect to the given application. We show survey

Table 3. Winning Representatives from DesRept in the Application of Parameter Selection

Winner for Conditions	Percentage	Winner for Graphs	Percentage
SCR Wins	*45%*	*Nearest/Medoid Wins*	*43%*
MCR Wins	28%	Summarized Wins	24%
ACR Wins	21%	Combined Wins	27%
None Wins	6%	None Wins	6%

Table 4. Winning Representatives from DesRept in the Application of Simulation Tools

Winner for Conditions	Percentage	Winner for Graphs	Percentage
SCR Wins	18%	Nearest/Medoid Wins	15%
MCR Wins	*47%*	*Summarized Wins*	*49%*
ACR Wins	30%	Combined Wins	33%
None Wins	5%	None Wins	3%

results in different targeted applications. In these surveys, when users indicate "none wins", no representative is adequate, i.e., the estimation itself does not seem acceptable to users (compared to results of lab experiments) and is considered inaccurate. DesRept survey results are summarized here (values rounded off to the nearest percentage).

Parameter Selection: In the applications of parameter selection, the computational estimation provided as the output of AutoDomainMine is used to select process parameters in industry, e.g., [37]. The users have conducted 53 tests in this category. Winners in these applications are depicted in Table 3. As observed here, the "none wins" region is much smaller than the others indicating that very few tests resulted in inaccurate estimation. It is seen that the Nearest or Medoid Representatives for graphs and **Single Conditions Representatives** (**SCR**) for conditions are winners for most tests (italicized in the table). The reason is possibly that in parameter selection, typically most users want one right answer that is displayed in a concise manner.

Simulation Tools: In simulation tool applications, users run computer simulations of real laboratory experiments using results of the AutoDomainMine estimation, e.g., [36]. The simulation users have conducted 62 tests with AutoDomainMine. Table 4 portrays the winning displays in simulation tools. Again, we notice high estimation accuracy. Here, the **Multiple Conditions Representatives** (**MCR**) for conditions and the Summarized Representatives for graphs are winners in most tests (shown in italics). This is probably because simulation tool users generally want to use ranges of information to increase sample space of the simulations, but they also care about complexity since the simulations are time-consuming. Hence, there is a trade-off here.

Intelligent Tutors: Intelligent tutoring systems are designed to play the role of a real human tutor in a given subject via a computerized environment, e.g., [15]. In many scientific domains, they can be used to study in detail the behavior of processes. Hence, the computational estimation provided by AutoDomainMine serves as a means to provide computer-based tutoring in the given domain. Totally, 37 tests have been conducted by users in this category. Table 5 indicates what type of display was found the best by the users of these applications. From here it is clear that in most cases the **All Conditions Representatives** (**ACR**) are winners for conditions and the Combined Representatives are winners for graphs (italicized herewith). This is most likely due to the fact that in most intelligent tutoring applications, users are interested in learning more details about the system and are possibly not much concerned about complexity. Therefore, more detail is appreciated. It is also observed here that the estimation is found to be accurate for most tests.

Table 5. Winning Representatives from DesRept in the Application of Intelligent Tutors

Winner for Conditions	**Percentage**	**Winner for Graphs**	**Percentage**
SCR Wins	17%	Nearest/Medoid Wins	12%
MCR Wins	24%	Summarized Wins	34%
ACR Wins	*52%*	*Combined Wins*	*50%*
None Wins	7%	None Wins	4%

Table 6. Winning Representatives from DesRept in the Application of Expert Systems

Winner for Conditions	**Percentage**	**Winner for Graphs**	**Percentage**
SCR Wins	28%	Nearest/Medoid Wins	31%
MCR Wins	36%	Summarized Wins	30%
ACR Wins	32%	Combined Wins	34%
None Wins	4%	None Wins	5%

Expert Systems: In expert systems, e.g., [61], AutoDomainMine performs the task of an expert in giving advice/providing consultation based on the computational estimation. We have had 44 tests conducted by users in this category. The distribution of winning displays in expert systems is illustrated in Table 6 for conditions and graphs, respectively. As in other applications, we find high estimation accuracy. It is observed that there is a fairly good mix of winners in these applications. This could be because different users of expert systems seem to be interested in different levels of detail. In expert systems for high-level business decision-making, at-a-glance retrieval of information is important without much emphasis on detail. Some expert system users, however, focus on process optimization and need to scrutinize the information in much more detail. Hence, on the whole, we derive the recommendation that it is desirable to retain all the representatives in such applications, and to display information in increasing levels of detail.

Thus, in general it is observed that representatives designed by DesRept are found suitable in targeted applications as per user evaluation. This is a significant manner of assessment since the purpose of designing representatives is to present the output of the estimation in a suitable fashion, paying sufficient attention to detail while also avoiding visual clutter, and thus catering to user appeal. This caters to the interpretability aspect of AutoDomainMine, important in our problem definition and goals. Having hereby presented the evaluation of the LearnMet and DesRept techniques within the context of AutoDomainMine, we now focus on assessing the performance of AutoDomainMine as a whole with some pertinent aspects.

6.2 AutoDomainMine Performance Assessment and Discussion

6.2.1 Assessment of the AutoDomainMine System.

User Surveys: An important method of assessing the performance of the AutoDomainMine system is through user surveys. Many users here are the participants of biannual seminars conducted in Massachusetts by the **Center for Heat Treating Excellence** (**CHTE**), an international industry-university consortium. Users are presented with the results of the AutoDomainMine estimation and are asked to compare it with the results obtained from real laboratory experiments that have not been used during the knowledge discovery process in AutoDomainMine. If the users indicate via the survey questionnaire that the estimation is acceptable as per their satisfaction to such an extent that they would use it within a real industrial process, it is considered correct, else incorrect. Accuracy is thus reported as the percentage of the correct estimations over all the estimations assessed. The results are collected with respect to the targeted applications of AutoDomainMine

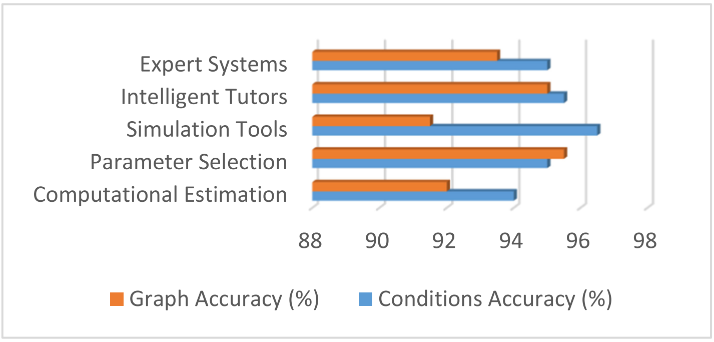


Fig. 20. Accuracy of autodomainmine in targeted applications.

Table 7. Assessment of AutoDomainMine using the Holdout Strategy

Holdout Size	Conditions Estimation Accuracy %	Graph Estimation Accuracy %	Response Time (seconds)
25	95.8%	94.6%	0.11
50	94.1%	96.3%	0.12
75	95.2%	93.7%	0.14
100	93.2%	95.4%	0.13
125	94.8%	92.5%	0.11
150	92.4%	93.1%	0.14

as described earlier, i.e., parameter selection, simulation tools, intelligent tutors, and expert systems. In addition, the general category of computational estimation is also considered based on its use for miscellaneous purposes. We have a total of more than 200 survey results. These are summarized in Figure 20 below for estimation of conditions and graphs. As seen here the accuracy is in the 92%–96% range for different categories. This indicates the domain-specific usefulness of AutoDomainMine with respect to effectiveness in targeted applications.

Holdout Strategy: We conduct evaluations of AutoDomainMine using the holdout strategy. Thus, we set aside a certain number of samples for testing that are not used for training, and vary the number of samples for different evaluations. The samples here are the real laboratory experiments used for learning in AutoDomainMine. The dataset consituting these laboratory experiments is designed using Taguchi methods often acclaimed in STEM fields [40]. Hence, the effective sample space represented by the experiments is 3 times the number of experiments themselves. We use a dataset with 500 laboratory experiments here that effectively represent a sample space of 1,500 experiments. Similar arguments can be applied to other scientific domains where conducting an experiment consumes huge time and resources. In this evaluation, we use the holdout strategy to alter the number of samples held out for testing and observe the accuracy accordingly. Besides accuracy, the response time to perform an estimation is also recorded in seconds. This is in order to assess the efficiency of AutoDomainMine. The observations from these evaluations are depicted in Table 7 herewith. As seen here, the estimation accuracy for both conditions and graph remains in the 92%–96% range regardless of the number of samples set aside for testing in the holdout strategy. For example, in the 1st row, the holdout size is 25, i.e., 25 samples for testing and 475 for training, which gives an estimation accuracy of around 95%; while in the last row the holdout size is 150, i.e., 350 samples for training and 150 for testing, and the estimation accuracy is approximately 93% here. Thus, even if we use fewer samples for training, it leads to high accuracy in

Table 8. Comparative Evaluation of AutoDomainMine

Category	**Similarity Search**	**Mathematical Modeling**	**AutoDomainMine**
Conditions Estimation Accuracy (%)	67.5%	85.7%	93.8%
Graph Estimation Accuracy (%)	69.2%	87.3%	94.6%
Conditions Estimation Time (seconds)	0.34	550	0.12
Graph Estimation Time (seconds)	0.32	525	0.15

estimation. The response time is around 0.1 second for each estimation, which is fairly consistent. Since this is speedy, it indicates the high efficiency of the AutoDomainMine system.

Comparative Studies: There are comparative studies attempted with AutoDomainMine and techniques such as naïve similarity search, domain-specific mathematical modeling, neural networks, and Bayesian classifiers. AutoDomainMine outperforms similarity search and mathematical modeling for estimation of graphs as well as conditions as seen in Table 8. These numbers are averages obtained after more than 10 executions.

Regarding Bayesian classifiers, there is lack of prior information on probability estimates required therein due to which it is not feasible to proceed further with comparative evaluation. Obtaining such data would entail another piece of contribution per se. Neural networks present a black box due to which we cannot obtain paths for reasoning in order to conduct the estimation. The same argument can be applied for deep learning based on neural models. In general, symbolic approaches present some advantages over neural models and vice versa based on the nature of the application. These are discussed in an ACM WSDM 2021 tutorial [50] where knowledge bases are addressed in addition to deep learning. Our research in this article falls under the paradigm of symbolic approaches that are more suitable when comprehensibility and explicability are desirable.

A brief discussion on the computational complexity of AutoDomainMine is presented herewith. It incurs a one-time process of knowledge discovery that entails an integration of clustering and classification, as well as a recurrent process of estimation that involves finding the closest match with decision tree paths (for estimation of the graph, given the conditions), and finding the closest match with representative graphs of clusters (for estimation of the conditions, given the graph). Hence, these steps have complexities as follows.

$$Knowledge\ Discovery\ \ (KD),\ one-time:\ KD\ Complexity =\ O\left(dkI\right) + O\left(dlog_2 d\right) \tag{19}$$

$$Estimation\ of\ Graph\left(EG\right), recurrent:\ EG\,Complexity = O\left(H\right), \tag{20}$$

$$Estimation\ of\ Conditions\left(EC\right), recurrent:\ EC\,Complexity = O\left(k\right), \tag{21}$$

$$\begin{aligned} Complexity\ of\ AutoDomainMine\left(ADM\right):\ ADM\,Complexity &= O\left(dkI\right) + O\left(dlog_2 d\right) \\ &+ R_1 \times O\left(H\right) + R_2 \times O\left(k\right). \end{aligned} \tag{22}$$

Here, d is the number of data samples, k is the number of clusters, I is the number of iterations in clustering, H is the height of the decision tree, whereas R_1 and R_2 are the number of times the graphs and conditions are estimated, respectively. In the knowledge discovery step, the complexity of clustering is the standard value of $O(dkI)$ using a clustering algorithm such as k-means, while the complexity of classification is another standard value $O(dlog_2 d)$ using decision tree induction with an algorithm such as J4.8 or equivalent [39]. These add up to the complexity shown in Equation (19). In the estimation of graphs, the tree path matching involves the complexity of searching the decision tree which is typically proportional to the depth (or height) of the tree, i.e., H in this context as presented in Equation (20). In the estimation of conditions, the number of graph comparisons is proportional to the number of clusters, which is equal to k here as in Equation (21). If R_1 is the number of times the estimation of graphs is performed while R_2 is the number of the estimation of conditions is performed, that gives a total complexity of AutoDomainMine as depicted in Equation (22). Here R_1 and R_2 are simple numbers that would maintain the same complexity with respect to the order of magnitude O. Hence, on the whole, the complexity of AutoDomainMine is linear/logarithmic. It does not attain quadratic or higher order complexity.

This is further corroborated by the fact that the estimation time values shown in Table 8 for AutoDomainMine are very low. They are somewhat lower than those of Similarity Search (1/3 to 1/2 times lower) and distinctly lower than those of Mathematical Modeling (approximately 4,500 to 3,500 times lower). Hence, the complexity and efficiency of the AutoDomainMine framework is commendable. Comparative studies are conducted with these 2 techniques as synopsized here. Any more work on comparison could be a matter of future research.

There are studies conducted in the literature on real networks, some of which may seem potentially relevant to this entire realm of scientific data mining. For example, considering applications in domains such as Biology and Physics, Squartini et al. [56] propose an efficient analytical approach based on maximum entropy to discover patterns in real networks. They put forth a fast technique for procuring expectation values as well as standard deviations of topological properties for binary, weighted networks that could be directed or undirected, without requiring randomized variants of the real network to be produced. They conduct experiments with H. pylori's protein network, small-size directed food webs, geospatial systems such as airport networks and interbank webs and so on. Their results indicate that maximally random networks depict varied behaviors that can be rather sensitive to certain constraints. This research aids identification of pertinent data in real networks. In more recent work, Liu et al. [35] develop a **generalized mechanics model (GMM**) that can be useful to recognize the significance of nodes in multi-faceted real networks. Their research stands out by harnessing local and global data for effectively distinguishing vital nodes, such that it can be beneficial in several applications spanning scientific domains. In their work, a novel approach is propounded that encompasses network-based quality evaluation in an empirical manner in order to assess the technique of spotting crucial nodes in real networks that can be highly complex. While such research on real networks is deemed to be highly useful in various contexts, it does not appear directly relevant to the specific problem we discuss in this article. Hence, we do not conduct comparative studies with such approaches though they provide significant contributions. As stated earlier, any further extensions and comparisons with respect to our research in this article can potentially be addressed in future work as needed, based on further issues emanating from our research.

6.2.2 *Impacts and Extensions.*

The AutoDomainMine approach along with its components of LearnMet and DesRept have aroused interest among other researchers. Ever since some part of this work has been published [63–67], it has received attention as evident from some publications. We mention a few examples of the impacts here.

LearnMet Impacts: The fairly recent work of Eker [11] deals with clustering methodologies in the area of material selections with a focus on quality. In this regard, they find the work on LearnMet [65] useful since it proposes a technique for learning a domain-specific notion of distance useful in clustering by capturing adequate semantics. This is particularly important when the quality of the clustering is critical as per the requirements of the domain. Hence, LearnMet is interesting here in assisting with the application of material selection via its contributions to clustering by distance-metric learning.

Another piece of research that finds LearnMet worthy of mention is by Lei [30]. This research is in the paradigm of big data in general with specific focus on recurring query processing. It proposes a novel scalable framework called Redoop for recurring querying on big evolving data. Among the many *Vs* of big data, the LearnMet technique touches upon the variety aspect and arouses curiosity in works such as these.

Xiang et al. [69] cite LearnMet in their work on distance-metric learning in the generic Mahalanobis family of distances. Their research published in the Pattern Recognition journal proposes a technique to learn such a metric for the purpose of clustering and classification. Since LearnMet delves into different types of distance metrics including position-based, statistical and critical distances, and asserts the fact that the learned metric is useful as a notion of similarity in clustering, it is found worthy of citation in this journal article that addresses Mahalanobis distance metric learning.

DesRept Impacts: Arcas et al. [3] focus on research pertaining to acquiring general concepts that depict a set of objects via the deployment of ontologies helpful in intelligent systems. They find the DesRept approach interesting with respect to its DesCond component [64] that designs representatives for sets of conditions in scientific experiments so as to preserve semantics with different levels of detail. There can be an analogy drawn between DesCond and the entire concept of ontology derivation and usage, significant in numerous applications within intelligent systems. Researchers Martinez et al. [38] find the DesCond technique significant as well. They mention it in their work in the Knowledge-Based Systems journal pertaining to centroid construction for data with textual parameters emphasizing on semantically-grounded development. Since DesCond deals with some plain text from input conditions of experiments and focuses on harnessing semantics therein for constructing suitable representatives, it is cited in the given journal article addresses textual data for building centroids.

Likewise, Chandra Shekar et al. [8] find DesRept useful in terms of its DesGraph component [66]. They address the issue of knowledge discovery in the context of conducting computational estimation in the domain of mechatronics taking into account with respect to aluminum alloys. In their research, they address domain-specific mechatronics perspectives as well as graphical data. Our work on representative design for graphical plots emerging from scientific experiments provides significant foundations here serve as the basis for data mining over the concerned graphical data and deriving useful inferences for estimation in mechatronics.

AutoDomainMine Overall Impacts: Haris et al. [20] present interesting research in the overall area of optimization and data mining with the goal of decision-making as applicable to computer applications and information systems in general. They refer to our work on the AutoDomainMine system demonstration [67] where we present this as a tool for computational estimation based on graphical data mining with its objectives entailing process optimization. For example, the estimation performed by AutoDomainMine can be useful in predicting concerned experimental parameters in advance so as to enhance the overall processes. Hence, our work is found useful in decision-making to support process optimization.

It is notable that the AutoDomainMine system demo [67] has been cited in the Encyclopedia of Iron, Steel, and Their Alloys published by Taylor and Francis [24]. This is edited by Totten et al., author of a classic textbook useful in the overall Mechanical Engineering area [57]. The specific chapter that cites AutoDomainMine in this encyclopedia is by Hernández-Morales [24] and focuses on the detailed analysis of scientific plots or curves obtained as the results from laboratory experiments. In this respect, the AutoDomainMine system that actually uses such scientific plots for conducting analyses in order to draw inferences to perform computational estimation is considered beneficial. Hence, we notice that AutoDomainMine is deemed impactful enough to receive a mention in an encyclopedia.

There is recent research in the realm of spatial clustering within the area of hierarchical modeling by Khairullah et al. [28] at KU Leuven Belgium (among the top 50 universities worldwide). This piece of research cites the overall AutoDomainMine approach from its initial AAAI publication [63]. Our short paper therein attracts attention in terms of the manner in which it proposes the harnessing of clustering and classification to automate learning strategies of scientists with the aim of estimating process variables. Accordingly, it is advantageous in the broad realm of spatial clustering with specific reference to data modeling for plasticity as mentioned in this recent work [28]. These are some works that refer to AutoDomainMine and its components.

Discussion and Applications: Considering the above examples, it can be claimed that AutoDomainMine and its components LearnMet and DesRept have received attention among researchers in Computer Science as well as other domains. Consequently, our current journal article narrates the big picture of AutoDomainMine along with the specific details of the concerned techniques and their evaluation. This is the first journal article describing AutoDomainMine elaborately, motivated by its overall reception and the impacts after its inception. We anticipate that this would be useful to numerous researchers and practitioners. It can serve as the basis for the development of customized tools and applications in specific domains. It can also be useful in fostering further multi-disciplinary research with contributions to data mining and related fields.

Furthermore, studies in this area are ongoing including the development of suitable methodologies inspired by AutoDomainMine and its components. Regarding adaptation, techniques drawing parallels with LearnMet have been designed by us in some interesting works. The FeaturesRank method [62] has been proposed for learning to rank significant features from scientific images in domains such as Nanotechnology and Bioinformatics and has been found beneficial to researchers from departments such as Electrical Engineering and Biology. In recent research [10], we have developed a technique known as MetChar for handwritten Chinese character recognition that addresses the interpretability aspect of distance metric learning among others. Analogous to LearnMet in graphical plots from scientific experiments, the MetChar approach builds upon components. It addresses components in Chinese handwriting pertaining to aspects such as the longest vertical stroke, the longest horizontal stroke, and so on in the writing of a character without requiring additional details on its pinyin and English equivalents. The relative importance of components is learned with different methods including classical greedy and exhaustive approaches as well as a hybrid approach stated therein. This work makes contributions to **Optical Character Recognition** (**OCR**) in an interpretable manner. It can be used for handwriting detection in scientific fields as well as others involving calligraphic textual data that entails pictorial writing. Note that Romanization via pinyin is not needed, and interpretability is facilitated.

Potential applications in various domains involving graphical plots and scientific images can be considered based on suitable problem definitions, availability of data and needs of researchers. Graphs analogous to those in AutoDomainMine are found in various fields. Examples are illustrated in Figure 21 herewith [2]. These are from Geoscience, depicting distance-time plots with domain-specific aspects. Since we have ongoing research with colleagues in Earth and Environmental Science, such data can provide inputs for investigatory studies on computational estimation, modeling/simulation, data visualization, and mobile application (app) development. Other examples of such graphs are in Figure 22 from the domain of Biology [18]. Such graphs contain much semantics as evident from the plots and the associated captions. Techniques such as LearnMet can be suitably adapted here for preserving the semantics in simulation studies, pedagogical systems, app development, and so on.

The impacts of our proposed framework can thus be further explored in such targeted applications. This would be in line with our erstwhile research collaborations that have already led

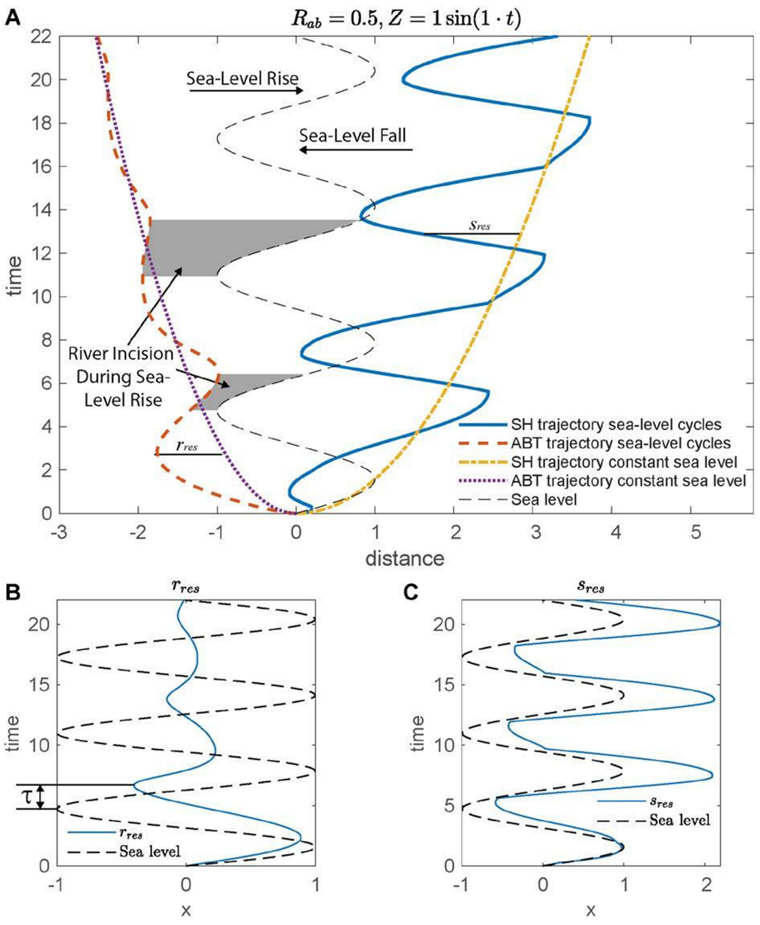
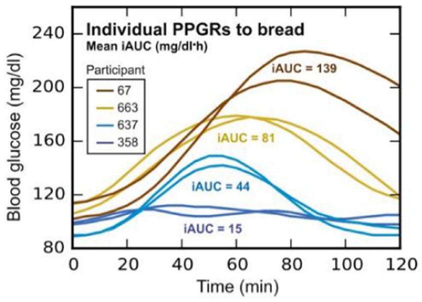


Fig. 21. Examples of graphical plots in the domain of Geoscience [2].



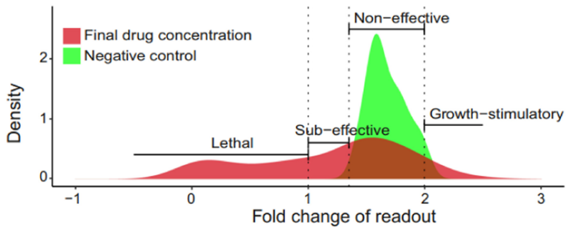


Fig. 22. Left – Post Prandial Glycemic Response (PPGR) as defined by iAUC (incremental Area Under Curve) with respect to blood glucose; Right – Categorization of medicines using fold change (a term describes how much a quantity changes between an original and a subsequent measurement) in different conditions [18].

to successful outcomes e.g., [25, 48]. Likewise, we are planning on working with colleagues in Nanotechnology on related problems that stem from the research proposed in this article. One such problem involves studying the nanostructure of titanium alloys based on machine learning models [70]. In this process, approaches such as LearnMet that are used to learn domain-specific notions of distance from graphical plots can be further adapted to learn notions of similarity for images involving nanostructures. Some of this has been addressed by us in our approach Features-Rank [62] mentioned herewith. Furthermore work in such area can lead to interesting applications with real-world impacts including selection of lightweight alloys for dental implants, kneecaps, and other applications. Such avenues of research constitute our ongoing work.

The overall usefulness of the proposed framework AutoDomainMine and its facets of LearnMet and DesRept have already been noticed through citations of the work in the previous subsection on “Impacts and Extensions”. We can summarize the following applications there.

- Clustering methods while choosing materials, e.g., [11].
- Query processing in a recurrent manner, e.g., [30].
- Distance-metric learning, e.g., [69].
- Ontology usage in intelligent systems, e.g., [3].
- Centroid construction for data, e.g., [38].
- Computational estimation in mechatronics, e.g., [8].
- Optimization in decision support, e.g., [20].
- Analysis of scientific plots, e.g., [2].
- Spatial clustering in data modeling, e.g., [28].

In general, we can list application areas for our own ongoing work based on the contributions of this article. Some of this work can be pursued with colleagues in Earth and Environmental Science analogous to [25, 48], and with researchers in Nanotechnology as in [70]. These application areas are as follows.

- Parameter selection.
- Simulation tools.
- Intelligent tutors.
- Expert systems.
- Data visualization.
- App development.

Note, however, that we make a clear disclaimer here: the proposed framework can only be used to guide applications such parameter selection, expert systems, and so on. The learning is clearly heuristic and is not based on formulas that are guaranteed to work. The entire paradigm in this work is heuristic. The word heuristic has Greek roots. It is derived from the word “heuriskein” that somewhat translates as “to find out”. It is believed that Archimedes actually said “Heureka” when he discovered the principle of flotation, and this later became the popular expression “Eureka!” which means “I have found it”. Hence, heuristic paradigm typically refers to experience-based techniques for problem-solving, learning, and discovery [39]. It is not always accompanied by theorems and proofs. We do not provide theorems and proofs in this work. Yet, in the real world, heuristics are often found to solve problems well and are often used in practice. This has been found in our research in this article and in other works by our research teams. There are indeed advantages and disadvantages of heuristic-based approaches, some of which are discussed in the article [60] that addresses comparisons between mathematical and heuristic approaches for scientific data analysis.

7 RELATED WORK

7.1 Data Mining Techniques and their Integration with Suitable Applications

Rule-Based and Case-Based Approaches: Some domain-specific problems have been solved by integrating rule-based and case-based approaches. For example, in the domain of law [44], rules are laid down by the constitution and legal cases solved in the past are typically documented. In dealing with a new case, a legal expert system works as follows. It applies rules relevant to the new case and retrieves similar cases in the past to learn from experience. These approaches combined can predict a more accurate solution to the new case, than either approach individually [44]. They have been used for cases that involve text-based documents which is common in domains such as law. However, it is not feasible to apply these to graphs found in our context. If for example, one input condition differs between the old and the new case, then the knowledge about the difference of conditions is not sufficient to modify a graph from the old case as a solution for the new one.

Classification and Association Rule Mining: Liu et al. propose an integrated framework called associative classification that combines classification and association rule mining [34]. Classification aims to discover rules in the database that form a classifier. Association rule mining finds all rules in the database with minimum support and confidence constraints. The proposed framework focuses on mining a special subset of association rules called **Class Association Rules** (**CARs**) [34]. This is useful in quantitative and categorical applications. However, in our context since graphs are involved, the data is more complex and it does not seem feasible to apply association rules considering frequent itemsets, minimum confidence, and minimum support. Clustering is a more intuitive method of dealing with graphs because it groups them based on their similarity.

Association Rules and Clustering: An **Association Rule Clustering System** (**ARCS**) is proposed which clusters 2-dimensional association rules in large databases [53]. It clusters association rules of the form $A \wedge B \Rightarrow X$ where Left Hand Side (A,B) are quantitative and Right Hand Side (X) categorical, e.g., (age = 40) ⇒ (ownHome = yes) and (age = 41) ⇒ (ownHome = yes) are clustered as: (40 ≤ age ≤ 42) ⇒ (ownHome = yes). They define segmentation as the collection of all clustered association rules for a specific value X. Their goal is to find the smallest number of clusters that cover association rules in a segmentation. However, their requirements are that the LHS is numeric and the RHS is categorical. These are not applicable to our problem which involves a significant amount of graphical data.

Deep Learning and Transfer Learning: The paradigm of deep learning with neural models such as CNN and others [17, 29] is sometimes integrated with transfer learning where knowledge discovered from one data set can be transferred to another often bigger data set, typically bigger than the original. Computer vision models such as VGG-16, VGG-19 [54], and ResNet-101 [21] are often utilized with transfer learning in applications involving imagery. For example, classification of COVID-19 versus pneumonia symptoms from online chest X-ray data to distinguish them from X-rays of healthy patients is conducted [26] with high detection accuracy using a few images from big data in a benchmark open source. Other computer vision models are used for studies in pathological brain image classification [14] to aid medical diagnosis with successful results. Deep learning resonates with biological foundations of our brain and has a plethora of applications spanning numerous domains with research on advancing its capabilities [12, 16, 42, 46]. However, in our work in this article, the fundamental concept of neural models itself is not quite the best choice since it is important for us to reason about causes of similarities and differences among the concerned images (graphs) whereas a neural network is typically a black box. Reasoning in our work is achieved more appropriately via decision trees having clear paths for comprehensibility, hence leading to the estimation of graphical plots and experimental conditions as per the problem requirements. In fact, the advantages of symbolic models over neural models and vice versa are surveyed in a recent tutorial addressing deep learning as well as knowledge bases [50]. Aspects such as explicability are mentioned here as advantages of symbolic models. As regards transfer learning, we consider that to be future work. Knowledge discovered from our work here can potentially be transferred to other tasks of a similar nature. While some of this work has been found useful by others already, there is scope for further research.

Similarity Search, Regression, and Bayesian Classifiers: A simple similarity search can potentially be executed on any data to obtain results useful in predictive analytics. Methods such as nearest neighbors in high-dimensional spaces [22] can be used in similarity search over graphical plots and other data. Techniques in the area of mathematics and statistics, such as linear/polynomial regression can be used for estimation [43]. Some of these encompass classical paradigms in use for several years [19, 39]. In our work, we follow the fundamental concept of similarity search and regression-based learning in some of our sub-processes since much of our data can be mapped to numeric variables. This can also be considered analogous to the fundamentals of backpropagation learning in ANN [52]. However, we step beyond such work in order to adequately design methodologies specific to scientific domains involving graphical data in experiments. We address several concepts such as harnessing domain semantics in clustering, capturing appropriate cluster representatives for classification, and automating learning methods of scientists in areas potentially relevant to estimation [2, 37, 55, 57, 59]. Likewise, models based on the fundamental notion of Bayes Theorem with probabilistic concepts could perhaps be used in such type of work [39]. However, these would require the calculation of probability values for the respective instances and we do not have prior data on that due to which such models alone would not be feasible. Furthermore, research on Bayesian classifiers in this context would entail a different set of contributions. Similar arguments may be applicable to other techniques that arouse the readers' interests. Upon a detailed study of various techniques, we found it viable to propose a method based on the integration of clustering and decision tree classification in such a manner as to automate learning strategies of scientists for computational estimation in scientific domains. This unique method of automation to mimic scientists' learning strategies along with related issues such as domain semantics has been found interesting in the Artificial Intelligence and Data Science communities [63–67].

Dynamical Clustering: A novel dynamical approach has been recently proposed by Li et al. [31] for clusters of complex networks so as to ascertain optimal cluster configuration convergence by detecting configurations promoted by key leaders in communities. Considering the optimization of a quality function, nodes are allotted to clusters driven by some precomputed leaders that are determined by analyzing a 2-stage game in which leader group members make contributions to the follower group. This quality optimization serves the purpose of convergence to the optimal situation within a limited number of iterations. Their proposed approach achieves high efficiency, in particular for sparse networks, such that its complexity is almost linear. This approach is found to be highly useful in e-commerce to detect important clusters and optimize the behaviors. However, based on the experimentation and applications discussed in this work, it seems to be mainly relevant to social networks and e-commerce, e.g., users buy products since these were promoted by their friends and hence it is beneficial to investigate the hidden user relationships in order to foster e-commerce. It would be useful for clusters or communities in Amazon, Facebook, Twitter, and so on. Our work in this article relates more specifically to scientific data mining where we do not have such communities based on leaders and followers. Hence, these specific techniques, though very interesting, would not be highly feasible in our context. We propose other methods.

7.2 Distance Metric Learning and Feature Selection for Complex Data

Metrics for Graphical and Multi-media Data: An **Approximate Neighborhood Function** (**ANF**) for comparison is proposed by researchers [45] such that it focuses on a very fast scalable method for mining, however, compromising somewhat on accuracy. For our goals, accuracy is very important. Chen and Ozsu [7] compare different metrics for similarity-based retrieval of time-series data. However, they do not emphasize domain semantics, and do not learn a combined metric involving several individual metrics. During our early studies in this area prior to proposing LearnMet, we conducted baseline experiments with similarity-based retrieval. Upon discussion with domain experts we found that there were anomalies in clustering, e.g., an experiment with and without a certain critical region were placed in the same cluster [55, 65]. This had adverse effects on accuracy and interpretability. Hence, we devised a technique for domain-specific distance metric learning.

Keim et al. [27] present an overview of various distance metrics for similarity search useful in multi-media databases. They focus on the content-based retrieval of similar objects. However, they do not propose the learning of a single distance metric that combines various components. Nor do they define the concept of critical distance that is extremely important in our targeted domains. Our focus is on the detailed analysis of specific graphs, as opposed to a general search over different categories of data.

Neural Networks, SVM, and Ensemble Learning: Neural networks could possibly be used for distance metric learning [39, 52]. However, our data is such that the distance between pairs of plots is not known in advance for supervised learning due to which we prefer to go with a clustering-based approach (unsupervised learning) in order to learn a suitable distance metric. Similar issues hold for other learning techniques such as SVM, i.e., **Support Vector Machines** (**SVM**) [39] since we do not have positive and negative training samples available in advance as required for learning. Zhou et al. [71] propose an approach for ensemble neural networks. They train a number of neural networks at first, then assign random weights to them and employ a genetic algorithm to evolve the weights to characterize the fitness of the neural network in constituting an ensemble. Although, we do not use neural networks, each distance metric in our problem could possibly be viewed as a learner, thus in combining them we get an ensemble. We can thus draw some analogy here in our process of distance metric learning in LearnMet. There may be scope for potential improvements during the adaptation of our approaches in other domains. This could leverage the use of deep unsupervised learning methods that we have not heretofore explored. This is an aspect of future work that calls for further study as needed.

Dimensionality Reduction in Complex Data: There is a vast body of research in the area of dimensionality reduction especially as data heads towards big data. In addition to classical approaches such as Principal Components Analysis and Singular Value Decomposition [19, 39], recent deep learning advances such as Autoencoders [17, 29] can be useful in dimensionality reduction. Other works include Chernoff dimensionality reduction where Fischer and Fukunaga Koontz transforms can be harnessed [49]. In our work in this article, we have deployed a fundamental approach for dimensionality reduction via Fourier transforms as found useful in some applications in the literature [1]. This has been considered suitable in our work with respect to our early experiments conducted. However, if any other concerned domain of deployment has bigger data or entails data of different types, the adequate dimensionality reduction method can be selected accordingly. This can be dealt with in an application-oriented manner, addressing the respective *Vs* of the big data such as volume, velocity, variety, and veracity. If the data in the concerned domain is small data, there may not be the need for dimensionality reduction. Hence, this is applicable with reference to context.

Streaming Feature Selection for Multi-Label Data: Recently, streaming feature selection methods have been devised by Paul et al. [47] for multi-label data such that the multiple labels therein are reduced to a lower dimensional space. In this approach, the authors first categorize similar labels together and thereafter conduct the selection in order to improve the quality as well as efficiency. They deploy a multi-objective rendering of the cuckoo search approach in order to choose the optimal feature set by proposing a new objective function, and propound 2 adaptations of the streaming feature selection method. One of these is applicable to the features appearing individually while the other one is suitable when the features are found to be entering as a batch. In order to enhance the efficacy of attaining feature-relevance as per the class labels, they harness label co-relation to categorize related labels. They utilize different multi-label datasets including those from biology, audio, and social media for testing their streaming feature selection methods, and obtain effective results. While this work is impressive, it is mainly pertinent to multi-label streaming data. In our work, the data is not of that nature. Hence, we do not require these methods. Yet, they are notable advances in the field and are worthy of mention.

Feature Level Data Fusion: There is an interesting concept known as feature level fusion, whereby feature sets that are produced from various algorithms on either one common dataset or multiple datasets are joined or fused together such that they depict the concerned data. In this area, there has been contemporary research on linear fusion of numerous signal matrices encompassing noise wherein the given features can be stored as eigenvectors. An algorithm called EigFuse [32] is proposed that helps to accurately predict the latent signal eigenspace. This enables optimization despite various noise levels. The authors encapsulate random matrix theory within their solution and thereby obtain outputs based on the product of features from multiple matrices. They achieve high efficiency in this process as corroborated by their experiments comparing their work with other approaches in the literature that deal with several noise levels. This research is very useful in cases where such fusion is advisable, as shown in the experimentation across a variety of fields including health monitoring, human activity detection, remote sensing, and image segmentation. With respect to our work, we do not deem such fusion as being imperative so far, given our overall problem definition, data sources and the targeted applications. At a later juncture, if we require this type of fusion while addressing some future issues emerging from our work, we would certainly consider the potential deployment of such approaches.

7.3 Designing Suitable Representatives in Various Contexts

Design of Representative Objects for Databases and Web Information: Some researchers [4] address similarity search in database systems by visualizing hierarchical clustering structure of a database of objects to speed up the search. They consider reachability plots to extract significant clusters in hierarchical manner along with suitable cluster representatives. Their constraint is that the representative must be a real object of the data set. We do not have this constraint, thus giving us the freedom to consider alternative design strategies.

Other interesting work [23] involves building representatives for web information. They have user-interfaces for web-based applications. They use an approach of image rating based on data quality in terms of color, clarity, and frequency of access by targeted users. The image with the highest rating is the representative image for a given group. They employ manual selection of representatives by groups of targeted users. Their methods involve considerable user-intervention while actually building the representatives which is not desirable in our problem. Moreover, in our work the data quality per se of individual graphs in the cluster is quite similar.

Evaluation of Measures for Summarization and Clustering: Nomoto et al. propose text summarization based on exploiting diversity concepts in text [41]. They propose an information-centric approach where text summaries are judged by how well they represent source documents in retrieval and text categorization. They use the MDL principle [51] to determine the number of clusters needed for text summarization. Their MDL encoding takes into account captures the probability of occurrence of words, the number of parameters, and the number of data objects. However, they neither construct nor evaluate different types of representatives.

Another piece of research [5] utilizes the MDL principle to evaluate how a clustering algorithm aids a classification algorithm. They consider a distribution over (X, Y) where X is the input and Y is a hidden label. The assumption is that Y can take one of L possible values where $L > 1$. If c is the number of clusters, r is the number of initializations of the clustering algorithm, and s is the number of clustering algorithms considered, the MDL encoding is given by: $Length = c \log L + \log r + \log((c-1)c) + \log s$. Minimizing this length gives the best set of clusters. However, they only evaluate clustering in the context of classification. They do not actually design cluster representatives and choose the best one for each cluster to aid classification through proposing objective evaluation measures capturing subjectivity in domain-specific contexts.

Similarity Measures for Categorical Data: An algorithm called **Iterated Contextual Distances** (**ICD**) is proposed to learn distances between attributes based on such inter-dependencies [9]. The ICD algorithm starts with an arbitrary distance function between attributes to derive a vector representation for rows, which gives a vector representation for sub-relations. The sub-relation distance function is used to get a new distance value for attributes. Starting with random initial values, ICD converges to stable distances between attributes. However, the type of inter-dependencies that they define do not exist in our datasets.

Learnable similarity measures for strings are presented [6] based on SVM and **Expectation Maximization** (**EM**). These measures are applied for duplicate detection. They deal with natural language text strings and the involved semantics, while our data is different. We work with domain-specific input conditions that involve a mixture of attributes such as numeric, categorical, and ordinal. We do not deal with strings of text whose meaning has to be interpreted in a broader natural language context. Moreover, in our context domain knowledge has already been derived from decision trees and can directly be applied.

Communities for Partially Observable Data: Community structures can often represent significant parts of networks since they help to understand the topology and functions of the concerned networks. The detection of such representative communities can be rather challenging when the data is only partially observable, e.g., when social network data is built from missing nodes and edges (which occurs frequently). Tran et al. [58] focus on the problem of finding overlapping community structures in such incomplete networks such that these communities can adequately represent the network topology and functions. A novel approach called KroMFac [58] is proposed to perform community detection through regularized nonnegative matrix factorization via the adaption of the Kronecker graph model. In their approach, they predict the missing part of the network, typify and choose influential nodes by ranking, insert them within the prevailing graph, and then unveil the community structures by likelihood maximization of the given graph, thus achieving optimization. KroMFac is evaluated over real as well as synthetic networks, and proves to be highly effective compared to the state-of-the-art. This approach seems really feasible in cases where there is missing, hidden, and partially observable data. In our research within this article, the data is fully observable. Hence, such approaches are not essential for our work. Such work stands out in the context of the overall literature in the area, due to which we mention it herewith.

Likewise, there is much research in the areas pertaining to our article and the body of research is ever-growing with the demands of big data, deep learning, and domain-specific applications. We have hereby abridged a few pertinent studies as broadly applicable to our work with the disclaimer that there might be other works potentially relevant, and that any technique explored or surveyed herewith probably arouses curiosity in the area of another technique. More work in related areas is ongoing with further research and is expected to provide even better advances in the near future and in the long run.

8 CONCLUSIONS

This research entails a computational estimation approach for scientific domains called AutoDomainMine based on a framework that integrates clustering and classification to automate a typical learning strategy of scientists. AutoDomainMine performs clustering followed by classification to discover knowledge from existing scientific experimental data. Graphs from existing experiments are clustered using a suitable clustering algorithm. Decision tree classification is used to learn clustering criteria from which a representative pair of input conditions and graph is designed per cluster. Decision trees and representative pairs are the knowledge discovered from existing experiments, used for estimation. Given the input conditions of a new experiment, the closest path of the decision tree is traced to estimate its cluster. The representative graph of that cluster is proffered as the estimated graph for the experiment. Given a desired graph, the closest matching representative graph is found and the corresponding conditions are tendered as the estimated conditions to obtain the given graph.

AutoDomainMine has been evaluated rigorously using real data with the holdout strategy as well as user surveys. It gives estimation accuracy of around in the range of 92%–96%, approximately. It performs the estimation in distinctly less time than a laboratory experiment, does not require manual intervention during estimation, and adheres well to the needs of interpretability and explicability. Applications of AutoDomainMine include parameter selection, simulation tools, intelligent tutors, and expert systems. The main contributions of this work are as follows.

(1) The AutoDomainMine Learning Strategy for Computational Estimation
- Proposing an integrated framework of clustering and decision tree classification for estimation, thereby automating learning methods of scientists.
- Suitably adapting clustering techniques originally developed for points to graphical curves.
- Developing search methods for approximate matching over decision trees for estimation.
- Implementing a system for computational estimation using the AutoDomainMine approach.
- Conducting evaluations with real data catering to various targeted applications.

(2) LearnMet for Domain-Specific Distance Metric Learning
- Assigning distance components with basic domain knowledge.
- Defining a suitable notion of error for graphs.
- Proposing appropriate weight adjustment heuristics.

(3) DesRept for Designing Semantics-Preserving Representatives
- Deriving a notion of distance for sets of conditions.
- Developing strategies to design candidate representatives.
- Proposing encodings for comparison of candidates to find winners.

As the contributions of this work are listed, it is important to make a disclaimer here. The research presented in this article models a type of hypothesis generation and "learning" that scientists often accomplish, however, it does not attempt to truly mimic the creative human task of generating new explanatory knowledge. In other words, the proposed techniques here certainly *do not* automate scientific discovery. The essence of scientific discovery entails deeply human creative tasks. The work in this article therefore does not enter the domain of AGI, i.e., Artificial General

Intelligence. Instead, it simply provides useful insights into learning methods already used by scientists and thereby helps to achieve computational estimation via predictive analysis. *Note that while there are many advances in related fields, our research described in this article provides the 2 cents to the area of computational estimation in scientific domains by automating some learning methods of scientists via an integrated framework based on data mining techniques.*

While the advantages of this research have been touched upon at various places within this article, it is also important to briefly dwell upon its disadvantages and limitations. One of the main issues is the fact that AutoDomainMine needs data from real laboratory experiments in order to proceed with the knowledge discovery and estimation. In the absence of such data, the computational estimation cannot be performed. This is in contrast with mathematical models where simulations can be conducted using the formulas available to model the concerned processes. This is one of the advantages of mathematical modeling over AutoDomainMine. However, the mathematical models needed to perform simulations may not always be available, and in many cases, they take too long to execute. In such situations, the heuristic estimations provided by AutoDomainMine would be useful. Another limitation of this work is that some interaction with domain experts is needed, e.g., in the process of assigning suitable components to distance metrics for the graphical plots, and in conducting the evaluation with respect to targeted applications to ascertain the effectiveness of the framework in order to release it for usage. This is in contrast with similarity search that can proceed without any domain expert intervention, and hence that is an advantage of similarity search over AutoDomainMine. However, much of this interaction in AutoDomainMine is just a one-time process, not required recurrently for performing computational estimation. One more disadvantage of this overall work is that we have conducted comparative evaluations with just a few techniques such as similarity search and mathematical modeling. This is done since they are deemed the most relevant techniques with respect to the specific problem defined in our research. Moreover, since the AutoDomainMine framework has been evaluated in the context of targeted applications and has lived up to user expectations, this is considered sufficient as of now. Furthermore, the impacts of the work given its current state of evaluations are already seen in the subsection on "Impacts and Extensions". As this work is extended further, we would consider including further comparative studies on our roadmap as needed.

This research has tremendous potential for future work. Some of the specific issues we would address in further research are as follows:

- We aim to investigate the specific adaptation of AutoDomainMine, LearnMet, and DesRept in other domains such as Geoscience [2], Nanotechnology [33, 70], and Electrical Engineering where estimation techniques in general [59] are useful. We have interacted with scientists with respect to outlining sub-problems where some of our research would be beneficial.
- We would consider developing customized tools to address some targeted applications here, e.g., Expert Systems and Intelligent Tutors. One such system called QuenchMiner [61] has already been developed as an Expert System in the Heat Treating of Materials. Likewise, other tools can be developed that would cater to the specific needs of users for other applications. Pedagogy would be an important aspect here that we would address while developing Intelligent Tutors.
- We could potentially explore neural networks and deep learning [17, 29, 54] in this context, especially in domains where there is a huge amount of training data available as required for such learning. Since these paradigms are used in several data mining studies, it would be interesting to investigate them with respect to the research sub-problems such as those outlined in this article.

- We can probably head towards expanding this general avenue by building knowledge bases that integrate commonsense knowledge [50] along with domain-specific knowledge, and consider potential use cases in the context of the overall research here.

Other future work includes possibly investigating transfer learning with respect to our research, considering the development of mobile applications (apps) that disseminate our findings, interacting with other professionals to build customized tools based on pertinent work here, and pursuing cross-disciplinary avenues indulging in educational goals. The AutoDomainMine approach and its components can be usable and adaptable in various domains that involve graphical data, especially scientific fields. This article can be of interest to the scientific databases, data mining, and machine learning communities, as well as multi-disciplinary professionals working on R&D and education in facets of computational science.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. 1983. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*. 69–84.

[2] W. Anderson, J. Lorenzo-Trueba, and V. Voller. 2019. A geomorphic enthalpy method: Description and application to the evolution of fluvial-deltas under sea-level cycles. *Computers & Geosciences Journal* 130, (2019), 1–10.

[3] F. M. Arcas. 2012. *Obtaining General Concepts that Represent a Set of Objects using Ontologies*. Master's Thesis. Master on Computer Security and Intelligent Systems, Universitat Rovira i Virgili, Catalan, Spain.

[4] S. Brecheisen, H. Kriegel, P. Kroger, M. Pfeifle, and M. Viermetz. 2003. Representatives for visually analyzing cluster hierarchies. In *Proceedings of the 4th International Workshop on Multimedia Data Mining*.

[5] A. Banerjee and J. Langford. 2004. An objective evaluation criterion for clustering. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 515–520.

[6] M. Blenko and R. Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 39–48.

[7] L. Chen and M. Tamer Ozsu. 2003. *Similarity-based Retrieval of Time-Series Data using Multi-Scale Histograms*. Technical Report CS-20003-31. University of Waterloo, Canada.

[8] D. V. Chandra Shekar, V. Sesha Srinivas, and J. Pratap Reddy. 2010. Graphical data mining and knowledge discovery for computational estimation in AI alloys. *International Journal of Mechatronics and Manufacturing Systems* 3, 1–2 (2010), 131–143. DOI: https://doi.org/10.1504/IJMMS.2010.029885

[9] G. Das and H. Mannila. 2000. Context-based similarity measures for categorical databases. In *Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery*. 201–210.

[10] B. Dong, A. S. Varde, D. Stevanovic, J. Wang, and L. Zhao. 2019. Interpretable distance metric learning for handwritten chinese character recognition. IEEE DaraCom. arXiv:2103.09714.

[11] A. A. Eker. 2015. Clustering techniques for material selections. In *Proceedings of the International Quality Conference*. 77–80.

[12] D. Garcia-Gasulla, F. Parés, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, and T. Suzumura. 2018. On the behavior of convolutional nets for feature extraction. *Journal of Artificial Intelligence Research* 61, 1 (2018), 563–592.

[13] M. Gangopadhyaya, P. Mukherjee, and B. Gupta. 2009. The resonant frequency optimization of aperture-coupled microstrip antenna using particle swarm optimization algorithm. In *Proceedings of the Applied Electromagnetics Conference*. 1–4.

[14] T. K. Gandhi. 2019. Automated brain image classification based on VGG16 and transfer learning. In *Proceedings of the International Conference on Information Technology*. 94–98.

[15] R. Gheorghiu and K. Van Lehn. 2008. XTutor: An intelligent tutor system for science and math based on excel. In *Proceedings of the International Conference on Intelligent Tutoring Systems*. Lecture Notes in Computer Science, Springer, Berlin. 5091. DOI: https://doi.org/10.1007/978-3-540-69132-7_98

[16] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. 2016. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38, 1 (2016), 142–158.

[17] I. Goodfellow, Y. Bengio, and A. Courville. 2016. *Deep learning*. MIT Press. ISBN 978–026203561.

[18] A. Gupta, P. Gautam, K. Wennerberg, and T. Aittokallio. 2020. A normalized drug response metric improves accuracy and consistency of anticancer drug sensitivity quantification in cell-based screening. *Communications Biology* 3, 1 (2020), 1–12.

[19] J. Han and M. Kamber. 2001. *Data mining: Concepts and techniques*. Morgan Kaufman Publishers, San Francisco, California.

[20] N. A. Haris, M. Abdullah, A. T. Othman, and F. A. Rahman. 2014. Optimization and data mining for decision making. In *Proceedings of the 2014 World Congress on Computer Applications and Information Systems*. 1–4. DOI: https://doi.org/10.1109/WCCAIS.2014.6916587

[21] K. He, X. Zhang, S. Ren, and J. Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.

[22] A. Hinneburg, C. Aggarwal, and D. Keim. 2000. What is the nearest neighbor in high dimensional spaces. In *Proceedings of the 26th International Conference on Very Large Data Bases*. 506–515.

[23] J. Helfman and J. Hollan. 2001. Image representations for accessing and organizing web information. In *Proceedings of the SPIE International Society for Optical Engineering Internet Imaging II Conference*. 91–101.

[24] B. Hernández-Morales. 2012. *Cooling: Curve Analysis. Encyclopedia of Iron, Steel, and Their Alloys*. (Eds.). R. Colás, G. E. Totten, Taylor & Francis Group, 1st edition. CRC Press. Boca Raton, FL.

[25] D. Karthikeyan, S. Shah, A. S. Varde, and C. Alo. 2020. Interactive visualization and app development for precipitation data in Sub-Saharan Africa. In *Proceedings of the IEEE International IOT, Electronics and Mechatronics Conference*. 302–308.

[26] D. Karthikeyan, A. S. Varde, and W. Wang. 2020. Transfer learning for decision support in Covid-19 detection from a few images in Big Data. In *Proceedings of the IEEE International Conference on Big Data 2020*. 4873–4881.

[27] D. Keim and B. Bustos. 2004. Similarity search in multimedia databases. In *Proceedings of the 20th International Conference on Data Engineering*. 873–874.

[28] M. Khairullah, J. Gawad, D. Roose, and A. Van Bael. 2017. Spatial clustering strategies for hierarchical multi-scale modelling of metal plasticity. *Modelling & Simulation in Materials Science & Engineering Journal* 25, 07 (2017), 4003.

[29] Y. LeCun, Y. Bengio, and G. E. Hinton. 2015. Deep learning. *Nature Journal* 521, 7553 (2015), 436–444.

[30] C. Lei. 2015. *Recurring Query Processing on Big Data*. PhD Thesis. Department of Computer Science, WPI. MA.

[31] H. J. Li, Z. Bu, Z. Wang, and J. Cao. 2020. Dynamical Clustering in Electronic Commerce Systems via Optimization and Leadership Expansion. *IEEE Transactions on Industrial Informatics* 16, 8 (2020), 5327–5334.

[32] H. J. Li, Z. Wang, J. Cao, J. Pei, and Y. Shi. 2020. Optimal estimation of low-rank factors via feature level data fusion of multiplex signal systems. *IEEE Transactions on Knowledge and Data Engineering*. TKDE. DOI: https://doi.org/10.1109/TKDE.2020.3015914.

[33] Y. Li, X. Wang, J. Liang, K. Wu, L. Xu, and J. Wang. 2020. Design of a high performance zeolite/polyimide composite separator for lithium-ion batteries. *Polymers* 12, 4 (2020), 764. DOI: http://dx.doi.org/10.3390/polym12040764

[34] B. Liu, W. Hsu, and Y. Ma.1998. Integrating classification and association rule mining. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*. 80–86.

[35] F. Liu, Z. Wang, and Y. Deng. 2020. GMM: A generalized mechanics model for identifying the importance of nodes in complex networks. *Knowledge Based Systems* 193, C (2020), 105464.

[36] Q. Lu, R. Vader, J. Kang, and Y. Rong. 2002. Development of a computer-aided heat treatment planning system. *Heat Treatment of Metals* 29, 3 (2002), 65–70.

[37] M. Maniruzzaman, J. Chaves, C. McGee, S. Ma, and R. Sisson Jr. 2002. CHTE quench probe system: A new Quenchant characterization system. In *Proceedings of the International Conference on Frontiers in Design and Manufacturing*. 13–17.

[38] S. Martinez, A. Valls, and D. Sánchez. 2012. Semantically-grounded construction of centroids for datasets with textual attributes. *Knowledge-Based Systems* 35, (2012), 160–172.

[39] T. Mitchell. 1997. *Machine Learning*. WCB McGraw Hill.

[40] V. N. Nair. 1992. Taguchi's parameter design: A panel discussion. *Technometrics* 34, 2 (1992), 127–161.

[41] T. Nomoto and Y. Matsumoto. 2001. A new approach to unsupervised text summarization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 26–34.

[42] S. Ogden, X. Kong, and T. Guo. 2021. PieSlicer: Dynamically improving response time for cloud-based CNN inference. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 249–256.

[43] J. Petrucelli, B. Nandram, and M. Chen. 1989. *Applied Statistics for Engineers and Scientists*. Prentice Hall.

[44] K. Pal and J. Campbell. 1997. An application of rule-based and case-based reasoning within a single legal knowledge-based system. *The Data Base for Advances in Information Systems* 28, 4 (1997), 48–63.

[45] C. Palmer, P. Gibbons, and C. Faloustos. 2002. ANF: A fast and scalable tool for data mining in massive graphs. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[46] G. Pang, L. Cao, and C. Aggarwal. 2021. Deep learning for anomaly detection: Challenges, methods, and opportunities. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 1127–1130.

[47] D. Paul, R. Kumar, S. Saha, and J. Mathew. 2021. Multi-objective cuckoo search-based streaming feature selection for multi-label dataset. *ACM Transactions on Knowledge Discovery from Data* 15, 6 (2021), Article No. 93, 1–24.

[48] M. Pawlish, A. Varde, S. Robila, and A. Ranganathan. 2014. A call for energy efficiency in data centers. *SIGMOD Record* 43, 1 (2014), 45–51.

[49] J. Peng, G. Seetharaman, W. Fan, and A. Varde. 2013. Exploiting fisher and fukunaga-koontz transforms in chernoff dimensionality reduction. *ACM Transactions on Knowledge Discovery in Data* 7, 2 (2013), 1–25.

[50] S. Razniewski, N. Tandon, and A. Varde. 2021. Information to Wisdom: Commonsense knowledge extraction and compilation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 1143–1146. Retrieved from https://dl.acm.org/doi/10.1145/3437963.3441664.

[51] J. Rissanen. 1987. Stochastic complexity and the MDL principle. *Econometric Reviews* 6, 1 (1987), 85–102.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Parallel distributed processing: Explorations in the microstructure of cognition, Vol. 1. *Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA. 318–362.

[53] A. Swami, B. Lent, and J. Widom. 1997. Clustering association rules. In *Proceedings of the 13th International Conference on IEEE ICDE*, 220–231.

[54] K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. *ICLR*. arXiv:1409.1556.

[55] R. Sisson, Jr. M. Maniruzzaman, and S. Ma. 2004. Quenching: Understanding, controlling and optimizing the process. In *Proceedings of the International Seminar of the Center for Heat Treating Excellence*. Metal Processing Institute, MA.

[56] T. Squartini and D. Garlaschelli. 2011. Analytical maximum-likelihood method to detect patterns in real networks. *New Journal of Physics* 13, 8 (2011), 083001.

[57] G. Totten, C. Bates, and N. Clinton. 1993. *Handbook of Quenchants and Quenching Technology*. ASM International.

[58] C. Tran, W. Y. Shin, and A. Spitz. 2022. Community detection in partially observable social networks. *ACM Transactions on Knowledge Discovery from Data* 16, 2 (2022), Article No. 22, 1–24.

[59] G. A. Tsihrintzis and C. L. Nikias. 1996. Fast estimation of the parameters of alpha-stable impulsive interference. *IEEE Transactions on Signal Processing* 44, 6 (1996), 1492–1503.

[60] A. S. Varde, S. Ma, M. Maniruzzaman, D. C. Brown, E. A. Rundensteiner, and R. D. Sisson Jr. 2008. Comparing mathematical and heuristic approaches for scientific data analysis. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 22, 1 (2008), 53–69.

[61] A. Varde, M. Maniruzzaman, E. Rundensteiner, and R. Sisson Jr. 2003. The QuenchMiner expert system for quenching and distortion control. In *Proceedings of the 22nd Heat Treating Society Conference and the 2nd International Surface Engineering Congress*. 174–183.

[62] A. Varde, E. Rundensteiner, G. Javidi, E. Sheybani, and J. Liang. 2007. Learning the relative importance of features in image data. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*. 237–244.

[63] A. Varde, E. Rundensteiner, C. Ruiz, D. Brown, M. Maniruzzaman, and R. Sisson Jr. 2006. Integrating clustering and classification for estimating process variables in materials science. In *Proceedings of the 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*.

[64] A. Varde, E. Rundensteiner, C. Ruiz, D. Brown, M. Maniruzzaman, and R. Sisson Jr. 2006. Designing semantics-preserving cluster representatives for scientific input conditions. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management*. 706–717.

[65] A. Varde, E. Rundensteiner, C. Ruiz, M. Maniruzzaman, and R. Sisson Jr. 2005. Learning semantics-preserving distance metrics for clustering graphical data. In *Proceedings of the 6th International Workshop on Multimedia Data Mining*. 107–112.

[66] A. Varde, E. Rundensteiner, C. Ruiz, M. Maniruzzaman, and R. Sisson Jr. 2005. Effectiveness of domain-specific cluster representatives for graphical plots. In *Proceedings of the ACM SIGMOD Conference,on IQIS Workshop*.

[67] A. Varde, E. Rundensteiner, and R. Sisson Jr. 2007. AutoDomainMine: A graphical data mining system for computational estimation. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*. 1103–1105.

[68] Q. C. Vega, C. A. Worby, M. S. Lechner, J. E. Dixon, and G. R. Dressler. 1996. Glial cell line-derived neurotrophic factor activates the receptor tyrosine kinase RET and promotes kidney morphogenesis. *National Academy of Sciences* 93, 20 (1996), 10657–10661.

[69] S. Xiang, F. Nie, and C. Zhang. 2008. Learning a Mahalanobis distance metric for data clustering and classification. *Pattern Recognition Journal* 41, 12 (2008), 3600–3612.

[70] Z. Yang. 2021. *Developing Machine Learning Models to Predict Influence of Heat Treatments on the Tensile Properties of Ti6Al4V Parts Prepared by Selective Laser Melting*. PhD Thesis, WPI, Massachusetts.

[71] Z. Zhou, J. Wu, and W. Tang. 2002. Ensembling neural networks: Many could be better than all. *Artificial Intelligence* 137, 1–2 (2002), 239–263.

Received July 2021; revised November 2021; accepted November 2021