Graphical Data Mining for Computational Estimation in Materials Science Applications

by

Aparna S. Varde

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

in

Computer Science

by

August 15, 2006

APPROVED:

Abstract

In domains such as Materials Science experimental results are often plotted as two-dimensional graphs of a dependent versus an independent variable to aid visual analysis. Performing laboratory experiments with specified input conditions and plotting such graphs 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 needed to obtain a desired graph. State-of-the-art estimation approaches are not found suitable for targeted applications.

In this dissertation, an estimation approach called AutoDomainMine is proposed. In AutoDomainMine, graphs from existing experiments are clustered and decision tree classification is used to learn the conditions characterizing these clusters in order to build a representative pair of input conditions and graph per cluster. This forms knowledge discovered from existing experiments. Given the conditions of a new experiment, the relevant decision tree path is traced to estimate its cluster. The representative graph of that cluster is the estimated graph. Alternatively, given a desired graph, the closest matching representative graph is found. The conditions of the corresponding representative pair are the estimated conditions.

One sub-problem of this dissertation is preserving semantics of graphs during clustering. This is addressed through our proposed technique, LearnMet, for learning domain-specific distance metrics for graphs by iteratively comparing actual and predicted clusters over a training set using a guessed initial metric in any fixed clustering algorithm and refining it until error between actual and predicted clusters is minimal or below a given threshold. Another sub-problem is capturing the relevant details of each cluster through its representative yet conveying concise information. This is addressed by our proposed methodology, DesRept, for designing semantics-preserving cluster representatives by capturing various levels of detail in the cluster taking into account ease of interpretation and information loss based on the interests of targeted users.

The tool developed using AutoDomainMine is rigorously evaluated with real data in the Heat Treating domain that motivated this dissertation. Formal user surveys comparing the estimation with the laboratory experiments indicate that AutoDomainMine provides satisfactory estimation.

Acknowledgments

My heartfelt thanks goes to all those who inspired, supported and encouraged me through my dissertation. First of all, I thank my advisor Prof. Elke Rundensteiner, Professor of Computer Science at WPI, for guiding me with her knowledge and experience during my Masters and Doctorate here. Besides being an outstanding professor, known in the research community worldwide, she came across as an understanding friend with whom I could discuss almost anything. The contributions of my committee members Prof. Carolina Ruiz, Prof. David Brown, and Prof. Neil Heffernan from Computer Science at WPI are also important. Professors Elke and Carolina often spent their weekends working with me to meet paper deadlines while balancing their family duties. Prof. Brown even took the trouble to teleconference from a sabbatical in Germany to attend my comprehensive examination talk. His quick and yet thorough feedback on my research papers was really helpful.

The contributions of the Materials Science [1] domain experts are grate-

[1]This work is supported by the Center for Heat Treating Excellence (CHTE) and its member companies and by the Department of Energy - Industrial Technology Program (DOE-ITP) Award DE-FC-07-01ID14197.

fully acknowledged. I sincerely thank Prof. Richard Sisson Jr., Director of the Materials Science Program at WPI for serving as an external member of my Ph.D. committee. I also thank Dr. Mohammed Maniruzzaman, Research Faculty in Materials Science for his valuable inputs. Prof. Sisson and Dr. Maniruzzaman gave me the opportunity to be a Doctoral Research Assistant in Materials Science while I started work there as a summer intern. This really propelled my dissertation. Both these experts played an active role in critiquing my research. The co-operation of the members of CHTE (Center for Heat Treating Excellence) and of its general body MPI (Metal Processing Institute), is also acknowledged. I especially thank Prof. Diran Apelian, Director of MPI for his support and enthusiasm. His appreciation of my work made it seem really outstanding. I earnestly value his kind, encouraging words and his involvement with my research. I am also grateful to all the Materials Science users who spent their precious time in completing the evaluation surveys of the software tool developed in my dissertation.

I express my gratitude towards the faculty and staff in the Departments of Computer Science, Materials Science and Mechanical Engineering at WPI for their help during this work. I thank Prof. Michael Gennert, Head of the Computer Science Department for his encouragement and help in recommending me for an Instructor position in WPI and supporting some of my conference travel. This gave considerable exposure to my teaching and research. I also thank Prof. Micha Hofri, former Department Head of Computer Science for his support. Another person worthy of mention is Mrs. Rita Shilansky from the Materials Science Administration who treated us students almost like her children.

I value the interesting discussions with members of the Database Systems Research Group, the Artificial Intelligence Research Group and the Knowledge Discovery and Data Mining Research Group in Computer Science, and the Heat Treating research team in Materials Science. In particular, I wish to mention Dr. Shuhui Ma, Dr. Sujoy Chaudhury, Virendra Warke and Olga Karabelchtchikova from Materials Science and Mariana Jbantova, Dr. Maged-El-Sayed, Dr. Bin Liu and Rimma Nehme from Computer Science. In addition, I thank all the other students and professors who attended my research presentations and gave me useful comments.

Moreover, I thank the audience and reviewers of conferences and journals where my papers got published for their reception of my work and their interesting critique. Getting my research recognized through international publications was one of the greatest motivating factors for me in completing this dissertation.

The encouragement provided by my family is truly worthy of mention. My parents Dr. Sharad Varde and Dr. (Mrs.) Varsha Varde, both Doctorates in Statistics from the University of Bombay and both very successful in academia and industry with international acclaim, inspired me to set my goals high ever since my childhood. My enterprising grandfather Mr. D.A. Varde with his authorship of several books as a teacher of Mathematics and Science, and my affectionate grandmother Mrs. Vimal Varde with her words of wisdom, both motivated me during my Ph.D. studies. My brother Ameya Varde, a Masters in Computer Science and Engineering from Purdue University and a holder of many patents in the USA, and my sister-in-law Deepa Varde also a Computer Engineer, encouraged me as well. We are a very closely-knit family and I cannot find enough words to thank everyone at home. Their everlasting love, care and support means a great deal to me.

I really value my friends and colleagues in Computer Science and Materials Science at WPI for providing a healthy research atmosphere and a good camaraderie. Their companionship made the long hours of research seem fun. My house-mates, Lydia Shi and Shimin Li, provided a warm, friendly and yet studious environment at home. In addition, all my other close friends during the various stages of my student life meant a lot to me academically, culturally and personally. I also value the compliments of my extended family members in India and the USA. Their appreciation made me aim even higher during my graduate studies. Sharing the joys of work with family and friends is indeed a great pleasure and is extremely important to me in my career.

Finally, I thank God for guiding me through every path of my doctoral journey. The support of God helped me solve all my problems so far including this dissertation. I have complete faith that the blessings of God will continue to steer me forever.

Contents

List of Figures

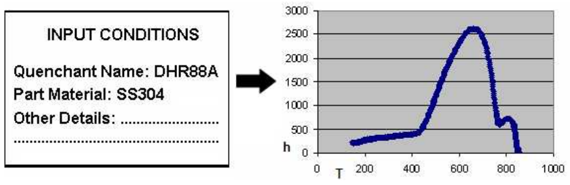
Chapter 1

Introduction

1.1 Motivation

In scientific domains such as Materials Science and Mechanical Engineering experiments are performed in the laboratory with specified input conditions and the results are often plotted as graphs. The term graph in this dissertation refers to a two-dimensional plot of a dependent versus an independent variable depicting the behavior of process parameters. These graphs serve as good visual tools for analysis and comparison of the corresponding processes. Performing a real laboratory experiment and plotting these graphs consumes significant time and resources, motivating the need for computational estimation.

We explain this with an example in the domain of Heat Treating of Materials [M95] that motivated this dissertation. 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 [M95]. Figure 1.1 shows an example of the input conditions and graph in a laboratory experiment in *quenching*, namely, the rapid cooling step in heat treatment.


INPUT CONDITIONS

Quenchant Name: DHR88A
Part Material: SS304
Other Details: ....
....

Figure 1.1: Experimental Input Conditions and Graph

The input conditions shown in this experiment such the Quenchant Name (cooling medium) and Part Material are the details of the experimental setup used in quenching. The result of the experiment is plotted as a graph called a heat transfer coefficient curve. This depicts the heat transfer coefficient h_c versus temperature T. The heat transfer coefficient, a parameter measured in $Watt/meter^2Kelvin$, characterizes the experiment by representing how the material reacts to rapid cooling. Materials scientists are interested in analyzing this graph to assist decision-making about corresponding processes. For instance, for the material *ST4140*, a kind 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. However, performing such an experiment in the laboratory takes approximately 5 hours and the involved resources require a capital investment of thousands of dollars and recurring costs worth hundreds of dollars.

It is thus desirable to computationally estimate the resulting graph given the input conditions. Conversely, given the graph desired as a result, it is also useful to estimate the experimental input conditions that would achieve it. This inspires the development of a technique that performs such an estimation.

1.2 Dissertation Problem: Computational Estimation

The estimation problem we address in this dissertation is explained as follows.

Goals:

- Given the input conditions of an experiment, estimate the resulting graph.
- Given the desired graph in an experiment, estimate input conditions that would obtain it.

These goals are illustrated in Figure 1.2.

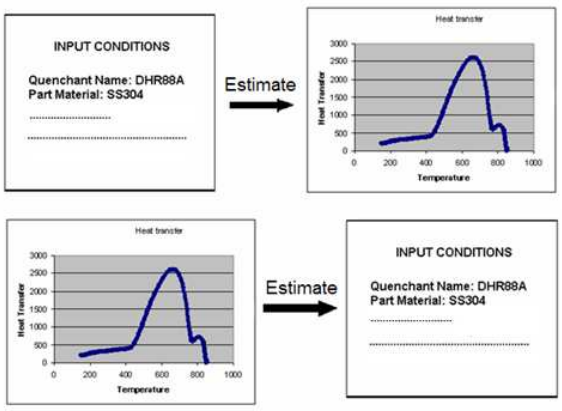


Figure 1.2: Goals of Estimation Technique

The estimation is to be performed under the assumption that the input conditions and graphs of performed experiments in the domain are stored in a database.

Desired Properties of Estimation Technique:

1). Domain expert intervention should not be required each time estimation is performed.

2). The estimation should be such that it is effective for targeted applications in the domain. Thus it should accurately resemble the real domain experiment. Moreover, it should convey as much information to the users as would be conveyed by a real experiment. This effectiveness is to be judged by the users on comparison with laboratory experiments not used for training the technique.

3). The time required for each estimation should be distinctly less than the time required to perform a laboratory experiment in the domain.

With reference to property 3, it is to be noted that computational simulations [LVKR02] of experiments that require as much time as the real laboratory experiment. Nevertheless these simulations are considered useful since they save the cost of resources. The aim of this dissertation however is to save time as well as resources.

1.3 State-of-the-art in Estimation

1.3.1 Similarity Search

Naive Similarity Search

A naive approach to estimation is a similarity search over existing data [HK01]. When the user supplies input conditions of an experiment, these are compared with the conditions stored in the database. The closest match is selected in terms of the number of matching conditions. The corresponding graph is output as the estimated result. However the non-matching condition(s) could be significant in the given domain. For example, in Heat Treating, the user-submitted experimental conditions may match many conditions except the cooling medium used in the experiment and the material being cooled. Since these two factors are significant as evident from basic domain knowledge [TBC93], the resulting estimation would likely be incorrect.

Weighted Similarity Search

A somewhat more sophisticated approach is performing a weighted search [WF00]. Here the search is guided by the knowledge of the domain to some extent. The relative importance of the search criteria, in our context, experimental input conditions, is coded as weights. The closest match is determined using a weighted sum of the conditions. However, these weights are not precisely known, with respect to their impact on the resulting graph or even otherwise. For example, in Heat Treating, in some cases the agitation level in the experiment may be *more* crucial than the oxide layer on the surface of the part. In some cases, it may be *less* crucial. This may depend on factors such as the actual value of the conditions, e.g., high agitation may be more significant than a thin oxide layer, while low agitation may be less significant than a thick oxide layer [BC89]. Thus, there is a need to learn, i.e., to discover knowledge in some manner, for example from the results of experiments.

1.3.2 Case-Based Reasoning

Case-based reasoning (CBR) is an approach that utilizes the specific knowledge of previously experienced problem situations, i.e., cases in order to solve new problems [K93]. A case base is a collection of previously experienced cases. Memories and experiences are not directly mapped to rules, but can be construed as a library of past cases in a given domain [L96]. These existing cases serve as the basis for making future decisions on similar cases using different types of case-based reasoning approaches as described below [AP03].

Exemplar Reasoning

This is a simple reasoning approach that involves reasoning by example, namely, using the most similar past case as a solution to the new case [AP03, SM81]. In other words, this involves finding an existing case from a case base to match a new user-submitted case and using the concepts in the existing case to provide a solution to the new case [AP03]. In the context of our problem this would involve comparing the given input conditions with those of existing experiments, finding the closest match and reasoning that the graph of the closest matching experiment is the estimated result. However, this would face the same problem as in naive similarity searching, namely, the non-matching condition(s) could be significant with respect to the domain.

Instance-Based Reasoning

This approach involves the use of the general knowledge of the domain. This knowledge is stored in the form of instances which can be stored as feature vectors [AP03, M97]. This knowledge is used in addition to existing cases in making decisions about new cases. In our context, using this approach would involve storing the relative importance of the input conditions as feature vectors. In retrieving the closest matching case, this relative importance would be taken into account. However, this would face the problem described in weighted similarity searching, i.e., this relative importance is not known apriori. In this dissertation, one of the issues we address is learning this relative importance.

Case-Based Reasoning with Adaptation

A third and very commonly used approach is the regular Case-based reasoning (CBR) that usually follows the R4 cycle [K93, AP03]. This involves ″R″etrieving an existing case from the case base to match a new case, ″R″e-using the solution for the new case, ″R″evising the retrieved case to suit the new case referred to as Adaptation, and ”R”etaining the modified case to the case-base for further use [K93, AP03]. In adaptation, one manipulates a solution that is not quite right to make it better fit the problem description [K93]. Adaptation may be as simple as substituting one component of a solution for another or as complex as modifying the overall structure of a solution [K93, AV01]. In the literature, adaptation is done using various approaches as discussed below.

Adaptation using Domain-specific Rules. Rule-based approaches are very commonly used for case adaptation. In some systems, the rules may be available apriori from the fundamental knowledge of the domain. This occurs commonly in medical and legal CBR systems, which follow a typical reasoning method employed by the human experts in those domains [K93, PK97]. For example, a doctor may recall that some patient in the past with a certain set of symptoms pertaining to paternal history was diagnosed as diabetic. A new patient may have a similar set of symptoms, the only difference being that these pertain to maternal history. The doctor may then use the fundamental knowledge of the domain to realize that diabetic history is less significant if inherited from the maternal side. Hence in the case of the new patient, there is a relatively less chance of diabetes occurring. A medical CBR system can automate this reasoning for adaptation by coding the fundamental domain knowledge in the form of rules.

Adaptation using Cases. In addition to having a regular case base, some systems such as in [LKW95] build a library of adaptation cases. The adapted case along with the procedure for adaptation is stored in a library of adaptation cases for future use. Thus, when a new case is encountered, the case base is first searched to find the closest matching case. Then the library of adaptation cases is searched to adapt this closest matching retrieved case to the new case. If no adaptation case is found, then adaptation is done from scratch and the corresponding adaptation case is appended to the library. Thus the adaptation is done in an automated manner by using adaptation cases.

Adaptation using Manual Intervention. In some types of CBR systems such as in [DWD97], the system can play an advisory role as opposed to a problem solving role. In such systems, the goal is to guide the user in solving a problem, as opposed to conventional CBR systems, where the goal is to provide an outcome as a suggested solution to the problem. Since an advisory CBR system is only guiding the user, it retrieves an existing case from the case base as an advice, and the user manually adapts this case to solve the given problem. Such systems are not targeted towards naive users. Rather they aim to assist domain experts in providing solutions to problems by retrieving a similar case from the past, thus simulating the role of the memory of the experts in recalling past experience. Since manual intervention is required for adaptation, this approach relies on the knowledge of the domain experts.

Possible application of CBR with Adaptation in this Dissertation. If CBR is to be used as an estimation technique in this dissertation, we first need to define the concept of a case in the given context. Consider a case to be a combination of input conditions and the resulting graph in an experiment.

We first consider rule-based adaptation. If the retrieved case and new case differ by some input condition(s), then domain-specific rules may be applied to compensate for the difference. For example, suppose that in the retrieved case all conditions match except Agitation. Now, from the domain knowledge, we have the rule High Agitation => Fast Cooling. Another rule is Fast Cooling => High Heat Transfer Coefficient. Applying these two rules the system can estimate that the heat transfer coefficients in the new case would be relatively higher than in the retrieved case. However, this is a subjective notion and likely it cannot be enhanced to plot a new graph. It is not known precisely to what extent they would be higher since heat transfer coefficients are a combination of several factors such as part density, quenchant viscosity and so forth. The extent to which agitation impacts the cooling rate differs for different experiments. Note that in the literature on adaptation in CBR, the rule-based adaptation has been used where the case solution is textual, categorical and so forth. To the best of our knowledge, it has not been used for case solutions that involve graphs and pictures.

Consider the approach of using adaptation cases. For example, if a retrieved case is such that it matches the new case, in terms of everything except agitation, then the case base can be searched again to find another case that matches agitation. However this second retrieved case may not match some other parameter such as quenchant temperature. In such a situation, the average of the two graphs corresponding to the two retrieved cases can be suggested as the estimated graph in the new case. However, constructing such an average depends on the relative importance of the input conditions and also the significant aspects of the graphs. Thus it has to be a weighted average, and these weights are not known apriori. Moreover, irrespective of the method used to build a library of adaptation cases, this approach involves significant computation for each new case to be estimated. This may not be efficient.

We may instead consider the advisory role of CBR systems and only output the closest matching retrieved case, leaving the user to do the adaptation. However, this would mean that the system only targets domain experts, not general users. Moreover, the domain experts themselves may not always be able to adapt the solution manually. For example, using the same example above, where only the agitation of the retrieved case and the new case differ, the domain expert will at the most be able to infer that the heat transfer coefficients on the whole should be higher. Plotting the actual curve however would require performing the real experiment.

1.3.3 Mathematical Modeling

Mathematical modeling is the process of deriving relationships between various parameters of interest using numerical equations [PG60, S60]. It can be used in science and engineering domains to perform estimation of some parameters given others. This requires precise representation of the graphs in terms of numerical equations. However, existing models may not be sufficient under certain conditions. This is explained with reference to modeling in the Heat Treating domain.

Modeling in Heat Treating

In Heat Treating, there are analytical expressions that describe the temperature during heating or cooling as a function of time and position within a material. The most difficult variable in such unsteady-state situations is the heat transfer coefficient h_c governing energy transport between the surface of the material and the surroundings [PG60]. As we have stated earlier, the parameter h_c measures the heat extraction capacity of a quenching or rapid cooling process as determined by the characteristics of the part material, the type of cooling medium used and the quenching conditions.

Stolz et. al. [S60] developed a numerical technique for obtaining heat transfer coefficients during quenching from measurements of interior temperatures of a solid sphere. Using this method heat transfer coefficients for quenching oils were evaluated as a function of the surface temperature of the solid. Despite the importance of quenching operations in the heat treatment of alloys, the quantitative aspects of quenching heat transfer, strongly linked to boiling heat transfer could not be accurately estimated. Although boiling is a familiar phenomenon, from the energy transport point of view it is a complicated process [PG60]. There are several variables involved. Heat transfer coefficients depend on various aspects such as density, specific heat, part temperature, quenchant temperature and cooling rate [MMS02]. Cooling rate itself depends on other factors such as quenchant viscosity, agitation and part surface. Since there are many variables involved, and each one in turn may depend on some others, it is difficult to use these models to estimate heat transfer coefficients as a function of temperature, i.e., to predict a heat transfer coefficient curve.

In general, correlations of h_c values applicable to quenching operations have not been satisfactorily obtained due to the complexity of convection systems [PG60]. It is indicated by domain experts that this modeling does not work for multiphase heat transfer with nucleate boiling. Hence this not useful to estimate the required graph, especially in liquid quenching [M95]. Thus we propose heuristic methods to solve the given estimation problem.

1.4 Proposed Estimation Approach: AutoDomainMine

1.4.1 What is AutoDomainMine

In the dissertation have proposed a computational estimation approach called AutoDomainMine which works as follows: the two data mining techniques of clustering and classification are integrated into a learning strategy to discover knowledge from existing experiments. The graphs obtained from existing experiments are first clustered using any suitable clustering algorithm [M67]. Decision tree classification [Q86] is then used to learn the clustering criteria (input conditions characterizing each cluster) in order to build a representative pair of input conditions and graph per cluster. These representatives along with the clustering criteria learned through decision trees form the domain knowledge discovered from existing experiments. The discovered knowledge is used for estimation as follows.

Given a new set of input conditions, the relevant decision tree path is traced to estimate the cluster of the experiment. The representative graph of that cluster is estimated as the resulting graph of the new experiment. Given a desired graph, the closest matching representative graph is found. The corresponding representative conditions are estimated as the input conditions to achieve the desired graph.

An interesting issue in AutoDomainMine involves clustering graphs that are curves since clustering algorithms were originally developed for points. Since a curve is typically composed of thousands of points, a related issue here is dimensionality reduction. Another issue deals conveying an estimate based on approximate match of the decision tree if an exact match is not found. These issues are discussed in Chapter 3.

Besides these, two major challenges forming dissertation sub-problems are as follows:

- Incorporating domain knowledge in clustering through a suitable notion of distance.
- Preserving the semantics of the cluster in building representatives.

These sub-problems are addressed through our proposed techniques LearnMet and DesRept respectively. They are discussed briefly here and elaborated in Chapters 4 and 5 respectively.

1.4.2 Learning Domain-Specific Notion of Distance with LearnMet

In AutoDomainMine, clustering is performed based on the graphs resulting from experiments. An important aspect is thus the notion of similarity in clustering these graphs. Several distance measures have been developed in the literature. However, in our targeted domains, it is not known apriori which particular distance measure works best for clustering, preserving domain semantics. Worst yet, no single metric is considered sufficient to represent various features on the graphs such as the absolute position of points, statistical observations and critical phenomena represented by certain regions. State-of-the-art learning techniques [M97, HK01] are either found inapplicable or not accurate enough in this context. This inspires the development of a technique to learn domain-specific distance metrics for the graphs.

This is addressed through our proposed technique LearnMet [VRRMS0805, VRRMS06]. The input to LearnMet is a training set of actual clusters of graphical plots in the domain. These clusters are provided by experts. LearnMet iteratively compares these actual clusters with those predicted by an arbitrary but fixed clustering algorithm. In the first iteration a guessed distance metric (consisting of a combination of individual metrics representing features on graphs) is used for clustering. This metric is then refined using the error between the predicted and actual clusters until the error is minimal or below a given threshold. The metric corresponding to the lowest error is output as the learned metric.

Challenges in LearnMet involve intelligently guessing the initial metric, defining the notion of error, developing weight adjustment heuristics, developing additional heuristics for selecting suitable data in each iteration (epoch) to increase efficiency of learning as well as the accuracy of the learned metrics and learning metrics that are simple while yet capturing domain knowledge. These challenges are discussed in detail in Chapter 4.

1.4.3 Designing Semantics-Preserving Representatives with DesRept

In AutoDomainMine the clustering criteria are learned by classification in order to semantics-preserving cluster representatives. An arbitrary graph selected as a representative does not always convey all the relevant physical features of the individual plots in the cluster. Similarly any arbitrary set of input conditions selected as a cluster representative may not have certain input condition(s) considered crucial as per the domain. Thus it is important to embody domain knowledge in building the representatives in order to convey more appropriate information to the user.

In this dissertation a methodology called DesRept has been proposed [VRRBMS0606, VRRBMS1106] to design a representative pair of input conditions and graph per cluster incorporating domain knowledge. In DesRept, two design methods, guided selection and construction, are used to build candidate representatives of conditions / graphs capturing different levels of detail in the cluster. Candidates are compared using an encoding proposed in this dissertation analogous to the Minimum Description Length principle [R87]. The criteria in this encoding are ease of interpretation of the representative and information loss due to. Both these criteria take into account the interests of various users. Using this encoding, candidate representatives are compared with each other. The winning candidate, that with the lowest encoding, is output as the designed representative.

Challenges in DesRept involve outlining design strategies for building the candidate representatives, defining a notion of distance to compare the candidates and proposing a suitable encoding based on the given criteria. These are elaborated in Chapter 5.

1.5 System Development and Evaluation

A software tool for computational estimation based on the AutoDomainMine approach has been developed using real data from the Heat Treating domain that motivated this dissertation. The tool is developed in Java, using MySQL for the database and Javascript for the web interface. This is evaluated using data from laboratory experiments in Heat Treating not used for training. The tool is a trademark of the Center for Heat Treating Excellence (CHTE) that supported this research. The development of the tool and its evaluation involve three different stages as described below.

1.5.1 Stage 1: AutoDomainMine Pilot Approach

This includes the learning strategy of integrating clustering and classification. However, it does not include LearnMet and DesRept. This has been developed in order to evaluation is to assess the working of the basic learning strategy. This pilot tool also serves as a criteria for comparison with later versions of the tool. This has been evaluated with domain expert interviews. Data from laboratory experiments not used for training the technique is used for testing. Experts run tests comparing the estimation of AutoDomainMine with the laboratory experiment. If the estimation matches the real data then it is considered to be accurate. Accuracy is reported as the percentage of accurate estimations over all the tests conducted. Some evaluation in this stage is also automated using domain-specific thresholds for comparison between the real and the estimated output. Details of evaluation at this stage appear in Chapter 3. It is observed that the estimation accuracy in this stage is approximately 75%. This is found higher than the accuracy using similarity search which is approximately 65%. It also works better than existing mathematical models as confirmed by experts. However, they indicate that there is scope for further enhancement in AutoDomainMine.

1.5.2 Stage 2: Intermediate Stage of AutoDomainMine with LearnMet

The second stage of the tool includes LearnMet for learning domain-specific distance metrics to cluster graphs. LearnMet has been evaluated with the help of domain expert interviews. Experts provide actual clusters over test sets of graphs distinct from the training set. The distance metrics learned from LearnMet are used to obtain predicted clusters over the test set. These are compared with actual clusters over the test set. The extent to which the predicted and actual clusters match is reported as the clustering accuracy. The details of computing this accuracy are elaborated in Chapter 4. In addition, LearnMet has also been evaluated by integrating it with AutoDomainMine. The most accurate metric learned from LearnMet is used in the clustering step of AutoDomainMine. The rest of the AutoDomainMine strategy stays the same. Evaluation is then conducted similar to the basic AutoDomainMine approach. The results of the evaluation indicate that estimation accuracy in this stage goes up to approximately 87%.

1.5.3 Stage 3: Complete System of AutoDomainMine with DesRept

The third and final stage of the tool includes the DesRept methodology that designs semantics-preserving cluster representatives. DesRept has been evaluated using the proposed Mininum Description Length based encoding with domain experts giving inputs reflecting the user interests in various applications. Different data sets over the real experimental data are used for evaluation. The winning candidate representatives for each data set are determined with respect to the targeted applications. Details of this evaluation are presented in Chapter 5. In addition formal user surveys have been conducted at this stage since it is the complete system. The representatives designed by DesRept are used for estimation in this stage. The estimated output displayed to the users thus involves designed representatives. Users execute tests comparing the estimation with laboratory experiments in a distinct test set. For each test, they convey their feedback in terms of whether the estimation matches the real experiment, i.e., whether the estimation is accurate and if so, which designed representative that best meets their needs. It is found from the surveys that different candidate representatives win in different applications. The estimation accuracy of

AutoDomainMine increases to approximately 93%. This is as per the satisfaction of the targeted users. Details of the user surveys are presented in Chapter 6.

1.6 Dissertation Contributions

This dissertation makes the following contributions. We list each of them along with their significant tasks.

- AutoDomainMine pilot approach
  - Integrating clustering and classification as a learning strategy to discover knowledge for estimation
  - Adapting clustering algorithms to graphs that are curves
  - Finding a suitable method for approximate match in decision tree classification
- LearnMet technique for distance metric learning
  - Intelligently guessing an initial metric
  - Defining a notion of error
  - Developing weight adjustment heuristics
  - Developing additional heuristics to increase efficiency and accuracy
  - Learning simple metrics that capture domain knowledge
- DesRept methodology for designing cluster representatives

  - Defining a notion of distance for the input conditions
  - Developing suitable strategies for design of candidate representatives
  - Proposing a suitable encoding to compare the candidates to select winners in targeted applications

- Development of a computational estimation system, a trademarked tool in Heat Treating
  - Implementing a software tool based on AutoDomainMine using real data in Heat Treating
  - Conducting domain expert interviews to evaluate various stages of system development
  - Evaluating the complete AutoDomainMine system with formal user surveys in the context of targeted applications

1.7 Outline of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 gives an overview of the Heat Treating domain since we will use examples from this domain to explain the concepts in this dissertation. Chapter 3 explains the basic AutoDomainMine approach of integrating clustering and classification. This forms Stage 1 of AutoDomainMine. Chapter 4 gives the details of the LearnMet technique for distance metric learning. This refers to Stage 2 of AutoDomainMine. Chapter 5 elaborates on the DesRept methodology for designing cluster representatives. This corresponds to the 3rd and final stage of AutoDomainMine. System implementation, related work and evaluation of AutoDomainMine at each of its three stages is included in the respective chapters. In addition, a complete system evaluation based on user surveys is presented in Chapter 6. Chapter 7 states the conclusions, including dissertation summary with contributions, and future work.

Chapter 2

Overview of the Heat Treating Domain

2.1 General Background

2.1.1 Materials Science

Materials Science is a field that involves the study of materials such as metals, ceramics, polymers, semiconductors and combinations of materials that are called composites. More specifically, it is the study of the structure and properties of any material. It also encompasses the use of the knowledge of these properties to create new types of materials and to tailor the properties of a material for specific uses [C97].

In recent years, there has been considerable interest in the field of Computational Materials Science [HLHM00, LVKR02, PJFC99]. This involves the use of computational techniques to represent the behavior of physical phenomena [HALLN00]. It includes building mathematical models [PN96] and running simulations of laboratory experiments [FFC00, KR03]. This is helps gain a better understanding of the process parameters. Our research on Data Mining in Materials Science involves building heuristic models and falls under the realm of Computational Materials Science.

2.1.2 Heat Treating of Materials

The domain of focus in this dissertation is the Heat Treating of Materials [M95]. Heat Treating deals with operations involving controlled heating and cooling of a material in the solid state to obtain specific properties [M95]. Quenching is the process of rapid cooling of a material in a liquid and/or gas medium in order to achieve desired mechanical and thermal properties. It forms an important step of the Heat Treating operations in the hardening process [TBC93, BC89]. The setup used for quenching at the Center for Heat Treating Excellence (CHTE) at WPI is shown in Figure 2.1 [MCMMS02]. This is a typical CHTE Quench Probe System.

The CHTE Quench Probe System consists of a notebook-PC-based data acquisition system, pneumatic cylinder with air valve, a small box furnace, a 1 liter beaker for the quenchant (cooling medium) and a K-type thermocouple-connecting rod-coupling interchangeable probe tip assembly The pneumatic cylinder rod moves the probe down into the quench tank from the box furnace. The pneumatic cylinder is connected to the pneumatic valve by 2 tubes [MCMMS02].

Time-temperature data from the thermocouple placed at the center of the probe is acquired using the LabView Data Acquisition Software on a

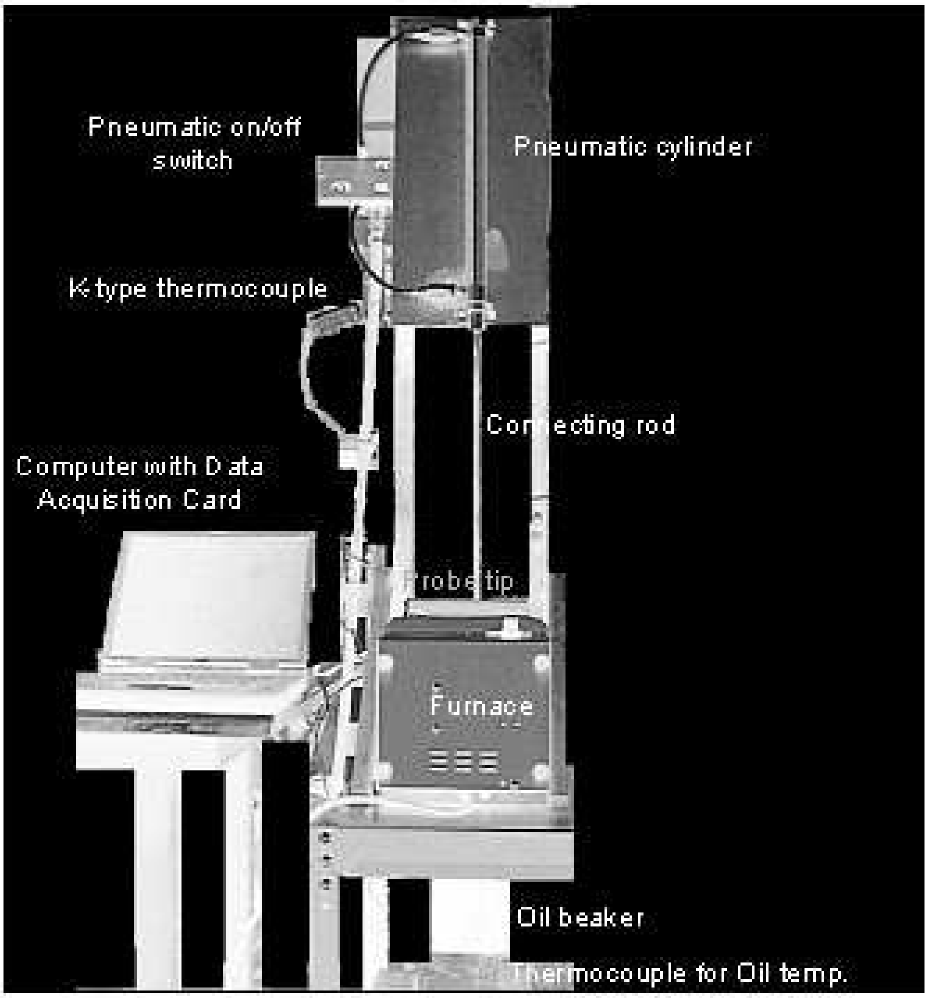


Figure 2.1: CHTE Quenching Setup

notebook computer running Windows 98. The thermocouple is connected to a connector box which is connected to the computer via the PCMCIA DAQCard and a cable. The data analysis and graphing is done using Microsoft Excel and SIGMAPLOT graphing software. The DAQCard is capable of sampling at the rate of 20 kilo samples per second (20 KS/sec) on a single channel with 16 bit resolution [MCMMS02].

Terminology

The material being quenched is referred to in the literature [TBC93] as the part. The part is made of a certain alloy. This has characteristics such as alloy composition and properties based on the microstructure of the alloy, for example the uniformity of the grains. These are identified by the name of the *Part Material* such *ST4140* and *SS304*. In addition the part has properties such as *Oxide Layer*, namely the presence and thickness of oxidation on its surface. A sample of the part called the probe is used for quenching. The probe has properties such as shape and dimension that are identified by the *Probe Type*, e.g., "CHTE Probe" and "IVF Probe" [BC89].

The cooling medium in which the part is placed is known as the *Quenchant*. Quenchants have properties such as the type of the quenchant (e.g., mineral oil, water, bio oil), viscosity, heat capacity and boiling point [HH92]. These properties are characterized by the *Quenchant Name*. Examples of quenchant names include "T7A", "DurixolHR88A".

During the quenching process, the quenchant is maintained at a certain temperature recorded in degrees Celsius. This is referred to as the *Quenchant Temperature*. The quenchant is also subjected to a certain level of agitation such as high, low or absent (no agitation). This is referred to as *Agitation Level*.

The details of the quenchant, part and other factors, such as agitation, form the quenching input conditions. These conditions determine the rate of cooling and consequently the heat transfer coefficients. After quenching, the part acquires desired properties, for example a specific level of hardness [TBC93].

Time and Resources

Performing a quenching experiment takes approximately 5 hours. This includes setting up the apparatus, filling up the tank with a suitable quenchant, making the parts ready for quenching by polishing, heating up the part to a very high temperature and then immersing it into the quenchant for rapid cooling. This is followed by capturing the resulting data by a computer, storing it in a suitable format and plotting graphs that serve as depictions of the results [MCMMS02].

The resources involved can be divided into capital investment and recurring costs. The capital investment includes the CHTE Quench Probe System [MCMMS02]. The furnace, thermocouple, notebook computer and other equipment in this system together costs thousands of dollars. This is a one-time cost. The recurring costs that are incurred each time a laboratory experiment is performed include the probe tip, Data Acquisition (DAQ) cards for the computer, quenchants and of course the human resources to perform the experiment. These costs are on the order of hundreds of dollars per quench test [HH92].

2.2 Graphical Plots in Heat Treating

The results of quenching experiments are plotted graphically. Three important graphs are the cooling curve, the cooling rate curve and the heat transfer coefficient curve. These are described below.

2.2.1 Cooling Curve or Time-Temperature Curve

The cooling curve is a direct plot of the measured part surface temperature T versus time t during the quenching process. This is also referred to as the time-temperature curve [TBC93]. The part temperature is measured in degrees Celsius and time is measured in seconds. The slope of this curve at any given point gives the cooling rate at that point. Figure 2.2 shows an example of a cooling curve.

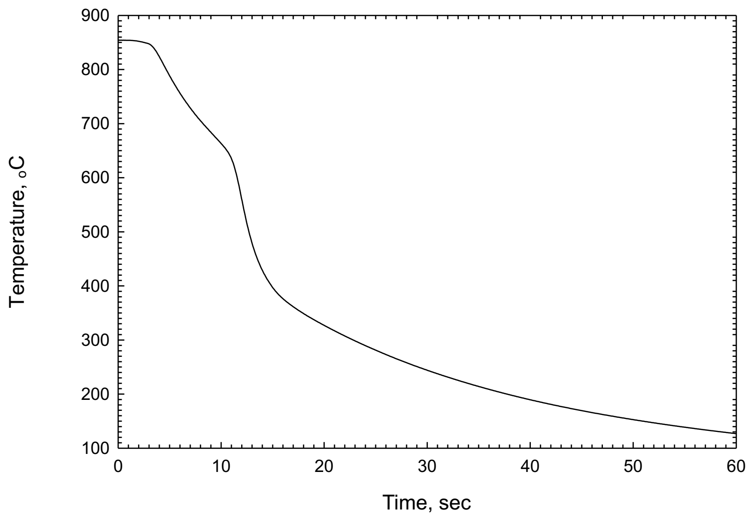


Figure 2.2: Cooling Curve

2.2.2 Cooling Rate Curve

This graph is a plot of part temperature versus cooling rate [TBC93, MCMMS02]. The cooling rates at different points are the derivatives of the part temperature values with respect to the time values denoted as (dT/dt). Thus this curve is a plot of part temperature T versus cooling rate (dT/dt). Part temperature is measured in degrees Celsius while cooling rate is measured in degrees Celsius per second. An example of a cooling rate curve is shown in Figure 2.3. The multiple curved lines seen here represent the cooling rates based on several experiments performed with the same input conditions. Each line represents one experiment. The middle one is the average of the values at the given time over all runs.

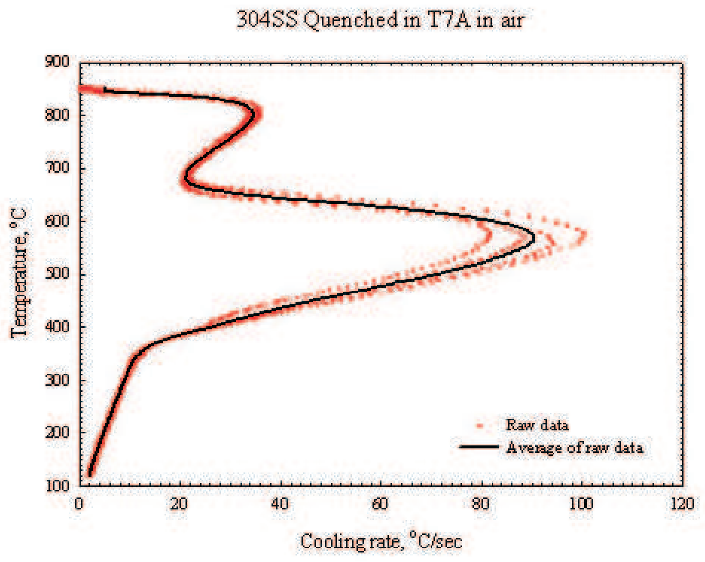


Figure 2.3: Cooling Rate Curve

The cooling rate curve is used to calculate the heat transfer coefficient curve as explained below.

2.2.3 Heat Transfer Curve

Heat Transfer Coefficient

The heat transfer coefficient measures the rate of heat extraction in a quenching process [MCMMS02, MMS02]. It is denoted by h_c. The following equation is used to obtain the heat transfer coefficient h_c [MMS02]

$h_c = \frac{\rho(\frac{V}{A})C_p(\frac{dT}{dt})}{T-T_c}$

where:

h_c = heat transfer coefficient averaged over the surface area measured in Watt per meter square Kelvin

A = surface area of the part in meter square

T = temperature of the part in degrees Celsius

T_c = temperature of the quenchant in degrees Celsius

ρ = density of the part material in kilogram per meter cube

V = volume of the part in meter cube

dT/dt = derivative of temperature with respect to time

Calculating the Heat Transfer Coefficient

The calculation of heat transfer coefficient is critical for characterizing the quenching performance of different quenching media. Generally there are 3 important modes of heat transfer. These are heat conduction, thermal radiation and heat convection. However the thermal resistance to conduction in the solid is small compared to the external resistance and also the probe tip in the CHTE experiments is very tiny. Hence it is assumed that the spatial temperature within a system is uniform. Thus only the convective heat transfer between probe tip and quenching fluid is considered. This approximation is called the lumped thermal capacity model [M95]. This model is valid when the Biot number (B_i) is less than (0.1) where the Biot number is given by the following equation called the Biot Number Equation.

$B_i = hL_c/k$

where:

h = mean heat transfer coefficient

L_c = volume / surface area

k = thermal conductivity

Under the given conditions the heat transfer coefficient is calculated using the Heat Transfer Coefficient Equation given above.

Plotting the Heat Transfer Curve

The plot of heat transfer coefficient h_c versus part temperature T is referred to as a heat transfer coefficient curve. The heat transfer coefficients calculated using the given equation at different points along a cooling rate curve are used to obtain this curve. An example of this curve appears in Figure 2.4. Here also, the middle curve shows the average of the experiments, as in the case of the cooling rate curve. This average curve is used for analysis by scientists. The heat transfer curve is also referred to as a *heat transfer coefficient curve*.

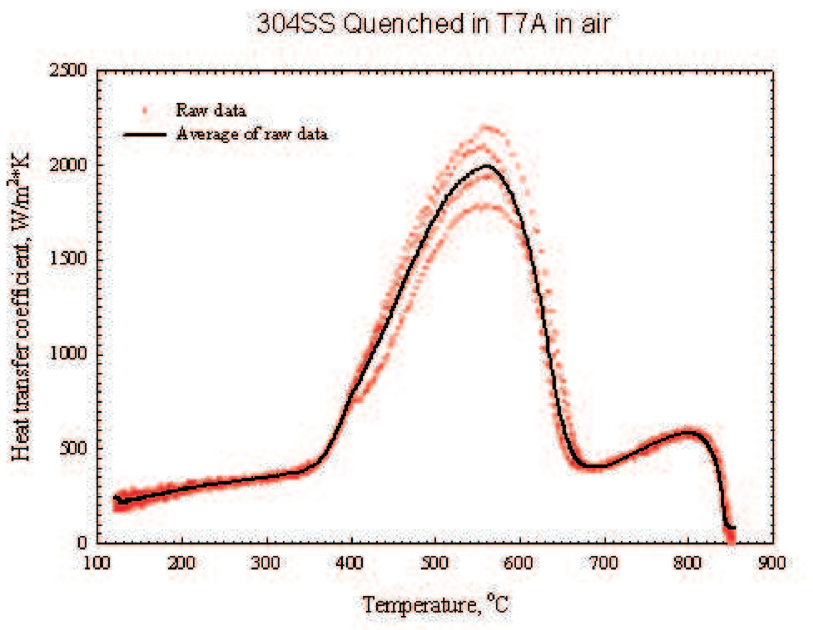


Figure 2.4: Heat Transfer Curve

Importance of Heat Transfer Curves

The heat transfer curve represents the heat extraction capacity in the experiment determined by the combination of the quenchant, part, surface conditions, temperature, agitation and other experimental inputs. The corresponding experimental conditions can be used for quenching in the industry in order to achieve similar results. Among all the graphs, this is of greatest interest to the scientists since it represents the overall heat extraction capacity in the process as determined by a combination of various input conditions. Hence the heat transfer curve is what needs to be estimated given the input conditions of a quenching experiment in order to save the costs of performing the real experiment in the laboratory.

Significant Features of Heat Transfer Curves

There are some points on heat transfer curves that are significant for making comparison. These correspond to certain features on the curve that represent domain-specific physical phenomena [TBC93, BC89]. Since these features help to understand the meaning of the heat transfer curve with respect to the domain they depict the semantics of the curves. These are listed below and illustrated in Figure 2.5.

- BP: The heat transfer coefficient at the boiling point of the quenchant.
- LF: The Leidenfrost point at which the vapor blanket around the part breaks.
- MAX: The point of maximum heat transfer.

- MIN: The point of minimum heat transfer.
- SC: The point where slow cooling ends.

The Boiling Point BP marks the beginning of the convection phase at which the temperature of the part being cooled is reduced to the boiling point of the cooling medium and slow cooling begins [BC89]. The Leidenfrost Point LF denotes the breaking of a vapor blanket resulting in the beginning of rapid cooling in the partial film boiling phase. Thus a curve with and without a Leidenfrost point denotes two different physical tendencies in quenching [BC89].

Also significant is the range of maximum heat transfer MAX achieved in a quenching process. This serves to separate the curves, and hence the corresponding experiments, statistically into different categories. Likewise the mean heat transfer achieved in the process is also a statistical distinguishing factor [BC89, MCMMS02].

Other important points on the curve are MIN, the point of minimum heat transfer, and SC, the point where slow cooling ends. This summarizes the semantics associated with heat transfer curves.

Experimental data about the conditions of the quenching setup and details of the quenchants and parts forms the input conditions of the quenching experiments. For each experiment its input conditions and its resulting graph, i.e., the heat transfer curve are stored in a database. This data is used as the basis for computational estimation in AutoDomainMine.

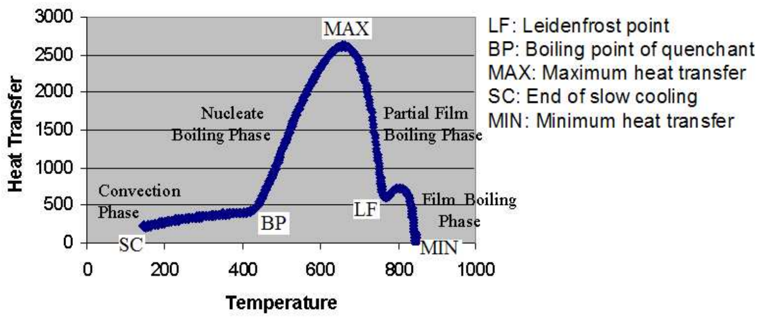


Figure 2.5: Heat Transfer Curve with its Semantics

Chapter 3

AutoDomainMine: Integrating Clustering and Classification for Computational Estimation

3.1 Steps of AutoDomainMine

The proposed computational estimation approach called AutoDomainMine involves a one-time process of knowledge discovery from existing data and a recurrent process of using the discovered knowledge for estimation. These two processes are illustrated along with their steps in Figure 3.1.

AutoDomainMine discovers knowledge from experimental results by integrating clustering and classification, and then uses this knowledge to estimate graphs given input conditions or vice versa. The two data mining techniques are integrated for knowledge discovery in AutoDomainMine as

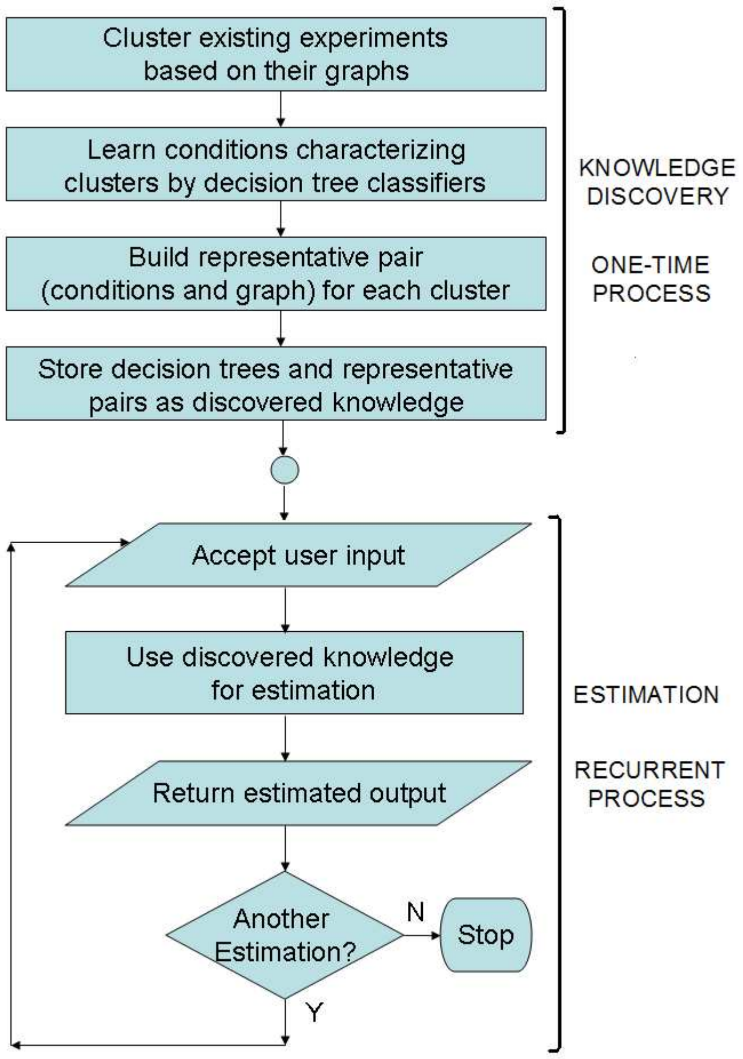

Cluster existing experiments based on their graphs
Learn conditions characterizing clusters by decision tree classifiers
Build representative pair (conditions and graph) for each cluster
Store decision trees and representative pairs as discovered knowledge
KNOWLEDGE DISCOVERY
ONE-TIME PROCESS
Accept user input
Use discovered knowledge for estimation
Return estimated output
Another Estimation?
N
Stop
Y
ESTIMATION
RECURRENT PROCESS

Figure 3.1: The AutoDomainMine Approach

explained below.

3.1.1 Knowledge Discovery in AutoDomainMine

The process of knowledge discovery is depicted in Figure 3.2. Clustering is first done over the graphical results of existing experiments that have been stored in the database. Since clustering techniques were originally developed for points [KR94], a mapping is proposed that converts a 2-dimensional graph into an n-dimensional point. A suitable notion of distance for clustering graphs is defined based on the knowledge of the domain [1]. Once the clusters of experiments are identified by grouping their graphs, the clustering criteria, i.e., the 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 for each cluster. The decision trees and representative pairs form the discovered knowledge. This knowledge is used for estimation as follows.

3.1.2 Estimation in AutoDomainMine

The process of estimation is shown in Figure 3.3. There are two processes here, estimating the graph given the conditions and estimating the conditions given the graph. These are explained as follows.

[1]In the pilot stage of AutoDomainMine the notion of distance between the graphs is based on Euclidean distance taking into account significant features of graphs as identified by experts. This is elaborated in the section on clustering. This is refined in later stages.

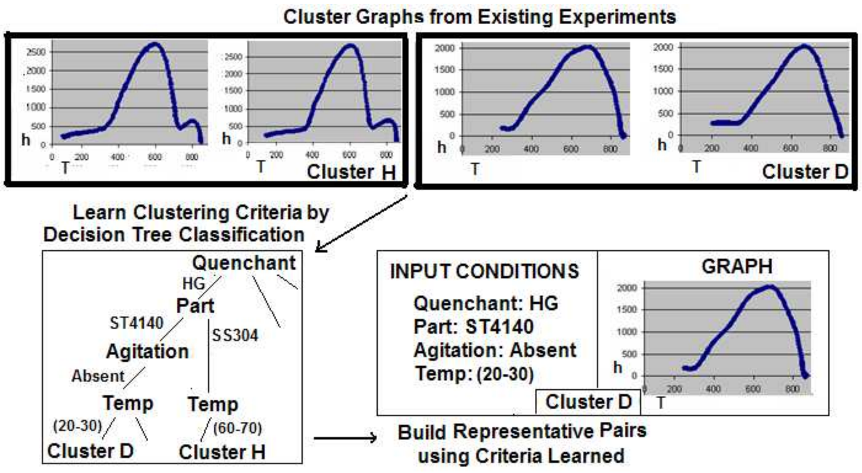


Figure 3.2: Discovering Knowledge from Experiments

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 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 given notion of distance for the graphs. The representative conditions corresponding to the match are the estimated input conditions that would obtain the desired graph. Note that this estimation takes into account the relative importance of the conditions as identified from the decision tree.

3.2 Related Work

In AutoDomainMine, two data mining techniques, clustering and classification are integrated into a learning strategy to discover knowledge for es-

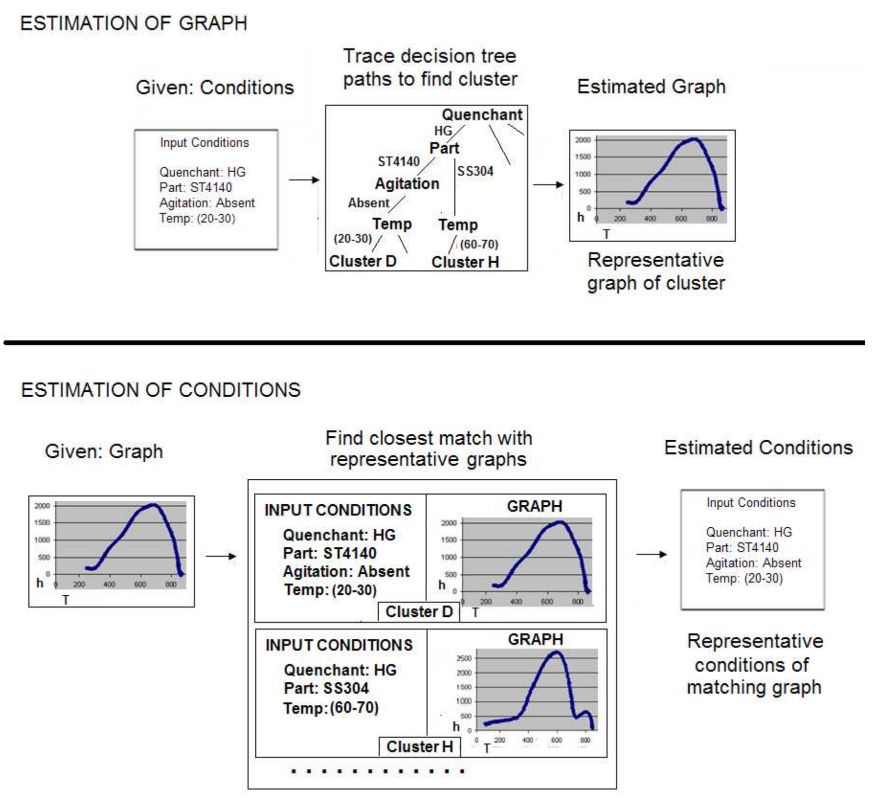


Figure 3.3: Using Discovered Knowledge for Estimation

timation. In the literature, integration of data mining techniques has been performed in the context of given problems. We briefly overview a few that are relevant to this dissertation.

3.2.1 Rule-Based and Case-Based Approaches

Rule-based and case-based approaches have been integrated in the literature to solve certain domain-specific problems [L96]. General domain knowledge is coded in the form of rules, while case-specific knowledge is stored in a case base and retrieved as necessary. For example, in the domain of law [PK97], 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 the rules relevant to the new case and also retrieves similar cases in the past to learn from experience. It has been observed that these two approaches combined derive a more accurate solution to the new case, than either approach individually [PK97, L96].

However, in the literature this approach has been used for cases that involve text-based documents which is common in the legal domain [PK97, L96]. For example, the solution for a past offense that involved an adult can be modified based on rules in the constitution if the offender in a new case is a minor. However, it is non-trivial to apply this approach to graphs 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 (estimation) for the new one. Moreover, in our targeted domains, we do not have a fixed set of rules available analogous to the constitution.

3.2.2 Classification and Association Rule Mining

Liu et. al. [LHM98] propose a framework called associative classification that combines the approaches of classification and association rule mining [LHM98]. Classification aims to discover rules in the database that forms an accurate classifier. Association rule mining finds all the rules existing in the database that satisfy some minimum support and minimum confidence constraints. The proposed framework focuses on mining a special subset of association rules called Class Association Rules (CARs) [LHM98]. The ultimate goal here is to classify a target. However, data that needs to be classified is likely to have a large number of associations. Using classification techniques alone would not mine these associations, leaving some useful rules undiscovered. Thus, it is feasible first discover associations and then use these for classification. Adaptation of existing association rule mining algorithms [AIS93] has been done to mine only the CARs is needed so as to reduce the number of rules generated, thus avoiding combinatorial explosion [LHM98]. The proposed integration also involves discretizing the continuous attributes based on the classification predetermined class target. They then build a classifier based on the generated CARs.

However the data in [LHM98] is of a quantitative and categorical nature. Hence it is meaningful to derive association rules between the attributes. We have used association rules in our earlier system QuenchMiner^{TM} that predicted only ranges of parameters. It was thus feasible to apply association rules to estimate ranges in terms of values such as high, medium and low. However, when graphs are involved, the issues are different. Consider for example a rule derived from QuenchMiner^{TM} such as *High Agitation* $=>$ *Fast Cooling*. This rule is not sufficient to estimate a graph though it is sufficient to estimate ranges of cooling rates. Any further discussion on this would go beyond the state-of-the-art and is not addressed here.

3.2.3 Association Rules and Clustering

In [LSW97] an Association Rule Clustering System (ARCS) is described. The ARCS system clusters 2-dimensional association rules in large databases. More specifically, they consider the problem of clustering association rules of the form $A \wedge B => X$ where the Left Hand Side (LHS) attributes A and B are quantitative and the Right Hand Side (RHS) attribute X is categorical. For example, the rules $(age = 40) => (ownHome = yes)$ and $(age = 41) => (ownHome = yes)$ are clustered as $(40 <= age < 42) => (ownHome = yes)$. They define a term called segmentation as the collection of all the clustered association rules for a specific value X of the RHS. Their goal is to find the fewest number of clusters that cover association rules in a segmentation.

However, their constraints are that the left hand side is numeric and the right hand side is categorical. Drawing an analogy with our problem, the left hand side consists of mixture of attributes that are numeric, categorical and ordinal [HK01] while the right hand side consists of graphs. Hence the constraints in their problem are not satisfied in our dissertation problem. Thus a direct application of their ideas is not applicable to us. Moreover, association rules are also not sufficient to solve the estimation problem for reasons discussed above.

3.2.4 Clustering to aid Classification

In [BL04] the Minimum Description Length (MDL) principle [R87] is applied to evaluate how a clustering algorithm run on data aids a classification algorithm. Their goal is to produce a set of clusters that best agree with a certain set of hidden labels forming classification targets. 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, such that $L > 1$. If c is the number of clusters, r is the number of random initializations of the clustering algorithm from which the best initialization is chosen 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, as per the goal in [BL04].

We can draw an analogy here in terms of using both clustering and classification in the approach. However, in [BL04] they evaluate clustering in the context of classification. They do not integrate clustering and classification into a learning strategy for knowledge discovery. Classification is not actually executed in this approach. Rather their goal is to evaluate clustering and produce the best set of clusters with the possible intention of classification. In our problem, we execute both clustering and classification for discovering knowledge from experimental results in order to use the discovered knowledge for estimation.

3.2.5 Learning Methods of Materials Scientists

From a detailed study of the relevant literature in Heat Treating and discussions with domain experts it has been noticed that Materials Scientists often use the following learning methods to discover certain facts from experimental results. They group experiments based on the similarity of their results and then reason the causes of similarity between the groups based on the input conditions of the experiments [SMMV04].

For example, it was learned experimentally that a thin oxide layer on the surface of a part causes fast cooling while a thick oxide layer causes slow cooling [SMMV04]. This was learned by conducting experiments with thin and thick oxide layers among the input conditions with other conditions being the same. The results showed that for all the 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 resulting in fast cooling while thick oxide acted as an insulator resulting in slower cooling. Thus, the learning was done by grouping experiments based on similarity of their results and reasoning based on their corresponding input conditions [SMMV04].

This grouping and reasoning is analogous to the data mining techniques of clustering and classification respectively. We thus automate these learning methods of scientists by integrating clustering and classification as a learning strategy for knowledge discovery in AutoDomainMine.

3.3 Details of Clustering

Clustering groups objects into classes such that objects within a class have high similarity but are different from items in other classes [KR94]. In AutoDomainMine, we can use any clustering algorithm such as k-means [M67]. In applying clustering there are three main issues that we need to deal with as follows.

- Applying clustering algorithms to graphs that are curves since these algorithms were originally developed for points.
- Addressing issues of dimensionality reduction since these graphs are composed of thousands of points.
- Defining a notion of similarity for the graphs.

These issues are explained in the following subsections.

3.3.1 Applying Clustering to Curves

In order to apply clustering algorithms [KR94] to curves, we propose a mapping that converts a 2-dimensional curve to an multi-dimensional point as follows. 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 *lengths* along these n dimensions respectively.

Mathematically this can be represented as $(x, y) \mapsto (dimension, length)$ where $\mapsto$ denotes "maps to". This mapping is shown in the Figure 3.4. In this figure $x_1 ... x_n$ indicate the n dimensions. For example, a point $(200, 400)$

on the original curve is mapped to length 400 on the x^{th}_{200} dimension. Likewise each point on the curve is mapped to a dimension and a length along that dimension. Thus after mapping each (x, y) point on the 2-dimensional curve to a *(dimension,length)* pair we effectively get an n-dimensional point. Note that in this figure the axes on the right hand side represent the dimensions, *not* the lengths along these dimensions.

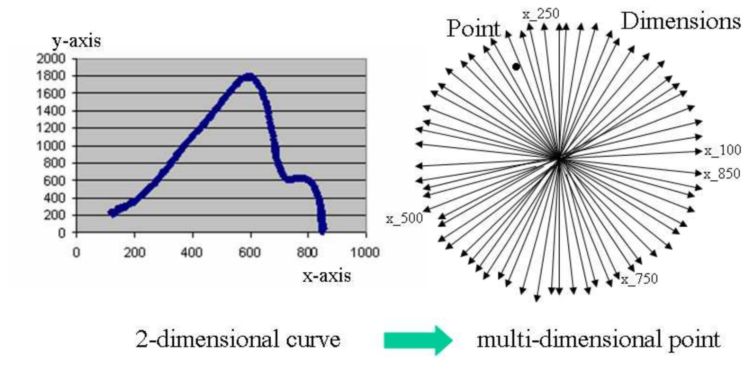


Figure 3.4: Mapping a 2-dimensional curve to an n-dimensional point

3.3.2 Dimensionality Reduction

Since a curve has typically thousands of points it may be inefficient to convert each x-coordinate into a separate dimension. Hence dimensionality reduction is often used. There are various methods of dimensionality reduction among which Sampling, [HK01] and Fourier Transforms [F55] are relevant to our problem. We briefly describe these below.

Selective Sampling

This method has some similarities with Random Sampling [HK01]. In Random Sampling, points are sampled at random intervals. In Selective Sampling [HK01] points are chosen based on certain criteria. In our problem, sample the points at regular intervals, and in addition sample critical points that correspond to significant features of the graph. Knowledge of the domain gathered from literature surveys and discussions with experts helps in determining the significant features.

This method is shown in Figure 3.5 where $x_{critical_i}$ indicates each critical dimension, corresponding to a critical point (x_i, y_i) on the original curve. In this figure, the axes on the right hand side represent the lengths along the dimensions drawn approximately to scale.

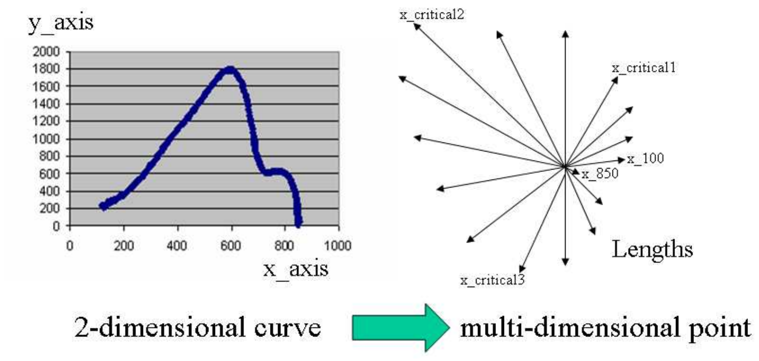


Figure 3.5: Selective Sampling for Dimensionality Reduction

Fourier Transform

A Fourier Transform decomposes a given curve into sinusoids of different frequencies such that they sum back to the original waveform [F55]. 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 can be used as follows. We first apply Equation 1 [AFS93], in order to map the n dimensions of the curve (n-dimensional point) into n Fourier Coefficients. Each Fourier Coefficient corresponds to a different frequency. In this equation F(s) refers to the frequency domain while f(t) refers to the time domain.

$$F(s) = (1/\sqrt{N})\Sigma_{t=0}^{N-1} e^{-j2\Pi f(t)/N} \tag{3.1}$$

We then retain the Fourier Coefficients that are considered useful with respect to the domain. As in Selective Sampling, domain knowledge obtained from literature surveys and discussions with experts is useful here.

For example in Heat Treating, generally the first 16 coefficients are considered to be useful. This is because the graphs (heat transfer curves) [M95] are such that these coefficients representing lower frequency values contain useful information. Our experimental evaluation confirms this as will be shown later. The remaining coefficients representing higher frequency values are regarded as noise [B68, TBC93].

3.3.3 Notion of Distance

The default notion of distance in clustering algorithms is often Euclidean. In the pilot AutoDomainMine approach this is used as the fundamental notion of distance between graphs. However, domain experts have a subjective notion that the 3 points on a heat transfer curve MAX, LF and

BP are more significant than others. These are respectively the point of maximum heat transfer coefficient, the Leidenfrost point at which rapid cooling begins, and the heat transfer corresponding to the Boiling Point of the Quenchant [TBC93]. Thus, in addition to the original curve, these regions are considered as 3 more dimensions in the representation of the 2-dimensional curve as an multi-dimensional point. Thus, the number of dimensions in the multi-dimensional point is $n + 3$ where n refers to the n x-coordinates while the additional 3 dimensions correspond to the 3 points MAX, LF and BP respectively.

3.3.4 Steps of Clustering

Clustering is done in AutoDomainMine by sending either the original curves ($n+3$-dimensional point) or their Selective Samples or Fourier Transforms to a clustering technique such as k-means. Once the clusters are obtained for each graph (which represents each experiment), the output of the clustering needs to be sent to the classifier. Therefore each experiment is stored in terms of its input conditions and cluster label. These steps are listed below.

Clustering in AutoDomainMine

1). Map each 2-dimensional graph into a multi-dimensional point

2). Perform dimensionality reduction as needed

3). Define notion of distance for graphs

4). Send each graph as multi-dimensional point to clustering technique such as k-means

5). Use the given notion of distance to cluster the graphs

6). Store the input conditions and cluster corresponding to each graph

3.4 Details of Classification

It is important to know the causes of similarities and differences between the experiments. This helps us to understand the reasoning behind the clustering, i.e., the clustering criteria. In other words, it helps to determine the relationships between the clusters of graphs and the corresponding input conditions. The method proposed in AutoDomainMine for determining the clustering criteria is classification using decision trees.

3.4.1 Decision Trees as Classifiers

A decision tree [KK95] 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. The reasons for selecting decision tree as classifiers in AutoDomainMine are:

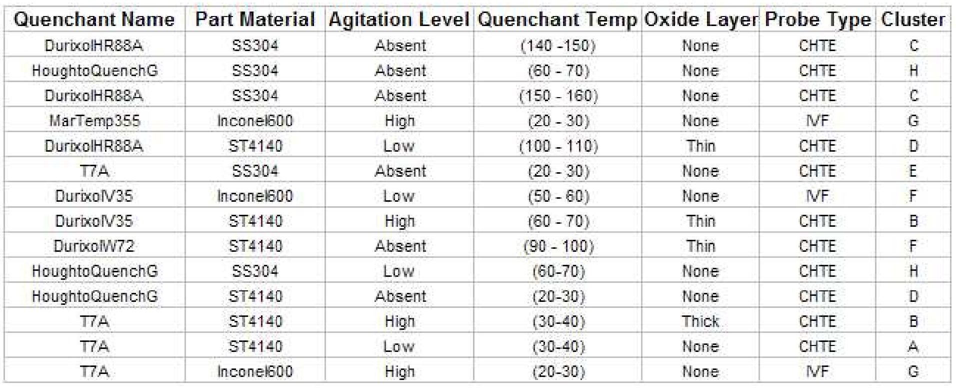
- It helps to identify the relative importance of the criteria used in classification. Thus in our context, this is useful in determining the relative importance of the input conditions leading to the clusters.

- It is an eager learning approach, i.e., it learns based on existing data. This is useful to us because our knowledge discovery step is a one-time process executed in advance over existing data.

- It provides reasons for its decisions. Thus in our context its structure is well-suited to provide partial matches if a complete set of input conditions does not match.

Decision trees in AutoDomainMine are constructed using the J4.8 algorithm [Q86]. J4.8 generates a decision tree for the given data by recursively splitting that data. The decision tree grows using a depth-first strategy. The J4.8 algorithm considers all the possible tests that can split the data and selects a test that gives the best information gain [Q86]. For each discrete attribute, one test is used to produce as many outcomes as the number of distinct values of the attribute. For each continuous attribute, the data is sorted, and the entropy gain is calculated based on binary cuts on each distinct value in one scan of the sorted data. This process is repeated for all continuous attributes. The J4.8 algorithm allows pruning of the resulting decision trees. The J4.8 algorithm can also deal with numeric attributes. This is useful in AutoDomainMine, because the input to the decision tree classifier is the output of the clustering step, which includes input conditions that may be numeric values.

Figure 3.6 shows a sample partial input to the classifier. This is obtained from the output of the clustering step in AutoDomainMine using Heat Treating data. It depicts the input conditions of the experiments and the cluster in which the corresponding graph was placed.



Quenchant Name	Part Material	Agitation Level	Quenchant Temp	Oxide Layer	Probe Type	Cluster
DurixolHR88A	SS304	Absent	(140 -150)	None	CHTE	C
HoughtoQuenchG	SS304	Absent	(60 - 70)	None	CHTE	H
DurixolHR88A	SS304	Absent	(150 - 160)	None	CHTE	C
MarTemp355	Inconel600	High	(20 - 30)	None	IVF	G
DurixolHR88A	ST4140	Low	(100 - 110)	Thin	CHTE	D
T7A	SS304	Absent	(20 - 30)	None	CHTE	E
DurixolV35	Inconel600	Low	(50 - 60)	None	IVF	F
DurixolV35	ST4140	High	(60 - 70)	Thin	CHTE	B
DurixolW72	ST4140	Absent	(90 - 100)	Thin	CHTE	F
HoughtoQuenchG	SS304	Low	(60-70)	None	CHTE	H
HoughtoQuenchG	ST4140	Absent	(20-30)	None	CHTE	D
T7A	ST4140	High	(30-40)	Thick	CHTE	B
T7A	ST4140	Low	(30-40)	None	CHTE	A
T7A	Inconel600	High	(20-30)	None	IVF	G

Figure 3.6: Sample Partial Input to Classifier

Figure 3.7 shows a snapshot of a partial decision tree created for this data. The sets of input conditions that lead to each cluster have been identified in this tree. For example, with reference to the given partial input and partial decision tree, it is clear that Cluster D consists of experiments with quenchant ”HoughtoQuenchG”, part material, ”ST4140”, agitation, ”Absent”, and quenchant temperature in the range of ”(90-140)”. Cluster F has all the same conditions, except the quenchant temperature. From this decision tree it can be inferred that a crucial clustering criterion is the quenchant name, since that forms the root of the tree. However, criteria such as quenchant temperature are also important since a difference of temperature range causes the experiments to be placed in different clusters. Thus this tree helps to reason the causes of similarities and differences between experiments with respect to their input conditions.

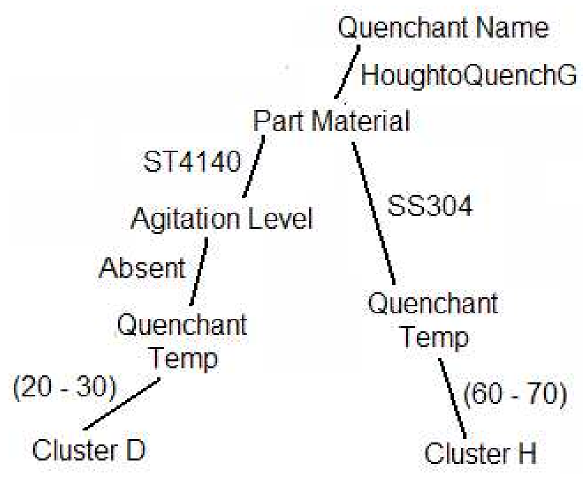


Figure 3.7: Snapshot of Partial Decision Tree

3.4.2 Selecting Representative Pairs

The decision trees form the basis for selecting a representative pair of input conditions and graph per cluster. In the pilot AutoDomainMine approach the process of selecting representative pairs is as follows.

Selecting Representatives for Classification in AutoDomainMine

1). Trace the paths from the root to each leaf of the decision tree.

2). Consider each path as a set of input conditions.

3). Treat the leaf of each path as the cluster for that set of conditions.

4). Among all graphs in that cluster select *any one* as a *representative graph*.

5). Among all the paths (sets of conditions) leading to a particular cluster, select *any one* path as *representative conditions*.

6). Store the *representative conditions* and *representative graph* for each cluster as its *representative pair*.

A sample representative pair is illustrated in Figure 3.8 with reference to the partial decision tree in Figure 3.7.

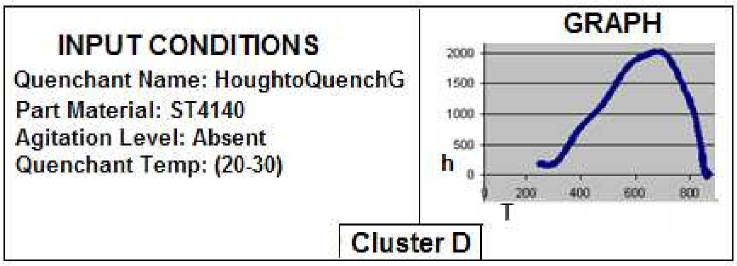


Figure 3.8: Sample Representative Pair for Cluster D

These representative pairs are used for classifying a user-submitted experiment, serving as the basis for estimation.

3.5 Estimation in AutoDomainMine

AutoDomainMine estimates the graph obtained in an experiment, given the input conditions and vice versa. This is explained in the two subsections below, with reference to Heat Treating domain.

3.5.1 Estimating the Graph

The user enters the input conditions of the new experiment, and requests AutoDomainMine to estimate the graph that would be obtained in a new experiment. AutoDomainMine traces the decision tree to find the appropriate path leading to the cluster in which the new experiment should be placed. The representative graph of that cluster is output as the estimated graph in the user-submitted experiment.

Since the relative importance of the input conditions has already been learned through the decision tree, this helps to make an educated guess about the closest matching cluster for the user-submitted input conditions. Hence, if the 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 from the lower level onwards no complete match is found, then it is considered acceptable to give an estimate based on a partial match. The distinction between high and low levels is made depending on the height of the tree. The levels at or above half the depth of the tree are considered as high and those below are half the depth as low. This is an intuitive approximation used in the pilot stage of AutoDomainMine [2].

The concept of selecting the cluster with the greatest number of experiments in case of a partial match is analogous to the concept of majority class in classifiers [M97].

The process of estimating the graph is as follows.

[2]In later stages of AutoDomainMine this approximation is justified by learning a heuristic for distances between sets of conditions in the decision tree paths

Estimation of Graph in AutoDomainMine

1). Accept new input conditions from the user.

2). Compare each path of decision tree of with new input conditions.

3). If one or more partial paths match upto less than or equal to half the height of the tree, convey that the graph cannot be estimated and Quit.

4). If one or more partial paths match upto greater than half the height of the tree, among all clusters emerging from the partial path(s), the representative graph of the cluster with the greatest number of experiments is the estimated graph, go to Step 6.

5). If a complete path matches upto the leaf node, then the representative graph of that cluster is the estimated graph.

6). Display the estimated graph to the user.

This estimation process is illustrated in Example 1.

Example 1

Estimate the heat transfer curve in the following experiment, given its quenching conditions.

- Quenchant Name: ″HoughtoQuenchG″
- Quenchant Temperature: ″25″
- Agitation Level: ″Absent″

- Part Material: ”ST4140”
- Oxide Layer: ”Thin”
- Probe Type: ”CHTE”

Analysis. The relevant path of the decision tree traced for the above set of conditions is shown in Figure 3.9.

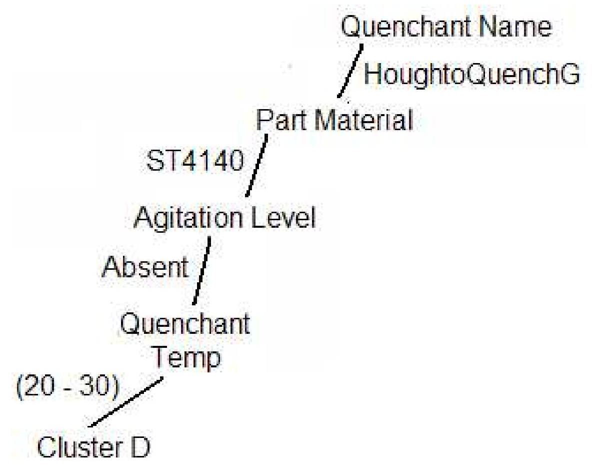


Figure 3.9: Relevant Partial Decision Tree

Estimated Cluster. From the above tree, the estimated cluster of the new experiment is D.

Representative Graph. The representative graph of the estimated cluster is shown in Figure 3.10

Output. The graph shown in Figure 3.11 is output as the estimated heat transfer coefficient curve for the user-submitted experiment. (This is the

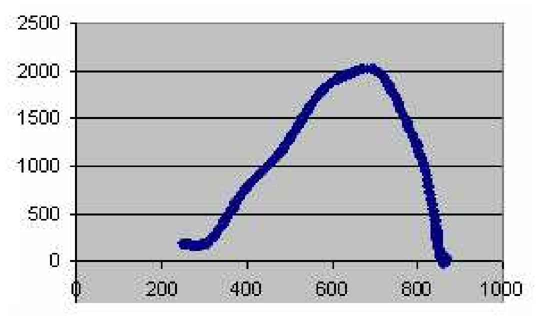


Figure 3.10: Representative Graph of Estimated Cluster

same as the representative graph in Figure 3.10).

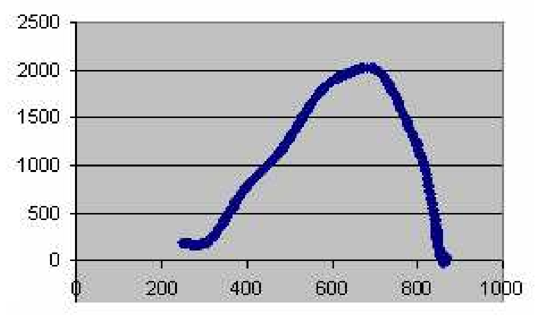


Figure 3.11: Estimated Heat Transfer Curve

3.5.2 Estimating the Conditions

The user enters a sample graph as a desired result and requests AutoDomainMine to estimate the required experimental conditions that would achieve this. AutoDomainMine compares this graph with the representative graphs in the clusters using the given notion of distance. AutoDomainMine selects the graph with the closest match, and outputs the representative conditions of that cluster, as the required experimental conditions to achieve this graph. We define a similarity threshold for graphs in the domain. In Heat Treating this threshold is 10%.

If no match is found within the given threshold, then it implies that the desired graph cannot be obtained based on the knowledge discovered from existing experimental data. Thus it is conveyed to the user that the conditions to obtain this graph cannot be estimated. If only one representative graph matches the desired graph within the threshold, then it is obvious that the corresponding representative conditions are the estimated conditions. If several representative graphs match, it is desirable to select the closest match. However, if two or more graphs match to the same extent then we approximate based on considering the representative conditions of the majority class. This is done analogous to classifiers in the literature [M97].

The process of estimating the conditions is as follows.

Estimation of Conditions in AutoDomainMine

1). Accept desired graph from the user.

2). Compare desired graph with all the representative graphs.

3). If no match is found within the given threshold then convey that the conditions cannot be estimated and Quit.

4). If only one graph matches within threshold, then representative conditions of that graph are the estimated conditions, go to Step 7.

5). If more than one graph matches within the threshold, then representative conditions of closest matching graph are the estimated conditions.

6). If two or more graphs match to the same extent, then the representative conditions corresponding to the graph with greater number of experiments in the corresponding cluster are the estimated conditions.

7). Display the estimated conditions to the user.

This estimation process is illustrated in Example 2.

Example 2.

Estimate the quenching conditions required to obtain the heat transfer curve shown in Figure 3.12.

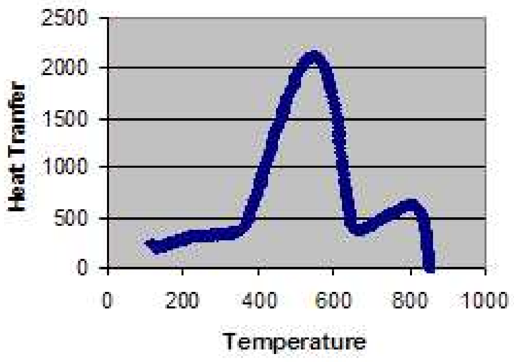


Figure 3.12: Desired Graph

Matching Graph. The closest matching graph to the desired graph is shown in Figure 3.13.

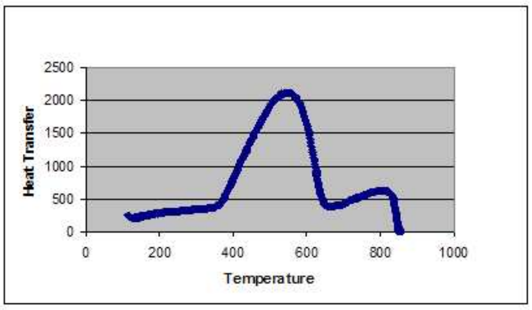


Figure 3.13: Matching Graph

Estimated Conditions. The representative conditions of the cluster of the matching graph are listed below.

- Quenchant Name: ”T7A”
- Agitation Level: ”Absent”
- Quenchant Temperature: ”(20-30)”
- Part Material: ”SS304”
- Oxide Layer: ”None”
- Probe Type: ”CHTE”

Output. The estimated conditions above are output as the required quenching conditions to obtain the desired heat transfer curve.

3.6 Evaluation of AutoDomainMine Stage 1: Pilot Tool

3.6.1 Implementation of Pilot Tool

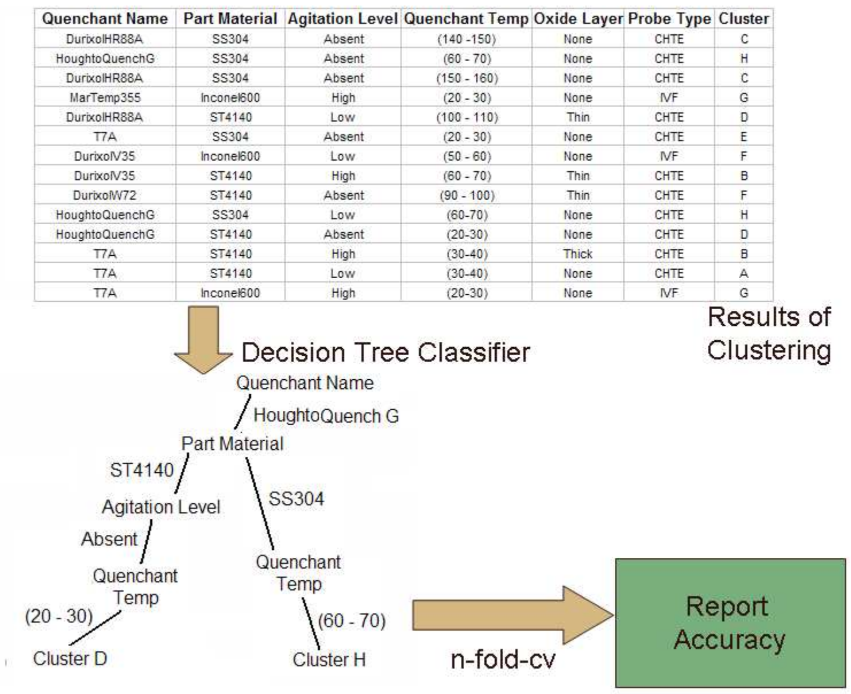
A pilot tool has been implemented with the basic learning strategy in AutoDomainMine, i.e., clustering followed by classification with the details as explained in this chapter.

The implementation of AutoDomainMine has been done in Java. The database has been built using MySQL [T02]. The tool developed using the AutoDomainMine approach is a property of the Center for Heat Treating Excellence, WPI. Existing tools such as the WEKA system [FHKH02, WF00] have been used to provide some of the basic functionalities required in the approach. These include the k-means algorithm for clustering [M67] and J4.8 decision trees [Q86] for building classifiers.

The pilot tool in AutoDomainMine has been evaluated with real data from the Heat Treating domain. The evaluation has been done using two methods, namely, 4-fold cross-validation, and domain expert interviews. The details of all the evaluation are described below.

3.6.2 Evaluation with Cross Validation

In this evaluation process, the AutoDomainMine approach of clustering followed by classification has been executed over different data set sizes. For each data set, the accuracy of the resulting classifier has been evaluated to observe whether it predicts the correct cluster of an experiment over unseen data. This has been done by 4-fold cross-validation (cv) as illustrated in Figure 3.14 and explained with an example below [M97].



Quenchant Name	Part Material	Agitation Level	Quenchant Temp	Oxide Layer	Probe Type	Cluster
DurixolHR88A	SS304	Absent	(140 -150)	None	CHTE	C
HoughtoQuenchG	SS304	Absent	(60 - 70)	None	CHTE	H
DurixolHR88A	SS304	Absent	(150 - 160)	None	CHTE	C
MarTemp355	Inconel600	High	(20 - 30)	None	IVF	G
DurixolHR88A	ST4140	Low	(100 - 110)	Thin	CHTE	D
T7A	SS304	Absent	(20 - 30)	None	CHTE	E
DurixolV35	Inconel600	Low	(50 - 60)	None	IVF	F
DurixolV35	ST4140	High	(60 - 70)	Thin	CHTE	B
DurixolW72	ST4140	Absent	(90 - 100)	Thin	CHTE	F
HoughtoQuenchG	SS304	Low	(60-70)	None	CHTE	H
HoughtoQuenchG	ST4140	Absent	(20-30)	None	CHTE	D
T7A	ST4140	High	(30-40)	Thick	CHTE	B
T7A	ST4140	Low	(30-40)	None	CHTE	A
T7A	Inconel600	High	(20-30)	None	IVF	G

Figure 3.14: Evaluation of AutoDomainMine with Cross Validation

Process of Evaluation. Consider a data set of 100 experiments. The graphs obtained from these experiments are clustered and the clustering output is sent to a decision tree classifier. This output consists of the input conditions of each experiment along with the cluster in which it was placed. The classifier is then constructed and evaluated as follows. In each fold, 75 experiments are used for training and the remaining 25 experiments for testing. Classifier accuracy is measured in terms of how well it predicts the correct cluster for the remaining 25 experiments. Thus if it predicts the correct cluster for 20 experiments out of 25, then its accuracy in that fold is 80%. The process is repeated 4 times, each time using 75 different experiments for training and the remaining 25 for testing. The average accuracy of these 4 folds is the classifier accuracy.

Parameters in the Tests. The parameters altered in these tests are as data set size, number of clusters, clustering seeds, decision tree classifier seeds and dimensionality reduction techniques. We consider selective sampling altering the number of selective samples and Fourier Transforms altering the number of Fourier coefficients. The results of the evaluation are shown below.

Effect of Data Set Size and Number of Clusters

These tests are conducted with data set sizes ranging from 100 to 500 graphs. For each data set, the number of clusters is varied from 10 to 50 in steps of 5. In these tests, the original curve, i.e., all n data points and in addition 3 points for the three critical regions are sent as the input to the clustering algorithm.

The evaluation results shown in the charts in Figures 3.15 to 3.19 are the accuracy values for each test. For these tests, and all the tests shown under cross validation, the time taken to build the model is observed to be approximately 0.1 seconds. Since it is almost the same for every test, it is not shown in the charts.

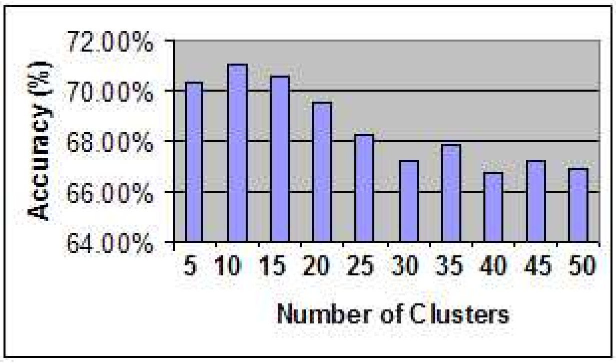


Figure 3.15: Effect of Number of Clusters with Data Set Size 100

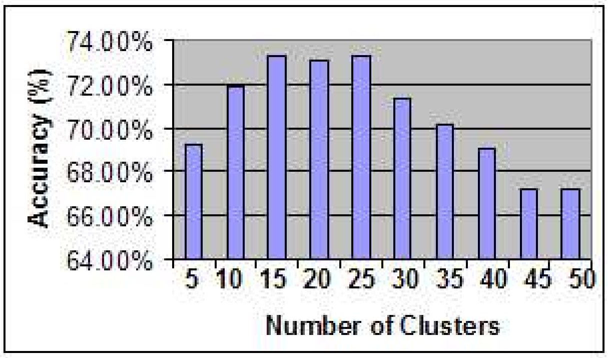


Figure 3.16: Effect of Number of Clusters with Data Set Size 200

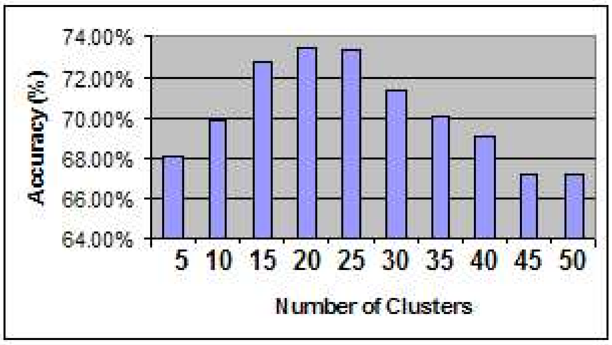


Figure 3.17: Effect of Number of Clusters with Data Set Size 300

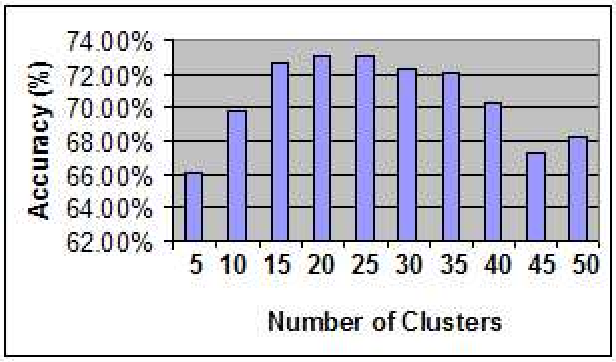


Figure 3.18: Effect of Number of Clusters with Data Set Size 400

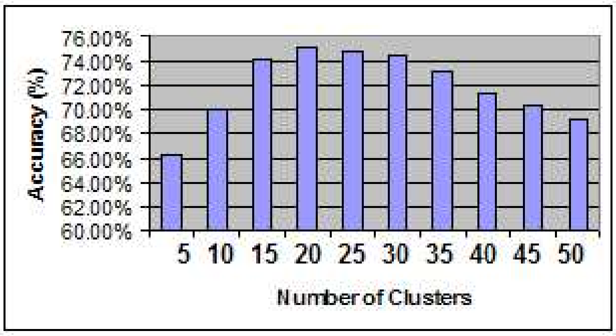


Figure 3.19: Effect of Number of Clusters with Data Set Size 500

Observations and Discussion

- As data set size increases, accuracy increases. This is because more training data is available for learning.
- In each data set, for very low values of k (number of clusters), accuracy is relatively low. This is probably because the number of experiments in each cluster in that case would be high, leading to too much generality in classifying new experiments.
- On the other hand, it is observed that for very high values of k also, the accuracy is fairly low. A probable interpretation for this would be that very high values of k lead to clustering that gets too specific, not allowing new experiments to be suitably categorized, thereby adversely affecting accuracy.
- An interesting observation is that for all the data sets, highest accuracy is observed at values of k close to square root of G, where k is the number of clusters and G is the number of graphs, i.e., the number of experiments in the data set, namely, the data set size. This seems to provide a middle ground between the two extremes discussed above pertaining to very low and very high values of k. These middle range values of k are likely to give a good trade-off between being too generic and too specific. Hence this is a parameter setting used in further experiments.

Effect of Selective Sampling

In these tests, equally spaced selective samples are taken per graph and sent as the input to the clustering technique. In addition to the selective samples on the original graph which include the critical regions, 3 additional samples are taken along the critical regions to emphasize their importance. Thus, for example, 53 selective samples, means that 50 samples are taken to capture the graph and 3 additional samples to capture its critical regions. The number of selective samples is varied from 23 to 203. The evaluation results in terms of accuracy are shown in Figures 3.20 to 3.24.

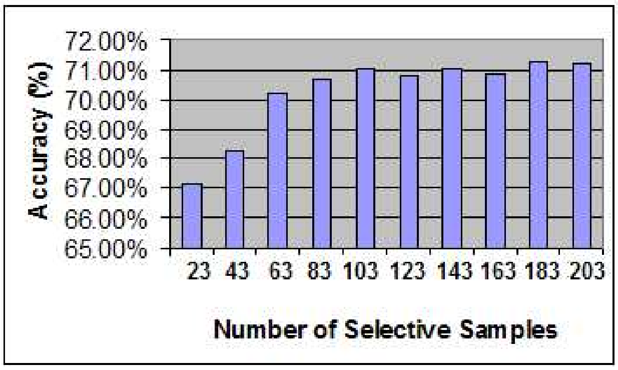


Figure 3.20: Effect of Selective Sampling with Data Set Size 100

Observations and Discussion

- As the number of selective samples increase, the accuracy tends to increase until about 103 to 123 samples.
- Accuracy levels out at approximately 103 to 123 selective samples and thereafter, increasing the number of samples does not substantially

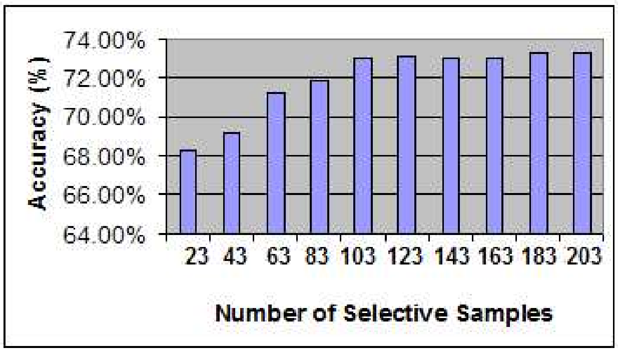


Figure 3.21: Effect of Selective Sampling with Data Set Size 200

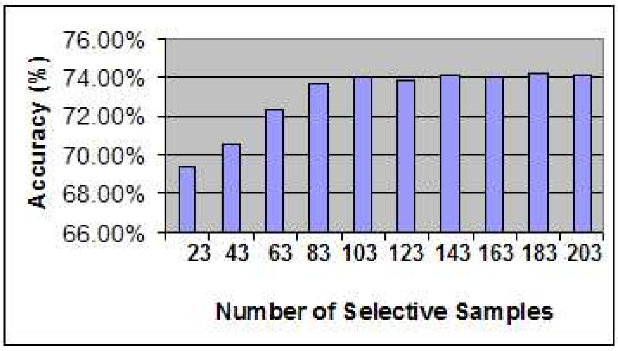


Figure 3.22: Effect of Selective Sampling with Data Set Size 300

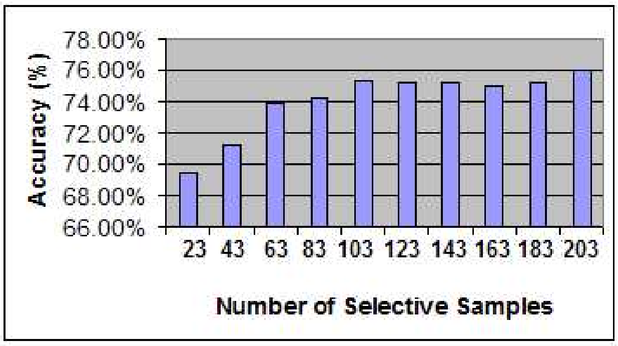


Figure 3.23: Effect of Selective Sampling with Data Set Size 400

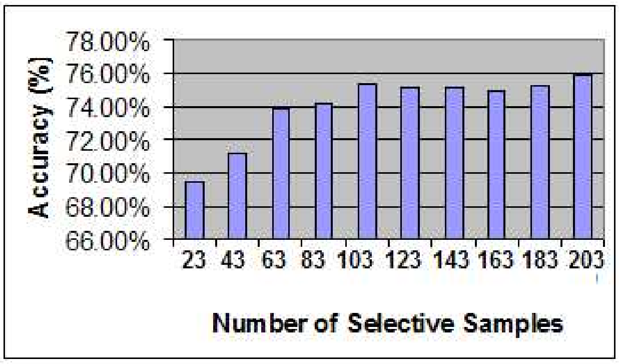


Figure 3.24: Effect of Selective Sampling with Data Set Size 500

increase the accuracy.

- As data set size increases, accuracy increases.
- Selective sampling provides higher efficiency since fewer dimensions are needed per curve.

Effect of Fourier Transforms

In these tests, the Fourier coefficients are used for clustering. The number of Fourier coefficients varies from 2 to 20. In these tests, there are no additional inputs for critical regions since the Fourier coefficients are such that they cannot be mapped to the critical regions of the original graph [F55]. The evaluation results are shown in Figures 3.25 to 3.29.

Observations and Discussion:

- For all data sets, highest accuracy is observed around the range of 16 Fourier coefficients. This corroborates the information given by

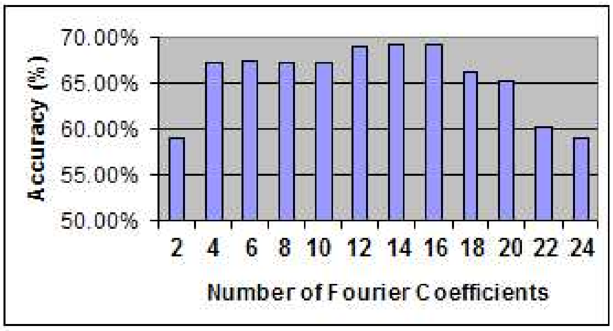


Figure 3.25: Effect of Fourier Transforms with Data Set Size 100

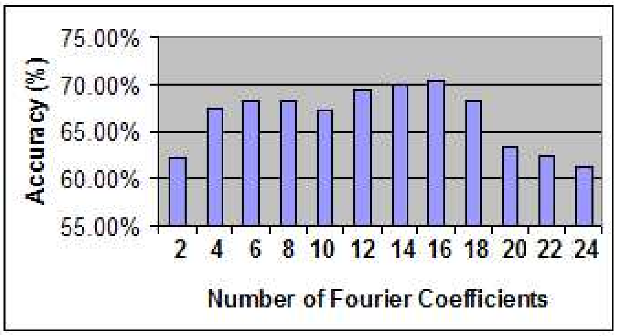


Figure 3.26: Effect of Fourier Transforms with Data Set Size 200

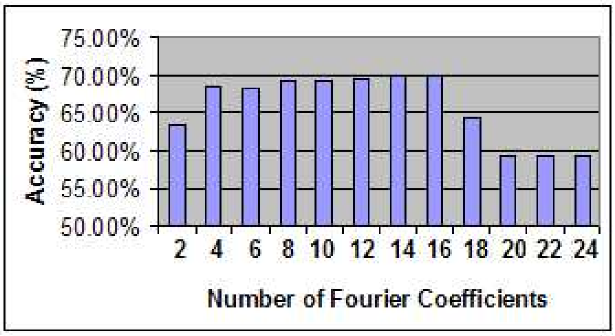


Figure 3.27: Effect of Fourier Transforms with Data Set Size 300

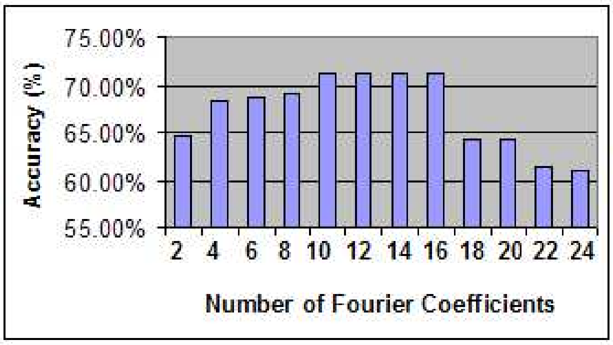


Figure 3.28: Effect of Fourier Transforms with Data Set Size 400

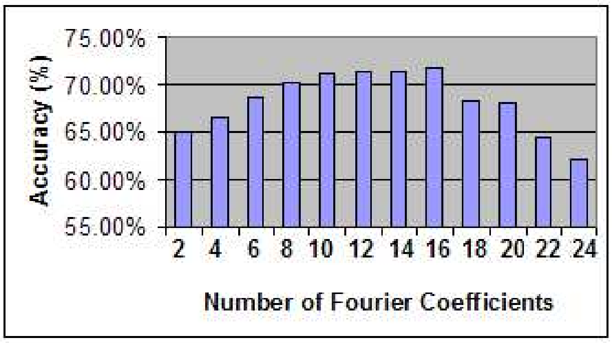


Figure 3.29: Effect of Fourier Transforms with Data Set Size 500

domain experts.

- Further increasing the number of Fourier coefficients reduces the accuracy. This is probably because the high frequency coefficients correspond to noise.

- For very few Fourier coefficients (around 2), the accuracy is also low. This could be because such few coefficients are not enough to capture the original graph.

- In general, the tests with Fourier coefficients give less accuracy than those with selective samples and with the whole graph. This can possibly be interpreted as follows. The basic property of the Fourier transform [F55] is that it converts a given waveform (in our case the graph) into frequency sinusoids such that they all sum back to the original waveform. Since it takes the integral of the waveform as a whole, the critical dimensions cannot be considered separately in mapping to the frequency domain. Hence the information about critical regions is not retained in the Fourier coefficients thus adversely affecting accuracy.

3.6.3 Comparative Evaluation with Clustering based on Conditions

In AutoDomainMine, the clustering is done based on the graphs. This is compared with the alternative of clustering based on the conditions. In the latter approach, the conditions of the experiments are used for clustering and the output of the clustering is sent to the decision tree classifier. The classifier accuracy is evaluated using 4-fold cross validation. The tests conducted with this approach are with data set sizes varying from 100 to 500 and number of clusters varying from 10 to 100 for each data set. The evaluation results are shown in Figures 3.30 to 3.34.

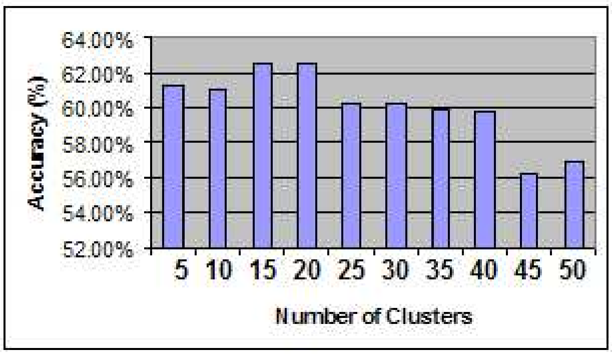


Figure 3.30: Effect of Clustering with Conditions on Data Set Size 100

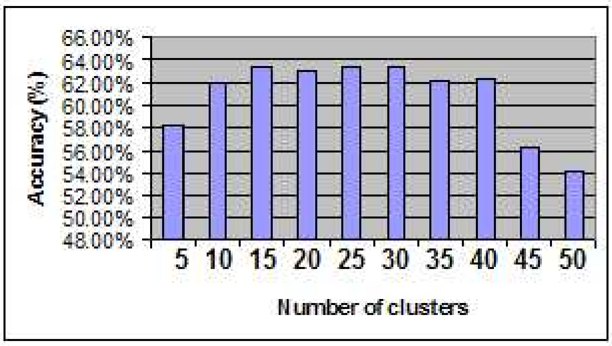


Figure 3.31: Effect of Clustering with Conditions on Data Set Size 200

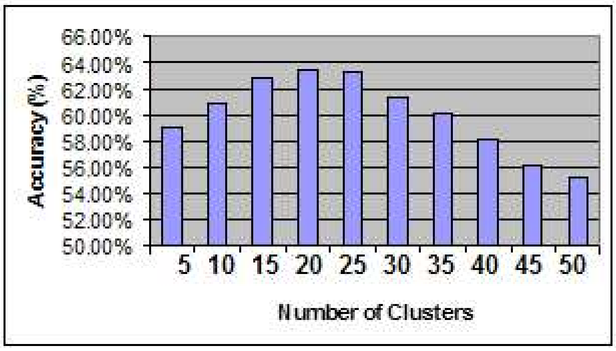


Figure 3.32: Effect of Clustering with Conditions on Data Set Size 300

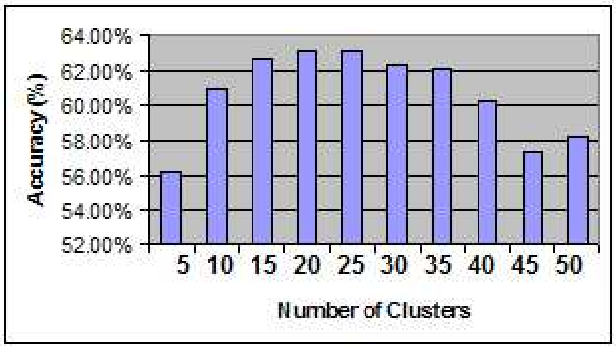


Figure 3.33: Effect of Clustering with Conditions on Data Set Size 400

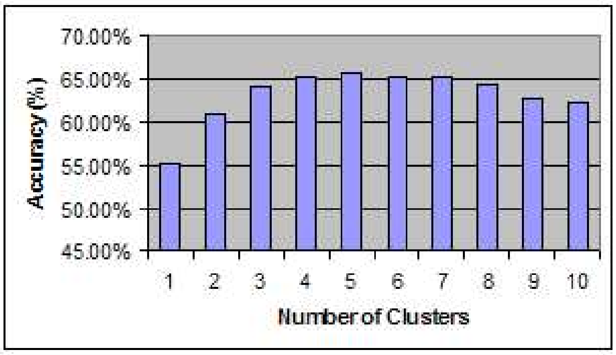


Figure 3.34: Effect of Clustering with Conditions on Data Set Size 500

Observations and Discussions

- As data set size increases, accuracy increases.
- The ranges of accuracy are around 60% to 65%.
- In general the accuracy in these tests is observed to be lower than in the tests with clustering based on graphs. Thus we prefer to cluster based on graphs which is also in keeping with the learning methods of scientists in the domain [SMMV04].

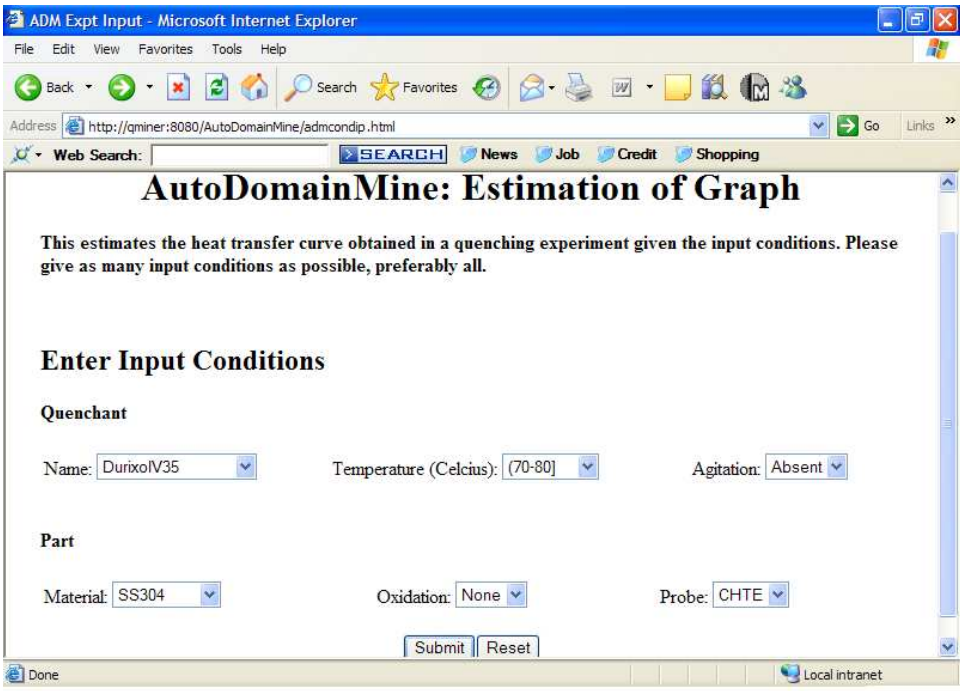
3.6.4 Evaluation with Domain Expert Interviews

This process of evaluation is explained below. Distinct test sets consisting of real laboratory experiments in Heat Treating not used for training the technique are used for testing. Domain experts execute test sets comparing the estimation with the real experiment. If the estimation matches the real experiment as per the satisfaction of the experts, then the test is considered accurate, else inaccurate. Accuracy is reported as the percentage of accurate tests over all the tests conducted. In addition, the efficiency of the technique is also noted. Efficiency refers to the response time of the tool, i.e., the amount of time required to perform the estimation given a new set of input conditions or graph.

Examples from evaluation are shown below with screen-dumps from the AutoDomainMine tool.

Estimation of Graph

The input conditions as shown in Figure 3.35 are submitted by a domain expert to AutoDomainMine for estimating the resulting graph, i.e., heat transfer coefficient curve.


ADM Expt Input - Microsoft Internet Explorer

Address http://qminer:8080/AutoDomainMine/admcondip.html

AutoDomainMine: Estimation of Graph

This estimates the heat transfer curve obtained in a quenching experiment given the input conditions. Please give as many input conditions as possible, preferably all.

Enter Input Conditions

Quenchant

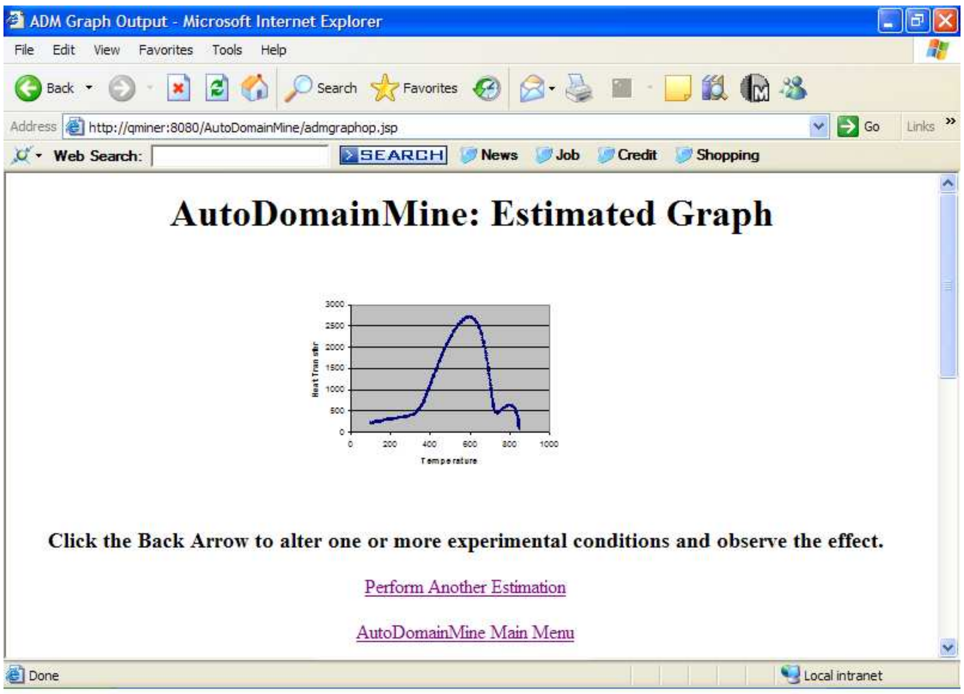
Name: DurixolV35 Temperature (Celcius): (70-80] Agitation: Absent

Part

Material: SS304 Oxidation: None Probe: CHTE

Figure 3.35: Given Conditions

The estimated graph is shown in Figure 3.36. On comparing this with the graph obtained in the laboratory experiment performed with the same input conditions (as stored in the test set), the domain experts conclude that this estimation is satisfactory.


ADM Graph Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://qminer:8080/AutoDomainMine/admgraphop.jsp
Go
Links
Web Search:
SEARCH
News
Job
Credit
Shopping
Heat Transfer
Temperature
Done
Local intranet

AutoDomainMine: Estimated Graph

Click the Back Arrow to alter one or more experimental conditions and observe the effect.

Perform Another Estimation

AutoDomainMine Main Menu

Figure 3.36: Estimated Graph

Estimation of Conditions

A domain expert submits the heat transfer coefficient curve as shown in Figure 3.37 to estimate the corresponding input conditions. This curve is selected from a given test set of curves not used for training the technique.

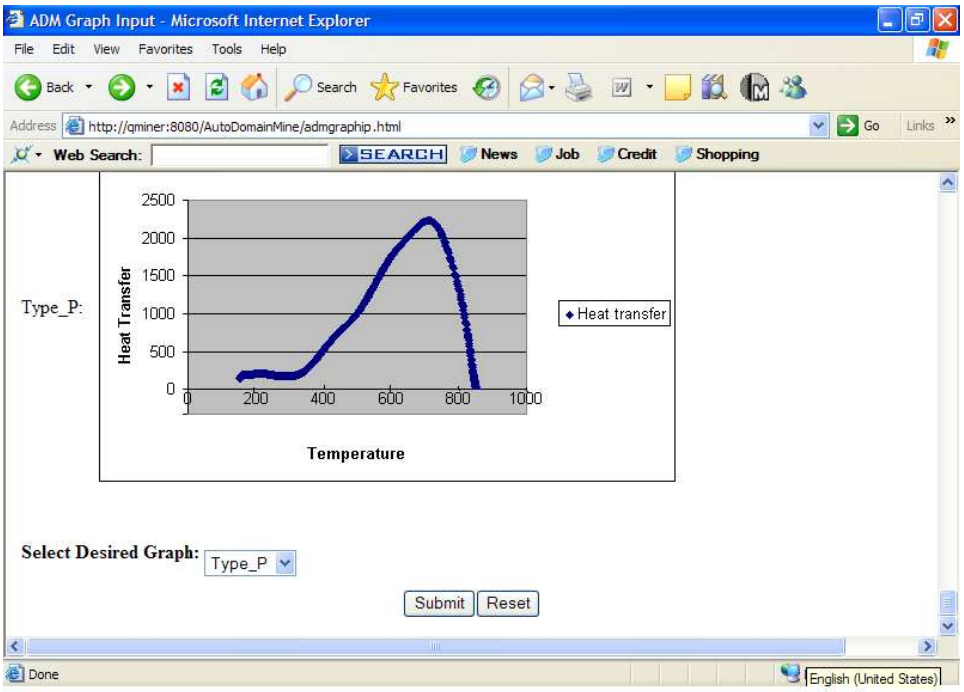
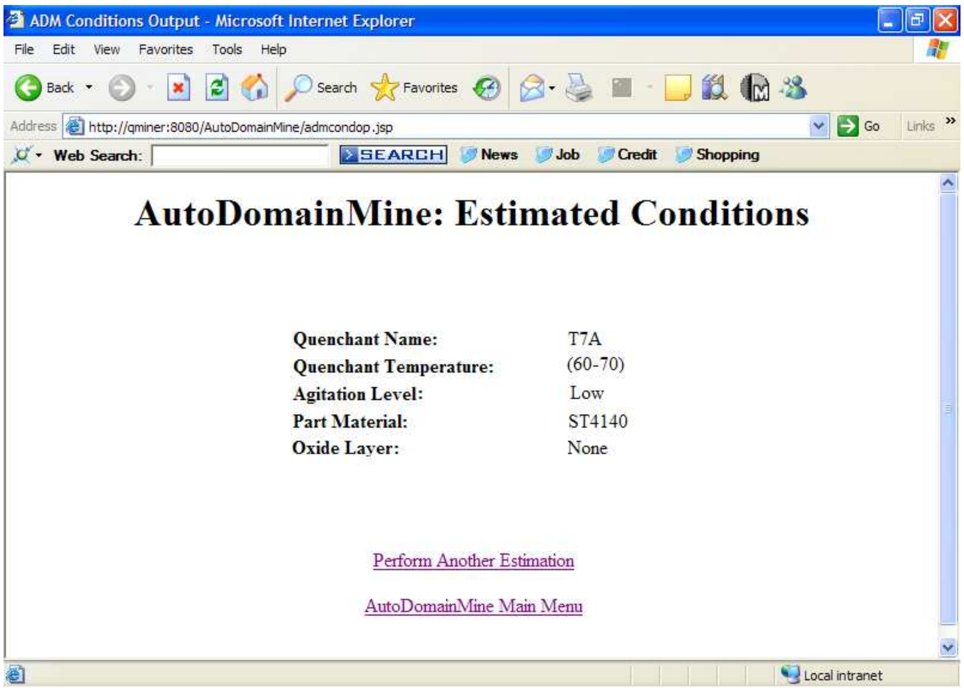


Figure 3.37: Given Graph

The estimated conditions are shown in Figure 3.38. On comparing these with the conditions of the laboratory experiment to achieve the same graph (as stored in the test set), the domain experts conclude that this estimation is satisfactory.

Likewise 100 tests have been conducted by domain experts with the help of work-study students in Materials Science. The estimation accuracy


ADM Conditions Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://qminer:8080/AutoDomainMine/admcondop.jsp
Go
Links
Web Search:
SEARCH
News
Job
Credit
Shopping
Local intranet

AutoDomainMine: Estimated Conditions

Quenchant Name:	T7A
Quenchant Temperature:	(60-70)
Agitation Level:	Low
Part Material:	ST4140
Oxide Layer:	None

Perform Another Estimation

AutoDomainMine Main Menu

Figure 3.38: Estimated Conditions

has been observed as approximately 75 percent, i.e., in 75 percent of the tests, the estimation matched the real experiment. The response time of the tool has been found on an average 0.1 second. This is illustrated in charts below along with the results of comparative evaluation.

3.6.5 Comparative Evaluation with Similarity Search

Domain experts have run tests using similarity search with the same test data that was used for AutoDomainMine estimation. The process of evaluation is described below.

A sample tool is built using naive similarity search. Given a set of input conditions, the tool compares them with the conditions of existing experiments. The closest matching conditions are found and the corresponding graph is output as the estimated result. This estimation is compared with the real laboratory experiment. If they match each other as per the satisfaction of the domain experts and students then the estimation is considered to be accurate. Accuracy is reported as the percentage of tests in which the estimation was accurate. The response time of the tool was also noted.

The results of the comparative evaluation between AutoDomainMine and similarity search are summarized in Figures 3.39 and 3.40 below. Figure 3.39 shows the estimation accuracy in terms of the percentage of accurate estimations out of the total number of tests conducted. Figure 3.40 shows the efficiency in terms of the average response time of the tool during all these tests.

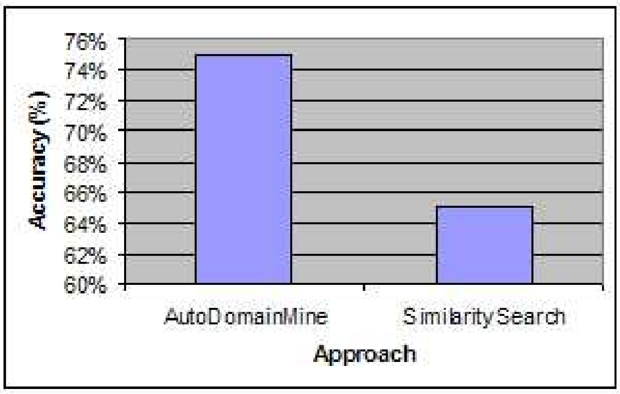


Figure 3.39: Comparison between Accuracy of AutoDomainMine and Similarity Search

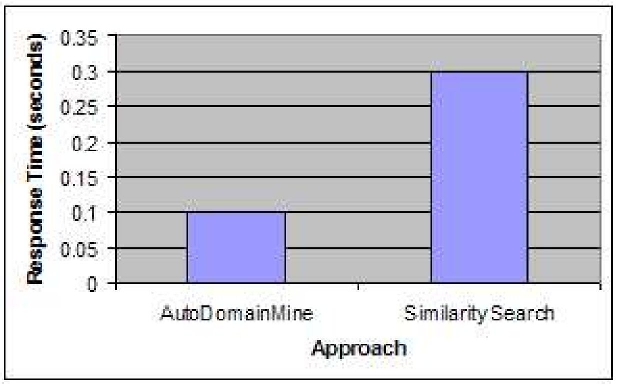


Figure 3.40: Comparison between Efficiency of AutoDomainMine and Similarity Search

Observations and Discussion

- The accuracy of AutoDomainMine is higher than that of similarity search. This is probably because AutoDomainMine performs the estimation based on discovering knowledge by automating the learning methods of domain experts.

- The response time of AutoDomainMine is lower than that of similarity search, implying that AutoDomainMine is more efficient. This happens because AutoDomainMine has to scan only the representatives as opposed to similarity search which scans the entire database of experiments to find the closest match.

3.6.6 Conclusions from the Pilot Tool Evaluation

It was inferred from the pilot tool that the basic AutoDomainMine approach works as a computational estimation technique and is better than state-of-the-art methods in the domain such as mathematical modeling. However potential for further improvement was identified at this stage.

The pilot tool, in addition to evaluating the effectiveness of the basic learning technique in AutoDomainMine, also serves as a yardstick for comparison with later stages of the tool.

Chapter 4

LearnMet: Learning Domain-Specific Distance Metrics for Graphical Plots

4.1 Need for Distance Metric Learning

4.1.1 Motivation

In the clustering step of AutoDomainMine the notion of similarity for the clustering algorithm [KR94] is based on distance. The default notion is Euclidean distance [HK01]. However this poses certain problems as illustrated with an example in Figure 4.1. Clustering with Euclidean distance places the two heat transfer curves shown in the figure in the same cluster relative to other curves, even though one has a visible Leidenfrost point while the other does not. This is a problem since the two curves depict distinctly different physical tendencies, as confirmed by domain experts. Hence the inferences drawn from such clustering would be incorrect.

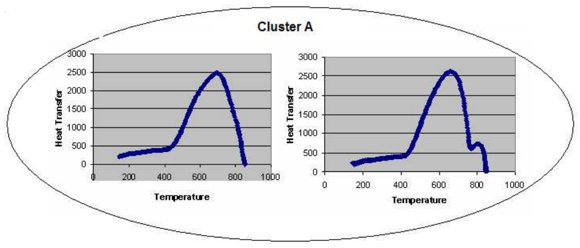


Figure 4.1: Example of Inaccuracy in Clustering

4.1.2 Distance Metrics

We now review distance metrics. Several distance metrics exist in the literature. We describe the distance metric types relevant to our domains with respect to 2 n-dimensional objects A and B given by $A(A_1, A_2 \ldots A_n)$ and $B(B_1, B_2 \ldots B_n)$ respectively.

Position-based Distances

They refer to distances based on the absolute position of the objects [HK01]. Examples of position-based distances are:

Euclidean Distance. This is the *as-the-crow-flies* distance between objects, calculated as:

$$D_{Euclidean}(A, B) = \sqrt{\Sigma_i^n (A_i - B_i)^2}$$

Manhattan Distance. This is the *city-block* distance between objects. It is calculated as:

$D_{Manhattan}(A, B) = \Sigma_i^n |A_i - B_i|$

Statistical Distances

These refer to distances based on statistical observations in the objects [PNC99]. Examples of statistical distances are:

Mean Distance. This is the distance between mean values of the objects, given as:

$D_{Mean}(A, B) = |Mean(A) - Mean(B)|$

Maximum Distance. This is the distance between maximum values of the objects. It is calculated as:

$D_{Max}(A, B) = |Max(A) - Max(B)|$

Minimum Distance. This is the distance between minimum values of the objects. It is calculated as:

$D_{Min}(A, B) = |Min(A) - Min(B)|$

Critical Distances

In addition to the distance metric types reviewed above, we introduce the concept of critical distances for graphical plots applicable to our targeted domains. A critical distance metric is defined in general as follows.

Critical Distance Metric. Given two graphical plots A and B, a critical distance metric is a metric that represents the distance between critical regions of A and B where a critical region represents the occurrence of a significant physical phenomenon. Each such metric is calculated in a domain-specific manner as explained with examples below.

In order to give examples of critical distances, we refer to the critical points on the heat transfer curve shown in Figure 4.2. These are the Leidenfrost point LF, the Boiling Point BP and the Slow Cooling Point SC. Note that this curve also shows the MAX and MIN points that serve to define statistical distances D_{Max} and D_{Min} respectively.

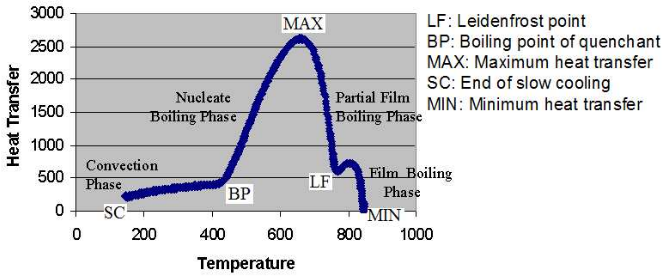


Figure 4.2: Points to define Distances on a Heat Transfer Curve

Given this, the Leidenfrost distance is the distance between the Leidenfrost points [TBC93] on two heat transfer curves calculated as follows. For curves A and B, $D_{LF}(A,B) = \sqrt{(A_{TLF} - B_{TLF})^2 + (A_{hLF} - B_{hLF})^2}$ where TLF is the temperature at Leidenfrost Point and hLF is the heat transfer coefficient at that point.

Another critical distance is the Boiling Point distance. This is the distance between points on the curves corresponding to the Boiling Points of the respective cooling media in heat treatment [TBC93]. Thus for two curves A and B, the Boiling Point distance D_{BP} is given as $D_{BP}(A,B) = \sqrt{(A_{TBP} - B_{TBP})^2 + (A_{hBP} - B_{hBP})^2}$ where TBP is the temperature at

Boiling Point and hBP is the heat transfer coefficient at that point.

Likewise the Slow Cooling distance can also be considered with reference to the heat transfer curves. This is the distance between points on the curves corresponding to the end of the Slow Cooling phase [TBC93] as shown by SC in Figure 1. Thus for any curves A and B, $D_{SC}(A,B) = \sqrt{(A_{TSC} - B_{TSC})^2 + (A_{hSC} - B_{hSC})^2}$ where TBP is the temperature at the Slow Cooling point and hBP is the heat transfer coefficient at that point.

4.1.3 Need to Learn Domain-Specific Metrics

Although a variety of metrics from the literature could be selected, it is seldom known which of them work best while preserving the domain semantics. When applied to a given problem, each of the selected metrics serves as a single distinguishing factor. That is, each such metric serves to separate the graphical plots based on a given feature. For example, Euclidean distance separates them based on the absolute position of points in the plots. Critical distances separates them based on the occurrence of the domain-specific physical phenomena they represent. However, in order to capture semantics accurately, it is essential to consider several distinguishing factors, each being represented by one or more of these metrics. Domain experts may at best have subjective notions about the usefulness of these metrics. It is thus desirable to learn a distance metric encompassing the various notions of distance applicable to the domain, in order to adequately represent domain semantics when analyzing the scientific plots. This is precisely the focus of this dissertation sub-problem.

4.2 Related Work

4.2.1 Metrics for Graphical Data

Approximate Neighborhood Function

Techniques have been developed for data mining over graphical data types. In [PGF02], they use an Approximate Neighborhood Function for comparison, focuses on a fast and scalable method for mining. However this compromises on accuracy. For our goals, accuracy is more important than efficiency.

Tri-Plots

The Tri-plots [TTPF01] technique provides a scalable tool for multidimensional data mining. This enables mining over data of various shapes and dimensions using cumulative distribution functions of pair-wise distances between two or more objects. However, they focus on the the intrinsic dimensionality of multi-dimensional objects. They not take into account the position of the objects nor critical data points in them. In the context of the problem defined here, the basic dimensionality of all the objects is the same, since they are all 2-dimensional graphs plotting the behavior of process parameters. Instead we need to focus on the semantics of individual graphs.

Edit Distances

Chen and Ozsu [CO03] compare the use of different metrics such as DTW (Dynamic Time Warping) and LCSS (Longest Common Sub-Sequence) for similarity-based retrieval of time series data. However, they do not consider semantics of the data in the retrieval. Also in this work they do not learn a single metric involving several individual metrics. In [CN04] Chen and Ng propose a new metric based on the marriage of Lp-Norm and Edit distance also for time-series data. This metric called ERP (Edit Distance with Real Penalty) satisfies metric properties and also local time shifting making it good for querying over a time series and also for index structures applicable to metrics. However this metric is more suitable for continuously varying data where some of the properties involved are needed for comparison. Our data is not of a time-varying nature and hence such properties are not needed.

4.2.2 Searching Multimedia Data

Keim et. al. [KB04] present an overview of various distance metrics for similarity searching in multimedia databases. They focus on the content-based retrieval of similar objects. They take into account a variety of data such as text, images and audio in a single query. Rather than finding an exact match, since that is not feasible for multimedia objects, they develop efficient and effective similarity search techniques using various state-of-the-art distance metrics. However, they do not propose the learning of a single distance metric that combines various components. Nor do they define the aspect of critical distance that is important in our targeted domains. They focus on a variety of data, while our data is primarily graphical. Our focus is on the detailed analysis of graphs, as opposed to a general search over different categories of multimedia data. Some of the distance types and subtypes overviewed in their work however could form the types and subtypes in learning our domain-specific distance metric.

4.2.3 Learning Metrics for Nearest Neighbor Searches

Learning Position-based Distances

It is often desirable to learn a distance metric in order to guide the search for the nearest neighbor, because the precise metric may not be known in advance in several applications. Xing et. al. propose a technique that, given examples of similar and dissimilar pairs of points, learns a distance metric that respects these relationships [XNJR03]. They start with a generic distance formula that parameterizes a family of Mahalanobis distances. They then try to learn the value of a particular variable in that formula which determines whether the distance is Euclidean, or Manhattan or any other. They use gradient descent and the idea of iterative projections. They repeatedly take a gradient step and then project the variable onto the given training set so that the constraints of similarity and dissimilarity among the pairs of points are satisfied [XNJR03]. However, in the learning process, they only focus on position-based distances defined by the generic Mahalanobis distance formula. They do not consider other aspects such as statistical distances, shape-based distances and critical distances. In the context of our problem, all these aspects are important. Depending on the domain, some may be more important than others. Thus, we cannot focus our metric learning on a general category of position-based distances only.

Learning Relative Importance of Dimensions

When the search for the nearest neighbor occurs in high dimensional spaces, an interesting problem is how to determine the relative importance of the dimensions. Hinneburg et. al. propose an algorithm to solve this generalized nearest neighbor problem [HAK00]. They use a greedy search and a quality criterion to select relevant dimensions or projections with respect to a given query. As an example of a meaningful quality criterion they rate how well the data is clustered around a given query point within the selected projection. They call this the projected nearest neighbor search. However, in the real world application, the quality function has to be modified due to the data dependency of the term "meaningful". Also, their search uses a basic distance metric such as Euclidean or Manhattan and then learning the relative importance of the dimensions only. For our work it is essential to learn the basic metric itself.

4.2.4 Genetic Algorithms, Neural Networks and Support Vector Machines

Genetic algorithms [F58] can be used to select features in graphs relevant for clustering thus trying to learn a distance metric. However, this does not give enough accuracy in our applications. Neural networks [B96] could possibly be used for distance metric learning. However our data is such that the distance between pairs of plots is not known in advance to serve as the training set required. Similar issues hold for other learning techniques such as support vector machines [M97] since we do not have positive and negative training samples available in advance as required for learning. However, exploring distance metric learning using these and other approaches as alternatives and comparing them with LearnMet presents topics for future work.

4.2.5 Ensemble Learning

Zhou et. al. [ZWT02] propose an approach for ensembling 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. At present, we use an approach analogous to greedy search to learn simple metrics [M97]. Considering other approaches for such subproblems within LearnMet and comparing their computational complexity and accuracy with our present approach presents interesting future issues.

4.3 Proposed Approach: LearnMet

We propose a technique called LearnMet [VRRMS0805] to learn semantics-preserving distance metrics for graphical plots. The input to LearnMet is a training set with actual clusters of such graphs provided by domain experts. The five basic steps of our technique are: (1) guess an initial metric D as a weighted sum of distance metrics applicable to the domain; (2) use that metric D 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 D based on the error between the predicted and actual clusters, and re-execute the clustering and evaluation until error is below a threshold or maximum number of epochs is reached; (5) output the metric D giving lowest error as the learned metric. Note that an epoch refers to a run of all the 5 steps of LearnMet.

The LearnMet technique is summarized in the flowchart in Figure 4.3.

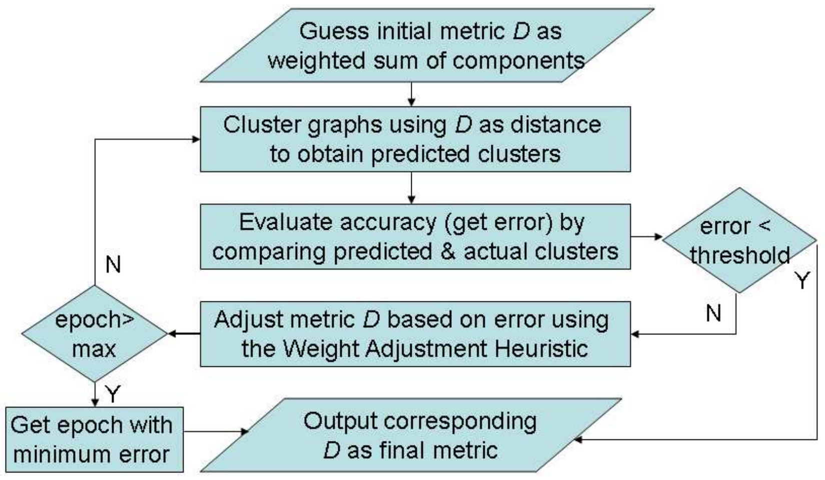

Guess initial metric D as weighted sum of components
Cluster graphs using D as distance to obtain predicted clusters
Evaluate accuracy (get error) by comparing predicted & actual clusters
error < threshold
N
Y
Adjust metric D based on error using the Weight Adjustment Heuristic
epoch> max
N
Y
Get epoch with minimum error
Output corresponding D as final metric

Figure 4.3: Flowchart on LearnMet

The details of LearnMet are discussed in the subsections to follow.

4.4 The LearnMet Distance

In order to proceed with the discussion of the learning strategy, we first define the following terminology.

4.4.1 Definition of LearnMet Distance

A LearnMet distance D is a weighted sum of components, where each component can be a position-based, a statistical, or a critical distance. The weight of each component is a numerical value indicating its relative importance in the domain.

Thus a LearnMet distance is of the form $D = \Sigma_{i=1}^{m} w_i D_i$ where each D_i is a component, w_i is its weight, and m is number of components.

It is desirable that the LearnMet distance is a metric so that clustering algorithms requiring the notion of distance to be a metric [KR94] can be applied. Also, pruning during similarity search can be done using triangle equality [HK01]. We now discuss distance metric properties with reference to LearnMet.

In general, the properties required for any distance function to be a metric are listed below with respect to objects P, Q and R in n-dimensional space [HK01].

1). Distance should be non-negative, i.e., $Distance(P, Q) >= 0$.

2). Distance of an object to itself should be zero, i.e., $Distance(P, P) = 0$.

3). Distance should be commutative, i.e., $Distance(P, Q) = Distance(Q, P)$.

4). Distance should satisfies triangle inequality, i.e., if 3 objects P, Q and R form a triangle in n-dimensional space,then sum of any two sides is greater than the third, i.e..,

- $Distance(P, Q) + Distance(Q, R) > Distance(P, R)$
- $Distance(P, Q) + Distance(P, R) > Distance(Q, R)$
- $Distance(P, R) + Distance(Q, R) > Distance(P, Q)$

With reference to the above properties, we state below the conditions for the LearnMet distance $D = \Sigma_{i=1}^{m} w_i D_i$ to be a metric.

4.4.2 Conditions for the distance function D in LearnMet to be a Metric

The sufficient conditions for the distance $D = \Sigma_{i=1}^{m} w_i D_i$ to be a metric are stated as Theorem 1.

Theorem 1

If each component D_i is a distance metric and each weight $w_i >= 0$ then $D = \Sigma_{i=1}^{m} w_i D_i$ is a distance metric, i.e., it satisfies the metric properties.

Proof of Theorem 1 Since each component D_i is a metric, it is known that:

1). $D_i(P, Q) >= 0$

2). $D_i(P, P) = 0$

3). $D_i(P, Q) = D_i(Q, P)$

4). If P, Q and R form a triangle in n-dimensional space then,

- $D_i(P,Q) + D_i(Q,R) > D_i(P,R)$
- $D_i(P,Q) + D_i(P,R) > D_i(Q,R)$
- $D_i(P,R) + D_i(Q,R) > D_i(P,Q)$

Now consider $D(P,Q)$ using $D = \Sigma_{i=1}^{m} w_i D_i$.

Thus $D(P,Q) = \Sigma_{i=1}^{m} D_i(P,Q) = w_1 D_1(P,Q) + w_2 D_2(P,Q) + \cdots + w_m D_m(P,Q)$

Consider each individual metric property for $D = \Sigma_{i=1}^{m} w_i D_i$

Property 1: Distance is non-negative Since each component D_i is a metric,

$D_1(P,Q) >= 0$, $D_2(P,Q) >= 0 \ldots D_m(P,Q) >= 0$

Hence for all $w_i >= 0$,

$w_1 D_1(P,Q) >= 0$ (Eqn. 1.1)

$w_2 D_2(P,Q) >= 0$ (Eqn. 1.2)

$\ldots$

$\ldots$

$\ldots$

$w_m D_m(P,Q) >= 0$ (Eqn. 1.m)

Summing the equations (1.1) $\ldots$ (1.m),

$w_1 D_1(P,Q) + w_2 D_2(P,Q) + \cdots + w_m D_m(P,Q) >= 0$

Hence $D(P,Q) = \Sigma_{i=1}^{m} w_i D_i(P,Q) >= 0$

Hence Property 1 is satisfied.

Property 2: Distance of an object to itself is zero Since each component D_i is a metric,

$Dc_1(P,P) >= 0$, $D_2(P,P) = 0 \ldots D_m(P,P) = 0$

Hence for all $w_i >= 0$,

$w_1 D_1(P,P) = 0$ (Eqn. 2.1)

$w_2 D_2(P,P) = 0$ (Eqn. 2.2)

$\ldots$

$\ldots$

$\ldots$

$w_m D_m(P,P) = 0$ (Eqn. 2.m)

Summing the equations (2.1) … (2.m),

$w_1 D_1(P,P) + w_2 D_2(P,P) + \cdots + w_m D_m(P,P) = 0$

Hence $D(P,P) = \Sigma_{i=1}^{m} w_i D_i(P,P) = 0$

Hence Property 2 is satisfied.

Property 3: Distance is commutative Since each component D_i is a metric,

$D_1(P,Q) = D_1(Q,P)$, $D_2(P,Q) = D_2(Q,P) \ldots D_m(P,Q) = D_m(Q,P)$

Hence for all $w_i >= 0$,

$w_1 D_1(P,Q) = w_1 D_1(Q,P)$ (Eqn. 3.1)

$w_2 D_2(P,Q) = w_2 D_2(Q,P)$ (Eqn. 3.2)

$\ldots$

$\ldots$

$\ldots$

$w_m D_m(P,Q) = w_m D_m(Q,P)$ (Eqn. 3.m)

Summing the equations (3.1) … (3.m),

$w_1 D_1(P,Q) + w_2 D_2(P,Q) + \cdots + w_m D_m(P,Q)$

$= w_1 D_1(Q,P) + w_2 D_2(Q,P) + \cdots + w_m D_m(Q,P)$

Hence $\Sigma_{i=1}^{m} w_i D_i(P,Q) = \Sigma_{i=1}^{m} w_i D_i(Q,P)$

Hence $D(P,Q) = D(Q,P)$

Hence Property 3 is satisfied.

Property 4: Triangle inequality Since each component D_i is a metric,

If P, Q and R form a triangle in n-dimensional space then,

$D_1(P,Q) + D_1(Q,R) > D_1(P,R)$, $D_1(P,Q) + D_1(P,R) > D_1(Q,R)$, $D_1(P,R) + D_1(Q,R) > D_1(P,Q)$;

$D_2(P,Q) + D_2(Q,R) > D_2(P,R)$, $D_2(P,Q) + D_2(P,R) > D_2(Q,R)$, $D_2(P,R) + D_2(Q,R) > D_2(P,Q)$;

…

…

…

$D_m(P,Q) + D_m(Q,R) > D_m(P,R)$, $D_m(P,Q) + D_m(P,R) > D_m(Q,R)$, $D_m(P,R) + D_m(Q,R) > D_m(P,Q)$;

Hence for all $w_i >= 0$,

$w_1 D_1(P,Q) + w_1 D_1(Q,R) > w_1 D_1(P,R)$ (Eqn. 4.1.1)

$w_1 D_1(P,Q) + w_1 D_1(P,R) > w_1 D_1(Q,R)$ (Eqn. 4.1.2)

…

$w_1 D_1(P,R) + w_1 D_1(Q,R) > w_1 D_1(P,Q)$ (Eqn. 4.1.m)

$w_2 D_2(P,Q) + w_2 D_2(Q,R) > w_2 D_2(P,R)$ (Eqn. 4.2.1)

$w_2D_2(P,Q) + w_2D_2(P,R) > w_2D_2(Q,R)$ (Eqn. 4.2.2)

$\dots$

$w_2D_2(P,R) + w_2D_2(Q,R) > w_2D_2(P,Q)$ (Eqn. 4.2.m)

$\dots$

$\dots$

$\dots$

$w_mD_m(P,Q) + w_mD_m(Q,R) > w_mD_m(P,R)$ (Eqn. 4.m.1)

$w_mD_m(P,Q) + w_mD_m(P,R) > w_mD_m(Q,R)$ (Eqn. 4.m.2)

$\dots$

$w_mD_m(P,R) + w_mD_m(Q,R) > w_mD_m(P,Q)$ (Eqn. 4.m.m)

Summing the equations (4.1.1), (4.2.1) ... (Eqn. 4.m.1),

$w_1D_1(P,Q)+w_1D_1(Q,R)+w_2D_2(P,Q)+w_2D_2(Q,R)+\cdots+w_mD_m(P,Q)+ w_mD_m(Q,R) > w_1D_1(P,R)$ + $w_2D_2(P,R)$ + ... + $w_mD_m(P,R)$ (Eqn. 5)

Equation 5 can also be written as,

$\Sigma_{i=1}^{m} w_iD_i(P,Q) + \Sigma_{i=1}^{m} w_iD_i(Q,R) > \Sigma_{i=1}^{m} w_iD_i(P,R)$

Similarly, summing the equations (4.1.2), (4.2.2) ... (4.m.2),

$w_1D_1(P,Q)+w_1D_1(P,R)+w_2D_2(P,Q)+w_2D_2(P,R)+\cdots+w_mD_m(P,Q)+ w_mD_m(P,R) > w_1D_1(Q,R) + w_2D_2(Q,R) + \cdots + w_mD_m(Q,R)$ (Eqn. 6)

Equation 6 can also be written as,

$\Sigma_{i=1}^{m} w_iD_i(P,Q) + \Sigma_{i=1}^{m} w_iD_i(P,R) > \Sigma_{i=1}^{m} w_iD_i(Q,R)$

Similarly, summing the equations (4.1.m), (4.2.m) ... (4.3.m),

$w_1D_1(P,R)+w_1D_1(Q,R)+w_2D_2(P,R)+w_2D_2(Q,R)+\cdots+w_mD_m(P,R)+w_mD_m(Q,R) > w_1D_1(P,Q)+w_2D_2(P,Q)+\cdots+w_mD_m(P,Q)$ (Eqn. 7)

Equation 7 can also be written as,

$\Sigma_{i=1}^{m} w_iD_i(P,R) + \Sigma_{i=1}^{m} w_iD_i(Q,R) > \Sigma_{i=1}^{m} w_iD_i(P,Q)$

Hence from equations (5), (6) and (7),

- $D(P,Q) + D(Q,R) > D(P,R)$
- $D(P,Q) + D(P,R) > D(Q,R)$
- $D(P,R) + D(Q,R) > D(P,Q)$

Hence Property 4 of triangle inequality is satisfied.

Conclusion: Hence it has been proved above that all properties of distance metrics are satisfied for $D = \Sigma_{i=1}^{m} w_iD_i$. Thus, if each component D_i is a metric and each weight $w_i >= 0$, then $D = \Sigma_{i=1}^{m} w_iD_i$ is a distance metric. This proves Theorem 1.

In our targeted applications, conditions in Theorem 1 are satisfied. Since the plots have interval-scaled variables, distances applicable to them are metrics [HK01], this is sufficient to say that each is component a metric. Also, we consider only non-negative weights since negative weights do not have a semantic interpretation in our targeted applications [SMMV04, TBC93].

4.5 Details of LearnMet

The steps of the technique are as follows.

1). Initial Metric Step

2). Clustering Step

3). Cluster Evaluation Step

4). Weight Adjustment Step

5). Final Metric Step

The details of each of these steps are explained below.

4.5.1 Initial Metric Step

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. If the experts have subjective notions about the relative importance of the components, this information is used to assign initial weights. An Initial Weight Heuristic is proposed.

Initial Weight Heuristic: Assign initial weights to the components in the LearnMet distance metric based on the relative importance of the components in the domain.

The relative importance of the components can be inferred from the subjective notion of the domain experts. If this relative importance is unknown then random weights are assigned to all components. Initial weights are typically assigned on a scale of 0 to 10.

4.5.2 Clustering Step

The domain experts provide a set of actual clusters over a training set of graphical plots. In order to perform clustering in LearnMet, an arbitrary but fixed clustering algorithm such as k-means [M67] is selected. Using $D = \Sigma_{i=1}^{m} w_i Dc_i$ as the distance metric, k clusters are constructed using the selected algorithm, where k is the number of actual clusters in the given training set. The clusters obtained from the algorithm using metric D are referred to as the predicted clusters.

4.5.3 Cluster Evaluation Step

The cluster evaluation involves comparing the predicted and actual clusters over the training set with each other. An example of predicted and actual clusters of plots is shown in Figure 4.4.

Ideally, the predicted clusters should match the actual clusters perfectly. Any difference between predicted and actual clusters 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_a and g_b, we say that:

- (g_a, g_b) is a True Positive (TP) pair if g_a and g_b are in the same actual cluster and in the same predicted cluster, e.g., (g_1, g_2).

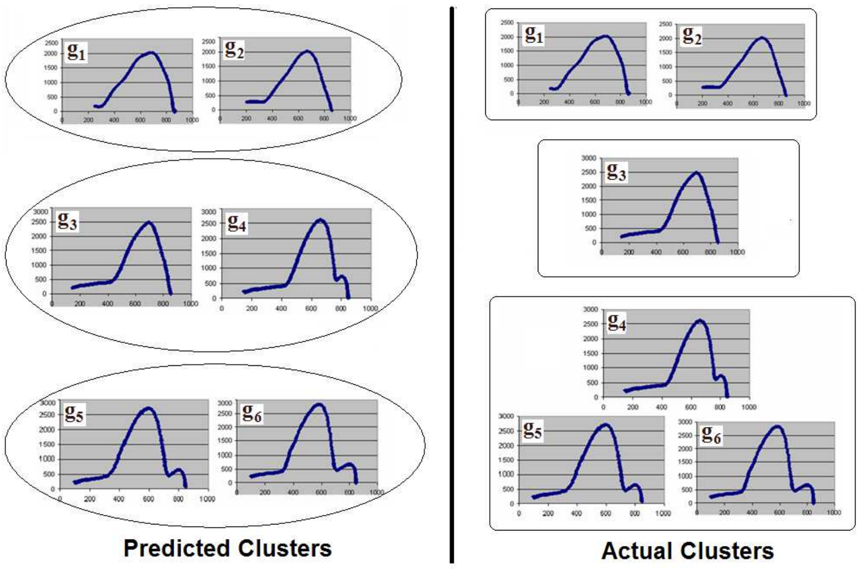


Figure 4.4: Predicted and Actual Clusters

- (g_a, g_b) is a True Negative (TN) pair if g_a and g_b are in different actual clusters and in different predicted clusters, e.g., (g_1, g_3).
- (g_a, g_b) is a False Positive (FP) pair if g_a and g_b are in different actual clusters but in the same predicted cluster, e.g., (g_3, g_4).
- (g_a, g_b) is a False Negative (FN) pair if g_a and g_b are in the same actual cluster but in different predicted clusters, e.g., (g_4, g_5).

Figure 30 includes examples of each of these kinds of pairs. The pair (g_1, g_2) is a true positive; (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 failure rate which is defined below [WF00].

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 $FR = (1 - SR)$ denote the Failure Rate.

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

Hence $FR = \frac{FP+FN}{TP+TN+FP+FN}$

In our context, false positives and false negatives are equally undesirable. Hence, our definition of failure rate weighs them equally.

Given a number G of graphs in the training set, the total number of pairs P is given by G *choose* 2, i.e, $P = \frac{G!}{2!(G-2)!}$ [PNC99]. Thus, for 25 graphs there are 300 pairs, for 50 graphs, 1225 pairs, etc.

Overfitting

To avoid overfitting in LearnMet, we use an approach analogous to incremental gradient descent [B96]. Instead of using all pairs of graphs for evaluation, 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. If $ppe = 15$, then we have a total of *300 choose 15* i.e, $\frac{300!}{15!285!}$ distinct pairs for learning [PNC99]. Thus in each epoch 15 randomly selected pairs can be used. This still gives a large number of epochs with distinct pairs for learning. This incremental approach reduces time complexity and helps avoid overfitting. Determining good ppe values is an enhancement issue and will be discussed in Section 4. Also in LearnMet, the random seed is altered in the clustering algorithm in different epochs as an additional method to avoid overfitting. This refers to the seed in the algorithm such as k-means [M67] used in clustering.

Error Threshold

Ideally, the error i.e., failure rate in an epoch should be zero. However, in practice a domain-specific error threshold t is used. A domain-specific error threshold t is the extent of error allowed per epoch in the domain, where error is measured by failure rate.

If the error is below threshold then the final distance metric is output as explained in step 5. However, if the error is not below threshold in a given epoch, then the metric is adjusted based on this error as explained below.

4.5.4 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 $D(g_a, g_b)$ between a pair of graphs g_a and g_b is the weighted sum of components in the plots using metric D. Thus, $D(g_a, g_b) = w_1 D_1(g_a, g_b) + \cdots + w_m D_m(g_a, g_b)$.

The concept of average distance between false positive and false negative pairs is now introduced.

Average False Negative (DFN) **and False Positive** (DFP) **Distances**

The distances DFN and DFP are defined as the average distance using the metric D of the false negative pairs and of the false positive pairs respectively. These are calculated as:

$DFN = (1/FN)\Sigma_{j=1}^{FN} D(g_a, g_b)$ where (g_a, g_b) denotes each FN pair.

$DFP = (1/FP)\Sigma_{j=1}^{FP} D(g_a, g_b)$ where (g_a, g_b) denotes each FP pair.

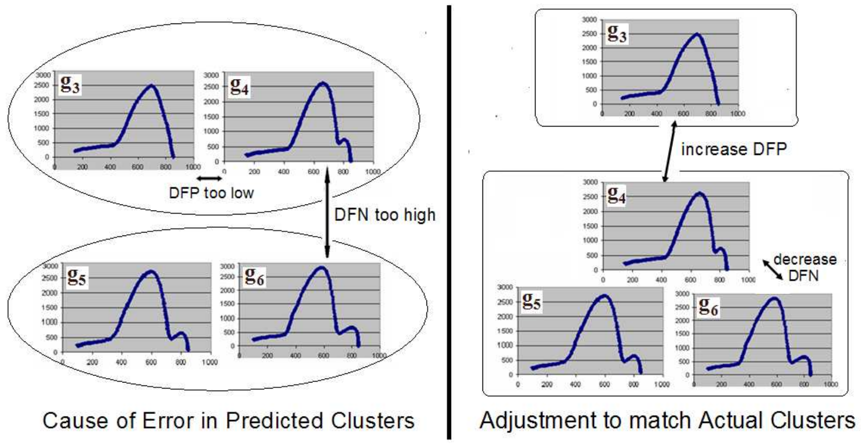

g3
g4
DFP too low
DFN too high
g5
g6
Cause of Error in Predicted Clusters
increase DFP
decrease DFN
Adjustment to match Actual Clusters

Figure 4.5: Distances used in Weight Adjustment

Given this notion of distances refer to Figure 4.5. Consider first the false negative pairs, 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 D the cause of the error is that the (average) distance DFN 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 but same predicted clusters, e.g., (g_3, g_4) in Figure 4.5, the cause of the error is that the (average) distance DFP is smaller than it should be. These distances are now used in altering weights using heuristics as follows.

Heuristics in Weight Adjustment

Consider the error due to the false negative pairs. To reduce this error it is desirable to decrease the distance DFN. In order to reduce DFN the weights of one or more components in the metric used to calculate the distance in the present epoch is decreased. For this we propose the FN Heuristic.

FN Heuristic Decrease the weights of the components in the metric D in proportion to their contributions to the distance DFN. That is, for each component:

New weight $w_i^{'} = w_i - \frac{DFN_i}{DFN}$

where DFN_i = DFN for component D_i alone.

Conversely, consider the FP pairs. To reduce their error we increase DFP. This is done by increasing the weights of one or more components in the metric using the FP Heuristic.

FP Heuristic Increase the weights of the components in the metric D in proportion to their contributions to the distance DFP. That is, for each component:

New weight $w_i^{''} = w_i + \frac{DFP_i}{DFP}$

where DFP_i = DFP for component D_i alone.

Combining these two we get the weight adjustment heuristic below.

Weight Adjustment Heuristic For each component D_i, its new weight is $w_i^{'''} = max(0, w_i - \frac{DFN_i}{DFN} + \frac{DFP_i}{DFP})$.

Thus, the new metric is: $D^{'''} = \Sigma_{i=1}^{m} w_i^{'''} D_i$

This new metric obtained after weight 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 this new metric.

As stated earlier, the final metric step is reached if the error is the below threshold or if the maximum number of epochs is reached. This step is explained below.

4.5.5 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.

Convergence:

LearnMet is not guaranteed to converge or to yield an optimal distance metric. However, thorough experimental evaluation in our application domain has consistently shown convergence to errors below the required threshold.

4.6 Algorithm for LearnMet

Given the overview and detailed discussion on the steps of the LearnMet technique, we now give the algorithm for LearnMet.

The LearnMet Algorithm Given: Training set with k actual clusters over G graphical plots, error threshold t, maximum number of epochs max, domain expert input on distance components D_i applicable to graphs

1). **Initial Metric Step**

  a. Assign each component D_i to D

  b. If relative importance of each D_i available then use *Initial Weight Heuristic* to assign each w_i

  c. Else assign a random w_i to each D_i

  d. Thus initialize $D = \Sigma_{i=1}^{m} w_i D_i$

2). **Clustering Step**

  a. Select arbitrary but fixed clustering algorithm

  b. Number of clusters = k (constant)

  c. Cluster plots using distance $D = \Sigma_{i=1}^{m} w_i D_i$

3). **Cluster Evaluation Step**

  a. Select *ppe* pairs of graphs

  b. Calculate TP, TN, FP, FN for *ppe* pairs

  c. Calculate failure rate $FR = (FP + FN)/(TP + TN + FP + FN)$

  d. If $(FR < t)$ or $(epoch == max)$ then go to 5. Final Metric Step

```
4). Weight Adjustment Step
    a. Calculate distances $DFN, DFP$
    b. Apply Weight Adjustment Heuristic to get new metric $D^{'''}$
    c. Go to 2(c) in Clustering Step using $D = D^{'''}$ as distance
5). Final Metric Step
    a. If $(FR < t)$ then return metric $D$
    b. Else find epoch with minimum failure rate $FR$
    c. Return corresponding metric $D$
```

4.7 Approach Enhancements

The LearnMet approach described in the algorithm above yields metrics that provide higher clustering accuracy than the default Euclidean distance [VRRMS0805]. This is elaborated in the section on evaluation. However, there is scope for further enhancement.

4.7.1 Goals in Enhancement

The three primary goals in enhancement are:

- *Quality:* This goal refers to improving the accuracy of the distance metrics in processes such as clustering.
- *Efficiency:* This involves reducing the number of epochs needed for convergence and hence the total training time.
- *Simplicity:* This deals with learning simple metrics that meet the requirements in the domain. Simplicity is concerned with the number of components in the distance metric.

The most significant among all these goals in the context of our problem is quality. This is because when the learned metrics are used as the distance in processes such as clustering, the resulting inferences should be as close as possible to the notion of correctness in the domain. This is to ensure that the inferences convey the right information for applications such as decision support in the domain.

The following approaches are used to meet one or more of the above goals:

1). *Selecting Pairs Per Epoch:* This refers to selecting a suitable number of pairs of plots in each epoch, denoted as pairs per epoch or *ppe*. This is to avoid overfitting and learn a generic hypothesis. This mainly aims to achieve the goal of quality. However it also impacts efficiency due to the lower execution time involved with the reduced number of pairs per epoch.

2). *Using Domain Knowledge in Weight Selection and Adjustment:* This deals with considering the semantics of the distance components in adjusting weights and intelligently guessing the initial weights so as to enhance the learning. This is in order to learn metrics closer to the notion of correctness in the domain and to converge faster. Thus it aims to meet the two goals of quality and efficiency.

3). *Learning Simple Metrics:* In this approach we apply the Occam's Razor principle [M97] in preferring simpler theories over complex ones. Thus we first consider metrics with a single component, then with two components, then three and so forth until convergence occurs or the training times out.

4.7.2 Selecting Pairs Per Epoch

The LearnMet approach by default considers the number of pairs per epoch, $ppe = G$ where G is the number of graphs in the training set [VRRMS0805]. The enhancement to LearnMet involves selecting a suitable number of pairs per epoch such that there are sufficient pairs in every epoch and yet enough randomization to avoid overfitting. This is addressed as follows.

Total Number of Pairs

Given that the number of graphs in the training set is G, the total number of pairs available for learning is $P = \frac{G!}{2!(G-2)!}$ [PNC99]. For example, if $G = 25$, $P = 300$. In every epoch of LearnMet, a random set of ppe pairs is selected for evaluation and weight adjustment. The total number of distinct combinations of ppe pairs that can be made from P pairs is given as $R = \frac{P!}{ppe!(P-ppe)!}$. While ppe denotes the amount of training data in each epoch, R denotes the extent of randomization. For example if $G = 25$ and $ppe = 15$, the total number of combinations available for learning is $R = 7.68 \times 10^{24}$. We now consider different ranges for ppe.

Low Range of ppe

In order to learn a hypothesis that does not overfit the training data, it is desirable to have more randomization. This would suggest using low ppe values, e.g., $ppe <= G$. These are likely to give a wider range of distinct combinations. However, consider for example an extreme of $ppe = 1$. If one distinct pair is used in each epoch for evaluation and weight adjustment, then it could happen that convergence to error below threshold occurs over the first few epochs. However the learned metric may only fit the few pairs that got considered in these epochs. The resulting hypothesis, namely, the learned metric would possibly not give high accuracy over unseen test data, since it is not generic. Likewise, ppe values higher than 1 but still in the low range are likely to yield a similar, though perhaps less serious problem of overfitting.

High Range of ppe

Consider the argument that a hypothesis is likely to be stronger if it is learned over a larger volume of data. This would suggest the other extreme, i.e., using ppe values closer to $P = \frac{G!}{2!(G-2)!}$. Mathematically this would yield a fairly large number of distinct combinations R. For example, consider $ppe = 285$. This gives $R = \frac{300!}{285!(300-285)!}$. Now consider $ppe = P - 285 = 15$. This gives $R = \frac{300!}{15!(300-15)!}$. Thus both $ppe = 285$ and $ppe - 15$ give $R = \frac{300!}{15!285!}$, thus mathematically producing the same number of distinct combinations, i.e., 7.68×10^{24}. However, there is a major difference between the two. For $ppe = 285$, there is the danger of the same pairs getting selected in each epoch, with only a few pairs distinct. For example, if pairs 2 through 286 are selected in one epoch, and pairs 3 to 287 in another, then only 2 pairs are distinct in these two epochs. Thus there is not really enough randomization which leads to the risk of overfitting. Moreover, with high ppe values, there is also a huge overhead in each epoch. An extreme case of this occurs for $ppe = P$, i.e., using all pairs per epoch. In this case, the exact same pairs would get selected in each epoch giving even less generality.

Middle Range of ppe

Let us now consider using ppe values close to $P/2$, i.e., half the total number of pairs in every epoch. This mathematically gives a large extent of randomization $R = \frac{P!}{(P/2)!(P/2)!}$. For example, for $P = 300$, this would yield $R = \frac{300!}{150!150!}$. Also this is likely to yield genuinely distinct combinations of pairs per epoch, since it does not consider almost the same the P pairs in each epoch. In addition, this gives a fairly large number of pairs in every epoch. Thus the learned metrics are likely to be more generic. This reduces the risk of overfitting. At the same time the overhead in each epoch is not as high as with ppe values close to P. For example, given $G = 25$, a good ppe value would be $ppe = 300/2 = 150$. In general it is thus advisable to select ppe values in the middle range, i.e., equal to or close to $P/2$. This is corroborated in the experimental evaluation section.

4.7.3 Using Domain Knowledge in Weight Selection and Adjustment

The basic LearnMet approach [VRRMS0805] proposes a weight adjustment heuristic that considers all individual metrics at par when assigning the blame for the error. This heuristic is based on the distance contribution of each component to the average false positive and false negative distances

DFP and DFN respectively. However, consider for example that a contribution of $DFN_i/DFN = 0.4$ for Euclidean distance may be more or less crucial than the same for Leidenfrost distance. It is thus desirable to incorporate the semantics of the components and scale the weight adjustment accordingly. Based on this, we now define the term scaling factor as follows.

Scaling Factor

A scaling factor for a distance component in the LearnMet metric is a number that determines the extent to which the weight of that component should be altered based on its semantics.

Scaling factors are calculated as follows. Consider a single component in the LearnMet metric used as the notion of distance in clustering. The greater the accuracy of the clusters given by that component alone, the greater is the significance of the component in the domain. This is because clustering accuracy in LearnMet is the extent to which the predicted clusters over a given data set match the notion of correctness depicted by actual clusters. Thus if a component alone used as the distance metric yields high clustering accuracy, it implies that this component by itself is a significant feature in preserving domain semantics. Now, if a component is more significant, then it is advisable to alter its weight to a greater extent in making adjustments. It is likely to give faster convergence and learn a more accurate metric. We therefore propose the following heuristic for scaling factors.

Scaling Factor Heuristic Assign a scaling factor to each component in the LearnMet metric directly proportional to the clustering accuracy obtained with that component alone used as the notion of distance.

With this discussion, we give the procedure for assigning scaling factors.

Scaling Factor Calculation

Scaling factors are calculated as follows.

```
Given: Training set with actual clusters of $G$ graphs, Set of Components applicable to them
1). For each component $D_i$
    a. Repeat $N$ times (where $N = A \times B$, such that $A =$ number of training sets and $B =$ number of clustering seeds)
        i. Perform clustering over training set using $D = D_i$
        ii. Calculate $TP, TN, FP, FN$ for all $P$ pairs
        iii. Calculate accuracy as Success Rate $SR_i = (TP + TN)/(TP + TN + FP + FN)$
    b. Calculate scaling factor $sf_i = (1/N)\Sigma_{j=1}^{N} SR_i$
2). Return scaling factors for all components
```

Scaled Weight Adjustment Heuristic

The Weight Adjustment Heuristic is now modified to incorporate scaling factors. We thus propose the Scaled Weight Adjustment Heuristic as follows.

Scaled Weight Adjustment Heuristic: For each component D_i, its new weight is:

$w_i^{'''} = max(0, w_i - sf_i \times \frac{DFN_i}{DFN} + sf_i \times \frac{DFP_i}{DFP})$.

Thus, the new metric is: $D^{'''} = \Sigma_{i=1}^{m} w_i^{'''} D_i$.

Initial Weight Selection

Scaling factors are likely to have a better impact on the learning if the weights in the initial metric are assigned according to the initial weight heuristic, i.e., based on the relative importance of each component. Then although learning the weight of a component with a low scaling factor may take longer, this effect is counterbalanced by the fast convergence of components with high scaling factors. Since the components with high scaling factors are more important, the overall convergence to the notion of correctness is likely to be quicker. Also the accuracy of the learned metric in clustering is likely to be higher since the weights of the more significant components have been scaled to emphasize their importance.

In the basic LearnMet approach [VRRMS0805], the relative importance of components is determined by the subjective notions of the domain experts and initial weights are assigned accordingly. However, if this relative importance is not known in advance then weights are assigned randomly. In the enhanced approach, we assign initial weights proportional to the accuracy of each individual component in clustering (accuracy is measured as the success rate SR using that component alone as explained in the Cluster Evaluation Step of LearnMet). Initial metrics thus selected exploit the power of scaling factors to an even greater extent, as corroborated experimentally. Thus the selection of initial weights and scaling factors both proportional to accuracy is likely to boost the performance of the algorithm.

4.7.4 Learning Simple Metrics

The Goal of Simplicity

This goal in the enhancement of LearnMet refers to learning a metric that is simple and yet meets the requirements in the domain. The main requirement in our targeted domains is that the learned metric should give high accuracy in clustering. The simplicity of the metric is measured in terms of two parameters, namely:

- *Number of components:* The fewer the components used in the metric, the simpler is the metric.
- *Amount of data for each component:* The less the amount of data needed to store a component, the simpler is that component and hence the corresponding metric.

It is to be noted that this notion of simplicity is subjective. Also, quality is more critical than simplicity for our goals. Thus a simple metric is preferred over a complex one only if both achieve the same quality.

The reasons for learning a simple metric are as follows.

1). Simple metrics are more efficient in terms of time complexity when used as the notion of distance in processes such as clustering.

2). Less storage space is required for simple metrics.

3). Experts cannot always identify components applicable to the plots. A brute force combination of components is not practical.

Principle in Learning

The main principle applied here is that of Occam's Razor which states that simpler theories are preferred over complex ones [M97]. In our case, the theory refers to the learned metric. Considering the two criteria of quality and simplicity, the process of learning, analogous to greedy search [M97] is outlined below.

Process of Learning Simple Metrics Given: Training set with actual clusters of plots; error threshold t

```
1). Identify all $m$ components $D_i : i = 1$ to $m$ in the domain
2). For each $D_i$
      a. Do clustering in LearnMet with $D = D_i$, get $FR$ and $SR$ (failure and success rates)
      b. If $(FR <= t)$ then set *final metric* = $D$ and go to step 5
3). If $(FR > t)$ then $m^{'} = 2$;
      • Repeat
          – Execute LearnMet with $D = \Sigma_{i=1}^{m^{'}} w_i D_i$ where $D_1 \ldots D_m^{'}$ are the $m^{'}$ components
            with highest clustering accuracies
          – If $(FR <= t)$ then set *final metric* = $D$ and go to step 4
          – Set $m^{'} = m^{'} + 1$
      • Until $m^{'} = m$
4). Output *final metric*
```

4.8 Evaluation of LearnMet

LearnMet has been rigorously evaluated in the Heat Treating domain [M95] that motivated this research. The source code used for experimentation is our LearnMet software tool [VRRMS1005]. This tool is developed in Java. We have implemented the k-means algorithm [M67] for clustering. The platform used for running these experiments has been a Mobile Intel Celeron (R) PC with a CPU Speed of 2 GHz, 192 MB of RAM and the Microsoft Windows XP Professional Version 2002 operating system. The general parameter settings in these experiments have been as follows.

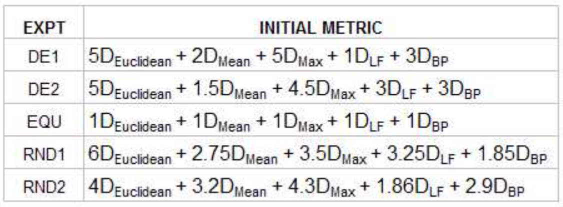
Domain experts have provided data sets of different sizes consisting of heat transfer curves placed in actual clusters. These have been used as training and test sets such that the training set for one experiment served as the test set for another. Thus training sets and test sets have been kept distinct in every experiment. We have used three different data set sizes, of $G = 15$, $G = 25$ and $G = 40$ graphs respectively. This has given $P = 105$, $P = 300$ and $P = 780$ pairs of graphs respectively. The maximum number of epochs has been set to 1000 [VRRMS1005] in all these experiments. The parameters altered have been the error thresholds, ppe values, scaling factors and initial metrics. The seeds in the clustering step of LearnMet have also been altered to provide randomization. Our experimental evaluation is presented below.

4.8.1 Effect of Initial Metrics and Error Thresholds

In these experiments the goal has been to observe the impact of general parameters such as error thresholds and initial metrics. The number of pairs per epoch has been maintained at the default value of $ppe = G$. The experiments shown below are for $G = 25$ giving $P = 300$, i.e., a training set of 25 graphs giving 300 pairs. Actual clusters over the training set have been provided by domain experts. The value of k (number of clusters) has been kept constant at $k = 5$ since this is the number of actual clusters. The test set consisted of 40 distinct graphs giving 780 pairs. Actual clusters over the test set are also provided by experts.

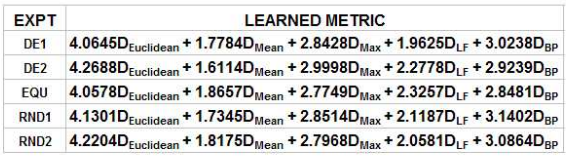
Effect of Initial Metrics

These experiments have been conducted to observe the impact of the initial metrics on the learning. Hence the other parameters have been maintained constant. Experts give an error threshold of 10 percent, i.e., 0.1 as acceptable in the domain for evaluation over test sets. As a default, we have used the same threshold for learning over the training set. The maximum number of epochs has been maintained at a constant value of 1000. The number of pairs per epoch, $ppe = G = 25$ for these experiments. Initial components in the metric have been given by experts. Two distinct assignments of initial weights have been given by two different experts. The corresponding two metrics are denoted by D_{DE1} and D_{DE2} respectively. A third initial metric D_{EQU} has been obtained by assigning equal weights to all components. Several experiments have been run by assigning random weights to components. We present here two such experiments with randomly generated metrics called D_{RND1} and D_{RND2}. Note that each experiment shown here represents the average of 4 experiments conducted using different seeds in the clustering algorithm. The initial metrics are shown in Figure 4.6 below. The learned metrics are shown in Figure 4.7. In these figures, $DE1$ denotes the experiment conducted with initial metric D_{DE1}, EQU denotes the experiment with initial metric D_{EQU} and so forth.



EXPT	INITIAL METRIC
DE1	$5D_{Euclidean} + 2D_{Mean} + 5D_{Max} + 1D_{LF} + 3D_{BP}$
DE2	$5D_{Euclidean} + 1.5D_{Mean} + 4.5D_{Max} + 3D_{LF} + 3D_{BP}$
EQU	$1D_{Euclidean} + 1D_{Mean} + 1D_{Max} + 1D_{LF} + 1D_{BP}$
RND1	$6D_{Euclidean} + 2.75D_{Mean} + 3.5D_{Max} + 3.25D_{LF} + 1.85D_{BP}$
RND2	$4D_{Euclidean} + 3.2D_{Mean} + 4.3D_{Max} + 1.86D_{LF} + 2.9D_{BP}$

Figure 4.6: Initial Metrics



EXPT	LEARNED METRIC
DE1	$4.0645D_{Euclidean} + 1.7784D_{Mean} + 2.8428D_{Max} + 1.9625D_{LF} + 3.0238D_{BP}$
DE2	$4.2688D_{Euclidean} + 1.6114D_{Mean} + 2.9998D_{Max} + 2.2778D_{LF} + 2.9239D_{BP}$
EQU	$4.0578D_{Euclidean} + 1.8657D_{Mean} + 2.7749D_{Max} + 2.3257D_{LF} + 2.8481D_{BP}$
RND1	$4.1301D_{Euclidean} + 1.7345D_{Mean} + 2.8514D_{Max} + 2.1187D_{LF} + 3.1402D_{BP}$
RND2	$4.2204D_{Euclidean} + 1.8175D_{Mean} + 2.7968D_{Max} + 2.0581D_{LF} + 3.0864D_{BP}$

Figure 4.7: Learned Metrics

Figures 4.8 through 4.12 depict the behavior of LearnMet during training. Experiments EQU, $RND1$ and $RND2$ take longer to converge than $DE1$ and $DE2$. However they all converge to approximately the same D.

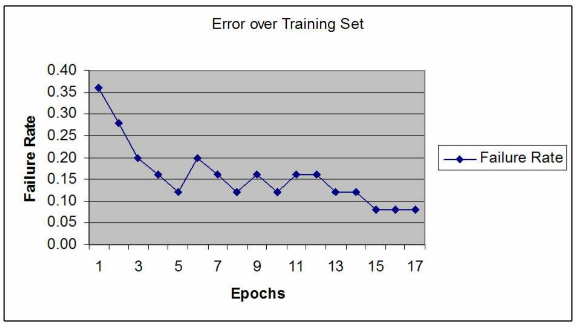


Figure 4.8: Training Behavior in Experiment $DE1$

The clustering accuracy of each of the learned metrics over a the test set is shown in Figure 4.13. The size of the distinct test set has been $G = 40$ graphs. The clustering accuracy as stated earlier has been measured by comparing clusters obtained from learned metrics with actual clusters

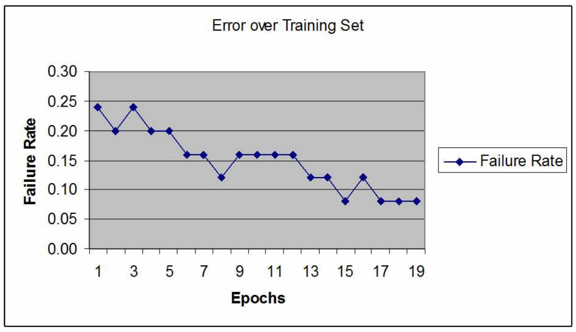


Figure 4.9: Training Behavior in Experiment $DE2$

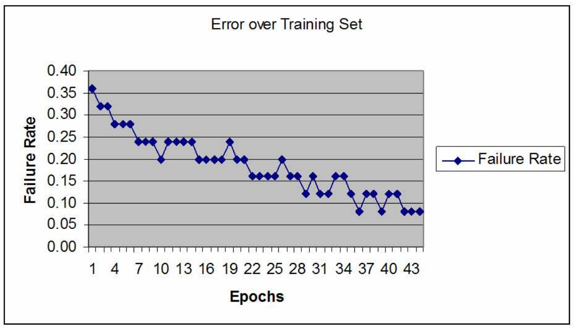


Figure 4.10: Training Behavior with Experiment EQU

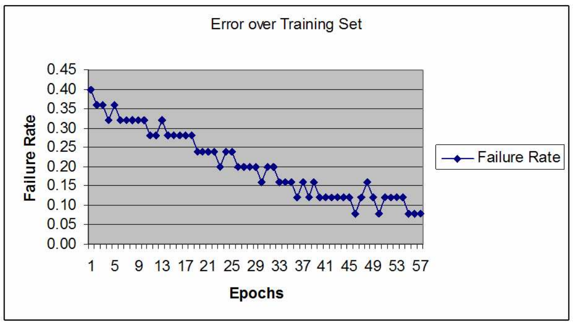


Figure 4.11: Training Behavior in Experiment $RND1$

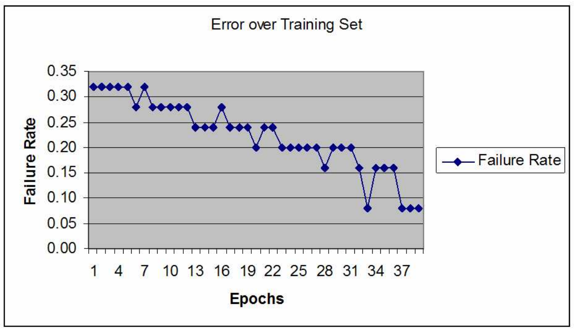


Figure 4.12: Training Behavior with Experiment $RND2$

over the test set. Comparative evaluation has also been performed using Euclidean distance (shown as ED in the figure). The learning efficiency in terms of the number of epochs for convergence is shown in Figure 4.14.

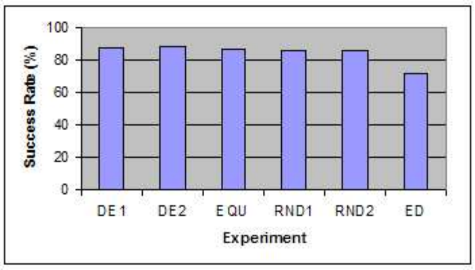


Figure 4.13: Clustering Accuracy with Different Learned Metrics and Euclidean Distance

Observations from experiments with different initial metrics

- Convergence to error below threshold occurs in all the experiments.
- The experiments with initial metrics provided by experts converge faster than the others.
- In each experiment the learned metrics have approximately the same

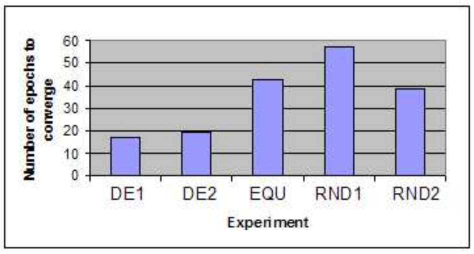


Figure 4.14: Learning Efficiency with Different Initial Metrics

weights, i.e., irrespective of the initial metrics the experiments converge to approximately the same learned metrics.

- Clustering accuracies of the learned metrics are higher than that of Euclidean distance.

Effect of the Error Threshold

The parameter of interest in these experiments has been the error threshold. The training and test sets used here have been the same as for the experiments on initial metrics. The impact of altering thresholds from 0.1 to 0.01 has been noted. Each experiment here represents the average of 4 experiments conducted using different initial metrics and altering the seeds in the clustering algorithm. The number of pairs per epoch has been maintained at a constant value of $ppe = G = 25$ in these experiments. All other parameters have also been constant, except the threshold, so that the effect of the threshold can be observed.

Figures 4.15 to 4.19 depict the behavior of LearnMet during training with different thresholds.

Figures 4.20 shows the clustering accuracy over the test set obtained

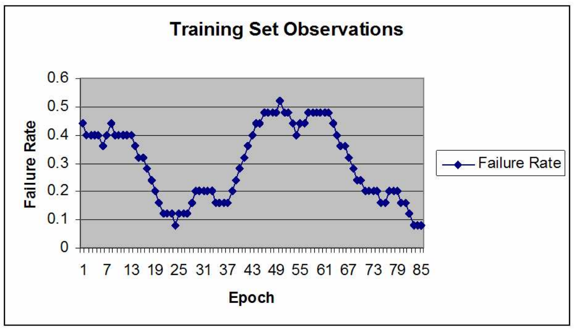


Figure 4.15: Training Behavior with Threshold = 0.1

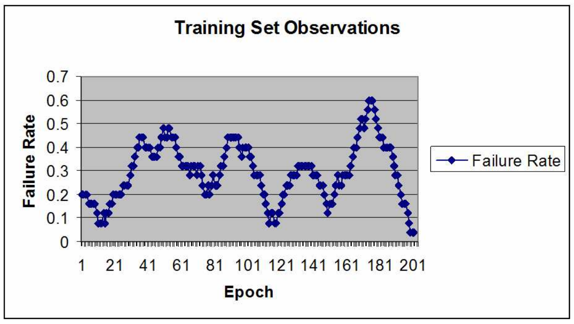


Figure 4.16: Training Behavior with Threshold = 0.075

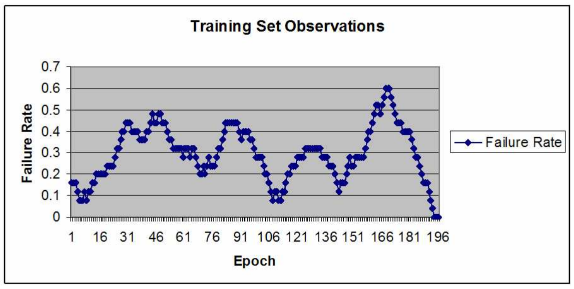


Figure 4.17: Training Behavior with Threshold = 0.05

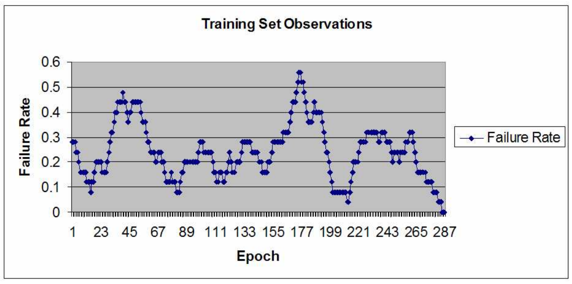


Figure 4.18: Training Behavior with Threshold = 0.025

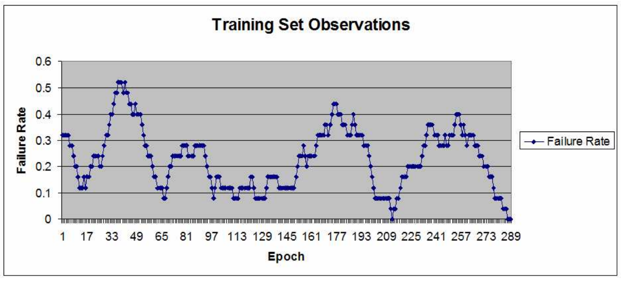


Figure 4.19: Training Behavior with Threshold = 0.01

with the learned metrics. Figure 4.21 shows the learning efficiency over the training set.

Observations from the experiments on varying error thresholds

1). Clustering with the learned metrics gives higher accuracy than clustering with Euclidean distance (as observed from earlier experiments).

2). As the error threshold is reduced, the number of epochs to converge tends to increase. Thus the learning efficiency reduces with a decrease in threshold.

3). With reduced thresholds however, the clustering accuracy over the test set is higher. As the threshold is reduced from 0.1 to 0.01, accu-

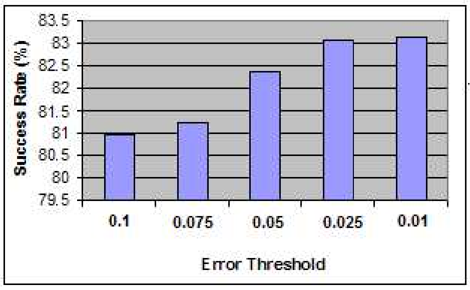


Figure 4.20: Clustering Accuracy with Varying Thresholds

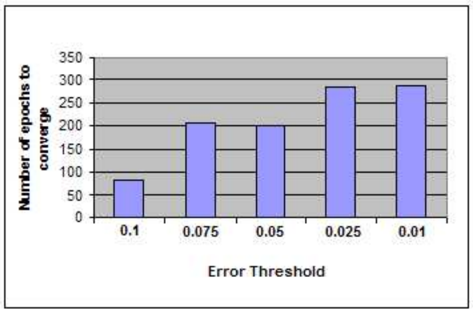


Figure 4.21: Learning Efficiency with Varying Thresholds

racy increases from approximately 81 to 83 percent.

4). Since accuracy is more critical for our goals it is advisable to select lower error thresholds to getter better quality results. Thus for the remaining experiments we use an error threshold of 0.01 throughout.

4.8.2 Effect of Parameters after Enhancement

In these experiments the parameters of interest have been the number of pairs per epoch, the scaling factors and the components in the metric. Each of these has been considered separately.

Effect of the Number of Pairs Per Epoch

The following experiments have been conducted to observe the impact of the number of pairs per epoch ppe on the learning. The parameters varied in these experiments have been the ppe values and seeds in clustering. The observations shown here are for number of graphs $G = 25$, number of clusters $k = 5$, error threshold constant at 0.01, maximum number of epochs set to 1000 and initial metric $D = 5D_{Euclidean} + 4D_{Mean} + 3D_{Max} + 2D_{LF} + 1D_{BP}$. Each experiment here is the average of 4 experiments with different initial weights and clustering seeds. Figures 4.22 through 4.33 show the training set observations for different ppe values. The failure rate is plotted versus the epoch showing the behavior during training.

The learned metrics have been used for clustering over a distinct test set. The test set size used here is $G = 40$. The clustering accuracy over the test set with the different learned metrics is shown in Figure 4.34. The learn-

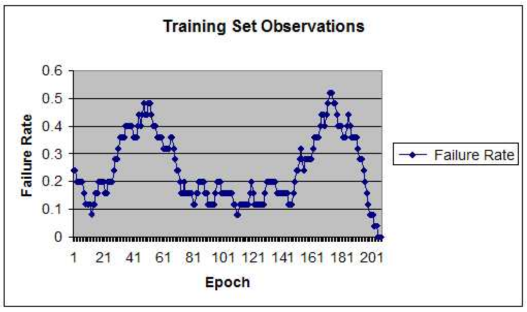


Figure 4.22: Training Behavior with $ppe = 25$

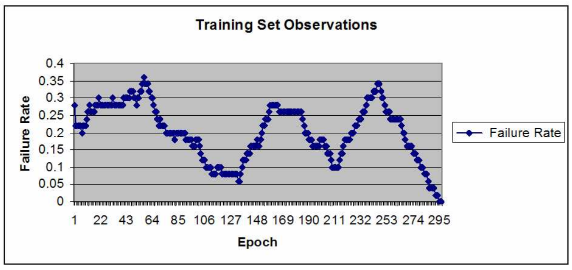


Figure 4.23: Training Behavior with $ppe = 50$

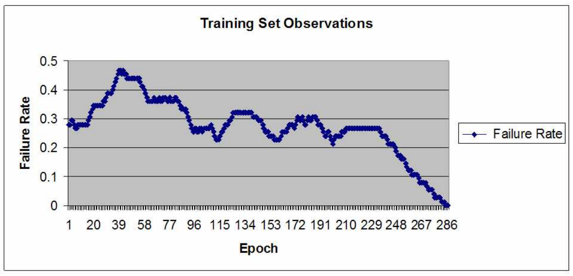


Figure 4.24: Training Behavior with $ppe = 75$

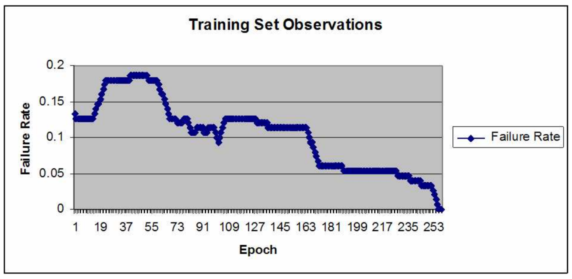


Figure 4.25: Training Behavior with $ppe = 100$

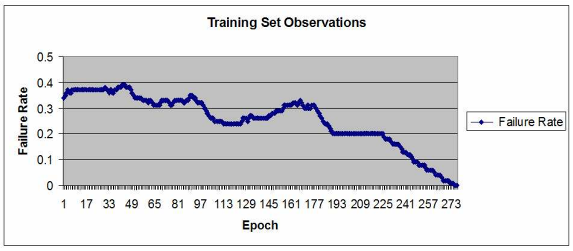


Figure 4.26: Training Behavior with $ppe = 125$

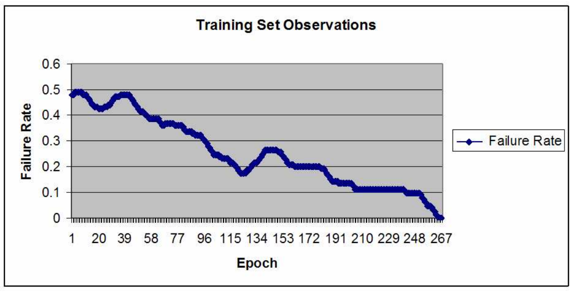


Figure 4.27: Training Behavior with $ppe = 150$

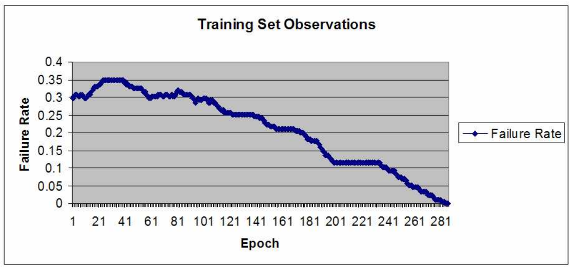


Figure 4.28: Training Behavior with $ppe = 175$

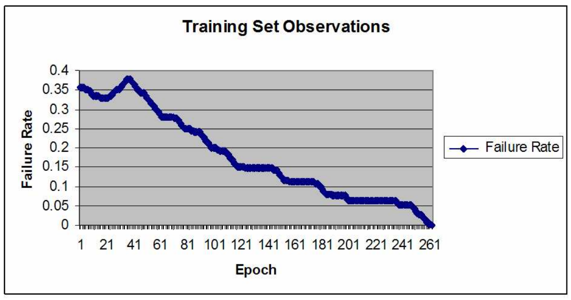


Figure 4.29: Training Behavior with $ppe = 200$

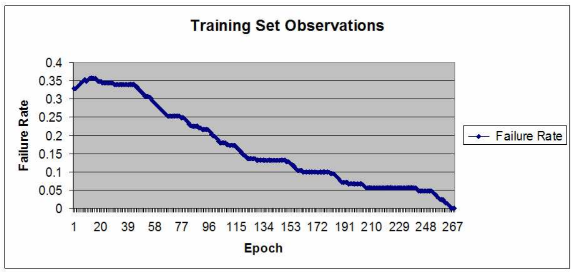


Figure 4.30: Training Behavior with $ppe = 225$

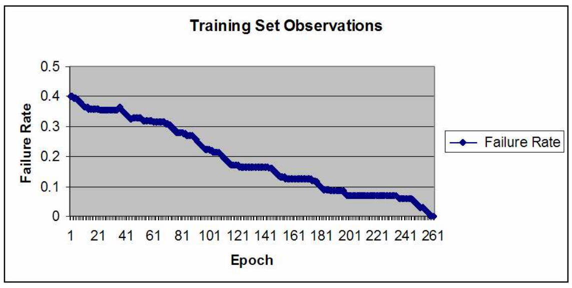


Figure 4.31: Training Behavior with $ppe = 250$

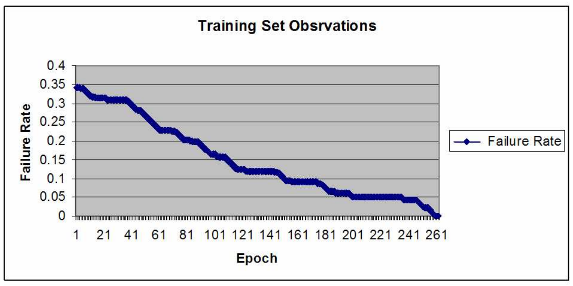


Figure 4.32: Training Behavior with $ppe = 275$

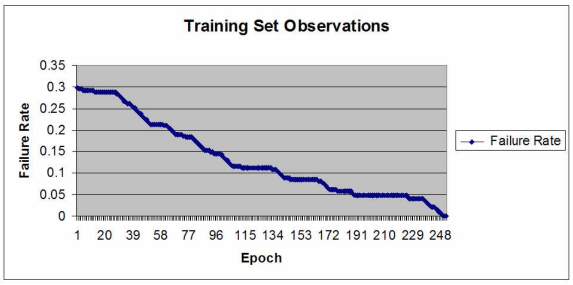


Figure 4.33: Training Behavior with $ppe = 300$

ing efficiency over the training set is also recorded in terms of the training time and the number of epochs to converge. This is shown in Figure 4.35. In addition, in Figure 4.36 a comparison between training behavior is presented for different ranges of ppe. In this figure the failure rate has been plotted versus the epoch for three different ppe values.

Observations from the experiments on pairs per epoch

1). Failure rate decreases monotonously for high ppe values but oscillates for lower ppe values. This is 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 val-

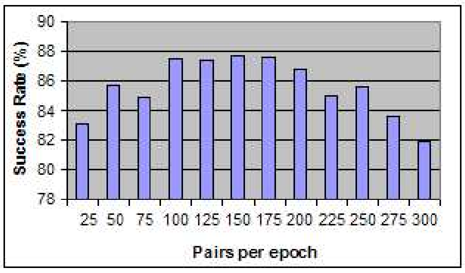


Figure 4.34: Clustering Accuracy with *ppe* Values

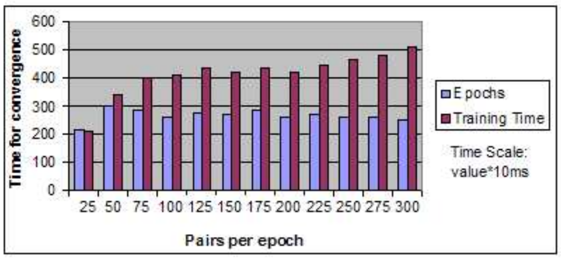


Figure 4.35: Learning Efficiency with *ppe* Values

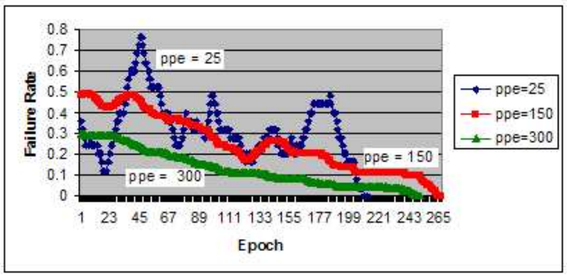


Figure 4.36: Training Behavior with *ppe* Values

ues, almost the same pairs get selected in each epoch, thus causing a uniform decrease in failure rate.

2). Low ppe values, such as $ppe < G$ may converge faster but the learned metrics give relatively lower clustering accuracy over the test set. Also some experiments with low ppe values may take as long to converge as ppe close to G. This depends on which ppe pairs get randomly selected for evaluation and weight adjustment.

3). Middle range ppe values take longer to converge than low range but give the best clustering accuracy over the test set.

4). High ppe values close to P, (i.e, all pairs) take still longer to converge and give less accuracy over test set than middle range values.

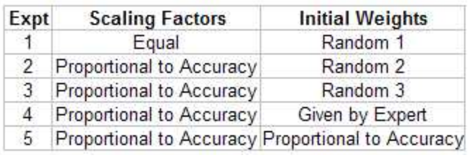
5). The setting used for further experiments is $ppe = P/2$, i.e., half the total number of pairs. This is because it is found to give best accuracy over test set while still giving acceptable efficiency.

Effect of Scaling Factors

The experiments below have been conducted to observe the impact of scaling factors (sf) on the learning. The experiments shown here are for $G = 25$ and $k = 5$. The distance components used have been the D_{SC}, $D_{Euclidean}$, D_{Min}, D_{Mean}, D_{Max}, D_{LF} and D_{BP} distances.

The number of pairs per epoch has been maintained at $ppe = P/2 = 150$ for these experiments, based on the results from the ppe experiments. The parameters varied have been the initial weights and scaling factors. Figure

4.37 summarizes the parameter settings used for the experiments shown here. The seeds in clustering are also altered to provide randomization and each experiment represents the average of 4 experiments with different seeds. Figures 4.38 through 4.42 show the training set observations in terms of the failure versus the epoch.



Expt	Scaling Factors	Initial Weights
1	Equal	Random 1
2	Proportional to Accuracy	Random 2
3	Proportional to Accuracy	Random 3
4	Proportional to Accuracy	Given by Expert
5	Proportional to Accuracy	Proportional to Accuracy

Figure 4.37: Parameter Settings in Experiments on Scaling Factors

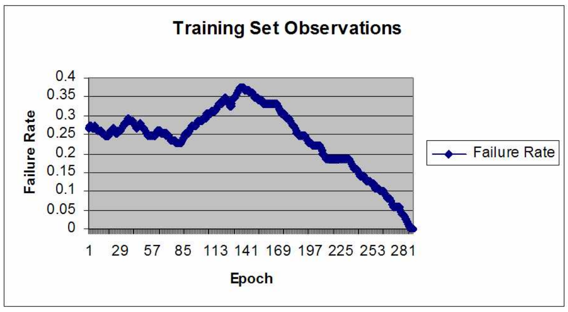


Figure 4.38: Training Behavior for Experiment 1 on Scaling Factors

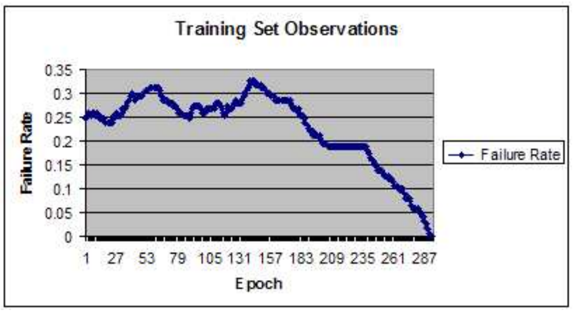


Figure 4.39: Training Behavior for Experiment 2 on Scaling Factors

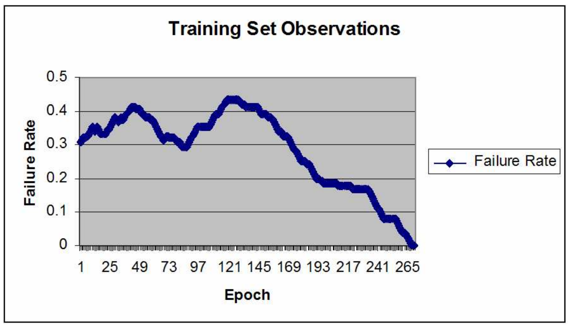


Figure 4.40: Training Behavior for Experiment 3 on Scaling Factors

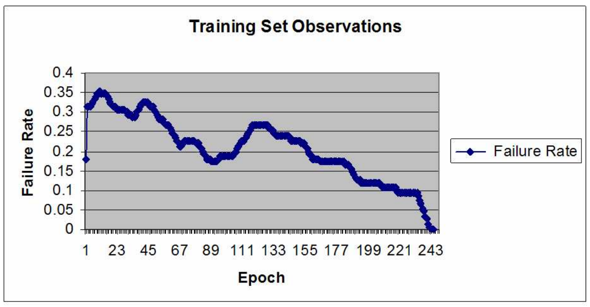


Figure 4.41: Training Behavior for Experiment 4 on Scaling Factors

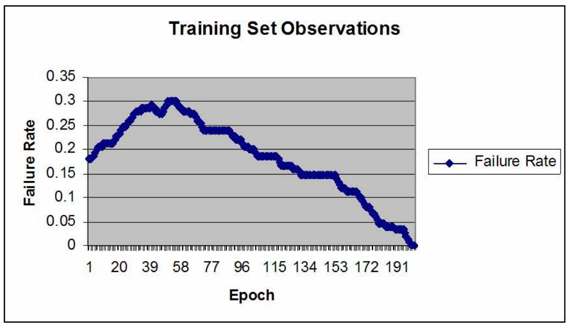


Figure 4.42: Training Behavior for Experiment 5 on Scaling Factors

The learned metrics are used for clustering over a distinct test set. The test set shown here is of size $G = 40$. Figure 4.43 shows the clustering accuracy over the test set. Figure 4.44 shows the learning efficiency over the training set.

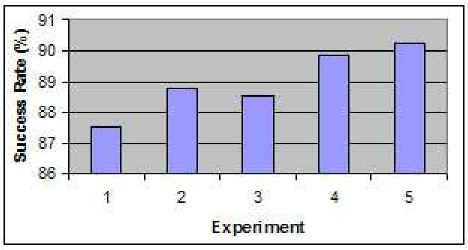


Figure 4.43: Clustering Accuracy in Experiments on Scaling Factors

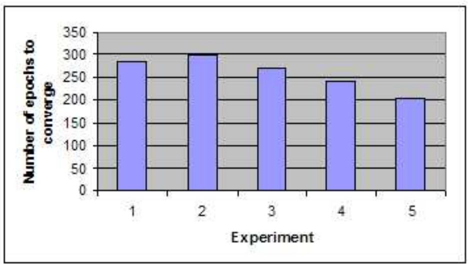


Figure 4.44: Learning Efficiency in Experiments on Scaling Factors

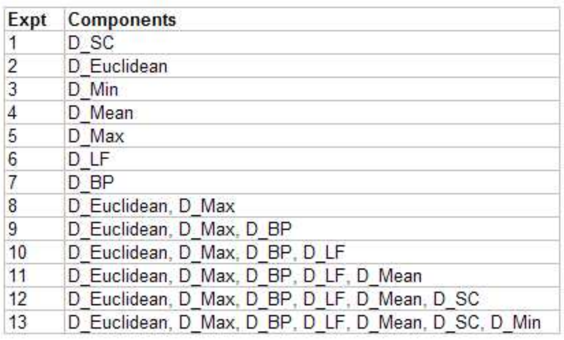
Observations from the experiments on scaling factors

1). With scaling factors proportional to the accuracy of each individual component and with random initial metrics, the convergence may occur faster or slower than with no scaling factors. This appears to depend on the initial metric.

2). With initial metrics provided by domain experts and with scaling factors proportional to accuracy, the results observed are better than those with random metrics.

3). Even better results are observed for initial metrics and scaling factors both proportional to accuracy of each individual component in clustering.

Effect of Components



Expt	Components
1	D_SC
2	D_Euclidean
3	D_Min
4	D_Mean
5	D_Max
6	D_LF
7	D_BP
8	D_Euclidean, D_Max
9	D_Euclidean, D_Max, D_BP
10	D_Euclidean, D_Max, D_BP, D_LF
11	D_Euclidean, D_Max, D_BP, D_LF, D_Mean
12	D_Euclidean, D_Max, D_BP, D_LF, D_Mean, D_SC
13	D_Euclidean, D_Max, D_BP, D_LF, D_Mean, D_SC, D_Min

Figure 4.45: Metrics used in Experiments on Components

These experiments have been conducted with the goal of learning simple metrics. The aim has been to learn metrics that have few components and yet achieve clustering accuracy acceptable in the domain. The experiments below show the impact of altering the components in the distance metric. The setup for the experiments shown here has involved the following fixed parameters. The number of graphs in the training set has been $G = 25$ from which $P = 300$ pairs of plots have been obtained. The number of clusters has been $k = 5$ from the actual clusters over the training set. The number of pairs per epoch has been maintained at $ppe = P/2 = 150$, since this has been found to be a good setting from previous experiments. The error threshold has been maintained at 0.01 and maximum number of epochs at 1000. The test set used has been of size $G = 40$.

The parameters altered have been the following. The number of components has been altered in each experiment. The possible components identified in the domain have been the individual metrics D_{SC}, $D_{Euclidean}$, D_{Mean}, D_{Min} D_{Max}, D_{LF} and D_{BP}. Figure 4.45 shows the components used in each experiment.

The results are summarized below for each experiment on components. Note that for those experiments involving only a single component, there are no weights involved. Hence the only observations are the failure and success rates, i.e., error and accuracy respectively. For the experiments with multiple components, the initial and final weights are shown. Also the number of epochs required for learning are recorded. Note that for the experiments with multiple components, the order of selection is based on the accuracy of each individual component in clustering. Thus we select the best two, then best three and so forth until all components are considered.

Results of Experiments on Components

- Experiment 1: Slow Cooling Distance alone
  - Metric: D_{SC}
  - Error (Failure Rate): 0.76
  - Accuracy (Success Rate): 0.24
- Experiment 2: Euclidean Distance alone
  - Metric: $D_{Euclidean}$

  - Error (Failure Rate): 0.2633
  - Accuracy (Success Rate): 0.7367

- Experiment 3: Minimum Distance alone
  - Metric: D_{Min}
  - Error (Failure Rate): 0.7733
  - Accuracy (Success Rate): 0.2267

- Experiment 4: Mean Distance alone
  - Metric: D_{Mean}
  - Error (Failure Rate): 0.5
  - Accuracy (Success Rate): 0.5

- Experiment 5: Maximum Distance alone
  - Metric: D_{Max}
  - Error (Failure Rate): 0.3267
  - Accuracy (Success Rate): 0.6734

- Experiment 6: Leidenfrost Distance alone
  - Metric: D_{LF}
  - Error (Failure Rate): 0.4133
  - Accuracy (Success Rate): 0.5867

- Experiment 7: Boiling Point Distance alone
  - Metric: D_{BP}
  - Error (Failure Rate): 0.3633
  - Accuracy (Success Rate): 0.6367

- Experiment 8: Euclidean and Maximum Distances
  - Initial Metric: $7.367D_{Euclidean} + 6.733D_{Max}$
  - Convergence: No

- Minimum Error (Failure Rate): 0.2267
- Number of Epochs for Learning: 1000
- Final Metric: $6.5449 D_{Euclidean} + 4.5927 D_{Max}$
- Training Time: 3425.0 milliseconds

- Experiment 9: Euclidean, Maximum and Boiling Point Distances
  - Initial Metric: $7.367 D_{Euclidean} + 6.733 D_{Max} + 6.367 D_{BP}$
  - Convergence: No
  - Minimum Error (Failure Rate): 0.1133
  - Number of Epochs for Learning: 1000
  - Final Metric: $6.4669 D_{Euclidean}$ $+4.3651 D_{Max}$ $+4.0522 D_{BP}$
  - Training Time: 3135.0 milliseconds

- Experiment 10: Euclidean, Maximum, Boiling Point and Leidenfrost Distances
  - Initial Metric: $7.367 D_{Euclidean} + 6.733 D_{Max} + 6.367 D_{BP} + 5.867 D_{LF}$
  - Convergence: Yes
  - Minimum Error (Failure Rate): 0.01
  - Number of Epochs for Convergence: 375
  - Final Metric: $5.6070 D_{Euclidean}$ $+4.4870 D_{Max}$ $+3.4299 D_{BP}$ $+2.8589 D_{LF}$
  - Training Time: 4426.0 milliseconds

- Experiment 11: Euclidean, Maximum, Boiling Point, Leidenfrost and Mean Distances
  - Initial Metric: $7.367 D_{Euclidean}$ $+6.733 D_{Max}$ $+6.367 D_{BP}$ $+5.867 D_{LF}$ $+5 D_{Mean}$
  - Convergence: Yes
  - Minimum Error (Failure Rate): 0.01
  - Number of Epochs for Convergence: 267
  - Final Metric: $4.8605 D_{Euclidean}$ $+3.1034 D_{Max}$ $+3.2270 D_{BP}$ $+2.5293 D_{LF}$ $+2.1589 D_{Mean}$
  - Training Time: 4216.0 milliseconds

- Experiment 12: Euclidean, Maximum, Boiling Point, Leidenfrost, Mean and Slow Cooling Distances

  - Initial Metric: $7.367D_{Euclidean} + 6.733D_{Max} + 6.367D_{BP} + 5.867D_{LF} + 5D_{Mean} + 2.4D_{SC}$
  - Convergence: Yes
  - Minimum Error (Failure Rate): 0.01
  - Number of Epochs for Convergence: 263
  - Final Metric: $4.7147D_{Euclidean} + 3.4635D_{Max} + 3.1183D_{BP} + 2.3700D_{LF} + 2.3119D_{Mean} + 0.2479D_{SC}$
  - Training Time: 4165.0 milliseconds

- Experiment 13: Euclidean, Maximum, Boiling Point, Leidenfrost, Mean, Slow Cooling and Minimum Distances
  - Initial Metric: $7.367D_{Euclidean} + 6.733D_{Max} + 6.367D_{BP} + 5.867D_{LF} + 5D_{Mean} + 2.4D_{SC} + 2.267D_{Min}$
  - Convergence: Yes
  - Minimum Error (Failure Rate): 0.01
  - Number of Epochs for Convergence: 272
  - Final Metric: $4.8456D_{Euclidean} + 3.4834D_{Max} + 3.5536D_{BP} + 2.4744D_{LF} + 2.5584D_{Mean} + 0.1505D_{SC} + 0.2104D_{Min}$
  - Training Time: 4243.0 milliseconds

The learned metrics in each experiment are used for clustering over the test set and the clustering accuracy is recorded. The clustering accuracy obtained with different learned metrics is shown in Figure 4.46.

Observations from experiments on components

- Among the individual metrics, Euclidean distance gives highest accuracy.
- The combination of Euclidean, Maximum, Leidenfrost and Boiling Point distances in Experiment 10 gives error below threshold.

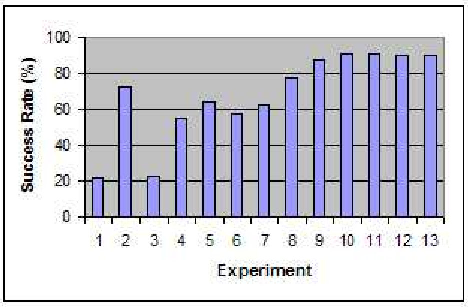


Figure 4.46: Clustering Accuracy with Different Components

- Further increasing the number of components does not increase accuracy.

- For some of the experiments on components, convergence to error below threshold does not occur, implying that the selected components are not sufficient for representing the graphs, irrespective of their weights. Thus the components need to be selected such that they depict the features needed to distinguish the graphs. Then the learning of weights can occur so as to achieve high clustering accuracy. Thus LearnMet also serves to identify the bare minimum components needed to represent the graphs, in the situation where the components are not provided in advance by domain experts.

- The simplest learned metric in these experiments has four components, $D_{Euclidean}$, D_{Max}, D_{LF} and D_{BP}. This gives accuracy approximately in the range if 90 percent, with settings of $ppe = P/2$, (i.e., half the total number of pairs), and both scaling factors as well as initial metrics proportional to accuracy of each component in clustering.

4.9 Evaluation of AutoDomainMine Stage 2: Intermediate Stage

Stage 2 of AutoDomainMine incorporates the LearnMet technique for learning domain-specific distance metrics for clustering graphs. Since LearnMet has been developed with the aim of capturing the semantics of the graphs through the domain-specific distance metrics, the learned metrics have been used as the notion of distance in clustering. This is with the intention of producing better clusters. The resulting clusters have been used for the classification step in AutoDomainMine. The estimation obtained using the classifiers obtained from the improved clusters is thus likely to be better than the earlier estimation in Stage 1, i.e., the Pilot Tool. This is because Stage 1 involved the default notion of Euclidean distance in clustering as opposed to Stage 2 which used the learned metrics that preserve domain semantics. AutoDomainMine has thus been evaluated in Stage 2 by measuring the accuracy of the estimation using clusters obtained with and without the learned metrics. The estimation obtained from clustering using the learned metrics has been compared with that from clustering using Euclidean distance. The difference in estimation accuracy represents the effectiveness of the metrics obtained from LearnMet. This process of evaluation is described below.

4.9.1 Process of Evaluating AutoDomainMine with LearnMet

The metric obtained from LearnMet has been assessed by measuring the accuracy of the estimation in AutoDomainMine. Data used for evaluation is from all the performed experiments has been stored in the database. Hence the domain experts did not need to provide a separate test set. The opinion of the experts however has been considered in evaluation.

The evaluation approach has been as follows. As a first step, clustering has been done with the metric obtained from LearnMet. The metric has been learned over the training set and then used for clustering all the data in the database. In the second step, the clustering output has been sent to a decision tree classifier in the required format, namely, input conditions and cluster of each graph. The results of the classification have been used as the basis for estimation. The accuracy of the estimation has been evaluated using n-fold cross-validation (cv) [WF00]. For example, in 4-fold-cv, in each fold, 75 percent of the clustering output has been used for building the tree and the remaining 25 percent has been used as new experiments whose cluster is to be predicted given the input conditions. If the correct cluster has been predicted then the estimation has been considered accurate else inaccurate. Percentage accuracy has been reported accordingly. This process is illustrated in Figures 4.47 and 4.48 respectively.

The estimation obtained from clustering using the learned metrics has been compared with that from clustering using Euclidean distance. Another criterion for comparison involved clustering using Euclidean distance. The resulting clusters have been sent to decision tree classifiers and the estimation accuracy has been observed. The observations for all the metrics are recorded.

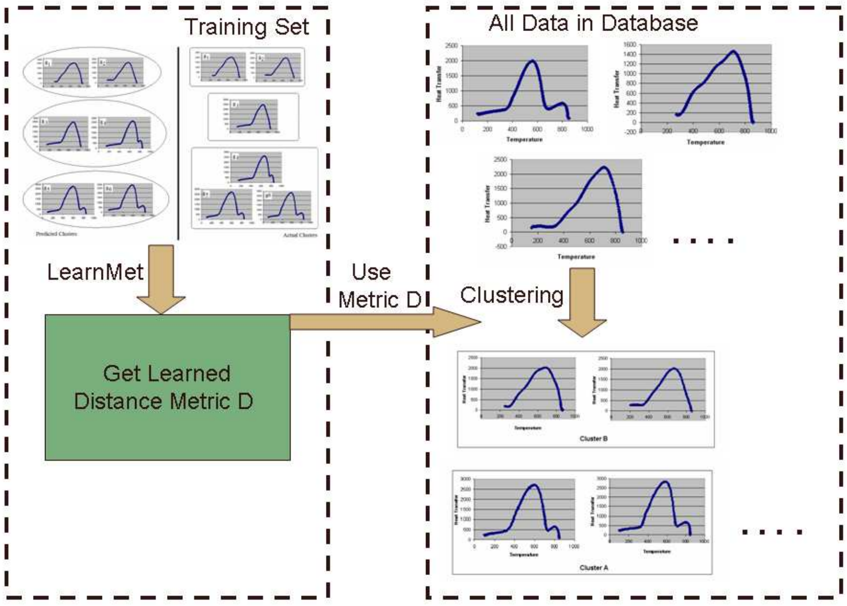
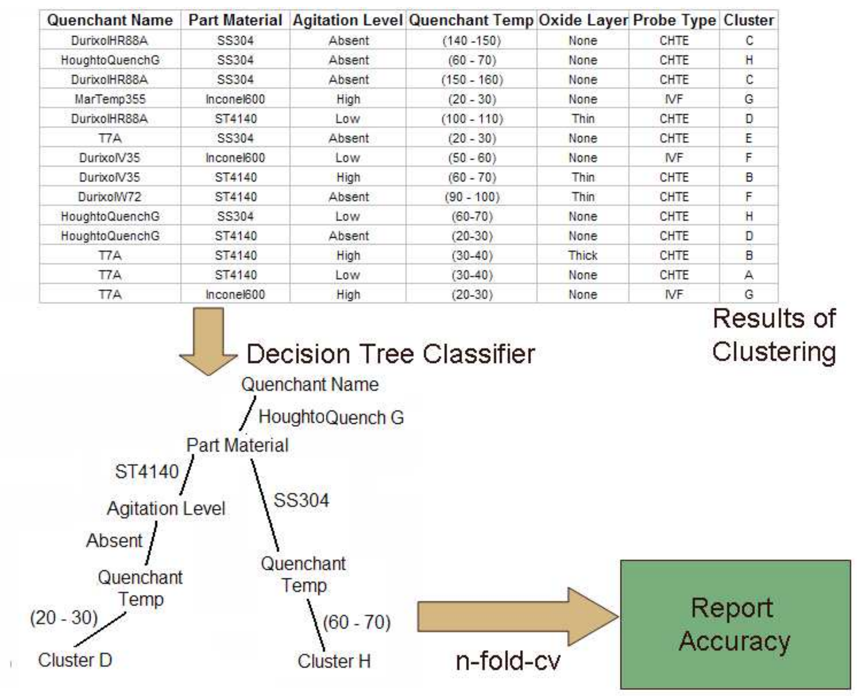


Figure 4.47: Process of Evaluating AutoDomainMine with LearnMet: Clustering Step



Quenchant Name	Part Material	Agitation Level	Quenchant Temp	Oxide Layer	Probe Type	Cluster
DurixolHR88A	SS304	Absent	(140 -150)	None	CHTE	C
HoughtoQuenchG	SS304	Absent	(60 - 70)	None	CHTE	H
DurixolHR88A	SS304	Absent	(150 - 160)	None	CHTE	C
MarTemp355	Inconel600	High	(20 - 30)	None	IVF	G
DurixolHR88A	ST4140	Low	(100 - 110)	Thin	CHTE	D
T7A	SS304	Absent	(20 - 30)	None	CHTE	E
DurixolV35	Inconel600	Low	(50 - 60)	None	IVF	F
DurixolV35	ST4140	High	(60 - 70)	Thin	CHTE	B
DurixolW72	ST4140	Absent	(90 - 100)	Thin	CHTE	F
HoughtoQuenchG	SS304	Low	(60-70)	None	CHTE	H
HoughtoQuenchG	ST4140	Absent	(20-30)	None	CHTE	D
T7A	ST4140	High	(30-40)	Thick	CHTE	B
T7A	ST4140	Low	(30-40)	None	CHTE	A
T7A	Inconel600	High	(20-30)	None	IVF	G

Figure 4.48: Process of Evaluating AutoDomainMine with LearnMet: Classification Step

4.9.2 Results of Evaluating AutoDomainMine with LearnMet

The results of evaluating LearnMet with AutoDomainMine are presented here. The metrics learned from the LearnMet experiments in section 4.8 have been used for clustering and the corresponding estimation accuracy has been recorded. For convenience we have used the same subtitles as used in Section 4.8 for the respective experiments.

Effect of Initial Metrics

We first consider the experiments on the effect of initial metrics in LearnMet. These are experiments $DE1$, $DE2$, EQU, $RND1$, $RND2$ for initial metrics and experiments on thresholds. The respective learned metrics have been used for clustering in AutoDomainMine. Figure 4.49 shows the estimation accuracy with learned metrics output from from experiments $DE1$ through $RND2$.

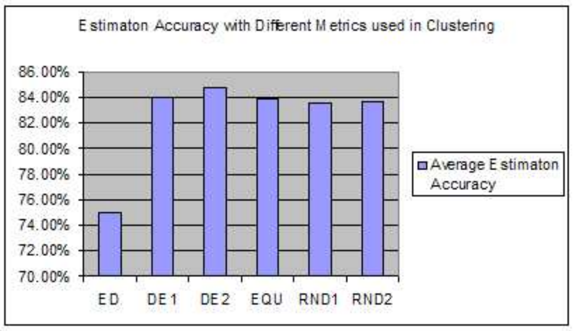


Figure 4.49: AutoDomainMine Estimation with the output of LearnMet Initial Metric Experiments

Observations from estimation with experiments on initial metrics

- The estimation accuracy with each metric output from LearnMet is higher than that with Euclidean distance.
- Estimation accuracy with metrics from experiments $DE1$ and $DE2$ is higher than the others.
- The highest estimation accuracy with these experiments is approximately 84.5 percent.

Effect of the Error Threshold

We now consider the experiments on the effect of the error threshold in LearnMet. The respective learned metrics have been used for clustering in AutoDomainMine. Figure 4.50 shows the estimation accuracy with these metrics output from LearnMet.

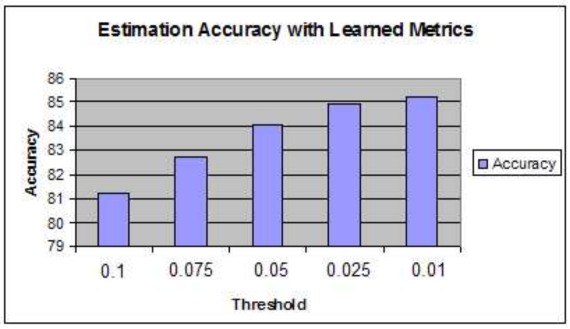


Figure 4.50: AutoDomainMine Estimation with the output of LearnMet Threshold Experiments

Observations from estimation with experiments on thresholds

- The estimation accuracy with these experiments on the whole is slightly higher than with the experiments on initial metrics.

- Estimation accuracy increases as the error threshold reduces.
- The highest estimation accuracy observed in these experiments is approximately 85.1 percent.

Effect of the Number of Pairs Per Epoch

Next we consider the experiments on the effect of the number of pairs per epoch (*ppe*) in LearnMet. The respective learned metrics have been used for clustering in AutoDomainMine. Figure 4.51 shows the estimation accuracy with the metrics output from the LearnMet *ppe* experiments.

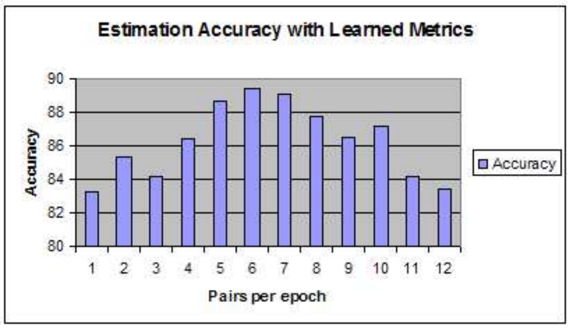


Figure 4.51: AutoDomainMine Estimation with output of LearnMet *ppe* Experiments

Observations from estimation with experiments on pairs per epoch

- The estimation accuracy with these experiments on the whole is higher than with the experiments on initial metrics and thresholds.
- Estimation accuracy is higher with middle range *ppe* values.

- Lower estimation accuracy is observed for *ppe* values close to the two extremes of high and low.
- The highest estimation accuracy observed in these experiments is approximately 89.5 percent.

Effect of Scaling Factors

We now use the output of the experiments on the effect of scaling factors in LearnMet. The respective learned metrics have been used for clustering in AutoDomainMine. Figure 4.52 shows the estimation accuracy with the metrics output from the LearnMet scaling factor experiments.

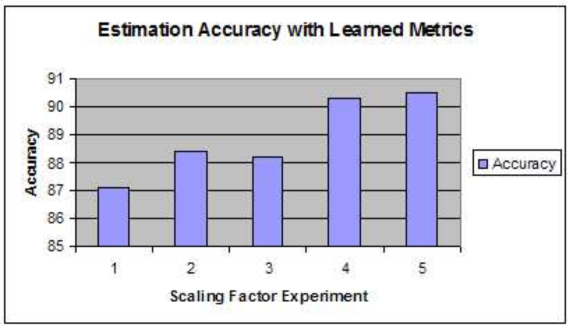


Figure 4.52: AutoDomainMine Estimation with output of LearnMet Scaling Factor Experiments

Observations from estimation with experiments on scaling factors

- The estimation accuracy with these experiments on the whole is higher than with the experiments on initial metrics and thresholds.
- Estimation accuracy is higher with those experiments where initial weights are proportional to scaling factors.

- Lower estimation accuracy is observed for experiment 1 with random initial weights and no scaling factors.
- The highest estimation accuracy observed in these experiments is approximately 90.5 percent.

Effect of Components

We then use the output of the experiments on the effect of scaling factors in LearnMet. The respective learned metrics have been used for clustering in AutoDomainMine. Figure 4.53 shows the estimation accuracy with the metrics output from the LearnMet scaling factor experiments.

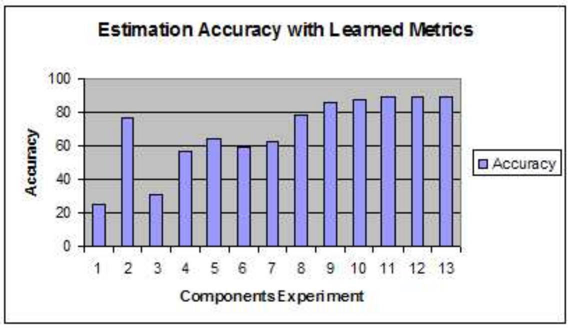


Figure 4.53: AutoDomainMine Estimation with LearnMet Metrics from Experiments on Components

Observations from estimation with experiments on components

- Estimation accuracy with individual components is lower than those with a combination of components.
- Among the individual components, Euclidean distance gives the highest estimation accuracy.

- The experiment using components $D_{Euclidean}$, D_{Max}, D_{LF}, D_{BP} and D_{Mean}, i.e., experiment 10 gives estimation accuracy approximately 90.

- Further increasing the number of components, i.e., adding D_{SC} and D_{Min} does not substantially increase the estimation accuracy.

4.9.3 Conclusions from the Evaluation of Stage 2 of AutoDomainMine

It is observed that the estimation accuracy in the experiments in Stage 2, i.e., the Intermediate Stage of AutoDomainMine is higher than in Stage 1, i.e., the Pilot Tool. In Stage 2, we get an estimation accuracy of approximately 86% to 87% on an average while in Stage 1, the estimation accuracy is approximately 74% to 75% on an average. This proves the effectiveness of the LearnMet approach in learning domain-specific distance metrics to preserve the semantics in the graphs. Thus Stage 2 of AutoDomainMine that incorporates the LearnMet approach provides better performance than Stage 1 with the pilot AutoDomainMine approach. Thus the effectiveness of AutoDomainMine as an estimation technique is increased in Stage 2.

Chapter 5

DesRept: Designing Semantics-Preserving Representatives for Clusters

5.1 Need for Designing Representatives

The pilot and intermediate stages of AutoDomainMine use a randomly selected set of input conditions and graph from each cluster as its representative pair. These representatives form the basis for estimation. However, a randomly selected representative may not adequately represent the cluster. Distinct combinations of input conditions could lead to a single cluster and graphs in a cluster could have some variations of ranges. Moreover, different applications may need different levels of detail in the cluster. For example, presenting all the information in the cluster causes inefficiency in certain applications such as simulation tools [LVKR02]. In other applications such as visual displays for parameter selection [MCMMS02] avoiding clutter on the screen is important. Hence it is advisable to design cluster representatives that preserve domain semantics in the context of targeted applications.

5.1.1 Motivating Example

We present a motivating example in the context of cluster representatives for conditions. Consider the sets of conditions S_1 through S_9 in Example 1 showing the input conditions in a given cluster.

Example 1

- S_1: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (70-80), Probe Type = CHTE
- S_2: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (80-90), Probe Type = CHTE
- S_3: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = High, Oxide Layer = Any, Quenchant Temperature = (50-60), Probe Type = CHTE
- S_4: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Low, Oxide Layer = None, Quenchant Temperature =

(60-70), Probe Type = CHTE

- S_5: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (20-30), Probe Type = CHTE
- S_6: Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = Any, Oxide Layer = Thin, Quenchant Temperature = (60-70), Probe Type = CHTE
- S_7: Quenchant Name = DurixolW72, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (60-70), Probe Type = CHTE
- S_8: Quenchant Name = MarTemp355, Part Material = SS304, Agitation Level = High, Oxide Layer = None, Quenchant Temperature = (30-40) C, Probe Type = CHTE
- S_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 lead to a similar experimental output, hence they have been assigned to the same cluster.

We now consider the application of simulation tools [LVKR02]. 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, then 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 the two extremes in such applications. However, 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 2000 to 2300 Watt per meter squared Kelvin, the Slow Cooling could occur between 200 to 250 degrees Celsius 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 take into account the interests of various users.

Thus there is a need to design semantics-preserving cluster representatives in the context of targeted applications.

5.1.2 Goals of the Design Process

The goals for designing cluster representatives are stated as follows.

- Given: Clusters of graphs in the domain with input conditions characterizing each cluster.
- Design: A representative pair per cluster embodying the semantics of the domain and comprising of:
  - Set of input conditions to represent the cluster.
  - Graph to represent the cluster.

Before explaining the details of the proposed approach, we review related work in the area.

5.2 Related Work

We overview related work applicable to the design of representatives for graphs and conditions in our problem. In addition, we review work on evaluating clusters and visual displays. We also review work on similarity measures for data analogous to the input conditions in this dissertation. Related work on similarity measures for graphs has already been discussed in Chapter 4.

5.2.1 Designing Representatives

Medoid Approach

In [BKKPV03] they address the problem of similarity search in database systems by visualizing the hierarchical clustering structure of a database of objects to speed up the similarity search. They consider the use of reachability plots to extract the significant clusters in a hierarchical cluster representation along with suitable cluster representatives. The reachability plot is a visualization of the clustering hierarchy that enables each object to be assigned to its closest cluster. The concept of reachability plots is based on a binary relation called reachability which is the minimum distance of each database object to one of its predecessors in the ordering. The reachability values are plotted for each object, and the cluster ordering is such that it minimizes the reachability values. The representatives are then constructed using a medoid-approach.

However, their constraint is that the representative must be a real object of the data set. Hence they consider the medoid of a cluster as the closest object to the mean or centroid of all the objects in the cluster. In our problem, we do not have the constraint that the representative must be an object of the cluster, thus giving us the freedom to consider alternative design strategies.

Image Rating

In [HH01] they build representatives for web information. They have user-interfaces for web-based applications. These interfaces need to facilitate access of relevant information without displaying unnecessary detail. Their interfaces involve both text as well as image data. The text is organized in hierarchies of web-page titles or URL taxonomies. Image representation involves choosing a single image from a group of images to adequately summarize the group. An example is when a single image needs to be chosen from a web page containing a group of images. They use an approach of image rating. Images are rated based on aspects of 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. Also they employ manual selection of representatives by groups of targeted users.

Their methods involve considerable user-intervention in building the representatives which is not desirable in our problem. Moreover in our work, the quality of all individual graphs in the cluster is the same in terms of color, clarity and so forth.

Common Sub-structures

In [FGOT03], the problem of document clustering is addressed. They cluster structurally similar Web documents, especially XML documents. Their goal is the analysis and management of Web data. They build cluster representatives as follows. Their notion of distance is based on edit distance between XML trees, namely, the minimum-cost sequence of operations to convert one tree into another. Using this distance, they consider the common sub-structures between XML documents within a cluster by XML tree matching, and then add all the uncommon ones to it by tree merging. They then prune the merge tree by removing the least frequent nodes. This serves as the representative of the given cluster. We can draw some analogy here when we are building a representative of conditions using a decision tree, i.e., retaining the top levels of the tree as ”common sub-structures”.

However, in our context, we need as a representative a complete set of conditions leading to a cluster. Thus using only part of the tree as a common substructure and then adding the uncommon ones does not seem practical. Instead we need to consider complete paths and then compute their distances from each other, defining a suitable distance function as needed.

5.2.2 Evaluating Clusters

Clusters of Text

Nomoto et. al. [NM01] propose an approach for text summarization based on exploiting diversity concepts in text. They propose an information-centric approach for evaluation where the text summaries are judged in terms of how well they represent their source documents in processes such as document retrieval and text categorization. They use the Minimum Description Length (MDL) principle to determine the number of clusters needed for text summarization. If the MDL encoding with two clusters is lower than that with greater than two clusters, then two clusters are used. This concept is extended to multiple clusters. Their MDL encoding takes into account probability of occurrence of words, the number of parameters and the number of data objects. However, they do not construct and evaluate different types of representatives.

Clusters of Association Rules

In the ARCS system [LSW97], clusters of association rules are constructed, based on the similarity of the left-hand side of the association rule. They consider error in terms of incorrectly clustered tuples with respect to a set of optimal clusters provided. Given this, the MDL encoding is as follows. The greater the number of clusters, the higher is the cost to describe them. The cost of encoding the sampled data using a given set of clusters is defined to be the sum of all errors for all clusters. If a tuple is not an error, then it is identified by a particular cluster and hence its cost is included within the description of the cluster. If a tuple is in error, it must be specifically identified as such and this incurs a cost. The goal in the ARCS system is to minimize this MDL cost. However they do not consider the cost of encoding the association rules themselves. In our context, to evaluate clusters we need to take into account the manner of storing all the graphs and input conditions in the clusters as well as the corresponding representatives. Nor do they evaluate different types of representatives.

Clustering to aid Classification

In [BL04] the MDL principle is applied to evaluate how a clustering algorithm run on data aids a classification algorithm. They propose a type of MDL bound with the notion that clustering algorithms with a small bound must have a small number of clusters which agree with a set of hidden labels. They consider a distribution over (X, Y) where X is the input and Y is the label. The assumption is that Y can take one of L possible values, such that $L > 1$. Their MDL encoding takes into account the number of clusters, the algorithm used for clustering, the number of random initializations of the algorithm and the number of hidden labels. Given this, they choose the encoding that minimizes the description length so that the clustering algorithm aids a classification algorithm. However, in their context, hidden labels are already provided, which is not true for our problem. Moreover, they need to evaluate a set of algorithms whereas in our context we consider an arbitrary but fixed clustering algorithm. Also, they do not construct and evaluate cluster representatives. In their work, it is essential to recover the original cluster from the encoding, which is not a requirement in our problem.

5.2.3 Visual Displays

Semantic Textual Units

Personal Digital Assistants (PDAs) often have displays with information displayed in levels of detail. In [BMP01] such an approach is described that involves the summarization of text from the web on handheld PDAs. They partition the original web page into "Semantic Textual Units" (STUs) which are fragments of pages such as paragraphs, lists, tables etc. They then consider various methods of displaying the STU. In the incremental approach, each STU is revealed gradually in terms of the first line, the first few lines and then the whole STU. In the keywords approach, the important keywords that appear within each STU are displayed. In the summary approach, the most significant sentence of each STU is displayed.

We can draw an analogy here in terms of displaying information in levels of detail in the context of designing representatives for conditions in our problem. However, keywords and significant sentences are not applicable in our context since we deal with input conditions of experiments. Moreover in our context, the relative importance of the input conditions needs to be conveyed when displaying the information. Also a complete set of input conditions needs to be displayed to the user for each estimation. Moreover, they do not propose objective evaluation measures for comparison between their STUs.

Semantic Fish Eye Views

Users often search information from a general to a specific level of detail. Janecek et. al. [JP03] address the problem of facilitating such searches based on the concept of "Semantic Fish Eye Views" (SFEVs). An SFEV consists of a collection of several objects in a small space. It visually emphasizes more interesting information and filters less important information. The degree of interest is calculated based on the apriori interest which reflects its relevance to a user query. Thus the apriori interest establishes the global context in which the user searches. Although the SFEV is an interesting form of display in searching information over the Web, in our context the only analogy is the emphasis on more interesting information such as the features of graphs.

We discuss this in the context of designing representatives for graphs in our context. It is essential in our work to see the representative as the output of an estimation. Hence conveying too many individual graphs as an output leads to too much generality. Some of this would be on the lines of simple query processing where all the images pertaining to a query are returned. Since our problem is estimation, we need to provide a single object (simple or complex) as an approximately correct answer. Moreover, in [JP03] no objective measures are proposed to evaluate the SFEVs.

5.2.4 Similarity Measures

Similarity of Complex Categorical Attributes

Das et. al. [DMR98, DM00] define similarity between categorical attributes based on inter-dependencies in datasets. They consider similarity depending not only on the values of the given attributes but also based on the values of other attributes that are inter-dependent. In [DMR98] they explain, for example, that in market basket data notions of similarity can be inferred from the buying tendencies of customers. Two products can be considered similar if the buying habits of customers are similar. They thus define similarity based on internal and external measures where an internal measure depends only on the values of the attributes while external measures depend on the data in other columns, which they call the probe columns. They use these similarity measures to build hierarchies using association rule mining algorithms [DMR98].

In [DM00] they present the Iterated Contextual Distances (ICD) algorithm to learn distances between attributes taking into account such interdependencies. The ICD algorithm starts with an arbitrary distance function between attributes which is used to derive a vector representation for rows which gives a vector representation for subrelations. The value of the subrelation distance function is used to get a new distance value for attributes. Starting with random initial values, the ICD algorithm converges to stable distances between attributes. From these, stable distances between rows and subrelations are defined.

However, the kind of inter-dependencies that they define do not exist in our datasets. The similarity between our attributes such as Quenchant Name, Part Material etc. do not depend on user interests. Hence we do not need to learn such similarity measures. Instead in our context, user interests are significant in displaying information in terms of factors such as levels of detail, information loss, ease of interpretation etc. and we take these into account in designing representatives.

Similarity of Strings in Text Documents

Learnable similarity measures for strings are presented in [BM03] based on support vector machines and expectation maximization. These measures are applied for duplicate detection. However, 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 to define a distance function for the conditions without further learning.

5.3 Proposed Approach: DesRept

We propose a methodology called DesRept to design semantics-preserving cluster representatives. Each cluster representative is a pair consisting of two parts, namely, a set of input conditions and a graph.

The inputs to DesRept are the clusters of graphs in the domain and the sets of conditions characterizing each cluster. Thus, the clusters from the clustering step and decision trees resulting from them in AutoDomainMine serve as the input to DesRept. The output of DesRept is a representative pair consisting of a set of input conditions and a graph for each cluster.

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 two parts are dealt with separately in DesRept. However, the general principles behind the two parts are the same. Hence the basic methodology of designing representative pairs in DesRept is summarized below.

The DesRept Methodology

1). Input: Clusters of graphs, decision tree paths of conditions leading to 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 Mininum Description Length principle.

5). For each cluster return the winner, i.e., representative of conditions/graph with the lowest encoding as the designed representative.

6). Output a representative pair of input conditions and graph for each cluster.

Note that we use the term *clusters of conditions* to refer to the sets of input conditions leading to the respective clusters of graphs.

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 in the next two subsections respectively.

5.4 DesCond: Designing and Evaluating Representatives for Conditions

We propose an approach called DesCond to design a representative set of input conditions for each cluster. We first define/refresh the following terminology.

- Input Condition: This refers to an individual process parameter input to an experiment. Each condition is defined by an attribute value pair, e.g., *Part Material = ST4140*.

- Attribute: This gives name of each condition, e.g., *Part Material*.

- Value: This gives the content of each condition, e.g., *ST4140*.
- Set of Conditions: This refers to all the input conditions in a given experiment, e.g., *Quenchant Name = DurixolV35, Part Material = ST4140, Agitation Level = low, Oxide Layer = none, Quenchant Temperature = (60-70)*. These are also referred to as the *conditions* or *input conditions* for an experiment.

The semantics of the domain is captured by defining a suitable a distance function for the set of conditions. Using this notion of distance, candidate representatives are designed for each cluster showing gradually increasing levels of detail.

In the first level, the representative is designed by selecting a set of conditions from the original cluster such that it forms the nearest neighbor for all other objects in the cluster. This candidate is called the *nearest representative*.

At the second level, a candidate known as the *summarized representative* is designed by forming sub-clusters within each original cluster using domain knowledge and the given notion of distance.

In the third level, the candidate is constructed by combining all information in the cluster and abstracting it in a suitable form. This candidate is called the *combined representative*.

Thus, the process of designing the nearest representative is *guided selection* in which the representative is an object belonging to the original cluster. On the other hand, the process of designing the summarized and combined representatives is *construction* where the representative is developed using the information within the cluster.

The candidates are compared using a measure called the *DesCond Encoding* analogous to the Minimum Description Length principle [M97]. This encoding takes into account the complexity of each representative measured as the number of data points stored for it and the information loss due to it measured as its distance from other objects in the cluster. The interests of targeted users based on the relative importance attached to the complexity and information loss are also taken into account in the encoding. The candidate giving the lowest value in the encoding is the winner and is returned as the designed representative. Note that there could be multiple winners based on the encoding, reflecting the corresponding user interests.

Thus, in our framework, the three main tasks in the design of domain-specific cluster representatives for conditions are as follows:

1). Defining a notion of distance for the set of conditions.

2). Obtaining candidate cluster representatives showing different levels of detail each capturing domain semantics.

3). Proposing an encoding to compare the candidates in order to find a suitable winner meeting specific application requirements.

These tasks are discussed in the following three subsections.

5.4.1 Notion of Distance

We consider three criteria in defining distance. The first is the data type of each attribute as applicable to the domain. The second criterion is the distance between the individual attribute values defined in a domain-specific manner. The third one is the weight of each attribute based on its relative importance in the domain. These are explained as follows.

Data Types of the Attributes

The attributes describing the input conditions are of different types such as numeric, categorical and ordinal [HK01]. Categorical attributes are of the character or string type and store descriptive information. Numeric attributes represent data that is of the integer or real number type. Ordinal attributes are those whose values are also of the string and character type but store information where the order matters.

The types of attributes applicable to the Heat Treating datasets in our problem are listed below. For ordinal attributes, their possible values are also stated. Each attribute represents an individual input condition in Heat Treating.

- Quenchant Name (QN): Categorical
- Part Material (PM): Categorical
- Probe Type (PT): Categorical
- Oxide Layer (OL): Ordinal [None, Thin, Thick, Any]
- Agitation Level (AL): Ordinal [Absent, Low, High, Any]
- Quenchant Temperature (QT): Numeric

Distance between the Attribute Values

We use the sets of conditions shown in Example 1 in Section 5.1.1 in order to explain the calculation of distance for each type of attribute.

Categorical Attributes. For categorical attributes, the distance is considered to be 0 if the attribute values are identical and 1 if they are not identical. Hence the distance is calculated as:

$D_{Categorical}(S_i, S_j) = 0$ if $v_i = v_j$ and $D_{Categorical}(S_i, S_j) = 1$ if $v_i <> v_j$ where S_i and S_j are the respective sets of conditions, while v_i and v_j are the respective values of the given categorical attribute.

Thus, considering the categorical attribute Part Material and referring to Example 1, we calculate distance between the Part Material values as $D_{PM}(S_1, S_3) = 1$, and $D_{PM}(S_1, S_2) = 0$, since Part Material values are not equal in the sets of conditions S_1 and S_3, while they are equal in S_1 and S_2.

Numeric Attributes. For numeric attributes, distance is calculated as the absolute difference of their attribute values. If the values are grouped into ranges as a data pre-processing step, then we consider the difference between the mean values of the respective ranges. Suitable scaling factors are applied if needed to maintain parity with other attributes. Thus, distance for numeric attributes is calculated as:

$D_{Numeric}(S_i, S_j) = SF \times |v_i - v_j|$ where S_i and S_j are the respective sets of conditions, v_i and v_j are the values (or mean value of ranges) of the respective numeric attributes, and SF is a scaling factor based on domain knowledge.

Thus in Example 1, for the numeric attribute Quenchant Temperature with scaling factor $SF = 1/10$ (given in the domain) we get distances between Quenchant Temperature values as $D_{QT}(S_1, S_2) = 1$ and $D_{QT}(S_1, S_3) = 2$.

Ordinal Attributes. For ordinal attributes, the distance is calculated as the absolute difference between their values after they are mapped to numeric based on their order. 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 zero from all other values. This mapping is a data preprocessing step. Distance for ordinal attributes is then given as:

$D_{Ordinal}((S_i, S_j) = |v_i^{'} - v_j^{'}|$ where S_i and S_j are the respective sets of conditions, while $v_i^{'}$ and $v_j^{'}$ are numeric values to which the respective ordinal values are mapped.

In Example 1 therefore, for the ordinal attribute Agitation Level, distance is calculated as $D_{AL}(S_1, S_2) = 3 - 3 = 0$ and $D_{AL}(S_3, S_4) = 3 - 2 = 1$.

Distance for Set of Conditions Given these distances for the attribute types, the distance function D_{cond} for the set of conditions is then defined in terms of the distances between individual attribute values and the weights of the respective attributes as follows:

$D_{cond} = \Sigma_{i=1}^{A} W_i \times D_i$ where each D_i is a distance function for the individual attributes, each W_i is a weight giving the relative importance of the corresponding attribute and A is the total number of attributes. The weights are obtained as explained in the next subsection.

Weights of the Attributes

As stated earlier, in our problem decision trees [Q86] are used to learn the relative importance of the conditions characterizing each cluster with respect to the domain. Hence the decision tree paths are used to derive the weights of the attributes depicting these conditions. The reasoning behind the method for deriving the weights is as follows.

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 Quenchant Name should get a higher weight than in a path having other attributes such as Part Material and Agitation Level.

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 the decision tree induction algorithms such as ID3 and J4.8 [Q86] in this reasoning. It is not feasible to directly use the weights from these algorithms, because the weights are different in each epoch and we need one uniform set of weights for the attributes. Moreover, if we were to use their weights we would need to define a constant of proportionality which is not known apriori. Also, running the ID3/J4.8 epochs again on the same dataset is likely to be inefficient, given that the tree has already been constructed. Thus, we use the analogy behind the induction of decision trees.

Given these considerations and applying the reasoning above, a heuristic for the weights of the attributes in the decision tree is defined below.

Decision Tree Weight Heuristic $W_i = \frac{1}{P}\Sigma_{j=1}^{P}\frac{H_{i,j}}{H_j} \times G_j$

where, W_i = weight of each attribute,

P = total number of paths in the decision tree,

G_j = number of graphs in the cluster of path j,

$H_{i,j}$ = height of node for attribute i in path j and

H_j = height of path j

such that, "height" H is defined number of nodes away from the leaf.

Thus, in a given path the leaf has a height of 0, the node immediately above the leaf has a height of 1 and so forth. The height of a path is basically the height of its root node.

The use of the decision tree heuristic in calculating weights is explained in Example 2 using the partial decision tree shown in Figure 5.1.

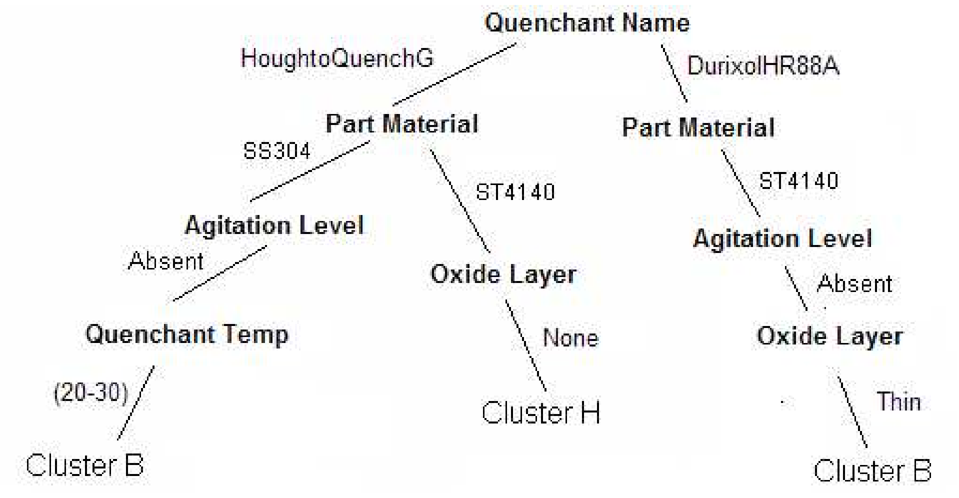


Figure 5.1: Partial Decision Tree

Example 2 For the given partial decision tree, assume that Cluster B has 10 experiments and Cluster H has 5 experiments. Then we get the following weights.

Quenchant Name: $W_{QN} = \frac{1}{3}(\frac{4}{4} \times 10 + \frac{3}{3} \times 5 + \frac{4}{4} \times 10) = 8.33$

Part Material: $W_{PM} = \frac{1}{3}(\frac{3}{4} \times 10 + \frac{2}{3} \times 5 + \frac{3}{4} \times 10) = 5.44$

Agitation Level: $W_{AL} = \frac{1}{3}(\frac{2}{4} \times 10 + 0 + \frac{2}{4} \times 10) = 3.33$

Oxide Layer: $W_{OL} = \frac{1}{3}(0 + \frac{1}{3} \times 5 + \frac{1}{4} \times 10) = 1.39$

Quenchant Temperature: $W_{QT} = \frac{1}{3}(\frac{1}{4} \times 10 + 0 + 0) = 0.83$

Probe Type: $W_{QT} = \frac{1}{3}(0 + 0 + 0) = 0$

We will use the weights derived from the corresponding *complete* decision tree in this example in order to illustrate the design of the candidate cluster representatives. The weights of the attributes inferred from the *complete* tree (whose partial snapshot is shown in Figure 5.1) are as follows.

- Quenchant Name (QN): weight W_{QN} = 8.12
- Part Material (PM): weight W_{PM} = 5.97
- Agitation Level (AL) : weight W_{AL} = 3.05
- Oxide Layer (OL): weight W_{OL} = 2.08
- Quenchant Temperature (QT): weight W_{QT} = 0.81
- Probe Type (PT): weight W_{PT} = 0

Thus the distance function derived from the complete tree is:

$D_{cond} = 8.12 \times D_{QN} + 5.97 \times W_{PM} + 3.05 \times D_{AL} + 2.08 \times D_{OL} + 0.81 \times D_{QT}$

where the individual distances D_{QN}, D_{PM} and so forth are calculated based on the values and types of the individual attributes. Given the manner in which it is derived, this distance function incorporates domain semantics and can be used for the design of candidate cluster representatives.

5.4.2 Levels of Detail

We consider the following levels of detail in designing the candidate representatives.

- **Level 1: Nearest Representative.** This is also known as the *Single Conditions Representative* and is one set of conditions closest to all others in the cluster using the giving notion of distance that incorporates domain semantics.

- **Level 2: Summarized Representative.** This is also called the *Multiple Conditions* and involves multiple sets of conditions summarizing cluster information through sub-clusters built using the same notion of distance.

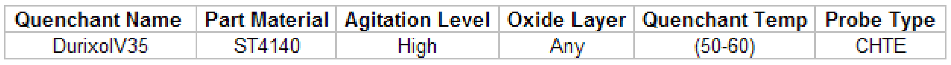
- **Level 3: Combined Representative.** This is also referred to as the *All Conditions Representative* and consists of all possible sets of conditions in the cluster abstracted using domain knowledge.

The process of designing each of these is explained below. In order to illustrate the concepts, we consider Example 1 showing all the sets of conditions, i.e., decision tree paths leading to a given cluster. These paths are obtained from the complete decision tree over the given data set. Using the distance function derived from the complete decision tree, candidate representatives are designed as follows.

Nearest Representative

The nearest representative is designed by guided selection, namely, it is chosen as one of the original objects in the given cluster. Using the distance function for conditions developed above that incorporates domain semantics, the nearest representative is selected as the set of conditions closest to all others in the cluster. It other words this representative is such that the sum of its distances from all other sets of conditions in the cluster is the least.

The nearest representative for the cluster in Example 1 is shown in Figure 5.2.



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

Figure 5.2: Example of Nearest Representative

This representative is designed to show the most important cluster information in a concise form. It is useful in applications where the user is interested in finding out the most likely set of input conditions that would give a desired nature of output. Since it consists of just one set of conditions from the original cluster, this representative is also called the *Single Conditions Representative*.

Summarized Representative

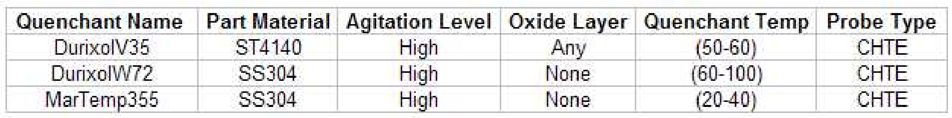
The summarized representative as the very name implies summarizes the information in the cluster. It is designed by construction, i.e., it is developed by using information from original cluster. The construction occurs as follows. The set of conditions in each cluster are grouped into sub-clusters based on the similarity of the conditions. The notion of similarity for sub-clustering the conditions is the distance function D_{cond} defined earlier.

The number of sub-clusters for each cluster is determined based on domain knowledge. For example, in Heat Treating we have the following information.

- *Quenchant Name* is the root of the tree and gets a higher weight than other attributes in the distance function.
- One important purpose of conducting the quenching experiments in Heat Treating is to categorize the quenchants.
- *Quenchant Name* has more distinct values than the other attributes closer to the root.

Based on this knowledge, the number of sub-clusters is set equal to the number of distinct values of Quenchant Name.

Sub-clustering is then done using any suitable clustering algorithm using D_{cond} as the notion of distance [HK01]. 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 summarized representative is an aggregation of all sub-cluster representatives displayed in a tabular form. The summarized representative for Example 1 is shown in Figure 5.3.



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

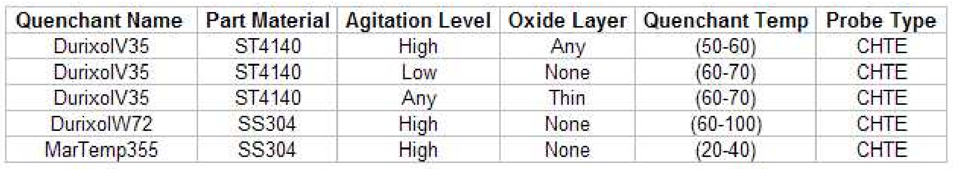
Figure 5.3: Example of Summarized Representative

The summarized representative is designed because it depicts a trade-off between the amount of detail displayed to the user and the amount of information captured within the cluster. It is useful in applications where the user wishes to find out, for example, distinct combinations of the most significant condition that would give a desired nature of output. Since this representative consists of multiple sets of conditions from the original cluster it is also called the *Multiple Conditions Representative (MCR)*.

Combined Representative

The combined representative is designed to capture all the data in the cluster with no information loss. This is also designed by construction. It is constructed by retaining all the original sets of conditions and displaying them by sorting based from the most to least significant attribute. The significance of the attributes is determined based on the distance function D_{cond}. The values of each set of conditions are abstracted using domain knowledge wherever possible. For example, in Heat Treating, if three 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. Likewise, if two sets of conditions are identical except that *Quenchant Temperature* has two consecutive ranges $(110 - 120)$ and $(120 - 130)$, then these are abstracted into a single set of conditions with *Quenchant Temperature* = $(110 - 130)$. This is in order to avoid visual clutter, while still displaying all information in the cluster.

The combined representative is an aggregation of all the sets of conditions sorted in ascending order from the most to the least significant. The combined representative for Example 1 is shown in Figure 5.4.



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

Figure 5.4: Example of Combined Representative

The combined representative is designed so as to convey all the information in the cluster in an organized manner. It is useful in applications where the user is interested in studying in detail all the possible inputs that would lead to a given nature of output. Since this representative consists of all sets of conditions within the cluster, it is also called an *All Conditions Representative (ACR)*.

Thus, three types of candidate representatives are designed for each cluster.

5.4.3 Comparison of Candidates

The candidate representatives are compared using an analogy with the Minimum Description Length (MDL) principle. The MDL principle proposed by Rissanen [R87] aims to minimize the sum of encoding the theory and the examples using the theory. In the literature, when MDL is used to encode cluster information, it is essential to be able to recover the original cluster from the encoding. However, in the context of our problem, we do not need to retrieve the cluster. Instead, we need to compare the cluster representatives with each other in order to evaluate them. Hence we propose a measure for comparison that is analogous to the Minimum Description Length of the cluster.

Our proposed measure is called the *DesCond Encoding* or the *DesRept Encoding for Conditions*. In our context, the theory (with respect to MDL) refers to the cluster representatives while the examples refer to all the other objects in the cluster. We take into account 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 refers to the capacity of the representative in capturing information within the cluster and is measured as the distance of the representative from all the objects in the cluster. The relative importance attached to the two terms of complexity and distance (information loss) is also taken into account in the encoding, based on the interests of targeted users. Given this, the DesCond Encoding is described below.

The DesCond Encoding $En_c = UBC \times log_2(AV) + UBD \times log_2 \frac{1}{s} \Sigma_{i=1}^{s} D(R, S_i)$

where, En_c = encoding for conditions,

A = number of attributes in the representative,

V = number of values for each attribute in the representative,

R = cluster representative,

S_i = each set of conditions in cluster,

$D(R, S_i)$ = distance between representative and every set

of conditions using the given distance function,

s = total number of sets of conditions in cluster,

UBC = percentage weight giving user bias for complexity,

UBD = percentage weight giving user bias for distance.

The first term in this encoding $log_2(AV)$ denotes 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 $log_2 \frac{1}{s} \Sigma_{i=1}^{s} D(R, S_i)$ denotes the 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 UBC and UBD are the percentage weights assigned to the complexity and distance terms respectively in order to give the user bias for those two 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 for the given cluster is considered the best and is returned as its designed representative.

5.4.4 Evaluation of DesCond

DesCond has been implemented in Java and evaluated using real data from the Heat Treating domain [TBC93]. Evaluation of DesCond has been conducted with domain expert interviews using the DesCond Encoding as described below.

Evaluation Process

Domain experts have provided different user bias weights in the DesCond Encoding based on their notions of targeted user interests. Using these weights candidate representatives have been evaluated. Different datasets consisting of Heat Treating experiments placed into clusters have been sent as input to DesCond. Parameters altered in DesCond besides the user bias weights have been dataset size and number of clusters. Any suitable algorithm such as k-means [KR94] has been used to generate the clusters over the datasets. In addition to altering the values of k, i.e., number of clusters, the clustering seeds have been altered to provide randomization. Given these clusters as input, the output of DesCond is the winning candidate for each cluster.

For comparison, a random representative has been considered per cluster in the evaluation process. Scores have been assigned to each representative as the number of clusters in the given dataset in which it is the winner. For example, in a dataset of 25 experiments placed in 5 clusters with $(50/50)$

weights, if the winner has been the nearest representative for one cluster and the combined representative for the other four, then the scores have been, Nearest:1, Summarized:0, Combined:4 and Random:0. The statistics are reported accordingly.

We show here the evaluation results over totally 210 experiments with a small dataset of 25 Heat Treating experiments placed in 5 clusters, a medium dataset of 150 experiments in 10 clusters and a large dataset of 400 experiments in 20 clusters. We consider 7 different user bias weights in the DesCond Encoding spreading over various possible applications as identified by experts. Each experiment shows the average of 10 experiments with different clustering seeds. Results are reported as scores for representatives in Figures 5.5, 5.6 and 5.7 respectively.

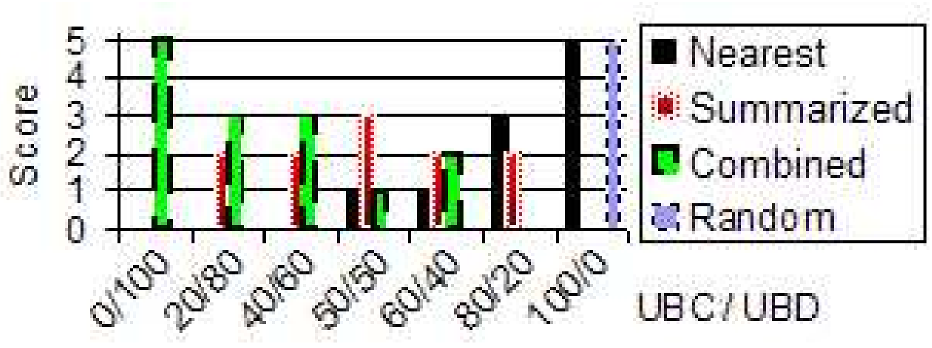


Figure 5.5: Statistics for Small Data Set

Observations and Discussion

- For (20/80) weights, the combined representatives generally win. Such weights are likely to occur in applications such as intelligent tutoring

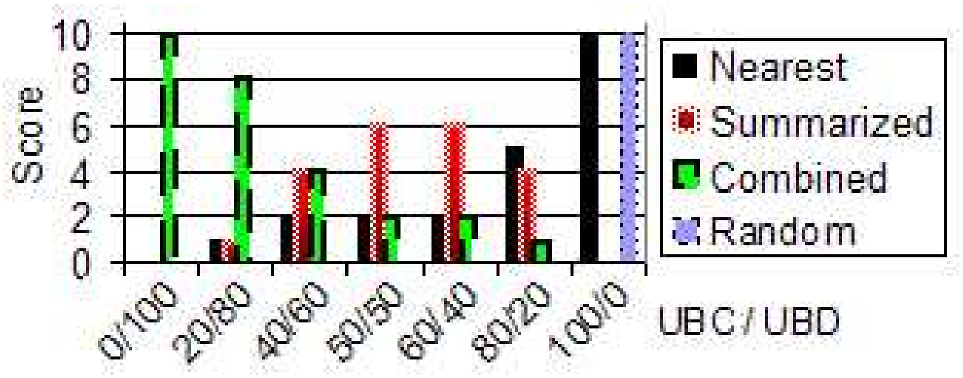


Figure 5.6: Statistics for Medium Data Set

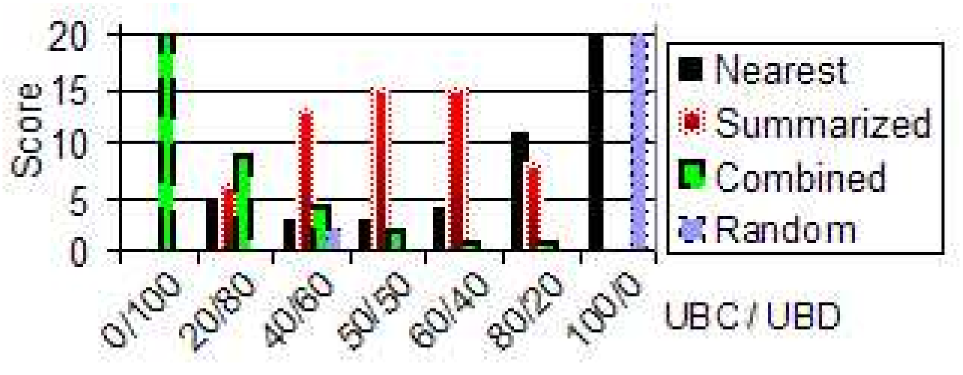


Figure 5.7: Statistics for Large Data Set

systems [BK88]. In such systems it is important to study all information in the cluster to analyze process behavior in detail. Complexity of the representative does not matter as much. Thus combined representatives would be useful here.

- For $(50/50)$ weights, the summarized representatives win for most data sets. These would probably be useful in simulation applications [LVKR02] where a trade-off between complexity and information loss is needed.

- For $(80/20)$ weights, the nearest representatives are often winners. These would most likely be useful in applications such as parameter selection [MCMMS02]. Here a representative is used to analyze the behavior of a cluster to compare processes for selecting process parameters in industry. Thus a simple representative is good and hence nearest representatives are useful especially for large data sets.

- For the $(40/60)$ and $(60/40)$ weights, combined representatives win for the small dataset while summarized representatives win with or without a tie for medium and large datasets. These would likely also be useful in various simulation applications where the user bias could tilt more or less in favor of complexity and distance, still requiring a trade-off.

- Random representatives lose in most cases. This indicates that designed representatives consistently outperform random ones in our targeted applications.

5.5 DesGraph: Designing and Evaluating Representatives for Graphs

We propose an approach called DesGraph that designs and evaluates domain-specific cluster representatives of graphs. In DesGraph also as in DesCond we utilize two design methods, namely, guided selection and construction. In guided selection, the representative is chosen to be one object of the cluster, e.g., the graph that forms the cluster medoid [KR94]. In construction, the representative is a new object developed using cluster information, e.g., by superimposing all graphs in the cluster. These selected and constructed objects form candidate representatives in DesGraph. An effectiveness measure for evaluating these representatives is proposed. The proposed measure called the DesGraph Encoding is analogous to the Minimum Description Length [M97] principle. The DesGraph Encoding incorporates the complexity of the cluster representative, information loss due to the representative and interests of targeted users. Candidate representatives are compared using this encoding. The candidate with the lowest encoding is the winner.

Thus the main tasks involved in DesGraph are as follows.

1). Specify a notion of distance for the graphs.

2). Design candidate cluster representatives using the design strategies of guided selection and construction.

3). Define an effectiveness measure to compare the candidates in order to return the best in the context of targeted application.

These tasks are discussed in the three subsections to follow.

5.5.1 Notion of Distance

In DesGraph, the distance metric learned from LearnMet is used as the notion of distance for graphs. For convenience the definition of this distance metric is stated here.

Distance Metric for Graphs A distance metric for graphs, namely, D_{graph} is a weighted sum of components, where each component can be a position-based, a statistical, or a critical distance metric applicable to the graphs. The weight of each component is a numerical value indicating its relative importance in the domain.

Hence, this distance metric is of the form $D_{graph} = \Sigma_{i=1}^{m} w_i D_i$ where each D_i is a component, w_i is its weight, and m is number of components applicable to the graphs.

The components and weights in this distance metric are learned from the executions of LearnMet. Among the metrics learned from various such executions, the metric that gives the highest accuracy in clustering after rigorous evaluation of LearnMet is the preferred notion of distance in DesGraph.

5.5.2 Building Candidate Representatives

Consider the example of Cluster A in Figure 5.8. We explain design of candidate representatives based on this example.

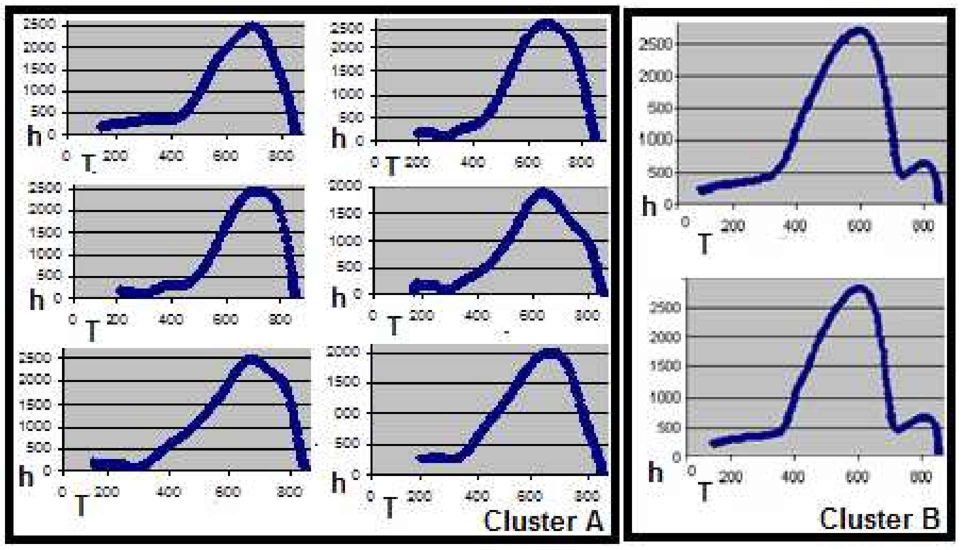


Figure 5.8: Clusters of Graphs

Design by Guided Selection

In guided selection the representative is chosen as one of the objects of the cluster. Two candidate representatives, *nearest* and *medoid* are selected as shown in Figure 5.9.

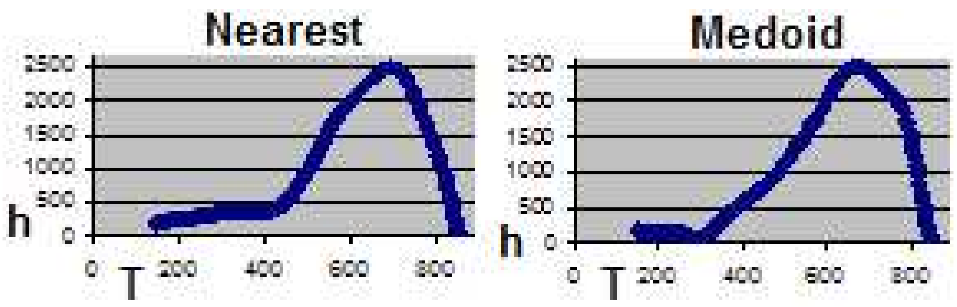


Figure 5.9: Selected Representatives

Nearest Representative The nearest representative is based on the concept of nearest neighbors using pairwise distances, as defined below.

```
$FOR\ f = 1$ *to* $g$
    $SUM(f) = \Sigma_{i=1}^{g} D(G_f, G_i)$
$ENDFOR$
$RETURN\ R = G_f$ *with lowest* $SUM(f)$
```

where G_f,G_i refer to individual graphs in the cluster, g is the total number of graphs in the cluster, R is the representative graph and D is the distance between graphs using the given metric. We use sum and not sum of squares because the assumption is that squared distances are already incorporated in the metric. This representative, the nearest graph, shows users 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. This representative is also called the *minimal distance graph* since it is the graph in the cluster with minimal distance from all others.

Medoid Representative A medoid representative, the graph in the cluster closest to its centroid, is defined below.

```
$FOR\ j = 1$ *to* $n$
    $Cen(j) = \frac{1}{g} \Sigma_{i=1}^{g} G_i(j)$
$ENDFOR$
$FOR\ i = 1$ *to* $g$
    $DIST(i) = \Sigma_{i=1}^{g} D(Cen, G_i)$
$ENDFOR$
$RETURN\ R = G_i$ with lowest $DIST(i)$
```

where G_i refers to each graph, $G_i(j)$ is the value of the dependent variable (y-coordinate) at the j^{th} value of the independent variable (x-coordinate), n is the number of x-coordinates on the graphs, g is the number of graphs in the cluster, Cen is the cluster centroid and D is the distance using the given metric. The assumption is that the x-coordinates for all graphs are the same. Hence in computing the centroid, we take a mean of the y-coordinates only. This representative, the medoid graph, helps users visualize the object in the cluster closest to the average behavior of the dependent variable on the graphs. This representative is also called the *selected average graph* since it is the graph in the original cluster closest to its average.

Design by Construction

In construction the representative is an object developed using data in the cluster. We describe two such representatives, *summarized* and *combined* as shown in Figure 5.10.

Summarized Representative The summarized representative presents a summary of information in the cluster. It is an average of graphs in the cluster with domain-specific upper and lower prediction limits. Average is computed as the cluster centroid while prediction limits are percentage upper and lower domain-specific thresholds added and subtracted from the average respectively, as follows.

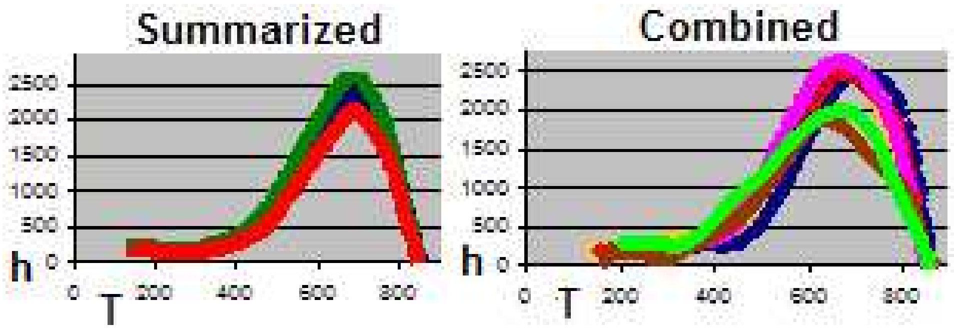


Figure 5.10: Constructed Representatives

```
$FOR\ j = 1\ to\ n$
  $R_{Av}(j) = \frac{1}{g}\Sigma_{i=1}^{g} G_i(j)$
  $R_{Up}(j) = R_{Av}(j) + \frac{U}{100} * R_{Av}(j)$
  $R_{Low}(j) = R_{Av}(X_j) - \frac{L}{100} * R_{Av}(j)$
$ENDFOR$
$RETURN\ R = R_{Up}, R_{Av}, R_{Low}$
```

where G_i refers to each graph, $G_i(j)$ is its y-coordinate at the j^{th} x-coordinate, n is the number of x-coordinates, g is the number of graphs in the cluster, R_{Av}, R_{Up} and R_{Low} are the average graph, upper limit and lower limit respectively, $R_{Av}(j)$, $R_{Up}(j)$ and $R_{Low}(j)$ being their respective y-coordinates at the j^{th} x-coordinate, U and L are percentage thresholds for upper and lower limits respectively, and R denotes the representative. Thresholds are obtained from a study of the data and discussions with experts. For example, in Heat Treating both thresholds are 10%. This representative, namely, the average graph with prediction limits, is a complex object consisting of 3 curves. It gives users a depiction of ranges of information in the cluster.

This is also called the *constructed average graph* since it is constructed as an average with prediction limits by using graphs in the original cluster.

Combined Representative The combined representative is constructed by superimposing all the graphs in a given cluster on each other as follows.

```
$FOR\ j = 1\ to\ n$
  $FOR\ i = 1\ to\ g$
    $R_i = (G_i(j))$
  $ENDFOR$
$ENDFOR$
$RETURN\ R = R_i : i = 1\ to\ g$
```

where G_i is each graph, $G_i(j)$ is its y-coordinate at the j^{th} x-coordinate, n is the number of x-coordinates, g is the number of graphs in the cluster, and R is the representative. This representative, called the superimposed graph, is a complex object composed of g curves. It shows users the whole cluster with no information loss and depicts possible subtleties in the cluster. For example, the combined representative in Figure 5.10 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 on each other.

5.5.3 Effectiveness Measure for Representative Graphs

We propose an effectiveness measure called the *DesGraph Encoding* or *DesRept Encoding for Graphs* for evaluating the representative graphs. This encoding is analogous to the Minimum Description Length (MDL) principle [M97]. MDL aims to minimize the sum of encoding a theory and the examples using a theory. In DesGraph, the theory is the representative itself and the examples are all the objects in the cluster. However, the difference is that in DesGraph, we do not need to retrieve the original cluster from the encoding. Rather, we aim to compare the quality of the representatives in terms of how well they capture cluster information and how complex they are taking into account user interests. Hence the complexity of storing the representative graph and its distance from all graphs in the cluster are incorporated in the DesGraph Encoding. This encoding aims to minimize the sum of the number of bits to store the representative and the distance of all graphs from the representative. The user bias for complexity and distance is considered as percentage weights for each term. The encoding is given below.

The DesGraph Encoding (DesRept Encoding for Graphs) $En_g = UBC * log_2(N_r) + UBD * log_2(\frac{1}{g}\Sigma_{i=1}^{g} D(R, G_i))$

where En_g = encoding for graphs

N_r = number of data points to store representative graph

R = the representative graph

G_i = each individual graph in the cluster

D = distance between graphs using the given metric

g = total number of graphs in the cluster

UBC = percentage weight giving user bias for complexity

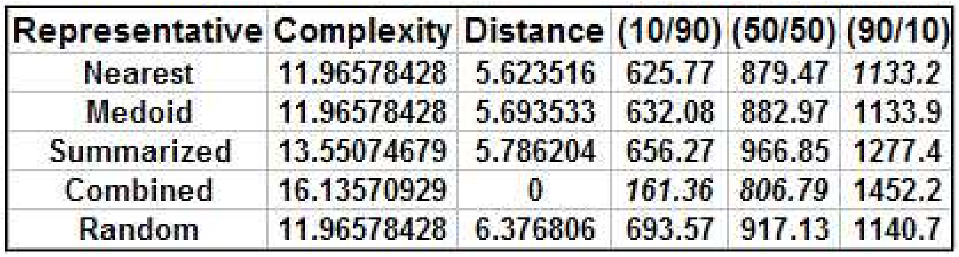
UBD = percentage weight giving user bias for distance

The first term in the encoding, $log_2(N_r)$, is the complexity of storing the representative. Given that N is the number of x-coordinates, $N_r = N$ if R is *nearest* or *medoid*, $N_r = 3 * N$ if R is summarized, and $N_r = g * N$ if R is combined.

The second term in the encoding, $log_2(\frac{1}{g}\Sigma_{I=1}^{g}D(R,G_i))$ 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. Distance is $D(R,G_i)$ if R is *nearest* or *medoid*, as the minimum of $D(R_{Av},G_i)$, $D(R_{Up},G_i)$ and $D(R_{Low},G_i)$ if R is *summarized* and as the minimum of all values $D(R_i,G_i) : i = 1$ *to* g for the given G_i if R is *combined*.

Percentage weights UBC and UBD give user bias for complexity and distance terms in the encoding respectively. Default weights are 50% each, indicating equal importance of both terms. In some situations users are interested in capturing more information in the cluster and do not 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.

Figure 5.11 shows calculations for measuring the effectiveness of representatives for Cluster A. Designed candidates are compared with each other and with a random representative. Complexity and Distance columns in the figure show values of the respective terms in the encoding without user bias. Columns (10/90), (50/50) and (90/10) give user bias for complexity and distance respectively. Winners for each column are shown in italics.



Representative	Complexity	Distance	(10/90)	(50/50)	(90/10)
Nearest	11.96578428	5.623516	625.77	879.47	*1133.2*
Medoid	11.96578428	5.693533	632.08	882.97	1133.9
Summarized	13.55074679	5.786204	656.27	966.85	1277.4
Combined	16.13570929	0	*161.36*	*806.79*	1452.2
Random	11.96578428	6.376806	693.57	917.13	1140.7

Figure 5.11: Effectiveness of Representatives

5.5.4 Evaluation of DesGraph

DesGraph has been implemented in Java and is experimentally evaluated using real data from Heat Treating. Evaluation of DesGraph has been performed by conducting domain expert interviews and using the DesGraph Encoding.

Evaluation Process

In the evaluation clusters of graphs over different data sets have been sent as input to DesGraph. Domain experts have provided different user bias terms for complexity and distance reflecting the interests of users in targeted applications. Input parameters altered have been the weights of complexity and distance, data set size, number of clusters, and clustering seeds. The clustering algorithm used has been k-means [KR94]. Output of DesGraph is the winning candidate for each cluster.

For comparison, a random representative has been considered per cluster in the evaluation process. Scores have been assigned to each representative as the number of clusters in the data set in which it is the winner. For example, in a data set of 25 graphs in 5 clusters with $(50/50)$ weights, if the winner has been the medoid representative for two clusters and the summarized representative for the remaining three, then the scores have been, Nearest:0, Medoid:2, Summarized:3, Combined:0 and Random:0. The statistics has been reported accordingly.

Evaluation Results

A summary of the evaluation of DesGraph in Heat Treating is presented here. We show the results of 330 experiments run with a small data set of 25 graphs in 5 clusters, a medium data set of 150 graphs in 10 clusters and a large data set of 400 graphs in 20 clusters. For each data set, user bias for complexity and distance is altered from $(0/100)$ to $(100/0)$ respectively in steps of 10. Each experiment is run 10 times, altering clustering seeds to build the clusters input to DesGraph. The average of 10 experiments is shown here. Results are reported as scores for representatives in Figures 5.12, 5.13 and 5.14 respectively. The observations made from the evaluation results are given below, followed by a discussion on their usefulness with respect to targeted applications.

Observations and Discussion

The following observations can be made from the evaluation results.

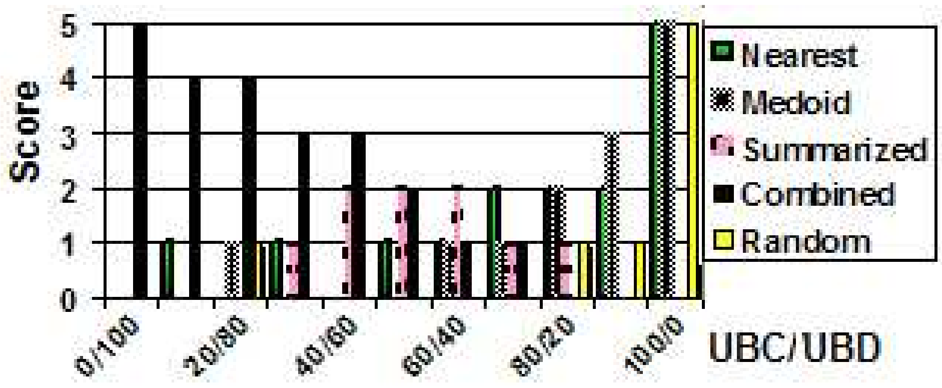


Figure 5.12: Results for Small Data Set

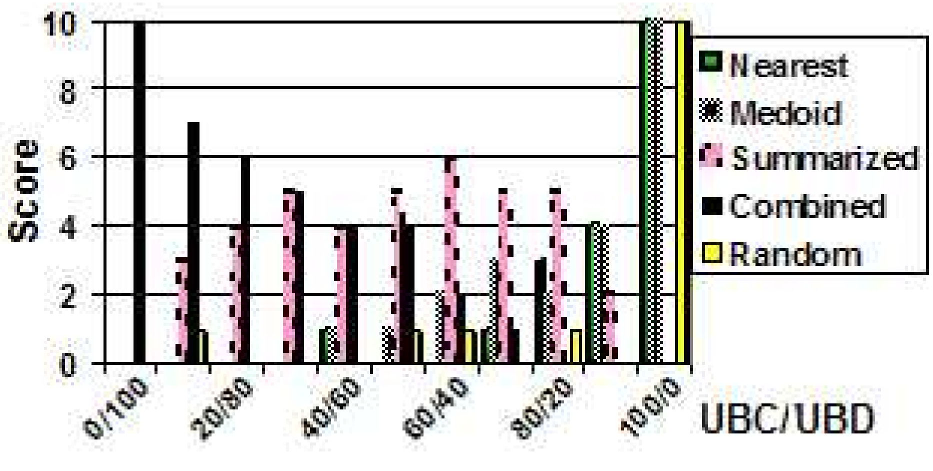


Figure 5.13: Results for Medium Data Set

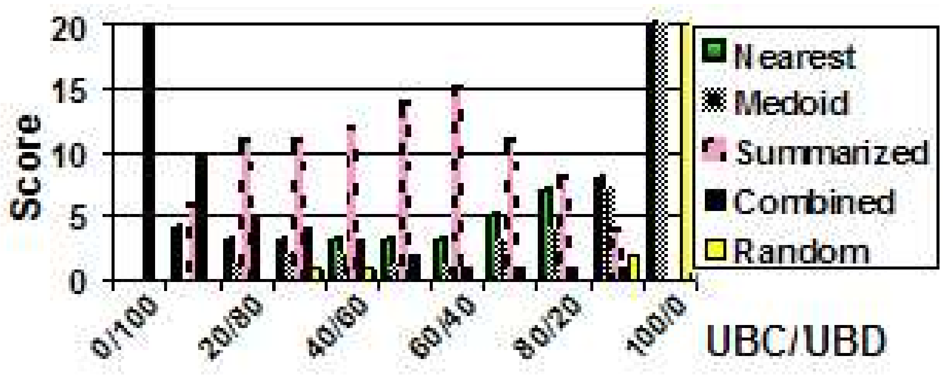


Figure 5.14: Results for Large Data Set

- For the small data set, combined representatives are often winners followed by nearest and medoid.
- For the medium data set, the winners are usually summarized and combined representatives.
- For the large data set, summarized representatives are winners in most cases.
- For (10/90) weights, combined representatives win regardless of data set size.
- For (50/50) weights, summarized representatives win (with or without a tie) for all data sets.
- For (90/10) weights, all data sets have nearest/medoid representatives as winners.
- Random representatives lose almost always, except when users give zero weight to the distance term.

These observations help design representatives in domain-specific applications as follows.

- The (90/10) weights are likely to arise in applications such as parameter selection [MCMMS02]. Here a representative is used to study the behavior of a cluster to compare processes for selecting process parameters in industry. Hence a simple representative is desirable. Thus nearest/medoid representatives are useful, especially for large data sets.

- The (50/50) weights are typically found in simulation applications [LVKR02]. Users run simulations with representatives depicting ranges of information in the cluster. Hence the distance term matters because it denotes information loss. Complexity matters because simulations are time-consuming. Hence summarized representatives are useful for most data set sizes.

- The (10/90) weights would probably occur in applications such as decision support systems [VTRWMS03] for experts. In such systems it is important to study all information in the cluster to analyze process behavior in detail. Complexity of the representative does not matter as much. Thus combined representatives are useful in such applications.

5.6 Estimation using Designed Representatives from DesRept

The output of DesRept is a designed representative pair of input conditions and graph per cluster. Each representative graph is stored as an n-dimensional point. The conditions are stored as attribute-value pairs. Note that the designed representatives are the best candidate representatives for every cluster. Estimation is now done using these representative pairs. The process of estimation is similar to that in the basic AutoDomainMine approach.

5.6.1 Estimation of Conditions

Consider the situation when the user submits a desired graph to estimate a set of conditions that would achieve this. In order to search for the closest matching graph, the metric D_{graph} is used as the notion of distance. The desired graph is compared with all the representative graphs. A threshold for similarity is defined.

If no match is found within the given threshold, then it is conveyed to the user that the conditions to achieve the desired graph cannot be estimated. However if within a given threshold a match is found, then the representative set of conditions corresponding to the representative graph are displayed to the user in plain text.

5.6.2 Estimation of Graph

Consider the scenario where the user submits a set of conditions to estimate the graph that would be obtained. The given set of conditions are compared with all the decision tree paths to trace the cluster as in the basic AutoDomainMine approach.

If the search stops at less than half the the height of the decision tree path, then it is conveyed to the user, that the estimation cannot be performed. If the search stops at greater than half the height, then any cluster from that point can be the estimated cluster. This is justified by learning the decision tree weight heuristic since the attributes above half the height of the tree get distinctly higher weights than those below.

If exactly one path matches, then that cluster is the estimated one. If more than one complete path matches, then a majority voting for clusters is done. For example, if three paths lead to cluster B and two to cluster C, then cluster B is the estimated cluster. In case two or more clusters win, then any one is selected. The designed representative graph of the estimated cluster is then the estimated graph for that experiment. This is displayed to the user as a plotted graph.

5.6.3 Displaying the Output

The output of the estimation displayed to the user are the best overall candidate representatives as determined by a thorough evaluation of DesCond and DesGraph respectively. The representative set of conditions is displayed as a table while the representative graph is displayed as a picture.

5.6.4 Evaluation with AutoDomainMine

The evaluation of the representatives of input conditions and graphs built by DesRept (DesCond and DesGraph respectively) has been done by incorporating them into AutoDomainMine, and judging the effectiveness of the overall estimation. A good representative should provide more accurate estimation since it takes into account domain semantics. The opinion of the targeted users is considered in this evaluation. This is discussed in detail in the next chapter.

Chapter 6

User Evaluation of the AutoDomainMine System

6.1 AutoDomainMine Stage 3: The Complete System

Stage 3 of AutoDomainMine is the complete system that incorporates all the parts, i.e.,

- The basic AutoDomainMine approach of integrating clustering and classification to discover knowledge for estimation.
- The LearnMet technique for learning semantics-preserving distance metrics for graphs
- The DesRept methodology for designing domain-specific cluster representatives.

The tool developed using AutoDomainMine after incorporating LearnMet and DesRept has been subjected to user evaluation in targeted applications of AutoDomainMine. The evaluation has been conducted with real data from the Heat Treating domain.

We give below the process of evaluation, the evaluation results and a discussion on the evaluation. This is followed by an assessment of the estimation accuracy of AutoDomainMine based on the user evaluation.

6.2 Process of User Evaluation with Formal Surveys

The AutoDomainMine system has been evaluated by the targeted users of its applications. Formal user surveys have been conducted for evaluation. The process of evaluating AutoDomainMine is as follows.

6.2.1 Holdout Strategy for Evaluation

Laboratory experiments in Heat Treating, namely, quenching experiments, have been used to evaluate the estimation provided by AutoDomainMine. The holdout strategy [RN95] has been used with the entries in the database, i.e., some entries have been held aside for testing. These have not been used for training in the clustering and classification steps of AutoDomainMine. For these entries held aside for testing, the estimation and real experiment have been compared by heat treating user through formal user surveys.

If the estimation matched the real experiment, then it has been considered accurate, else inaccurate. Likewise, accuracy has been evaluated using the data in the test set. Percentage accuracy has been reported as the percentage of tests for which the estimation has matched the real result. The purpose of this evaluation has been two-fold. First, it has helped to evaluate the effectiveness of the designed representatives for estimation. Second, has served as an evaluation of the complete AutoDomainMine system as an estimation tool.

6.2.2 Details of User Surveys

Users have executed tests by comparing the estimation of AutoDomainMine with the real laboratory experiments not used for training the technique. In each test executed by users, the designed representatives have been compared with each other in terms of their effectiveness in capturing information in the applications of AutoDomainMine. The applications include parameter selection, simulation tools, intelligent tutoring systems and decision support systems.

User Displays

In order to perform the evaluation, the estimated output of AutoDomainMine has been displayed to the users in three different levels of detail as follows.

- Display 1: Nearest Representative
- Display 2: Summarized Representative
- Display 3: Combined Representative

We have designed these displays based on the winning candidates in the DesCond and DesGraph evaluations based on the respective encodings. It has been found that for different sets of user bias weights, different candidates were winners. However, randomly selected representatives have consistently been losers and hence have not been included in these displays to avoid redundancy. Among the nearest and medoid graphs, it has been found that nearest graphs won overall based on a majority vote over the entire data set for those user bias weights where complexity mattered distinctly more. Hence the nearest graph has been used for the respective display. Note that in these surveys, the real users have been involved and we needed to consider the fact that they had limited amount of time to complete the surveys. Moreover, they wanted to see the system at its best. Hence the evaluation that has already been conducted with domain expert interviews using the encoding has been used to effectively design the displays for the evaluation with formal user surveys.

Survey Questionnaire

In the surveys, the users have been asked to indicate which display captured the real experiment most closely, with respect to their targeted application. They have been given the option of indicating that none matched implying that estimation itself is inaccurate. Accuracy has been reported as the percentage of accurate estimations (i.e., where users selected displays 1, 2 or 3). This has been compared with the accuracy in Stage 2 of AutoDomainMine that used randomly selected cluster representatives for estimation.

6.3 Evaluation Results in Targeted Applications

We now summarize the results of the user evaluation surveys with respect to different applications.

6.3.1 Computational Estimation

The survey results in this category are for the AutoDomainMine system as a whole indicating the effectiveness of the designed representatives in computational estimation [VRRBMS06]. The users have conducted 100 tests in this category.

Observations from Computational Estimation Applications

Figures 6.1 and 6.2 show pie charts giving the distribution of winners among the candidate representatives for conditions and graphs respectively. In these pie charts the region corresponding to *None Wins* shows the inaccurate estimations.

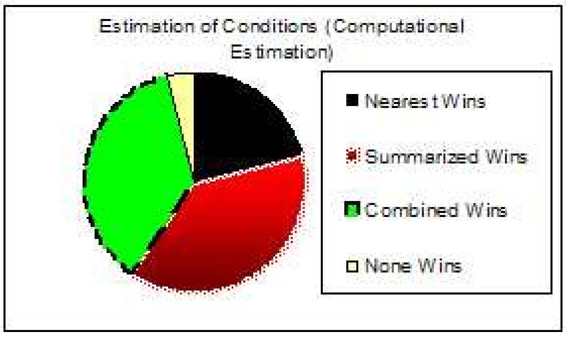


Figure 6.1: Winners for Conditions in Computational Estimation

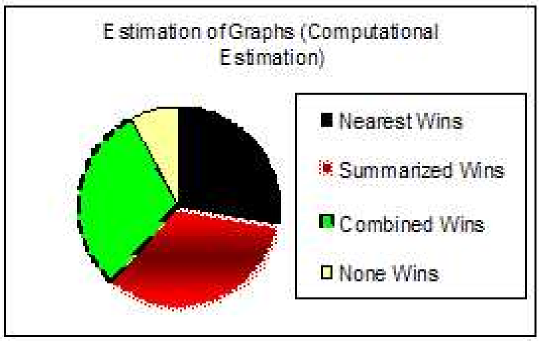


Figure 6.2: Winners for Graphs in Computational Estimation

Analysis of Computational Estimation Applications

It is seen that for computational estimation, the combined and summarized representatives are winners in most tests executed by users, with nearest representatives trailing closely behind. Since computational estimation has a broad range of users, different types of representatives are found to win.

6.3.2 Parameter Selection

In these applications, the output of AutoDomainMine is used to select process parameters in industry [MCMMS02]. The users have conducted 53 tests in this category.

Observations from Parameter Selection Applications

The winners in these applications are shown in the pie charts in Figures 6.3 and 6.4 for conditions and graphs respectively.

The *None Wins* region in these charts indicates the tests where none of the candidate representatives matched the real data as per the needs of parameter selection users, hence the estimation was inaccurate with respect to parameter selection applications.

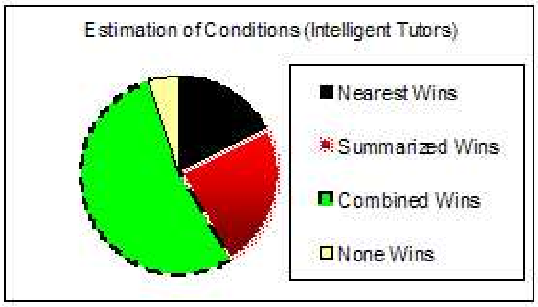


Figure 6.3: Winners for Conditions in Parameter Selection

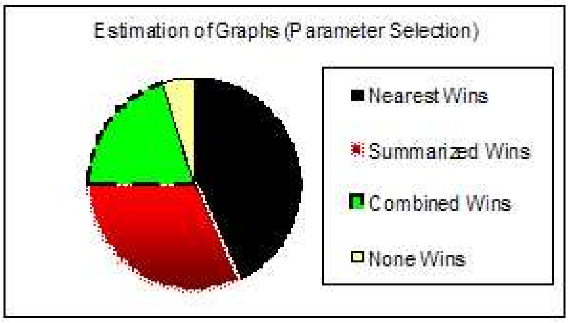


Figure 6.4: Winners for Graphs in Parameter Selection

Analysis of Parameter Selection Applications

As observed in the figures, nearest representatives are the winners for most tests. The reason for this likely would be that in parameter selection, typically most users want one right answer that is displayed in a concise manner.

6.3.3 Simulation Tools

In simulation tools, the users need the cluster representatives to run computer simulations of a real laboratory experiment [LVKR02]. The simulation users have conducted 62 tests with AutoDomainMine.

Observations from Simulation Tool Applications

Figures 6.5 and 6.6 show the winning candidates in simulation tool applications for conditions and graphs respectively.

The *None Wins* region in these chart indicates the tests where none of the candidate representatives matched the real data as per the needs of simulation tool users, hence the estimation was inaccurate with respect to simulation applications.

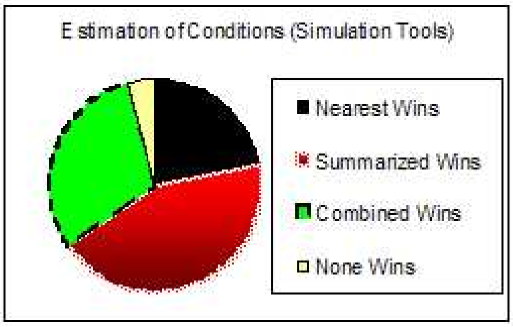


Figure 6.5: Winners for Conditions in Simulation Tools

Analysis of Simulation Tool Applications

From the pie charts, it is seen that summarized representatives are the winners in most tests. This is probably because simulation tool users generally

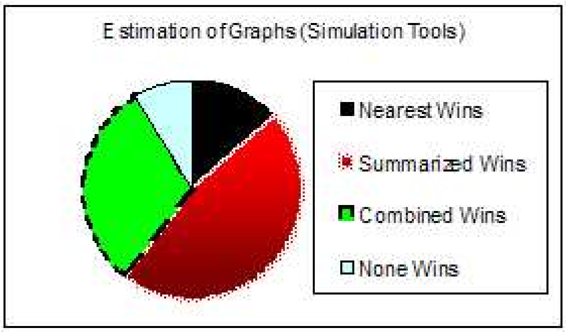


Figure 6.6: Winners for Graphs in Simulation Tools

want to use ranges of information in order to increase the sample space of the simulations, but they also care about complexity since simulations are time-consuming. Hence, we find that they prefer the summarized representatives.

6.3.4 Intelligent Tutoring Systems

Intelligent tutoring systems are used to study in detail the behavior of processes analogous to classroom study on the given topic [BK88]. Totally 37 tests have been conducted by users in this category.

Observations from Intelligent Tutoring System Applications

Figures 6.7 and 6.8 show what type of representatives suited the users of these applications for conditions and graphs respectively.

The *None Wins* region in these charts indicates the tests where none of the candidate representatives matched the real data as per the needs of intelligent tutoring system users, hence the estimation was inaccurate with respect to tutoring applications.

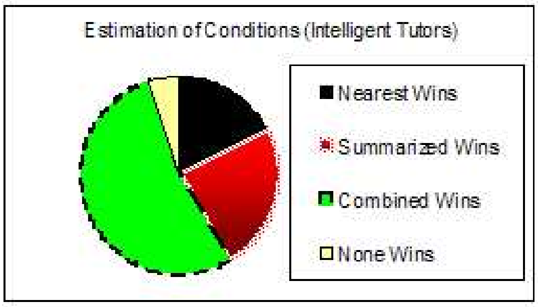


Figure 6.7: Winners for Conditions in Intelligent Tutoring Systems

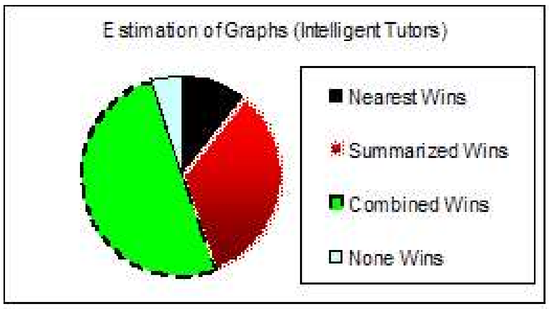


Figure 6.8: Winners for Graphs in Intelligent Tutoring Systems

Analysis of Intelligent Tutoring System Applications

From the charts it is clear that in most cases combined representatives are the winners. 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 do not care much about complexity. Hence, more detail is appreciated.

6.3.5 Decision Support Systems

Decision support systems [VTRWMS03] are used for various purposes. In high level business decision support, at-a-glance retrieval of information is important without much emphasis on detail. Some decision support users however, focus on process optimization and need to scrutinize information in more detail. We have had 44 tests conducted by decision support system users.

Observations from Decision Support System Applications

The distribution of winning candidates in decision support systems in shown in Figure 6.9 and 6.10 for conditions and graphs respectively.

The *None Wins* region in these charts indicates the tests where none of the candidate representatives matched the real data as per the needs of decision support system users, hence the estimation was inaccurate with respect to decision support applications.

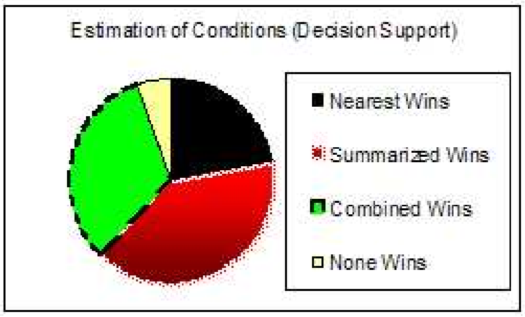


Figure 6.9: Winners for Conditions in Decision Support Systems

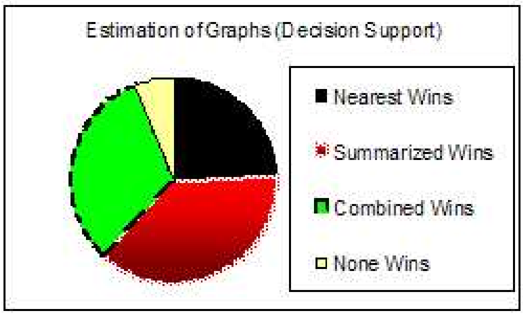


Figure 6.10: Winners for Graphs in Decision Support Systems

Analysis of Decision Support System Applications

From the figures, it is found that there is a fairly good mix of winners in these applications. This is because different decision support users are interested in different levels of detail. Hence it is desirable to retain all the representatives in designing such applications, and to display information in increasing levels of detail.

6.4 Discussion on User Surveys

The following conclusions can be drawn from the results of the user surveys.

- The use of the designed representatives enhance the estimation accuracy to 94% in AutoDomainMine [VRRBMS06]. This is higher than the earlier version of the system that used randomly selected representatives for estimation giving an accuracy of 87%.
- The results of the formal user surveys agree with the results of the

evaluation conducted with domain expert interviews using the DesRept Encodings. For example, summarized representatives win with $(50/50)$ weights in the encoding. These representatives are also the winners in simulation applications [LVKR02] which require a trade-off between complexity and information loss.

- All the designed representatives are useful (more or less) in computational estimation [VRRBMS06] and decision support applications [VTRWMS03]. Hence in designing these applications all of them would be retained, displaying the information in three different levels of detail.

- Nearest representatives are most useful in parameter selection [MCMMS02], summarized representatives in simulation tools [LVKR02] and combined representatives in intelligent tutoring systems [BK88]. Hence in designing the systems for the corresponding applications these representatives would be used respectively.

6.5 Estimation Accuracy of AutoDomainMine

Based on the results of the user surveys, the estimation accuracy of the AutoDomainMine system has been assessed. This depicts the final stage of AutoDomainMine, i.e., the completed system. The results are presented below with respect to the targeted applications of AutoDomainMine. Thus besides the main application of computational estimation, accuracy is also reported with respect to parameter selection, simulation tools, intelligent tutors and decision support systems.

6.5.1 Observations on Estimation Accuracy

Figures 6.11 and 6.12 show the accuracy for estimation of conditions and graphs respectively.

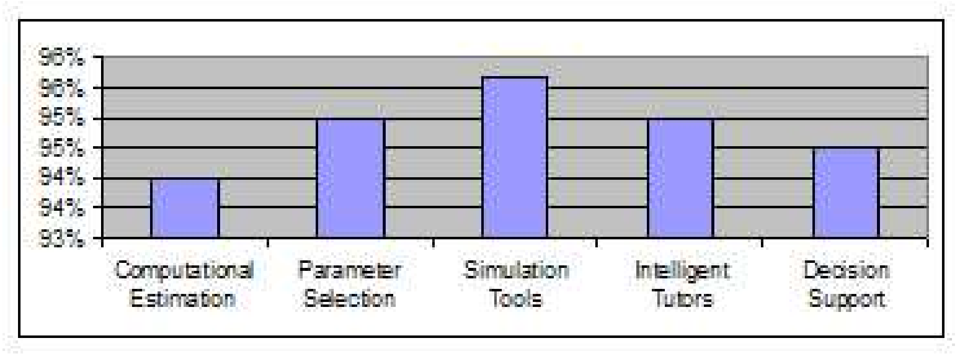


Figure 6.11: AutoDomainMine Accuracy for Estimation of Conditions

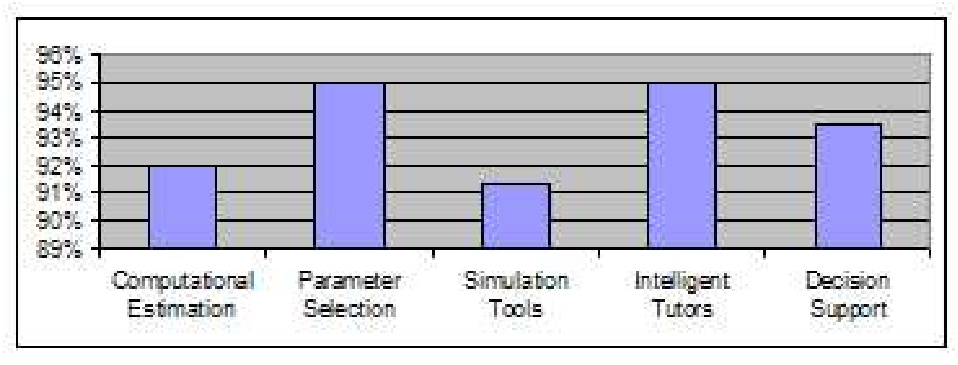


Figure 6.12: AutoDomainMine Accuracy for Estimation of Graphs

6.5.2 Discussion on Estimation Accuracy

Thus in this final stage, the overall estimation accuracy of AutoDomainMine is in the range of around 90% to 95%. This is higher than the intermediate stage (Stage 2) that gave an accuracy of approximately 86% to 87% and the pilot stage (Stage 1) that gave around 70% to 75%. This accuracy of 90% to 95% is considered acceptable in the domain.

The response time of the tool has been always found to be just a fraction of a second. It has not been separately noted in each user test, because it the same in each test. Thus AutoDomainMine takes distinctly less time than a real laboratory experiment.

AutoDomainMine also displays the estimation output to the users in an easy to interpret form while also conveying as much information as possible with respect to targeted applications.

6.5.3 Conclusions from User Evaluation of AutoDomainMine

The AutoDomainMine system has been considered an effective tool for computational estimation in the domain of Heat Treating of Materials as judged by the satisfaction of the users. It meets the targeted requirements of accuracy and efficiency while also conveying the output of the estimation in a form acceptable to the users.

Chapter 7

Conclusions

7.1 Summary and Contributions

Motivation and Goals. Experimental in scientific domains are often plotted as graphs to visually assist the analysis and comparison of the corresponding processes. The domain of focus in this dissertation is the Heat Treating of Materials that inspired this work. Performing an experiment in a laboratory and plotting graphs consumes significant time and resources motivating the need for computational estimation. The following are the goals of the required estimation technique.

- Given the input conditions of an experiment, estimate the resulting graph.
- Given the desired graph in an experiment, estimate the input conditions to obtain it.

The assumption is that existing experimental data is stored in a database as a set of input conditions and graph per experiment. It is found that state-of-the-art approaches, e.g., case-based reasoning, mathematical modeling and similarity search, are not satisfactory in the targeted applications.

Proposed Approach. In this dissertation, we have proposed a computational estimation approach called AutoDomainMine. AutoDomainMine integrates clustering and classification to discover knowledge from existing experimental data serving as the basis for estimation. Graphs from existing experiments are first clustered using a suitable clustering algorithm such as k-means. Decision tree classification with algorithms such as ID3 / J4.8 is then used to learn the clustering criteria (sets of input conditions characterizing each cluster) from which a representative pair of input conditions and graph is built per cluster. The decision trees and representative pairs form the knowledge discovered from existing experiments. Knowledge discovery is a one-time process. The discovered knowledge is used for the recurrent process of estimation. Given the input conditions of a new experiment, the relevant path of the decision tree is traced to estimate its cluster. The representative graph of that cluster is returned as the estimated graph for the experiment. Given a desired graph, the closest matching representative graph is found and its conditions are conveyed as estimated conditions to obtain the given graph. This estimation incorporates relative importance of conditions learned by decision trees.

AutoDomainMine follows a typical learning strategy of Materials Scientists. They often analyze by grouping experiments based on similarity of obtained graphs and reasoning causes of similarity group by group in terms of impact of input conditions on graphs. This learning strategy is automated for knowledge discovery in AutoDomainMine.

Challenges. A significant challenge in AutoDomainMine is capturing the semantics of the concerned graphs in clustering. Several distance metrics such as Euclidean and statistical distances exist in the literature. However it is not known a priori which metric(s) would best preserve semantics if used as the notion of distance in clustering. Experts at best have vague notions about the relative importance of regions on the graphs but do not have a defined metric. State-of-the-art distance learning methods are either not applicable or not accurate enough in this context. We therefore propose a technique called LearnMet to learn domain-specific distance metrics for graphs. A LearnMet metric D is a weighted sum of components where each component is an individual metric such as Euclidean or statistical distance and its weight gives its relative importance in the domain. LearnMet iteratively compares a training set of actual clusters given by experts with predicted clusters obtained from any fixed clustering algorithm, e.g., k-means. In the first iteration, a guessed metric D is used for clustering. This metric is adjusted based on error between predicted and actual clusters using our proposed weight adjustment heuristic until error is below a given threshold or a maximum number of epochs is reached. The metric with error below threshold or with minimal error among all epochs is returned as the learned metric. The output of LearnMet is used as the notion of distance for clustering the graphs.

Another challenge in AutoDomainMine is capturing relevant data in each cluster while building representatives. A default approach of randomly selecting a representative pair consisting of a set of input conditions and graph per cluster is not found to be effective in preserving the necessary information. Since several sets of conditions lead to a single cluster, randomly selecting any one as a representative causes information loss. Randomly selected representatives of graphs do not incorporate semantics and ease of interpretation based on user interests. Thus, we propose a methodology called DesRept to design domain-specific cluster representatives for input conditions and graphs. In DesRept, two design methods of guided selection and construction are used to build candidates capturing various levels of detail within the cluster. Candidates are compared using encodings proposed in this dissertation analogous to the Minimum Description Length principle. The criteria in these encodings are complexity of the representative and information loss due to it based on user interests. The winning candidate for conditions and graphs, i.e., with the lowest encoding, is output its as designed representative. Thus, a designed representative pair consisting of the winning set of input conditions and graph is stored for each cluster. Likewise, various designed representative pairs showing information in different levels of detail are output by DesRept based on the interests of respective users as conveyed in the encoding. The designed representative pairs are used for estimation in AutoDomainMine.

Contributions. The main contributions of this dissertation are:

- The AutoDomainMine approach of integrating clustering and classification as a learning strategy to discover knowledge for estimation.

- The LearnMet technique for learning domain-specific distance metrics for graphs.
- The DesRept methodology of designing semantics-preserving cluster representatives for input conditions and graphs.
- The system developed using AutoDomainMine for computational estimation, a trademarked tool in Heat Treating.

Evaluation. AutoDomainMine has been evaluated rigorously in the Heat Treating domain. AutoDomainMine estimation is compared with real experiments from a distinct set not used for training. Formal user surveys are conducted for evaluation. Users execute tests comparing the estimation of AutoDomainMine with the real experiment. If the estimation matches the real experiment as per the needs of users in targeted applications, then it the estimation is considered to be accurate, else inaccurate. Accuracy is reported as the percentage of accurate estimations over the test set. The AutoDomainMine applications include parameter selection, decision support systems, intelligent tutoring systems and simulation tools. On being evaluated by the respective users in these categories, it is observed that AutoDomainMine gives an estimation accuracy in the range of 90% to 95% in different targeted applications.

7.2 Future Work

Image Mining. One interesting problem that stems from this dissertation is mining over complex data such as images with the goal of making domain-specific comparisons. For example, consider the field of Nanotechnology, a popular area in scientific databases today. In nanotechnology, there are images depicting nanostructures of materials. Comparing such images is important to understand the difference between material properties such as hardness. Data mining could prove useful here in order to automate some of these comparisons.

There are significant challenges in this process that provide the potential for research. One major challenge is the selection and/or development of appropriate techniques in order to automate image comparison. Another challenge is defining the notion of similarity for comparison. Yet another challenge is to propose interestingness measures analogous to the ideas of domain experts in drawing inferences from the comparison.

The dissertation contributions mentioned here could be enhanced to address some of these issues. For example, the AutoDomainMine approach of integrating clustering and classification as a learning strategy for knowledge discovery could be useful here. Images can be clustered based on their similarity and classification can be used to learn the causes of similarity. This could be useful to draw inferences in the given domain. The LearnMet technique for distance metric learning could be used to determine the notion of similarity between images. The steps and heuristics in LearnMet could be modified to deal with images that are more complex than graphs.

Research on domain-specific image comparison would involve a thorough study of the literature in data mining and in the given domain. The involvement of domain experts would also be crucial in order to provide the necessary inputs to solve the concerned problems.

Visualization of Text and Image Data. The manner of designing semantics-preserving representatives in this dissertation can be used for image and text summarization in visual displays. Examples of such displays include handheld PDA devices, web pages and other GUIs (Graphical User Interfaces). The encodings proposed in DesRept based on the Minimum Description Length Principle can be used to objectively evaluate representatives for visual displays. Since these encodings take into account the interests of targeted users in various applications, they can be used to design displays catering to the respective categories of users. These DesRept encodings can be modified as needed to suit specific applications based on the nature of the data.

Moreover, the design strategies of guided selection and construction can be used to build different types of representatives for displays. These strategies can also be enhanced to consider other methods of design giving more types of representatives as required in given applications. The DesRept criteria for design and evaluation, namely, domain semantics, visual ease of interpretation and interests of targeted users can be used in various other systems. These can also be altered to include other aspects in design and evaluation, such as display space which is often critical.

Further research on this topic presents interesting issues in data mining and visualization that are worth exploring in the future.

Data Stream Matching. There are interesting problems in the field of data stream matching. Continuous line matching techniques are often used for time series data. Retrieval of such data has several applications in the real world in engineering, medical and financial domains.

In medical applications, for example, a patient's heartbeat can be monitored and compared with existing data in a medical database to detect irregularities.

In financial domains, stock market analysis typically involves comparing trends in various time periods to find fluctuations in stock prices and use them to predict future trends. Several financial applications also involve intrusion detection based on outliers in time-series data. If a particular pattern is drastically different from normal, then it is used to indicate the possibility of intrusion.

In engineering, there are sensors that often send data streams. The system needs to monitor these streams to find relevant patterns. Instant reactions need to be delivered accordingly in some applications.

All these applications involve continuous time-series data with domain semantics. The distance metric learning technique LearnMet proposed in this dissertation could be used with various distance types in the literature to propose notions of similarity for the data in the respective domains. Enhancement would probably be needed to deal with streaming data.

Besides distance learning, there are other issues involved in data stream matching such as defining interestingness measures, proposing thresholds for matching and so forth. Exploring these issues would involve a detailed study of state-of-the-art in order to outline specific research problems and propose solutions.

Likewise several potential future issues are likely to emerge from this dissertation. Addressing them would pose interesting challenges. Some of these will probably be addressed in the near future.

Bibliography

[AIS93] R.Agrawal T.Imlinski and A.Swami; Mining Association Rules between Sets of Items in Large Databases; In *ACM SIGMOD*, pages 207–216, May 1993.

[AFS93] R.Agrawal, C.Faloutsos and A.Swami; Efficient Similarity Search in Sequence Databases; In *4th International Conference on the Foundations of Data Organization and Algorithms*, pages 69–84, October 1993.

[AP03] A.Aamodt and E.Plaza; Case Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches; *AICom: Artificial Intelligence Communications*, 7(1):39–59, 2003.

[AV01] R.Amen and P.Vomaka; Case Based Reasoning as a Tool for Materials Selection; *Materials and Design*, 22:353–358, 2001.

[B68] E.Butkov; *Mathematical Physics*; Addison-Wesley Publishing Company, 1968.

[B96] C.Bishop; *Neural Networks for Pattern Recognition*; Oxford University Press, England, 1996.

[BC89] H.Boyer and P.Cary; *Quenching and Control of Distortion*; ASM International, Metals Park, OH, USA, 1989.

[BK88] D.Bierman and P.Kamsteeg; Elicitation of Knowledge for Intelligent Tutoring Systems; Technical Report, University of Amsterdam, Netherlands, 1988.

[BKKPV03] S.Brecheisen, H.Kriegel, P.Kroger, M.Pfeifle and M.Viermetz; Representatives for Visually Analyzing Cluster Hierarchies; In *KDD-MDM*, August 2003.

[BL04] A.Banerjee and J.Langford; An Objective Evaluation Criterion for Clustering; In *ACM SIGKDD*, pages 515–520, August 2004.

[BM03] M.Bilenko and R.Mooney; Adaptive Duplicate Detection using Learnable String Similarity Measures; In *KDD*, pages 39–48, August 2003.

[BMP01] O.Buyukkokten H.Garcia Molina and A.Paepcke; Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices; In *WWW10*, pages 652–662, May 2001.

[C97] W.Callister Jr.; *Materials Science and Engineering: An Introduction*; John Wiley and Sons Inc., New York, USA, 1997.

[CN04] L.Chen and R.Ng; On the Marriage of Lp-Norm and Edit Distance; In *International Conference on Very Large Data Bases*, pages 792 – 803, August 2004.

[CO03] L.Chen and M.T. Ozsu; Similarity-based Retrieval of Time-Series Data Using Multi-Scale Histograms; Technical Report CS-20003-31, University of Waterloo, Canada, September 2003.

[DM00] G.Das and H.Mannila; Context-Based Similarity Measures for Categorical Databases; In *PKDD*, pages 201–210, September 2000.

[DMR98] G.Das, H.Mannila and P.Ronkainen; Similarity of Attributes by External Probes; In *KDD*, pages 23–29, August 1998.

[DWD97] S.Dutta, B.Wieranga and A.Dalebout; Case-Based Reasoning Systems: From Automation to Decision Aiding and Stimulation; *IEEE Transactions on Knowledge and Data Engineering*, 9(6):911–922, 1997.

[F55] J.Fourier translated by A.Freeman; *The Analytical Theory of Heat*; Dover Publictaions Inc., NY, USA, 1955.

[F58] R.Friedberg; A Learning Machine: Part I; *IBM Journal*, 2:1–13, 1958.

[FFC00] A.Freborg, B.Ferguson and M.Callabresi; Using Computer Simulation to Improve a Carburization Process; In *20th ASM Heat Treating Conference*, pages 70–78, 2000.

[FGOT03] F.Francesca G.Gordano R.Ordale and A.Tagarelli; Distance-based Clustering of XML Documents; In *1st International Workshop on Mining Graphs, Trees and Sequences*, pages 75–78, September 2003.

[FHKH02] E.Frank,G.Holmes R.Kirkby and M.Hall; Racing Committees for Large Datasets; In *International Conference on Discovery Science*, pages 153–164, Nov 2002.

[HALLN00] D.Huang, K.Arimoto, K.Lee, D.Lambert and M.Nazaraki; Prediction of Quench Distortion on Steel Shaft with keyway by Computer Simulation; In *20th ASM Heat Treating Society Conference*, pages 708–712, October 2000.

[HAK00] A.Hinneburg, C.Aggarwal and D.Keim; What is the Nearest Neighbor in High Dimensional Spaces; In *26th VLDB Conference*, pages 506–515, August 2000.

[HH92] J.Hasson and E.Houghton; Quenchants: Yesterday, Today and Tomorrow. In *1st International Conference on Quenching and Control of Distortion*, pages 13–17, September 1992.

[HH01] J.Helfman and J.Hollan; Image Representations for Accessing and Organizing Web Information; In *SPIE International Society for Optical Engineering Internet Imaging II Conference*, pages 91–101, 2001.

[HK01] J.Han and M.Kamber; *Data Mining: Concepts and Techniques;* Morgan Kaufman Publishers, San Francisco, California, USA, 2001.

[HK01] J.Han and M.Kamber; *Data Mining: Concepts and Techniques*; Morgan Kaufman Publishers, San Francisco, California, USA, 2001.

[HLHM00] M.Hunkel, T.Lubben, F.Hoffman and P.Mayr; Simulation of Transformation Behavior of Steels during Quenching; In *5th ASM Heat Treatment and Surface Engineering Conference*, pages 43–53, 2000.

[JP03] P.Janecek and Pearl Pu; Opportunistic Search with Semantic Fisheye Views; Technical Report IC/2004/42, Swiss Federal Institute of Technology, Lausanne (EPFL), 2003.

[K93] J.Kolodner; *Case-Based Reasoning*; Morgan Kaufmann Publishers, 1993.

[KB04] D.Keim and B.Bustos; Similarity Search in Multimedia Databases; In *International Conference on Data Engineering*, pages 873–874, March 2004.

[KK95] H.Kim and G.Koehler; Theory and Practice of Decision Tree Induction; *Omega*, 23(6):637–652, December 1995.

[KR03] J.Kang and Y.Rong; Numerical Simulation of Heat Transfer in Loaded Heat Treatment Furnaces; In *2nd International Conference on Thermal Process Modeling and Simulation*, April 2003.

[KR94] L.Kaufman and P.Rousseeuw; *Finding Groups in Data: An Introduction to Cluster Analysis*; TMS Publishers, 1994.

[KR94] L.Kaufman and P.Rousseeuw; *Finding Groups in Data: An Introduction to Cluster Analysis*; TMS Publishers, 1994.

[L96] D.Leake; *Case-Based Reasoning: Experiences, Lessons and Future Directions*; AAAI Press, 1996.

[LKW95] D.Leake, A.Kinley and D.Wilson; Learning to Improve Case Adaptation by Introspective Reasoning and CBR; In *International Conference on Case-Based Reasoning*, 1995.

[LHM98] B.Liu, W.Hsu and Y.Ma; Integrating Classification and Association Rule Mining; In *ACM SIGKDD*, pages 80–86, Aug 1998.

[LSW97] B.Lent, A.Swami and J.Widom; Clustering Association Rules; In *ACM SIGKDD*, August 1997.

[LVKR02] Q.Lu, R.Vader, J.Kang and Y.Rong; Development of a Computer-Aided Heat Treatment Planning System; *Heat Treatment of Metals*, 3:65–70, 2002.

[LVKR02] Q.Lu, R.Vader, J.Kang and Y.Rong; Development of a Computer-Aided Heat Treatment Planning System; *Heat Treatment of Metals*, 3:65–70, 2002.

[M67] J.MacQueen; Some Methods for Classification and Analysis of Multivariate Observations; In *5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 1:281–297, 1967.

[M95] A.Mills; *Heat and Mass Transfer*; Richard Irwin Inc., 1995.

[M97] T.Mitchell; *Machine Learning*; WCB McGraw Hill, USA, 1997.

[MCMMS02] M.Maniruzzaman, J.Chaves, C.McGee, S.Ma and R.Sisson Jr.; CHTE Quench Probe System: A New Quenchant Characterization System; In *5th International Conference on Frontiers in Design and Manufacturing*, pages 13–17, July 2002.

[MCMMS02] M.Maniruzzaman, J.Chaves, C.McGee, S.Ma and R.Sisson Jr.; CHTE Quench Probe System: A New Quenchant Characterization System; In *5th International Conference on Frontiers in Design and Manufacturing*, pages 13–17, July 2002.

[MMS02] S.Ma, M.Maniruzzaman and R.Sisson Jr. Characterization of the Performance of Mineral Oil Based Quenchants using the CHTE Quench Probe System; In *1st International Surface Engineering Congress and the 13th IFHTSE Congress*, October 2002.

[NM01] T.Nomoto and Y.Matsumoto; A New Approach to Unsupervised Text Summarization; In *ACM SIGIR*, pages 26–34, September 2001.

[PK97] K.Pal and J.Campbell; 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):48–63, 1997.

[PG60] D.Poirier and G.Geiger; *Transport Phenomena In Materials Processing*; John Wiley and Sons, 1960.

[PGF02] C.Palmer, P.Gibbons and C.Faloustos; ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs; In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2002.

[PJFC99] R.Pakalapati, L.Jin, T.Farris and S.Chandrasekar; Simulation of Quenching of Steels: Effect of Different Multiphase Constitute Models; In *19th ASM Heat Treating Society Conference Including Steel Heat Treating in the new Millenium*, pages 416–423, November 1999.

[PN96] J.Pritchard and G.Nurnberg; Computer Modeling of Pressure Gas Quenching in Vaccuum Furnaces; In *16th ASM Heat Treating Conference*, pages 399–404, 1996.

[PNC99] J.Petrucelli, B.Nandram and M.Chen; *Applied Statistics for Engineers and Scientists*; Prentice Hall, 1999.

[Q86] J.R.Quinlan; Induction of Decision Trees; *Machine Learning*, 1:81–106, 1986.

[Q86] J.R.Quinlan; Induction of Decision Trees; *Machine Learning*, 1:81–106, 1986.

[R87] J.Rissanen; Stochastic Complexity and the MDL Principle; *Econometric Reviews*, 6:85–102, 1987.

[R87] J.Rissanen; Stochastic Complexity and the MDL Principle; *Econometric Reviews*, 6:85–102, 1987.

[R87] J.Rissanen; Stochastic Complexity and the MDL Principle; *Econometric Reviews*, 6:85–102, 1987.

[RN95] S.Russell and P.Norvig; *Artificial Intelligence: A Modern Approach*; Prentice Hall, USA, 1995.

[S60] G.Stolz Jr; *Heat Transfer*; John Wiley and Sons, 1960.

[SM81] E.Smith and D.Medin; *Categories and Concepts*; Harvard University Press, 1981.

[SMMV04] R.Sisson Jr., M.Maniruzzaman, S.Ma and A.Varde; Quenching - Understanding, Controlling and Optimizing the Process; In *CHTE Seminar*, October 2004.

[T02] J.Turner; *MySQL and JSP Web Applications*; Sams Publishing, Indianapolis, IN, USA, 2002.

[TBC93] G.Totten C.Bates and N.Clinton; *Handbook of Quenchants and Quenching Technology*; ASM International, 1993.

[TTPF01] A.Traina, C.Traina, S.Papadimitriou and C.Faloutsos; Tri-plots: Scalable Tools for Multidimensional Data Mining; In *ACM SIGKDD*, pages 184–193, 2001.

[VRRMS0805] A.Varde, E.Rundensteiner,C.Ruiz, M.Maniruzzaman and R.Sisson Jr.; Learning Semantics-Preserving Distance Metrics for Clustering Graphical Data; In *ACM SIGKDD's Multimedia Data Mining Workshop*, pages 107–112, August 2005.

[VRRMS1005] A.Varde, E.Rundensteiner, C.Ruiz, M.Maniruzzaman and R.Sisson Jr.; The LearnMet Approach for Learning Domain-Specific Distance Metrics to Preserve the Semantic Content in Mining Graphical Data; Technical Report CS-CHTE-10-2005, CHTE, WPI, Worcester, MA, USA, October 2005.

[VRRMS06] A.Varde, E.Rundensteiner, C.Ruiz, M.Maniruzzaman and R.Sisson Jr.; LearnMet: Learning Domain-Specific Distance Metrics for Plots of Scientific Functions; *Journal of Multimedia Technology and Applications (MTAP) Special Issue on Multimedia Data Mining*, 2006.

[VRRBMS0606] A.Varde, E.Rundensteiner, C.Ruiz, D.Brown, M.Maniruzzaman and R.Sisson Jr.; Effectiveness of Domain-Specific Cluster Representatives for Graphical Plots; In *ACM SIGMOD'S IQIS*, June 2006.

[VRRBMS06] A.Varde, E.Rundensteiner, C.Ruiz, D.Brown, M.Maniruzzaman and R.Sisson Jr.; Integrating Clustering and Classification for Estimating Process Variables in Materials Science; In *AAAI*, July 2006.

[VRRBMS1106] A.Varde, E.Rundensteiner, C.Ruiz, D.Brown, M.Maniruzzaman and R.Sisson Jr.; Designing Semantics-Preserving Cluster Representatives for Scientific Input Conditions; In *CIKM*, November 2006.

[VTRWMS03] A.Varde M.Takahashi E.Rundensteiner M.Ward M.Maniruzzaman and R.Sisson Jr.; QuenchMiner^{TM}: Decision Support for Optimization of Heat Treating Processes; In *IEEE IICAI*, December 2003.

[VTRWMS03] A.Varde, M.Takahashi, E.Rundensteiner, M.Ward, M.Maniruzzaman and R.Sisson Jr.; QuenchMiner^{TM}: Decision Support for Optimization of Heat Treating Processes; In *IEEE IICAI*, December 2003.

[WF00] I.Witten and E.Frank; *Data Mining: Practical Machine Learning Algorithms with Java Implementations.*; Morgan Kaufmann Publishers, 2000.

[XNJR03] E.Xing, A.Ng, M.Jordan and S.Russell; Distance Metric Learning with Application to Clustering with Side Information; In *Neural Information Processing Systems*, pages 505–512, December 2003.

[ZWT02] Z.Zhou, J.Wu and W.Tang; Ensembling Neural Networks: Many Could Be Better Than All; *Artificial Intelligence*, 137(1-2):239–263, 2002.