

An incremental approach for calculating dominance-based rough set
dependency



Author

Rana Muhammad Kaleem Ullah

FALL 2016-MS-16(CSE) 00000171489

MS-16 (CSE)

Supervisor

Dr. Usman Qamar

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
ISLAMABAD
AUG, 2020

An incremental approach for calculating dominance-based rough set
dependency

Author

Rana Muhammad Kaleem Ullah

FALL 2016-MS-16(CSE) 00000171489

A thesis submitted in partial fulfillment of the requirements for the degree of
MS Software Engineering

Thesis Supervisor:

Dr. Usman Qamar

Thesis Supervisor's Signature: _____

Thesis Co-Supervisor's Signature: _____

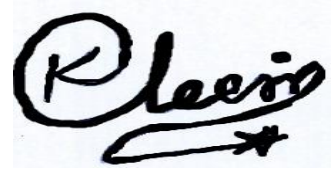


DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD

AUG, 2020

DECLARATION

I certify that this research work titled “*A novel incremental approach for calculating dominance-based rough set dependency*” is my work under the supervision of Dr. Usman Qamar. This work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged/referred.

A handwritten signature in black ink, appearing to read 'Rana Muhammad Kaleem Ullah', with a stylized flourish at the end.

Signature of Student

Rana Muhammad Kaleem Ullah

FALL 2016-MS-16(CSE) 00000171489

LANGUAGE CORRECTNESS CERTIFICATE

This thesis is free of typing, syntax, semantic, grammatical and spelling mistakes. The thesis is also according to the format given by the University for MS thesis work.



Signature of Student

Rana Muhammad Kaleem Ullah

FALL 2016-MS-16(CSE) 00000171489

Signature of Supervisor

Dr. Usman Qamar



Signature of Co-Supervisor

Dr. Summair Raza

COPYRIGHT STATEMENT

- Copyright in the text of this thesis rests with the student author. Copies (by any process) either in full or of extracts, may be made only in accordance with instructions given by the author and lodged in the Library of NUST College of E&ME. Details may be obtained by the Librarian. This page must form part of any such copies made. Further copies (by any process) may not be made without the permission (in writing) of the author.
- The ownership of any intellectual property rights which may be described in this thesis is vested in NUST College of E&ME, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the College of E&ME, which will prescribe the terms and conditions of any such agreement.
- Further information on the conditions under which disclosures and exploitation may take place is available from the Library of NUST College of E&ME, Rawalpindi.

ACKNOWLEDGEMENTS

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You put in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout every department of my life.

I would also like to express my gratitude to my supervisor **Dr. Usman Qamar** for his constant motivation and help throughout this thesis. Also, for Advance Software Engineering (ASE) and Data Engineering (DE) courses which he has taught me. I can safely say that I haven't learned any other engineering subject in such depth than the ones which he has taught.

I would like to pay special thanks to **Dr. Summair Raza** for his incredible cooperation and for providing help at every phase of this thesis. He has guided me and encouraged me to carry on and has contributed to this thesis with a major impact. Thank you for guiding me, often with big doses of patience.

I would also like to thank my Guidance Committee Members **Dr. Saad Rehman** and **Dr. Wasi Haider Butt** for being on my thesis guidance and evaluation committee. Some special words of gratitude go to my friends **Muhammad Sufian, Faisal Masood and Zirak Khan** who has always been a major source of technical support and cooperation when things would get a bit discouraging. Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

*Dedicated to my exceptional parents and whole family whom
tremendous support, cooperation and their prayers led me to this
wonderful accomplishment.*

ABSTRACT

Feature selection and classification are widely used in machine learning to handle the immense amount of data. In many datasets, conditional attributes and decision classes are preference-ordered and to perform feature selection on these types of datasets, an extension of rough set theory (RST) is used which is known as a dominance-based rough set approach (DRSA). A dominance-based rough set approach follows a dominance principle which states that objects relating to a certain decision class must follow the preference order and this preference order states that an object having higher values of conditional attributes must have higher decision class. The dependency measure of a dataset is used in DRSA to calculate the suitable reducts of a dataset. The conventional DRSA uses lower and upper approximations to calculate the dependency of the dataset. The shortcomings of this conventional method of dependency calculation are high complexity and huge utilization of computational resources. This paper proposes a novel methodology named as “Incremental Dominance-based Dependency Calculation” (IDDC) to mitigate the aforementioned problems regarding the conventional approach of dependency calculation. The proposed methodology uses an incremental approach to find the dependency of datasets by scanning the data records one-by-one and comparing each record with every other record in the dataset. For comparison of records, IDDC uses a set of proposed dominance-based dependency classes. To justify the proposed approach, both IDDC and conventional approaches are compared using various datasets from the UCI dataset repository. Results have shown that the proposed approach outperforms the conventional approach by depicting on average 46% and 98% decrease in execution time and required runtime memory, respectively.

Keywords: Dominance-Based rough set approach (DRSA), Incremental Dominance-based dependency calculation Method (IDDC), Dependency classes, Rough set theory (RST), Lower Approximations, Upper Approximations, Reducts, Fast Reduct Generating Algorithm (FRGA), UCI repository.

TABLE OF CONTENTS

DECLARATION	i
LANGUAGE CORRECTNESS CERTIFICATE.....	ii
COPYRIGHT STATEMENT.....	iii
ACKNOWLEDGEMENTS	iv
ABSTRACT	vi
LIST OF FIGURES.....	ix
LIST OF TABLES.....	x
CHAPTER 1: INTRODUCTION.....	12
1.1. Background Study	12
1.2. Feature Selection and Rough Set Theory.....	14
1.3. Dominance based Rough Set Approach (DRSA)	17
1.4. Problem Statement.....	18
1.5. Proposed Methodology	19
1.6. Research Contribution	20
1.7. Thesis Organization	21
CHAPTER 2: DOMINANCE-BASED ROUGH SET APPROACH.....	24
2.1. Main Concept	24
2.2. Dominance	26
2.3. Decision Classes and Class Unions	27
2.4. Approximations.....	28
2.4.1. Lower Approximations.....	28
2.4.2. Upper Approximations	29
2.5. Dependency / Quality of Sorting.....	31
CHAPTER 3: LITERATURE REVIEW.....	34
3.1. Background	34
3.2. Research Sequence.....	35
3.2.1. Research Questions	35
3.2.2. Inclusion/Exclusion Criteria	35
3.2.3. Keywords	35
3.3. Related Work	36
CHAPTER 4: CHALLENGES IN DRSA AND PROPOSED SOLUTION	42
4.1. Challenges in DRSA.....	42

4.2. Proposed Methodology	45
CHAPTER 5: IMPLEMENTATION AND VALIDATION	55
5.1. Datasets and Algorithms	55
5.1.1. Fast Reduct Generating Algorithm	57
5.1.2. Genetic Algorithm.....	58
5.1.3. Particle Swarm Optimization Algorithm.....	60
5.2. Case Study using Conventional Approximation’s Methodology	62
5.3. Case Study using Proposed Methodology	67
CHAPTER 6: COMPARATIVE ANALYSIS.....	78
6.1. Comparison Framework	78
6.1.1. Execution Environment.....	78
6.1.2. Accuracy	78
6.1.3. Execution Time.....	80
6.1.4. Memory Usage.....	84
6.1.5. Complexity.....	89
CHAPTER 7: CONCLUSION AND FUTURE WORK.....	93
REFERENCES	94

LIST OF FIGURES

Figure 1:KDD Process	12
Figure 2: Complete Thesis Flow	22
Figure 3: Pseudocode of 1 st Step for calculating lower approximation.	42
Figure 4: Pseudocode of 2 nd Step for calculating lower approximation.	44
Figure 5: Pseudocode of Step-3 for calculating lower approximation	44
Figure 6: Pseudocode of the proposed methodology	53
Figure 7: Pseudocode of FRGA Algorithm	58
Figure 8: Selected chromosomes for ordered based crossover method	58
Figure 9:Chromosomes after ordered based crossover	59
Figure 10: Chromosome after bit flip mutation	59
Figure 11: Chromosome after adjacent two mutation.....	59
Figure 12: A Typical Flow Chart of Genetic Algorithm	60
Figure 13: Pseudocode of Particle Swarm Optimization (PSO).....	62
Figure 14: Graphical Comparison of Execution time	84
Figure 15: Big-O Complexity Comparison.....	91

LIST OF TABLES

Table I: Example of Information Table	25
Table II: Decision System.....	26
Table III: Comparisons of techniques used in different papers	38
Table IV: Comparison Possibilities of two objects.....	45
Table V: Dataset information.....	56
Table VI: Dominance Relation Matrix of Table II	63
Table VII: Dominance Relation Matrix of Table II for proposed method	68
Table VIII: Number of comparison statements	76
Table IX: Comparison of Size of AlphaSet	79
Table X: Comparison of the number of Reducts Generated.....	79
Table XI: Comparison of Execution time	83
Table XII: Comparison of Memory Usage	88
Table XIII: Percentage Reduction in Memory Usage.....	88

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

This section provides a detailed introduction to the research and research concepts. This section is organized in multiple sub-sections. **Section 1.1** provides the background study, fundamentals of rough set approach are discussed in **Section 1.2**, **Section 1.3** presents the basic understanding of the dominance-based rough set approach, **Section 1.4** presents the problem statement of research, **Section 1.5** discuss the proposed methodology, **Section 1.6** gives the detail about research contribution, and thesis organization is presented in **Section 1.7**.

1.1. Background Study

Knowledge is only valuable when it can be used efficiently and effectively; therefore, knowledge management is increasingly being recognized as a key element in extracting its value. An example of this is a Knowledge Discovery in Databases (KDD)[1]. Traditionally, data was turned into knowledge employing manual analysis and interpretation. For many applications, this form of manual probing of data is slow, costly, and highly subjective. Indeed, as data volumes grow dramatically, this type of manual data analysis is becoming completely impractical in many domains. This motivates the need for filtering the data. The steps involved in the knowledge discovery process are discussed below and Figure 1 gives its pictorial view.

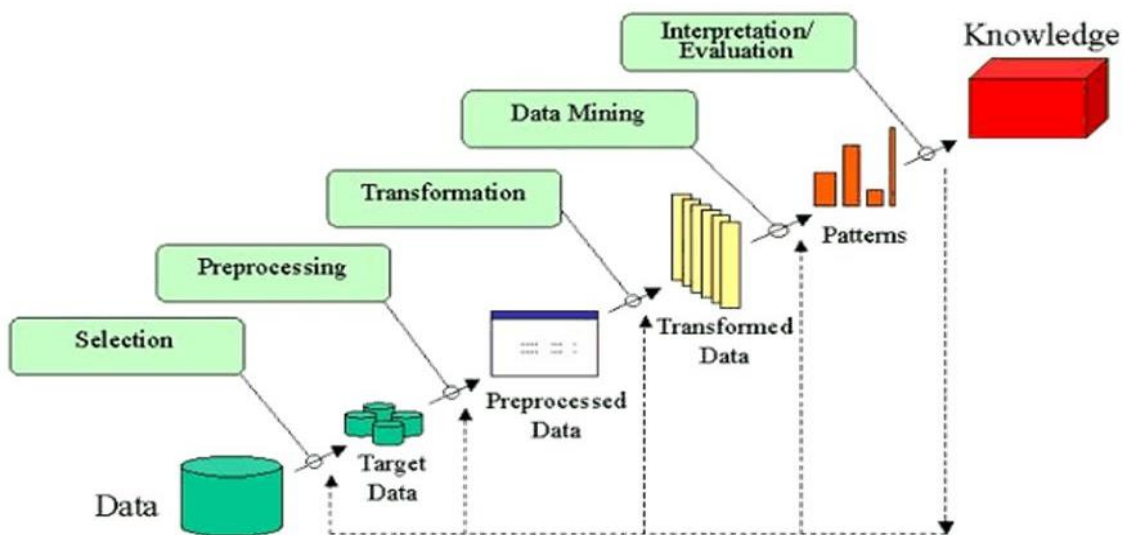


Figure 1: KDD Process

- **Data Selection**

Data selection comprises selecting the data for knowledge discovery. This may require selecting the data from an existing repository or creating a single source of data (a new repository) from multiple sources. The data is selected based on the analysis task. This is an important step where all the data relevant to analysis should be considered, failed to do so may lead to failure of the entire process.

- **Data Cleansing/ Pre-processing**

This step refers to the increasing reliability and accuracy of data. The majority of the times, the selected data may contain records that are potential outliers, may contain insufficient details (e.g. missing attribute values), noise or incorrect values, etc. Using such data may lead to incorrect models, may affect classification accuracy or performance of induction algorithms, etc at later stages. Data cleansing or pre-processing refers to the removal of all such factors to enhance the quality and reliability of selected data. There are many techniques for data cleansings. We may use outlier detection algorithms to find out outliers.

- **Data Transformation/Reduction**

This step refers to transforming the data to make it appropriate for underlying analysis and knowledge discovery. The data may contain redundant attributes that do not add much to our information or may contain irrelevant attributes. Such attributes are removed at this stage. Various techniques are used at this stage e.g. feature selection, feature extraction, attribute discretization, etc. The basic purpose is to transform/reduce the data to enhance performance at later stages.

- **Data Mining**

Once the data is ready, we can apply our data mining algorithms to discover the hidden information/patterns from our data. The use of a particular mining algorithm depends on the nature of the analysis and goal of the knowledge discovery e.g. either we want prediction or description based on data?

- **Interpretation/Evaluation**

Once the knowledge has been discovered (patterns have been identified), it is evaluated based on our defined goals to validate accuracy, usefulness, novelty, etc. It should be noted that we may need to repeat previous steps to enhance the above-mentioned measures, e.g. by including a greater number of features and repeating the steps.

This research focuses on the third step i.e. data reduction of Knowledge Discovery in Datasets process. The size of a dataset comprises two perspectives i.e. number of distinct samples to be processed per dataset and the number of attributes per sample. The former only affects the training process in data mining, depending on its use, however, the latter i.e. number of attributes per sample also called dimensionality, affects the training process as well as the performance of an algorithm. Many algorithms exhibit non-polynomial execution time with respect to dimensionality.

A large number of dimensions in a dataset lead to a phenomenon called the curse of dimensionality. The term was first coined by Bellman [2] resulting out of the volume increase by adding extra dimensions to mathematical space. Curse of dimensionality is the problem faced by many data analysis algorithms for their practical implementation on datasets with the larger size. As already mentioned, the performance of data mining algorithms is inversely proportional to the dimensionality of datasets, so higher dimensionality not only challenges the performance of such algorithms but makes their implementation impractical for many real-life applications where datasets increase beyond smaller size.

1.2.Feature Selection and Rough Set Theory

In a dataset, against each instance, there is a big count of features. Not all features of an instance are important. Some features are important and can give complete information. This subset of features can easily replace the whole dataset with all the features [3]. Feature selection is a way to reduce the number of features by applying different techniques. The selected feature set can be easily used against the entire dataset. An efficient feature selection algorithm selects features that can give complete, accurate and important information [4].

There are two kinds of features. Strongly relevant features are those which play an important role in predictions. Weakly relevant features are the second type. These features cannot contribute much to the accuracy of the dataset[5]. There are two major feature selection techniques.

- Transformation based techniques are used when underlying semantics are not important because this technique destroys the semantics. Examples of transformation based techniques are MDS [6] and PCA [7].

- Selection based techniques preserve semantics and the original meaning of datasets remains the same. Rough Set Theory (RST) is the most effective and common technique in selection based techniques [8].

In 1982, Pawlak proposed Rough Set Theory (RST) to analyze the incomplete, inconsistent, and unknown information or data [9,10]. RST has become an important tool to find data reduction, data dependencies, rule induction, and approximate set classification from databases [11]. Since its inception, RST has been comprehensively used for knowledge discovery in economy and finance[12], medical imaging etc.[13]. However, RST is unable to generate accurate results for preference-ordered datasets and this makes classical RST unsuitable for preference-ordered data domains. Some of the applications of preference-ordered data domains are student performance evaluation, website rating system, etc.

RST has become a topic of great interest over the past ten years and has been successfully applied to many domains by researchers. For a given dataset it is possible to find out a smaller attribute set (called reduct) that contains most of the information. So, attributes other than the reduct set can be removed from the dataset with minimal information loss. Pawlak has proposed RST for knowledge discovery in datasets. In contrast to conventional discrete sets, RST is based on the concepts of upper and lower approximations as discussed below.

Most of the sets cannot be identified unambiguously, so we use approximation. For an information system where $B \subseteq A$, we can approximate the decision class X by using the information contained in B . The lower and upper approximations are defined as follows [14]:

$$X: \underline{B}X = \{x | [x]_B \subseteq X\} \tag{1}$$

$$X: \overline{B}X = \{x | [x]_B \cap X \neq \emptyset\} \tag{2}$$

Lower approximation defines the objects that are members of X with respect to information in “ B ”. Upper approximation on the other hand contains objects that with respect to “ B ” can be members of “ X ”. The boundary region defines the difference between lower and upper approximations.

$$X: BN_B(X) = \overline{BX} - \underline{BX} \quad (3)$$

One way of dimensional reduction is keeping only those attributes that preserve the indiscernibility relation i.e. classification accuracy. Using a selected set of attributes provides the same set of equivalence classes that can be obtained by using the entire attribute set. The remaining attributes are redundant and can be reduced without affecting classification accuracy. There are normally many subsets of such attributes called reducts. Calculating the reducts comprises of two steps. First, we calculate the dependency of the decision attribute on the entire dataset. Normally this is “1”, however, for inconsistent datasets, this may be any value between “0” and “1”. In the second step, we try to find the minimum set of attributes on which decision attribute has the same dependency value as that of its value on the entire set of attributes. In this step, we may use any Rough Set based feature selection algorithm. It should be noted that there may be more than one reduct sets in a single dataset.

The reduct set must be optimal i.e. it should contain a minimum number of attributes to better realize its significance, however, finding optimal reduct is a difficult task as it requires exhaustive search with a greater number of resources. Normally exhaustive algorithms are used to find reducts in smaller datasets, however, for datasets beyond smaller size, the other category of algorithms i.e. random or heuristics-based search is used, but the drawback of these algorithms is that they do not produce the optimal result. So, getting the optimal reducts is a trade-off between the resources and reduct size. The Core is another important concept in Rough Set Theory. Normally the reduct set is not unique in a dataset i.e. we may have more than one reduct set. Although the Reduct may contain the same amount of information otherwise represented by the entire attribute subset, even in reduct some attributes are more important than others i.e. these attributes cannot be removed without affecting the classification accuracy of the reducts. Mathematically it can be written as $Core = \bigcap_{i=1}^n R_i$ where R_i is i^{th} Reduct Set. So, the Core is the attribute or set of attributes common to all reduct sets.

RST provides many concepts to thoroughly analyze datasets and find irrelevant and redundant features. Given a dataset with discretized attribute values, it is possible to find a subset of the original attributes using RST that are the most informative: all other attributes can be removed from the dataset with minimum information loss. Unlike statistical correlation-reducing

approaches, this requires no human input or intervention. Most importantly, it also retains the semantics of the data, which makes the resulting models more transparent to human scrutiny.

In a dataset, there may be redundant attributes that may be eliminated without much of the essential information loss. Rough sets [3] let us define strong and weak relevance levels, so that redundant attributes may be removed. The concept of the reduct is fundamental in RST. Being a subset of attributes, it can distinguish all the objects in a dataset that are discernible with respect to the entire attribute set. Both reduct and core are important concepts that are used in feature selection and dimensionality reduction. But there are some limitations to the RST one of them is that it cannot work properly with the dataset having dominance relation between the records of data. Now to handle this dominance-based or multi-criteria dataset another form of RST was introduced which is known as Dominance-based rough set theory (DRSA).

1.3. Dominance based Rough Set Approach (DRSA)

Therefore, to process the data and information based on preference-ordered attributes, Greco et al. [15] have introduced the Dominance-based Rough Set Approach (DRSA) in contrast with RST, DRSA considers dominance relation between records and can process the information with preference-ordered attribute domains. In DRSA, considering attributes with preference-ordered domains, the conceptions to be characterized are upward and downward unions of classes rather than the particular classes. The basic idea behind the dominance-based rough set approach is to replace the equivalence relation in the Pawlak's rough set theory with a dominance relation, which authorizes taking into account the preference order in the value set of the criteria. In recent decades, due to its ability to process information using preference-ordered domains, DRSA has been widely used in many real-life applications such as rural sustainable development potentialities evaluation [16], airline services evaluation [17–19], group decision [20], etc.

In real-life applications, consideration of preference order and handling such inconsistencies becomes critical and DRSA which deduces the theory by replacing the indiscernibility relation of classical RST with dominance relation offers several advantages in this regard. It should be noted that the majority of the algorithms based on DRSA use dependency as an underlying criterion measure for different tasks. However, calculating dependency by using the conventional DRSA approach requires the calculation of lower and upper approximations, and this calculation makes

conventional DRSA computationally too expensive to be used for datasets beyond smaller size. DRSA is going to be explained in detail in the next chapter as we move along.

1.4.Problem Statement

As we discussed earlier that calculating lower approximations and upper approximations is computationally too expensive that affects the performance of the algorithms using these measures. This calculation has three main steps. Calculating the first step requires the calculation of class unions which is Cl_t^{\geq} or Cl_t^{\leq} structure based on which approximation we need to compute. For calculating those class unions, we have to traverse through complete Universe from 1 to length of $|U|$, which can be time taking and computationally expensive for larger datasets. In the case of our example, we have to repeat this step seven times but this can easily reach thousands of times with larger Universe and can cause much trouble computationally. Class unions for our example which are shown in Table II are Cl_1^{\leq} , Cl_2^{\leq} , Cl_3^{\leq} and Cl_4^{\leq} .

$$Cl_1^{\leq}(x) = \{X_3, X_4, X_7\}$$

$$Cl_2^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$Cl_2^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$$

$$Cl_2^{\geq}(x) = \{X_2\}$$

Now, in Step 2 of this conventional method of approximation calculation, we have to calculate D_P^+x and D_P^-x , this step requires two loops for its implementation one will be from 1 to the cardinality of Cl_t^{\geq} to get objects one by one from the set of class union which is obtained in step 1 and the second loop will be from 1 to cardinality of Universe to compare that selected object from loop one with all the members of Universe. This step is repeated for all the objects of all the class unions we acquired in Step 1.i-e loops for $Cl_1^{\leq}(x)$ will run $3 * 7 = 21$ times, where 3 is the cardinality of $Cl_1^{\leq}(x)$ and 7 is the cardinality of the Universe in our example. So, Step 2 is much more complex and needs more time for computation than step 1.

Finally, in Step 3 after the calculation of D_P^+x and D_P^-x , $\underline{P}(Cl_t^{\geq})$ and $\overline{P}(Cl_t^{\geq})$ is to be calculated which are the lower approximations and upper approximation of class unions and it is done for all the class unions. For this step, three loops are required. The first loop is from 1 to the cardinality

of Cl and the second loop is from 1 to cardinality of D_p^+x or D_p^-x . These two loops are used to reach the indexes of two-dimensional arrays of D_p^+x or D_p^-x . After that, we have to compare these indexes with the class unions to check if intersection and subset property hold. For this purpose, the third loop from 1 to cardinality of the class union is used. Nested loops increase the execution time and computational complexity of the conventional approach.

It is evident that the conventional approach carries a grave challenge to the performance of the algorithms when it is about huge datasets. To overcome all the above-described issues, we have proposed a solution that will help in reducing the computational complexity of calculating the quality of sorting without using approximations approach. This will contribute to an overall reduction in the computational time of DRSA. Therefore, this would be an efficient method to calculate the quality of sorting which helps further in finding the reducts.

1.5. Proposed Methodology

To overcome these problems, a new approach has been proposed in this paper which calculates the dominance-based rough set dependency measure without calculating the lower and upper approximations. The proposed methodology is called the “**Incremental Dominance-based Dependency Calculation Method**” (IDDC). To justify its flexibility and effectiveness, the proposed approach is implemented using the Fast Reduct Generating Algorithm (FRGA). The reason for selecting feature selection is that **it is the common pre-process used to complete various tasks and the majority of DRSA based feature selection algorithms use dependency measure as criteria to select features**. Results have shown that the proposed approach is more efficient and effective as compared to the conventional method of dependency calculation without affecting the accuracy of the algorithms. It is valuable to use the feature selection algorithms with the proposed methodology because of the following advantages as compared to conventional approximation methods:

- IDDC effectively avoids the calculation of the upper and lower approximations and therefore it can also be used for large scale datasets.
- IDDC calculates the same dependency value of a dataset as calculated by the conventional approach.

- Experimental results have shown that as compared to the conventional approach, calculating dependency using IDDC has reduced the execution time by 46% on average for selected datasets.
- Experiments have also shown that calculating dependency using IDDC requires almost 98% less runtime memory on average as compared to the conventional dependency calculation approach.
- The overall performance of the algorithms using IDDC has been increased substantially.

The time saved by IDDC in feature selection approaches can be used to work on other machine learning tasks such as classification, clustering, or pattern recognition.

1.6. Research Contribution

Dominance-based Rough Set approach uses equivalence structures for calculating lower and upper approximations. The approximations are further used for performing different tasks during data analysis. Calculating equivalence class structures is a computationally complex job, so in this research, we have provided a heuristics-based approach for calculating both of these approximations. The heuristics-based approach calculates these approximations without calculating equivalence class structures and thus significantly enhancing the efficiency.

Similarly, Traditional dominance-based rough set-based approaches use a positive region-based dependency measure for the feature selection process. However, using a positive region is a computationally expensive approach that makes it inappropriate to use for large datasets. We have developed an alternate way to calculate dependency comprising of dependency classes. A dependency class is a heuristic which defines how the dependency measure changes as we scan new records during traversal of the dataset.

In this proposed method, we start from the first record and calculate the dependency or quality of sorting of the dataset based on dominance relation and decision attributes of records. Then after adding every single record the dependency or quality of sorting is refreshed based on comparisons classes. In our proposed method, we formulate six different comparison classes that provide a much efficient and simple way to find the quality of sorting without finding the approximations which can be a complex and tedious task.

Based on the proposed methodology, feature selection was performed by using feature selection algorithms. The positive region-based dependency calculation step in these algorithms was replaced with proposed IDDC. Results were compared with conventional ones and it was observed that proposed IDDC based feature selection algorithms provide the same accuracy with a substantial increase in overall performance. These are the main contributions of our research:

- Improved DRSA algorithm
- The less computational cost of DRSA algorithm
- Less memory consumption while using the DRSA algorithm
- Less time complexity of DRSA algorithm
- No compromise on accuracy

1.7.Thesis Organization

The overall thesis is structured as follows and Figure 2 also represents the organization of the thesis.

- Error! Reference source not found. deals with the introduction having detailed background study about the concepts used in the research, problem statement, research contribution and thesis organization.
- Error! Reference source not found.-**BASED ROUGH SET APPROACH (DRSA)** discusses the major concepts of the DRSA in detail. Each concept is explained by using formulas and examples.
- Error! Reference source not found. contains the literature review which provides a description of work done in the field of dominance-based rough set approach. In the Literature review, we also highlight the advantages and disadvantages of the different approaches that we encountered.
- Error! Reference source not found. explains the challenges that we face in the conventional approach and also covers the details of the proposed methodology that is used to mitigate the performance bottleneck of the conventional methodology.
- Error! Reference source not found. provide the implementation regarding the proposed methodology and selection of multiple datasets, different algorithms. Validation of the proposed methodology is also performed in this chapter using a case study.

- Error! Reference source not found. **ANALYSIS** contains a brief performance analysis of our proposed methodology and conventional methodology based on the experimental results of both methodologies.
- Error! Reference source not found. This section concludes the thesis. A summary of all of the findings along with an overview of future work is presented.

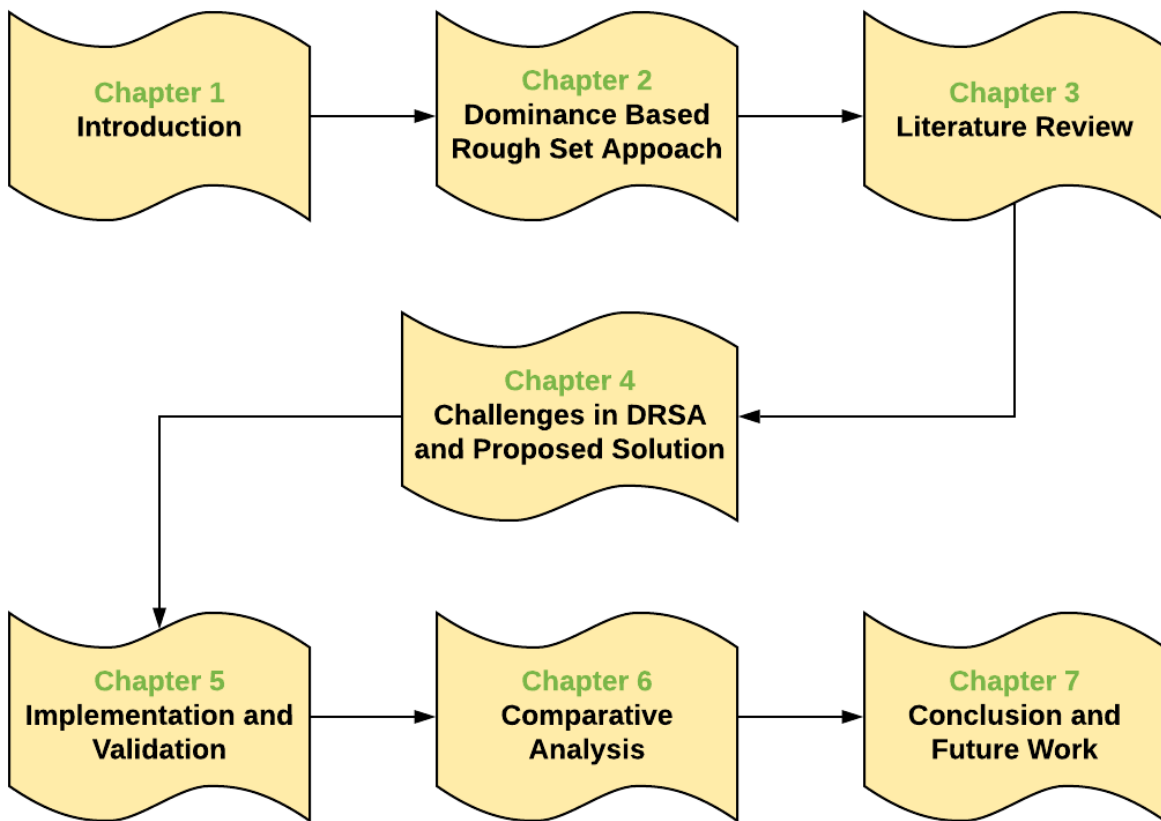


Figure 2: Complete Thesis Flow

Chapter 2

Dominance-based Rough Set Approach

CHAPTER 2: DOMINANCE-BASED ROUGH SET APPROACH

In this chapter, we will discuss major concepts regarding DRSA. In section 2.1 the main concept is elaborated. Further, in section 2.2, the dominance relation is discussed. In section 2.3, the concept of a decision class and class unions is described. In section 2.4, approximation and its types are explained with its calculation and in the last section 2.5, how to find the dependency of a dataset and then by using this dependency, how to calculate reducts and core attributes is explained.

2.1.Main Concept

As in conventional Rough Set Theory, there is a decision system which is a combination of a finite set of objects known as Universe (U). Each object is categorized by a set of special attributes known as conditional attributes (C) and decision attributes (D) . Mathematically it is shown in equation 4.

$$\alpha = (U, C \cup D) \quad (4)$$

In the context of DRSA, a decision table has four components and mathematically it is represented as shown in equation 5.

$$A = (U, Q, V, f) \quad (5)$$

Here, U is Universe and Q represents a finite set of criteria i.e. those attributes having an ordinal scale-based domain. $Q = (C \cup D)$ which means that both conditional attributes and decision attribute(s) are included in Q. Whereas, V can be mathematically represented as shown in equation 6.

$$V = U_{q \in Q} V_q \quad (6)$$

Here, V_q is the value set of criteria q . f in equation 5 represents a function of the form $f(x, q)$ which assigns a particular value V_q to an item x for attribute q . The sample decision system is shown in Table II.

In our sample decision system, Universe only contains a total of seven records i.e. $U = \{X_1, X_2, X_3 \dots \dots, X_7\}$. Here, conditional criteria include $\{Mathematics, English\}$ and the decision criterion is $\{Final-Result\}$. DRSA is an extended version of conventional RST that can be considered for knowledge gathering from non-ordinal datasets [21]. From the start of DRSA appearance, many domains have used it for better results such as multi-criteria web mining [22], fault diagnosis [23], in the manufacturing industry [24], finance projects [25,26], better project selection [27], and in data mining [28]. We have stated some core preliminaries of DRSA in the following sub-section to discuss the benefits, limitations, and the need for increased computational performance in conventional DRSA.

Table I: Example of Information Table

Universe	Mathematics	English
X₁	“A”	“B”
X₂	“A”	“A”
X₃	“B”	“C”
X₄	“A”	“B”
X₅	“B”	“A”
X₆	“A”	“B”
X₇	“C”	“B”

Table II: Decision System

Universe	Mathematics	English	Final-Result
X₁	A	B	Very Good
X₂	A	A	Excellent
X₃	B	C	Good
X₄	A	B	Good
X₅	B	A	Very Good
X₆	A	B	Very Good
X₇	C	B	Good

2.2.Dominance

For a set of criteria $P \subseteq C$, an item x dominates item y if item x is better than y on all the criterion in P i.e. $\{\forall q \in P, x \succcurlyeq q y\}$. It will be said that “ x dominates y ” which can be denoted by $D_p^-(x)$ and that can be defined mathematically as shown in equation 7. $D_p^-(x)$ is a set of items that are being dominated by x based on the information in $P \subseteq C$.

$$D_p^-(x) = \{y \in U: xD_p y\} \quad (7)$$

Similarly, the set of items which are dominating x are denoted by $D_p^+(x)$ and that can be defined mathematically as shown in equation 8.

$$D_p^+(x) = \{y \in U: yD_p x\} \quad (8)$$

For example, if we consider the item X_3 in Table I as our origin and $P = \{\text{Mathematics}\}$ then:

$$D_p^-(x_3) = \{x_3, x_5, x_7\}$$

Similarly,

$$D_p^+(x_3) = \{x_1, x_2, x_3, x_4, x_5, x_6\}$$

2.3. Decision Classes and Class Unions

In traditional RST, based on decision attributes the Universe is partitioned into a finite number of decision classes. Likewise in DRSA, decision attributes splits the whole Universe into a finite number of decision classes, $Cl = \{Cl_1, Cl_2, Cl_3, \dots, Cl_m\}$. Keep in mind that each item can be a part of only one decision class. Unlike the conventional RST, the decision classes in DRSA are supposed to be preference ordered. So, for $r, s = \{1, 2, 3, \dots, m\}$, an item from Cl_r is preferred over the item from Cl_s for $r > s$. Thus in place of simple approximation, as it is done in conventional RST, the approximations in DRSA are downward and upward unions based on decision classes. Mathematically these unions are shown in equations 9 and 10.

$$Cl_t^{\geq}(x) = \bigcup_{s \geq t} Cl_s \quad t = 1, \dots, n. \quad (9)$$

$$Cl_t^{\leq}(x) = \bigcup_{s \leq t} Cl_s \quad t = 1, \dots, n. \quad (10)$$

Here, $Cl_t^{\geq}(x)$ represents the set of items from class Cl_t or a more preferred class. However, $Cl_t^{\leq}(x)$ represents the set of items from class Cl_t or to a less preferred class. In Table II, the decision attribute ‘‘Final-Result’’ comprises three different decision classes which are labeled as ‘‘Excellent’’, ‘‘Very Good’’ and ‘‘Good’’. Here, class ‘‘Excellent’’ is preferred over ‘‘Very Good’’ which is preferred over ‘‘Good’’. For ease of use, we will consider *Excellent* = 3, *Very Good* = 2 and *Good* = 1. For example, based on data from Table II the $Cl_t^{\geq}(x)$ and $Cl_t^{\leq}(x)$ for $t = 2$ are following:

$$Cl_t^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$$

$$Cl_t^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$Cl_t^{\geq}(x)$ represents the set of items either from class ‘‘Very good’’ or better class such as ‘‘Excellent’’. Similarly, $Cl_t^{\leq}(x)$ represents the set of items relating to either class ‘‘Very good’’ or lower class such as ‘‘Good’’.

2.4.Approximations

An approximation is a key concept to find reducts and core. To decide whether an item belongs to a class of decision attribute or not, approximations are calculated. There are two kinds of approximations.

- Lower Approximation
- Upper Approximation

2.4.1. Lower Approximations

Lower approximations in traditional RST describe the set of those items that with certainty belong to a decision class based on certain conditional attributes. In DRSA, provided that $P \subseteq C$, P-lower approximation of $Cl_t^{\geq}(x)$ contains all those items which will, with certainty belong to $Cl_t^{\geq}(x)$. Likewise, the P-lower approximation of $Cl_t^{\leq}(x)$ will contain all those items that, with certainty belong to $Cl_t^{\leq}(x)$. Mathematically it is shown in equations 11 and 12.

$$\underline{P}(Cl_t^{\geq}) = \{x \in U: D_P^+(x) \subseteq Cl_t^{\geq}\} \quad (11)$$

$$\underline{P}(Cl_t^{\leq}) = \{x \in U: D_P^-(x) \subseteq Cl_t^{\leq}\} \quad (12)$$

Calculating P-lower approximation for both class unions Cl_t^{\geq} and Cl_t^{\leq} includes the following three steps which are described using data from Table II.

1st Step: All the items related to the upward class union Cl_t^{\geq} are identified in this step. It is similar to computing equivalence class structure with the help of decision attributes in traditional RST. In this example lower approximation $\underline{P}(Cl_t^{\geq})$ is calculated for $t = 2$ and class union Cl_t^{\geq} for $t = 2$ is as follows:

$$Cl_t^{\geq} = \{X_1, X_2, X_5, X_6\}$$

2nd Step: In this step, we will compute $D_P^+(x)$ for every item selected in the 1st Step. So, $D_P^+(x)$ for every item of class union Cl_t^{\geq} is as follows:

For $X_1: D_P^+(X_1) = \{X_1, X_2, X_4, X_6\}$

For $X_2: D_P^+(X_2) = \{X_2\}$

For $X_5: D_P^+(X_5) = \{X_2, X_5\}$

For $X_6: D_P^+(X_6) = \{X_1, X_2, X_4, X_6\}$

It is apparent from the above example that we have to calculate $D_P^+(x)$ for each item of class unions. This step is computationally very expensive especially for huge size datasets because to calculate $D_P^+(x)$ we have to iterate the whole dataset multiple times and these multiple iterations use lots of computational resources.

3rd Step: In this step, we calculate lower approximation using the formula aforementioned in equations 11. Those sets identified in 2nd step which are subsets of the set identified in 1st step become part of the lower approximation. We can say with certainty about these items which are selected using the formula in equation 8 that they are part of the class union Cl_t^{\geq} for $t = 2$. The calculated $\underline{P}(Cl_t^{\geq})$ is as follows:

$$\underline{P}(Cl_t^{\geq}) = \{X_2, X_5\}$$

Similarly, to calculate the lower approximations for downward class unions Cl_t^{\leq} , all the aforementioned three steps are performed but in 2nd step $D_P^-(x)$ is calculated instead of $D_P^+(x)$ and in 3rd step formula from equation 12 is used to calculate lower approximation.

2.4.2. Upper Approximations

In the classical RST approach, the upper approximations describe those sets of items that probably relate to concept X. Whereas in DRSA, for $P \subseteq C$ the P-upper approximations of $Cl_t^{\geq}(x)$ describe the set of those items that may relate to the class unions $Cl_t^{\geq}(x)$. Likewise, the P-upper approximations of $Cl_t^{\leq}(x)$ describe the set of those items that may relate to the class unions $Cl_t^{\leq}(x)$. Mathematically this is shown in equations 13 and 14.

$$\overline{P}(Cl_t^{\geq}) = \{x \in U: D_P^-(x) \cap Cl_t^{\geq} \neq \emptyset\} \quad (13)$$

$$\overline{P}(Cl_t^{\leq}) = \{x \in U: D_P^+(x) \cap Cl_t^{\leq} \neq \emptyset\} \quad (14)$$

Similar to lower approximation, the Computation of the upper approximation involves three crucial steps and these steps are computationally very expensive. Executing these three steps using the conventional approach results in grave performance holdups for feature selection algorithms. Using Table II, we will compute the p-upper approximation $\overline{P}(Cl_t^{\geq})$ for $t = 2$.

1st Step: For Computing P-upper approximation, the 1st step is to identify all the items relating to the class union Cl_t^{\geq} . Items of class union Cl_t^{\geq} for $t = 2$ are :

$$Cl_t^{\geq} = \{X_1, X_2, X_5, X_6\}$$

2nd Step: In this step, $D_P^-(x)$ is calculated for each item of the set identified in the 1st step. $D_P^-(x)$ of all the items of the identified set are:

$$\text{For } X_1: D_P^-(x_1) = \{X_1, X_3, X_4, X_6, X_7\}$$

$$\text{For } X_2: D_P^-(x_2) = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$$

$$\text{For } X_5: D_P^-(x_5) = \{X_3, X_5, X_7\}$$

$$\text{For } X_6: D_P^-(x_6) = \{X_1, X_3, X_4, X_6, X_7\}$$

It must be noted that this step considerably reduces the performance and efficiency because for every record we have to find all those records dominating it. This searching process needs a whole traversal of the dataset for every individual record. As there are four records in class union Cl_t^{\geq} identified in the 1st step. So, the specified dataset is traversed four times to calculate $D_P^-(x)$ for every item of the class union Cl_t^{\geq} .

3rd Step: Lastly, we identify those records relating to the P-upper approximation using equation 13 and this involves selecting those records whose $D_P^-(x)$ (identified in 2nd Step) have a non-empty intersection with the set identified in the 1st Step.

In our example, all of the calculated $D_P^-(x)$ have a non-empty intersection with set identified in 1st step. So, $\overline{P}(Cl_t^{\geq})$ for $t = 2$ is as follows:

$$\overline{P}(Cl_t^{\geq}) = \{X_1, X_2, X_5, X_6\}$$

Similarly, to calculate the upper approximations for downward class unions Cl_t^{\leq} , these aforementioned three steps are performed but in 2nd step $D_P^+(x)$ is calculated instead of $D_P^-(x)$ and in 3rd step formula from equation 14 is used to calculate upper approximation.

All of those algorithms which are based on conventional DRSA use these upper and lower approximation calculations and due to this their overall performance decreases. Therefore, Algorithms based on conventional DRSA are not feasible to be used for huge size datasets.

2.5. Dependency / Quality of Sorting

Dependency specifies the relation between the P-correctly stored objects and all of the objects of the Universe. P-correctly stored objects in the datasets are those object of the dataset which does not fall in any boundary region (doubtful region). The dependency of the dataset is a ratio between the P-correctly stored objects and all of the objects of the Universe. This can be mathematically represented as shown in equation 15. [29]

$$\gamma C(Cl) = \frac{|Universe - \left(\left(\bigcup_{t \in T} Bn_P(Cl_t^{\geq}) \right) \cup \left(\bigcup_{t \in T} Bn_P(Cl_t^{\leq}) \right) \right)|}{|Universe|} \quad (15)$$

Here, $Bn_P(Cl_t^{\geq})$ represents the boundary region of the upward class union which we can be calculated by finding the difference of the upper approximation ($\overline{P}(Cl_t^{\geq})$) and lower approximation ($\underline{P}(Cl_t^{\geq})$) of the upward class union. $Bn_P(Cl_t^{\leq})$ represents the boundary region of the downward class union which we can be calculated by finding the difference of the upper approximation ($\overline{P}(Cl_t^{\leq})$) and lower approximation ($\underline{P}(Cl_t^{\leq})$) of the downward class union. With the help of upper and lower approximation of class unions, we can calculate the P-boundaries (P-doubtful regions) of these class unions. The formulas to calculate P-boundary regions of upward and downward class unions are mentioned in equations 16 and 17, respectively.

$$Bn_P(Cl_t^{\geq}) = \overline{P}(Cl_t^{\geq}) - \underline{P}(Cl_t^{\geq}) \quad (16)$$

$$Bn_P(Cl_t^{\leq}) = \overline{P}(Cl_t^{\leq}) - \underline{P}(Cl_t^{\leq}) \quad (17)$$

It can also be said that dependency is the ratio of objects that are not in any doubtful region and all the objects of the Universe. This set of objects that are not in any doubtful region is called “*AlphaSet*” [29]. In the conventional approach, this *AlphaSet* is identified by subtracting the union of all the boundary region objects from the Universe as shown in equation 15. In the conventional methodology, you have to find all those key components like classes, Class Unions, upper and lower approximations and then boundary regions to calculate the dependency of datasets which is very complex and computationally too expensive for large size datasets.

To avoid these expensive computations, in the proposed methodology objects from the Universe are taken one-by-one then they are incrementally compared with all the other objects of the Universe. This comparison is carried out using proposed dominance-based dependency classes. As a result of this comparison, *AlphaSet* is obtained. This *AlphaSet* helps us to calculate the dependency of datasets without calculating complex approximations. Finally, the dependency of a dataset is calculated by dividing the cardinality of the *AlphaSet* with the cardinality of the Universe. The formula of dependency calculation is shown in equation 18.

$$\gamma_C(Cl) = \frac{|AlphaSet|}{|Universe|} \quad (18)$$

Chapter 3

Literature Review

CHAPTER 3: LITERATURE REVIEW

This chapter presents research work conducted in the area of dominance-based rough set approach and discusses its multiple real-time applications like in the field of manufacturing, airlines customer care and many more. After a brief literature review of work conducted in this area, we enlightened the research gaps that we found in previous works. Section 3.1 provides the background knowledge related feature selection, use of rough set theory and dominance-based rough set theory. Section 3.2 explains the research sequence of the performed research. Section 3.3 gives detail information about related work in the field of DRSA and also provides a comparison of multiple techniques that were used to improve the performance of the DRSA algorithm.

3.1. Background

For different medical, banking or telecom datasets the performance and achievements of machine learning algorithms are affected by a variety of factors. In most of these datasets and especially in medical datasets the major issues are the quality of data collected. If the information gathered from data is redundant or maybe irrelevant or sometimes noisy and as well as unreliable, then during the training process knowledge discovery becomes much harder. Feature selection is used to identify and remove redundant, noisy and irrelevant information from the dataset. In processing various datasets, from these large numbers of features, an optimum subset of features should be elected which must not lose any necessary information. Collecting data is not and never was an easy task especially in medical applications because it is a slow and expensive process; less amount of data needs to be collected after the features are successfully reduced. In recent years, substantial research efforts have been made to automatically discover vital knowledge and rules in the medical field using data mining or machine learning techniques. Numerous classification and feature selection techniques based on a dominance-based rough set approach have been presented. Accurate, efficient and precise classification is most critical particularly in the medical field for the treatment of patients. As compared to multiple existing algorithms in resolving large-scale classification tasks, DRSA provides better and efficient results that were not offered by

conventional RST. To handle the uncertainty issues in datasets DRSA was proposed which is an extension of the conventional RST approach.

3.2. Research Sequence

Here, we will explain the steps which we follow to do the literature review for our thesis.

3.2.1. Research Questions

The following are the research questions for our thesis.

- What are the ways to calculate lower and upper approximations in DRSA?
- What are the ways to reduce the computational cost of calculating lower and upper approximations in DRSA?
- What is the worth of the traditional algorithm of calculating lower and upper approximations in DRSA?
- What is the impact of parallel and incremental computation in calculating lower and upper approximations in DRSA?

3.2.2. Inclusion/Exclusion Criteria

Inclusion and exclusion criteria of our reviewed papers are as follows:

- Subject Relevant papers are selected.
- Papers published after 2010 are selected and literature published before 2010 is not considered.
- Papers from renowned digital libraries are selected such as IEEE, Elsevier, Springer.

3.2.3. Keywords

Following are the keywords for our literature review:

- Dominance based Rough Set Theory
- Rough Set Theory

- Lower Approximation
- Upper Approximation
- Computational Complexity
- Feature Selection
- Classification

3.3.Related Work

A systematic study of related work has been performed based on a defined set of criteria, consulting the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines [30].

As the main benefit of DRSA is to process information for preference ordered domains, thus it has been widely used for multi-criteria analysis [31]. For calculating the upper and lower approximations in DRSA, different ways have been adopted. One of them is the conventional method which has the highest computational cost whereas other methods are incremental and parallel processing methods. A brief review of different research papers along with their comparison is provided in this section.

In [32], authors have used improved DRSA for the classification of medical data. The presented approach is used for nominal attributes whereas conventional DRSA is used for multi-criteria and ordinal attributes. Here, the decision table is used to find the dominance relation of all the instances of Universe, the proposed approach has been employed to find out the lower and upper approximation of the complete dataset. Lastly, for classification purposes, researchers have extracted the reduced attribute set by using attribute reducing technique. The proposed approach uses conventional lower and upper approximations which degrades the performance of the proposed approach.

In [33], authors have presented a classification technique based on DRSA for the management of spare parts. The proposed method uses conventional upper and lower approximation measures. It is based on the three-step framework. In the first step, multiple “if.....else” rules are extracted from the historic datasets using DRSA. In the second step, the proposed rules are validated both manually and automatically using cross-validation techniques. Finally, in the third step, the unused

set of spare parts is classified in a real environment. The authors tested the proposed approach by using real-world data and the accuracy was found to be 96%. However, the proposed approach uses a conventional DRSA which results in highly expensive computations.

In [34], DRSA has been used by the authors to forecast the strength of schoolchildren that will probably fail the Massive Open Online Course (MOOC) class coming week by using the data of the preceding week. This approach divides the students into a couple of classes, Cl_1 which represents “At-risk Learners” and Cl_2 represents the “Active Learners”. This technique is based on a two-step approach, step one establishes a preference mode and step 2 divides the students into classes Cl_1 and Cl_2 . The step one itself comprises of three phases. The First phase extracts the learning examples of students. While phase two forms the comprehensible criteria group for students’ profile classification and phase three is to deduce a preference model which results in a collection of decision rules. The proposed technique is based on traditional DRSA. Thus, it inherits all the bottlenecks of the conventional approach and as a result, it considerably decreases the performance of the algorithm.

In [35], authors have presented DRSA based technique for forecasting customer’s behavior in airline companies. This may assist managers to get new customers and get hold of esteemed and treasured customers. To evaluate its forecasting ability a collection of rules is extracted from a huge sample of international airline customers. This approach was based on traditional DRSA and uses upper and lower approximations which can result in the inherent performance drawbacks.

In [36], authors have presented a novel method for discovering the reducts in DRSA. They have inspected the attribute reduction in DRSA accompanied by defining class-based reducts and their relation with earlier reducts. There are three types of class-based reducts. First reducts are called L-reducts and they protect the lower approximations of the decision classes. Second reducts are called U-reducts and they protect the upper approximations of the decision classes. Third reducts are called B-reducts and they protect the boundary regions of the decision classes. Moreover, they have demonstrated that all types of reducts can be computed broadly depending on two discernibility matrices related to generalized decisions.

In [37], the authors have proposed a parallel algorithm for approximations of DRSA and compared them on three different MapReduce runtime systems i.e. Twister, Phoenix, and Hadoop. Execution time and speed parameters are used to compare the approaches. In [38], the authors have

introduced three different matrixes based on parallel methods for the DRSA approximations. This approach tackles large incomplete datasets with missing values. Twister MapReduce has been used to apply this approach.

In [39], authors have implemented the Variable Consistency Dominance-based Rough Set Approach (VC-DRSA) to develop airline service strategies by making airline service decision rules that model passenger preferences for airline service quality. Flow graphs are used to deduce the decision rules. The results were improved slightly but the use of conventional methods was still a bottleneck for improving the overall performance. In [40], the authors have suggested a matrix-based dynamic incremental approach for the DRSA approximations under the process of attribute generalization. Five pre-processed data sets from UCI are used. Their proposed approach is time-efficient but has a complex algorithm structure. Whereas, some authors have also combined incremental and parallel approaches to mitigate the bottlenecks of conventional DRSA [41].

Table III shows a comparative summary of the approaches discussed above.

Table III: Comparisons of techniques used in different papers

Algorithms	Technique used	Advantages	Disadvantages
“Improved dominance rough set-based classification system” [32]	DSRA based classification	Improved results	Conventional lower and upper approximation techniques affect the performance
“Spare parts classification in industrial manufacturing using the dominance-based rough set approach” [33]	DSRA based approach for spare parts classification	High accuracy	Conventional approximation based approaches are used

“Weekly predicting the At-Risk MOOC learners using Dominance-Based rough set approach” [34]	DSRA based approach for classifying students	Prediction of “At-risk” students	Conventional approach used
“A dominance-based rough set approach to customer behavior in the airline market” [35]	DSRA based prediction approach	Prediction and retaining of high valued customer	Conventional approach affects performance
“A unified approach to reducts in dominance-based rough set approach” [36]	DSRA based approach for finding reducts	New types of reducts are introduced	Conventional approximation approach affects the performance of the algorithm
“A comparison of parallel large-scale knowledge acquisition using rough set theory on different MapReduce runtime systems” [37]	Phoenix, Twister, Hadoop (MapReduce) Parallel approach	Improved efficiency, low execution time	Use of more hardware (multiple processors)
“A Parallel Matrix-Based Method for Computing Approximations in Incomplete Information Systems” [38]	Twister (MapReduce) Parallel approach	Efficient and works on incomplete missing data sets, low computational complexity	Use of more hardware (multiple processors)
“Variable Consistency Dominance-based Rough	Variable Consistency Dominance-based Rough Set Approach (VC-DRSA) to	The use of flow graphs to visualize rules makes them more reasonable	Conventional approximation-based approaches are used

Set Approach to formulate airline service strategies” [39]	formulate airline service strategies.	than traditional methods.	
“Dynamic dominance rough set approach for processing composite ordered data” [40]	DRSA based dynamic incremental approach	Low computational complexity	The complex architecture of the algorithm

Chapter 4

Challenges In DRSA And Proposed Solution

CHAPTER 4: CHALLENGES IN DRSA AND PROPOSED SOLUTION

This chapter explains the challenges that make the conventional DRSA algorithm less efficient and complex. The reason behind these challenges is also described in detail. In the later part of the chapter, the solution is also presented to overcome these challenges. This chapter has two sections. In the first section, we have discussed the challenges that we face in conventional DRSA and the second section has further a sub-section that explains the proposed dominance-based dependency classes that are used to calculate the dependency of the data set incrementally.

4.1. Challenges in DRSA

As we have mentioned earlier that calculating dependency using the conventional approach requires a three-step process of calculating lower and upper approximations. This process makes it computationally an expensive task and significantly affects the performance of the algorithms. The challenges faced while performing this three-step process are discussed in this section.

The first step for calculating approximations requires the calculation of class unions which are represented as Cl_t^{\geq} and Cl_t^{\leq} . While calculating these class unions, we have to traverse through the complete Universe. This could be a time-taking process and may require lots of computing resources for larger datasets. For our example dataset, we only have to repeat this step seven times but this can easily reach thousands of times with larger Universe size. The pseudocode of step 1 is shown in figure 3.

```
For i = 1 to |U|  
if  $Cl_i \geq Cl_t$   
 $Cl_t^{\geq} = Cl_t^{\geq} \cup Cl_i$   
End – if  
End – For
```

Figure 3: Pseudocode of 1st Step for calculating lower approximation.

The following are the class unions for our example dataset.

$$Cl_1^{\leq}(x) = \{X_3, X_4, X_7\}$$

$$Cl_2^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$Cl_2^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$$

$$Cl_2^{\geq}(x) = \{X_2\}$$

In the 2nd Step of this conventional method of approximations calculation, we have to calculate $D_p^+(x)$ and $D_p^-(x)$ for all the items of the calculated class unions. This step requires a nested loop for its implementation outer loop will execute from 1 to the cardinality of the class union to incrementally get the items from the set of the class union. The inner loop will execute from 1 to cardinality of the universe ($|U|$) to compare the selected item from the outer loop with all the items of the Universe. This step is repeated for all the items of all the class unions identified in the 1st Step. For example, in our case inner loop for $Cl_1^{\leq}(x)$ will run $3 * 7 = 21$ times. Here, 3 is the cardinality of the set $Cl_1^{\leq}(x)$ and 7 is the cardinality of the Universe. The repetitions of the loop in the 2nd step can easily reach hundreds of thousands when datasets are huge in size. So, 2nd Step is much more complex and needs more time for computation than the 1st step. The pseudocode for calculating $D_p^+(x)$ and $D_p^-(x)$ is shown in figure 4.

```

//For Dominance Positive
For i = 1 to |Cl_t^{\geq}/Cl_t^{\leq}|
  For j=1 to |U|
    If X_j \ge X_{it}
       $D_p^+(X_i) = D_p^+(X_i) \cup X_j$ 
    End – If
  End – For
End – For

//For Dominance Negative
For i = 1 to |Cl_t^{\geq}/Cl_t^{\leq}|
  For j=1 to |U|
    If X_j \le X_{it}

```

```

 $D_P^-(X_i) = D_P^-(X_i) \cup X_j$ 
End – If
End – For
End – For

```

Figure 4: Pseudocode of 2nd Step for calculating lower approximation.

Finally in the 3rd Step, $\underline{P}(Cl_{\bar{t}}^{\geq})$ and $\bar{P}(Cl_{\bar{t}}^{\geq})$ are to be calculated which are the lower approximation and upper approximation of class unions and these are calculated for all the class unions. To perform this step, a triple nested loop is required. The outermost loop will execute from 1 to the cardinality of the class union. The middle loop will execute from 1 to the cardinality of $D_P^+(x)$ or $D_P^-(x)$. These two loops are used to traverse a two-dimensional array of $D_P^+(x)$ or $D_P^-(x)$. After that, we have to compare these indexes with the class unions to check if intersection and subset properties hold. For this purpose, the innermost loop will also execute from 1 to the cardinality of the class unions. These triple nested loops tremendously increase the execution time and computational complexity of the conventional approach. The pseudocode of step 3 is shown in figure 5.

```

For i = 1 to  $|Cl_{\bar{t}}^{\leq}|$ 
For j = 1 to  $|D_N^+(X_i)|$ 
For k = 1 to  $|Cl_{\bar{t}}^{\leq}|$ 
Calculate  $D_N^+(X_{ji}) \subseteq Cl_{\bar{k}t}^{\leq}$ 
End – For
End – For
End – For

```

Figure 5: Pseudocode of Step-3 for calculating lower approximation

Therefore, the conventional approach for calculating dependency poses grave challenges to the performance of the algorithms when it comes to larger datasets. To overcome all of the above-described performance bottlenecks of the conventional approach, we have proposed a solution that reduces the computational complexity of dependency calculation by avoiding the approximation approach. This proposed method greatly reduces the computational complexity and resource

utilization for dependency calculation which ultimately enhances the performance of the algorithms.

4.2. Proposed Methodology

In our proposed solution, to make the reduct generation process more efficient we have replaced the approximation calculation part with a heuristic-based method. The proposed method is called the “**Incremental Dominance-Based Dependency Calculation (IDDC)**”. The proposed method reduces the computational complexity and resource utilization to calculate the dependency of a dataset. IDDC is a combination of two phases. The first phase of the IDDC is to calculate the dominance table also known as a dominance matrix which provides us information regarding the dominance relation between all objects of the Universe. This two-dimensional matrix stores the comparison results of all the objects of the Universe with four possible relations. Four possible relations between two objects $\{a, b \in U\}$ can hold as follows:

- i. First relation which shows that a is dominant over b . In the dominance matrix, it is represented by the symbol “ \geq ”.
- ii. The second relation shows that a is dominated by b . In the dominance matrix, it is represented by the symbol “ \leq ”.
- iii. The third relation shows that a and b are identical and have the same values for all the conditional attributes in the dominance matrix. In dominance matrix, it is represented by the symbol “ $=$ ”.
- iv. The fourth relation shows that both of the objects are indiscernible which means that the value of a is greater than or equal to b for certain features and less than and equal to b for remaining features. In the dominance matrix, it is represented by the symbol “ \neq ”.

Table IV: Comparison Possibilities of two objects

S. No.	Possible Relations	Stored Values
1	<i>A is dominating B</i>	$A \geq B$
2	<i>A is being dominated by B</i>	$A \leq B$
3	<i>A is identical to B</i>	$A = B$
4	<i>A and B are indiscernible</i>	$A \neq B$

After the calculation of the dominance matrix, the second phase of IDDC is to find the dependency of a dataset by comparing all the records of the Universe based on proposed dominance-based dependency classes. This second phase is the main difference between both proposed and conventional methods of calculating the dependency of the dataset. In the proposed approach, we will avoid calculating approximations and boundary regions to calculate the dependency of the dataset. In this phase of the proposed approach, the following steps are taken to calculate the dependency of the dataset:

- Two arrays of variable length by the name of *AlphaSet* (it provides a set of records to calculate dependency of the dataset) and *Checked_Objects* (it is used to avoid the comparison repetitions by storing those records which have already been compared) are created.
- Records from Universe are taken one-by-one and compared with all the records in the *Checked_Objects* array and during comparison, their dominance relation and the decision classes are compared.
- The dominance relation based on conditional attributes is compared using a dominance matrix.
- Records are added or removed from the *AlphaSet* based on their comparison but every record taken from the Universe is added into the *Checked_Objects* array to avoid repetitive comparisons of records.
- Proposed dominance-based dependency classes are used to determine the result of every comparison of the records.
- When all the records from the Universe are traversed than the cardinality of *AlphaSet* is divided by the cardinality of the Universe to get the dependency also known as the quality of sorting of the dataset. The formula to calculate the dependency is aforementioned in equation 15.

Each minimal subset of attribute set $P \subseteq C$ will be called *Reduct* of Cl if $\gamma_P(Cl) \geq \gamma_C(Cl)$ which means that reduct is considered valid if the dependency of the reduct is equal to or better than the

dependency of the complete dataset. We can have multiple reducts and their intersection is called *Core* and it is comprised of *Core attributes*.

4.2.1. Proposed Dominance-Based Dependency Classes

We have defined dominance-based dependency classes that determine the outcome of a comparison of records. They explain how dependency of dataset changes during traversal of the dataset. It starts from the first record to calculate the dependency of the dataset and as it traverses to next record the dependency of the dataset is updated. This incremental method avoids the complex dependency calculation method of the conventional approach. Table II is considered as an example of a dataset to demonstrate the working of each dominance-based dependency class.

- **Initial value class**

This class only checks that if the selected record $i \in U$ is the first record of the Universe. If it is true then this record will be selected and added in both *AlphaSet* as well as *Checked_Objects*. This class determines that if the selected record is the first record of the Universe. Therefore, it is called “*Initial Value Class*”. Dependency value for this class will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} \quad (19)$$

Here, $|Checked_Objects|'$ represents the previous cardinality of *Checked_Objects* without adding the newly traversed object. Similarly, $|AlphaSet|'$ represents the previous cardinality of *AlphaSet* without adding the current object.

Initially $|AlphaSet|' = 0$ and $|Checked_Objects|' = 0$ because we are traversing the first object of the dataset. In our case when we picked our first record which is X_1 from Table II, then the dependency of dataset became:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{0 + 1}{0 + 1} = 1 \quad (20)$$

After adding the first record, $|AlphaSet|' = 1$ and $|Checked_Objects|' = 1$.

- **True Positive class**

This class shows the consistency in the dataset and this class has two sides, one is an object having a higher decision class with a dominant relation and the other is an object having a lower decision class with a dominated relation. Dominance relation and decision classes of these records are used to determine which record is better. Mathematically, For two records \mathbf{a} , \mathbf{b} and attribute set $\mathbf{c} \in \mathbf{C}$ first side of this class can be written as shown in equation 21.

$$(C(a) > C(b) \text{ AND } D(a) \geq D(b)) \quad (21)$$

This class is called “*True Positive Class*” because \mathbf{a} has dominant relation with \mathbf{b} and also has a higher decision class than \mathbf{b} . This does not cause any inconsistency in the dataset.

Similarly, if an object \mathbf{a} is dominated by object \mathbf{b} with lower decision class then mathematically the other side of this class can be written as shown in equation 22.

$$(c(a) < c(b) \text{ AND } D(a) \leq D(b)) \quad (22)$$

Dependency value for this class will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} \quad (23)$$

In our example Table II, X_2 is dominant over X_1 with a higher decision and is not causing any inconsistency in the dataset. So, X_2 will be added to *AlphaSet* and dependency will become:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{1 + 1}{1 + 1} = 1 \quad (24)$$

Similarly, as we traverse to the 3rd record in Table II and compare the X_3 with X_1 . This comparison represents the other side of the class. Where X_3 is dominated by X_1 and also has a lower decision class than X_1 . This does not cause any inconsistency in the dataset. So, X_3 is added into *AlphaSet* and dependency of dataset becomes:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{2 + 1}{2 + 1} = 1 \quad (25)$$

- **Distinct Decision Class**

This class defines one of the inconsistencies we face in the dominance-based datasets where values of conditional attributes for two records are similar yet they have different decision classes. Therefore, this class is called “*Distinct Decision Class*”. Mathematically, For two records **a**, **b** and attribute set **c** $\in \mathbf{C}$ this class can be written as shown in equation 26.

$$(c(a) == c(b) \& D(a) \neq D(b)) \quad (26)$$

Because of this inconsistency, **b** will not be added into the *AlphaSet* and we will also remove all those records from *AlphaSet* which cause this inconsistency when compared to **b**. Dependency value for this class will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked_Objects|' + 1} \quad (27)$$

Here, *N* represents the number of records in *AlphaSet* that cause this inconsistency when compared with this newly traversed record. For example, as we traverse through Table II and compare X_4 with X_1 . This comparison shows that both records have the same attribute values but different decision class. Therefore, both of these records will be excluded from *AlphaSet*. While comparing X_4 with the records of *Checked_Objects*, this inconsistency appeared only once when X_4 is compared to X_1 . As only one member of *AlphaSet* has caused this inconsistency while comparing with X_4 so, we put $N = 1$. Therefore, the dependency of the dataset becomes:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked_Objects|' + 1} = \frac{3 - 1}{3 + 1} = 0.5 \quad (28)$$

- **Indiscernible class**

This class explains the indiscernible relation between the records of the datasets where one record has preferred values for some conditional attributes but worse values for others so, it does not matter what their decision classes are because both of them will be added into the *AlphaSet*. Therefore, this class is called “*Indiscernible Class*”. Mathematically, For two records \mathbf{a}, \mathbf{b} and attribute set $\{\mathbf{c}_1, \mathbf{c}_2 \in \mathbf{C} | (\mathbf{c}_1 \cap \mathbf{c}_2) \neq \emptyset\}$ this class can be written as shown in equation 29.

$$((c_1(a) \geq (c_1(b) \text{ AND } (c_2(a) \leq (c_2(b))) \quad (29)$$

This shows that object \mathbf{a} is dominant over \mathbf{b} for criteria \mathbf{c}_1 but at the same time, \mathbf{a} is being dominated by \mathbf{b} for criteria \mathbf{c}_2 . Therefore, both objects will be kept in *AlphaSet* and the dependency of the dataset will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} \quad (30)$$

In Table II, when we traversed to record X_5 and compare it with X_1 it results in an indiscernible relation. As this does not cause any inconsistency in the dataset so, X_5 is added into *AlphaSet* and dependency of dataset becomes:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{2 + 1}{4 + 1} = 0.6 \quad (31)$$

- **Identical class**

This class defines the relationship between two identical records with an identical decision. Therefore, this class is called “*Identical Class*”. This shows the consistency in the dataset. Mathematically, For two records \mathbf{a}, \mathbf{b} and attribute set $\mathbf{c} \in \mathbf{C}$ this class can be written as shown in equation 32.

$$(c(a) == c(b) \& D(a) == D(b)) \quad (32)$$

Ideally, the newly traversed object \mathbf{b} should be added into $AlphaSet$ but if object \mathbf{a} has already been a part of any inconsistency then the object \mathbf{b} will not be added into $AlphaSet$. Ideally, dependency for this class will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} \quad (33)$$

In Table II, when X_6 is compared to X_1 it shows that both records have identical attributes values with an identical decision class. This represents the consistency in the dataset. Ideally, X_6 should be added into $AlphaSet$ but X_6 is not added into the $AlphaSet$ because X_1 has already been a part of an inconsistency. Therefore, the dependency of the dataset becomes:

$$\gamma C(Cl) = \frac{|AlphaSet|'}{|Checked_Objects|' + 1} = \frac{3}{5 + 1} = 0.5 \quad (34)$$

- **False Positive Class**

This class also has two sides, one is an object having a lower decision class with a dominant relation and the other is an object having a higher decision with dominated relation. Therefore, it is called “*False Positive Class*”. Mathematically, For two records \mathbf{a}, \mathbf{b} and attribute set $\mathbf{c} \in \mathbf{C}$ first side of this class can be written as shown in equation 35.

$$(c(a) > c(b) \text{ AND } D(a) < D(b)) \quad (35)$$

Here, \mathbf{a} has a dominant relation over \mathbf{b} with a lower decision class. This causes inconsistency in the dataset.

Mathematically, For two records \mathbf{a}, \mathbf{b} and attribute set $\mathbf{c} \in \mathbf{C}$ second side of this class can be written as shown in equation 36.

$$(c(a) < c(b) \text{ AND } D(a) > D(b)) \quad (36)$$

This states that \mathbf{a} is dominated by \mathbf{b} with a higher decision class. This also causes inconsistency in the dataset. Therefore, the dependency of the dataset decreases due to these inconsistencies. The dependency in both scenarios will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked_Objects|' + 1} \quad (37)$$

Here, N represents the number of records in $AlphaSet$ that causes the same inconsistency when compared with the newly traversed record. Such as if two records of $AlphaSet$ are causing this inconsistency when compared with the newly traversed record then N will be put $N = 2$.

The pseudocode of the proposed methodology is shown in figure 6.

```

//For Extracting AlphaSet to calculate dependency of the dataset
Inputs: Universe and Dominance Matrix
Output: Dependency of Dataset
Step 1: Find AlphaSet
For  $i = 0$  to  $|U|$ 
    Inconsistency = False
    Pick objects from Universe one by one on every iteration let's name it  $\{a \in U\}$ .
    If (the first iteration of the loop)
        Add object  $a$  to both Checked_Objects and AlphaSet arrays.
    Else
        For  $j=0$  to  $|Checked_Objects|$ 
            Pick objects from Checked_Objects iteratively and let's name it  $\{b \in$ 
Checked_Objects\}.
            If ( $a$  and  $b$  are indiscernible)
                If (Inconsistency = False)
                    keep object  $a$  in both Checked_Objects and AlphaSet arrays.
                End – If
            Else – If ( $C(a) > C(b)$  AND  $D(a) \geq D(b)$ ) or ( $c(a) < c(b)$  AND  $D(a) \leq$ 
D(b))
                If (Inconsistency = False)
                    keep object  $a$  in both Checked_Objects and AlphaSet arrays.
                End – If
            Else – If ( $c(a) = c(b)$  &  $D(a) = D(b)$ )

```

```

    If(Inconsistency = Flase)
        keep object a in both Checked_Objects and AlphaSet arrays.
    End – If
Else – If(c(a) == c(b)& D(a) ≠ D(b))
    Remove object a and b from AlphaSet array but keep in Checked_Objects.
    Inconsistency = True
Else – If(c(a) > c(b) AND D(a) < D(b)) or(c(a) < c(b) AND D(a) > D(b))
    Remove object a and b from AlphaSet array but keep in Checked_Objects.
    Inconsistency = True
End – If
End – For
End – If
End – For
Step 2: Find Dependency which is the ratio of the cardinality of AlphaSet and cardinality of
Checked_Objects which will be equal to the cardinality of Universe at the end of complete
traversal.
Dependency =  $\frac{|AlphaSet|}{|Universe|}$ 
Finally, Output is Dependency.

```

Figure 6: Pseudocode of the proposed methodology

This proposed approach is accurate and much more efficient than the conventional approach which uses the approximations. To illustrate our proposed algorithm more clearly, we have used Table II as a case study. With the help of this Case study, we made a comparison of statements from both approaches and find out a percentage decrease in the number of comparison statements. This case study is presented in the next chapter.

Chapter 5

Implementation and Validation

CHAPTER 5: IMPLEMENTATION AND VALIDATION

This chapter deals with the implementation detail for our proposed methodology and describes all the datasets and algorithms we used to implement our technique. In section 5.1, datasets and all the algorithms that are used to validate the proposed methodology are discussed. In section 5.2, a case study is solved using the conventional method to get its results and in section 5.3, a case study is solved using the proposed methodology to compare the results.

5.1. Datasets and Algorithms

To provide better experimentation results we applied this approach on 10 different datasets from the UCI repository[42]. We took data sets with a different number of instances and attributes so that it is easy to visualize and understand the trend of reduction of time and memory usage from a smaller number of instances datasets to a huge number of instances datasets. We selected those datasets on the bases of dominance properties. The detail of these datasets is as follows:

- The 1st dataset that we used is **Iris**. Features of the dataset include sepal length, sepal width, petal length and petal width in cm. The data type is real. The number of instances is 150 and the number of attributes is 4.
- The 2nd dataset that we used is **Abalone**. Features of the dataset include sex, age and different physical measurements of the shell. The data type is real and integer. The number of instances is 4177 and the number of attributes is 8.
- The 3rd dataset that we used is **Breast Cancer Wisconsin**. Features of the dataset include the size, shape of cells and information related to nuclei. The data type is an integer. The number of instances is 699 and the number of attributes is 10.
- The 4th dataset that we used is **Wine Quality**. Features of the dataset include physicochemical variables that are used for testing the quality of the wine. The data type is real. The number of instances is 4898 and the number of attributes is 12.
- The 5th dataset that we used is the **Electrical Grid Stability data**. Features of the dataset include the information related to the properties of the current. The data type is real. The number of instances is 10000 and the number of attributes is 14.

- The 6th dataset that we used is **EEG Eye State**. Features of the dataset include values that are generated according to the eye state during egg measurement. The data type is real and integer. The number of instances is 10200 and the number of attributes is 15.
- The 7th dataset that we used is **Pen-Based Handwritten Digit**. Features of the dataset include integers in the range 0...100 and the last attribute is the class code 0...9. The data type is an integer. The number of instances is 10992 and the number of attributes is 16.
- The 8th dataset that we used is the **Hepatitis C Virus**. Features of the dataset include all the relevant information related to the body like BMI, age, gender, etc. The data type is real and integer. The number of instances is 1395 and the number of attributes is 29.
- The 9th dataset that we used is **Musk 2**. Features of the dataset include the exact shape and conformation of molecules. The data type is an integer. The number of instances is 6598 and the number of attributes is 168.
- The 10th dataset that we used is **Isolet**. Features of the dataset include post-sonorant features, spectral coefficients, sonorant features and pre-sonorant features. The data type is real. The number of instances is 7797 and the number of attributes is 617.

All the datasets have an ordinal preference domain. A summary of datasets is shown in Table V.

Table V: Information of Datasets

S. NO.	Dataset	Number of Instances	Number of Attributes	Type of Dataset's Attributes
1	Iris	150	4	Real
2	Abalone	4177	8	Integer, Real
3	Breast Cancer Wisconsin	699	10	Integer
4	Wine Quality	4898	12	Real
5	Electrical Grid Stability data	10000	14	Real
6	EEG Eye State	10200	15	Integer, Real
7	Pen-Based Handwritten Digit	10992	16	Integer
8	Hepatitis C Virus	1380	29	Integer, Real
9	Musk 2	6598	168	Integer
10	ISOLET	7797	617	Real

Our proposed methodology can be implemented using different algorithms to get the reducts in lesser time but for experimentation purposes, we have picked three different algorithms to prove that this approach of our can be easily implemented and get accurate results in lesser time by reduced usage resources. These three algorithms are briefly explained below along with their pseudocodes.

5.1.1. Fast Reduct Generating Algorithm

This algorithm is an exhaustive reduct generating algorithm which helps in finding all the possible Quality reducts of dataset no matter how much time it takes to complete the process[20]. Quality Reducts or Q-Reducts are those which preserves the quality of sorting of the dataset. So, this algorithm is very helpful when you need to find all the possible Q-reducts. We used this algorithm to find the time is taken and memory used by both of the approaches because of its exhaustive reducts generating ability. Then, this time is taken and memory usage values are used to find the percentage reduction in usage of resources like time and memory by our newly proposed approach. The pseudocode of FRGA is presented in Figure 7[20]:

Input: A set of objects in Universe U . These objects are described by values of attributes.

Output: A set K which is comprised of all the possible reducts of set U .

Phase 1: Creation of Sorted, Absorbed Dominance Retaining List(*SADRL*).

Step 1: Create a dominance retaining list (*DRL*) which contains all of those attributes and criteria
 . that retains dominance between all the appropriate pairs of objects. The resulting list will
 . contain ($C_1, C_2, C_3 \dots C_n$)

Step 2: Absorb the *DRL* by eliminating empty and non-minimal elements from *DRL*.

. $ADRL = \{C_i \in DRL: C_j \neq \emptyset, \text{ is unique in } ADRL \text{ and for no } C_j \in ADRL: C_j \subseteq C_i\}$

. The resulting *ADRL* is usually less than *DRL*.

Step 3: Sort the *ADRL* in ascending order of its element's cardinality and we have *SADRL*.

Phase 2: A Breadth-First search for reducts.

Step 1: $Red_0 = \{\emptyset\}$

Step 2: For every $i=1 \dots d$ compute.

<ul style="list-style-type: none"> . $S_i = \{R \in Red_i - 1: R \cap C_i \neq \emptyset\}$. $T_i = U_{q \in C_i} U_{R \in Red_{i-1}: R \cap C_i = \emptyset} \{R \cup \{q\}\}$. $MIN_i = \{R \in T_i : FPI(R) = true\}$. $Red_i = S_i \cup MIN_i$. the final result is $K = R_d$.
--

Figure 7: Pseudocode of FRGA Algorithm

5.1.2. Genetic Algorithm

A genetic algorithm is a key tool to find the optimal solution when there are millions of potential solutions are available and search space is also huge. The genetic algorithm is a heuristic-based algorithm and heuristic can be set by the user of this algorithm. In [43] authors present a rough set based genetic algorithm (GA) for feature selection. So, we have implemented a genetic algorithm with both conventional and our proposed approaches to find the dependency of datasets and selecting optimal reducts. Then the comparison of both approaches explicitly highlights the time and resources saved by the proposed methodology.

These randomly initialized generations were used to construct the gene pool used to determine the intermediate region used for crossover and mutation operators. For crossover, order-based and partially matched crossover methods were used. In order based method, a random number of solution points are selected from parent chromosomes. In the first chromosome selected gene will remain at its place whereas, in the second chromosome, the corresponding gene will be beside that of the first chromosome that occupies the same place. Order based crossover method is shown in Figures 8 and 9. Zeros and ones in the figures represent the missing and selected attributes from the datasets. Figure 8 shows the selected chromosomes and Figure 9 shows the resultant chromosomes.

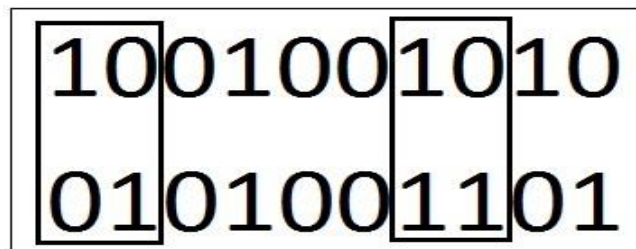


Figure 8: Selected chromosomes for ordered based crossover method

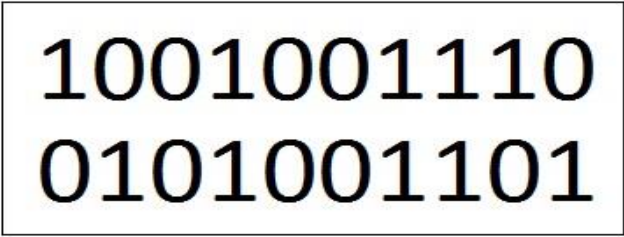


Figure 9: Chromosomes after ordered based crossover

For mutation, bit flip and two change mutation operators were used. In the bit flip method, the mutation operator takes the chosen genome and inverts the bits like if the value of bit was 1 after flipping it will become 0 whereas in adjacent two input change mutation method, adjacent two genes are selected and place of genes are inverted. Figures 10 and 11 show both mutation methods.

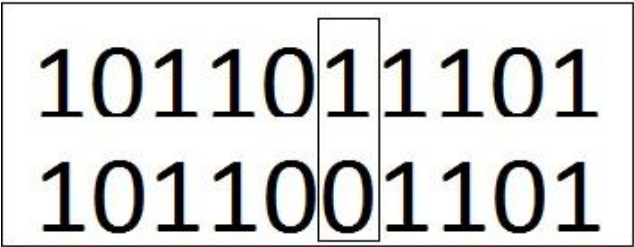


Figure 10: Chromosome after bit flip mutation

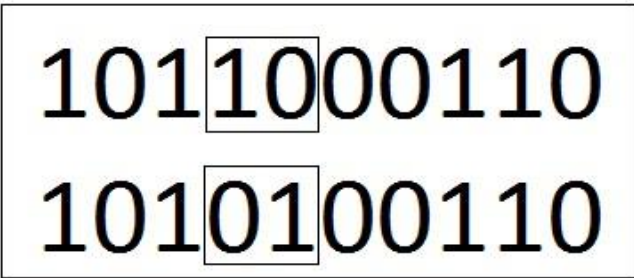


Figure 11: Chromosome after adjacent two mutation

The typical flow chart of the Genetic Algorithm is presented in Figure 12.

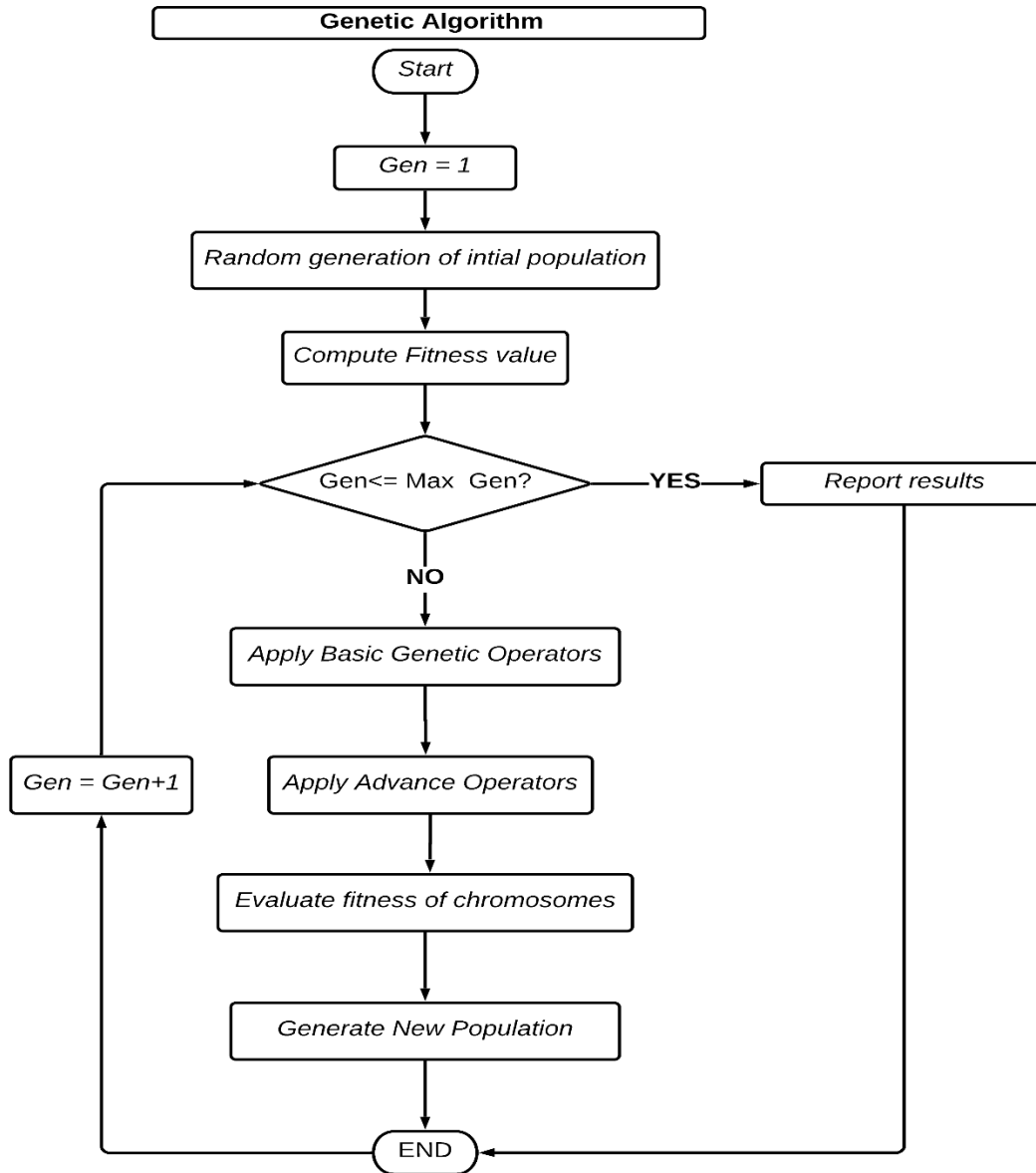


Figure 12: A Typical Flow Chart of Genetic Algorithm

5.1.3. Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart [43]. The original intent was to graphically simulate the graceful but unpredictable movements of a flock of birds. Initial simulations were modified to form the original version of PSO. Later, Shi introduced inertia weight into the particle swarm optimizer to produce the standard PSO [44][45]. The basic pseudo-code of this PSO is presented in Figure 9. Particles represent the different subsets of complete set attributes and fitness represents the

dependency or quality of sorting. Only those particles are selected which has equal or greater dependency than complete attribute set.

Input:

m: the swarm size; *c*₁, *c*₂: positive acceleration constants; *w*: inertia weight
MaxV: maximum velocity of particles
MaxGen: maximum generation
MaxFit: maximum fitness value

Output:

*P*_{gbest}: Global best position

Begin

Swarms {*x*_{*id*}, *v*_{*id*}} = Generate(*m*); /* Initialize a population of particles with random positions and velocities on *S* dimensions*/

*P*_{best}(*i*) = 0; *i* = 1, ..., *m*, *d* = 1, ..., *S*

*G*_{best} = 0; *Iter* = 0;

While(*Iter* < *MaxGen* and *G*_{best} < *MaxFit*)

{For(every particle *i*)

{Fitness(*i*) = Evaluate(*i*);

IF(Fitness(*i*) > *P*_{best}(*i*))

{*P*_{best}(*i*) = Fitness(*i*); *P*_{*id*} = *X*_{*id*}; *d* = 1, ..., *S*}

IF(Fitness(*i*) > *G*_{best})

{*G*_{best} = Fitness(*i*); *g*_{best} = *i*;}
 }

For(every particle *i*)

{For(every *d*)

$V_{id} = W * V_{id} + c_1 * rand() * (P_{id} - X_{id}) + c_2 * Rand() * (P_{gd} - X_{id})$

IF(*V*_{*id*} > *MaxV*) {*V*_{*id*} = -*MaxV*;}
 IF(*V*_{*id*} < -*MaxV*) {*V*_{*id*} = -*MaxV*;}
*X*_{*id*} = *X*_{*id*} + *V*_{*id*}
 }
 }

}

```

    Iter = Iter + 1;
}/* rand() and Rand() are two random functions in the range [0,1] */
Return Pgbest
End

```

Figure 13: Pseudocode of Particle Swarm Optimization (PSO)

Next section deals with the validation of our proposed methodology with the help of a case study. The case study is discussed and documented in descriptive form. The results of the “Student Academic Record” case study are discussed and validated by solving a small dataset shown in Table II with both the techniques. Due to a small number of instances, it is easy to see that both techniques produced the same results which further validates that our proposed methodology gives accurate results. A complete comparison of both techniques with the help of multiple datasets is presented in the next chapter of comparative analysis.

5.2. Case Study using Conventional Approximation’s Methodology

We have used the dataset shown in Table II for this case study. In this dataset we have a universe of seven elements, two criteria attributes and a decision attribute containing three possible decisions $\{Excellent, Very Good, Good\}$. Based on these decisions, the dataset is distributed into three classes and by using these classes, class unions of the dataset are calculated. To find the dependency of dataset or quality of dataset using the conventional method can be divided into six steps. These steps are explained below one by one.

Step 1: In this step, we will class sets based on every instance’s decision. Decision ‘Good’ is represented as Cl_I , decision ‘Very Good’ is represented as Cl_{II} and decision ‘Excellent’ is represented as Cl_{III} .

$$Cl_I = \{X_3, X_4, X_7\}$$

$$Cl_{II} = \{X_1, X_5, X_6\}$$

$$Cl_{III} = \{X_2\}$$

Step 2: Now, based on these class sets we will perform our second step which is finding class unions and the class unions are calculated based on decision classes as follows:

$$Cl_I^{\leq}(x) = \{X_3, X_4, X_7\}$$

$$Cl_{II}^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$Cl_{II}^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$$

$$Cl_{III}^{\geq}(x) = \{X_2\}$$

Step 3: After finding the class unions we move on to the next step which is very important of calculation dominance relation matrix which helps us in finding the D_p^+x and D_p^-x . The relation between records is illustrated using mathematical symbols. Every row depicts the relation of an individual record with all the records of the dataset. Like, the first row shows the dominance relation of X_1 with all the records of the dataset. In Table VI, equal to sign (=) represents the identical relation, not equal to (\neq) sign represents indiscernible relation, greater than equal to (\geq) sign represents that record is dominating the other record and less than equal to (\leq) sign represents that record is being dominated by another record. The complete dominance matrix of our case study dataset is shown in Table VI.

Table VI: Dominance Relation Matrix of Table II

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_1	=	\leq	\geq	=	\neq	=	\geq
X_2	\geq	=	\geq	\geq	\geq	\geq	\geq
X_3	\leq	\leq	=	\leq	\leq	\leq	\neq
X_4	=	\leq	\geq	=	\neq	=	\geq
X_5	\neq	\leq	\geq	\neq	=	\neq	\geq
X_6	=	\leq	\geq	=	\neq	=	\geq
X_7	\leq	\leq	\neq	\leq	\leq	\leq	=

Step 4: Now, with the help of this dominance matrix we will find D_p^+x and D_p^-x of every element of the Universe. D_p^+x will provide us with the set of elements which are dominating the respective element and D_p^-x will provide us with a set of elements that are dominated by the respective element. Then these D_p^+x and D_p^-x will be used to find the upper and lower approximations. First, find D_p^+x for all the seven elements of our case study.

$$D_p^+ x_1 = \{X_1, X_2, X_4, X_6\}$$

$$D_p^+ x_2 = \{X_2\}$$

$$D_p^+ x_3 = \{X_1, X_2, X_3, X_4, X_5, X_6\}$$

$$D_p^+ x_4 = \{X_1, X_2, X_4, X_6\}$$

$$D_p^+ x_5 = \{X_2, X_5\}$$

$$D_p^+ x_6 = \{X_1, X_2, X_4, X_6\}$$

$$D_p^+ x_7 = \{X_1, X_2, X_4, X_5, X_6, X_7\}$$

Similarly, let us find $D_p^- x$ for all the elements of our dataset.

$$D_p^- x_1 = \{X_1, X_3, X_4, X_6, X_7\}$$

$$D_p^- x_2 = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$$

$$D_p^- x_3 = \{X_3\}$$

$$D_p^- x_4 = \{X_1, X_3, X_4, X_6, X_7\}$$

$$D_p^- x_5 = \{X_3, X_5, X_7\}$$

$$D_p^- x_6 = \{X_1, X_3, X_4, X_6, X_7\}$$

$$D_p^- x_7 = \{X_7\}$$

Step 5: After calculating the $D_p^+ x$ and $D_p^- x$ for every element of the Universe we will further calculate the upper and lower approximations of all the union classes using $D_p^+ x$ and $D_p^- x$ of every element present in those union classes. The equations for calculating these approximations are as following:

Mathematically lower approximations of class unions are calculated by using these two equations.

For $Cl_t^{\geq}(x)$:

$$\underline{P}(Cl_t^{\geq}) = \{x \in U: D_p^+(x) \subseteq Cl_t^{\geq}\} \quad (38)$$

For $Cl_t^{\leq}(x)$:

$$\underline{P}(Cl_t^{\leq}) = \{x \in U: D_p^-(x) \subseteq Cl_t^{\leq}\} \quad (39)$$

Mathematically lower approximations of class unions are calculated by using these two equations.

For $Cl_t^{\geq}(x)$:

$$\bar{P}(Cl_t^{\geq}) = \{x \in U: D_p^-(x) \cap Cl_t^{\geq} \neq \emptyset\} \quad (40)$$

For Cl_t^{\leq} :

$$\bar{P}(Cl_t^{\leq}) = \{x \in U: D_p^+(x) \cap Cl_t^{\leq} \neq \emptyset\} \quad (41)$$

And Boundary region of each class union is calculated by calculating the difference between Upper and Lower approximations.

$$B_{nC} = \bar{P}(Cl_t^{\leq}) - \underline{P}(Cl_t^{\leq}) \quad (42)$$

The Boundary region, Lower and Upper approximations of class union $Cl_I^{\leq}(x)$ are as follows:

$$\underline{P}(Cl_I^{\leq}) = \{X_3, X_7\}$$

$$\bar{P}(Cl_I^{\leq}) = \{X_1, X_3, X_4, X_6, X_7\}$$

$$B_{nC}(Cl_I^{\leq}) = \{X_1, X_3, X_4, X_6, X_7\} - \{X_3, X_7\}$$

$$B_{nC}(Cl_I^{\leq}) = \{X_1, X_4, X_6\}$$

The Boundary region, Lower and Upper approximations of class union $Cl_{II}^{\leq}(x)$ are as follows:

$$\underline{P}(Cl_{II}^{\leq}) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$\bar{P}(Cl_{II}^{\leq}) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$B_{nC}(Cl_{II}^{\leq}) = \{X_1, X_3, X_4, X_5, X_6, X_7\} - \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$B_{nC}(Cl_{II}^{\leq}) = \{\Phi\}$$

The Boundary region, Lower and Upper approximations of class union $Cl_{II}^{\geq}(x)$ are as follows:

$$\underline{P}(Cl_{II}^{\geq}) = \{X_2, X_5\}$$

$$\bar{P}(Cl_{II}^{\geq}) = \{X_1, X_2, X_4, X_5, X_6\}$$

$$B_{nC}(Cl_{II}^{\geq}) = \{X_1, X_2, X_4, X_5, X_6\} - \{X_2, X_5\}$$

$$B_{nC}(Cl_{II}^{\leq}) = \{X_1, X_4, X_6\}$$

The Boundary region, Lower and Upper approximations of class union $Cl_I^{\leq}(x)$ are as follows:

$$\underline{P}(Cl_{III}^{\geq}) = \{X_2\}$$

$$\bar{P}(Cl_{III}^{\geq}) = \{X_2\}$$

$$B_{nC}(Cl_{III}^{\geq}) = \{X_2\} - \{X_2\}$$

$$B_{nC}(Cl_{III}^{\leq}) = \{\Phi\}$$

So, from all the calculated boundary regions we get that

$$B_{nC}(Cl_{II}^{\leq}) = B_{nC}(Cl_{III}^{\geq})$$

And

$$B_{nC}(Cl_I^{\leq}) = B_{nC}(Cl_{II}^{\geq})$$

Step 6: We have calculated Boundary regions of the dataset and these regions have given us those elements of the universe which belong to the doubtful region. Now in this step, by subtracting those elements from Universe we will get those elements that belong to the undoubtful region. This set of objects that are not in any doubtful region is called “*AlphaSet*”. In the conventional approach, this *AlphaSet* is identified by subtracting the union of all the boundary region objects from the Universe. With the help of this *AlphaSet*, we will calculate the dependency or quality of the sorting of the dataset. So, now the elements belonging to the doubtful region are:

$$B_{nC}(Cl_I^{\leq}) \cup B_{nC}(Cl_{II}^{\leq}) = \{X_1, X_4, X_6\} \cup \{\Phi\}$$

$$B_{nC}(Cl_I^{\leq}) \cup B_{nC}(Cl_{II}^{\leq}) = \{X_1, X_4, X_6\}$$

And the *AlphaSet* is calculated as follows:

$$AlphaSet = U - (B_{nC}(Cl_I^{\leq}) \cup B_{nC}(Cl_{II}^{\leq}))$$

$$AlphaSet = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7\} - \{X_1, X_4, X_6\}$$

$$AlphaSet = \{X_2, X_3, X_5, X_7\}$$

Finally, the dependency of a dataset is calculated by dividing the cardinality of the *AlphaSet* with the cardinality of the Universe.

$$\text{Quality of sorting } \gamma P^{(cl)} = \frac{\text{AlphaSet}}{\text{Universe}} \Rightarrow \frac{4}{7}$$

The elements of *AlphaSet* are very important for the reduct generation process because they are not in any doubtful region and they must remain out of every doubtful region for every possible reduct.

Now, we will find the *AlphaSet* of our case study using our proposed methodology to make sure that the proposed methodology is as accurate as the conventional one. To prove that our method is as accurate as the conventional method, the *AlphaSet* from our methodology must have the same cardinality and must possess the same instances of Universe which are present in the *AlphaSet* of the conventional method.

5.3. Case Study using Proposed Methodology

The proposed approach is illustrated using a case study. For this purpose, Table II is used as a dataset. There are seven records and two conditional attributes in Table II. The proposed method has two phases, the first phase is to generate the dominance relation matrix and the second phase is to compare all the records using proposed dominance-based dependency classes. Following is the descriptions of both of these phases:

Phase 1: As there were seven records in Table II, so the relation matrix of order 7×7 was generated as shown in Table VII. The relation between records is illustrated using mathematical symbols. Every row depicts the relation of an individual record with all the records of the dataset. Like, the first row shows the dominance relation of X_1 with all the records of the dataset. In Table VII, equal to sign (=) represents the identical relation, not equal to (\neq) sign represents indiscernible relation, greater than equal to (\geq) sign represents that record is dominating the other record and less than equal to (\leq) sign represents that record is being dominated by another record.

Table VII: Dominance Relation Matrix of Table II for proposed method

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_1	=	\leq	\geq	=	\neq	=	\geq
X_2	\geq	=	\geq	\geq	\geq	\geq	\geq
X_3	\leq	\leq	=	\leq	\leq	\leq	\neq
X_4	=	\leq	\geq	=	\neq	=	\geq
X_5	\neq	\leq	\geq	\neq	=	\neq	\geq
X_6	=	\leq	\geq	=	\neq	=	\geq
X_7	\leq	\leq	\neq	\leq	\leq	\leq	=

Phase 2: After the dominance relation matrix was generated, the second phase of the proposed methodology was performed. In 2nd phase, two arrays of variable size named *AlphaSet* (it provides a set of records to calculate dependency) and *Checked_Objects* (avoids comparison repetitions by storing those records which have already been compared) were created. Then using a double nested loop, a comparison of records was carried out. The outer loop was used to select the records iteratively from the Universe and the inner loop was used to select the records from *Checked_Objects* array. After that, both of these selected records were compared. This comparison was performed based on the proposed dominance-based dependency classes.

Here, results are shown after each iteration of the outer loop which was executed seven times because our dataset has seven records. Whereas, inner loop ran according to the size of the *Checked_Objects* array which was incremented by one after every iteration of the outer loop.

- **1st Iteration**

Based on “Initial value class” X_1 was selected as X_1 was the first record and at that time, there were no records in *Checked_Objects* array to compare with. So, X_1 was added in both arrays. Members of *AlphaSet* and *Checked_Objects* arrays after the 1st iteration were as follows:

$$Checked_Objects = \{X_1\}$$

$$AlphaSet = \{X_1\}$$

X_1						
-------	--	--	--	--	--	--

Records in *Checked_Objects*

X_1						
-------	--	--	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 1st iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{0 + 1}{0 + 1} = 1 \quad (43)$$

Initially, $|AlphaSet|' = 0$ and $|Checked_Objects|' = 0$ because we were traversing the first record of the dataset.

- **2nd Iteration**

In this iteration, X_2 was taken from the Universe and at that time *Checked_Objects* was not empty. So, we compared X_2 with X_1 and based on “*True positive class*” X_2 was selected and added into both arrays because it did not cause any inconsistency in the dataset. Members of *AlphaSet* and *Checked_Objects* arrays after the 2nd iteration were as follows:

$$Checked_Objects = \{X_1, X_2\}$$

$$AlphaSet = \{X_1, X_2\}$$

X_1	X_2					
-------	-------	--	--	--	--	--

Records in *Checked_Objects*

X_1	X_2					
-------	-------	--	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 2nd iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{1 + 1}{1 + 1} = 1 \quad (44)$$

- **3rd Iteration**

In this iteration, X_3 was selected and compared with X_1 and X_2 based on proposed dependency classes. While comparing X_3 did not cause any inconsistency. So, as a result X_3 was added in both arrays. Members of *AlphaSet* and *Checked_Objects* arrays after the 3rd iteration were as follows:

$$Checked_Objects = \{X_1, X_2, X_3\}$$

$$AlphaSet = \{X_1, X_2, X_3\}$$

X_1	X_2	X_3				
-------	-------	-------	--	--	--	--

Records in *Checked_Objects*

X_1	X_2	X_3				
-------	-------	-------	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 3rd iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{2 + 1}{2 + 1} = 1 \quad (45)$$

- **4th Iteration**

In this iteration, X_4 was selected and compared with all the records in *Checked_Objects* array. Comparison between X_1 and X_4 showed that their decision class was not the same but they had dominance relation of being identical. So, based on “*Distinct Decision Class*” both the records were excluded from *AlphaSet* because they were causing inconsistency in the dataset. Members of *AlphaSet* and *Checked_Objects* arrays after the 3rd iteration were as follows:

$$Checked_Objects = \{X_1, X_2, X_3, X_4\}$$

$$AlphaSet = \{X_2, X_3\}$$

X_1	X_2	X_3	X_4			
-------	-------	-------	-------	--	--	--

Records in *Checked_Objects*

X_2	X_3					
-------	-------	--	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 4th iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked_Objects|' + 1} = \frac{3 - 1}{3 + 1} = 0.5 \quad (46)$$

X_1 was the only record that had caused inconsistency when compared with X_4 . So, X_1 was removed from *AlphaSet* and we put $N = 1$.

- **5th Iteration**

In this iteration, no inconsistencies were found during the comparison of X_5 with all the records in *Checked_Objects* array. So, X_5 was added into both arrays. Members of *AlphaSet* and *Checked_Objects* arrays after the 5th iteration were as follows:

$$Checked_Objects = \{X_1, X_2, X_3, X_4, X_5\}$$

$$AlphaSet = \{X_2, X_3, X_5\}$$

X_1	X_2	X_3	X_4	X_5		
-------	-------	-------	-------	-------	--	--

Records in *Checked_Objects*

X_2	X_3	X_5				
-------	-------	-------	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 5th iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{2 + 1}{4 + 1} = 0.6 \quad (47)$$

- **6th Iteration**

In this iteration, X_6 was selected and compared with all the members of *Checked_Objects* array. Comparison between X_6 and X_4 showed that their decision class was not the same but they had dominance relation of being identical. So, based on “*Distinct Decision Class*” both the records were excluded from *AlphaSet* because they were causing inconsistency in the dataset. Members of *AlphaSet* and *Checked_Objects* arrays after the 6th iteration were as follows:

$$Checked_Objects = \{X_1, X_2, X_3, X_4, X_5, X_6\}$$

$$AlphaSet = \{X_2, X_3, X_5\}$$

X_1	X_2	X_3	X_4	X_5	X_6	
-------	-------	-------	-------	-------	-------	--

Records in *Checked_Objects*

X_2	X_3	X_5				
-------	-------	-------	--	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 6th iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|U|' + 1} = \frac{3 - 0}{5 + 1} = 0.5 \quad (48)$$

X_4 was the only record that had caused inconsistency when compared with X_6 . As X_4 was already removed from *AlphaSet* so, we put $N = 0$.

- **7th Iteration**

In this iteration, no inconsistency was found during the comparison of X_7 with all the members of *Checked_Objects* array. So, X_7 was added in both of the arrays. Members of *AlphaSet* and *Checked_Objects* arrays after the 7th iteration were as follows:

$$Checked_Objects = \{X_1, X_2, X_3, X_4, X_5, X_6, X_7\}$$

$$AlphaSet = \{X_2, X_3, X_5, X_7\}$$

X_1	X_2	X_3	X_4	X_5	X_6	X_7
-------	-------	-------	-------	-------	-------	-------

Records in *Checked_Objects*

X_2	X_3	X_5	X_7			
-------	-------	-------	-------	--	--	--

Records in *AlphaSet*

The updated dependency of the dataset after the 7th iteration was as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked_Objects|' + 1} = \frac{3 + 1}{6 + 1} = 0.57 \quad (49)$$

After complete traversal of the dataset, the cardinality of the *Checked_Objects* becomes similar to the cardinality of the Universe. Dependency after the final iteration of the dataset depicts the overall dependency of the dataset. Members of *AlphaSet* after the final iteration are vital for generating the reducts of the dataset. The records in *AlphaSet* are not part of any doubtful region. Therefore, every reduct must keep them outside of all the doubtful regions. The overall dependency of the dataset was calculated as follows:

$$\gamma C^{(cl)} = \frac{|AlphaSet|}{|Universe|} = \frac{4}{7} = 0.57 \quad (50)$$

The dependency of the dataset calculated by both of the approaches was the same. This depicts that the proposed approach has calculated accurate dependency as a conventional method.

The comparison of the number of comparison statements (CS) executed to calculate dependency by both methodologies is given below:

In our case study, while executing the conventional algorithm we found that there were four union classes. To calculate the $D_p^+(x)$ and $D_p^-(x)$, records from each union class were compared with all other records of the Universe. The following were the four union classes extracted from Table II.

$$Cl_1^{\leq}(x) = \{X_3, X_4, X_7\}$$

$$Cl_2^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$$

$$Cl_2^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$$

$$Cl_3^{\geq}(x) = \{X_2\}$$

Here, we have calculated the number of comparison statements for finding $D_p^+(x)$ and $D_p^-(x)$ of all the records of the union classes.

⇒ In $Cl_1^{\leq}(x)$, the class union had three records. These three records were compared with all seven records of the dataset to calculate both $D_p^+(x)$ and $D_p^-(x)$. So, comparison statements were executed $2 * (3 * 7) = 42$ times in total.

- ⇒ In $Cl_2^{\leq}(x)$, the class union had six records. These six records were compared with all seven records of the dataset to calculate both $D_P^+(x)$ and $D_P^-(x)$. So, comparison statements were executed $2 * (6 * 7) = 84$ times in total.
- ⇒ In $Cl_2^{\geq}(x)$, the class union had four records. These four records were compared with all seven records of the dataset to calculate both $D_P^+(x)$ and $D_P^-(x)$. So, comparison statements were executed $2 * (4 * 7) = 56$ times in total.
- ⇒ In $Cl_3^{\geq}(x)$, the class union had only one record. This single record was compared with all seven records of the dataset to calculate $D_P^+(x)$ and $D_P^-(x)$. So, comparison statements were executed $2 * (1 * 7) = 14$ times in total.

Therefore, in the 2nd step of the conventional method comparison statements were executed a total of 196 times. In the 3rd step of the conventional method, upper and lower approximations for every class union were calculated and the number of comparison statements for the 3rd step was calculated as follows:

- ⇒ For lower approximations of two downward class unions, we compared these class unions with all the seven $D_P^-(x)$. So, comparison statements were executed $2 * 7 = 14$ times.
- ⇒ For upper approximations of two downward class unions, we compared these class unions with all the seven $D_P^+(x)$. So, comparison statements were executed $2 * 7 = 14$ times.
- ⇒ For lower approximations of two upward class unions, we compared these class unions with all the seven $D_P^+(x)$. So, comparison statements were executed $2 * 7 = 14$ times.
- ⇒ For upper approximations of two upward class unions, we compared these class unions with all the seven $D_P^-(x)$. So, comparison statements were executed $2 * 7 = 14$ times.

The total number of comparison statements executed in the 3rd step was $4 * 14 = 56$. Therefore, to get the lower and upper approximations of these four class unions we had to execute $196 + 56 = 252$ comparison statements in total. Finally, to calculate the dependency of the dataset six more comparison statements were executed. After calculating the dominance relation matrix we had to execute a total of $252 + 6 = 258$ comparison statements to get the dependency of our case study dataset using the conventional method.

Whereas, while using the proposed methodology to calculate the dependency of the dataset we had to execute a double nested loop. The number of comparison statements for the proposed methodology was calculated as follows:

- ⇒ The outer loop was executed 7 times which represents the cardinality of the Universe.
- ⇒ The inner loop ran based on the size of the *Checked_Objects* array which was incremented by one after every iteration of the outer loop.
- ⇒ We compared all the seven records of the Universe with all the records in the *Checked_Objects* array.
- ⇒ Like when the 1st object from the Universe was selected no comparison was made because *Checked_Objects* array was empty. Similarly, when the 2nd object was selected 1 comparison was made and when the 3rd object was selected 2 comparisons were made. This way the comparison statements for every object of the universe were different.
- ⇒ Therefore, to extract the *AlphaSet* of our example dataset, comparison statements were executed $0 + 1 + 2 + 3 + 4 + 5 + 6 = 21$ times in total.
- ⇒ This above summation of the number of comparison statements can be simplified as shown in equation 51.

$$\text{Number of CS} = \text{number of records} * \frac{(\text{number of records} - 1)}{2} \quad (51)$$

- ⇒ This difference in the number of comparison statements is shown in Table VIII.

$$\% \text{ reduction in the number of CS} = 100 - \left(\frac{CS_1}{CS_2} \right) * 100 \quad (52)$$

In the formula mentioned in equation 52, CS_1 represents the number of comparison statements for the proposed method and CS_2 represents the number of comparison statements for the conventional method. Therefore, based on this formula percentage reduction in the number of comparison statements was 91.86%. Due to this reduction in the number of comparison statements, IDDC has shown a considerable amount of reduction in execution time, memory usage and computational

complexity. This reduction shown by IDDC in the use of computational resources is further elaborated in the “Comparative Analysis” chapter with the help of multiple datasets. Table VIII shows the comparison of the number of comparisons for both of the methodologies.

Table VIII: Number of comparison statements

	Conventional Algorithm	Proposed Algorithm
Comparison statements	Runs 258 times	Runs 21 times

Chapter 6

Comparative Analysis

CHAPTER 6: COMPARATIVE ANALYSIS

The previous chapter dealt with the implementation and validation aspects of the proposed work. In this chapter, the proposed methodology is compared using different datasets and implementing fast reduct generating algorithm. Our proposed approach provided a major contribution in the field of dominance-based rough set approach. In the section 6.1, we have presented a comparative analysis of our proposed approach and the conventional approach. It is evident from the comparison that our approach has shown accurate results but using fewer resources.

For experimental analysis, a comparison framework was developed and conclusions were made based on conditions specified in that framework. A detailed description of the framework is provided here.

6.1. Comparison Framework

There are four main components of this comparison framework. These components are:

- The execution environment for both approaches.
- The accuracy of the generated results.
- The execution time of approaches to generate results.
- Memory utilization by both approaches.
- The asymptotic computational complexity of both approaches.

6.1.1. Execution Environment

The execution environment used to perform the comparative analysis was a desktop computer having processor intel core i7 3.70GHZ, 16GB RAM and Graphics Card of 6GB, GTX1060 Nvidia. No extra process is in working condition when the algorithms are running. During implementation, special care and attention are given to make sure that the processor condition remains the same throughout.

6.1.2. Accuracy

The term accuracy describes the correctness of the results produced. In our comparison of both approaches, the dependency calculated by using the IDDC was accurate and similar to the

dependency calculated by the conventional approach. The reducts generated by both approaches were also similar. Tables IX and X clearly show that the size of the *AlphaSets* and the number of generated Reducts by both methodologies were identical for every dataset. Therefore, it is evident from the comparison that both the proposed methodology and the conventional approach are accurate.

Table IX: Comparison of Size of AlphaSet

S. NO.	Dataset	<i>AlphaSet</i> 's Size (Conventional)	<i>AlphaSet</i> 's size (Proposed)
1	Iris	145	145
2	Abalone	342	342
3	Breast cancer Wisconsin	667	667
4	Wine Quality	3081	3081
5	Electrical Grid Stability data	10000	10000
6	EEG Eye State	2673	2673
7	Pen-Based Handwritten Digit	7294	7294
8	Hepatitis C Virus	1385	1385
9	Musk 2	6598	6598
10	ISOLET	7797	7797

Table X: Comparison of the number of Reducts Generated

S. NO.	Dataset	Reducts by Conventional Method	Reducts by Proposed Method
1	Iris	2	2
2	Abalone	1	1
3	Breast cancer Wisconsin	2	2
4	Wine Quality	1	1
5	Electrical Grid Stability data	256	256
6	EEG Eye State	1	1
7	Pen-Based Handwritten Digit	2	2

8	Hepatitis C Virus	1	1
9	Musk 2	1	1
10	ISOLET	2	2

6.1.3. Execution Time

Execution time specifies the time taken by an algorithm to generate the output. It is a key factor to distinguish the algorithm's performance in terms of how fast it produces the results. Therefore, to compare the execution time of both proposed and conventional approaches, their execution time was observed using the system stopwatch, which was started after feeding the input and stopped when results were produced. Then we used the following formula to calculate the percentage reduction in the execution time of IDDC.

$$\% \text{ reduction in execution time} = 100 - \left(\frac{T1}{T2} \right) * 100 \quad (53)$$

In equation 53, $T1$ represents the time taken by the proposed approach and $T2$ represents the time taken by the conventional approach. Table VIII shows the time taken by both methods for all the datasets. For instance, the time taken by both proposed and conventional approaches for the *Abalone* dataset was 54 seconds and 107 seconds, respectively. The percentage reduction in the execution time of IDDC for the *Abalone* dataset is almost 50% which was calculated using the formula from equation 53.

$$\% \text{ reduction in execution time} = 100 - \left(\frac{54}{107} \right) * 100 \cong 50\%$$

For the *Iris* dataset, the conventional algorithm code took 0.10 seconds to run and the proposed algorithm code took 0.06 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 40% in the case of the *Iris* dataset.

$$\% \text{ reduction in execution time of } \mathbf{Iris} = 100 - \left(\frac{0.06}{0.10} \right) * 100 \cong 40\%$$

For the *Breast Cancer Wisconsin (BCW)* dataset, the conventional algorithm code took 3.4 seconds to run and the proposed algorithm code took 1.9 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 44% in the case of the *BCW* dataset.

$$\% \text{ reduction in execution time of } \mathbf{BCW} = 100 - \left(\frac{1.9}{3.4}\right) * 100 \cong 44\%$$

For the *Wine Quality* dataset, the conventional algorithm code took 96 seconds to run and the proposed algorithm code took 51 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 47% in the case of the *Wine Quality* dataset.

$$\% \text{ reduction in execution time of } \mathbf{Wine Quality} = 100 - \left(\frac{51}{96}\right) * 100 \cong 47\%$$

For the *Electrical Grid Stability (EGS)* dataset, the conventional algorithm code took 377 seconds to run and the proposed algorithm code took 200 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 47% in the case of the *EGS* dataset.

$$\% \text{ reduction in execution time of } \mathbf{EGS} = 100 - \left(\frac{200}{377}\right) * 100 \cong 47\%$$

For the *EEG Eye State* dataset, the conventional algorithm code took 516 seconds to run and the proposed algorithm code took 278 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 46% in the case of the *EEG Eye State* dataset.

$$\% \text{ reduction in execution time of } \mathbf{EEG Eye State} = 100 - \left(\frac{278}{516}\right) * 100 \cong 46\%$$

For the *Pen-Based Handwritten Digit (PBHD)* dataset, the conventional algorithm code took 244 seconds to run and the proposed algorithm code took 124 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 49% in the case of the *PBHD* dataset.

$$\% \text{ reduction in execution time of } \mathbf{PBHD} = 100 - \left(\frac{124}{244}\right) * 100 \cong 49\%$$

For the *Hepatitis C Virus (HCV)* dataset, the conventional algorithm code took 10 seconds to run and the proposed algorithm code took 5.3 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 47% in the case of the *HCV* dataset.

$$\% \text{ reduction in execution time of } \mathbf{HCV} = 100 - \left(\frac{5.3}{10}\right) * 100 \cong 47\%$$

For the *MUSK 2* dataset, the conventional algorithm code took 197 seconds to run and the proposed algorithm code took 110 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 44% in the case of the *MUSK 2* dataset.

$$\% \text{ reduction in execution time of } \mathbf{MUSK 2} = 100 - \left(\frac{110}{197}\right) * 100 \cong 44\%$$

For the *Isolet* dataset, the conventional algorithm code took 599 seconds to run and the proposed algorithm code took 329 seconds to run. When we put these values in the equation, the percentage decrease in execution time is 45% in the case of the *Isolet* dataset.

$$\% \text{ reduction in execution time of } \mathbf{Isolet} = 100 - \left(\frac{329}{599}\right) * 100 \cong 45\%$$

It can be concluded from Table XI that bigger datasets have shown more percentage reduction in execution time. The percentage decrease in execution time was between 40% to 50% for all datasets. The average of Table XI was calculated using equation 54 to get a better picture of average time reduction.

$$\text{Avg reduction in execution time} = \frac{459}{10} = 46\% \text{ approx} \quad (54)$$

The calculated average time reduction by the proposed methodology was 46%. Therefore, it can be concluded that the proposed methodology can save almost half of the execution time as compared to the conventional method. Figure 14 provides a pictorial comparison of both approaches.

Table XI: Comparison of Execution time

S. NO.	Dataset	Conventional Time (secs)	Proposed Time (secs)	% Reduction in Time Taken/
1	Iris	0.10	0.06	40%
2	Abalone	107	54	50%
3	Breast cancer Wisconsin	3.4	1.9	44%
4	Wine Quality	96	51	47%
5	Electrical Grid Stability data	377	200	47%
6	EEG Eye State	516	278	46%
7	Pen-Based Handwritten Digit	244	124	49%
8	Hepatitis C Virus	10	5.3	47%
9	Musk 2	197	110	44%
10	ISOLET	599	329	45%

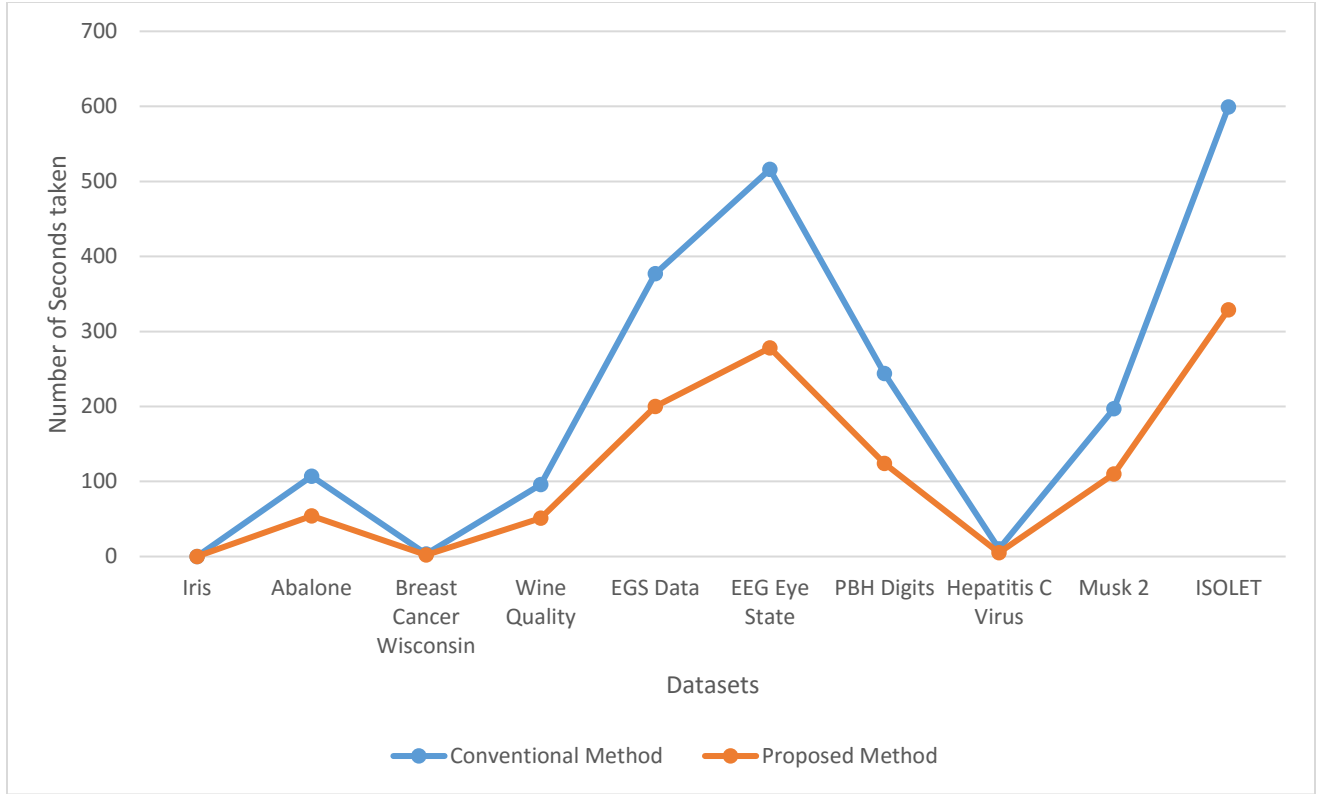


Figure 14: Graphical Comparison of Execution time

6.1.4. Memory Usage

Memory usage specifies the maximum amount of runtime memory taken by the algorithm during its execution. Memory utilization is a very important parameter to judge the efficiency of an algorithm. Algorithms that utilize less storage space are mostly preferred. We have manually calculated the memory by adding the sizes of the intermediate data structures used by both approaches. However, the common variables used by both approaches were ignored. Two arrays were used by the conventional method. One of these arrays was a one-dimensional (1-D) array and second was a two-dimensional (2-D) array. The 1-D array was used to store class unions Cl^{\geq} , Cl^{\leq} and the 2-D array was used to store the D_p^+ and D_p^- for all the records in class unions. The formula used for the calculation of memory usage of the conventional method is mentioned in equation 55.

$$\begin{aligned}
 \text{Memory} &= 2 * \text{Size of datatype} * (|U|) \\
 &+ \text{Size of datatype} * (|U| * |U|)
 \end{aligned} \tag{55}$$

On the contrary, the proposed methodology has used only two 1-Dimensional arrays. The first array (*Checked_Objects*) was used to store all those instances which were checked once and this helped us avoid repetitive comparisons of the same records of Universe. The second array (*AlphaSet*) was used to store those records which were selected after comparison. Therefore, to calculate the memory usage of the proposed methodology, the formula showed equation 56 was used.

$$Memory = 2 * (Size\ of\ datatype) * (|U|) \quad (56)$$

The formula used for calculating the percentage reduction in memory utilization is given in equation 57. In this formula, $M1$ represents memory used by the proposed methodology and $M2$ represents memory used by conventional methodology.

$$\% \text{ Reduction in memory usage} = 100 - \left(\frac{M1}{M2}\right) * 100 \quad (57)$$

For the *Iris* dataset, the conventional algorithm utilizes 0.005 MB of memory and the proposed algorithm has utilized 0.001 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 98% in the case of the *Iris* dataset.

$$\% \text{ reduction in memory usage of } \mathbf{Iris} = 100 - \left(\frac{0.001}{0.005}\right) * 100 \cong 98\%$$

For the *Abalone* dataset, the conventional algorithm utilizes 34.9 MB of memory and the proposed algorithm has utilized 0.03 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the *Abalone* dataset.

$$\% \text{ reduction in memory usage of } \mathbf{Abalone} = 100 - \left(\frac{0.03}{34.9}\right) * 100 \cong 99\%$$

For the *Breast Cancer Wisconsin (BCW)* dataset, the conventional algorithm utilizes 0.98 MB of memory and the proposed algorithm has utilized 0.005 MB of memory. When we put these values

in the equation, the percentage decrease in memory utilization is 99% in the case of the **BCW** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{BCW} = 100 - \left(\frac{0.005}{0.98} \right) * 100 \cong 99\%$$

For the **Wine Quality** dataset, the conventional algorithm utilizes 48 MB of memory and the proposed algorithm has utilized 0.04 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **Wine Quality** dataset.

$$\% \text{ reduction in memory usage } \mathbf{Wine Quality} = 100 - \left(\frac{0.04}{48} \right) * 100 \cong 99\%$$

For the **Electrical Grid Stability (EGS)** dataset, the conventional algorithm utilizes 200 MB of memory and the proposed algorithm has utilized 0.08 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **EGS** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{EGS} = 100 - \left(\frac{0.08}{200} \right) * 100 \cong 99\%$$

For the **EEG Eye State (EES)** dataset, the conventional algorithm utilizes 208 MB of memory and the proposed algorithm has utilized 0.08 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **EES** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{EES} = 100 - \left(\frac{0.08}{208} \right) * 100 \cong 99\%$$

For the **Pen-Based Handwritten Digit (PBHD)** dataset, the conventional algorithm utilizes 241.8 MB of memory and the proposed algorithm has utilized 0.09 MB of memory. When we put these

values in the equation, the percentage decrease in memory utilization is 99% in the case of the **PBHD** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{PBHD} = 100 - \left(\frac{0.09}{241.8} \right) * 100 \cong 99\%$$

For the **Hepatitis C Virus(HCV)** dataset, the conventional algorithm utilizes 3.8 MB of memory and the proposed algorithm has utilized 0.01 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **HCV** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{HCV} = 100 - \left(\frac{0.01}{3.8} \right) * 100 \cong 99\%$$

For the **Musk 2** dataset, the conventional algorithm utilizes 87.1 MB of memory and the proposed algorithm has utilized 0.05 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **Musk 2** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{Musk\ 2} = 100 - \left(\frac{0.05}{87.1} \right) * 100 \cong 99\%$$

For the **Isolet** dataset, the conventional algorithm utilizes 121.6 MB of memory and the proposed algorithm has utilized 0.006 MB of memory. When we put these values in the equation, the percentage decrease in memory utilization is 99% in the case of the **Isolet** dataset.

$$\% \text{ reduction in memory usage of } \mathbf{Isolet} = 100 - \left(\frac{0.06}{121.6} \right) * 100 \cong 99\%$$

A comparison of memory usage of both of these methodologies is shown in Table XII. On average, a 98% reduction in required runtime memory was found for ten datasets. Reduction in the memory usage is shown in the Table XIII. It is evident from the comparison that a huge amount of memory was saved by the proposed approach.

Table XII: Comparison of Memory Usage

S. NO.	Dataset	Dataset's Number of Records	Memory by Conventional (MB)	Memory by Proposed (MB)
1	Iris	150	0.05	0.0012
2	Abalone	4177	34.9	0.33
3	Breast cancer Wisconsin	699	0.98	0.006
4	Wine Quality	4898	48	0.04
5	Electrical Grid Stability data	10000	200	0.08
6	EEG Eye State	10200	208	0.081
7	Pen-Based Handwritten Digit	10992	241	0.09
8	Hepatitis C Virus	1380	3.8	0.01
9	Musk 2	6598	87.1	0.05
10	ISOLET	7797	121.6	0.06

Table XIII: Percentage Reduction in Memory Usage

S. NO.	Dataset	Memory by Conventional (MB)	Memory by Proposed (MB)	% Reduction in usage
1	Iris	0.05	0.0012	≈98%
2	Abalone	34.9	0.33	≈99%
3	Breast cancer Wisconsin	0.98	0.006	≈99%
4	Wine Quality	48	0.04	≈99%
5	Electrical Grid Stability data	200	0.08	≈99%
6	EEG Eye State	208	0.081	≈99%
7	Pen-Based Handwritten Digit	241	0.09	≈99%
8	Hepatitis C Virus	3.8	0.01	≈99%

9	Musk 2	87.1	0.05	≈99%
10	ISOLET	121.6	0.06	≈99%

6.1.5. Complexity

The complexity of an algorithm depends on many factors such as code execution time, memory usage, etc. The Big-O notation was used for the complexity analysis of both approaches. Big-O is a mathematical notation [46,47]. It classifies algorithms based on the number of steps, execution time and the number of inputs. For example, if both nested loops run n times then Big-O will be $O(n^2)$. Algorithms with less complexity are usually preferred. For comparison purposes, Big-O complexity for only those parts of algorithms was calculated which were different in both approaches. The conventional algorithm which is based on lower and upper approximations comprises three main steps. The Big-O for each of these three main steps was calculated as follows:

- Big-O for the first step of the conventional algorithm was $O(U)$ because a single loop was executed U times. Here, U represents the cardinality of the Universe. The algorithm had four class unions and this step was executed for all four class unions. Therefore, the Big-O for 1st step became $4 * (O(U))$.
- Big-O for the second step of the conventional algorithm was $O(Cl * U)$ because a double nested loop was executed. Here, Cl represents the cardinality of the class unions. The algorithm had four class unions and this step was executed for all four class unions. Therefore, the Big-O for 2nd step became $4 * (O(Cl * U))$.
- Big-O for the third step of the conventional algorithm was $O(Cl * Cl * D_P^+ / D_P^-)$ because a triple nested loop was executed. Here, D_P^+ / D_P^- represents the cardinality of the dominance sets. The algorithm had four class unions and this step was executed for all four class unions. Therefore, the Big-O for 3rd step became $4 * (O(Cl * Cl * D_P^+ / D_P^-))$. The three loops in this step were the main reason for the high computational cost.

For the complete conventional algorithm, Big-O was as follows:

$$4 * (O(U)) + (4 * (O(Cl * U))) + (2 * O(Cl * Cl * D_P^+)) + (2 * O(Cl * Cl * D_P^-)) = O(Cl * Cl * D_P^+ / D_P^-).$$

In a worst-case scenario, Cl and D_P^+/D_P^- can be equal to the cardinality of the Universe. So in that case, Big-O will become $O(Cl * Cl * D_P^+/D_P^-) \cong O(U^3)$.

Whereas in the proposed approach, approximations were not calculated instead of that the objects of Universe were compared based on proposed dominance-based dependency classes. Big-O complexity of the proposed approach was calculated as follows:

- For the comparison of records, the proposed approach uses a double nested loop. For the outer loop, Big-O was $O(U)$ because we had to compare all the records of the Universe.
- The inner loop runs equal to the cardinality of the *Checked_Objects* Array. The cardinality of the *Checked_Objects* was incremented after every outer loop's iteration. So, Big-O at worst for inner loop was also $O(U)$. Therefore, the overall Big-O for the proposed approach was $O(U) * O(U) = O(U^2)$.

It can be seen after calculating the Big-O notations for both the approaches that the complexity of the conventional approach was $O(U^3)$ which is reduced by the proposed approach to $O(U^2)$. Therefore, with the complexity $O(U^2)$ of the proposed approach is considered suitable for large datasets and the use of the conventional approach becomes inappropriate for such datasets. This comparison of Big-O complexity is also illustrated in Figure 15.

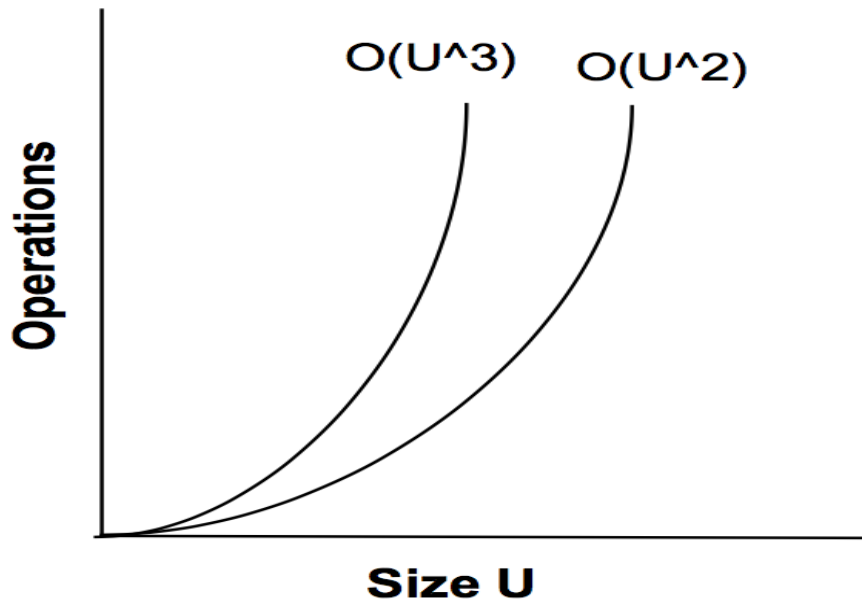


Figure 15: Big-O Complexity Comparison

Chapter 7

Conclusion and Future Work

CHAPTER 7: CONCLUSION AND FUTURE WORK

In this research work, we have proposed a new incremental approach to calculate the dependency of datasets. The proposed approach is called the “Incremental Dominance-based Dependency Calculation (IDDC)”. This method avoids the calculation of approximations and union classes. It incrementally scans all the objects of the Universe and compares them based on proposed dominance-based dependency classes to find the dependency of a dataset. Whereas, the conventional approach uses approximations for calculating the dependency of the datasets. This calculation of approximations comprises three computationally expensive steps that degrade the performance of algorithms. Due to this performance degradation, the conventional approach has become inappropriate for datasets beyond smaller sizes. To avoid such performance bottlenecks, the proposed method incrementally compares every object of the dataset with the rest of the objects and calculates the dependency of the dataset. To justify the effectiveness of the proposed approach, both IDDC and conventional approaches were compared using various datasets from the UCI dataset repository. Results have shown that the proposed approach outperforms the conventional approach by depicting on average 46% and 98% decrease in execution time and required runtime memory, respectively. The Big-O complexity of the algorithm was reduced by the proposed approach from $O(n^3)$ to $O(n^2)$ as well. To validate the effectiveness of the proposed approach, IDDC was implemented using FRGA, GA and PSO algorithms.

The proposed approach is suitable for supervised datasets. However, we may find ourselves in a situation where class labels are not available. So, the future direction is to study how to use the proposed approach for unsupervised datasets as well. Efforts will also be made to use it for incomplete datasets having missing values. In its current state, the proposed approach requires complete datasets with values that follow the dominance principle and datasets having missing values should be pre-processed to fill the missing values. As results have shown that our proposed approach has taken 46% less execution time on average. So, there is a good chance of further decreasing the execution time of the proposed approach by using parallel processing methodologies.

REFERENCES

- [1] Jensen, R., & Shen, Q. (2008). Computational intelligence and feature selection: rough and fuzzy approaches (Vol. 8). John Wiley & Sons.
- [2] R. Bellman, "Dynamic programming: Princeton Univ. press." (1957).
- [3] Qian, W., Shu, W., Yang, B., & Zhang, C. (2015). An incremental algorithm to feature selection in decision systems with the variation of feature set. *Chinese Journal of Electronics*, 24(1), 128-133. DOI: 10.1049/cje.2015.01.021
- [4] Inbarani, H. H., Bagyamathi, M., & Azar, A. T. (2015). A novel hybrid feature selection method based on rough set and improved harmony search. *Neural Computing and Applications*, 26(8), 1859-1888. DOI: 10.1007/s00521-015-1840-0
- [5] John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994* (pp. 121-129). Morgan Kaufmann. DOI: 10.1016/B978-1-55860-335-6.50023-4
- [6] Machado, J. T., & Lopes, A. M. (2017). Multidimensional scaling analysis of soccer dynamics. *Applied Mathematical Modelling*, 45, 642-652. DOI: 10.1016/j.apm.2017.01.029
- [7] Halai, A. D., Woollams, A. M., & Ralph, M. A. L. (2017). Using principal component analysis to capture individual differences within a unified neuropsychological model of chronic post-stroke aphasia: revealing the unique neural correlates of speech fluency, phonology, and semantics. *Cortex*, 86, 275-289. DOI: 10.1016/j.cortex.2016.04.016
- [8] Zhang, H. Y., & Yang, S. Y. (2017). Feature selection and approximate reasoning of large-scale set-valued decision tables based on α dominance-based quantitative rough sets. *Information sciences*, 378, 328-347. Z. Pawlak. "Rough sets-theoretical aspect of reasoning about data." (1991): 29-29. DOI: 10.1016/j.ins.2016.06.028
- [9] Pawlak, Z., Grzymala-Busse, J., Slowinski, R., & Ziarko, W. (1995). Rough sets. *Communications of the ACM*, 38(11), 88-95. DOI:10.1145/219717.219791
- [10] Pawlak, Z., & Slowinski, R. (1994). Decision analysis using rough sets. *International Transactions in Operational Research*, 1(1), 107-114. DOI:10.1016/0969-6016(94)90050-7

- [11] Anaraki, J. R., & Eftekhari, M. (2013, May). Rough set based feature selection: a review. The 5th Conference on Information and Knowledge Technology (pp. 301-306). IEEE. DOI:10.1109/IKT.2013.6620083
- [12] Podsiadło, M., & Rybiński, H. (2014). Rough sets in economy and finance. In Transactions on Rough Sets XVII (pp. 109-173). Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-642-54756-0_6
- [13] Xie, C. H., Liu, Y. J., & Chang, J. Y. (2015). Medical image segmentation using rough set and local polynomial regression. *Multimedia Tools and Applications*, 74(6), 1885-1914. DOI: 10.1007/s11042-013-1723-2
- [14] Pawlak Z. 1982. Rough sets. *International Journal of Computer and Information Science* 11 (Oct. 1982), 341–356.
- [15] Greco, S., Matarazzo, B. & Slowinski, R. (2001). Rough sets theory for multicriteria decision analysis. *European journal of operational research*, 129(1), 1-47. DOI:10.1016/S0377-2217(00)00167-3
- [16] Boggia, A., Rocchi, L., Paolotti, L., Musotti, F., & Greco, S. (2014). Assessing rural sustainable development potentialities using a dominance-based rough set approach. *Journal of environmental management*, 144, 160-167. DOI: 10.1016/j.jenvman.2014.05.021
- [17] Liou, J. J. (2009). Novel decision rules approach for customer relationship management of the airline market. *Expert Systems with Applications*, 36(3), 4374-4381. DOI: 10.1016/j.eswa.2008.05.002
- [18] Liou, J. J., & Tzeng, G. H. (2010). A dominance-based rough set approach to customer behavior in the airline market. *Information Sciences*, 180(11), 2230-2238. DOI: 10.1016/j.ins.2010.01.025
- [19] Liou, J. J., Yen, L., & Tzeng, G. H. (2010). Using decision rules to achieve mass customization of airline services. *European journal of operational research*, 205(3), 680-686. DOI: 10.1016/j.ejor.2009.11.019
- [20] Chakhar, S., Ishizaka, A., Labib, A., & Saad, I. (2016). Dominance-based rough set approach for group decisions. *European Journal of Operational Research*, 251(1), 206-224. DOI: 10.1016/j.ejor.2015.10.060

- [21] BŁaszczyński, J., Greco, S., & SŁowiński, R. (2012). Inductive discovery of laws using monotonic rules. *Engineering Applications of Artificial Intelligence*, 25(2), 284-294. DOI: 10.1016/j.engappai.2011.09.003
- [22] do Couto, A. B. G., & Gomes, L. F. A. M. (2016). Multi-criteria web mining with DRSA. *Procedia Computer Science*, 91, 131-140. DOI: 10.1016/j.procs.2016.07.050
- [23] Rawat, S., Patel, A., Celestino, J., & dos Santos, A. L. M. (2016). A dominance-based rough set classification system for fault diagnosis in electrical smart grid environments. *Artificial Intelligence Review*, 46(3), 389–411. DOI:10.1007/s10462-016-9468-8
- [24] Hu, Q., Chakhar, S., Siraj, S., & Labib, A. (2017). Spare parts classification in industrial manufacturing using the dominance-based rough set approach. *European Journal of Operational Research*, 262(3), 1136-1163. DOI: 10.1016/j.ejor.2017.04.040
- [25] Mohamad, M., & Selamat, A. (2018, March). Analysis of hybrid dominance-based rough set parameterization using private financial initiative unitary charges data. In *Asian Conference on Intelligent Information and Database Systems* (pp. 318-328). Springer, Cham. DOI: 10.1007/978-3-319-75417-8_30
- [26] Augeri, M. G., Colombrita, R., Greco, S., Lo Certo, A., Matarazzo, B., & Slowinski, R. (2011). Dominance-based rough set approach to budget allocation in highway maintenance activities. *Journal of infrastructure systems*, 17(2), 75-85. DOI: : 10.1061/(ASCE)IS.1943-555X.0000051
- [27] Marin, J. C., Zaras, K., & Boudreau-Trudel, B. (2014). Use of the dominance-based rough set approach as a decision aid tool for the selection of development projects in Northern Quebec. *Modern Economy*, 2014. DOI: 10.4236/me.2014.57067
- [28] Susmaga, R. (2014). Reducts and constructs in classic and dominance-based rough sets approach. *Information Sciences*, 271, 45-64. DOI: 10.1016/j.ins.2014.02.100
- [29] Susmaga, R., Słowiński, R., Greco, S., & Matarazzo, B. (2000). Generation of reducts and rules in multi-attribute and multi-criteria classification. *Control and Cybernetics*, 29, 969-988.
- [30] Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Reprint—Preferred Reporting Items for Systematic Reviews and Meta-Analyses: The PRISMA Statement. *Physical Therapy*, 89(9), 873–880. DOI:10.1093/ptj/89.9.873

- [31] Pourahmadi, A., Ebadi, T., & Nikazar, M. (2017). Industrial Wastes Risk Ranking with TOPSIS, Multi-Criteria Decision Making Method. *Civil Engineering Journal*, 3(6), 372–381. DOI:10.28991/cej-2017-00000098
- [32] Azar, A. T., Inbarani, H. H., & Devi, K. R. (2017). Improved dominance rough set-based classification system. *Neural Computing and Applications*, 28(8), 2231-2246. DOI: 10.1007/s00521-016-2177-z
- [33] Hu, Q., Chakhar, S., Siraj, S., & Labib, A. (2017). Spare parts classification in industrial manufacturing using the dominance-based rough set approach. *European Journal of Operational Research*, 262(3), 1136-1163. DOI: 10.1016/j.ejor.2017.04.040
- [34] Bouzayane, S., & Saad, I. (2017, May). Weekly predicting the at-risk MOOC learners using a dominance-based rough set approach. In *European Conference on Massive Open Online Courses* (pp. 160-169). Springer, Cham. DOI: 10.1007/978-3-319-59044-8_18
- [35] Liou, J. J., & Tzeng, G. H. (2010). A dominance-based rough set approach to customer behavior in the airline market. *Information Sciences*, 180(11), 2230-2238. DOI: 10.1016/j.ins.2010.01.025
- [36] Kusunoki, Y., & Inuiguchi, M. (2010). A unified approach to reducts in a dominance-based rough set approach. *Soft Computing*, 14(5), 507-515. DOI: 10.1007/s00500-009-0450-0
- [37] Zhang, J., Wong, J.-S., Li, T., & Pan, Y. (2014). A comparison of parallel large-scale knowledge acquisition using rough set theory on different MapReduce runtime systems. *International Journal of Approximate Reasoning*, 55(3), 896–907. DOI:10.1016/j.ijar.2013.08.003
- [38] Zhang, J., Wong, J.-S., Pan, Y., & Li, T. (2015). A Parallel Matrix-Based Method for Computing Approximations in Incomplete Information Systems. *IEEE Transactions on Knowledge and Data Engineering*, 27(2), 326–339. DOI:10.1109/tkde.2014.2330821
- [39] Liou, J. J. (2011). Variable Consistency Dominance-based Rough Set Approach to formulate airline service strategies. *Applied Soft Computing*, 11(5), 4011-4020. DOI: 10.1016/j.asoc.2011.03.002
- [40] Huang, Q., Li, T., Huang, Y., Yang, X., & Fujita, H. (2020). Dynamic dominance rough set approach for processing composite ordered data. *Knowledge-Based Systems*, 187, 104829. DOI:10.1016/j.knosys.2019.06.037

- [41] Raza, M. S., & Qamar, U. (2019). A parallel approach to calculate lower and upper approximations in dominance-based rough set theory. *Applied Soft Computing*, 84, 105699. DOI:10.1016/j.asoc.2019.105699
- [42] "UCI," [Online]. Available: <https://archive.ics.uci.edu/ml/index.php>.
- [43] Zuhtuogullari, K., Allahverdi, N., & Arikan, N. (2013). Genetic algorithm and rough sets based hybrid approach for reduction of the input attributes in medical systems. *International Journal of innovative computing and information control*, 9, 3015-3037.
- [44] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 4, pp. 1942-1948)*. IEEE. DOI: 10.1109/ICNN.1995.488968
- [45] Shi, Y., & Eberhart, R. C. (1998, March). Parameter selection in particle swarm optimization. In *International conference on evolutionary programming* (pp. 591-600). Springer, Berlin, Heidelberg. DOI: 10.1007/BFb0040810
- [46] Shi, Y. (2001, May). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546) (Vol. 1, pp. 81-86)*. IEEE. DOI: 10.1109/CEC.2001.934374
- [47] Allen Weiss, M. (2004). *Data Structures in C++*. Chapman & Hall/CRC Computer & Information Science Series, 42-1-42-17. DOI: 10.1201/9781420035179.ch42
- [48] Sahni, S. (2004). *Analysis of Algorithms*. Chapman & Hall/CRC Computer & Information Science Series, 1-1-1-25. DOI:10.1201/9781420035179.pt1