Soft Computing, Volume 28, March 2024, pp. 3757-3781
DOI: 10.1007/s00500-023-09567-x

An incremental approach for calculating dominance-based rough set dependency

Rana Muhammad Kaleem Ullah[1], Usman Qamar[2], Muhammad Summair Raza[3], John Ahmet Erkoyuncu[4]
[1,2] Computer & Software Engineering Department, College of Electrical and Mechanical Engineering
National University of Sciences and Technology (NUST), Islamabad, Pakistan
[3] Department of Computer Science, Virtual University, Pakistan
[4] School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, UK
{[1]rana.kaleem16, [2]usmanq}@ceme.nust.edu.pk
[3]sraza@vu.edu.pk
[4]j.a.erkoyuncu@cranfield.ac.uk

ABSTRACT

Feature selection and classification are widely used in machine learning in the context of big data. In many datasets, both attributes and decision classes can be preference ordered. Therefore, to process the data and information based on preference-ordered attributes, Dominance-based Rough Set Approach (DRSA) has been proposed. DRSA considers dominance relation between objects and can process the information with preference-ordered attribute domains. The 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 which is a computationaly expensive task. 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 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).

1 Introduction:

In 1982, Pawlak proposed Rough Set Theory (RST) to analyze the incomplete, inconsistent, and unknown information or data [1,2]. RST has become an important tool to find data reduction, data dependencies, rule induction, and approximate set classification from databases [3]. Since its inception, RST has been comprehensively used for knowledge discovery in economy and finance

[4], medical imaging etc. [5]. 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.

Therefore, to process the data and information based on preference-ordered attributes, Greco et al. [6] have introduced the Dominance-based Rough Set Approach (DRSA) in contrast with RST, DRSA considers dominance relation between objects 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 [7], airline services evaluation [8–10], group decision [11], etc.

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 which is a computationally expensive task.

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.

The proposed methodology has 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.
- 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 rest of the paper is organized as follows. Section 2 discusses some of the core fundamental concepts of DRSA. Section 3 explains the challenges of the conventional method while calculating

dependency. Section 4 presents the related work. The proposed methodology is explained in Section 5. Datasets and the selected algorithm is discussed in Section 6. Section 7 presents the experimental results and analysis. Finally, conclusion and future recommendations are discussed in Section 8.

2 Preliminaries

In RST, a decision system 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 1.

$$\alpha = (U, C \cup D) \tag{1}$$

In DRSA, decision system is represented as shown in equation 2.

$$A = (U, Q, V, f) \tag{2}$$

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 3.

$$V = U_q \in Q\, V_q \tag{3}$$

Here, V_q is the value set of criteria q. f in equation 2 represents a function of the form $f\ (x, q)$ which assigns a particular value V_q to an item x for attribute q. A simple decision system is shown in Table I.

In our sample decision system, Universe only contains a total of seven objects 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 [12]. From the start of DRSA appearance many domains have used it for better results such as multi-criteria web mining [13], fault diagnosis [14], in the manufacturing industry [15], finance projects [16,17], better project selection [18], and in data mining [19]. 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: Decision System

Universe	Mathematics	English	Final-Result
X_1	A	B	Very Good
X_2	A	A	Excellent
X_3	B	C	Good
X_4	A	B	Good
X_5	B	A	Very Good
X_6	A	B	Very Good
X_7	C	B	Good

2.1 Dominance

For a set of criteria $P \subseteq C$, an object x dominates object y if object 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 4. $D_p^-(x)$ is a set of objects that are being dominated by x based on the information in $P \subseteq C$.

$$D_p^-(x) = \{y \in \cup : x D_P y\} \quad (4)$$

Similarly, the set of objects which are dominating x are denoted by $D_p^+(x)$and that can be defined mathematically as shown in equation 5.

$$D_p^+(x) = \{y \in \cup : y D_P x\} \quad (5)$$

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

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

Similarly,

$D_p^+(x_3) = \{x_1, x_2, x_3, x_{4,} x_5, x_6\}$

2.2 Decision classes

In DRSA, decision attributes splits the whole Universe into a finite number of decision classes, $Cl = \{Cl_1, Cl_2, Cl_3, \ldots, Cl_m\}$. DRSA are supposed to be preference ordered. So, for $r, s = \{1, 2, 3 \ldots, m\}$, 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 6 and 7.

$$Cl_t^{\geq}(x) = \cup_{s \geq t} Cl_s \ \ t = 1, \ldots, n. \quad (6)$$
$$Cl_t^{\leq}(x) = \cup_{s \leq t} Cl_s \ \ t = 1, \ldots, n. \quad (7)$$

Here, $Cl_t^{\geq}(x)$ represents the set of objects from class Cl_t or a more preferred class. However, $Cl_t^{\leq}(x)$ represents the set of objects from class Cl_t or to a less preferred class. For example, based on data from Table I the $Cl_t^{\geq}(x)$ and $Cl_t^{\leq}(x)$ for t = 2 are following:

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

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

2.3 Lower approximations

Lower approximations in traditional RST describe the set of those objects 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 objects which will, with certainty belong to $Cl_t^{\geq}(x)$. Likewise, the P-lower approximation of $Cl_t^{\leq}(x)$ will contain all those objects that, with certainty belong to $Cl_t^{\leq}(x)$. Mathematically it is shown in equations 8 and 9.

$$\underline{P}(Cl_t^{\geq}) = \{x \in U: D_P^+(x) \subseteq Cl_t^{\geq}\} \quad (8)$$

$$\underline{P}(Cl_t^{\leq}) = \{x \in U: D_P^-(x) \subseteq Cl_t^{\leq}\} \quad (9)$$

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 I.

1st Step: All the objects 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:

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

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

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

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

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: We calculate lower approximation using the formula aforementioned in equations 8. 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 objects 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 9 is used to calculate lower approximation.

2.4 Upper approximations

In the classical RST approach, the upper approximations describe those sets of objects 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 objects 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 objects that may relate to the class unions $Cl_t^{\leq}(x)$. Mathematically this is shown in equations 10 and 11.

$$\overline{P}(Cl_t^{\geq}) = \{x \in U: D_P^-(x) \cap Cl_t^{\geq} \neq \emptyset\} \quad (10)$$

$$\overline{P}(Cl_t^{\leq}) = \{x \in U: D_P^+(x) \cap Cl_t^{\leq} \neq \emptyset\} \quad (11)$$

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 I, 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 objects relating to the class union $Cl_t^{\geq}$. Objects 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 objects of the identified set are:

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

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

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

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

This step considerably reduces the performance and efficiency because for every object we have to find all those objects dominating it. This searching process needs a whole traversal of the dataset for every individual object. As there are four objects 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 objects relating to the P-upper approximation using equation 10 and this involves selecting those objects 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 11 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 12 [20].

$$\text{Dependency } (\gamma C(Cl)) = \frac{|Universe - \left(\left(\cup_{t\in T} Bn_P(Cl_t^{\geq})\right) \cup \left(\cup_{t\in T} Bn_P(Cl_t^{\leq})\right)|}{|Universe|} \tag{12}$$

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 13 and 14, respectively.

$$Bn_P(Cl_t^{\geq}) = \overline{P}(Cl_t^{\geq}) - \underline{P}\ (Cl_t^{\geq}) \tag{13}$$

$$Bn_P(Cl_t^{\leq}) = \overline{P}(Cl_t^{\leq}) - \underline{P}\ (Cl_t^{\leq}) \tag{14}$$

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"* [20]. 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 12. 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 15.

$$\text{Dependency } (\gamma C(Cl)) = \frac{|AlphaSet|}{|Universe|} \tag{15}$$

3 Challenges in Approximation Calculation

As we have discussed 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 1.

```
For i = 1 to |U|
if Cl_i ≥ Cl_t
Cl_t^≥ = Cl_t^≥ ∪ Cl_i
End − if
End − For
```

Figure 1: Pseudocode for calculating lower approximation (1st Step).

The following are the class unions for our example dataset.

$Cl_1^{\leq}(x) = \{X_3, X_4, X_7\}$ for t =1

$Cl_2^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$ for t = 2

$Cl_2^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$ for t = 2

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

objects of the Universe. This step is repeated for all the objects 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 2.

```
//For Dominance Positive
For i = 1 to |Cl_t^≥ / Cl_t^≤|
  For j=1 to |U|
    If X_j ≥ X_it
      D_P^+(X_i) = D_P^+(X_i) ∪ X_j
    End − If
  End − For
End − For

//For Dominance Negative
For i = 1 to |Cl_t^≥ / Cl_t^≤|
            For j=1 to |U|
    If X_j ≤ X_it
      D_P^-(X_i) = D_P^-(X_i) ∪ X_j
    End − If
  End − For
End − For
```

Figure 2: Pseudocode of calculating lower approximation 2nd Step).

Finally in the 3rd Step, $\underline{P}(Cl_t^{\geq})$ and $\overline{P}(Cl_t^{\geq})$ 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 $D_P^-(x)$. These triple nested loops are computationally very expensive. The pseudocode of step 3 is shown in figure 3.

```
For i = 1 to |Cl_t^≤|
  For j = 1 to |D_N^+(X_i)|
    For k = 1 to |Cl_t^≤|
      Calculate D_N^+(X_ji) ⊆ Cl_kt^≤
    End − For
  End − For
End − For
```

Figure 3: Pseudocode of Step-3 for calculating lower approximation (3rd Step)

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 Related work

Rough Set Theory (RST) is proposed to analyze the incomplete, inconsistent, and unknown information or data [1,2]. However, RST is unable to generate accurate results for preference-ordered datasets and this makes classical RST unsuitable for preference-ordered data domains. Therefore, to process the data and information based on preference-ordered attributes, Greco et al. [6] have introduced the Dominance-based Rough Set Approach (DRSA) in contrast with RST. As the main benefit of DRSA is to process information for preference ordered domains, thus it has been widely used for multi-criteria analysis [22]. 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.

In [23], the paper introduces redefined preliminaries for the dominance-based rough set approach, offering potential improvements in terms of efficiency and effectiveness. In [24], authors have presented a classification technique based on DRSA for the management of spare parts. 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 [25], 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. 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 [26], authors have presented DRSA based technique for forecasting customer's behavior in airline companies. This approach was also based on traditional DRSA and uses upper and lower approximations which can result in the inherent performance drawbacks.

In [27], 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. Moreover, they have demonstrated that all types of reducts can be computed broadly depending on two discernibility matrices related to generalized decisions.

In [28], 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 [29], 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. These both approaches need high hardware capabilities which render this non ideal situation for dependency calculation.

In [30], the authors propose several novel algorithms and methodologies to handle the challenges posed by the evolving nature of data. They provide a comprehensive analysis of the proposed approaches, including experimental results and comparisons with existing methods. But the paper focuses on a specific type of data, namely evolving ordered data. It does not address the applicability of the proposed approaches to other types of data or domains. In [31], The authors propose novel fuzzy rough set models that incorporate fuzzy dominance relations and fuzzy

lower/upper approximations. They provide a detailed analysis of the proposed models, including mathematical formulations and algorithms. This approach is computationally complex and lack real world examples. In [32], the authors introduce an innovative framework that combines ERM with dominance-based rough set approaches, providing valuable insights into improving decision-making processes. However, there are limitations related to practical implementation and scalability of the proposed approach. Whereas, some authors have also combined incremental and parallel approaches to mitigate the bottlenecks of conventional DRSA [33].

A brief review of different research papers along with their comparison is provided in this section. Table II shows a comparative summary of the approaches.

Table II: Comparisons of techniques used in different papers

Algorithms	Technique used	Advantages	Disadvantages
"Spare parts classification in industrial manufacturing using the dominance-based rough set approach" [24]	Dominance-based rough set dependency	• High accuracy	• Uses conventional approach to calculate upper and lower approximations • High complexity
"Weekly predicting the At-Risk MOOC learners using Dominance-Based rough set approach" [25]	Dominance-based rough set theory	• High prediction accuracy	• Conventional approach to calculate approximations • Poor performance
"A dominance-based rough set approach to customer behavior in the airline market" [26]	Dominance-based rough set theory	• High prediction of loyal customers	• Uses conventional approach to calculate dependency • Technique limited to small datasets
"A unified approach to reducts in dominance-based rough set approach" [27]	Dominance-based rough set theory for finding reducts	• New sets of reducts produced	• Uses conventional approach to calculate dependency
"A comparison of parallel large-scale knowledge acquisition using rough set theory on different MapReduce runtime systems" [28]	Parallel based Rough set theory approach	• High efficiency	• Increased hardware requirements
"A Parallel Matrix-Based Method for Computing Approximations in Incomplete Information Systems" [29]	Parallel based Rough set theory approach	• Fast processing time	• Increased hardware requirements
"Improved dominance rough set-based classification system" [34]	Dominance-based rough set dependency	• High classification accuracy	• Uses conventional approach to calculate dependency • Poor performance

"Variable Consistency Dominance-based Rough Set Approach to formulate airline service strategies" [35]	Variable Consistency Dominance-based Rough Set Approach	• Easy to understand rules	• Uses conventional approach to calculate dependency
"Dynamic dominance rough set approach for processing composite ordered data" [36]	Incremental Dominance-based Rough Set Approach	• Fast processing time	• High computational complexity

5 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 $\{\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{U}\}$ can hold as follows:

i. First relation which shows that $\boldsymbol{a}$ is dominant over $\boldsymbol{b}$. In the dominance matrix, it is represented by the symbol "≥".
ii. The second relation shows that $\boldsymbol{a}$ is dominated by $\boldsymbol{b}$. In the dominance matrix, it is represented by the symbol "≤".
iii. The third relation shows that $\boldsymbol{a}$ and $\boldsymbol{b}$ 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 $\boldsymbol{a}$ is greater than or equal to $\boldsymbol{b}$ for certain features and less than and equal to $\boldsymbol{b}$ for remaining features. In the dominance matrix, it is represented by the symbol "≠".

After the calculation of the dominance matrix, the second phase of IDDC is to find the dependency of a dataset by comparing all the objects 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 objects to calculate dependency of the dataset) and *Checked_Objects* (it is used to avoid the comparison repetitions by storing those objects which have already been compared) are created.
- Objects from Universe are taken one-by-one and compared with all the objects 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.
- Objects are added or removed from the *AlphaSet* based on their comparison but every object taken from the Universe is added into the *Checked_Objects* array to avoid repetitive comparisons of objects.
- Proposed dominance-based dependency classes are used to determine the result of every comparison of the objects.
- When all the objects 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*.

5.1 Proposed Dominance-Based Dependency Classes

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

- **Initial value class**

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} \tag{16}$$

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 object which is X_1 from Table I, then the dependency of dataset became:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{0 + 1}{0 + 1} = 1 \tag{17}$$

After adding the first object, $|AlphaSet|' = 1$ 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 objects are used to determine which object is better. Mathematically, for two objects $\boldsymbol{a}, \boldsymbol{b}$ and attribute set $\mathbf{c} \in \mathbf{C}$ first side of this class can be written as shown in equation 18.

$$(C(a) > C(b)\ AND\ D(a) \geq D(b)) \tag{18}$$

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

Similarly, if an object $\boldsymbol{a}$ is dominated by object $\boldsymbol{b}$ with lower decision class then mathematically the other side of this class can be written as shown in equation 19.

$$(c(a) < c(b)\ AND\ D(a) \leq D(b)) \tag{19}$$

Dependency value for this class will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} \tag{20}$$

In our example Table I, 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:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{1+1}{1+1} = 1 \tag{21}$$

Similarly, as we traverse to the 3rd object in Table I 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:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{2+1}{2+1} = 1 \tag{22}$$

- **Distinct Decision Class**

This class defines one of the inconsistencies we face in the dominance-based datasets where values of conditional attributes for two objects are similar yet they have different decision classes. Therefore, this class is called *"Distinct Decision Class"*. Mathematically, For two objects $\boldsymbol{a}, \boldsymbol{b}$ and attribute set $\mathbf{c} \in \mathbf{C}$ this class can be written as shown in equation 23.

$$(c(a) == c(b) \,\&\, D(a) \neq D(b)) \tag{23}$$

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

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked\_Objects|' + 1} \quad (24)$$

Here, N represents the number of objects in *AlphaSet* that cause this inconsistency when compared with this newly traversed object.

For example, as we traverse through Table 1 and compare X_4 with X_1. This comparison shows that both objects have the same attribute values but different decision class. Therefore, both of these objects will be excluded from *AlphaSet.* While comparing X_4 with the objects 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:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked\_Objects|' + 1} = \frac{3-1}{3+1} = 0.5 \quad (25)$$

- **Indiscernible class**

This class explains the indiscernible relation between the objects of the datasets where one object 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 objects $\boldsymbol{a, b}$ and attribute set $\{\mathbf{c_1}, \mathbf{c_2} \in \mathbf{C} | (\mathbf{c_1} \cap \mathbf{c_2}) \neq \boldsymbol{\varphi}\}$ this class can be written as shown in equation 26.

$$((c_1(a) \geq (c_1(b)\ AND\ (c_2(a) \leq (c_2(b)) \quad (26)$$

This shows that object $\boldsymbol{a}$ is dominant over $\boldsymbol{b}$ for criteria $\boldsymbol{c_1}$but at the same time, $\boldsymbol{a}$ is being dominated by $\boldsymbol{b}$ for criteria $\boldsymbol{c_2}$. Therefore, both objects will be kept in *AlphaSet* and the dependency of the dataset will be updated as follows:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} \quad (27)$$

In Table I, when we traversed to object 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:

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{2+1}{4+1} = 0.6 \quad (28)$$

- **Identical class**

This class defines the relationship between two identical objects with an identical decision. Therefore, this class is called "*Identical Class*". This shows the consistency in the dataset. Mathematically, For two objects $\boldsymbol{a, b}$ and attribute set $\mathbf{c} \in \mathbf{C}$ this class can be written as shown in equation 29.

$$(c(a) == c(b) \& D(a) == D(b)) \quad (29)$$

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} \tag{30}$$

In Table I, when X_6 is compared to X_1 it shows that both objects 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:

$$\gamma C(Cl) = \frac{|AlphaSet|'}{|Checked\_Objects|' + 1} = \frac{3}{5 + 1} = 0.5 \tag{31}$$

- **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 objects $\boldsymbol{a}, \boldsymbol{b}$ and attribute set $\mathbf{c} \in \mathbf{C}$ first side of this class can be written as shown in equation 32.

$$(c(a) > c(b)\ AND\ D(a) < D(b)) \tag{32}$$

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

Mathematically, For two objects $\boldsymbol{a}, \boldsymbol{b}$ and attribute set $\mathbf{c} \in \mathbf{C}$ second side of this class can be written as shown in equation 33.

$$(c(a) < c(b)\ AND\ D(a) > D(b)) \tag{33}$$

This states that $\boldsymbol{a}$ is dominated by $\boldsymbol{b}$ 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:

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked\_Objects|' + 1} \tag{34}$$

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

The pseudocode of the proposed methodology is shown in figure 4.

//For Extracting *AlphaSet* to calculate dependency of the dataset. $Checked\_Objects$ and *AlphaSet* are two arrays denoted by M and N respectively.
Inputs: Universe and Dominance Matrix
Output: Dependency of Dataset
Step 1: Find *AlphaSet*

$For\ i = 0\ to\ |U|$
  $Inconsistency\ =\ False$
  $x = 0\ to\ |U|$
  Pick an object from Universe on every iteration let's name it $\{x \in U\}$.
  If (i = = 0)
    $M\ \leftarrow x$
    $N\ \leftarrow x$
  Else
    For j = 0 to $|M|$
      Pick an object from *Checked_Objects* iteratively and let's name it $\{y \in M\}$.
      $If\ (\boldsymbol{a}\ and\ \boldsymbol{b}\ are\ indiscernible)$
        $If(Inconsistancy == Flase)$
          $M\ \leftarrow x$
          $N\ \leftarrow x$
        $End - If$
      $Else - If(C_x > C_y\ \&\&\ D_x \geq D_y)\ ||(C_x < C_y\ \&\&\ D_x \leq D_y)$
        $If(Inconsistancy == Flase)$
          $M\ \leftarrow x$
          $N\ \leftarrow x$
        $End - If$
      $Else - If(C_x == C_y\ \&\&\ D_x == D_y)$
        $If(Inconsistancy == Flase)$
          $M\ \leftarrow x$
          $N\ \leftarrow x$
        $End - If$
      $Else - If(C_x == C_y\ \&\&\ D_x \neq D_y)$
        $M - \{x\}$
        $N\ \leftarrow x$
        $Inconsistency = True$
      $Else - If(C_x > C_y\ \&\&\ D_x < D_y)\ ||\ (C_x < C_y\ \&\&\ D_x > D_y)$
        $M - \{x\}$
        $N\ \leftarrow x$
        $Inconsistency = True$
      $End - If$
    $End - For$

$End - If$
$End - For$
Step 2: Find Dependency

$$Dependency = \frac{|M|}{|U|}$$

Figure 4: Pseudocode of the proposed methodology

5.2 Analytical Proof of Proposed Methodology

To show the proof of the proposed methodology in the mathematical sense an analytical proof is presented in this section. We calculate the dependency of the dataset using the equation 12 which is as follows:

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

Let $AlphaSet\ \alpha = U - \left(\left(\cup_{t\in T} Bn_P(Cl_t^{\geq})\right) \cup \left(\cup_{t\in T} Bn_P(Cl_t^{\leq})\right)\right.$

$$Bn_P(Cl_t^{\geq}) = \overline{P}(Cl_t^{\geq}) - \underline{P}(Cl_t^{\geq}) \rightarrow \left\{\forall x \in U \middle| x \in \overline{P}(Cl_t^{\geq}) \wedge x \notin \underline{P}(Cl_t^{\geq})\right\} \rightarrow \{\forall x \in U | x \in \overline{P'}(Cl_t^{\geq}) \text{ where } \overline{P'}(Cl_t^{\geq}) \subseteq \overline{P}(Cl_t^{\geq}) \text{ and } x \in \overline{P'}(Cl_t^{\geq}) \rightarrow x \notin \underline{P}(Cl_t^{\geq})$$

$$Bn_P(Cl_t^{\leq}) = \overline{P}(Cl_t^{\leq}) - \underline{P}(Cl_t^{\leq}) \rightarrow \left\{\forall x \in U \middle| x \in \overline{P}(Cl_t^{\leq}) \wedge x \notin \underline{P}(Cl_t^{\leq})\right\} \rightarrow \{\forall x \in U | x \in \overline{P'}(Cl_t^{\leq}) \text{ where } \overline{P'}(Cl_t^{\leq}) \subseteq \overline{P}(Cl_t^{\leq}) \text{ and } x \in \overline{P'}(Cl_t^{\leq}) \rightarrow x \notin \underline{P}(Cl_t^{\leq})$$

So:

$$\alpha = U - \left(\left(\cup_{t\in T} Bn_P(Cl_t^{\geq})\right) \cup \left(\cup_{t\in T} Bn_P(Cl_t^{\leq})\right)\right. \rightarrow \alpha = U - \left\{\overline{P'}(Cl_t^{\geq}) \cup \overline{P'}(Cl_t^{\leq})\right\} \rightarrow \alpha = \left\{\underline{P}(Cl_t^{\geq}) \cup \underline{P}(Cl_t^{\leq})\right\}$$

Now:

$$\underline{P}(Cl_t^{\geq}) \rightarrow D_P^{+}(x) \subseteq Cl_t^{\geq} \rightarrow \forall y, x \in U | y \succcurlyeq_P x \wedge y \succcurlyeq_D x \qquad \text{(Implication-1)}$$

Similarly:

$$\underline{P}(Cl_t^{\leq}) \rightarrow D_P^{-}(x) \subseteq Cl_t^{\leq} \rightarrow \forall k, x \in U | k \preccurlyeq_P x \wedge k \preccurlyeq_D x \qquad \text{(Implication-2)}$$

Thus for an object $x \in \alpha$, it should fulfill any of the implications-1 and 2. If an object does not fulfill both of these implications, it will not be part of α.

Now we will explain that how the proposed classes result in objects that fulfill the implications mentioned above.

- Initial Value class specifies an object that is the first object in dataset which means that it fulfills both of the implications and thus must be part of α.
- Now considering the True Positive Class represented by equations 18 and 19. The equation 18, fulfill the implication-1, whereas the equation 22, fulfills the implication-2. Thus all the objects fulfilling these conditions will be part of α.
- Distinct decision class represented by equation 23 specifies the condition where an object fulfills the first part (i.e. y≽_P x or k≼_P) of the implication-1 or 2 but does not fulfill the second part (i.e. y≽_D x or k≼_D x). All such objects should not be part of the α and if adding a new object causes this condition, all the objects having same values of conditional attributes must be removed from α as shown by equation 23.
- Indiscernible class represented by equation 26, also does not result into any inconsistency so it will be part of the α.
- The identical class, represented by equation 29 specifies the objects that can fulfill both implication-1 and 2 at the same time, so all the objects belonging to this class will be part of the α.
- Finally, the False Positive class as shown by equations 32 and 33 specifies the objects that violate both the implications, so all such objects belonging to this class will not be part of the α.

Above description clearly explains that the proposed dependency classes specify only those objects that fulfill the implications-1 and 2. Thus the α produced by proposed dependency classes will contain the same objects as given by the conventional method. This is the reason that proposed classes result in the same classification accuracy as that of the conventional method.

5.3 Case Study

The proposed approach is illustrated using a case study. For this purpose, Table I is used as a dataset. There are seven objects and two conditional attributes in Table I. 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 objects using proposed dominance-based dependency classes. Following is the descriptions of both of these phases:

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

Table III: Dominance Relation Matrix of Table I

	X_1	X_2	X_3	X_4	X_5	X_6	X_7
X_1	=	≤	≥	=	≠	=	≥
X_2	≥	=	≥	≥	≥	≥	≥
X_3	≤	≤	=	≤	≤	≤	≠
X_4	=	≤	≥	=	≠	=	≥
X_5	≠	≤	≥	≠	=	≠	≥
X_6	=	≤	≥	=	≠	=	≥
X_7	≤	≤	≠	≤	≤	≤	=

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 objects to calculate dependency) and *Checked_Objects* (avoids comparison repetitions by storing those objects which have already been compared) were created. Then using a double nested loop, a comparison of objects was carried out. The outer loop was used to select the objects iteratively from the Universe and the inner loop was used to select the objects from *Checked_Objects* array. After that both of these selected objects 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 objects. 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 object and at that time, there were no objects 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						

Objects in *Checked_Objects*

X_1						

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{0+1}{0+1} = 1 \tag{35}$$

Initially, $|AlphaSet|' = 0$ and $|Checked\_Objects|' = 0$ because we were traversing the first object 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					

Objects in *Checked_Objects*

X_1	X_2					

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{1+1}{1+1} = 1 \quad (36)$$

- **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				

Objects in *Checked_Objects*

X_1	X_2	X_3				

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{2+1}{2+1} = 1 \quad (37)$$

- **4th Iteration**

In this iteration, X_4 was selected and compared with all the objects 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 objects 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			

Objects in *Checked_Objects*

X_2	X_3					

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|Checked\_Objects|' + 1} = \frac{3-1}{3+1} = 0.5 \quad (38)$$

X_1 was the only object 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 objects 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		

Objects in *Checked_Objects*

X_2	X_3	X_5				

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{2+1}{4+1} = 0.6 \quad (39)$$

- **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 objects 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	

Objects in *Checked_Objects*

X_2	X_3	X_5				

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' - N}{|U|' + 1} = \frac{3 - 0}{5 + 1} = 0.5 \tag{40}$$

X_4 was the only object 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

Objects in *Checked_Objects*

X_2	X_3	X_5	X_7			

Objects in *AlphaSet*

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

$$\gamma C(Cl) = \frac{|AlphaSet|' + 1}{|Checked\_Objects|' + 1} = \frac{3 + 1}{6 + 1} = 0.57 \tag{41}$$

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 objects 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:

$$\gamma C^{(Cl)} = \frac{|AlphaSet|}{|Universe|} = \frac{4}{7} = 0.57 \tag{42}$$

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 must have calculated.

6 Datasets

To provide a better comparison between proposed and conventional approaches, ten different datasets were selected from the UCI dataset repository [37]. The details of these datasets are provided in Table IV.

Table IIV: Dataset information

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	1385	29	Integer, Real
9	Musk 2	6598	168	Integer
10	ISOLET	7797	617	Real

Our proposed methodology can be successfully implemented using any feature selection algorithm as a replacement of conventional dependency measures. This shows the flexibility and effectiveness of the proposed methodology. To prove our point, Fast Reduct Generating Algorithm (FRGA) was picked for the implementation of the proposed methodology. FRGA is an exhaustive algorithm and it can extract all the possible reducts of the datasets. This extraction of all the possible reducts helped compare both proposed and conventional approaches. The accuracy of the proposed approach was also validated by FRGA.

Both approaches are coded in python for the comparison purpose and these codes are uploaded on the github at https://github.com/RanaKaleemUllah/Incremental-dependency-calculation.

7 Results and analysis

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 in the following section.

7.1 Algorithmic Parameters

The following parameters are used in both proposed and conventional methodology for calculating dependency of dataset:

- The information system $IS = (U, Q, V, F)$ can be represented as follows:
  - U represents a decision system, which is a finite set of objects.
  - Q is a finite set of attributes, including both conditional and decision attributes. It can be represented as $Q = C \cup D$, where C is the set of conditional attributes and D is the set of decision attributes.

- V is the set of value sets for each attribute in Q. For each attribute $q \in Q$, Vq represents the value set of that attribute.
- F is a function $f(x, q)$ that defines dominance by assigning a specific value from Vq to an item x for a particular attribute q.

- Other than conventional parameters, the dominance relation between object is used and on the basis of this dominance relation the proposed methodology uses proposed dominance-based dependency classes as input parameter. These classes are already explained in details in section 5.1.

7.2 Comparison Framework

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

- 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.

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. The experiments to justify the effectiveness and validity of IDDC were conducted on the datasets aforementioned in Table V.

All these components of the devised framework and comparison results of both approaches are discussed below.

- **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 V and VI 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 V: Comparison of Size of AlphaSet

S. NO.	Dataset	*AlphaSet* 's Size (Conventional)	*AlphaSet* '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 VI: 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

- **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.

$$\%\ \text{reduction in execution time} \ = \ 100 - \left(\frac{\mathrm{T1}}{\mathrm{T2}}\right) * 100 \tag{43}$$

In equation 43, $T1$ 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 106 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 44.

$$\%\ \text{reduction in execution time} = \ 100 - \left(\frac{54.39}{106.57}\right) * 100 \cong 50\% \tag{44}$$

It can be concluded from Table VII 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 VII was calculated to get a better picture of time reduction.

$$\text{Avg reduction in execution time} = \frac{459}{10} = 46\%\ approx \tag{45}$$

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 5 provides a pictorial comparison of both approaches.

Table VII: Comparison of Execution time

S. NO.	Dataset	Conventional Time (secs)	Proposed Method 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%

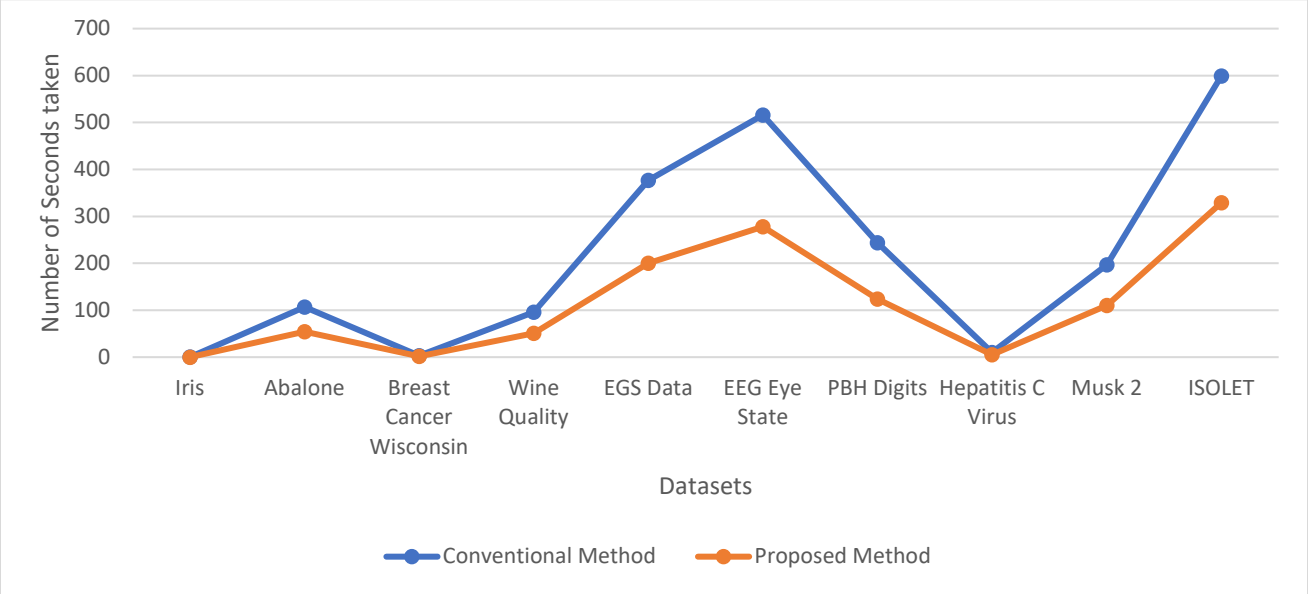

Figure 5: Graphical Comparison of Execution time

- **Memory Usage**

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 objects in class unions. The formula used for the calculation of memory usage of the conventional method is mentioned in equation 46.

$$Memory = Size\ of\ datatype * (total\ number\ of\ objects) + Size\ of\ datatype * \left(\frac{total\ number\ of\ objects}{2}\right)^2 \quad (46)$$

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 objects of Universe. The second array (*AlphaSet)* was used to store those objects which were selected after comparison. Therefore, to calculate the memory usage of the proposed methodology, the formula showed equation 47 was used.

$$Memory = 2 * (Size\ of\ datatype) * (total\ number\ of\ objects) \quad (47)$$

A comparison of memory usage of both of these methodologies is shown in Table VIII. On average, a 98% reduction in required runtime memory was found for ten datasets. It is evident from the comparison that a huge amount of memory was saved by the proposed approach. The formula used for calculating the percentage reduction is given in equation 48. In this formula, $M1$ represents memory used by the proposed methodology and $M2$ represents memory used by conventional methodology.

$$\text{Reduction in required runtime memory} = 100 - \left(\frac{M1}{M2}\right) * 100 \quad (48)$$

Table VIII: Comparison of memory usage

S. NO.	Dataset	Memory Usage by Conventional (MB)	Memory Usage by Proposed Method (MB)	% Reduction in usage
1	Iris	0.0231	0.0012	≈95%
2	Abalone	17.46404	0.33146	≈98%
3	Breast cancer Wisconsin	0.49139	0.00559	≈98%
4	Wine Quality	24.00999	0.03918	≈99%
5	Electrical Grid Stability data	100.04	0.08	≈99%
6	EEG Eye State	104.0808	0.0816	≈99%
7	Pen-Based Handwritten Digit	120.86803	0.08793	≈99%
8	Hepatitis C Virus	1.90992	0.01104	≈99%
9	Musk 2	43.55999	0.06237	≈98%
10	ISOLET	60.82	0.05278	≈99%

- **Computational 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 [38,39]. 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 $4 * (O(Cl * Cl * D_P^+/D_P^-))$. 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 objects, the proposed approach uses a double nested loop. For the outer loop, Big-O was $O(U)$ because we had to compare all the objects 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 6.

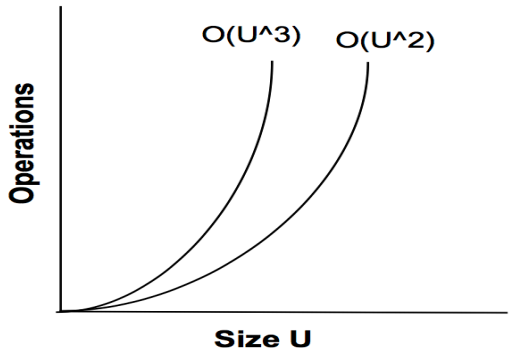

Figure 6: Big-O Complexity Comparison

7.3 Comparison with Heuristic Method

Another comparison of proposed approach is performed with a heuristic method [40] that use an optimized technique to calculate approximations to further calculate the dependency of a dataset. The comparison framework used is same as explained in the section 7.1 of paper. All these components of the devised framework and comparison results of both approaches are discussed below.

- **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 Heuristic Method [36]. 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 Heuristic approach are accurate.

Table IX: Comparison of Size of AlphaSet

S. NO.	Dataset	*AlphaSet* 's Size produced by Heuristic Method [36]	*AlphaSet* 's size by Proposed Method
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 VIII: Comparison of the number of Reducts Generated

S. NO.	Dataset	Reducts by Heuristic Method [36]	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

- **Memory Usage**

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. The formula used for the calculation of memory usage of the heuristic method is mentioned in equation 49.

$$Memory = Size\ of\ datatype * (total\ number\ of\ objects) + Size\ of\ datatype * \left(\frac{total\ number\ of\ objects * (\ total\ number\ of\ objects - 1)}{2}\right) \quad (49)$$

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 objects of Universe. The second array (*AlphaSet)* was used to store those objects which were selected after comparison. Therefore, to calculate the memory usage of the proposed methodology, the formula showed equation 50 was used.

$$Memory = 2 * (Size\ of\ datatype) * (total\ number\ of\ objects) \quad (50)$$

A comparison of memory usage of both of these methodologies is shown in Table XI. On average, a 99% reduction in required runtime memory was found for ten datasets. It is evident from the comparison that a huge amount of memory was saved by the proposed approach. The formula used for calculating the percentage reduction is given in equation 51. In this formula, $M1$ represents memory used by the proposed methodology and $M2$ represents memory used by heuristic methodology.

$$\text{Reduction in required runtime memory} = 100 - \left(\frac{M1}{M2}\right) * 100 \quad (51)$$

Table IXI: Comparison of memory usage

S. NO.	Dataset	Memory Usage by Heuristic Method [36] (MB)	Memory Usage by Proposed Method (MB)	% Reduction in usage
1	Iris	0.0453	0.0012	≈97%
2	Abalone	34.903	0.33146	≈99%
3	Breast cancer Wisconsin	0.9786	0.00559	≈99%
4	Wine Quality	47.990	0.03918	≈99%
5	Electrical Grid Stability data	200.02	0.08	≈99%
6	EEG Eye State	208.10	0.0816	≈99%
7	Pen-Based Handwritten Digit	241.67	0.08793	≈99%
8	Hepatitis C Virus	3.8392	0.01104	≈99%
9	Musk 2	87.0804	0.06237	≈99%
10	ISOLET	121.602	0.05278	≈99%

- **Execution Time**

We used the following formula to calculate the percentage reduction in the execution time of IDDC.

$$\%\ \text{reduction in execution time} = 100 - \left(\frac{\text{T1}}{\text{T2}}\right) * 100 \tag{52}$$

In equation 52, $T1$ represents the time taken by the proposed approach and $T2$ represents the time taken by the heuristic approach. Table XIII shows the time taken by both methods for all the datasets. For instance, the time taken by both proposed and heuristic approaches for the *Abalone* dataset was 52 seconds and 65 seconds, respectively. The percentage reduction in the execution time of IDDC for the *Abalone* dataset is almost 17% which was calculated using the formula from equation 53.

$$\%\ \text{reduction in execution time} = 100 - \left(\frac{54.39}{65.2}\right) * 100 \cong 17\% \tag{53}$$

The percentage decrease in execution time was around 20% for all datasets. The average of Table XIII was calculated to get a better picture of time reduction.

$$\text{Avg reduction in execution time} = \frac{192}{10} = 19\%\ approx \tag{54}$$

The calculated average time reduction by the proposed methodology was 19%. Therefore, it can be concluded that the proposed methodology can save considerable amount of the execution time as compared to the heuristic method.

Table X: Comparison of Execution time

S. NO.	Dataset	Heuristic Method [36] Time (secs)	Proposed Method Time (secs)	% Reduction in Time Taken
1	Iris	0.075	0.06	20%
2	Abalone	65	54	17%
3	Breast cancer Wisconsin	2.4	1.9	21%
4	Wine Quality	62	51	18%
5	Electrical Grid Stability data	244	200	20%
6	EEG Eye State	341	278	19%
7	Pen-Based Handwritten Digit	151	124	18%
8	Hepatitis C Virus	6.7	5.3	21%
9	Musk 2	137	110	20%
10	ISOLET	397	329	18%

7.4 Discussion

The experimental results demonstrate that the proposed approach for calculating dependency in dominance-based datasets is highly efficient in terms of both time and memory usage. This is evidenced by the findings presented in Table VII and Table VIII, which show a decrease in execution time and runtime memory. Moreover, the proposed approach is compared to a heuristic methodology, and the results clearly indicate that the proposed approach outperforms it in terms of both memory utilization and execution time. These findings highlight the effectiveness and efficiency of the proposed approach in handling dominance-based datasets. Furthermore, it can be assessed form the number of comparison done in both conventional and the proposed approach that the proposed approach performs better. 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 $D_P^-(x)$, objects from each union class were compared with all other objects of the Universe. The following were the four union classes extracted from Table I.

$Cl_1^{\leq}(x) = \{X_3, X_4, X_7\}$ for t = 1

$Cl_2^{\leq}(x) = \{X_1, X_3, X_4, X_5, X_6, X_7\}$ for t = 2

$Cl_2^{\geq}(x) = \{X_1, X_2, X_5, X_6\}$ for t = 2

$Cl_3^{\geq}(x) = \{X_2\}$ for t = 3

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

- $\Rightarrow$ In $Cl_1^{\leq}(x)$, the class union had three objects. These three objects were compared with all seven objects 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.
- $\Rightarrow$ In $Cl_2^{\leq}(x)$, the class union had six objects. These six objects were compared with all seven objects 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 objects. These four objects were compared with all seven objects 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 object. This single object was compared with all seven objects 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 objects of the Universe with all the objects 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 55.

$$Number\ of\ CS = number\ of\ Objects * \frac{(number\ of\ objects - 1)}{2} \quad (55)$$

$\Rightarrow$ This difference in the number of comparison statements is shown in Table XIV.

$$\% \text{ reduction in the number } of \text{ CS} = 100 - \left(\frac{\text{CS}_1}{\text{CS}_2}\right) * 100 \quad (56)$$

In the formula mentioned in equation 56, 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 "Results and Analysis" section with the help of multiple datasets.

Table XXI: Number of comparison statements

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

7.5 Limitations

As the results have shown that the proposed approach outperforms the conventional approach but there are certain limitations that are mentioned below:

- The proposed approach is suitable for supervised datasets only.
- Class labels should in numerical form like 0,1,2 etc. If classes are not numerical then they must be converted to numerical form. Class labels must start from 0 but can go up to any number of classes that particular dataset have.
- 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.
- The proposed approach has been tested using different datasets ranging from tens of attributes to hundreds of attributes. The approach performs better when number of attributes are less.

8 Conclusion and Future Work:

In this research work, we have proposed a new incremental approach to calculate the dependency of dominance-based 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 reduction in execution time and memory usage was determined by comparing the proposed approach with conventional methods on a set of 10 diverse datasets. The observed decrease in the execution time and required runtime memory can be attributed to two primary factors. Firstly, the proposed approach avoids approximation and union class calculations, which typically consume substantial time and memory resources. Secondly, the number of comparison statements required to determine dependencies is reduced in the proposed approach, further contributing to the overall efficiency gains. 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.

In terms of future work, one promising direction for the proposed approach is its application on unsupervised datasets. The proposed approach has primarily been evaluated on supervised datasets, where the class labels are known. Extending the approach to unsupervised scenarios would involve exploring how the IDDC can be adapted to handle data without explicit class labels. This would enable the application of the technique in a wider range of data analysis tasks, such as clustering or outlier detection, where class information may not be available.

Another important avenue for future research is to focus on reducing the execution time of the proposed approach. While the incremental nature of the technique offers computational advantages compared to traditional dependency calculating approaches, further improvements can be made by leveraging parallel processing methodologies. Exploiting parallelism can help distribute the computational load across multiple processing units, potentially leading to significant reductions in execution time. This could involve exploring parallel algorithms or utilizing parallel computing architectures, such as GPUs or distributed computing frameworks, to enhance the scalability and efficiency of the incremental dominance-based dependency calculations.

Compliance with Ethical Standards:

Conflict of Interest: The authors declare that they have no conflict of interest.

Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References:

[1]. 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

[2]. 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

[3]. Anaraki, J. R., & Eftekhari, M. (2013, May). Rough set based feature selection: a review. In The 5th Conference on Information and Knowledge Technology (pp. 301-306). IEEE. Doi:10.1109/IKT.2013.6620083

[4]. 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

[5]. 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

[6]. 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

[7]. 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

[8]. Liou, J. J. (2009). A 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

[9]. 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

[10]. 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

[11]. 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

[12]. 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

[13]. 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

[14]. 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

[15]. 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

[16]. Mohamad, M., & Selamat, A. (2018, March). Analysis on 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

[17]. 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

[18]. 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

[19]. 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

[20]. 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.
[21]. 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
[22]. 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
[23]. Nosheen, F., Qamar, U., & Raza, M. S. (2022). Redefining preliminaries of dominance-based rough set approach. Soft Computing, 26(3), 977-1002.
[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]. Bouzayane, S., & Saad, I. (2017, May). Weekly predicting the at-risk MOOC learners using 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
[26]. 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
[27]. Kusunoki, Y., & Inuiguchi, M. (2010). A unified approach to reducts in dominance-based rough set approach. Soft Computing, 14(5), 507-515. Doi: 10.1007/s00500-009-0450-0
[28]. 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
[29]. 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
[30]. Hu, C., Zhang, L. Dynamic dominance-based multigranulation rough sets approaches with evolving ordered data. Int. J. Mach. Learn. & Cyber. 12, 17–38 (2021). https://doi.org/10.1007/s13042-020-01119-1
[31]. Palangetić, M., Cornelis, C., Greco, S., & Słowiński, R. (2021). Fuzzy extensions of the dominance-based rough set approach. International Journal of Approximate Reasoning, 129, 1-19.
[32]. Kusunoki, Y., Błaszczyński, J., Inuiguchi, M., & Słowiński, R. (2021). Empirical risk minimization for dominance-based rough set approaches. Information Sciences, 567, 395-417.
[33]. 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
[34]. 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
[35]. 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
[36]. 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
[37]. "UCI," [Online]. Available: https://archive.ics.uci.edu/ml/index.php.
[38]. Allen Weiss, M. (2004). Data Structures in C++. Chapman & Hall/CRC Computer & Information Science Series, 42–1–42–17. Doi: 10.1201/9781420035179.ch42
[39]. Sahni, S. (2004). Analysis of Algorithms. Chapman & Hall/CRC Computer & Information Science Series, 1–1–1–25. Doi:10.1201/9781420035179.pt1
[40]. Ahmad, A., Qamar, U., & Raza, M. S. (2020). An optimized method to calculate approximations in Dominance based Rough Set Approach. Applied Soft Computing, 97, 106731.

An incremental approach for calculating dominance-based rough set dependency

Ullah, Rana Muhammad Kaleem

2024-01-09

Ullah RMK, Qamar U, Raza MS, Erkoyuncu JA. (2024) An incremental approach for calculating dominance-based rough set dependency. Soft Computing, Volume 28, March 2024, pp.3757-3781 https://doi.org/10.1007/s00500-023-09567-x