An Improved Approach for Finding Rough Set Based Dynamic Reducts

ASMAT IQBAL\textsuperscript{1}, MUHAMMAD SUMMAIR RAZA\textsuperscript{1}, MUHAMMAD IBRAHIM\textsuperscript{2}, ABDULLAH BAZ\textsuperscript{3}, (Senior Member, IEEE), HOSAM ALHAKAMI\textsuperscript{4}, (Member, IEEE), AND MUHAMMAD ANWAAR SAEED\textsuperscript{1}

\textsuperscript{1}Department of Computer Science, Virtual University of Pakistan, Rawalpindi 75300, Pakistan
\textsuperscript{2}Department of Information Technology, The University of Haripur, Haripur 22620, Pakistan
\textsuperscript{3}Department of Computer Engineering, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21421, Saudi Arabia
\textsuperscript{4}Department of Computer Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21421, Saudi Arabia

Corresponding author: Muhammad Ibrahim (ibrahimmayar@uoh.edu.pk)

This work was supported in part by the U.S. Department of Commerce under Grant BS123456.

ABSTRACT This is the era of information and the amount of data has immensely increased since the last few years. This increase has resulted in dilemmas like the curse of dimensionality where we need a large number of resources to process the huge number of features. So far there are many tools available that process such large volumes of data among which Rough Set Theory is a prominent one. It provides the concept of Reducts which represent the set of attributes that provide the maximum of the information. However, the problem with Reducts is that they are not stable and keep on changing with the addition of more and more data. So, the concept of Dynamic Reducts is introduced in the literature to provide a more stable version of Reducts. There are many Dynamic Reduct finding algorithms available; however, these dynamic Reduct finding algorithms are computationally too expensive and inefficient for large datasets. In this research, we have presented an improved dynamic Reduct finding technique based on rough set theory. In this technique, Reducts are selected, optimized, and further generalized through Parallel Feature Sampling (PFS) algorithm. An in-depth investigation is performed using various benchmark datasets to justify the effectiveness of our proposed approach. Results have shown that the proposed approach outperforms the state-of-the-art approaches in terms of both efficiency and effectiveness. Overall, 96% average accuracy is achieved and 46.13% reduction in execution time is observed by the proposed algorithm against the compared contemporary approaches.

INDEX TERMS Dynamic reducts, feature selection, relative reducts, parallel feature sample, rough set theory.

I. INTRODUCTION

We belong to the information age where data accumulation is easy and storing it is reasonably priced. Data generation in every minute is adding new data with huge dimensionality to the existing data. Growth is observed in the number of records as well as features, which demands additional utilization of resources to store, process and evaluate it. Applications in real life comprise several hundred or even thousands of attributes. Therefore, the lack of resources is a big issue to process such a massive volume of data. Consequently, we find relief in the reduction of information [1]. Here feature selection comes to the rescue. It is a process to extract a reduced set of attributes which behaves like the entire dataset and provides the same information. In the past few years, rapid advancement in high dimensional datasets has evolved and generated an emergent requirement to extract the underlying principle knowledge [2]. Extreme dimensionality has emerged as a curse and has attracted worthwhile attention in the data mining research domain [3]. High dimensional data collected for several research purposes has an exceptionally large number of features [4]. In machine learning methods, it is not easy to deal with these attributes. Therefore, researchers find it an interesting challenge. Pre-processing of data is necessary for the effective use of machine learning methods. The machine learning process considers feature selection as an important, indispensable element and most frequently used technique in pre-processing of data [5]. Hence extraction of attribute subsets from large datasets is a non-trivial task. Several exhaustive, as well as heuristic feature selection methods and
techniques, have already been proposed in the literature to extract feature subsets or relevant features based on rough set theory [6] due to its analysis friendly nature. A new statistical approach proposed by Bazan [7] offers generalized dynamic reduct. The whole dataset is divided into multiple subsets by randomly selecting objects. Reducts are extracted from these subsets. Rough set theory is appropriate for the mining of generalized dynamic reducts from dataset having several objects. A heuristic feature reduction algorithm [8] takes the benefit of reduced sub-tables. Attributes are randomly selected from the dataset and further generalized based on certain criteria. This approach favors the mining of data from information set having several condition attributes, where rough set theory is used to extract features [9]. To deal with challenges of high dimensionality, an effective, efficient algorithm to find dynamic reducts with reduced computational complexity is required that promises optimal and stable output without irrelevant and redundant features, with accuracy like original dataset and is appropriate for small as well as large datasets.

Overall, this research work provides following research contributions:

- The research work presents an efficient dynamic reduct generation algorithm that combines the properties of both the heuristic and exhaustive approaches. Heuristics based approaches enhance the performance but cannot ensure the optimal solution as they do not explore the entire solution space. Exhaustive approaches although ensure optimal solution but require large amount of resources. Using the features of heuristic approaches an initial set of dynamic reducts is generated which is then optimized by using the exhaustive search in the final stage. This ensures both the optimal results with enhanced performance.

- The proposed method is implemented using parallel approach where dynamic reducts from different sub-tables are generated in parallel. This significantly enhances the performance and ensures the optimal use of resources.

- In contrast to conventional approach, we have used relative dependency instead of positive region based dependency. The reason behind is that relative dependency does not require the calculation of computationally expensive positive region. This results in further increase in performance of the algorithm as compared to conventional approaches without compromising the accuracy.

II. BACKGROUND

In 1991, it was affirmed that the size of data warehoused in the world doubles every twenty months [10]. However, we find growing expectations and the realization that data will be a valuable resource for some reasonable benefit, if evaluated and presented intelligently [11]. To abridge the remoteness concerning data generation and understanding, there is an exigent necessity for novelty and innovation in computational theories, techniques and tools that help in extraction or selection of expedient knowledge from the colossal volume of data [12] with less computational complexity and within available capacity of resources. Real datasets encompass a large number of features (attributes) and objects (records) and the process of selecting subsets of features from the massive data is called Feature Selection that claims to deliver useful information similar to the entire dataset [13]. Further processing is supported by these carefully chosen features. A feature selection method that extracts fewer attributes but provides maximum information, is considered to be worthy. Redundant and irrelevant features are removed and only subsets comprised of relevant features are selected to ensure the optimality of the solution. So far, different kinds of searching and mining approaches have been proposed. One of the most intuitive strategies to find a candidate subset is to search the entire space [13]. These exhaustive searching methods are suitable for smaller datasets but they are expensive in terms of computational time and space for large datasets. An alternative way is to search in random fashion [14]. Features are selected randomly from a dataset and evaluated for being the required feature subset. This procedure continues to execute for a specified time or until it finds a potential candidate solution. Heuristic-based search is another frequently used technique for feature subset selection [14]. The search is guided through some heuristic function. The maximum value of the function is targeted and search is directed towards it. The process continues to evaluate features using these fitness functions until it gets the solution with requisite value but the accuracy of the solution is not guaranteed. Recently, an alternative tactic to reduce the computation time is introduced i.e., the use of dynamic reducts [15]. Instead of taking into account the whole dataset, multiple sub-tables are constructed by randomly selecting objects and features are evaluated based on these objects to extract sets of reducts. These reducts are further generalized based on the stability coefficient compared with the stability threshold and are named as dynamic reducts. Dynamic reducts are considered to be stable reducts. Although various tools exist for feature selection, however, Rough Set Theory (RST) [6] has shown remarkable reliability. It is a mathematical tool to analyze data through evaluation measures which help in feature reduction. Therefore, it is better to process fewer attributes instead of the whole set of variables. Several domains are taking benefit of this approach like rule optimization [16], forecasting [17], classification [18], dependency analysis [19] and stability assessment [20]. Literature is full of these heuristic-based search algorithms for the same purpose. However, there are some common glitches with the existing rough set-based feature selection algorithms [21]. One of these problems is the accuracy of the optimal solution.

Genetic Algorithm (GA) and Rough Set Attribute Reduction (RSAR), the designed Fuzzy Rough Set [22], [23] do not guarantee the optimal reduction.

Similarly, Particle Swarm Optimization (PSO) [24], Discrete Particle Swarm Optimization (DPSO) [25]–[28], Improvised PSO algorithm [29], [30], Fish Swarm [31] involve
several computational complexity in finding the global minimal reduct whereas Quick reduct [32] algorithm is exhaustive and time consuming method. Similarly, there are other similar approaches [33]–[35]. Therefore, the fundamental impression of this research is the question that “Can there be a dynamic reduct finding algorithm using the rough set theory that is efficient and effective both in terms of computational time and accuracy?” Hence, this research opts to propose an efficient method to find Rough Set based dynamic reducts. The main objective is to present a feature selection technique that could be computationally efficient and effective for large datasets while still maintain accuracy. Section III contemplates the rough set theory related to preliminaries. The concept of dynamic reducts is outlined in section IV. In section V, a comparison of the state of the art approaches is elaborated whereas, Section VI critically reviews the existing dynamic methods. The discussion concerning the proposed approach is delineated in Section VII. Section VIII discusses experiments and their results followed by comparative analysis. Finally, Section IX summarizes the research work through findings and future work.

III. ROUGH SET THEORY PRELIMINARIES

Pawlak proposed Rough Set Theory (RST) [6], [36]. Various domains use this theory for analysis of data. Some preliminaries of RST are discussed here.

### TABLE 1. An example of decision system.

| Object | Attributes | Conditional Attributes | Decision Class |
|--------|------------|------------------------|----------------|
| x1     | Academic Qualification | Master | Lecturer |
| x2     | Academic Qualification | Master | Lecturer |
| x3     | MPhil | 5 | Assistant Professor |
| x4     | PhD | 10 | Professor |
| x5     | MPhil | 5 | Assistant Professor |
| x6     | PhD | 10 | Professor |
| x7     | MPhil | 4 | Assistant Professor |

The decision system is any information system having a decision property along with the conditional attributes. Decision attribute values depend on conditional attributes. Any information system of the form \( S = (U, C UD) \) is a decision system, where ‘U’ symbolizes the non-empty finite set of objects, ‘C’ denotes set of conditional attributes and ‘D’ character is used for decision class. An example of a decision system is shown in table 1, where \( U = \{ x1, x2, x3, x4, x5, x6, x7 \} \), \( C = \{ \text{Academic Qualification, Period of Service} \} \) and \( D = \{ \text{Designation} \} \).

### A. INDISCERNIBILITY RELATION

Indiscernibility relation is the basic idea in rough set theory. It is a relation between two or more similar objects, where all the values are indistinguishable relative to a subset of the considered attributes. An equivalence relation is formed based on this indiscernibility relation, where all the identical objects of the set are considered as elementary [38]. U is considered a non-empty finite set object and is partitioned into a family of equivalence classes of (C), denoted by \( U / (C) \) or \( U/C \). An equivalence relation is a binary relation \( R \subseteq X \times X \) which is reflexive (relation of an object with itself i.e. \( xRx \)), symmetric (i.e. if \( xRy \) then \( yRx \) ) as well as transitive (i.e. if \( xRy \) and \( yRz \) then \( xRz \)). An equivalence class of an element consists of all objects such that \( xRy \) [39]. For the above given decision system \( S = (U, C UD) \) in Table1, a discernibility relation is represented as \( \text{INDs}(C) \):

\[
\text{INDs}(C) = \{(i, xj) \in U^2 | \forall c \in C(x(i) = c(xj))\}
\]

In the above equation, \( \text{INDs}(C) \) is named as “C-indiscernibility” relation and can also be denoted by \( [x]c \). We can omit subscript S if it is clear which information system is under consideration. Equation 1 specifies that objects \( xi \) and \( xj \) are indiscernible concerning attribute set \( c \in C \). In table 1, it is clear that objects \( x1, x2 \) are indiscernible by attribute from set \( C = \{ \text{Academic Qualification, Period of Service} \} \).

### B. RELATIVE DEPENDENCY

Relative dependency [40] is an important concept in Rough Set Theory to measure dependency. This way, the efficiency of algorithms can be improved in order to find the optimal reducts from the set of condition attributes. Using this approach, distinct instances of the subset of the dataset are counted to calculate the degree of relative dependency. We do not need to generate discernibility functions or find a positive region using a complex methodology. Hence, the computational efficiency of the algorithms for finding minimum possible reducts can be improved prominently.

Normally, most of the existing rough set-based attribute selection algorithms are based on positive region or discernibility function calculations. This family suffers from intensive computational complexity and inefficiency in terms of execution time and memory usage. However, the relative dependency technique avoids the complex and expensive calculations and can be defined as follows for a subset of attributes \( R \):

\[
\kappa_R(D) = \frac{|U / \text{IND}(R)|}{|U / \text{IND}(R \cup D)|}
\]

In the above equation, \( R \) is a reduct such that \( \kappa R \) (D) and \( V X \subseteq R, \kappa x (D) = \kappa C \) (D).

The relative dependency-based algorithm utilizes the technique of backward elimination to find optimal reducts. Each attribute is tested for its exclusion possibility. If the removal of an attribute does not affect the dependency of decision attribute on the rest of the attributes and dependency remains equal to “1”, then only, in that case, this attribute is removed. Every attribute is considered one at a time. We start from the first attribute, evaluate its relative dependency; if dependency does not change then this attribute is removed. This attribute removal process goes on until no more attribute can be eliminated without disturbing the dependency. The primary focus of the relative dependency approach is to provide an
easy dependency calculation technique. Therefore, the main advantage of this approach is that it sidesteps a positive region or discernibility functions, which is a computationally expensive task and makes RST centered dependency measure unsuitable for large datasets. The pseudo-code for the Relative dependency algorithm is given in figure 1.

**C. REDUCTS**

Dimensionality can be reduced by considering only those conditional attributes which can preserve the indiscernibility relation. A set of attributes is selected if it constructs the set of equivalence classes similar to equivalence classes constructed by the complete set of attributes. All the remaining conditional attributes are considered redundant and do not affect the classification accuracy, hence these can be reduced. These attribute subsets are called Reducts.

**IV. DYNAMIC REDUCTS**

Reducts are generated from an information system. Whenever there is a change in the system, reducts may get affected. Therefore, we can say that reducts are sensitive to changes in the system. This can be observed by removing some indiscriminately selected object set from the original set of objects. Here comes the need of reducts that are stable. Those reducts that are common or are frequently occurring in randomly chosen sub-tables are considered to be stable reducts. Dynamic reducts comprehend these types of reducts [15].

Let us consider a decision table $S = (U, C \cup d)$, now any system $T = (U', C \cup d)$ ($U' \subseteq U$) is a sub-table of $S$. All the sub-tables of $S$ are known as family $F$ and represented by equation 3.

$$DR(S, F) = Red(S, d) \cap \{ \bigcap_{T \in F} Red(T, d) \}$$

Here, Equation 3 defines the F-dynamic reducts set of $S$ i.e. $DR(S, F)$. It also delineates that if all sub-tables in family $F$ have some common reducts denoted by $Red(T,d)$, then these relative reducts of $S$ represented by $Red(S,d)$ will be considered as dynamic. This definition is restrictive in most of the cases when all the sub-tables do not have common reducts. In this situation, there is a need for some general view of dynamic reducts.

To generalize the concept of dynamic reducts, we introduce a new parameter called threshold. Its value lies between 0 and 1 i.e. $0 \leq \varepsilon \leq 1$. Now we can redefine the concept of $(F, \varepsilon)$ dynamic reducts as:

$$DR_{\varepsilon}(S, F) = \{ C \in Red(S, d) : S_F(C) \geq \varepsilon \}$$

where

$$S_F(C) = \frac{|\{T \in F : C \in RED(T, d)\}|}{|F|}$$

Here, $S_F(C)$ is the stability coefficient of $C$. It is used to generalize the concept of strict restriction that a dynamic reduct must be part of every generated reduct of sub-tables. It delineates that a reduct is dynamic on the off chance that it appears in a specific proportion, determined by $\varepsilon$, of sub-tables. For example, 0.5 value means that a reduct is dynamic if it is part of not less than half of the sub-tables. It is also noteworthy that if $F = \{ S \}$ then $DR(S,f) = Red(S,d)$.

Algorithm for the calculation of dynamic reducts is given in Figure 2: In this algorithm, the first step is to input a given decision table $S$. All the reducts of this decision table are calculated. The second step deletes one or more rows from $S$ and generates new subsystems $S_j$. Reducts for each subsystem are calculated in the third step. The last step computes dynamic reducts based on a significant factor $s_F(C, R)$ for $C$, which is found in all reducts $R$.
chosen sample of a given decision table. These stable sets of features are called dynamic reducts. Hence, the portion of this section briefly compares the state of the art dynamic reduct approaches based on rough set theory.

Generalized dynamic reduct [15] is a statistical approach, which takes subsets of data instead of the whole decision table and extracts reducts from multiple subsets. These reducts are further generalized based on a stability coefficient value. Hence, dynamic reducts prove to be more stable than other reducts calculated by non-dynamic algorithms. Being a relatively new concept, dynamic reducts are scarcely found in the literature; however, we tried to comprehend this idea completely through inadequate literature. Efforts were focused on finding the most relevant papers and comparison is based on the best work provided, up to the best of our knowledge.

Sengupta and Das [41] emphasized on knowledge discovery from incremental data and presented an algorithm to generate dynamic reduct using rough set theory. Discrete Particle Swarm Optimization (DPSO) Algorithm took advantage of the discernibility matrix and frequency value of features to divide these attributes into two categories i.e. core set and non-core set. A hybridized algorithm proposed by Kudo and Murai [42] is based on an attempt of generalization of dynamic reducts and heuristic attribute reduction using reduced decision tables to achieve attribute reduction. Wang et al. [43] view reducts as the core of RST; whereas, dynamic reduct from sub-tables is an effective reduct extraction method. The authors state that sub-table size is an important and effective factor in this regard. Therefore, the study highlights the problem of sub-table sampling and presented a new method of defining the size of the sub-table family. In addition, dynamic reduct sub-table quality-related parameters are also specified. Kudo and Murai [8], [44] proposed a hybrid algorithm of generalized dynamic reducts and heuristic attribute reduction using reduced decision tables to achieve attribute reduction. Switching between two types of reduction methods (heuristic/exhaustive) is based on the size of the reduced decision tables. Preliminary core concepts were redefined in [46] for feature selection.

VI. CRITICAL REVIEW AND LIMITATIONS OF EXISTING DYNAMIC REDUCT ALGORITHMS
Various RST based algorithms for feature selection have been proposed. Here, we will critically analyze the strengths and weaknesses of some of the existing dynamic reduct algorithms. In RSTR algorithm [45], RST based reducts are calculated using computationally expensive positive region-based dependency calculation technique and utilize more memory. Sometimes algorithm is unable to find reducts for which dependency of decision attribute is equal to the dependency of decision attribute on the entire set of attributes. This type of exhaustive search consumes more time and is computationally expensive for datasets with huge size. Wang et al. [43] analyzed the deficiencies of traditional GDR algorithms. Based on this assessment, the authors proposed a Fast GDR algorithm. They claim that the algorithm needs to calculate only a portion of sub-tables from the family F to extract a set of generalized dynamic reducts that meets the requirements of the stability threshold. Fast Algorithm takes complete decision table as an input along with stability threshold value and family of sub-tables with a certain size. It constructs two families F1 and F2 sub-tables through the random selection of objects. In the next step, it selects sub-tables from family F and considers these sub-tables as Family F1. The remaining sub-tables belong to family F2. Now, compute the reducts from family F1, set them as the candidate reduct set and calculate the frequency of occurrence of each reduct. Consider Family F1 as Family F3 and the number of reducts for family F1 is equal to the number of reducts for family F3. Now select a sub-table B from family F2, such that \( F_3 = F_1 \cup B \). If the reduct extracted from sub-table B is any reduct that is part of candidate reduct set from family F1, then add it to the candidate reduct set of family F3 else, consider another sub-table family F2, extract reducts and compare it with the candidate reduct set of family F1. In the next step, calculate the stability coefficient of each reduct. If the stability coefficient is larger than the stability threshold, then the reduct is generalized dynamic reduct. A technique of sub-table reduction and dynamic reduct computation was proposed by Bazan et al. [15]. The algorithm is based on reduct traces calculation technique. In the first step, this algorithm calculates reducts from a given information system. The second step is to randomly delete some of the rows from the decision table and extract a reduced decision table. The third step calculates reducts for the reduced sub-table. Next step is to analyze the size of the decision table if the size of the received table is more than 40% - N of objects (here, N is the number of objects in the given information system), then the second step is repeated and more reducts are calculated for further reduced decision tables. In the next step, for each reduct generated in step 1, take the number of reducts extracted in step 3 comprising of this reduct. This number of reducts in step 3 is ultimately the trace of a given reduct. The algorithm considers those reducts having maximal traces that are ultimately the stable reducts and are designated as dynamic reducts. However, this technique suffers from high execution time and repeatedly calculates reducts. Optimality of such reducts is not guaranteed and parameters and variables take more memory space and utilize more resources.

VII. PROPOSED SOLUTION
By analyzing the traditional generalized dynamic reduct algorithms’ deficiencies, we have proposed an improved feature selection algorithm for finding rough set based dynamic reducts. The algorithm uses Parallel Feature Samples (PFS) and GDR. A feature sample is comprised of attributes that are selected randomly from a set of conditional attributes. Attributes can be selected through any mechanism. However, we have selected the hit and trial method. The original dataset is divided into multiple sub-tables with numerous attributes and objects.
TABLE 2. Summary of dynamic reduct approaches.

| Algorithm | Technique used                                      | Advantages                                                                 | Disadvantages                                                                 |
|-----------|-----------------------------------------------------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------------|
| Discrete  | Particle Swarm Optimization [41]                    | Irrelevant attributes are removed through backward elimination. Algorithm   | Algorithm suffers from computation complexity as it is based on the             |
| Hybrid    | method of GDR and reduced decision tables [42]     | The Proposed approach extracts GDRs from the dataset if the size of the      | conventional dependency calculation technique and not suitable for large        |
| Fast      | Algorithm [43]                                      | The algorithm avoids exhaustive search.                                      | bottleneck.                                                                   |
| Exhaustive| / Heuristic attribute reduction algorithm [8]       | The proposed algorithm is useful for parallel attribute reduction           | Hybridization process does not promise the removal of redundant features in    |
| Rough      | set theory based reducts [45]                       | Algorithm proves to be better for rule extraction and classification of      | the case, heuristic attribute reduction does not guarantee the generation of   |
| Reduct     | Traces [15]                                         | Stable reducts are obtained based on reduct traces.                         | relative reducts.                                                             |

PFSs are generated from these sub-tables until we find the best fit sample having a maximum dependency. The best fit sample must have dependency equal to the entire set of features. Optimization is the next step for the removal of redundant features. Consequently, we have multiple relative reducts. These relative reducts are evaluated based on a significant factor i.e. stability coefficient and finally, dynamic reducts are extracted compared to a threshold value and are considered to be stable. This dynamic reduct algorithm is better than non-dynamic approaches as it does not have complex operators or mechanisms like PSO, GA, etc.

A. ALGORITHM DESCRIPTION

In this research work, efforts were made to extract optimal and stable dynamic reducts using rough set theory through an improved yet simple feature selection technique. Here, we present some theoretical basis for our proposed algorithm. The algorithm utilizes the PFS reduction technique and generates GDRs using a decision table with a large number of attributes and numerous objects. Essentially, the algorithm has three main phases.

1) Construction phase: Construct parallel feature samples from the sub-table of the given information system. Attributes are selected randomly using the rough set theory dependency measure until we find a PFS with maximum dependency.

2) Optimization phase: Optimize the best fit PFS from the first phase by eliminating redundant and irrelevant features using the relative dependency technique to generate as many relative reducts as possible.

3) Evaluation phase: Evaluate each relative reduct for its competency to be designated as dynamic reduct by evaluating against the condition of GDR i.e. comparing the stability coefficient of reduct against an admissible threshold value specified by the user.

Here, we provide a detailed description and try to cover what each of these phases entails.

1) PHASE1

In the first step of the extraction phase, different families of sub-tables are developed with numerous objects and a set of condition attributes. This is done manually using a static dataset and ‘n’ numbers of sub-tables are settled. The second step of this phase is to further reduce these tables by constructing parallel feature samples through a random selection of condition attributes from the available search space. Features are selected without replacement within a sub-table and with replacement among the family of sub-tables. Essentially, it means that when a feature is selected from a sub-table, it should be selected only once. On the other hand, when parallel features are sampled, then a feature could be selected in multiple samples. The selection of features is based on 50% probability i.e., a feature is selected or not selected. Intentionally each attribute is offered an equal opportunity for selection. The size of PFS is taken equivalent to the total number of attributes in the given information system. Features present in PFS are represented by their index number and features not present in PFS are denoted by ‘0’. For example, a feature sample “1,0,0,4,0,6” demonstrates that features with index numbers 1, 4 and 6 are part of the current feature vector, whereas feature numbers 2, 3 and 5 are not present in the current selection of PFS. Hence, the construction of PFS further reduces the sub-tables. After the development
of a parallel feature sample, the algorithm computes the relative dependency based on the selected attributes. If the dependency value for the chosen set of attributes is equal to the complete set of condition attributes, then this PFS is the candidate for feature subset extraction. Otherwise, repeat the same mechanism of PFS construction and its evaluation.

The algorithm will repeat the construction of PFS and its evaluation until a right feature sample with maximum dependency equal to the entire set of condition attribute is obtained.

2) PHASE 2
Candidate PFS with maximum dependency is fed to the next phase as it may consist of many redundant features and only a few might qualify for the resultant optimal reduct set. Unnecessary features are removed from PFS in optimization phase. If the removal of a feature does not upset the dependency of the remaining feature sample, then the feature is irrelevant and dispensible. All the attributes part of the PFS are scanned turn by turn and evaluated for further reduction of feature vector and extraction of optimal reduct set at the same time. Optimization process continues until optimal feature subset generation is ensured at the output.

3) PHASE 3
Evaluation phase analyzes each relative reduct extracted from phase 2 for its fitness to be nominated as dynamic reduct. These are analyzed against condition of GDR i.e. stability coefficient of each reduct is compared against a stability threshold value specified by the user and if $\gamma_S \geq \varepsilon$, then the reduct is dynamic. Figure 3 presents the proposed algorithm.

B. WORKING EXAMPLE
We now explain the proposed methodology using a simple example. A sample dataset for the working example is divided into three sub-tables as provided in the Table 3, 4 and 5:

**TABLE 3. A Sample dataset (Sub-table 1).**

| $x \in U$ | a | b | c | d | e | Y |
|-----------|---|---|---|---|---|---|
| 1         | 2 | 1 | 2 | 1 | 2 | 2 |
| 2         | 0 | 0 | 3 | 0 | 3 | 3 |
| 3         | 1 | 1 | 2 | 1 | 3 | 2 |
| 4         | 0 | 0 | 1 | 0 | 2 | 1 |
| 5         | 2 | 1 | 3 | 0 | 2 | 3 |
| 6         | 2 | 0 | 3 | 1 | 3 | 2 |

**TABLE 4. A Sample dataset (Sub-table 2).**

| $x \in U$ | a | b | c | d | e | Y |
|-----------|---|---|---|---|---|---|
| 1         | 0 | 1 | 3 | 0 | 3 | 3 |
| 2         | 1 | 2 | 0 | 2 | 1 | 2 |
| 3         | 2 | 0 | 3 | 1 | 2 | 2 |
| 4         | 0 | 2 | 1 | 3 | 2 | 2 |
| 5         | 1 | 1 | 2 | 0 | 3 | 1 |
| 6         | 2 | 0 | 3 | 1 | 2 | 2 |

The sample decision system consists of 18 objects and 6 conditional attributes $C = a, b, c, d, e$ including 01 decision attribute $D = Y$. Here, ‘a’ is first attribute; ‘b’ is second and so on. Decision attribute is comprised of 03 decision classes i.e., 1, 2, 3.

Given decision table is divided into three equal size sub-tables i.e. Table 3, 4 and 5. Dependency of the given dataset is: 

$\gamma_C(D) = 1.0$

Here $\gamma$ is used to represent relative dependency instead of $\kappa$ as mentioned in equation 2.

1) THREAD1 (SUB-TABLE1)
Thread 1 performs calculations based on Table 3.

Construction Phase
Suppose the algorithm first construct a feature sample for the sub-table given in Table 3: $V = a, b, 0, d, 0$

$$\gamma(V)(D) = \frac{|U| \text{IND}(a, b, d)}{|U| \text{IND}(a, b, d, Y)}$$

$$= \frac{|\{1\}, \{2, 4\}, \{3\}, \{5\}, \{6\}|}{|\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}|}$$

$\gamma(V)(D) = 0.7$
Here $\gamma(V)$ is not equal to $\gamma(C)$, therefore, this feature sample does not contain feature subset and is not considered for further optimization. Now we construct another feature sample. Suppose, following feature sample is constructed this time: $V = \{a, 0, c, d, 0\}$

$$
\gamma(V)(D) = \frac{|U \times IND(a, c, d)|}{|U \times IND(a, c, Y)|}
= \frac{|\{(1), (2), (3), (4), (5), (6)\}|}{|\{(1), (2), (3), (4), (5), (6)\}|}
$$

(7)

$\gamma(V)(D) = 1.0$

Here, the dependency of this feature sample is equal to $\gamma(C)$. Therefore, this feature sample contains a feature subset. With this, the first step completes and the selected feature sample is: $a, c, d$.

Optimization Phase

In this phase, the feature sample with relative dependency equal to ‘1’ is selected. Now we will optimize this feature sample by removing irrelevant features. First, we eliminate attribute d:

$$
\gamma(v-(d))(D) = \frac{|U \times IND(a, c)|}{|U \times IND(a, c, Y)|}
= \frac{|\{(1), (2), (3), (4), (5), (6)\}|}{|\{(1), (2), (3), (4), (5), (6)\}|}
$$

(8)

$\gamma(v-(d))(D) = 0.833$

$\gamma(V) - \gamma(v-(d))(D) = 1.0 - 0.833 = 0.167$

Removing feature ‘d’ reduces dependency, therefore, feature ‘d’ is indispensable and we cannot eliminate this feature. Next, we consider the removal of feature ‘c’.

$$
\gamma(v-(c))(D) = \frac{|U \times IND(a, d)|}{|U \times IND(a, d, Y)|}
= \frac{|\{(1), (2), (3), (4), (5)\}|}{|\{(1), (2), (3), (4), (5)\}|}
$$

(9)

$\gamma(v-(c))(D) = 0.8$

$\gamma(V) - \gamma(v-(c))(D) = 1.0 - 0.8 = 0.2$

Once again, feature ‘c’ is indispensable too and cannot be eliminated. Next, we consider the removal of feature ‘a’.

$$
\gamma(v-(a))(D) = \frac{|U \times IND(c, d)|}{|U \times IND(c, d, Y)|}
= \frac{|\{(1), (2), (3), (4)\}|}{|\{(1), (2), (3), (4)\}|}
$$

(10)

$\gamma(v-(a))(D) = 1.0$

$\gamma(V) - \gamma(v-(a))(D) = 1.0 - 1.0 = 0.0$

This means that the removal of feature ‘a’ from the feature sample does not affect dependency. So, it is irrelevant and can be removed. Hence, the final feature subset is comprised of the attributes: $V = c, d$.

2) THREAD2 (SUB-TABLE2)

Thread 2 performs calculation based on Table 4.

Construction Phase

Suppose the algorithm first constructs a feature sample for the sub-table given in Table 4: $V = a, b, 0, 0, e$

$$
\gamma(v)(D) = \frac{|U \times IND(a, b, e)|}{|U \times IND(a, b, Y)|}
= \frac{|\{(1), (2), (3), (4), (5)\}|}{|\{(1), (2), (3), (4), (5)\}|}
$$

(11)

$\gamma(v)(D) = 1.0$

Here, the dependency of this feature sample is equal to $\gamma(C)$. Therefore, this feature sample contains a feature subset. With this, the first step completes and the selected feature sample is: $a, b, e$.

Optimization Phase

In this phase, the feature sample with relative dependency equal to ‘1’ is selected. After then, we will optimize this feature sample by removing irrelevant features. First, we eliminate attribute d:

$$
\gamma(v-(d))(D) = \frac{|U \times IND(a, b)|}{|U \times IND(a, b, Y)|}
= \frac{|\{(1), (2), (3), (4)\}|}{|\{(1), (2), (3), (4)\}|}
$$

(12)

$\gamma(v-(d))(D) = 1.0$

$\gamma(V) - \gamma(v-(d))(D) = 1.0 - 1.0 = 0.0$

Removing feature ‘d’ does not affect dependency, therefore, feature ‘e’ is irrelevant and we can eliminate this feature. Next, we consider the removal of feature ‘b’.

$$
\gamma(v-(b))(D) = \frac{|U \times IND(a)|}{|U \times IND(a, Y)|}
= \frac{|\{(1), (2), (3), (4)\}|}{|\{(1), (2), (3), (4)\}|}
$$

(13)

$\gamma(v-(b))(D) = 0.75$

$\gamma(V) - \gamma(v-(b))(D) = 1.0 - 0.75 = 0.25$

Once again, feature b is essential and cannot be eliminated. Next, we consider the removal of feature ‘a’.

$$
\gamma(v-(a))(D) = \frac{|U \times IND(b)|}{|U \times IND(b, Y)|}
= \frac{|\{(1), (2), (3), (4)\}|}{|\{(1), (2), (3), (4)\}|}
$$

(14)

$\gamma(v-(a))(D) = 0.6$

$\gamma(V) - \gamma(v-(a))(D) = 1.0 - 0.6 = 0.4$

This means that the removal of feature ‘a’ from the feature sample reduces dependency. So, it is not irrelevant and cannot be removed. Consequently, the final feature subset is comprised of the attributes: $V = a, b$.

3) THREAD3 (SUB-TABLE3)

Thread 3 performs calculation based on Table 5.
Construction Phase
Suppose, following feature sample is constructed for the sub-table given in Table 5: \( V = a, 0, c, 0, e \)

\[
\gamma (V)(D) = \frac{|U|IND(a, c, e)}{|U|IND(a, c, e, Y)} = \frac{|11, 2, 3, 4, 5, 6|}{|11, 2, 3, 4, 5, 6|} \quad \text{(15)}
\]

\[
\gamma (V) (D) = 1.0
\]

Here, the dependency of this feature sample is equal to \( \gamma (C) \). Therefore, this feature sample contains a feature subset. With this, the first step completes and the selected feature sample is \( a, c, e \).

Optimization Phase
In this phase, the feature sample with relative dependency equal to ‘1’ is selected. Now, we will optimize this feature sample by removing irrelevant features. First, we eliminate attribute \( e \):

\[
\gamma (V\backslash\{e\})(D) = \frac{|U|IND(a, c)}{|U|IND(a, c, Y)} = \frac{|11, 2, 3, 4, 5, 6|}{|11, 2, 3, 4, 5, 6|} \quad \text{(16)}
\]

\[
\gamma (V\backslash\{e\}) (D) = 1.0
\]

Removing feature \( e \) does not disturb dependency, therefore, feature ‘e’ is irrelevant and we can eliminate this feature. Then, feature ‘e’ is removed from the feature set.

\[
\gamma (V\backslash\{c\})(D) = \frac{|U|IND(a)}{|U|IND(a, Y)} = \frac{|11, 2, 3, 4, 5, 6|}{|11, 2, 3, 4, 5, 6|} \quad \text{(17)}
\]

\[
\gamma (V\backslash\{c\}) (D) = 0.5
\]

Once again, the removal of feature ‘c’ reduces dependency. Therefore, the feature ‘c’ is imperative and cannot be eliminated. Next, we consider the removal of feature ‘a’.

\[
\gamma (V\backslash\{a\})(D) = \frac{|U|IND(e)}{|U|IND(e, Y)} = \frac{|11, 2, 3, 4, 5, 6|}{|11, 2, 3, 4, 5, 6|} \quad \text{(18)}
\]

\[
\gamma (V\backslash\{a\}) (D) = 0.5
\]

This means that the removal of feature ‘a’ from the feature sample reduces dependency. So, it is quintessential and cannot be removed. Hence, the final feature subset is comprised of the attributes: \( V = a, c \).

Evaluation Phase
Now, the algorithm parses each relative reduct extracted from all the sub-tables. These reducts are examined for their fitness to be selected as dynamic reduct. The stability coefficient of each reduct is compared against a stability threshold value specified by the user and if \( S_F \geq \varepsilon \), then the reduct is dynamic. Here, the stability threshold value is 0.5. It means a reduct has to appear in half of the sub-tables to be selected as dynamic reduct.

\[
S_F \geq (a) = 0.667
\]

\[
S_F \geq (b) = 0.333
\]

\[
S_F \geq (c) = 0.667
\]

\[
S_F \geq (d) = 0.333
\]

From the above given example:

\[
S_F \geq (a) > \varepsilon \\
S_F \geq (b) < \varepsilon \\
S_F \geq (c) > \varepsilon \\
S_F \geq (d) < \varepsilon
\]

Relative reduct from sub-table 1 is c,d, whereas, extracted reduct from sub-table 2 is a,b and relative reduct generated from sub-table 3 is a,c. Now calculating the stability coefficient of each reduct:

Here, we compare the stability coefficient of reducts against the stability threshold.

Reduce ‘a’ and ‘c’ have stability coefficient values greater than the stability threshold. Therefore, these are the Generalized Dynamic reducts.

VIII. RESULTS AND DISCUSSION

This section provides a comprehensive description of the experimentations and the obtained results. Detailed experiments were performed to justify the effectiveness and efficiency of the proposed algorithm. For this purpose, various publicly available datasets from UCI were utilized [47]. Outputs are organized in tables where the proposed algorithm is compared with three other dynamic reducts RST based algorithms for optimality of reducts, accuracy, and reduction in execution time.

A. EXPERIMENTAL CONTEXT

In this research, efforts were made to propose an algorithm for finding rough set based dynamic reducts. For this purpose, an in-depth analysis of existing algorithms was performed to find out their strengths and weaknesses. Based on this analysis, a novel dynamic reduct finding approach is proposed to overcome the deficiencies in existing methods and optimize the solution. Rough set theory was used as a reliable tool to deal with vague and imprecise features of a dataset. Data dependencies were discovered and the number of features was reduced to generate stable reducts. To justify the proposed solution, various benchmark datasets [47] from UCI were used to prove the efficiency and effectiveness of the proposed solution. The simulation experiments were executed on a workstation equipped with Intel Core i5-8500 Quad-core processor (3.0 GHz clock speed) and 8 GBs of main memory. All the dynamic reducts algorithms were implemented in a multi-threading environment to produce results. Each thread generates reducts and the parent process analyses these reducts to produce stable reducts. Reducts of these algorithms along with the proposed solution were compared against RSTR [45], FAST [43], and RTraces [15] approaches, using the comparison framework.
As the purpose of this research work is to find a feature selection technique to reduce the curse of dimensionality; therefore, large datasets from UCI machine learning repository were selected purposely (see Table6). These datasets are publicly available. Therefore, the proposed methodology along with other algorithms was tested by these datasets. These datasets (tables) comprising of features and objects were divided into a family of multiple equal size sub-tables. Each table has a set of conditional attributes and a decision attribute having two or more classes. Parallel processing of these sub-tables was ensured utilizing the multi-threading feature of VBA. Each thread performed calculations on a single sub-table separately; however, these threads are executed simultaneously. Each thread produces a subset of features based on rough set theory. These reducts are fed back to the main process that evaluates these results based on a threshold value specified by the user, to generate dynamic reducts. These dynamic reducts are considered to be stable reducts. In this way, we can get only the optimal subsets of features. PFS approach has proved to be more operative and proficient as compared to the other algorithms. It produces the minimal optimal and stable dynamic reducts with maximum accuracy in reduced execution time (see the results) for the large datasets and meets the objectives of this research work. The obtained results advocates the effectiveness of the proposed approach as compared to the state-of-the-art compared approaches.

B. COMPARISON PARAMETERS

Comparison parameters were devised and taken into consideration to compare the proposed algorithm with three dynamic RST based approaches. The results are compared concerning accuracy, optimality and percentage decrease in time against the compared approaches. The discussion about each of the obtained results is contemplated below:

1) ACCURACY

Accuracy of results is measured by comparing the closeness of the reducts of algorithm-A with algorithm-B. It is considered in percentage for simplicity by analyzing the outputs of two compared algorithms. If algorithm-A and algorithm-B both produce the same results for the same input, then the accuracy of the algorithm is 100%. Here, it is worth noting that due to the random nature, the results of heuristic-based algorithms may not be the same after each run even for the same input. In that case, the accuracy of the results is measured through a dependency test. In this simple test, the dependency of the selected attributes is tested against the complete set of features. If the selected feature subset and the entire feature set produce the same dependency, then the output was considered accurate.

2) PERCENTAGE DECREASE IN EXECUTION TIME

Reduction in execution time for an algorithm A as compared to algorithm B is ascertained by a measure called percentage decrease in execution time. This is one of the important measures that ensure the execution competence of an algorithm. Mathematically, it is represented by:

$$P\Delta t = 100 - \frac{E_1}{E_2} \times 100$$  \hspace{1cm} (19)

Formula for $P\Delta t$ is derived from [48]. Execution time reduction is measured in percentage for simplicity. For example, if an algorithm A takes 9 seconds for its complete execution and algorithm B executes in 3 seconds, then the improvement for algorithm B will be calculated as given in the equation below:

$$P\Delta t = 100 - \frac{3}{9} \times 100 = 66\%$$  \hspace{1cm} (20)

The algorithm was executed on a simple computer system and the system timer of the computer was used to measure the execution time of an algorithm. System timer starts as soon as the algorithm starts its execution and during its execution, the algorithm takes input from an excel file, performs calculations in the VBA environment and generates reducts. The output is received back into excel sheet and timer stops immediately after the computation of GDRs and acceptance of output. The start and stop time difference was calculated to measure the execution time of the algorithm.

3) OPTIMALITY

Rough set based feature selection technique uses the optimality criterion to find the shortest or minimal possible reducts while maintaining the originality of the dataset. Optimality measure specifies the minimal number of reducts obtained after the complete execution of the algorithm. Mathematically, it is represented by:

$$Opt(R) = |R|$$  \hspace{1cm} (21)

Here, Opt (R) represents the optimality and it is the cardinality of all the possible minimal number of reducts obtained as a result of an algorithm.

C. RUNTIME CONTEXT

Specification of runtime context includes overall runtime environment (hardware, software, processor, memory, etc.) where all the experiments were conducted. The simulation experiments were executed on a workstation equipped with Intel Core i5-8500 Quad-core processor (3.0 GHz clock speed) and 8 GBs of main memory. All the algorithms were implemented, run and tested using the same environment.

In order to prevent the potential impact of bias and deliver results without any influence, the following three things were taken care of: 1) environment settings for the execution of each algorithm for each dataset were kept the same. 2) Parameter’s settings (discussed in the subsequent paragraph) were retained the same. 3) It was ensured that all the Rough Set algorithms have a dynamic reduce based approach. Furthermore, while executing each algorithm, the priority of the algorithm was set to “high” so that the algorithm gets maximum CPU time and any parallel running process could have a minimum effect. Furthermore, each
algorithm was executed ten times and the average of the measures was taken to further reduce the effect of bias.

The number of sub-tables was equal for each dataset and does not depend upon the size of the dataset. This was done just to maintain consistency. Some of the parameters that were adjusted before the experiments are given in the Table 7. Each algorithm was executed 10 times for each dataset. The execution time of all algorithms for each dataset was noted down and the average of the execution time was recorded. Results accuracy was certified through dependency testing and optimal reducts along with their sizes were also taken into observations.

### Table 6. Sample datasets used.

| Sr. No. | Dataset             | Instances | Attributes | Dataset Characteristics                  | Classes |
|---------|---------------------|-----------|------------|------------------------------------------|---------|
| 1       | Vehicle             | 846       | 19         | Multivariate/ Categorical, Integer, Real  | 4       |
| 2       | Soyabeans           | 47        | 36         | Multivariate/ Categorical                | 4       |
| 3       | Tic-Tac-To          | 958       | 10         | Multivariate/ Categorical                | 2       |
| 4       | Spect               | 267       | 23         | Multivariate/ Integer, Categorical       | 2       |
| 5       | Optdigits           | 1797      | 65         | Multivariate, Integer                    | 10      |
| 6       | Landsat Satellite   | 2000      | 37         | Multivariate, Integer                    | 6       |
| 7       | SML 2010            | 2000      | 37         | Multivariate, Sequential, Text/ Real     | 6       |
| 8       | Wine Quality (White)| 4898      | 12         | Multivariate/Real                         | 6       |
| 9       | Miskolc II Hybrid IPS | 4601  | 58         | Text/Integer                             | 2       |
| 10      | Letter Recognition  | 2760      | 17         | Multivariate, Integer                    | 14      |

### TABLE 7. Parameters used for experiments.

| Parameter                  | Value |
|----------------------------|-------|
| No of Sub-tables (m)      | 03    |
| No. of Iterations (its)   | 10    |
| Size of sub-table          | Size of decision table / 3 |
| Stability Threshold (ε )   | 0.5   |

It should be noted that there are two parameters that are sensitive concerning the performance and accuracy. First, the number of parallel threads. Normally it is considered that more the number of parallel threads, the more enhanced is the performance. However, having more threads may have cost as well. In this research, three threads were used for dataset, the reason behind is that each dataset was divided into three subtables. It should be noted that increasing the number of subtables may affect the performance for large datasets while a smaller number of subtables may affect accuracy. So, based on our experience we divided each dataset in three subtables to get maximum accuracy and performance. Second important parameter is stability threshold ‘ε’, $0 \leq \varepsilon \leq 1$. ‘0’ means reduct instability whereas value of ‘1’ means a reduct is dynamic if it is also a reduct of all the sub-tables. In most cases it is too restrictive. Therefore, to generalize the concept of dynamic reducts, a reduct is considered to be stable and dynamic if it appears in certain proportion of generated sub-tables. Here, we set value of ‘ε’ equals to 0.5, which means that a reduct is dynamic if it is part of not less than half (at least two) of the sub-tables.

### D. RESULT ANALYSIS

Experiments were repeatedly performed using ten publicly accessible datasets from the UCI machine repository dataset. Various approaches described comprehensively in the previous section were compared with the proposed solution. The obtained results assert that in comparison with the contemporary algorithms, the proposed method is more effective and efficient.

1) ACCURACY

The Table 9 shows the accuracy achieved by the proposed approach against the compared contemporary algorithms. Experiments were performed on the selected UCI benchmark datasets. Accuracy test was performed manually using complete datasets and dependency was calculated for the complete dataset as well as the obtained GDRs. It was observed that the GDRs obtained by the proposed approach provides the maximum accuracy in comparison with the other GDR based algorithms. Experiments were performed ten times for each dataset and average accuracy was computed in percentage. Table 9 shows that the algorithms achieve 96% average accuracy showing 6%, 10%, and 31% improvement in accuracy as compared to RSTR [45], FAST [43], and RTraces [15] algorithms respectively. This proves the effectiveness of the proposed algorithm.

2) PERCENTAGE DECREASE IN EXECUTION TIME

The Soyabean dataset was normalized by converting the decision attribute values to integer due to its large memory requirements. Similarly, the Tic-Tac-To dataset’s decision classes were also normalized by converting into integer. The complete dataset was divided into three equal sub-tables and these equal-sized sub-tables were fed to all the algorithms for execution on the same machine. However, value for the number of iterations was “10” or it can be any value specified by the user. Execution time and output GDRs were noted for each algorithm. Each algorithm was executed ten times and the average time was calculated for ten datasets individually. It was observed that the Miskolc II Hybrid IPS dataset took maximum time to compute stable reducts as it is comprised of real values and has a large number of instances as well as a large number of attributes. It was also noted that those
datasets having a large number of attributes took more time in the execution and computation of stable reducts like Miskolc and Wine datasets, Letter Recognition, and Optidigits.

From the above given table, the experimental results show that:

1) The proposed algorithm reduces execution time by 31% as compared to the Fast Algorithm as exhibited in Figure 4. Although Fast algorithm does not compute GDRs from the entire dataset, still the computation time for the proposed approach is less than the Fast algorithm for each dataset.

2) In comparison with RSTR, the overall reduction in execution time was observed to be 47.13% as shown in Figure 5. RSTR algorithm is based on the conventional rough set based dependency calculation, therefore, it requires more computation time than the proposed approach.

3) In the case of RTraces, the proposed approach decreases execution time by 60.39% as plotted in Figure 6. From the obtained results, it is observed that the computation of traces for the extracted reducts takes more time and the proposed approach lacks such additional overhead.

4) Overall 46.13% reduction in execution time was witnessed for ten datasets by the PFS algorithm which proves that the proposed solution avoids any kind of exhaustive search and does not indulge in any complex computational process or operators. Consequently, performance is enhanced.

3) OPTIMALITY

It is also evident from the experimental results given in the above Table 8 that the result of the proposed solution is

**TABLE 8. Experimental results.**

| Sr. No | Dataset          | PFS     | FAST    | RSTR    | RTraces |
|--------|------------------|---------|---------|---------|---------|
|        |                  | Time(s) | GDRs    | Time(s) | GDRs    | Time(s) | GDRs    |
| 1      | Vehicle          | 2.9     | 1       | 4.7     | 1       | 6.8     | 1       | 7.3     | 3       |
| 2      | Soyabean         | 1.2     | 1       | 1.8     | 1       | 2.8     | -       | 2.8     | 3       |
| 3      | Tic-Tac-To       | 1.5     | 1       | 1.7     | 4       | 3.3     | 1       | 5.8     | 3       |
| 4      | Spect            | 2.2     | 1       | 2.5     | 11      | 2.2     | 11      | 2.4     | 9       |
| 5      | Optidigits       | 18.3    | 1       | 36.2    | 4       | 56.5    | 4       | 49.3    | 4       |
| 6      | Landsat Satellite| 15.5    | 1       | 16.8    | 3       | 16.3    | 3       | 31.7    | 3       |
| 7      | SML_2010         | 12.6    | 1       | 16.7    | 3       | 32.5    | 2       | 34.1    | 3       |
| 8      | Wine Quality (White) | 20.7 | 2       | 33.8    | 6       | 45.9    | 3       | 112.4   | 4       |
| 9      | Miskolc II Hybrid IPS | 68.6 | 1       | 229.9   | 3       | 213.1   | 1       | 294.5   | 8       |
| 10     | Letter Recognition| 27.5   | 5       | 34.1    | 6       | 48.5    | 6       | 78.3    | 9       |

**TABLE 9. Parameters used for experiments.**

| Algorithm | Overall accuracy |
|-----------|------------------|
| FAST      | 87%              |
| RSTR      | 73%              |
| RTraces   | 91%              |
| PFS       | 96%              |

Therefore, the reduction in execution time is observed for the proposed solution.
TABLE 11. Time complexity of each step of the proposed approach.

| Step            | Time Complexity |
|-----------------|-----------------|
| U\(\times\)C  | Complexity of Creating three random tables |
| \(3\times C\)  | Complexity of creating random feature vector |
| \(\frac{1}{2}\times(\text{DR} \times C')\) | Complexity of creating reduct using Relative Dependency |
| DR\(\times C\)  | Complexity of finding stable reduct |

So overall time complexity of the proposed approach is: \(|\text{DR}|^2 + \frac{1}{2}|U|^2|C'| + |U||C| + |C'|\). Here: DR = Dynamic Reduct set having maximum cardinality produced out of the three tables. U = Universe size C' = Randomly selected set of features C = Complete set of features present in dataset.

F. COMPARATIVE ANALYSIS

Experimental results show that our proposed algorithm PFS finds the minimum optimal dynamic reducts as compared to the challenging algorithms i.e. FAST, RSTR and RTraces. Reason behind such lesser number of reducts is that attributes are processed twice in PFS i.e. initially heuristically and later exhaustively. Hence, the result is optimized reducts. Use of relative attribute dependency technique allows the algorithm to generate reduct set. So that it will be considered as the minimum possible reduced set of attributes that can be designated as the final required optimal and stable reduct set. However, calculation of attribute dependency does not affect the overall execution time as the construction phase triggers the random selection of such attributes which qualify through dependency test and are fed to the optimization phase just to further squeeze these reducts. Whereas, FAST and RSTR algorithms suffer from exhaustive attribute search and use positive region based dependency calculation technique that consumes more execution time and sometimes may result in redundant, irrelevant and non-optimal set of reducts as evident in Table 8. RSTR algorithm is unable to produce stable reducts for Soyabean dataset. RTraces also does not guarantee optimal reducts as this approach uses exhaustive search technique to calculate reducts by repeatedly reducing datasets and analyzing traces of reducts. This results in non-optimal reduct set. Proposed solution achieves 96% accuracy whereas the challenging algorithms were able to achieve accuracy less than the proposed algorithm. The main reason behind this high accuracy is that proposed approach explicitly performs optimization to get rid of irrelevant and redundant features in optimization phase and results in a reduced optimal and stable reducts, out of which none can be removed further without affecting the dependency. Table 10 provides the %age decrease in execution time and results show that the proposed algorithm reduces execution time by 31% as compared to FAST algorithm, 47.13% as compared to RSTR and 60.39% as compared to RTraces algorithm. It is all because of the simplicity of computing relative attribute dependency by the proposed approach. PFS does not imply in any complex operators or computational process as well as avoids exhaustive search in the beginning with the complete set of attributes. Therefore, time efficiency of finding optimal reducts improves performance of the proposed algorithm.

G. STRENGTHS OF THE PROPOSED METHOD

This research work presents a novel feature selection approach that is based on simple construction, optimization and evaluation steps without indulging in complex operators. Some of the strengths of the proposed algorithm are as follows:

- The proposed algorithm shows favorable and inexpensive performance as compared to other computational intelligent tools in terms of solution potentials since it provides a better way to obtain stable attribute reduct subsets.
- One of the highlights of the proposed approach is that it optimizes the feature subset by removing irrelevant features.
- The proposed algorithm does not utilize any complex operators like local and global best in the fish swarm and particle swarm algorithms, crossover, and mutation in genetic algorithm. Hence, the simplicity of the proposed approach consequences in improved performance as compared to the other computationally expensive algorithms.
- Generation of stable reducts through feature sampling enables us to avoid exhaustive search, which ultimately cuts down the execution time. This makes the proposed approach suitable for average as well as large datasets.
- Use of relative dependency as optimizer circumvents the calculation of computationally expensive positive region which excludes the practice of traditional rough set based dependency measure for selection of feature subset in large datasets.
- Calculation of relative dependency during the optimization step does not affect the execution time in the second phase, as the sub-table has already been reduced in the first phase during the construction of parallel feature samples.
- PFS reduction method can be used as a reliable tool for feature selection using rough set theory as it generates optimal stable reducts within reduced execution time.

**H. LIMITATIONS**

Although the proposed approach significantly decreases the execution time of finding the Dynamic Reducts as compared to other approaches, however, it has some limitations as well.

- Algorithm works only for consistent data as relative reducts can only be derived from consistent dataset. However, in real life, it is common to have inconsistent datasets. For all such datasets, the proposed algorithm may compromise on accuracy. However, once the dataset is made consistent by removing the inconsistent records, the algorithm works fine with it.
- Algorithm computes dependency twice. Firstly, it calculates relative dependency by adding features into PFS and secondly, it calculates relative dependency for each attribute through backward elimination which affects the performance of the algorithm especially for large datasets. Although algorithm significantly enhances the performance, however, dealing with this issue may enhance the performance further.
- We were able to test our algorithm with medium to large datasets. However, for very large datasets, the algorithm still needs to be tested.

**I. THREAT TO VALIDITY**

Sometimes proposed algorithm compromises the accuracy over the reduction in execution time. It means that if the number of iterations is increased to obtain a maximum dependency-based feature subset, then the algorithm will take more time to execute. In contrast, if the number of iterations is reduced to have less execution time then the accuracy may suffer. Hence, it is observed from the experiments that there is a trade-off between execution time and accuracy. Although this problem is not observed all the time for all datasets. This may happen for some of the datasets. Mostly, the proposed algorithm generates stable reducts with maximum accuracy within reduced execution time.

**IX. CONCLUSION AND FUTURE WORK**

The fundamental objective of this research work is to find a feature selection technique based on rough set theory that is effective and efficient. A novel dynamic reduct finding algorithm has been implemented and empirically tested to realize this objective. The proposed solution ensures enhanced performance, while at the same time it achieves a reduction in dimensionality for large datasets and generates minimum optimal reducts with maximum accuracy. These dynamic reducts are stable as well. The parallel feature sampling algorithm takes a dataset as an input, divides it into multiple sub-tables. These sub-tables are further reduced through the construction of feature samples simultaneously for each sub-table by a random selection of features. These feature samples may contain some irrelevant or redundant features. Therefore, these are optimized and relative reducts are computed through backward eliminating strategy. These relative reducts are evaluated for their fitness against a stability threshold value and stability coefficient of each reduct is compared with this threshold. Relative reducts having stability coefficient larger than the threshold are nominated as dynamic reducts. Computational efficiency is achieved by avoiding the conventional dependency calculation technique and the use of relative dependency. Results show that the PFS algorithm handles irrelevant and redundant features for datasets beyond small size and outperforms its competitors. Hence, objectives are achieved in a real sense and the proposed algorithm meets the objective of reduction of the curse of dimensionality in true spirit. It improves the efficiency and speed of calculating generalized dynamic reducts. This algorithm is proved to be efficient with the evaluation of its optimality, percentage decrease in execution time and accuracy. In this research work, three comparison parameters were utilized for the evaluation purpose. However, the proposed algorithm can be tested using some other evaluation parameters. Further refinement of the proposed solution, experiments exhausting datasets with a larger number of attributes and objects, and comparison between the proposed algorithm and other approaches are attention-grabbing and key future concerns. Furthermore, the proposed algorithm was developed for static datasets, but it can be modified with little improvement for incremental data as well.

**REFERENCES**

[1] M. S. Raza and U. Qamar, “A parallel rough set based dependency calculation method for efficient feature selection,” *Appl. Soft Comput.*, vol. 71, pp. 1020–1034, Oct. 2018.
[2] A. Kaur and A. Datta, “A novel algorithm for fast and scalable subspace clustering of high-dimensional data,” *J. Big Data*, vol. 2, no. 1, p. 17, Dec. 2015.
[3] V. Boldón-Canedo, N. Sánchez-Maroto, and A. Alonso-Betanzos, “Feature selection for high-dimensional data,” *Prog. Artif. Intell.*, vol. 5, no. 2, pp. 65–75, 2016.
[4] K. Sutha and J. J. Tamilselvi, “A review of feature selection algorithms for data mining techniques,” *Int. J. Comput. Sci. Eng.*, vol. 7, no. 6, p. 63, 2015.
[5] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, “Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 826–830, 2017.
[6] Z. Pawlak, “Rough sets,” *Int. J. Comput. Inf. Sci.*, vol. 11, no. 5, pp. 341–356, Oct. 1982.
[7] J. Bazan, “A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables,” *Rough Sets Knowl. Discovery*, vol. 1, pp. 321–365, 1998.
Y. Chen, Q. Zhu, and H. Xu, “An attribute reduction algorithm by switching exhaustive and heuristic computation of relative reducts,” in *Proc. IEEE Int. Conf. Granular Comput.*, Aug. 2010, pp. 265–270.

Y. Kudo and Y. Okada, “A heuristic method for discovering biomarker candidates based on rough set theory,” *Bioinformatics*, vol. 6, no. 5, p. 200, 2011.

S. S. Shrivastavat, A. Sant, and R. P. Aharwal, “An overview on data mining approach on breast cancer data,” *Int. J. Adv. Comput. Res.*, vol. 3, no. 4, p. 256, 2013.

C.-H. Lee, S.-H. Seo, and S.-C. Choi, “Rule discovery using hierarchical classification structure with rough sets,” in *Proc. Joint 9th IFSA World Congr. 20th NAIPS Int. Conf.*, vol. 1, Jul. 2001, pp. 447–452.

U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Mag.*, vol. 17, no. 3, p. 37, 1996.

R. Jensen and Q. Shen, *Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches*, vol. 8. Hoboken, NJ, USA: Wiley, 2008.

J. Hua, W. D. Tembe, and E. R. Dougherty, “Performance of feature-selection methods in the classification of high-dimension data,” *Pattern Recognit.*, vol. 42, no. 3, pp. 409–424, Mar. 2009.

J. G. Bazan and A. S. P. Synak, “Dynamic reducts as a tool for extracting laws from decision tables,” in *Proc. Int. Symp. Methodol. Intell. Syst.* Berlin, Germany: Springer, 1994, pp. 346–355.

T. Chowdhury, S. K. Setua, and S. Chakraborty, “A novel rules optimizer with feature selection using rough-entropy-coverge partitioning based reduct,” in *Proc. 3rd Int. Conf. Comput., Commun., Control Inf. Technol. (CIST)*, Feb. 2015, pp. 1–7.

H. Son and C. Kim, “Forecasting short-term electricity demand in residential sector based on support vector regression and fuzzy-rough feature selection with particle swarm optimization,” *Proc. Eng.*, vol. 118, pp. 1162–1168, 2015.

J. H. Lee, J. R. Anarak, C. W. Ahn, and J. An, “Efficient classification system based on Fuzzy–Rough Feature Selection and Multitree Genetic Programming for intension pattern recognition using brain signal,” *Expert Syst. Appl.*, vol. 42, no. 3, pp. 1644–1651, 2015.

W. Ziarko, “Dependency analysis and attribute reduction in the probabilistic approach to rough sets,” in *Feature Selection for Data and Pattern Recognition*, Berlin, Germany: Springer, 2015, pp. 93–111.

X. Gu, Y. Li, and J. Jia, “Feature selection for transient stability assessment based on kernelized fuzzy rough sets and memetic algorithm,” *Int. J. Electr. Power Energy Syst.*, vol. 64, pp. 664–670, Jan. 2015.

R. Jensen, “Combining rough and fuzzy sets for feature selection,” Ph.D. dissertation, School Inform., Univ. Edinburgh, Edinburgh, U.K., 2005.

T. Maini, A. Kumar, R. K. Misra, and D. Singh, “Fuzzy rough set-based feature selection with improved seed population in PSO and IDS,” in *Computational Intelligence: Theories, Applications and Future Directions*, vol. 2. Singapore: Springer, 2019, pp. 137–149.

C. Wang, Y. Huang, M. Shao, and X. Fan, “Fuzzy rough set-based attribute reduction using distance measures,” *Knowl.-Based Syst.*, vol. 164, pp. 205–212, Jan. 2019.

J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proc. Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov./Dec. 1995, pp. 1942–1948.

A. Zainal, M. A. Maaroof, and S. M. Shamsuddin, “Feature selection using Rough-DPSO in anomaly intrusion detection,” in *Proc. Int. Conf. Comput. Sci. Appl.* Berlin, Germany: Springer, 2007, pp. 512–524.

C.-S. Yang, L.-Y. Chuang, J.-C. Li, and C.-H. Yang, “Chaotic maps in binary particle swarm optimization for feature selection,” in *Proc. IEEE Conf. Soft Comput. Ind. Appl.*, Jun. 2008, pp. 107–112.

S. Abdul-Rahman, Z.-A. Mohamed-Hussein, and A. A. Bakar, “Integrating rough set theory and particle swarm optimisation in feature selection,” in *Proc. 10th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2010, pp. 1009–1014.

N. Wahid, Y. Y. Chung, W.-C. Yeh, and G. Liu, “Feature selection using a novel swarm intelligence algorithm with rough sets,” in *Proc. DMIN*, 2010, pp. 294–300.

A. Bhattacharya, R. T. Goswami, and K. Mukherjee, “A feature selection technique based on rough set and improvised PSO algorithm (PSORS-FS) for permission based detection of Android malware,” *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 7, pp. 1893–1907, Jul. 2019.

R. K. Huda and H. Banka, “Efficient feature selection and classification algorithm based on PSO and rough sets,” *Neural Comput. Appl.*, vol. 31, no. 8, pp. 4287–4303, Aug. 2019.

Y. Chen, Q. Zhu, and H. Xu, “Finding rough sets with fish swarm algorithm,” *Knowl.-Based Syst.*, vol. 81, pp. 22–29, Jun. 2015.
MUHAMMAD IBRAHIM received the Ph.D. degree in computer science from the Capital University of Science and Technology, Islamabad, in 2019. He is currently working as an Assistant Professor with the Department of Information Technology, The University of Haripur. He has published a number of Papers in Top rank Journals (i.e., Cluster Computing, IEEE Access), and several conference papers. His areas of research include cloud computing, fog computing, the IoT, and blockchain.

ABDULLAH BAZ (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering from UQU, in 2002, the M.Sc. degree in electrical and computer engineering from KAU, in 2007, and the M.Sc. degree in communication and signal processing and the Ph.D. degree in computer system design from Newcastle University, in 2009 and 2014, respectively. He was a Vice-Dean, and then the Dean of the Deanship of Scientific Research with UQU, from 2014 to 2020. He is currently an Assistant Professor with the Computer Engineering Department, a Vice-Dean of DFMEA, the General Director of the Decision Support Center, and the Consultant of the University Vice Chancellor with UQU. His research interests include data science, ML, AI, VLSI design, EDA/CAD tools, coding and modulation schemes, image and vision computing, and computer system and architecture. Since 2015, he has been served as a Review Committee Member of the IEEE International Symposium on Circuits and Systems (ISCAS) and a member of the Technical Committee of the IEEE VLSI Systems and Applications. He served as a Reviewer in a number of journals, including the IEEE INTERNET OF THINGS, IET Computer Vision, Artificial Intelligence Review, and IET Circuits, Devices and Systems.

HOSAM ALHAKAMI (Member, IEEE) received the B.Sc. degree in computer science from King Abdulaziz University, Saudi Arabia, in 2004, the M.Sc. degree in internet software systems from Birmingham University, Birmingham, U.K., in 2009, and the Ph.D. degree in software engineering from De Montfort University, in 2015. From 2004 to 2007, he worked with Software Development Industry, where he implemented several systems and solutions for a national academic institution. His research interests include algorithms, semantic web, and optimization techniques. He focuses on enhancing real-world matching systems using machine learning and data analytics in a context of supporting decision-making.

MUHAMMAD ANWAAR SAEED received the Ph.D. degree in computer science from the National College of Business Administration and Economics (NCBA&E), Lahore, Pakistan. He has joined the Virtual University (VU) of Pakistan, in April 2006, and is currently working as an Assistant Professor with the Computer Science Department. His area of research is key generation for data encryption and information security. He is also interested in quantum computing especially encryption mechanisms used in this field. He is the author of a monograph on framework for self-organizing encryption in ubiquitous environment, published by VDM Verlag, in 2010. He has also published research papers on his area of interest. Before joining VU, he has ample experience of both software development and network management.