Dynamic Fuzzy Rule Interpolation and Its Application to Intrusion Detection

Nitin Naik, Ren Diao, and Qiang Shen

***Abstract*—Fuzzy rule interpolation (FRI) offers an effective approach for making inference possible in sparse rule-based systems (and also for reducing the complexity of fuzzy models). However, requirements of fuzzy systems may change over time and hence, the use of a static rule base may affect the accuracy of FRI applications. Fortunately, an FRI system in action will produce interpolated rules in abundance during the interpolative reasoning process. While such interpolated results are discarded in existing FRI systems, they can be utilized to facilitate the development of a dynamic rule base in supporting subsequent inference. This is because the otherwise relinquished interpolated rules may contain possibly valuable information, covering regions that were uncovered by the original sparse rule base. This paper presents a dynamic fuzzy rule interpolation (D-FRI) approach by exploiting such interpolated rules in order to improve the overall system's coverage and efficacy. The resulting D-FRI system is able to select, combine, and generalize informative, frequently used interpolated rules for merging with the existing rule base while performing interpolative reasoning. Systematic experimental investigations demonstrate that D-FRI outperforms conventional FRI techniques, with increased accuracy and robustness. Furthermore, D-FRI is herein applied for network security analysis, in devising a dynamic intrusion detection system (IDS) through integration with the Snort software, one of the most popular open source IDSs. This integration, denoted as D-FRI-Snort hereafter, delivers an extra amount of intelligence to predict the level of potential threats. Experimental results show that with the inclusion of a dynamic rule base, by generalising newly interpolated rules based on the current network traffic conditions, D-FRI-Snort helps reduce both false positives and false negatives in intrusion detection.**

***Index Terms*—Dynamic rule generalization, fuzzy rule interpolation (FRI), interpolated rules, intrusion detection, Snort, sparse rule base.**

I. Introduction

FUZZY rule interpolation (FRI) [1]–[4] offers the most effective reasoning mechanism to perform fuzzy reasoning based on a sparse rule base. The classical fuzzy inference methods cannot work to their full potential in such circumstances because given knowledge does not cover the entire problem domain. However, requirements of fuzzy systems may change over time and therefore, the use of a static rule base may affect the effectiveness of FRI due to the absence of the most concurrent (dynamic) rules. Designing a dynamic rule base yet needs additional information. Fortunately, a fuzzy reasoning system that utilizes FRI may produce a large number of interpolated rules during the interpolative reasoning process. Such interpolative results are always discarded once the required outcomes have been obtained in the present applications of FRI. Nonetheless, these relinquished interpolated rules may contain possibly valuable information, covering regions that were uncovered by the original sparse rule base and thus, may be collected and utilized to create a dynamic rule base through generalization.

Several approaches have been proposed for dynamic adaptation of the rule base within conventional fuzzy systems where a dense rule base is employed [5]–[7]. In particular, optimization-based techniques have been developed for the automatic generation of fuzzy rule-based models [8]–[10]. These techniques provide a dynamic and possibly, real-time dense rule base for the system, thereby entailing more accurate reasoning results. Unfortunately, the learning methods developed for dense rule-based systems cannot be directly applied to sparse rule-based systems due to their notion of the rules fully covering the problem space, and the underlying computational differences between the use of classical fuzzy inference and that of FRI. Besides, a useful dynamic approach devised for sparse rule bases has to cope with the demand of involving less computational overheads since typically, the interpolation process already incurs significant additional cost.

Inspired by the above observations and based on the initial results from preliminary investigations [11]–[13], this paper presents a comprehensive approach to dynamic FRI (D-FRI). The work is conceived such that it can improve not only the overall interpolative coverage and efficacy, but also help reduce the overheads of subsequent interpolation where dynamically promoted rules match future observations. The development of a dynamically enriched and revised rule base is enabled by exploiting previously experienced interpolated rules, which are generated as a byproduct of the interpolative reasoning process. In particular, the collection of the interpolated rules is prepartitioned into hypercubes (or multidimensional blocks), in order to reduce the complexity of required generalization process. All those nonempty hypercubes (i.e., those hit by at least one of the interpolated results) are discovered and fed as the input into a GA-based clustering algorithm, which finds the "best"

Manuscript received December 29, 2016; revised May 29, 2017 and August 14, 2017; accepted September 8, 2017. Date of publication December 4, 2017; date of current version August 2, 2018. *(Corresponding author: Qiang Shen.)*

N. Naik is with the Defence School of Communications and Information Systems, Ministry of Defence, U.K. (e-mail: nitin.naik100@mod.gov.uk).

R. Diao is with the Wandera Ltd., London W1W 8HJ, U.K. (e-mail: ren.diao@wandera.com).

Q. Shen is with the Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, Aberystwyth SY23 3DB, U.K. (e-mail: qqs@aber.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TFUZZ.2017.2755000

cluster arrangement based on a predefined fitness function (here, the Dunn Index (DI) [14] is adopted for implementation). The resulting densest clusters that have accumulated a sufficient number of candidate rules are selected for rule aggregation and promotion, and this is done through an iterative process. Systematic comparative investigations are carried out against conventional FRI that uses just the original sparse rule base, demonstrating that D-FRI possesses higher accuracy and robustness level.

In addition to evaluation of D-FRI against benchmark datasets, it is important to examine how it may work in a real-world application setting. Security is one of the major concerns of any organization regardless of their size and nature of work. Security attacks and their types are countless, however, network intrusion attack is one of the key concerns, being an illicit attempt that compromises the confidentiality, integrity, or availability of the organizational IT infrastructure [15], [16]. Consequently, D-FRI is employed herein to support network security analysis, in building an intelligent intrusion detection system (IDS). It is used in conjunction with Snort, one of the most popular open source IDSs, resulting in a dynamic IDS named D-FRI-Snort. Experimental studies show that the integration of D-FRI and Snort delivers an extra level of intelligence in an effort to predict the level of potential threats. With a dynamic rule base obtained by promoting newly interpolated rules that reflect more relevant network traffic conditions, D-FRI-Snort helps reduce both the false positives and the false negatives produced by the original Snort.

The remainder of this paper is organized as follows. Section II briefly outlines the background techniques that are useful to implement the proposed work, including: a specific and popular form of FRI, termed scale and move transformation based FRI (T-FRI) [17], [18], genetic algorithms (GAs), intrusion detection, and Snort. Section III presents the theoretical design and implementation of D-FRI. Section IV provides the simulation results of D-FRI on benchmark datasets, verifying its correctness and accuracy by comparing it with T-FRI. Section V describes the specification of D-FRI-Snort, applying D-FRI to network security analysis. Section VI presents the experimental results of D-FRI-Snort, demonstrating its effectiveness. Finally, Section VII concludes the paper and suggests a number of future areas of extension.

II. Background

A. FRI Approaches

Many fuzzy reasoning systems cannot contain a dense rule base because of the nature of a particular application area and therefore, they rely on a sparse rule base. Such reasoning tools compute approximate conclusions by employing FRI, a well-established technique in developing fuzzy systems. Over the past 25 years, a number of important FRI techniques have been introduced, reflecting a variety of viewpoints that are followed in the implementation of the underlying approaches. These include pioneering work based on: exploiting fuzzy relations in the Cartesian product of input and output space [1], extending classical linear interpolation from the object level to rules [2], [3], computing interpolation over splines [4], minimizing nonconvex conclusion issues [19], exploring vague environment by the use of crisp sets [20], inferring from modified alpha-cuts [21]–[23], ensuring the preservation of fuzziness [24], reinforcing the modified alpha-cut technique over multidimensions [25], manipulating the slopes of the fuzzy sets [26], transferring similarity constraints to guarantee convex outcomes [27], performing transformation that preserves spatial geometric properties [28], assuring the maintenance of graduality in representation [29], working on the flank functions [30], utilizing scale and move transformations of representative values [17], [18], integrating cutting and transformation techniques [31], making use of basis-spline (B-Spline) functions [32], conducting interpolation via linguistic term shifting and polar cuts [33], ranking values of fuzzy sets [34], and retaining conservation of shape types specific to domain partition [35]. Note that while interpolation is commonly used as a term in describing these approaches, many of which can be equivalently applied to perform reasoning with fuzzy rule extrapolation.

Much of the aforementioned seminal work on FRI has inspired the present research as well as many recently reported further developments that support more sophisticated FRI. The latter include those using weighted [36], higher order [37]–[39] or hierarchical [40] rule representation, and those allowing for self error-correction [41], [42] or adaptation [43] of interpolative results and for interpolation and extrapolation via backward rule chaining [44]. A full coverage and detailed analysis of the FRI literature is beyond the scope of this paper, but many recent algorithms have followed the transformation-based approach which exploits generalized modus ponens in fuzzy inference [17], [18]. Here, in developing D-FRI, the T-FRI approach is also adopted to serve as the underlying FRI mechanism. However, other FRI methods may be utilized as alternatives if preferred.

B. Transformation-Based FRI

This section provides a brief overview of T-FRI, including both the underlying concepts and the interpolation procedure. For simplicity and owing to their popularity, in this work, fuzzy sets are represented using triangular membership functions. Suppose that an original, sparse rule base $\mathbb{R}$ exists, with rules $R_i \in \mathbb{R}$ and an observation O:

R_i: IF x_1 is $A_{i,1}$, …, and x_j is $A_{i,j}$, …, and X_N is $A_{i,N}$, THEN y is B_i
O: $A_{\circ,1}, \ldots, A_{\circ,j}, \ldots, A_{\circ,N}$

where i indexes rule R_i in the sparse rule base, $A_{i,j} = (a_0, a_1, a_2)$ is the triangular linguistic term defined on the domain of the antecedent variable x_j, $j \in \{1, \ldots, N\}$, with N being the total number of antecedents, and B_i is the consequent.

Let a given observed fuzzy value of the variable x_j be denoted by $A_{\circ,j}$, and the representative value $\mathrm{rep}(A)$ of a triangular fuzzy set A be defined as the mean of the X coordinates of the triangle's three odd points: the left and right extremities of the support a_0, a_2 (with membership values $= 0$), and the normal point a_1 (with membership value $= 1$)

$$\mathrm{rep}(A) = (a_0 + a_1 + a_2)/3. \quad (1)$$

Given the above notations, the core of the T-FRI can be summarized as follows, while more details can be found in [17], [18].

1) Determine Closest Rules for New Observation: The distance between R_i and O is determined by computing the aggregated distance of all antecedent variables

$$d(R_i, O) = \sqrt{\sum\nolimits_{j=1}^{N} d_j^2},\ \ d_j = \frac{d(A_{i,j}, A_{\circ,j})}{\text{range}_{x_j}} \tag{2}$$

where $d(A_{i,j}, A_{\circ,j}) = |\text{rep}(A_{i,j}) - \text{rep}(A_{\circ,j})|$ is the distance between the representative values of the two fuzzy sets in the jth antecedent, with $\text{range}_{x_j} = \max x_j - \min x_j$ over the domain of the variable x_j. $d_j \in [0,1]$ is therefore the normalized result of the otherwise absolute distance measure, so that distances are compatible with each other across different variable domains. The M, $M \geq 2$ rules which have the least distance measurements, with regard to the observed values $A_{\circ,j}$ are then chosen to perform the interpolation in order to obtain the required conclusion $B_\circ$.

2) Construct Intermediate Rule: Guided by the new observation, an intermediate rule is needed to approximately approach the final outcome of the consequent, by linearly interpolating the previously identified M closest rules to the observation. The antecedents of this rule are initially estimated by manipulating the antecedents of the M rules

$$A_j^{\dagger\dagger} = \sum_{i=1}^{M} \omega_{i,j} A_{i,j} \tag{3}$$

where

$$\omega_{i,j} = \frac{\omega_{i,j}^{\dagger}}{\sum_{k=1}^{M} \omega_{i,j}^{\dagger}},\ \omega_{i,j}^{\dagger} = \exp^{-d(A_{i,j}, A_{\circ,j})}. \tag{4}$$

These $A_j^{\dagger\dagger}$ are then shifted to $A_j^{\dagger}$ such that they have the same representative values as those of $A_{\circ,j}$

$$A_j^{\dagger} = A_j^{\dagger\dagger} + \delta_j \text{range}_{x_j} \tag{5}$$

where δ_j is the bias between $A_{\circ,j}$ and ${A_j}^{\dagger}$ on the jth variable domain

$$\delta_j = \frac{\text{rep}(A_{\circ,j}) - \text{rep}({A_j}^{\dagger})}{\text{range}_{x_j}}. \tag{6}$$

From this, the shifted intermediate consequent $B^{\dagger}$ can be computed, with the parameters ω_{B_i} and δ_B being aggregated from those regarding the antecedents of ${A_j}^{\dagger}$, such that

$$\omega_{B_i} = \frac{1}{N}\sum_{j=1}^{N} \omega_{i,j},\ \delta_B = \frac{1}{N}\sum_{j=1}^{N} \delta_j. \tag{7}$$

3) Scale and Move Transformations: The above intermediate rule ensures that the representative values of its antecedents are the same as those of the corresponding elements in the given observation. In order to make the fuzzy values in this rule also the same as the observation (so that the observation matches the resulting rule), scale and move transformations will be required.

Thus, guided by the observation, the current support of $A_j^{\dagger}$, $({a_0}^{\dagger}, {a_2}^{\dagger})$ is first rescaled to a new support $({a_0}^{+}, {a_2}^{+})$, such that ${a_2}^{+} - {a_0}^{+} = s_j \times (a_2^{\dagger} - a_0^{\dagger})$

$$\begin{cases} a_0^{+} = \frac{{a_0}^{\dagger}(1+2s_j) + {a_1}^{\dagger}(1-s_j) + {a_2}^{\dagger}(1-s_j)}{3} \\ {a_1}^{+} = \frac{{a_0}^{\dagger}(1-s_j) + {a_1}^{\dagger}(1+2s_j) + {a_2}^{\dagger}(1-s_j)}{3} \\ {a_2}^{+} = \frac{{a_0}^{\dagger}(1-s_j) + {a_1}^{\dagger}(1-s_j) + {a_2}^{\dagger}(1+2s_j)}{3} \\ s_j = \frac{{a_2}^{+} - {a_0}^{+}}{{a_2}^{\dagger} - {a_0}^{\dagger}} \end{cases}. \tag{8}$$

From this, the scaling factor s_B for the consequent can then be calculated by

$$s_B = \frac{\sum_{j=1}^{N} s_j}{N}. \tag{9}$$

The resulting rescaled fuzzy values are subsequently moved using the following move rate m_j, so that the final transformed fuzzy sets match the corresponding elements in the observation

$$\begin{cases} m_j = \frac{3(a_0 - {a_0}^{+})}{{a_1}^{+} - {a_0}^{+}},\ a_0 \geq {a_0}^{+} \\ m_j = \frac{3(a_0 - {a_0}^{+})}{{a_3}^{+} - {a_2}^{+}},\ \text{otherwise} \end{cases}. \tag{10}$$

From this, the move factor m_B for the consequent is calculated such that

$$m_B = \frac{\sum_{j=1}^{N} m_j}{N}. \tag{11}$$

The final interpolated result $B_\circ$ can now be estimated by applying the scale and move transformation to $B^{\dagger}$, using the parameters s_B, and m_B. Note that given both transformations are linear operations, the order of applying the scale and move transformations can be reversed.

C. Genetic Algorithms

A GA is a metaheuristic search method for solving constrained and unconstrained optimization problems, based on Darwinian principle of survival of the fittest individuals in natural selection [45]. The computational implementation of a GA chiefly depends on the two important operators: *crossover* and *mutation*. In the beginning, the potential solution to the problem is encoded in the form of a *chromosome*. The initial population, a set of chromosomes, is generated randomly and their members are then selected for the reproductive process based on their fitness (or quality) values. The chromosomes with better fitness values are more likely to take part in the reproduction of offsprings. The reproductive process is repeated until certain desired or limiting conditions are met, such as achieving the desired fitness level for a candidate solution or reaching a maximum number of generations. The generic procedure of GAs can be summarized as follows [46], [47]:

1) *Initialization:* Generate random population $\mathbb{P}$ of $|\mathbb{P}|$ chromosomes $[X_1, X_2, ..., X_{|\mathbb{P}|}]$, where each chromosome $X_i, i = 1, ..., |\mathbb{P}|$, is an order collection of genes $= [x_1^i, \ldots, x_r^i, x_{r+1}^i, \ldots, x_{|X_i|}^i]$.
2) *Fitness Calculation:* Evaluate the fitness $f(X_i)$ of each chromosome X_i in the population $\mathbb{P}$, $\forall i \in \{1, ..., |\mathbb{P}|\}$,

where the quality function $f(.)$ is predefined in the domain.

3) *Chromosome Selection:* Select two parent chromosomes X_p and X_q from a population $\mathbb{P}$ according to their fitness; generally, the fitter, the bigger chance to be selected.
4) *Crossover:* With a crossover rate δ_c, cross over the parents X_p and X_q to form new offsprings ($X_p^{'}$ and $X_q^{'}$); in the event that no crossover is performed, the offsprings are exact copies of their parents.
5) *Mutation:* With a mutation rate δ_m, mutate the offsprings ($X_p^{'}$ and $X_q^{'}$) at each locus (position in chromosome).
6) *Acceptance:* The new offsprings ($X_p^{'}$ and $X_q^{'}$) then together form the new population $\mathbb{P}_{\text{new}}$, and are used for the subsequent generation.
7) *Iteration:* If the given condition is not satisfied, repeat the process from the step-*Fitness Calculation*.
8) *Termination:* If the termination condition is satisfied, stop, and return the best chromosome X^{best} in the final population (as the solution to the problem).

D. Intrusion Detection

An IDS is a software which monitors all inbound and outbound traffic and attempts to identify unauthorized, illicit, and anomalous behavior in the traffic that may compromise system and network security. Advanced IDSs may also differentiate between insider attacks and external attacks [48]. An intrusion prevention system (IPS) is the enhanced version of an IDS which can block threats in addition to detecting them. Unlike IPS, an IDS is mostly a passive device used for gathering, identifying, logging, and alerting purposes. Plethora of IDSs/IPSs are available, including: Snort, OSSEC, OSSIM, Suricata, Bro, Fragroute, BASE, Kismet, and Sguil.

IDSs can be categorized into anomaly-based, misuse/signature-based, network-based, and host-based [49]. In an anomaly-based IDS, the system administrator defines the baseline state of the network. Subsequently, the IDS monitors network traffic and compares it against stored patterns of normal behavior. It alerts the administrator or user when traffic is detected which is significantly different from the normal behavior. In misuse-based IDSs, the system administrator maintains a large database of known attack signatures. Subsequently, the IDS monitors network traffic and compares it against stored patterns of signatures or attributes from known malicious threats. In network-based IDSs (NIDS), the alert software is installed only at specific points, such as servers, switch, gateway, or router, which act as an interface between the outside environment and the network segment to be protected. Subsequently, an NIDS processes and flags any suspicious traffic through captured packets. In host-based IDSs (HIDS), the alert agent/application is installed on every network computer that has two-way access to the outside environment, such as the Internet. Subsequently, an HIDS monitors traffic and also produces an active response.

E. Snort

Snort is one of the most popular open source network intrusion detection and prevention systems. It was inducted into InfoWorld's open source hall of fame in 2009 [50], capable of performing real-time traffic analysis and packet-logging on IP networks [51]. Snort is based on *libpcap* (library packet capture) package, which is a system-independent interface for user-level packet capture and widely used in TCP/IP traffic sniffers and analyzers [52]. It is mainly a signature-based IDS, however, it can also act as an anomaly-based IDS and protocol analyzer by employing certain extra mechanisms. Snort uses various preprocessors and performs protocol analysis and content searching and matching for detecting several attacks, such as denials of service, buffer overflows, CGI attacks, port scanning attacks, SMB probes, worms, and OS fingerprinting attempts [53]. Whenever Snort detects suspicious activity, it sends a real-time alert to syslog file, a user-specified file, a UNIX socket, an alert database or screen/dashboard depending on its configuration.

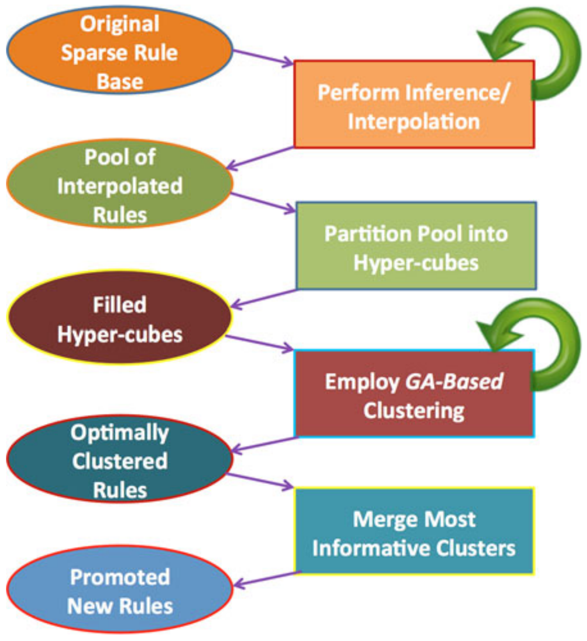

Original Sparse Rule Base
Perform Inference/ Interpolation
Pool of Interpolated Rules
Partition Pool into Hyper-cubes
Filled Hyper-cubes
Employ GA-Based Clustering
Optimally Clustered Rules
Merge Most Informative Clusters
Promoted New Rules

Fig. 1. Process of D-FRI.

III. D-FRI: Design and Implementation

A. D-FRI Outline

The overall D-FRI process is shown in Fig. 1. It starts with the initial running of a certain FRI system (here, T-FRI), which contains a set of original (sparse) rules $\mathbb{R}$. It generates an increasingly large number of interpolated rules which are accumulated as a pool of interpolated rules $\mathbb{R}'$. The domains of all the antecedents of $\mathbb{R}'$ are jointly divided into a set of hypercubes $\mathbb{H}$. Among all hypercube, only those nonempty hypercubes $\mathbb{H}^*$ (i.e., those being hit by at least one interpolated rule) are selected as input to the GA-based clustering algorithm, which helps reduce the computational complexity of the GA (with superfluous hypercubes removed). The GA-based clustering algorithm finds the "best" clustering arrangement that provides a set of strong hypercubes $\mathbb{H}^1$ and another set of weak hypercubes $\mathbb{H}^0$. Here, each strong hypercube contains many rules (over a certain predefined number) while each weak hypercube is less densely populated with rules. After this, weak hypercubes are merged to become strong hypercubes. The benefit of using such an approach is that the GA can find the best clusters without assigning a predetermined number of emerging clusters. Once the clustering process is completed, only those clusters containing adequate interpolated rules (again, with the number higher than a certain threshold) are selected for the subsequent rule promotion process, through an aggression procedure to update the rule base $\mathbb{R}$.

Such a D-FRI process is of intuitive appeal by taking advantage of the interpolated results produced by an FRI system over time. In doing so, it reduces the interpolation overheads for frequent and similar observations once they have been dealt with, by directly employing the compositional rule of inference (CRI). As outlined above, D-FRI is a flexible and generalized approach with no restriction of using any particular FRI, fitness function, and rule promotion scheme. The various stages within a D-FRI process, including input space partitioning, interpolated rule clustering, and dynamic rule promotion are detailed below.

B. Partitioning of Antecedent Space

A grid decomposition approach is employed to discover the regions delimited by antecedents that are uncovered by the original sparse rules in $\mathbb{R}$; and the regions that are covered by the collection $\mathbb{R}'$ of interpolated rules. Based on the value ranges of the antecedent variables, the product space of the antecedent domains is thus divided into a set of hypercubes $\mathbb{H}$. Subsequently, the antecedent values of a rule R (an original rule R_k or an interpolated rule $R_k^{'}$) is checked whether it lies within the boundaries of a certain H_p. If so, it is said to hit the hypercube H_p

$$R \in H_p \text{ if } \mathrm{rep}(A_{k,j}) \in [\min H_{p,j}, \max H_{p,j}), j \in \{1, \ldots, N\} \tag{12}$$

where $A_{k,j}$ is the value of the jth antecedent of the rule R.

Depending on the requirement of a sparse rule-based system, the size and number of hypercubes can be adjusted. While there is no such specific domain-dependent knowledge is available, the initial setting is determined by the underlying numbers of possible fuzzy values per individual antecedent variables. Without losing generality, in the present work, each antecedent dimension is evenly divided into η intervals. Therefore, the total number of hypercubes is $|\mathbb{H}| = \eta^N$. Initially, all the hypercubes are examined, but in performing the clustering of the interpolated results, only nonempty hypercubes $\mathbb{H}^*$ (i.e., those with at least one hit) are to be used.

$$\mathbb{H}^* \subseteq \mathbb{H} \quad \forall H \in \mathbb{H}^*, |H| \neq 0. \tag{13}$$

C. Clustering of Interpolated Rules

A GA-based algorithm is proposed for the clustering of interpolated rules, which is shown in Algorithm 1. It forms the clusters of similar interpolated rules $R' \in H$ by utilizing only nonempty hypercubes, reducing its computational complexity by discarding all the empty hypercubes. The implementation procedure of the proposed GA-based clustering algorithm is detailed below.

Algorithm 1: Genetic Algorithm for Clustering.

```
$\mathbb{P}_{new}$, new population
$X_i^{'}, i^{th}$ chromosome of $\mathbb{P}_{new}$
$X^{best}, X^{best} \in \mathbb{P}, f(X^{best}) = \max_{\forall X \in \mathbb{P}} f(X)$
$f(X_i)$, fitness value of $X_i$
$\delta_c$ , crossover rate
$r$ , random number, where $0 \leq r \leq 1$
$k_{max}$ , maximum number of generations
while $(k_{max} \neq 0)$ do
    $\mathbb{P}_{new} = \phi$
    for $(i = 1; i < |\mathbb{P}|; i + 2)$ do
        $X_i^{'} = roulette\_wheel\_selection(\mathbb{P})$
        $X_{i+1}^{'} = roulette\_wheel\_selection(\mathbb{P})$
        if $(r < \delta_c)$ then
            $X_i^{'} = crossover(X_i^{'}, X_{i+1}^{'}, true)$
            $X_{i+1}^{'} = crossover(X_i^{'}, X_{i+1}^{'}, false)$
        $X_i^{'} = mutate(X_i^{'})$
        $X_{i+1}^{'} = mutate(X_{i+1}^{'})$
        $\mathbb{P}_{new} = \mathbb{P}_{new} + [X_i^{'}, X_{i+1}^{'}]$
    $k_{max} = k_{max} - 1$
    $\mathbb{P} = \mathbb{P}_{new}$
return $X^{best}$
```

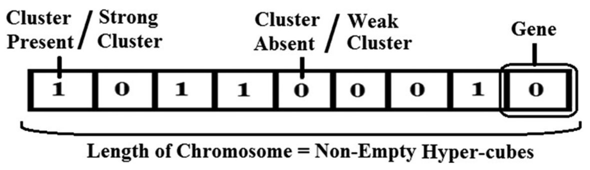

Cluster Present / Strong Cluster
Cluster Absent / Weak Cluster
Gene
1 0 1 1 0 0 0 1 0
Length of Chromosome = Non-Empty Hyper-cubes

Fig. 2. Chromosome representation in D-FRI.

1) Representation of Chromosome and Population: As previously mentioned, the GA-based clustering algorithm utilizes only all the nonempty hypercubes. Thus, the length of a chromosome $|X|$ can be determined by the initial total number of nonempty hypercubes $|\mathbb{H}^*|$, represented as gene series of $0s$ and $1s$. In illustration, the structure of a chromosome can be shown in Fig. 2, where a gene 0 denotes a weak cluster at that location (with the number of hits being lower than a given threshold), and a gene 1 denotes a strong cluster at that location (where the number of hits is higher than the threshold). The threshold is provided by the knowledge engineer to determine how frequently a hypercube is hit may be worth considering being promoted into a future inference rule, namely signalling the potential presence of a cluster. To begin the GA optimization process, the first population $\mathbb{P}$ $[X_1, X_2, ..., X_{|\mathbb{P}|}]$ is generated randomly. A population size of between 20 and 30 chromosomes is commonly used, however, a bigger population size may also be utilize if preferred [54], [55]. Here, a fixed population size $|\mathbb{P}|$ is intuitively determined according to the number of nonempty hypercubes $\mathbb{H}^*$ for the present investigation.

2) Fitness of Chromosomes: In GAs, the fitness function is the most important parameter as it decides the quality and

Algorithm 2: roulette_wheel_selection($\mathbb{P}$).

```
$\mathbb{P} = [X_1, ...., X_{|\mathbb{P}|}]$, population
$X_i, i^{th}$ chromosome of population $\mathbb{P}$
$f(X_i)$, fitness value of $X_i$
$r$ , random number, where $0 \leq r \leq 1$
$threshold = r \times \sum_{i=1}^{|\mathbb{P}|} f(X_i)$
for $\forall i \in \{1, \ldots, |\mathbb{P}|\}$ do
    if $(threshold > 0)$ then
        $threshold = threshold - f(X_i)$
    else
        return $X_i$
```

Algorithm 3: crossover($X_i^{'}$,$X_{i+1}^{'}$, $left$).

```
$X_i = [x_1^i, \ldots, x_r^i, x_{r+1}^i, \ldots, x_{|X|}^i]$
$x_r^i, r^{th}$ gene of $i^{th}$ chromosome $X_i$
$r$ , random integer, where $1 \leq r \leq |\mathbb{P}|$
if $(left = true)$ then
    return $[x_1^i, \ldots, x_r^i] + [x_{r+1}^{i+1}, \ldots, x_{|X_{i+1}'|}^{i+1}]$
else
    return $[x_1^{i+1}, \ldots, x_r^{i+1}] + [x_{r+1}^i, \ldots, x_{|X_i'|}^i]$
```

Algorithm 4: mutate($X_i^{'}$).

```
$r$ , random number, where $0 \leq r \leq 1$
$\delta_m$ , mutation rate
$x_j^i, j^{th}$ gene of $i^{th}$ chromosome $X_i$
for $\forall j \in \{1, \ldots, |X_i^{'}|\}$ do
    if $(r < \delta_m)$ then
        $x_j^i = \neg x_j^i$
return $X_i^{'}$
```

selection criteria for all chromosomes, but it is specified depending upon the given problem. Here, the chromosome denotes a potential cluster arrangement; therefore, the fitness function is used to assess the quality of clusters. Consequently, it is designed to be derived from the DI [14], which is popular to capture and reflect the concepts of cluster isolation and compactness. The greater value of DI indicates better clustering result

$$f(X_i) = \min_{p,q \in \{1,\ldots,i\}, p \neq q} \left\{ \frac{m_{pq}}{\max_{r \in \{1,\ldots,i\}} s_r} \right\} \tag{14}$$

where s_r and m_{pq} are the intracluster (compactness) and intercluster (isolation) distance measurement, respectively

$$s_r = \sqrt{\sum\nolimits_{R' \in C_r} \frac{d(R', \mu_r)^2}{|C_r|}}, \; m_{pq} = d(\mu_p, \mu_q). \tag{15}$$

In the above, C_r is the rth emerging cluster, the distance between a given interpolated rule R' and the centroid μ_q of a cluster C_q is calculated in a way similar to the distance measure $d(R_p, R_q)$ between any two given rules R_p and R_q, with the latter defined by (though alternative distance metrics may be used for the same purpose)

$$d(R_p, R_q) = \sqrt{\sum\nolimits_{i=1}^{N} \frac{(\text{rep}(A_{p,i}) - \text{rep}(A_{q,i}))^2}{\text{range}_{x_i}}} \tag{16}$$

where range$_{x_i}$ stands for the domain range of the variable x_i. Thus,

$$\forall R_j', R_k' \in \mathbb{R}', d(R_j', \mu_q) = d(R_k', \mu_q) \tag{17}$$

where

$$d(R', \mu_q) = \sqrt{\sum\nolimits_{i=1}^{N} (\text{rep}(A_i') - \mu_{q,i})^2}, R' \in \mathbb{R}'. \tag{18}$$

3) Selection of Chromosomes: The standard roulette wheel selection algorithm [56], as depicted in Algorithm 2, is used for the selection of parent chromosomes to produce offsprings of the next round of population. Each chromosome is allocated a portion of roulette wheel based on its fitness value. Inherently, the higher the fitness value, the bigger the portion of the wheel.

4) Crossover and Mutation of Chromosomes: The contribution of the two selected parent chromosomes in the production of offsprings is governed by the two core genetic operators: *crossover* and *mutation*. The crossover process governs the way information is exchanged between two parent chromosomes for the production of the two offsprings, which is depicted in Algorithm 3. The commonly used crossover rate δ_c in GA optimization is about $70 - 95\%$ [54]. The mutation process decides how different offsprings can be produced while avoiding the possibility of the same offsprings in the next round, which is depicted in Algorithm 4. It is desirable to set a very low value of the mutation operator because a high mutation rate can adversely affect GA's performance [54].

5) Termination of Optimization Process: The termination of a GA's operation process is generally dependent on the given problem. Without domain-specific information, in this work, the termination condition is set to a prescribed maximum number of generations $k_{\max}$ in the reproduction of offsprings. Once it is terminated, the best chromosome X^{best} of the final population is considered as the final clustering outcome.

6) Merging and Filtering of Clusters/HyberCubes: The GA optimization process returns the best clustering arrangement. This arrangement contains a number of strong and weak hypercubes, where strong hypercubes $H^1 \in \mathbb{H}^1$ are considered candidate cluster centers and weak hypercubes $H^0 \in \mathbb{H}^0$ are to be merged with their closest strong hypercubes. The merging process of the strong and weak hypercubes/clusters is outlined in Algorithm 5, guided by the clustering metric, DI as shown before. This merging process constructs the final clustering outcome. After which, a filtering process is performed to select one or more clusters of rules as the candidate for the next rule promotion process. The selection of the clusters is based on the predefined threshold that was used previously above which a hypercube is deemed dense.

D. Dynamic Rule Promotion

Resulting from the aforementioned partitioning, clustering, merging, and filtering process, a set of informative rules $R' \in C_q \subseteq H^*$ is obtained for further generalization to create new,

Algorithm 5: merge($\mathbb{H}^1$, $\mathbb{H}^0$).

```
$H_i^0 \in \mathbb{H}^0$, $i^{th}$ weak hyper-cube
$H^1_{H_i^0} \in \mathbb{H}^1$, the closest strong hyper-cube to $H_i^0$
$\mu_{H_i^0}$, centroid of hyper-cube $H_i^0$
for $\forall H_i^0 \in \mathbb{H}^0, i \in \{1, ..., |\mathbb{H}^0|\}$ do
    find $H^1_{H_i^0} = \arg\min_{H^1 \in \mathbb{H}^1} |\mu_{H^1} - \mu_{H_i^0}|$
    $H^1_{H_i^0} = H^1_{H_i^0} \cup H_i^0$
return $\mathbb{H}^1$
```

aggregated rules. Without losing generality, denote such a newly created rule as R^*.

To generate an R^*, a weighted aggregation method is employed that calculates the contribution of every candidate rule in the selected cluster with respect to the cluster centroid μ_q. This process is similar to the construction of intermediate rules in T-FRI, where a matrix w_{ij} of the rank $C_q \times (N+1)$ is involved. It reflects the weighting of the antecedent A'_{ij} of an interpolated rule $R'_i \in C_q$ in relation to the jth antecedent A^*_j of R^* such that

$$w_{i,j} = \frac{1}{d(A'_{i,j}, \mu_{q,j})}, i \in \{1, \ldots, |C_q|\}, j \in \{1, \ldots, N\} \tag{19}$$

and similarly, that of B'_i of the interpolated rule to B^*

$$w_{i,N+1} = \frac{1}{d(B'_i, \mu_{q,N+1})}. \tag{20}$$

The weights are then normalized, resulting in

$$w'_{i,j} = \frac{w_{i,j}}{\sum_{i=1}^{|C_q|} w_{i,j}}. \tag{21}$$

With the resultant calculated weights, a new rule R^* is thus, dynamically constructed, such that

$$A^*_j = \sum_{i=1}^{|C_q|} w'_{i,j} A'_{i,j}, j \in \{1, \ldots, N\}, B^* = \sum_{i=1}^{|C_q|} w'_{i,N+1} B'_i.$$

Once constructed, each new rule R^* is added to the original (sparse) rule base so that $\mathbb{R} := \mathbb{R} \cup \{R^*\}$, and that the corresponding interpolated rules included in the aggregation process are eliminated from the pool of interpolated rules: $\mathbb{R}' := \mathbb{R}' \setminus C_q$. This process iterates until each selected cluster has been promoted into a rule. That is, when the above procedure of partitioning, clustering, merging, filtering, and promotion will have been applied to all the hypercubes which are dense. For completeness, Algorithm 6 summarizes the entire D-FRI algorithm.

E. Complexity Analysis of D-FRI

The computational complexity of D-FRI can be estimated on the basis of its three core subprocedures:

1) partitioning of antecedent space,
2) clustering of interpolated rules, and
3) dynamic rule promotion.

Algorithm 6: GA-based Dynamic Interpolation($\mathbb{R}$, $\mathbb{R}'$,σ).

```
$\mathbb{R}$, original sparse rule base
$\mathbb{R}'$, interpolated rule base
$R^*$, dynamically generated new rule
$\mathbb{H}$, all partitioned hyper-cubes
$\mathbb{H}^* = \mathbb{H}^1 \cup \mathbb{H}^0$, set of non-empty hyper-cubes
$\mathbb{H}^1$, set of strong hyper-cubes
$\mathbb{H}^0$, set of weak hyper-cubes
$\mathbb{C}$, set of clusters
$C_i$, $i^{th}$ cluster of $\mathbb{C}$
$\sigma$ , threshold for promoting new rules
$\mathbb{H} = partition(\mathbb{R}')$
$\mathbb{H}^* = \{H | H \in \mathbb{H}, |H| \neq 0\}$
$\mathbb{H}^1 = GA(\mathbb{H}^*)$
$\mathbb{C} = merge(\mathbb{H}^1, \mathbb{H}^0)$
for $\forall C_i \in \mathbb{C}$ do
    if $|C_i| > \sigma$ then
        $R^* = aggregate(C_i)$
        $\mathbb{R} = \mathbb{R} \cup \{R^*\}$
        $\mathbb{R}' = \mathbb{R}' \setminus C_i$
```

The complexity of the first subprocedure is

$$O_{\text{partition}} = O(|\mathbb{R}'|N\eta). \tag{22}$$

It is dependent on the total number of rules in the interpolated rule base $|\mathbb{R}'|$, the total number of rule antecedents N, and the total number of possible fuzzy values per antecedent η.

The complexity of the second subprocedure is

$$O_{\text{ga}} = O(|\mathbb{P}|k_{\max}) \cdot O_{\text{fitness}} \tag{23}$$

where

$$O_{\text{fitness}} = O\left(|\mathbb{H}| + |\mathbb{R}'|^2 + \frac{|\mathbb{R}'|^2}{|\mathbb{H}|^2} + |\mathbb{H}|^2\right). \tag{24}$$

This is because the complexity of this procedure is mainly dependent on the maximum number of generations $k_{\max}$, the size of the population $|\mathbb{P}|$, and the complexity of the fitness evaluation O_{fitness}. However, the genetic operators also affect it, although their impact varies depending on how they are employed. For simplicity, the above fitness complexity O_{fitness} takes into consideration of the complexity of these chromosome transformations: $O(|\mathbb{H}|)$, that of the hypercube merging process: $O(|\mathbb{R}'|^2)$, and that of the DI computation.

Finally, the complexity of the last subprocedure (of rule promotion) is

$$O_{\text{promotion}} = O\left(\frac{|\mathbb{R}'|N}{|\mathbb{C}|}\right). \tag{25}$$

It depends upon the total number of clusters $|\mathbb{C}|$ (obtained from the output of GA-based clustering), the total number of interpolated rules $|\mathbb{R}'|$, and the total number of antecedent dimensions N. Putting the above three estimates together, the overall computational complexity of the proposed D-FRI algorithm is: $O_{\text{partition}} + O_{\text{ga}} + O_{\text{promotion}}$.

TABLE I
COMPARATIVE EVALUATION OF FRI AND D-FRI

	$\eta = 4$			$\eta = 5$			$\eta = 6$		
	$\epsilon_{\%dvi}$	$\epsilon_{\%dvt}$	$\epsilon_{\%ivt}$	$\epsilon_{\%dvi}$	$\epsilon_{\%dvt}$	$\epsilon_{\%ivt}$	$\epsilon_{\%dvi}$	$\epsilon_{\%dvt}$	$\epsilon_{\%ivt}$
AVG	2.68	2.24	2.07	2.38	**1.24**	2.45	3.47	**2.06**	3.74
SD	2.77	2.01	3.35	2.70	**1.25**	2.63	3.03	**1.97**	3.73

IV. D-FRI: SIMULATION-BASED EVALUATION

This section presents the experimental simulation of D-FRI for comparative evaluation of its efficacy. Initially, a numerically sampled sparse rule base $\mathbb{R}$ of size 100 is generated using the function of three crisp input variables as given below:

$$y = 1 + \sqrt{x_1} + \frac{1}{x_2} + \frac{1}{\sqrt{x_3^3}}, x_1, x_2, x_3 \in [1, 20]. \tag{26}$$

This is then transformed into a fuzzy sparse rule base by fuzzifying the crisp inputs and their corresponding function outputs. In this transformation, a numerical value a is changed to a fuzzy set A with a support length of 1: $A = (a - 0.5, a, a + 0.5), \mathrm{Rep}(A) = a$. This process delivers a basic nonlinear sparse rule base appropriate for the purpose of this simulation.

To ensure the generality of this experimental investigation, the simulations to be carried out using three different η values: 4, 5, 6. That is, the antecedent variable domains are evenly divided for 3 times, into 3 different numbers of intervals, resulting in $4^3 = 64$, $5^3 = 125$, and $6^3 = 216$, hypercubes, respectively. The GA parameters are set as typically adopted in the literature, to the following values: crossover rate $\delta_c = 0.7$, mutation rate $\delta_m = 0.05$, population size $|\mathbb{P}| = 20$, and maximum generation $k_{\max} = 100$.

A. Comparative Evaluation of FRI and D-FRI

D-FRI is applied on a pool of 500 interpolated rules for three different values of $\eta \in \{4, 5, 6\}$, and 90, 132, and 167 new rules have been dynamically promoted for these three cases, respectively. From this, the representative values of all the consequents of the dynamically promoted rules are calculated in order to compare against the results of conventional (T-FRI) interpolation ($\epsilon_{\%dvi}$), and with the ground truths that are calculated using the base function ($\epsilon_{\%dvt}$). For completeness of analysis, the comparison between conventional interpolation and the ground truths ($e_{\%ivt}$) is also performed. The percentage error $\epsilon_{\%} = \epsilon/\mathrm{range}_y$ is calculated corresponding to the range of the consequent variable. D-FRI is repeated 50 times for each set of the parameter values due to the presence of stochastic elements in the preliminary rule generation and within the GA-based clustering procedure. Table I illustrates the comparative evaluation of FRI and D-FRI on the basis of the averaged value of $\epsilon_{\%}$ and the standard deviation of $\epsilon_{\%}$.

It can be seen from this table that for $\eta = 5$ and 6, D-FRI produces significantly more accurate results than T-FRI, the conventional interpolation. Also, the resulting consequent values of the dynamic rules in D-FRI are closer to the ground truths as compared to the results achievable by T-FRI. In these experiments, the most accurate and stable dynamically promoted rules are generated for the parameter configuration with $\eta = 5$. When the parameter configuration with $\eta = 6$ is used, very good performance is also obtained with more accurate and closer to the ground truth results as compared to the use of T-FRI. Importantly, if these dynamically generated rules using intervals $\eta = 5$ or $\eta = 6$ are added to the original sparse rule base, then, the system will not only improve the overall reasoning accuracy (i.e., the quality of the sparse rule base), but also reduce the interpolation overheads by avoiding the need of interpolations for similar observations in the future.

Note that however, when the parameter setting with a large interval $(\eta = 4)$ is assumed, the approach does not generally produce high quality dynamic rules, but the rules produced are still rather stable (giving the much smaller standard deviation in comparison with the use of the original sparse rules by T-FRI). This is of course expected and indeed, rather obvious since large hypercubes are less likely to form any meaningful clustering arrangement. Nevertheless, one benefit of using GAs for optimizing the clustering process is to start clustering without specifying exact starting conditions given their stochastic properties. Thus, overall, the system is insensitive to such initial settings, as confirmed below.

B. Dynamic Rule Promotion and Rule Base Fulfilment

This second set of simulations is focused on the dynamic rule promotion process of D-FRI and its efficiency. It is again carried out with a different number of interpolation rules 250, 500, and 750, corresponding to different numbers of initial interval settings: $\eta \in \{4, 5, 6\}$, respectively. This examination of the dynamic rule promotion process is to observe the effects of D-FRI running over time and the level of fulfilment of the regions within the original sparse rule base. The experimental process is based on the (correct) presumption that the D-FRI process is running normally (i.e., performing interpolation uninterruptedly).

Fig. 3 shows the number of fulfilled regions $\mathbb{H}^*$ in $\mathbb{R}$, and the number of rules $|\mathbb{R}|$ in relation to the number of iterations. The plots within this figure illustrate the continuous variation in the values of $|\mathbb{R}|$ and $|\mathbb{H}^*|$ throughout the dynamic rule promotion process. It clearly indicates that certain rules which have been given or promoted previously may be removed later due to the new interpolated rules are logged in the subsequent process. This is important to ensure rules are up to date while the rule base is maintained without duplications.

Examining more closely, this gradual and continuous rule promotion process improves the coverage of $\mathbb{R}$ by dynamically

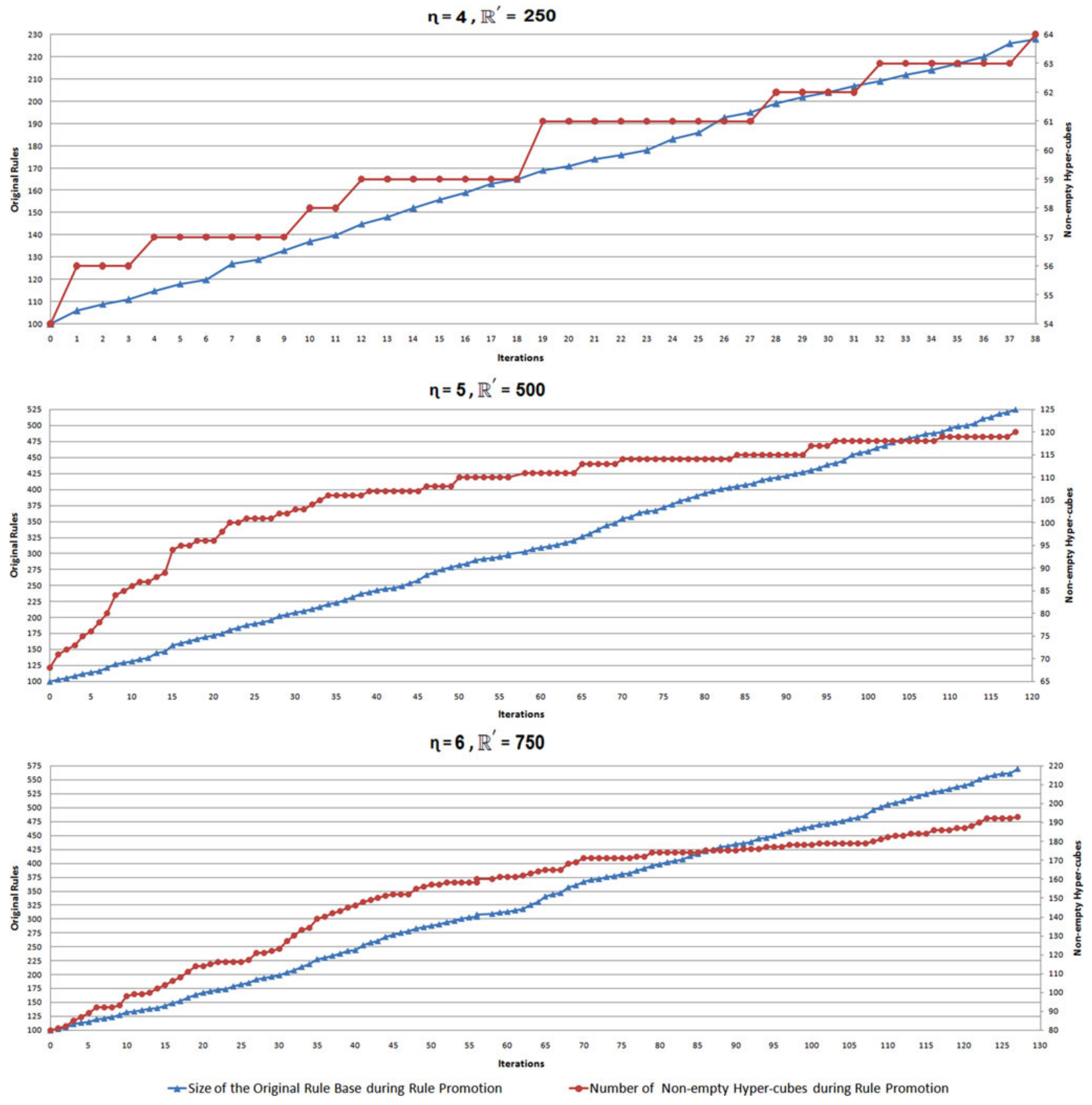


Fig. 3. Dynamic rule promotion and rule base fulfilment in D-FRI.

promoting and adding new rules to it. In particular, for the case of having an initial setting of the interval size $\eta = 4$, all of the potential 64 regions associated with the sparse rule base are filled in 38 iterations, and the final rule base consists of 238 rules. For the cases with $\eta = 5$ and $\eta = 6$, not all but a great majority of the regions are filled: 1) for $\eta = 5$, 120 regions are filled out of 125 regions in 118 iterations, and the final rule base consists of 525 rules; and 2) for $\eta = 6$, 193 regions are filled out of 216 regions in 127 iterations, and the final rule base consists of 570 rules. Thus, while keeping the rules up to date, the rule base is to steadily enriched to provide better coverage of the underlying problem domain, avoiding the overheads otherwise required to perform interpolative computation when a new observation is not matched.

Note that as shown in all three cases above, with continued rule base enrichment through dynamic learning, D-FRI may be employed to generate a dense fuzzy rule base required for classical fuzzy reasoning methods, such as the CRI. Of course, there is a tradeoff to balance between the potential efficiency gained from direct rule firing via CRI with the resulting ever richer rule base and the possible efficiency loss due to searching through an increasingly more complex dense rule base. An interesting way

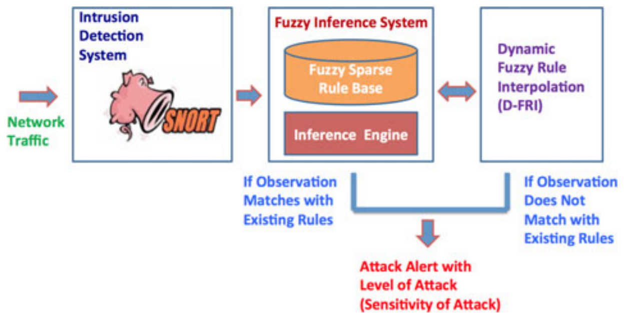

Intrusion Detection System
Network Traffic
SNORT
Fuzzy Inference System
Fuzzy Sparse Rule Base
Inference Engine
Dynamic Fuzzy Rule Interpolation (D-FRI)
If Observation Matches with Existing Rules
If Observation Does Not Match with Existing Rules
Attack Alert with Level of Attack (Sensitivity of Attack)

Fig. 4. Architectural diagram of D-FRI-Snort.

to investigate such a balance is to consider closed-form FRI as with [57], but this is beyond scope of this paper.

V. D-FRI for Intrusion Detection: D-FRI-Snort

To demonstrate the real success of D-FRI, it is herein applied to addressing a challenging real-world problem: network intrusion detection. Security is one of the major concerns of any organization regardless of their size and nature of work. Security attacks and their types are countless, however, network intrusion attack is one of the key concerns, being an illicit attempt that compromises the confidentiality, integrity, or availability of the organizational IT infrastructure [15], [16]. In this application, D-FRI is integrated with Snort to construct an intelligent and dynamic IDS, called D-FRI-Snort. The framework of this integration is shown in Fig. 4, consisting of three main subsystems: an IDS - Snort, a conventional fuzzy inference subsystem, and a D-FRI subsystem.

In implementation, Snort is installed on the host computer, on which the attacks are to be carried out. Prior to actual application of D-FRI, a network baselining process is performed to determine the normal traffic conditions of the host computer and network, including the normal values for: the average packet time, the number of packets sent, and the number of packets received. A number of artificial attacks with various levels of severity are introduced on the host computer for data collection in support of the design of the fuzzy inference system (i.e., fuzzy sets and the original fuzzy rule base). In doing on, the fuzzy inference system offers an extra amount of intelligence to predict the level of potential attacks if it finds a suitable matching rule in the existing rule base, which is not possible by the standard Snort [58]. The D-FRI system provides further information to detect the attack level by running FRI if it does not find a suitable matching rule in the existing rule base. It stores all the interpolated results as a pool of training rules for running GA-based dynamic rule generation and promotion. As such the integrated system is expected to significantly improve its performance over the standard Snort, reducing both false positive detections and false negatives. Detailed designs of this system are presented below.

A. Network Baselining and Background Analysis

In network security analysis, it is a common prerequisite to determine the normal working condition of the network/host, and this process is called network baselining. It relies on the use of a set of metrics used in network performance monitoring to define the normal working conditions of a network infrastructure [59]. These metrics and their values may differ for the similar type of network due to varying network environment and organizational requirements. Snort baselining (and hence, the baselining for D-FRI-Snort) is carried out on the particular host to study its normal traffic conditions before it may detect any malicious attacks.

As indicated previously, in this investigation, Snort captures network traffic containing the following three parameters (though more may be utilized if desired):

1) the average time between received packets by the destination/victim in milliseconds (ATP);
2) the number of packets sent by the source in seconds (NPS); and
3) the number of packets received by the destination/victim in seconds (NPR).

The baseline values of these three parameters are determined on the basis of local proxy network traffic and are set to: ATP $>$ 18 ms, NPS $<$ 270 packets/s, and NPR $<$ 1000 packets/s. They are hereafter regarded as fuzzy input variables and used in the specification of the fuzzy rule base.

D-FRI-Snort is devised to focus on port scan attack (PSA) activities through analyzing collected data. Network scanning tools NMAP [60], Advanced Port Scanner [61], and Ping [62] are used to carry out the PSA, and traffic data is collected by Snort itself. In particular, five rounds of port scanning attack activities are performed on the host machine. In the first round, the PSA on the host computer is carried out by one computer; in the second round, the attack is done by two computers; and so on. Experiment on each round is repeated ten times to acquire the average values of the selected parameters (ATP, NPS, and NPR). The entire experimental activity lasts for one week, during which the final abnormal traffic data for the three chosen parameters are, respectively, recorded as: ATP $\approx$ 4.4–18 ms, NPS $\approx$ 270–1700 packets/s, and NPR $\approx$ 1000–4000 packets/s.

Data analysis is performed mainly using the Wireshark analyser [63] and Snort. In each round, the values of the selected parameters are analyzed in order to understand the attack patterns and statistics. Time is one of the most important factors in analyzing the PSA [58], therefore, the average time between packets is considered as one of the decisive criteria for this investigation. In the first round, when the port scanning activity is very low, ATP shows a very high value, then, it gradually decreases as the number of rounds increases. The other two parameters, NPS and NPR, also show a very low value in the first round each, but they steadily increase their values as the round number increases.

B. Generation of Fuzzy Sets and Fuzzy Rules

From the aforementioned background analysis of collected data, and also following practical advice of domain experts, the values of three parameters ATP, NPS, and NPR can be separated into different ranges. Five ranges are herein set with respect to the five attack rounds, indicating the different levels of a

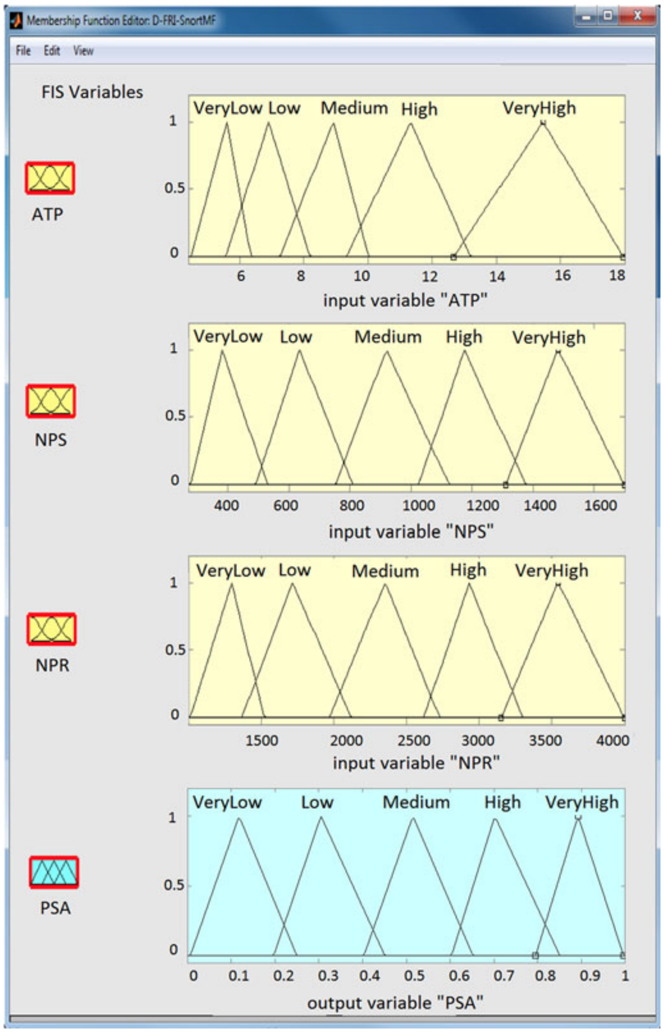
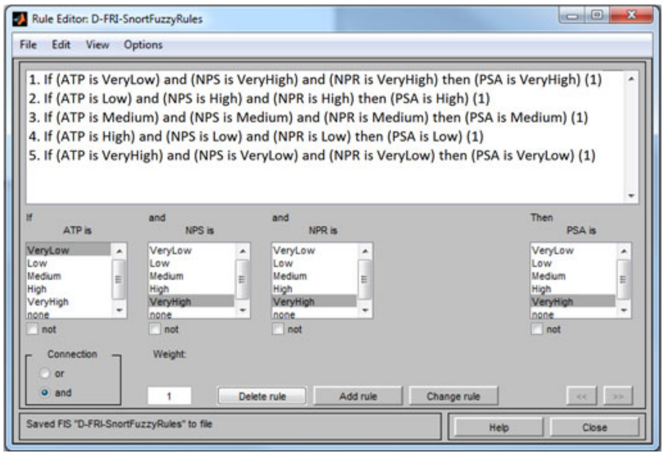


Fig. 5. Fuzzy membership functions used for D-FRI-Snort.



1. If (ATP is VeryLow) and (NPS is VeryHigh) and (NPR is VeryHigh) then (PSA is VeryHigh) (1)
2. If (ATP is Low) and (NPS is High) and (NPR is High) then (PSA is High) (1)
3. If (ATP is Medium) and (NPS is Medium) and (NPR is Medium) then (PSA is Medium) (1)
4. If (ATP is High) and (NPS is Low) and (NPR is Low) then (PSA is Low) (1)
5. If (ATP is VeryHigh) and (NPS is VeryLow) and (NPR is VeryLow) then (PSA is VeryLow) (1)

Saved FIS "D-FRI-SnortFuzzyRules" to file

Fig. 6. Example fuzzy rules used by D-FRI-Snort.

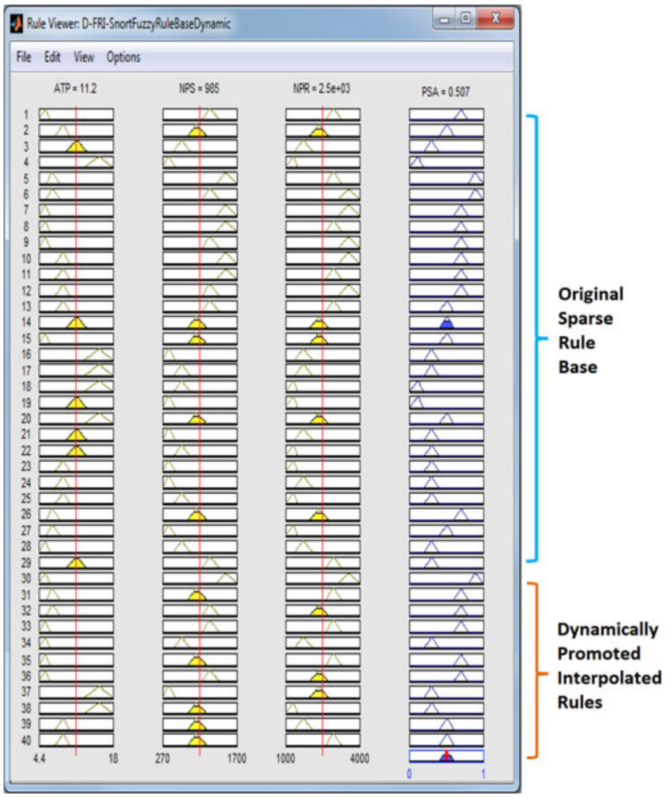

Rule Viewer: D-FRI-SnortFuzzyRuleBaseDynamic
File Edit View Options
ATP = 11.2
NPS = 985
NPR = 2.5e+03
PSA = 0.507
4.4
18
270
1700
1000
4000
0
1
Original Sparse Rule Base
Dynamically Promoted Interpolated Rules

Fig. 7. Original sparse fuzzy rule base and dynamically promoted interpolated rules.

potential attack. The underlying fuzzy sets and fuzzy rules are designed using the fuzzy logic toolbox in Matlab, based on Mamdani's fuzzy inference method [64]. For simplicity, all three variables take values from the same quantity space of normalized fuzzy sets—very low (VL), low (L), medium (M), high (H) and very high (VH), are used whose membership functions as shown in Fig. 5. The output fuzzy variable is the level of PSA, and it is also divided into five fuzzy sets on the scale of 0–1, as given in the same figure.

These fuzzy variables and their corresponding fuzzy sets are employed in the design of original fuzzy rule base. All the rules are generated based on the following artefacts: recorded value ranges; expert knowledge of detecting the PSA; and relationship between the parameters used to detect the underlying attack. Example fuzzy rules are shown in Fig. 6, and the (original) sparse fuzzy rule base, containing 30 rules, is shown in Fig. 7, in terms of the fuzzy sets involved.

C. Fuzzy Inference and D-FRI Operation

With the given sparse rule base, D-FRI-Snort is ready to perform monitoring the network traffic on the host. It generates an alert with an extra level/sensitivity of a potential attack if it finds a suitable matching rule in the existing rule base with the given observation. However, it is challenging to acquire an exhaustive set of rules to cover all the input conditions from the captured traffic data. Fortunately, D-FRI-Snort can generate alerts for those observations that are not covered by the sparse rule base through interpolation, unlike the use of only a conventional fuzzy inference system. This substantially reduces, if not completely resolves, the main failure of the most fuzzy IDSs. Thus, it helps minimize the possibility of generating false results due to a lack of sufficient knowledge.

TABLE II
ATTACK ALERT GENERATION FROM D-FRI-SNORT

Obs.	Input			Output - Attack Alert
No.	*ATP*	*NPS*	*NPR*	*D-FRI-Snort PSA*
1	15.9	322	1206	Very low attack alert
2	10.48	565	1918	Low attack alert
3	7.84	1032	2458	Medium attack alert
4	5.8	1398	3068	High attack alert
5	4.5	1500	3300	Very high attack alert

TABLE III
COMPARISON ON ATTACK ALERT GENERATION

Obs.	Input			Output - Attack Alert	
No.	*ATP*	*NPS*	*NPR*	*Snort PSA*	*D-FRI-Snort PSA*
1	17.78	283	1167	No alert	Very low attack alert
2	11.21	605	1764	No alert	Low attack alert
3	8.03	1105	2506	No alert	Medium attack alert
4	6.57	1317	3068	No alert	High attack alert
5	5.28	1642	3657	No alert	Very high attack alert

Furthermore, D-FRI-Snort accumulates and learn from the interpolated results of repeatedly observed attack conditions, generating new rules to modify the original rule base dynamically. This enables D-FRI-Snort to possess and utilize a most updated (dynamic) fuzzy rule base for traffic monitoring. The dynamically learned rule base allows the system not only to perform inferences with reduced false positives and false negatives, but also to enjoy increased efficiency over time when its coverage is enlarged. This is because the dynamically enriched rule base can better support direct rule firing with improved likelihood of matching an observation than the use of the original sparse rule base.

VI. D-FRI-SNORT: EXPERIMENTAL RESULTS

This experimental investigation covers a wide range of testing and evaluation to confirm the success of D-FRI-Snort (and henceforth, that of D-FRI) in the real world application. Five sets of experiments are discussed here, respectively, on:

1) potential attack conditions upon which D-FRI-Snort generates alerts,
2) comparison between the standard Snort and D-FRI-Snort,
3) effects of dynamically promoting new rules,
4) accuracy of newly promoted rules, and
5) effectiveness of the dynamic rules upon the detection of attack conditions.

A. Attack Alert Generation: D-FRI-Snort

The first set of experiments demonstrate that D-FRI-Snort is capable of monitoring and alerting across a range of attack conditions. A number of PSAs are carried out with different parameter settings. Table II illustrates the five different attack conditions for which D-FRI-Snort has generated attack alerts. In effect, D-FRI-Snort has acted as an anomaly-based IDS in this case, which is a new feature added onto the standard Snort. It is clear from the results that D-FRI-Snort can produce detailed and easy to understand attack alerts.

B. False Negatives and False Positives: D-FRI-Snort Versus Snort

This set of experiments compare the standard Snort and D-FRI-Snort, showing that Snort may miss attack alerts (false negatives) under several conditions, from low to very high risk level of attack, if it merely matches them with its own rules. Since port scanning may be conducted in numerous ways, if a particular form of scanning activity is not explicitly encoded as a *Snort rule*, then Snort can easily miss that type of PSA. Besides, due to the likely erroneous configuration and parameter settings of its inherent *snort.config* file or the nature of its rules, Snort may not be able to detect certain PSAs (such as stealth/slow port scan) or modified signature attacks [65]. This is reflected in Table III, where the standard Snort is shown to have failed in generating attack alerts because it could not find abnormality based on its original rules. However, D-FRI-Snort can act as an anomaly-based IDS, being able to detect attack alerts in all these abnormal conditions. Thus, D-FRI-Snort improves the performance of the standard Snort by helping to reduce its false negatives.

The standard Snort generates alerts based on its given rule base. Thus, in theory, certain false negatives (including above) may be removed by hard-wiring specific rules directly related to the particular types of clandestine PSA, if such knowledge is indeed available. However, there are several downsides of using this Sort of highly specific rules, causing the problems of having increased size of rule base, increased processing power and overheads, increased processing time, and above all, increased false positives in generating attack alert. In extremity this can lead to the generation of continuous attack alerts for every valid activity. Fortunately, D-FRI-Snort offers the use of only a sparse rule base, controlling the size of the default knowledge that Snort possesses. If it fails to find any matching rule it can dynamically promote a new rule for that situation. This helps Snort in reducing false positives based on enumerating abnormal behaviors of the host/network while maintaining a reasonable size of its rule base.

C. Dynamic Rule Promotion and Enrichment of Rule Base

The third set of experiments show the effects of dynamically promoting new rules for the enrichment of the D-FRI-Snort rule base. D-FRI-Snort uses FRI to generate alerts if it does not find a suitable matching rule in the existing rule base. These interpolated rules are accumulated to conduct GA-based learning. Here, 200 accumulated interpolated rules are utilized for the purpose of dynamic rule promotion. The other GA parameters are set to the following values: crossover rate $\delta_c = 0.7$, mutation rate $\delta_m = 0.05$, population size $|\mathbb{P}| = 20$, and maximum generation $k_{\max} = 100$. As a result, 10 new rules are promoted

TABLE IV
DYNAMIC RULE PROMOTION ACCURACY

	$\eta = 5$		
	$\epsilon_{\%dvi}$	$\epsilon_{\%dvt}$	$\epsilon_{\%ivt}$
AVG	2.40	**1.31**	2.56
SD	2.72	**1.32**	2.68

TABLE V
ATTACK ALERT GENERATION AFTER RULE PROMOTION

Obs.	Input			Attack Alert
No.	*ATP*	*NPS*	*NPR*	*D-FRI-Snort PSA*
1	6.95	1267	2385	High attack alert
2	5.23	643	1875	Low attack alert
3	4.61	996	3010	High attack alert
4	7.91	1005	2805	Medium attack alert
5	15.64	310	2266	Low attack alert

into the original sparse rule base, significantly improving its coverage as shown in Fig. 7.

D. Accuracy of Dynamically Promoted Rules

This set of experiments evaluate the accuracy of the newly promoted rules in comparison with that of interpolated rules ($\epsilon_{\%dvi}$) and also, that of the corresponding ground truth rules ($\epsilon_{\%dvt}$). Interpolation results are also compared with the ground truths ($e_{\%ivt}$). The percentage error $\epsilon_{\%} = \epsilon/\text{range}_y$ is computed with respect to the consequent variable. Table IV lists the outcomes of average and standard deviation comparisons, once again confirming that D-FRI leads to more accurate results than the use of T-FRI. Also, the consequent values obtained from firing the dynamic rules are closer to the ground truths.

E. Attack Alert Generation After Dynamic Rule Promotion

The fifth and final set of experiments test the effectiveness of the newly promoted 10 dynamic rules under different potential attack conditions. As summarized in Table V, D-FRI-Snort successfully generates five attack alerts owing to the newly created rules, which were not possible without D-FRI. Importantly, this time, D-FRI-Snort has generated these attack alerts based on firing the (enriched) rules using the classical fuzzy inference without recarrying out interpolative reasoning. Of course, as previously mentioned, D-FRI-Snort can also generate attack alerts in the absence of any matching rule using fuzzy interpolation and gradually improves its sparse rule base using the interpolated results.

VII. CONCLUSION

This paper has presented a D-FRI approach for designing a dynamic rule-based fuzzy system and its application to network security analysis, building an intelligent dynamic IDS. D-FRI is used to select, combine, and promote informative, frequently used interpolated rules into an existing sparse rule base. Systematic experimental results have shown that D-FRI can achieve higher accuracy and robustness than those achievable by the use of conventional FRI. While developing a dynamically enriched rule base, D-FRI helps significantly reduce the interpolation overheads for frequent and similar observations once they have been dealt with, by directly employing the CRI. In conjunction with Snort, one of the most popular open source software systems for intrusion detection, the resulting D-FRI-Snort system can deliver an extra level of intelligence to accurately predict the level of potential threats, with decreased false positives and false negatives.

D-FRI is a flexible and general approach, with no theoretical restriction over the employment of any particular FRI in performing interpolation nor in the computational mechanisms to implement fitness evaluation and rule promotion. Although the current experimental investigation is based on popularly adopted settings (e.g., using scale and move transformation-based method for FRI), they may be replaced with alternative techniques available in the literature. Exactly how such alternatives may practically affect the run-time performance of the system requires further evaluation.

The present implementation assumes a prespecified threshold for determining strong or weak clustering partition and also, an initial grid definition. It would be very interesting to develop a data-driven approach to construct a fully automated grid decomposition mechanism, in support of dynamic partitioning of the underlying domain. In addition, instead of utilizing GAs to optimize the generalization of the interpolated rules, it may be worthwhile to examine the use of different dynamic rule learning techniques [8]–[10]. Similarly, it would be useful to employ and test other evolutionary algorithms [66], [67] and nature-inspired clustering algorithms [46], [47], [68] for the further enhancement of D-FRI. Furthermore, to improve the accuracy of dynamically promoted rules, rule aggregation methods [69], [70] may be introduced to facilitate more effective combination of the selected rules, be they interpolated or original. Finally, the present work has focused on rule promotion (addition); nonetheless, it is important to also consider the removal of redundant and inconsistent rules, for making D-FRI and D-FRI-Snort more efficient. This is possible because FRI can be used in a reverse manner to simplify a given rule base by replacing a neighborhood of rules with their interpolated one(s).

REFERENCES

[1] D. Dubois and H. Prade, "On fuzzy interpolation," *Int. J. Gen. Syst.*, vol. 28, no. 2/3, pp. 103–114, 1999.

[2] L. Koczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," *Int. J. Approx. Reason.*, vol. 9, no. 3, pp. 197–225, 1993.

[3] L. Koczy and K. Hirota, "Interpolative reasoning with insufficient evidence in sparse fuzzy rule bases," *Inf. Sci.*, vol. 71, no. 1/2, pp. 169–201, 1993.

[4] S. Saga, H. Makino, and J. I. Sasaki, "A method for modelling freehand curves—The fuzzy spline interpolation," *Syst. Comput. Jpn.*, vol. 26, pp. 77–87, 1995.

[5] K. J. Astrom and B. Wittenmark, *Adaptive Control*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1995.

[6] S. Mohan and S. Bhanot, "Comparative study of some adaptive fuzzy algorithms for manipulator control," *Int. J. Comput. Intell.*, vol. 3, pp. 303–311, 2006.
[7] H. Zhang and Z. Bien, "Adaptive fuzzy control of MIMO nonlinear systems," *Fuzzy Sets Syst.*, vol. 115, no. 1, pp. 191–204, 2000.
[8] S. Wu and M. Joo, "A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 4, pp. 578–594, Aug. 2001.
[9] P. Angelov and R. Buswell, "Automatic generation of fuzzy rule-based models from data by genetic algorithms," *Inf. Sci.*, vol. 150, no. 1/2, pp. 17–31, 2003.
[10] P. Angelov, "An evolutionary approach to fuzzy rule-based model synthesis using rules indices," *Fuzzy Sets Syst.*, vol. 137, no. 3, pp. 325–338, 2003.
[11] N. Naik, R. Diao, C. Quek, and Q. Shen, "Towards dynamic fuzzy rule interpolation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2013, pp. 1–7.
[12] N. Naik, R. Diao, and Q. Shen, "Genetic algorithm-aided dynamic fuzzy rule interpolation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2014, pp. 2198–2205.
[13] N. Naik, R. Diao, and Q. Shen, "Choice of effective fitness functions for genetic algorithm-aided dynamic fuzzy rule interpolation," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2015, pp. 1–8.
[14] J. C. Dunn, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," *J. Cybern.*, vol. 3, no. 3, pp. 32–57, 1973.
[15] N. Naik, "Fuzzy inference based intrusion detection system: FI-Snort," in *Proc. IEEE Int. Conf. Dependable, Autonomic Secure Comput.*, 2015, pp. 2062–2067.
[16] N. Naik, R. Diao, and Q. Shen, "Application of dynamic fuzzy rule interpolation for intrusion detection: D-FRI-Snort," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2016, pp. 78–85.
[17] Z. Huang and Q. Shen, "Fuzzy interpolative reasoning via scale and move transformations," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 2, pp. 340–359, 2006.
[18] Z. Huang and Q. Shen, "Fuzzy interpolation and extrapolation: A practical approach," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 13–28, 2008.
[19] G. Vass, L. Kalmoar, and L. T. Koczy, "Extension of the fuzzy rule interpolation method," in *Proc. Int. Conf. Fuzzy Sets Theory Appl.*, 1992, pp. 1–6.
[20] S. Kovacs and L. T. Koczy, "Approximate fuzzy reasoning based on interpolation in the vague environment of the fuzzy rulebase as a practical alternative of the classical CRI," in *Proc. 7th Int. Fuzzy Syst. Assoc. World Congr.*, 1997, pp. 144–149.
[21] P. Baranyi, D. Tikk, Y. Yam, L. T. Koczy, and L. Nadai, "A new method for avoiding abnormal conclusion for α-cut based rule interpolation," in *Proc. IEEE Int. Fuzzy Syst. Conf. Proc.*, vol. 1, Aug. 22–25, 1999, pp. 383–388.
[22] D. Tikk and P. Baranyi, "Comprehensive analysis of a new fuzzy rule interpolation method," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 3, pp. 281–296, Jun. 2000.
[23] Y. Yam and L. T. Koczy, "Representing membership functions as points in high-dimensional spaces for fuzzy interpolation and extrapolation," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 6, pp. 761–772, Dec. 2000.
[24] T. D. Gedeon and L. T. Koczy, "Conservation of fuzziness in rule interpolation," in *Proc. Symp. New Trends Control Large Scale Syst.*, 1996, pp. 13–19.
[25] K. W. Wong, T. D. Gedeon, and D. Tikk, "An improved multidimensional α-cut based fuzzy interpolation technique," in *Proc. Int. Conf. Artif. Intell. Sci. Technol.*, 2000, pp. 29–32.
[26] W. Hsiao, S. Chen, and C. Lee, "A new interpolative reasoning method in sparse rule-based systems," *Fuzzy Sets Syst.*, vol. 93, no. 1, pp. 17–22, 1998.
[27] S. Yan, M. Mizumoto, and W. Z. Qiao, "An improvement to Koczy and Hirota's interpolative reasoning in sparse fuzzy rule bases," *Int. J. Approx. Reason.*, vol. 15, pp. 185–201, 1996.
[28] P. Baranyi, L. T. Koczy, and T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 6, pp. 820–837, Dec. 2004.
[29] B. Bouchon-Meunier, C. Marsala, and M. Rifqi, "Interpolative reasoning based on graduality," in *IEEE Int. Conf. Fuzzy Syst.*, 2000, pp. 483–487.
[30] S. Jenei, E. P. Klement, and R. Konzel, "Interpolation and extrapolation of fuzzy quantities—(II) The multiple-dimensional case," *Soft Comput.*, vol. 6, pp. 258–270, 2002.
[31] Y. K. Ko and S. M. Chen, "Fuzzy interpolative reasoning via cutting and transformations techniques," in *Proc. Int. Conf. Ind., Eng., Other Appl. Appl. Intell. Syst.*, 2007, pp. 238–249.
[32] M. F. Kawaguchi and M. Miyakoshi, "A fuzzy rule interpolation technique based on bi-spline in multiple input systems," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2000, pp. 488–492.
[33] Z. C. Johanyak and S. Kovacs, "Fuzzy rule interpolation based on polar cuts," *Comput. Intell., Theory Appl.*, vol. 4, pp. 499–511, 2006.
[34] L. W. Lee and S. M. Chen, "Fuzzy interpolative reasoning for sparse fuzzy rule-based systems based on the ranking values of fuzzy sets," *Expert Syst. Appl.*, vol. 35, no. 3, pp. 850–886, 2008.
[35] Z. C. Johanyak and S. Kovacs, "Fuzzy rule interpolation by the least squares method," in *Proc. Int. Symp. Hung. Researchers Comput. Intell.*, 2006, pp. 495–506.
[36] S. H. Cheng, S. M. Chen, and C. L. Chen, "Weighted fuzzy interpolative reasoning for sparse fuzzy rule-based systems based on piecewise fuzzy entropies of fuzzy sets," *Inf. Sci.*, vol. 329, pp. 503–523, 2016.
[37] S. Chen, Y. Chang, and J. Pan, "Fuzzy rules interpolation for sparse fuzzy rule-based systems based on interval type-2 gaussian fuzzy sets and genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 412–425, 2013.
[38] S. M. Chen, S. H. Cheng, and Z. J. Chen, "Fuzzy interpolative reasoning based on the ratio of fuzziness of rough-fuzzy sets," *Inf. Sci.*, vol. 299, pp. 394–411, 2015.
[39] C. Chen, N. MacParthalain, Y. Li, C. Price, C. Quek, and Q. Shen, "Rough-fuzzy rule interpolation," *Inf. Sci.*, vol. 351, pp. 1–17, 2016.
[40] K. Balazs and L. T. Koczy, "Hierarchical-interpolative fuzzy system construction by genetic and bacterial programming algorithms," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 2011, pp. 2116–2122.
[41] L. Yang and Q. Shen, "Adaptive fuzzy interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 6, pp. 1107–1126, Dec. 2011.
[42] L. Yang, F. Chao, and Q. Shen, "Generalised adaptive fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 4, pp. 839–853, Aug. 2017.
[43] S. M. Chen and Z. J. Chen, "Adaptive fuzzy interpolation based on ranking values of polygonal fuzzy sets and similarity measures between polygonal fuzzy sets," *Inf. Sci.*, vol. 342, pp. 176–190, 2016.
[44] S. Jin, R. Diao, C. Quek, and Q. Shen, "Backward fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 6, pp. 1682–1698, Dec. 2014.
[45] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: Univ. Michigan Press, 1975.
[46] B. B. Chaudhuri and G. Garai, "Grid clustering with genetic algorithm and tabu search process," *J. Pattern Recognit. Res.*, vol. 4, no. 1, pp. 152–168, 2009.
[47] H. L. Lin, F. W. Yang, and Y. T. Kao, "An efficient ga-based clustering technique," *Tamkang J. Sci. Eng.*, vol. 8, no. 2, pp. 113–122, 2005.
[48] J. Barnes, Intrusion detection system, 2009. [Online]. Available: http://www.brainia.com/essays/Intrusion-Detection-System/22016.html
[49] M. Biswanath, L. H. Todd, and N. Karl, "Network intrusion detection," *IEEE Netw.*, vol. 8, no. 3, pp. 26–41, May/Jun. 1994.
[50] B. Claypool, Stealth port scanning methods, 2002. [Online]. Available: http://www.giac.org/paper/gsec/1985/stealth-port-scanning-methods/103446
[51] N. Horn, Snort (2.9.5.6), 2014. [Online]. Available: http://slackbuilds.org/repository/14.1/network/snort/
[52] M. Rouse, Snort, 2014. [Online]. Available: http://searchmidmarketsecurity.techtarget.com/definition/Snort
[53] M. Roesch, Snort, 2014. [Online]. Available: https://www.snort.org/
[54] M. Younes, M. Rahli, and L. A. Koridak, "Economic power dispatch using evolutionary algorithm," *J. Electr. Eng.*, vol. 57, no. 4, pp. 211–217, 2006.
[55] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA, USA: Addison-Wesley, 1989.
[56] R. Kumar and Jyotishree, "Economic power dispatch using evolutionary algorithm," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 4, pp. 365–370, 2006.
[57] L. Yang and Q. Shen, "Closed form fuzzy interpolation," *Fuzzy Sets Syst.*, vol. 225, pp. 1–22, 2013.
[58] W. El-Hajj, H. Hajj, Z. Trabelsi, and F. Aloul, "Updating snort with a customized controller to thwart port scanning," *Secur. Commun. Netw.*, vol. 4, no. 8, pp. 807–814, 2010.
[59] M. Brandenburg, How to set a network performance baseline for network monitoring, 2010. [Online]. Available: http://searchnetworking.techtarget.com/How-to-set-a-network-performance-baseline-for-network-monitoring
[60] Nmap, Chapter 15: Nmap Reference Guide, 2014. [Online]. Available: http://nmap.org/book/man.html
[61] Famatech, Advanced Port Scanner, 2016. [Online]. Available: http://www.advanced-port-scanner.com/
[62] Microsoft, Ping. [Online], 2016. Available: https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ping.mspx?mfr=true

[63] J. Bone, Wireshark, 2011. [Online]. Available: http://download.cnet.com/Wireshark/3000–2085_4-10668290.html

[64] E. H. Mamdani and S. Assilina, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.

[65] M. Richard, Intrusion detection FAQ: Are there limitations of intrusion signatures?, 2001. [Online]. Available: http://www.sans.org/security-resources/idfaq/limitations.php

[66] M. Drobics and J. Botzheim, "Optimization of fuzzy rule sets using a bacterial evolutionary algorithm," *Mathware Soft Comput.*, vol. 15, no. 1, pp. 21–40, 2008.

[67] Z. C. Johanyak and P. G. Ailer, "Particle swarm optimization based tuning for fuzzy cruise control," in *Proc. IEEE 15th Int. Symp. Comput. Intell. Informat.*, 2014, pp. 21–26.

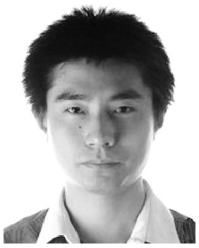
[68] B. Gabrys and A. Bargiela, "General fuzzy min-max neural network for clustering and classification," *IEEE Trans. Neural Netw.*, vol. 11, no. 3, pp. 769–783, May 2000.

[69] T. Boongoen and Q. Shen, "Nearest-neighbor guided evaluation of data reliability and its applications," *IEEE Trans. Syst., Man, Cybern., Part B (Cybern.)*, vol. 40, no. 6, pp. 1622–1633, Dec. 2010.

[70] P. Su, C. Shang, T. Chen, and Q. Shen, "Exploiting data reliability and fuzzy clustering for journal ranking," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 5, pp. 1306–1319, Oct. 2017.

Nitin Naik received the B.Sc. degree in electronics from Devi Ahilya University, Indore, India in 1994, the Polytechnic in electrical engineering from MP Technical Board, Bhopal, India in 1994, the M.Sc. degree in electronics and communications from Devi Ahilya University, Indore, India in 1997, the MSW degree in social work from Devi Ahilya University, Indore, India in 2000, the M.Tech. degree in computer science from Devi Ahilya University, Indore, India in 2006, the MBA degree in finance from Birmingham City University, Birmingham, U.K., in 2009, the PGCTHE in higher education from Aberystwyth University, Aberystwyth, U.K., in 2014, and the Ph.D. degree in computer science from Aberystwyth University, Aberystwyth, U.K., in 2015.

He is currently an Associate Professor and the Head with Cyber Security & Big Data, IT Wing, Defence School of Communications and Information Systems at the Ministry of Defence, UK. He has authored approximately 40 peer-reviewed papers in the areas of artificial intelligence, systems security, big data, cloud computing, and internet of things.



Ren Diao received the B.A. and M.A. degrees in computer science from the University of Cambridge, Cambridge, U.K., in 2007 and 2011, respectively, and the Ph.D. degree from Aberystwyth University, Aberystwyth, U.K., in 2013.

He is currently the Director of Data Science with Wandera, Ltd., London, U.K. By leveraging a mixture of machine learning algorithms and big data technologies, he, together with a group of scientists and engineers, designs and builds innovative products in the mobile security domain. He has authored approximately 30 peer-reviewed papers in the areas of fuzzy systems, nature-inspired metaheuristics, and machine learning.

Qiang Shen received the Ph.D. degree in computing and electrical engineering from Heriot-Watt University, Edinburgh, U.K., in 1990 and the D.Sc. degree in computational intelligence from Aberystwyth University, Aberystwyth, U.K., in 2013.

He is appointed as the Chair of computer science and the Director of the Institute of Mathematics, Physics and Computer Science, Aberystwyth University. He has authored two research monographs and over 360 peer-reviewed papers.

Prof. Shen was recipient of an Outstanding Transactions Paper Award from the IEEE. He is a long-serving Associate Editor for the IEEE TRANSACTIONS ON CYBERNETICS and the IEEE TRANSACTIONS ON FUZZY SYSTEMS, among other editorial roles.