Cluster Comput (2018) 21:667–680
https://doi.org/10.1007/s10586-017-0971-8

A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection

Arif Jamal Malik[1] · **Farrukh Aslam Khan**[2,3]

Received: 28 March 2017 / Revised: 25 May 2017 / Accepted: 1 June 2017 / Published online: 4 July 2017

Abstract A major drawback of signature-based intrusion detection systems is the inability to detect novel attacks that do not match the known signatures already stored in the database. Anomaly detection is a kind of intrusion detection in which the activities of a system are monitored and these activities are classified as normal or anomalous based on their expected behavior. Tree-based classifiers have been successfully used to separate the abnormal behavior from the normal one. Tree pruning is a machine learning technique used to minimize the size of a decision tree (DT) in order to reduce the complexity of the classifier and improve its predictive accuracy. In this paper, we attempt to prune a DT using particle swarm optimization (PSO) algorithm and apply it to the network intrusion detection problem. The proposed technique is a hybrid approach in which PSO is used for node pruning and the pruned DT is used for classification of the network intrusions. Both single and multi-objective PSO algorithms are used in the proposed approach. The experiments are carried out on the well-known KDD99Cup dataset. This dataset has been widely used as a benchmark dataset for network intrusion detection problems. The results of the proposed technique are compared to the other state-of-the-art classifiers and it is observed that the proposed technique performs better than the other classifiers in terms of intrusion detection rate, false positive rate, accuracy, and precision.

✉ Farrukh Aslam Khan
fakhan@ksu.edu.sa

[1] Department of Software Engineering, Foundation University, Defense Avenue, Phase-I, DHA, Islamabad, Pakistan

[2] King Saud University, Riyadh 11653, Saudi Arabia

[3] National University of Computer and Emerging Sciences, Islamabad 44000, Pakistan

Keywords Intrusion detection · Decision tree · Tree pruning · Particle swarm optimization

1 Introduction

Over the past few years, Internet technologies have grown up to a large extent and have presented tremendous increase in the exchange of information online. Internet has indeed become a public platform for communication and delivery of information. Due to this growth, the attacks over the Internet are growing more rapidly and have seriously threatened our networks. It includes attempting to destabilize a network by making machines too busy, misuse of software, and unauthorized access to files and privileges. Intrusion detection is a technology that intelligently monitors events occurring in a computer system or a network and analyzes them for any sign of violation of the security policies. The goal of an intrusion detection system (IDS) is to ensure the availability, integrity, and confidentiality of a network information system. Network intrusion detection can be considered as a classification problem, where the basic goal of a classifier is to separate anomalous network traffic from the normal one. Intrusion detection techniques fall into two main categories: misuse detection and anomaly detection. In misuse detection, the IDS analyzes the information it gathers and compares it with large databases of attack signatures. When a novel attack appears whose signatures are not identified by the system, it is treated as a normal traffic, while in anomaly detection; there is a clear boundary between intrusive and normal traffic. Whenever a network connection deviates from the normal behavior, it is considered as abnormal or anomalous network connection. Anomaly detection systems can detect novel attacks but have a high false positive rate, whereas a misuse detection system cannot detect new attacks.

A decision tree (DT) is a set of nodes that process the given data and return a binary decision (*Yes* or *No*) on the basis of the decision made by the condition associated with that node. Based on the decision made, either the left or the right child of that node is given the input vector and the associated function is applied. This process continues until the leaf node is encountered and the final decision is made as *Yes* or *No*. Among various tree classifiers, a decision tree is appreciated for its simplicity and better classification accuracy. A DT can also be pruned to get more generalized results. Replacing some of the sub-trees of a DT with leaves is known as tree pruning. An un-pruned tree may lead to over-fitting the training data, which may result in poor predictive performance. Pruning is a widely used technique in the field of machine learning. A general observation is that a pruned tree leads to more generalized results as compared to un-pruned trees in case of noisy data. Due to the long history and intense interest in this approach, several surveys on decision trees are available in the literature [1–3].

There are two main approaches towards DT pruning: pre-pruning and post-pruning. DT pruning can be done by following any of these approaches. In pre-pruning, the tree is restricted not to be full grown before it perfectly classifies the training set. One method is to set up a threshold for each sample when arriving at the node; the other is to set up a threshold and restrict each expansion if the system performance is less than the predefined threshold. In pre-pruning, there is no need to grow a full tree. Post-pruning has two stages: fitting and pruning. First, the full decision tree is allowed to be grown by over-fitting the training data. Then, this fully-grown tree is pruned to achieve at least these two goals: (1) to improve the general classification accuracy, and (2) to reduce the tree size. In practice, post-pruning methods have better performance than pre-pruning methods. Post-pruning has been used by a number of researchers in the literature. A minimal cost complexity pruning (MCCP) is presented in [4]. A continuity correlation based binomial distribution algorithm, namely pessimistic error pruning (PEP), is presented by Quinlan in [5], which provides more realistic error rate instead of the optimistic error rate in the training set. Reduced error pruning (REP) technique that finds the smallest version of the most accurate sub-tree is presented in [6], but this technique actually over-prunes the tree. A recent approach called cost and structural complexity (CSC) pruning is presented in [7] that takes care of both the classification accuracy and the structural complexity of the resulting tree.

This paper attempts to address the problem of network intrusion detection using a combination of soft computing techniques. The proposed technique is a hybrid approach in which binary particle swarm optimization (PSO) algorithm is used for the decision tree pruning, and the pruned decision tree is then used for classification of the network intrusions. The paper mainly focuses on the classification improvement on the KDD99Cup dataset, because a well-pruned decision tree results in improved classification accuracy and reduced training time. In this work, we use post-pruning technique for the network intrusion detection problem. We initially let the tree to be fully-grown and then an evolutionary technique i.e., PSO, is applied to prune this tree with a focus on reduced tree size and improved performance. Both single and multi-objective PSO algorithms are applied and their results are presented in the paper. We also demonstrate the results without tree pruning. We compare our results with seven other state-of-the-art classification techniques. Our results demonstrate the power of the optimized tree-pruning algorithm, where a set of arbitrarily selected nodes in a decision tree results in improved accuracy and reduced false positive rate.

The remainder of the paper is organized as follows: The related work is discussed in Sect. 2. In Sect. 3, single and multi-objective PSO algorithms are presented. Section 4 elaborates the proposed work. Experimental setup along with results and discussion are presented in Sect. 5. Finally, Sect. 6 concludes the paper with possible future directions.

2 Related work

In order to investigate the problem of network intrusion detection, researchers have used various types of methods over the past few years. A classification rule-mining algorithm using artificial immune systems (AIS) and fuzzy systems is presented in [8]. An evolutionary approach using artificial ant clustering and K-PSO clustering to network security is presented in [9]. In [10], the authors proposed an easy and efficient selfish cognitive radio attack detection technique, called COOPON. A lightweight intrusion detection framework, integrated for clustered sensor networks, is presented in [11]. The authors also provided algorithms to minimize the triggered intrusion modules in clustered wireless sensor networks. In [12] and [13], authors presented two approaches for network intrusion detection where single and multi-objective optimization approaches are used for feature selection, and random forests algorithm is used for attacks classification. These approaches have the ability to find a set of arbitrary features that cannot be found otherwise. The set of features found this way results in improved classification accuracy. Different attacks have different connections, as some of the attacks have few network connections such as U2R and R2L; whereas others may have hundreds of network connections such as DoS and U2R [14]. There are different feature values for normal and attack connections in the packet header, and the packet contents can be used as signatures for the intrusion detection. In [15], the KNN classifier is used to classify the suspicious data patterns, which are grouped into clusters to trace the anomaly. In [16], the authors propose an optimal feature selection algorithm based on a local search algorithm by applying k-means clustering algorithm to the training data set to measure the goodness of a feature subset as a cost function. The performance of the proposed algorithm is evaluated by performing comparisons with a feature set containing 41 features over the NSL-KDD data set using a multi-layer perceptron neural network. In [17], a diversity-based centroid mechanism is used for the problem of network intrusion detection. In [18], a hybrid technique is developed using multi-objective PSO and Random Forests for the detection of PROBE attacks in a network. Recently, authors in [19] proposed fuzziness based semi-supervised learning approach by utilizing unlabeled samples along with supervised learning algorithm to improve the performance of the classifier for the intrusion detection problem. For this purpose, they used a single hidden layer feed-forward neural network (SLFN) to give a fuzzy membership vector, where the fuzzy quantity is used for the sample categorization on unlabeled samples.

Data mining techniques have been applied for various problems by a number of research projects [20–22]. ADAM [20] and MADAM ID [22] employ association rule mining algorithms. In [21], the authors proposed class-balanced or data-balanced systems of nested dichotomies (ECBND or EDBND). Using C4.5 as a base learner, they show that runtime can be improved especially on problems with many classes without compromising the classification accuracy. In [23], the authors proposed a hybrid classifier based on decision table/naïve Bayesian and investigated a naive Bayesian ranking method by combining naïve Bayes with the induction of decision tables. At each point in the search, the algorithm evaluates the merit of dividing the attributes into two disjoint subsets: one for the decision table, and the other for naïve Bayes. A forward-selection search is used, where at each step, selected attributes are modeled by naïve Bayes, and the remaining are modeled by the decision table. In [24], Jiang et al. proposed a simple, efficient, and effective discriminative parameter learning method, called discriminative frequency estimate (DFE). They aimed to turn the generative parameter learning method i.e., frequency estimate (FE), into a discriminative one by injecting a discriminative element into it. DFE discriminatively computes frequencies from data, and then estimates parameters based on the appropriate frequencies. In [25], Chebrolu et al. proposed ensemble of Bayesian networks (BN) and classification and regression trees (CART) that combines the complementary features of the base classifiers. Their ensemble approach basically exploits the differences in misclassification and improves the overall performance. In [26], the authors introduced PSO for k-nearest neighbors (k-NN) classification by making adjustments to the Euclidean distance formula in the original k-NN classification algorithm and added weight to each feature.

Decision tree is another widely used classification technique that is found effective in disciplines such as data mining, machine learning, and pattern recognition. Decision trees are also implemented for many real-world applications. Although there are a number of other classification techniques available such as genetic algorithm, neural networks, Bayesian belief networks, support vector machines, etc., DT is appreciated for its efficiency, accuracy, simple structure, and wide applicability on real-time problems. Recently, different improvements have been suggested for the DTs in [27–29]. One of the most successful methods used in decision tree construction is pruning. In [30–34], various decision tree pruning methods are compared. The results indicate that some methods (such as cost-complexity pruning, reduced-error pruning etc.) tend to over-prune, whereas other methods (like error-based pruning, pessimistic-error pruning, and minimum-error pruning) bias towards under-pruning. A simple decision tree pruning method known as reduced-error pruning has been suggested by Quinlan in [33]. In this method, each internal node is checked whether replacing it with the most frequent class reduces the tree's accuracy while traversing over the internal nodes from the bottom to the top. This procedure continues until further pruning results in accuracy depletion. Four different methods are described and compared on a test-bed of decision trees from different domains.

In [35], Niblett and Bratko proposed the minimum-error pruning algorithm. It performs bottom-up traversal of the internal nodes. In each node, it compares the probability-error rate estimation with and without pruning. Guaranteeing optimality algorithm, called optimal pruning (OPT), was introduced by Bohanec and Bratko in [36]. They found the optimal pruning based on dynamic programming. An improvement of OPT called OPT-2 was proposed by Almuallim [37], which also performed optimal pruning using dynamic programming. Since the pruned tree is habitually much smaller than the initial tree and the number of internal nodes is smaller than the number of leaves, OPT-2 is usually more efficient than OPT in terms of computational complexity. Rissanen [38], Quinlan and Rivest [39], and Mehta et al. [40] used the minimum description length (MDL) for evaluating the generalized accuracy of a node. In this method, the size of the decision tree is measured by the number of bits required to encode the tree. The decision tree encoded with fewer bits is preferred by MDL.

All the above tree-pruning algorithms have an intrinsic limitation of not analyzing an arbitrary combination of tree nodes (either branch or leaves) while performing the pruning steps. They all follow some kind of linear node pruning methodology (pruning single node at a time) on the basis of some quality measure, which stops further pruning the tree when the performance (in terms of classification accuracy) starts to deteriorate.

In this study, we propose an optimized tree-pruning algorithm to overcome the limitations of already implemented tree pruning algorithms. Instead of using a kind of greedy approach used by the previous tree pruning techniques, the proposed algorithm being an optimization technique, tries the arbitrary combinations of branch nodes and in this way, creates a smaller tree (lesser nodes) with better classification accuracy. The proposed algorithm being a swarm-based optimization technique can be easily parallelized so as to achieve better performance in a reasonable time.

3 Binary particle swarm optimization

PSO was originally developed by Kennedy and Eberhart to solve the real-valued optimization problems. Later on, to extend the real-valued version of PSO to a binary/discrete space, they proposed a binary PSO (BPSO) method [41], because many optimization problems are discrete in nature and have qualitative distinctions between variables and levels of variables.

In BPSO, the position of each particle is represented by Xp = {Xp1, Xp2, ..., Xpn} (where n is the number of particles) in the binary string form and is randomly generated; the bit values 0 and 1 represent a non-selected and selected feature, respectively. The velocity of each particle is represented by Vp = {Vp1, Vp2, ..., Vpn}. The initial velocities in the particles are probability limited to a range of {0.0∼1.0}. In BPSO, once the adaptive values *pbest* and *gbest* are obtained, the features of the respective particles can be tracked with regard to their position and velocity. Each particle is updated according to the following equations:

$$V_{pd}^{new} = w * V_{pd}^{old} + c1 * rand1(pbest_{pd} - X_{pd}^{old}) \\ + c2 * rand2(gbest_{pd} - X_{pd}^{old}) \quad (1)$$

$$if\, V_{pd}^{new} \notin (V_{min}, V_{max})\, then\, V_{pd}^{new} \\ = max\left(\min\left(V_{max}, V_{pd}^{new}\right), V_{min}\right) \quad (2)$$

$$S(V_{pd}^{new}) = 1/(1 + e^{-V_{pd}^{new}}) \quad (3)$$

$$if(rand < S(V_{pd}^{new}))then(X_{pd}^{new} = 1)else(X_{pd}^{new} = 0) \quad (4)$$

Equation (1) is the velocity update equation in which new velocity for a particle is calculated by adding three factors; current velocity, local best position of the particle so far, and the global best position of the swarm. In case of $w * V_{pd}^{old}$, the particle's current velocity is multiplied by the inertia weight that is linearly decreasing so as to reduce the particle's velocity over time. In the expression $c1 * rand1(pbest_{pd} - X_{pd}^{old})$, the particle's current position is subtracted from the local best position in order to attract it towards its best ever position. In case of $c2 * rand2(gbest_{pd} - X_{pd}^{old})$, the particle's current position is subtracted from the global best position so as to attract it towards the swarm's best ever position. In Eq. (1), w is the inertia weight; $c1$ and $c2$ are acceleration parameters; and *rand*, *rand1*, and *rand2* are the three independent random numbers between [0, 1]. V_{pd}^{new} and V_{pd}^{old} are the new and old velocities of the particles respectively. X_{pd}^{new} is the particle's updated position; whereas X_{pd}^{old} is the particle's old position. In Eq. (2), particle velocities of each dimension are tried to a maximum velocity *Vmax*. If the sum of accelerations causes the velocity of that dimension to exceed *Vmax*, then the velocity of that dimension is limited to *Vmax*. The updated features are calculated by the function $S\left(V_{pd}^{new}\right)$ in Eq. (3). If $S\left(V_{pd}^{new}\right)$ is larger than a randomly generated number that is within {0.0∼1.0}, then its position value is represented by 1 (which means that this feature is selected as a required feature for the next update). If $S\left(V_{pd}^{new}\right)$ is smaller than a randomly generated number that is within {0.0∼1.0}, then its position value is represented by 0 (which means that this feature is not selected as a required feature for the next update).

3.1 Multi-objective particle swarm optimization

In single-objective PSO, the global best particle is determined easily by selecting the particle with the best position. Eqs. (1), (2), (3), and (4) are used to update a particle's velocity, calculate its bit value using sigmoid function, and update its position respectively. Since multi-objective optimization problems (MOPs) yield not a single optimal solution but a set of Pareto optimal solutions in which one objective cannot be improved without sacrificing the other objectives; therefore, for the practical implementation, one solution must be selected among the set of Pareto optimal solutions. In this case, a compromised solution (one that satisfies different goals to some extent) can be the best candidate for the selection. The Pareto based approach utilizes the concepts of Pareto dominance in determining the set of solutions [42]. Pareto concepts allow for the determination of a set of optimal solutions in MOPs. Since MOPs have a possibly uncountable set of solutions, which when evaluated, produce vectors whose components represent trade-offs in the decision space; a key Pareto concept, Pareto dominance, is defined mathematically as presented in [43].

3.2 Pareto dominance for the minimization problem

A vector u is said to dominate another vector v if and only if u is partially less than v; i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \bigwedge \exists i \in \{1, \ldots, k\}: u_i < v_i$. A set of solutions within the search space whose corresponding objective vector components cannot be improved without sacrificing any objective vector component of any other solution, is said to be a Pareto optimal set. A non-dominated solution is one that performs better than all the other solutions in at least one objective.

4 Proposed technique

In this paper, we present a decision tree pruning technique using PSO algorithm for the network intrusion detection problem. We use a binary particle to select branches of the tree that are not to be pruned. Both the single-objective optimized decision-tree pruning (SO-DTP) and multi-objective optimized decision-tree pruning (MO-DTP) approaches are used.

Initially, a full tree is allowed to be grown, and then PSO is applied to prune this tree. In this technique, a competition among the branch nodes of the tree is established. The root node and the leaves are not part of the competition. The initial swarm of the PSO algorithm is a random binary bit-string of 1's and 0's. The number of 1's and 0's in every particle is equal to the number of branch nodes in the UDT. A separate pruned decision tree is created against every particle in the swarm. The binary value 1 means that the branch node is selected, whereas binary 0 in the bit-string means that the respective branch node will not be selected for the resulting tree. In this way, most of the decision trees created by the particle's population will have some missing branch nodes as compared to the UDT. The tree with missing branch nodes is considered as a pruned tree and is used for the classification of KDD99Cup network intrusions dataset. The intrusion detection rate (IDR) and false positive rate (FPR) are the two objectives that are to be optimized by the PSO algorithm. We propose two approaches i.e., SO-DTP and MO-DTP to optimize single and two objectives respectively.

In case of SO-DTP, IDR is the only objective to be achieved. IDR is defined as the rate at which the classifier correctly classifies the intrusive records. So, higher the IDR, better would the performance of the classifier. MO-DTP algorithm deals with more than one objective function simultaneously. IDR and FPR are the two competing objectives, which we attempt to optimize using MO-DTP algorithm. IDR is the rate of correctly classified attacks whereas FPR is the rate at which attacks are misclassified as normal traffic. These two objectives are inversely proportional to each other in nature and therefore, improvement in the performance of one objective function may result in decline in the performance of the other objective function. However, the MO-DTP algorithm tries to balance both the objectives in a sense that both the objectives are satisfied to some extent and one objective function is not optimized by completely ignoring the other objective function.

4.1 Structure of a decision tree

In case of SO-DTP, we create a single swarm of n particles with the dimensions of each particle equal to the number of branch nodes in the initial decision tree. The initial decision tree is created from the training dataset without pruning. We call this initial decision tree the un-pruned decision tree (UDT), as shown in Fig. 1. In case of MO-DTP, we create two swarms with n particles in each swarm with dimensions of each particle equal to the number of branch nodes in the initial decision tree.

4.2 Tree pruning

Tree pruning can be well understood with the help of an example. Consider an unpruned tree. From top to bottom, we mark every branch node of the tree with a numeric value. We now store these numbers in a list, which we call branch node list (BNL), as shown in Fig. 2.

BNL is a vector, which helps in creating the particles of PSO. The number of branch nodes in BNL decides the length of every particle, which is 36 in our case. Consider a binary particle whose bit string is shown in Fig. 3. The value 1 at any position means that the respective branch node from the BNL is selected, whereas 0 at any position means that the respective node from the BNL is not selected. For example, in this case, 0 in first, fourth, sixth, and seventh position etc., means that the branch nodes 2, 7, 9, 12 and so on, are not selected, as shown in Fig. 4. (Compare Figs. 2, 4).

The structure of a typical decision tree (DT) during the pruning process is shown in Fig. 5, where x represents an attribute; for example, $x23$ represents 23rd attribute in the dataset, which is the root node of this tree.

Unlike other tree pruning techniques, our algorithm does not prune in top-down or bottom-up fashion or prune a single node at a time. Rather, it prunes a set of arbitrary branch nodes at a time and checks whether it can improve the classification accuracy in this way. Every binary particle is initialized randomly based on the number of branch nodes in the BNL. If there are n branch nodes in the BNL, then every particle will have n binary values. The binary value 1 at a certain position in a particle's position vector means that the respective branch node of the BNL is not pruned, whereas 0 means pruning of the respective branch node of the BNL. To evaluate a particle's fitness, a temporary tree is created by copying the UDT but excluding those branch nodes whose respective binary value is 0 in the particle. This temporary tree will have the root node plus those selected branch nodes whose respective binary value is 1 in the particle, plus the leaf nodes associated with selected branch nodes. This pruned decision tree will most probably be having less number of branch nodes as compared to the UDT or at most, they will be equal. The root node and the leaf nodes are not part of the branch

```
Decision tree for classification
 1  if x23<44.5 then node 2 elseif x23>=44.5 then node 3 else attack
 2  if x39<0.225 then node 4 elseif x39>=0.225 then node 5 else normal
 3  if x6<2 then node 6 elseif x6>=2 then node 7 else attack
 4  if x5<44053 then node 8 elseif x5>=44053 then node 9 else normal
 5  class = attack
 6  class = attack
 7  if x3<46 then node 10 elseif x3>=46 then node 11 else normal
 8  if x37<0.47 then node 12 elseif x37>=0.47 then node 13 else normal
 9  if x3<21.5 then node 14 elseif x3>=21.5 then node 15 else attack
10  class = normal
11  class = attack
12  if x8<0.5 then node 16 elseif x8>=0.5 then node 17 else normal
13  if x6<20.5 then node 18 elseif x6>=20.5 then node 19 else attack
14  class = normal
15  class = attack
16  if x35<0.92 then node 20 elseif x35>=0.92 then node 21 else normal
17  class = attack
18  if x40<0.97 then node 22 elseif x40>=0.97 then node 23 else attack
19  class = normal
20  if x10<25 then node 24 elseif x10>=25 then node 25 else normal
21  if x3<48.5 then node 26 elseif x3>=48.5 then node 27 else attack
22  class = attack
23  class = normal
24  if x36<0.995 then node 28 elseif x36>=0.995 then node 29 else normal
25  if x1<17 then node 30 elseif x1>=17 then node 31 else attack
26  class = attack
27  class = normal
28  if x1<35051 then node 32 elseif x1>=35051 then node 33 else normal
29  if x5<333 then node 34 elseif x5>=333 then node 35 else normal
30  class = attack
31  class = normal
32  if x29<0.125 then node 36 elseif x29>=0.125 then node 37 else normal
33  class = attack
34  if x5<24 then node 38 elseif x5>=24 then node 39 else normal
35  if x23<3 then node 40 elseif x23>=3 then node 41 else attack
36  class = attack
37  if x11<0.5 then node 42 elseif x11>=0.5 then node 43 else normal
38  if x4<3.5 then node 44 elseif x4>=3.5 then node 45 else normal
39  if x37<0.115 then node 46 elseif x37>=0.115 then node 47 else normal
40  if x6<163 then node 48 elseif x6>=163 then node 49 else attack
41  class = normal
42  if x17<0.5 then node 50 elseif x17>=0.5 then node 51 else normal
43  class = attack
44  if x3<21.5 then node 52 elseif x3>=21.5 then node 53 else normal
45  class = attack
46  if x38<0.09 then node 54 elseif x38>=0.09 then node 55 else normal
47  class = attack
48  class = attack
49  class = normal
50  if x1<14847.5 then node 56 elseif x1>=14847.5 then node 57 else normal
51  if x33<6 then node 58 elseif x33>=6 then node 59 else normal
52  class = attack
53  class = normal
54  class = normal
55  class = attack
56  if x33<6.5 then node 60 elseif x33>=6.5 then node 61 else normal
57  class = normal
58  class = attack
59  class = normal
60  if x40<0.11 then node 62 elseif x40>=0.11 then node 63 else normal
61  if x36<0.825 then node 64 elseif x36>=0.825 then node 65 else normal
62  if x5<6.5 then node 66 elseif x5>=6.5 then node 67 else normal
63  class = attack
64  if x36<0.315 then node 68 elseif x36>=0.315 then node 69 else normal
65  class = normal
66  class = normal
67  if x5<320.5 then node 70 elseif x5>=320.5 then node 71 else normal
68  class = normal
69  if x34<0.295 then node 72 elseif x34>=0.295 then node 73 else normal
70  class = normal
71  if x5<450.5 then node 74 elseif x5>=450.5 then node 75 else normal
72  class = attack
73  class = normal
74  class = attack
75  class = normal
```

Fig. 1 Textual representation of a completely unpruned decision tree

```
2  3  4  7  8  9  12  13  16  18  20  21  24  25  28  29  32  34  35  37  38
39  40  42  44  46  50  51  56  60  61  62  64  67  69  71
```

Fig. 2 Branch node list of the UDT

```
0  1  1  0  1  0  0  1  0  0  1  1  1  0  1  0  1  1  0  1  0
0  1  1  0  1  0  0  0  1  1  0  0  0  1  0
```

Fig. 3 Position of 1's and 0's in a binary particle

```
3  4  8  13  20  21  24  28  32  34  37  40  42  46  60  61  69
```

Fig. 4 Selected branch nodes

node selection competition. If, by chance, a branch node is not selected (dropped) in the resultant (pruned) tree, then all of its descendent nodes are also dropped. Now this pruned temporary decision tree is used to classify the records. The structure of a final pruned tree is shown in Fig. 6.

The fitness of every particle in the PSO algorithm is calculated by classifying the test dataset using the pruned tree created by that particle. The evaluation is carried out on the basis of IDR in case of SO-DTP, and IDR and FPR in case of MO-DTP approaches. In this way, the evolution of particles continues until a stopping condition is met. To deal with multi-objective optimization problem, we use vector evaluated particle swarm optimization (VEPSO) technique proposed by Parsopoulos and Vrahatis [44] based on vector evaluated genetic algorithm (VEGA) developed by Schaffer [45]. As there are two separate swarms where each swarm tries to optimize a single objective function, there are two global best particles, one from each swarm. Our proposed multi-objective PSO algorithm involves the steps shown in Fig. 7.

We use sigmoid function to calculate the presence of a particular branch node in the resulting tree. If the value of $S(V_{pd}^{new})$ is greater than a randomly generated number between 0 and 1, then it is set to 1, which means that the respective branch node is selected. And if the value of $S(V_{pd}^{new})$ is less than the randomly generated number, then it is set to 0, which means that this particular branch node is not selected. The flowchart of the algorithm is shown in Fig. 8.

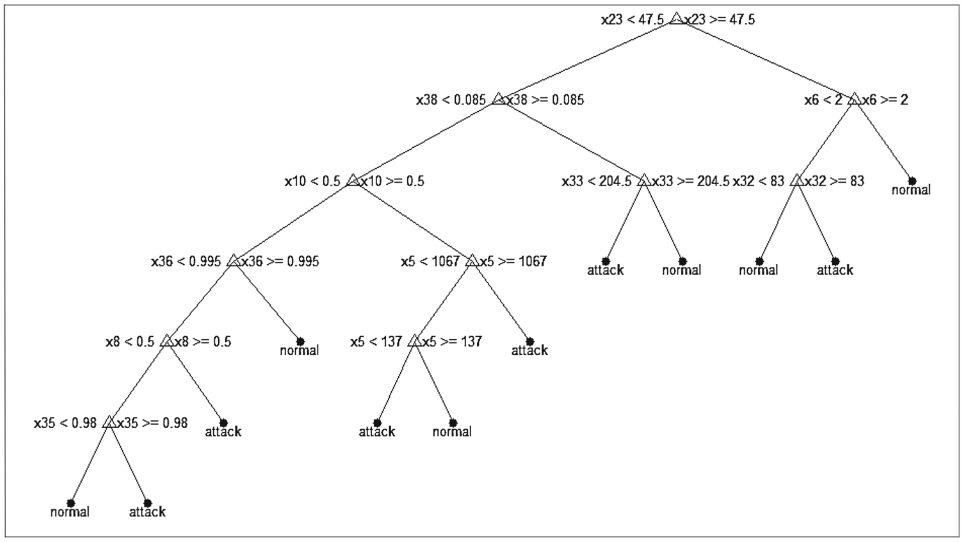
4.3 Fitness function

Two functions are used to evaluate the fitness in the proposed algorithm. In SO-DTP technique, the only objective is IDR, whereas in MO-DTP, the two objectives are IDR and FPR.

Fig. 5 A decision tree in the middle of pruning process **a** Textual representation and **b** Graphical representation

```
1  if x23<47.5 then node 2 elseif x23>=47.5 then node 3 else attack
 2  if x38<0.085 then node 4 elseif x38>=0.085 then node 5 else normal
 3  if x6<2 then node 6 elseif x6>=2 then node 7 else attack
 4  if x10<0.5 then node 8 elseif x10>=0.5 then node 9 else normal
 5  if x33<204.5 then node 10 elseif x33>=204.5 then node 11 else attack
 6  if x32<83 then node 12 elseif x32>=83 then node 13 else attack
 7  class = normal
 8  if x36<0.995 then node 14 elseif x36>=0.995 then node 15 else normal
 9  if x5<1067 then node 16 elseif x5>=1067 then node 17 else attack
10  class = attack
11  class = normal
12  class = normal
13  class = attack
14  if x8<0.5 then node 18 elseif x8>=0.5 then node 19 else normal
15  class = normal
16  if x5<137 then node 20 elseif x5>=137 then node 21 else normal
17  class = attack
18  if x35<0.98 then node 22 elseif x35>=0.98 then node 23 else normal
19  class = attack
20  class = attack
21  class = normal
22  class = normal
```

(a)



(b)

IDR and FPR are calculated according to the assumptions [46], as given in Eqs. (5) and (6) respectively.

$$\text{Intrusion Detection Rate (IDR)} = (\text{TP} / (\text{TP} + \text{FN})) * 100 \tag{5}$$

$$\text{False Positive Rate (FPR)} = (\text{FP} / (\text{FP} + \text{TN})) * 100 \tag{6}$$

where,

True Positive (TP) = truly classified attacks

False Positive (FP) = normal records misclassified as attacks

True Negative (TN) = truly classified normal records

False Negative (FN) = attacks misclassified as normal records

IDR is the number of successfully detected intrusive records from the total number of intrusive records, whereas FPR is the number of misclassified normal records as attacks from the total number of normal records. An intrusion is detected by the classifier whenever a network connection deviates from the normal behavior.

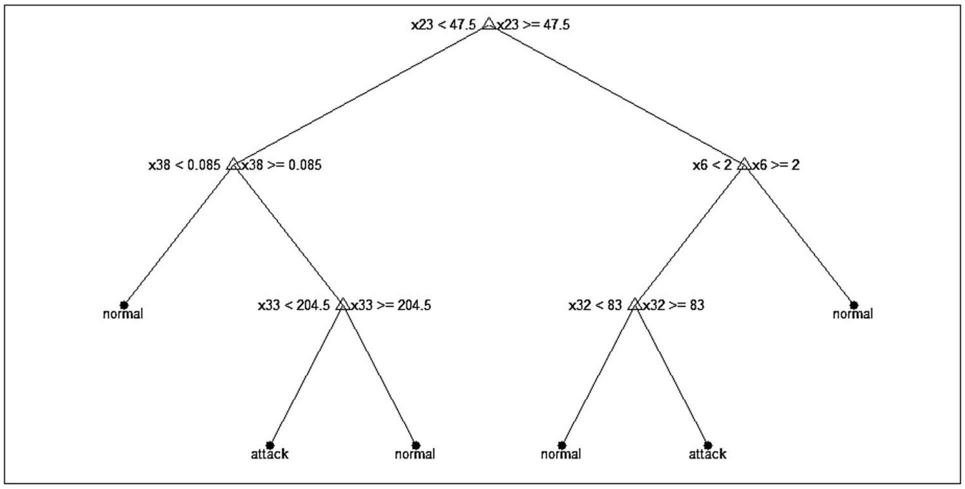
The velocity and position of each particle in the proposed multi-objective PSO algorithm is updated by the following equations:

$$S_1.v_{ij}(t+1) = wS_1.v_{ij}(t) + c_1 r_{1j}(t)(S_1.y_i j(t) - S_1.x_{ij}(t)) + c_2 r_2 j(t)(S_2.\widehat{y}_i(t) - S_1.x_{ij}(t)) \tag{7}$$

Fig. 6 A decision tree after pruning **a** Textual representation and **b** Graphical representation

```
1  if x23<47.5 then node 2 elseif x23>=47.5 then node 3 else attack
 2  if x38<0.085 then node 4 elseif x38>=0.085 then node 5 else normal
 3  if x6<2 then node 6 elseif x6>=2 then node 7 else attack
 4  class = normal
 5  if x33<204.5 then node 8 elseif x33>=204.5 then node 9 else attack
 6  if x32<83 then node 10 elseif x32>=83 then node 11 else attack
 7  class = normal
 8  class = attack
 9  class = normal
10  class = normal
11  class = attack
```

(a)



(b)

Fig. 7 Algorithm of the proposed technique

1. Specify the input parameters and the upper and lower boundaries of each variable.
2. Initialize position and velocity of each particle randomly.
3. Based on the position of 1's in a particle's position vector, select the branch nodes from un-pruned DT to be a part of the resultant DT.
4. Test the pruned DT on test dataset.
5. Update the particle's velocity for particles of swarm 1 and 2 according to the equations 1 and 2 respectively. Also update their position vectors according to equation 4.
6. Update local best and global best values for both swarms.
7. Increment the iteration counter *i* = *i*+1.
8. If stopping criteria met then go to step 9 else go to step 3.
9. Return non-dominated particles along with fitness parameter values.

$$S_2.v_{ij}(t+1) = wS_2.v_{ij}(t) + c_1r_{1j}(t)(S_2.y_{ij}(t) - S_2.x_{ij}(t)) + c_2r_{2j}(t)(S_1.\widehat{y}_j(t) - S_2.x_{ij}(t)) \quad (8)$$

$$S_k(v_{ij}) = 1/(1+e^{-v_{ij}}) \quad (9)$$

$$if(rand < S_k(v_{ij}))then(X_{ij} = 1)else(X_{ij} = 0) \quad (10)$$

We use sigmoid function to calculate the presence of a particular branch node in the resulting tree. If the value of $S(V_{pd}^{new})$ is greater than a randomly generated number between 0 and 1, then it is set to 1, which means that the respective branch node is selected, and if the value of $S(V_{pd}^{new})$ is less than the randomly generated number, then it is set to 0, which means that this particular branch node is not selected. The various PSO parameters are explained in Table 1.

5 Experimental setup and results

For our experiments, the classification results of CBND [47], DTNB [23], DMNBtext [24], CART-BN [25], CART [4], J48 (C4.5) [5], and REPTree [48] are obtained by using Weka [49], which is a data mining tool, whereas the decision tree pruning using single and multi-objective PSO algorithms is

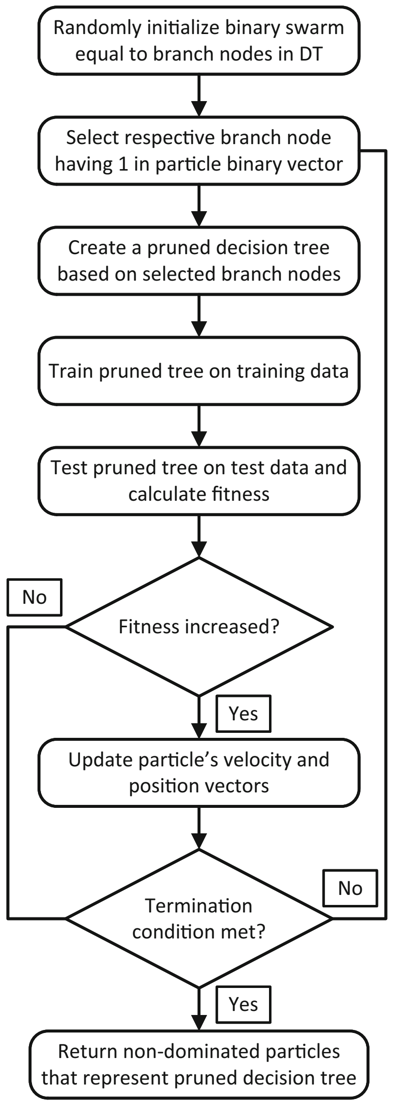

Randomly initialize binary swarm equal to branch nodes in DT
Select respective branch node having 1 in particle binary vector
Create a pruned decision tree based on selected branch nodes
Train pruned tree on training data
Test pruned tree on test data and calculate fitness
Fitness increased?
No
Yes
Update particle's velocity and position vectors
Termination condition met?
No
Yes
Return non-dominated particles that represent pruned decision tree

Fig. 8 Flow chart for decision tree pruning

Table 1 PSO parameters

Parameters	Description
S_1, S_2	Swarm 1 and 2 respectively
S_k	Current swarm 1 or 2
v_{ij}	Velocity of particle i in dimension j
t	Current iteration
w	Inertia weight
c_1, c_2	Cognitive and social acceleration constants for local and global exploration respectively
r_1, r_2	Random numbers between 0 and 1
x_{ij}	Current position of particle i in dimension j
X_{ij}	New position of particle i in dimension j
y_{ij}	Local best position of particle i in dimension j
y_i	Global best position

implemented using MATLAB on 1.73 GHz Core 2 PC. On all optimization runs, the population size is set to 40 for each swarm and maximum number of iterations is set to 100.

5.1 Dataset

The dataset used in our experiments is KDD99Cup 10% labeled dataset. This dataset consists of separate training and test datasets. Training set consists of 494,021 records, whereas test dataset contains 311,029 records. There are 41 attributes and one class label in both the datasets. A sample packet level information is shown in Table 2.

5.2 Preprocessing

Training dataset contains 23 types of records, where 1 is normal and the rest are 22 types of attacks, whereas test dataset contains 38 types of records, where 1 is normal and the rest are 37 types of attacks, as shown in Table 3. Test dataset contains more attacks as compared to the training dataset. The records in both datasets are then assigned to two major classes (Normal and Attack).

5.3 Sampling

Thirty random samples of the datasets are selected from 10% KDD99Cup training and test datasets. The number of records in each training and test datasets are 24000 and 12000 respectively. The distribution of normal records and attacks in each random dataset are shown in Table 4.

5.4 Results

Thirty random datasets are used to evaluate the performance of different tree classifiers. The results are shown in Tables 5 and 6. The results are calculated for the following performance parameters like IDR, FPR, classification accuracy, precision, tree nodes count (tree size), time consumed during classification (in seconds), and cost of classifying attacks.

We used IDR and FPR for the optimization of a decision tree in case of MO-DTP. We compared our results with seven other classification techniques including some state-of-the-art ensemble and tree classifiers that also involve the tree-pruning step. DTNB and DMNBtext being rule-based classifiers, do not create the trees, and therefore, have no node count values, as shown in Tables 5 and 6.

To see the improvement in classification accuracy by involving the pruning step, we also mention the results without involving the tree pruning step, which we call No-Pruning. Here, we discuss the performance of the above techniques on the basis of best and average results separately.

Table 2 Sample packet level information

Serial no.	Data attribute	Data samples		Serial no.	Data attribute	Data samples	
		1	2			1	2
1	duration	0	0	22	is_guest_login	0	0
2	protocol_type	Tcp	Icmp	23	count	6	511
3	service	http	ecr_i	24	srv_count	32	511
4	flag	SF	SF	25	serror_rate	0.00	0.00
5	src_bytes	296	1032	26	srv_serror_rate	0.00	0.00
6	dst_bytes	402	0	27	rerror_rate	0.00	0.00
7	land	0	0	28	srv_rerror_rate	0.00	0.00
8	wrong_fragment	0	0	29	same_srv_rate	1.00	1.00
9	urgent	0	0	30	diff_srv_rate	0.00	0.00
10	hot	0	0	31	srv_diff_host_rate	0.12	0.00
11	num_failed_logins	0	0	32	dst_host_count	26	255
12	logged_in	1	0	33	dst_host_srv_count	255	255
13	num_compromised	0	0	34	dst_host_same_srv_rate	1.00	1.00
14	root_shell	0	0	35	dst_host_diff_srv_rate	0.00	0.00
15	su_attempted	0	0	36	dst_host_same_src_port_rate	0.04	1.00
16	num_root	0	0	37	dst_host_srv_diff_host_rate	0.03	0.00
17	num_file_creations	0	0	38	dst_host_serror_rate	0.00	0.00
18	num_shells	0	0	39	dst_host_srv_serror_rate	0.00	0.00
19	num_access_files	0	0	40	dst_host_rerror_rate	0.00	0.00
20	num_outbound_cmds	0	0	41	dst_host_srv_rerror_rate	0.00	0.00
21	is_host_login	0	0	42	Class	normal	attack

Table 3 Attacks in the dataset

Dataset	Attack type
Training	Back, land, Neptune, pod, smurf, teardrop, Ipsweep, nmap, portsweep, satan, Buffer_overflow, loadmodule, perl, rootkit, F8tp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
Test	Apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm, Ipsweep, mscan, nmap, portsweep, saint, satan, Buffer_overflow, loadmodule, perl, ps, sqlattack, rootkit, xterm, Ftp_write, guess_passwd, httptunnel, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, warezmaster, worm, xlock, xsnoop

5.4.1 Best results

Best results achieved by the different classifiers are shown in Table 5. IDR is the rate at which the attacks are correctly classified. If a classifier achieves a high IDR, it means that it is detecting more attacks correctly. The highest IDR achieved by the SO-DTP is 92.71%, whereas MO-DTP stands at 2nd position with 92.6% IDR. IDR achieved by the non-evolutionary algorithms remain below 92%. Due to the stochastic search capability, both the SO-DTP and MO-DTP algorithms achieve the highest IDR and therefore, the ratio of the difference between evolutionary and non-evolutionary classifiers is almost 1:6.

FPR is the rate at which normal records are misclassified as attacks. Therefore, if an algorithm achieves low FPR, it means that it classifies the normal network traffic more accurately. In our case, MO-DTP algorithm achieves 0.136% FPR, which is the lowest, whereas DTNB (a decision tree and naïve Bayesian hybrid classifier) achieves 2nd lowest FPR, which is 0.2%. The SO-DTP algorithm achieves 0.819% FPR, which is very high as compared to the FPR achieved by MO-DTP algorithm. This is because the SO-DTP algorithm only optimizes the single objective that is IDR, so it completely ignores the FPR. DMNBtext algorithm has good IDR but on the other hand, it has the highest FPR, i.e., 7.6%, which is more than 10 times higher than any other classifier.

Accuracy of an algorithm is calculated by dividing the correctly classified data items with the total data items classified. MO-DTP algorithm achieves the highest classification accuracy that is 96.65%. REPTree algorithm achieves the 2nd highest accuracy, which is 93.36%. Precision is the proportion of the true positives against all the positive results. In our case, it is the amount of correctly classified attacks divided by the total attacks classified. Almost all the classifiers achieved above 99% precision, except DMNBtext classifier, which has

Table 4 Records distribution in training and test datasets

Random datasets	Training dataset		Test dataset	
	Normal records	Attack records	Normal records	Attack records
1	4782	19218	2369	9631
2	4831	19169	2320	9680
3	4606	19394	2274	9726
4	4738	19262	2266	9734
5	4795	19205	2292	9708
6	4637	19363	2337	9663
7	4881	19119	2359	9641
8	4973	19027	2273	9727
9	4692	19308	2361	9639
10	4730	19270	2250	9750
11	5269	18731	2318	9682
12	4662	19338	2276	9724
13	4893	19107	2252	9748
14	4537	19463	2391	9609
15	4705	19295	2365	9635
16	4998	19002	2279	9721
17	4687	19313	2259	9741
18	4721	19279	2386	9614
19	4904	19096	2270	9730
20	4789	19211	2292	9708
21	4738	19262	2384	9616
22	4978	19022	2468	9532
23	4623	19377	2242	9758
24	4840	19160	2388	9612
25	4761	19239	2372	9628
26	5037	18963	2154	9846
27	4874	19126	2337	9663
28	4754	19246	2243	9757
29	4792	19208	2256	9744
30	4680	19320	2487	9513

Table 5 Best performance of the classifiers

Classifier	IDR%	FPR%	Accuracy%	Precision%	Node count	Time (s)	Class. cost
CBND	91.3	0.5	92.87	99.90	29	2.95	0.071
DTNB	91.2	0.2	92.77	99.90	–	568.59	0.072
DMNBtext	91.9	7.6	91.7	98.10	–	**0.15**	0.083
CART-BN	91.7	0.3	93.2	99.90	14	12.32	0.068
J48(C4.5)	91.3	0.5	92.87	99.90	17	2.56	0.0774
CART	91.8	0.3	93.22	99.90	27	9.14	0.0818
REPTree	92	0.5	93.36	99.90	23	0.78	0.0794
No-Pruning	91.81	0.457	93.25	99.88	47	0.702	0.0675
SO-DTP	**92.713**	0.819	91.94	99.89	26	128.93	0.081
MO-DTP	92.6	**0.136**	**96.65**	**99.98**	**8**	383.58	**0.033**

Bold values indicate the best performance (results) by the classifiers

Table 6 Average performance by the classifiers

Classifier	IDR%	FPR%	Accuracy%	Precision%	Node count	Time(s)	Class. cost
CBND	91.02	0.8	92.596	99.78	39.4	3.46	0.074
DTNB	90.6	0.46	92.346	99.88	–	629.75	0.077
DMNBtext	91.46	8.9	91.394	97.73	–	**0.172**	0.086
CART-BN	90.97	0.64	92.594	99.81	31.9	14.672	0.073
J48(C4.5)	91.02	0.8	92.591	99.78	37.8	2.959	0.0741
CART	90.94	0.62	92.572	99.83	35.4	10.265	0.0743
REPTree	91.16	0.77	92.716	99.81	28.8	0.84	0.0728
No-Pruning	91.17	0.821	92.724	99.78	56.2	0.9672	0.073
SO-DTP	**91.765**	2.7429	91.69	99.83	42.1	196.53	0.083
MO-DTP	91.563	**0.201**	**93.53**	**99.957**	**10.8**	408.154	**0.065**

Bold values indicate the best performance (results) by the classifiers

the lowest precision value, i.e., 98.10%. The proposed MO-DTP algorithm is very precise in classifying the attacks with a precision value of 99.98%.

Node count represents the size of the tree. Higher node count means a bigger tree and lower node count value means a smaller tree. The classification time taken by a smaller tree is less than the classification time taken by a bigger tree. The smallest node count of the proposed MO-DTP algorithm is 8 nodes, which is almost half of the least node count value obtained by any other classifier.

The total time taken by a classifier in training and testing the dataset is measured in terms of seconds. DMNBtext, a naïve Bayesian classifier, consumed the shortest time i.e., 0.15 seconds in training and testing the dataset. The proposed (SO-DTP and MO-DTP) algorithms, being evolutionary techniques, took a long time to train and test. The classification cost in our case is the cost associated with classifying the attacks. Less classification cost means the better classifier. The classification cost associated with the proposed MO-DTP approach is least and is 0.033, whereas the 2nd least classification cost that is 0.0675, which is achieved by the decision tree with no-pruning.

From the above experimental results, we conclude that the proposed MO-DTP approach performed best in 5 out of 7 performance measures. Although we did not attempt to optimize all the performance measures, it was the better search capability of the PSO algorithm and the multi-objective approach that took care of more than one objective function simultaneously.

5.4.2 Average results

The average performance of the classifiers is shown in Table 6. On average, all the classifiers achieved above 90% IDR but SO-DTP algorithm achieved the highest IDR i.e., 91.76%, whereas MO-DTP algorithm stands next to the SO-DTP algorithm with 91.56% IDR.

The average FPR achieved by the MO-DTP algorithm is 0.201%, which is the lowest, whereas DTNB classifier stands 2nd by achieving 0.46% average FPR. Two classifiers; SO-DTP and DMNBtext, show high average FPR, i.e., 2.74 and 8.9 respectively. The average classification accuracy of all the classifiers ranges from 91 to 93% but the proposed MO-DTP approach achieved the highest average classification accuracy i.e., 93.53%.

The highest precision on average is achieved by the MO-DTP approach, i.e., 99.95%, whereas rest of the classifiers managed to achieve the precision between 99.78 and 99.88% on average except DMNBtext that has 97.73% average precision.

On average, the MO-DTP algorithm has the least node count value of 10.8, which is almost 3 times less than any other classifier's node count. The decision tree with no pruning grows the full tree and does not prune it; therefore, it has the highest average node count of 56.2. The average classification times (including training and testing) of the DTNB classifier and the proposed (SO-DTP and MO-DTP) classifiers are very high as compared to rest of the classifiers. The classification cost of the MO-DTP algorithm to classify attacks is least on average, which means that the MO-DTP algorithm classifies the normal traffic with highest accuracy. The average performance trend of the proposed MO-DTP algorithm is same as that of the best performance trend. It achieved the highest performance on average for the same parameters for which it achieved the best performance.

6 Conclusion

In this paper, we used the standard particle swarm optimization (PSO) algorithm with single and multi-objective perspectives to prune a decision tree. This pruned decision tree classifier is then used for the detection of anomalous network connections. Ten classifiers from different categories like tree-based, rule-based, evolutionary, and Bayesian are used for this purpose including the proposed approaches. From the results, we conclude that multi-objective optimized decision tree pruning (MO-DTP) approach suits best for minimizing the overall tree size. On average, it reduced the tree size up to three times as compared to any other tree classifier used. In addition to the minimum tree size, the MO-DTP approach also achieved minimum false positive rate (FPR) with lower classification cost, whereas it maximized the intrusion detection rate (IDR), classification accuracy, and the precision. The single-objective optimized decision tree pruning (SO-DTP) approach achieved highest IDR at the expense of the 2nd highest FPR, which means that it has a high rate of misclassification of the normal traffic. The proposed MO-DTP approach did not increase or decrease any objective at the cost of the other objective, whereas it created a balance among the goals to be achieved; therefore, it stood best in most of the performance measures. Because of the arbitrary node selection capability of the proposed approaches, the tree is very well pruned such that it avoided over-fitting and resulted in more generalized trees. By using these generalized trees for classifying attacks, a significant improvement in the performance is observed.

Acknowledgements The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through the Research Group Project no. RGP-214.

References

1. Safavin, S.R., Landgrebe, D.: A survey of decision tree classifier methodology. IEEE Trans. Syst. Man Cybern. **21**(3), 660–674 (1991)
2. Murthy, S.K.: Automatic construction of decision trees from data: a multidisciplinary survey. Data Min. Knowl. Disc. **2**(4), 345–389 (1998)
3. Kohavi, R., Quinlan, J.R.: Decision-tree discovery, In: Handbook of Data Mining and Knowledge Discovery, Klosgen, W., Zytkow, J.M. (eds.),ch. 16.1.3, pp. 267–276. Oxford University Press, London, UK (2002)
4. Breiman, L., Friedman, J., Olshan, R., Stone, C.: Classification and Regression Trees. Wadsworth International, California (1984)
5. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Inc, California (1993)
6. Quinlan, J.R.: Simplifying decision trees. Int. J. Man-Mach. Stud. **27**, 221–234 (1987)
7. Wei, J.M., Wang, S.Q., Yu, G., Gu, L., Wang, G.Y., Yuan, X.J.: A Novel method for pruning decision tree. In: Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12–15 July 2009
8. Alves, R.T., Delgado, M.R.B.S., Lopes, H.S., Freitas, A.A.: An Artificial Immune System for Fuzzy-rule Induction in Data Mining. Lecture Notes in Computer Science, pp. 1011–1020. Springer, Berlin (2004)
9. Srinoy, S., Kurutach, W.: Combination Artificial Ant Clustering and K-PSO Clustering Approach to Network Security Model. In: IEEE International Conference on Hybrid Information Technology (ICHIT'06) (2006)
10. Jo, M., Han, L., Kim, D., In, H.P.: Selfish attacks and detection in cognitive radio ad-hoc networks. IEEE Netw. **27**(3), 46–50 (2013)
11. Hai, T.H., Huh, E.N., Jo, M.: A lightweight intrusion detection framework for wireless sensor networks. Wirel. Commun. Mob. Comput. **10**(4), 559–572 (2010)
12. Malik, A.J., Shahzad, W., Khan, F.A.: Binary PSO and random forests algorithm for PROBE attacks detection in a network. In: IEEE Congress on Evolutionary Computation (CEC 2011), New Orleans, USA, pp. 662–668, 5–8 June 2011
13. Malik, A.J., Shahzad, W., Khan, F.A.: Network intrusion detection using hybrid binary PSO and random forests algorithm. Secur. Commun. Netw. **8**(16), 2646–2660 (2015)
14. Guo, L. et al.: Robust Prediction of Fault-Proneness by Random Forests. In: Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE'04), pp. 417–428, Brittany, France, November (2004)
15. Punithavathani, D.S., Sujatha, K., Jain, J.M.: Surveillance of anomaly and misuse in critical networks to counter insider threats using computational intelligence. Clust. Comput. **18**(1), 435–451 (2015)
16. Kang, S., Kim, K.J.: A feature selection approach to find optimal feature subsets for the network intrusion detection system. Clust. Comput. **19**(1), 325–333 (2016)
17. Gondal, M.S., Malik, A.J., Khan, F.A.: Network Intrusion Detection using Diversity-based Centroid Mechanism. In: 12th International Conference on Information Technology: New Generations (ITNG 2015), Las Vegas, Nevada, USA, 13–15 April 2015
18. Malik, A.J., Khan, F.A.: A Hybrid Technique using Multi-objective Particle Swarm Optimization and Random Forests for PROBE Attacks Detection in a Network. In: IEEE Conference on Systems, Man, and Cybernetics (SMC 2013), Manchester, UK, 13–16 October 2013
19. Ashfaq, R.A.R., Wang, X., Huang, J.Z., Abbas, H., He, Y.: Fuzziness based semi-supervised learning approach for intrusion detection system. Inf. Sci. **378**, 484–497 (2017)
20. Barbarra, D., Couto, J., Jajodia, S., Popyack, L., Wu, N.: ADAM: Detecting Intrusions by Data Mining. In: Proceedings of the 2001 IEEE, Workshop on Information Assurance and Security T1A3 1100 United States Military Academy, West Point, NY, June 2001
21. Random Forests: http://www.stat.berkeley.edu/~breiman/RandomForests/
22. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf. Syst. Secur. **3**(4), 227–261 (2000)
23. Hall, M., Frank, E.: Combining Naive Bayes and Decision Tables. In: Proceedings of Twenty-First International Florida Artificial Intelligence Research Society Conference, AAAI Press, Coconut Grove, Florida, USA , pp. 318–319 15–17 May 2008
24. Su, J., Zhang, H., Ling, C.X., Matwin, S.: Discriminative Parameter Learning for Bayesian Networks. In: Proceedings of the 25th international conference on Machine learning, pp. 1016–1023. New York, USA (2008)
25. Chebrolu, S., Abraham, A., Thomas, J.P.: Feature deduction and ensemble design of intrusion detection systems. Int. J. Comput. Secur. **24**, 295–307 (2005)
26. Wu, Q., Liu, H., Yan, X.: Multi-label classification algorithm research based on swarm intelligence. Clust. Comput. **19**(4), 2075–2085 (2016)
27. Mahmood, A.M., Rao, K.M., Reddi, K.K.: A novel algorithm for scaling up the accuracy of decision trees. Int. J. Comput. Sci. Eng. **2**(2), 126–131 (2010)
28. Jin, C., De-lin, L., Xiang, M.F.: An Improved ID3 Decision Tree Algorithm. In: Proceedings of 4th International Conference on Computer Science & Education, pp. 127–130 (2009)

29. Tsang, S., Kao, B., Yip, K.Y., Ho, W.S., Lee, S.D.: Decision trees for uncertain data. IEEE Trans. Data Eng. **23**(1), 441–444 (2009)
30. Esposito, F., Malerba, D., Semeraro, G.: A comparative analysis of methods for pruning decision trees. IEEE Trans. Pattern Anal. Mach. Intell. **19**(5), 476–492 (1997)
31. Breslow, L.A., Aha, D.W.: Simplifying decision trees: a survey. Knowl. Eng. Rev. **12**(1), 1–40 (1997)
32. Xizhao, W., Ziying, Y.: A brief survey of methods for decision tree simplification. Comput. Eng. Appl. **40**(27), 66–69 (2004)
33. Quinlan, J.R.: Simplifying decision trees. Int. J. Man-Mach. Stud. **27**, 221–234 (1987)
34. Mingers, J.: An empirical comparison of pruning methods for decision tree induction. Mach. Learn. **4**(2), 227–243 (1989)
35. Niblett, T., Bratko, I.: Learning decision rules in noisy domains, in Expert Systems. Cambridge Univ. Press, Cambridge, MA (1986)
36. Bratko, I., Bohanec, M.: Trading accuracy for simplicity in decision trees. Mach. Learn. **15**, 223–250 (1994)
37. Almuallim, H.: An efficient algorithm for optimal pruning of decision trees. Artif. Intell. **83**(2), 347–362 (1996)
38. Rissanen, J.: Stochastic Complexity and Statistical Inquiry. World Scientific, Singapore (1989)
39. Quinlan, J.R., Rivest, R.L.: Inferring decision trees using the minimum description length principle. Inf. Comput. **80**, 227–248 (1989)
40. Mehta, R.L., Rissanen, J., Agrawal, R.: Mdl-based decision tree pruning. In: Proc. 1st Int. Conf. Knowledge Discovery and Data Mining, pp. 216–221 (1995)
41. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. IEEE Int. Conf. Syst. Man Cybern. **5**, 4104–4108 (1997)
42. Fonseca, C.M., Fleming, P.J.: Multiobjective Optimization. In: Evolutionary Computation 2 Advanced Algorithms and Operators, Back, T., Fogel, D.B., Michalewicz, Z. (eds.) 2, pp. 25–37 (2000)
43. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
44. Parsopoulos, K.E., Vrahatis, M.N.: Particle Swarm Optimization Method in Multiobjective Problems. In: Proceedings of the ACM Symposium on Applied Computing, pp. 603–607 (2002)
45. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100 (1985)
46. Sarasama, S.T., Zhu, Q.A., Huff, J.: Hierarchical Kohonen net for anomaly detection in network security. IEEE Trans. Syst. Man Cybern. Part B **35**(2), 302–312 (2005)
47. Dong, L., Frank, E., Kramer, S.: Ensembles of Balanced Nested Dichotomies for Multi-class Problems. In: PKDD, pp. 84–95 (2005)
48. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 3rd edn. Morgan Kaufmann, San Francisco (2011)
49. WEKA Data Mining Software: http://www.cs.waikato.ac.nz/~l/weka/

Arif Jamal Malik is currently working as an Assistant Professor at the Department of Software Engineering, Foundation University, Islamabad, Pakistan. He did his M.S. and Ph.D. in Computer Science from National University of Computer and Emerging Sciences, Islamabad, Pakistan, in 2008 and 2014 respectively. His research interests include Evolutionary Computation, Swarm Intelligence, Network Security, and Machine Learning. He has several publications in prestigious international journals and conferences.

Farrukh Aslam Khan is an Associate Professor at the Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh, Saudi Arabia. He did his M.S. in Computer System Engineering from GIK Institute of Engineering Sciences and Technology, Pakistan, and Ph.D. in Computer Engineering from Jeju National University, South Korea, in 2003 and 2007 respectively. He has also done professional trainings from Massachusetts Institute of Technology (MIT), New York University, IBM, and other professional organizations. He has over 70 publications in refereed international journals and conferences. His research interests include Cyber Security, Body Sensor Networks, E-health, Bio-inspired and Evolutionary Computing, and Internet of Things (IoT). Dr. Khan is the founding director of Wireless Networking and Security (WiNGS) research group at National University of Computer and Emerging Sciences (NUCES), Islamabad, Pakistan. He has successfully supervised two Ph.D. students and sixteen M.S. theses students. Several Ph.D. students are currently working under his supervision. He has served as Associate Editor, Guest Editor, and Reviewer for various reputed international journals. He has also served as co-organizer and TPC member of numerous international conferences and workshops. He is a Senior Member of the IEEE.