Scientia Iranica D (2011) 18 (6), 1460–1468

Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science & Engineering and Electrical Engineering

www.sciencedirect.com

Research note

Anomaly detection using a self-organizing map and particle swarm optimization

M. Lotfi Shahreza [a,*], D. Moazzami [a,b,c], B. Moshiri [d], M.R. Delavar [e]

[a] *Department of Algorithms and Computations, Faculty of Engineering, University of Tehran, P.O. Box 14395-195, Tehran, Iran*

[b] *School of Mathematics, Institute for Research in Fundamental Sciences (IPM), P.O. Box 19395-5746, Tehran, Iran*

[c] *Center of Excellence in Geomatic Engineering and Disaster Management, Tehran, Iran*

[d] *Control & Intelligent Processing, Center of Excellence, School of ECE, Department of Electronic and Computer Engineering, University of Tehran, Faculty of Engineering, P.O. Box 14539-515, Tehran, Iran*

[e] *Center of Excellence in Geomatic Engineering and Disaster Management, Department of Surveying and Geomatic Engineering, University of Tehran, Faculty of Engineering, P.O. Box 11155-4563, Tehran, Iran*

Received 30 March 2010; revised 12 July 2010; accepted 18 January 2011

KEYWORDS

Anomaly detection;
Data fusion;
Neural network;
PSO;
Forest fire.

Abstract Self-Organizing Maps (SOMs) are among the most well-known, unsupervised neural network approaches to clustering, which are very efficient in handling large and high dimensional datasets. The original Particle Swarm Optimization (PSO) is another algorithm discovered through simplified social model simulation, which is effective in nonlinear optimization problems and easy to implement. In the present study, we combine these two methods and introduce a new method for anomaly detection. A discussion about our method is presented, its results are compared with some other methods and its advantages over them are demonstrated. In order to apply our method, we also performed a case study on forest fire detection. Our algorithm was shown to be simple and to function better than previous ones. We can apply it to different domains of anomaly detection. In fact, we observed our method to be a generic algorithm for anomaly detection that may need few changes for implementation in different domains.

1. Introduction

Anomaly detection refers to detecting patterns in a given data set that do not conform to an established, normal behavior. The patterns thus detected are called anomalies, and often translate to critical and actionable information in several application domains. Anomalies are also referred to as outliers, surprise, aberrant, deviation, peculiarity, etc.

We can use anomaly detection methods in a variety of domains, such as intrusion detection, fraud detection, system health monitoring, event detection in sensor networks, and detecting eco-system disturbances.

Three broad categories of anomaly detection techniques exist:

- Supervised anomaly detection techniques learn a classifier, using labeled instances belonging to normal and anomaly classes, and then assign a normal or anomalous label to a test instance [1,2].
- Semi-supervised anomaly detection techniques construct a model representing normal behavior from a given normal training data set, and then test the likelihood of a test instance to be generated by the learnt model [1–3].
- Unsupervised anomaly detection techniques detect anomalies in an unlabeled test data set, under the assumption that the majority of instances in the data set are normal [1,2].

Our method for anomaly detection is an unsupervised method based on clustering, so firstly we should explain it briefly.

* Corresponding author.
E-mail address: Maryam.lotfi@gmail.com (M. Lotfi Shahreza).

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
Peer review under responsibility of Sharif University of Technology.

doi:10.1016/j.scient.2011.08.025

Production and hosting by Elsevier

Clustering means the act of partitioning an unlabeled dataset into groups of similar objects. Each group, called a 'cluster', consists of objects that are similar between themselves and dissimilar to objects of other groups [4]. There are some different methods for measuring this similarity, some typical types of which we will describe.

First, you will see a brief survey of anomaly detection systems, and then we give an introduction to generic SOM and PSO in Sections 1.2 and 1.3, respectively. In Section 1.4, we explain about different attempted methods to use a composite of SOM and PSO. In Section 2, we have some definitions, and we represent main methods of similarity measuring in Section 2.1. We need a criterion for measuring the performance of any approach to compare it with other methods. Thus, some basic approaches for measuring the performance of clustering methods have been expressed in Section 2.2, and in Section 3 we represent our method. Section 4 is a case study of forest fire detection based on our suggested method. We have a discussion about our method and compare its results with some other methods in Section 5. Finally, the results are presented in Section 6.

1.1. A survey on anomaly detection

Anomaly detection systems work by trying to identify anomalies in an environment [5].

At the early stage, the research focus lies in using rule-based expert systems and statistical approaches. But when encountering larger datasets, the results of rule-based expert systems and statistical approaches become worse. Thus, many data mining techniques have been introduced to solve the problem. Among these techniques, the Artificial Neural Network (ANN) is widely used and has been successful in solving many complex practical problems [6].

Some discussed a generic method for anomaly detection that could be used in different areas, but others are about an exclusive subject. Here, we try to introduce their significance.

Some believe unsupervised methods are the best choice for anomaly detection, since they do not need any previous knowledge and only try to find anomalous patterns and cases. While supervised methods can detect only pre-known abnormal cases, unsupervised methods can recognize new and unknown objects. Rouil et al. [7], Eskin et al. [8] and Zakia and Akira [9] tried to express some unsupervised methods for anomaly detection. Also, Guthrie et al. [10] tried to develop an anomaly detection method for finding anomalous segments in a document. Their method is unsupervised; they assumed that there is no data with which to characterize "normal" language. This method is not a classification or clustering method. The method returns a list of all segments ranked, by how anomalous they are with respect to the whole document.

Meanwhile, results show that data fusion methods have good results in this area [5,11–18]. Chen and Aickelin [5] have constructed a Dempster–Shafer based anomaly detection system using the Java 2 platform. First, they used the Wisconsin Breast Cancer Dataset (WBCD) and then the Iris plant dataset, for their experiments. Thirdly, they experimented using an e-mail dataset, which had been created using a week's worth of e-mails (90 e-mails) from a user's sent box, with outgoing e-mails (42 e-mails) sent by a computer infected with the netsky-d worm. The aim of the experiment was to detect the 42 infected e-mails. They used D–S to combine features of the e-mails to detect the worm infected e-mails.

Some various intelligent approaches have also been used for anomaly detection, one of which is the artificial immune system. Greensmith et al. [19] represented a new algorithm for anomaly detection, based on simulation of the human immune system. According to the authors claim, the algorithm performs well on the task of detecting a ping-based port scan and may also be applied to other detection or data correlation problems, such as the analysis of radio signal data from space, sensor networks, internet worm detection and other security and defense applications. Another experiment in this field has undertaken by Twycross and Aickelin [20].

Artificial neural networks are another intelligent method used for anomaly detection. Brause et al. [21] used a compound method based on rule-based systems and an artificial neural network for credit card fraud detection. Other neural network-based credit card fraud detection has been undertaken by Hassibi [22], Dorronsoro et al. [23] and Syeda et al. [24]. Wang et al. [6] proposed a new approach called FC-ANN based on ANN and fuzzy clustering to solve the problem and help IDS achieve a higher detection rate.

An important part of anomaly detection methods is focused on computer intrusion detection.

The task of an intrusion detection system is to protect a computer system by detecting and diagnosing attempted breaches of the integrity of the system [17].

The process of automatically constructing models from data is not trivial, especially for intrusion detection problems. This is because intrusion detection faces problems, such as huge network traffic volumes, highly imbalanced data distribution, the difficulty of realizing decision boundaries between normal and abnormal behavior, and a requirement for continuous adaptation to a constantly changing environment. Artificial intelligence and machine learning have shown limitations in achieving high detection accuracy and fast processing times when confronted with these requirements [2].

Intrusion detection techniques can be categorized into signature detection and anomaly detection. Signature detection systems use patterns of well-known attacks or weak spots in the system to match and identify known intrusions. They perform a pattern matching between network traffic captured and the attack signature. If the matching succeeds, then the system generates an alarm. The main advantage of the signature detection paradigm is that it can accurately and efficiently detect instances of known attacks. The main disadvantage is that it lacks the ability to detect the newly invented attacks. Anomaly detection systems flag observed activities as anomalies when they deviated significantly from established normal usage profiles. The main advantage of anomaly detection is that it does not require prior knowledge of intrusion and can thus detect new intrusions. The main disadvantage is that it may be unable to describe what the attack is, and may have a high false positive rate [18].

Of course, we are able to obtain good ideas from these intrusion detection methods, or with some slight changes, use them for other anomaly detection fields.

Some main attempts towards computer intrusion detection have been done by Anderson et al. [25] who used an outlier detection method. Jake et al. [26] and Ghosh and Schwartzbard [27] who both examined neural network-based methods; Gravey and Lunt [28] who used an evidential reasoning approach; Kumar and Spafford [29] who used a misuse detection method based on rule-based analysis; Lee and Stolfo [30] who tried a classification method by association rules and so on.

A complete survey of fraud detection can be found in [31–33]. Also, Wu and Banzhaf [2] provide an overview of

the research progress in applying computational intelligence methods to the problem of intrusion detection. The scope of this review will encompass core methods of CI, including artificial neural networks, fuzzy systems, evolutionary computation, artificial immune systems, swarm intelligence and soft computing.

1.2. Self-organizing maps

Self-Organizing Maps (SOMs) are the most well known unsupervised neural network approach to clustering.

The architecture of the SOM is a feed-forward neural network with a single layer of neurons arranged into a rectangular array. When an input pattern is presented to the SOM, each neuron calculates how similar the input is to its weights. The neuron whose weights are most similar (minimal distance, d, in input space) is declared the winner of the competition for the input pattern, and the weights of the winning neuron are strengthened to reflect the outcome. The winning neuron receives the most learning at any stage; with neighbors receiving less, the further away they are from the winning neuron [34].

1.2.1. Advantages of SOM

- Working with high dimensional data sets is difficult; the SOM reduces information while preserving the most important topological relationships of the data elements on the two-dimensional plane [1], so that information from different sources can be efficiently fused.
- SOMs are trained using unsupervised learning, i.e. no prior knowledge is available and no assumptions are made about the class membership of data [1].
- The SOM algorithm is very efficient in handling large datasets. The SOM algorithm is also robust even when the data set is noisy [35].

1.2.2. Disadvantages of SOM

- The number of clusters needs to be specified. Clustering is a two-phase process: determining the number of clusters and clustering the data. Determining the number of clusters is not trivial, since the characteristics of the data set are usually not known a priori. This can be overcome by running the algorithm with varying numbers of clusters and selecting the most appropriate clustering result according to a figure of merit [35].
- A user has to either do manual inspection or apply traditional algorithms, like hierarchical or partitive, to find the cluster boundaries [1].

Recently, there have been significant research efforts to apply Evolutionary Computation (EC) techniques for the purpose of evolving one or more aspects of artificial neural networks.

Evolutionary computation methodologies have been applied to three main attributes of neural networks: network connection weights, network architecture (network topology, transfer function) and network learning algorithms.

Most work involving the evolution of ANN has focused on network weights and topological structure.

Over the past several years, there have been several papers that reported using PSO to replace the back-propagation learning algorithm in ANN. It showed PSO as a promising method to train ANN; it is faster and gets better results in most cases. It also avoids some problems met by other methods.

1.3. Particle swarm optimization

The original Particle Swarm Optimization (PSO) algorithm is discovered through simplified social model simulation. In PSO, physical position is not an important factor. The member that is called the particle is initialized by assigning random positions and velocities. During each iteration, every particle is accelerated towards its own personal best, as well as in the direction of the global best position. This is achieved by calculating a new velocity term for each particle, based on the distance from its personal best, as well as its distance from the global best position, which will in turn affect the next position of the particle during the next epoch [36].

1.3.1. Advantages of PSO

- PSO is effective in nonlinear optimization problems.
- It is easy to implement.
- Only a few input parameters need to be adjusted in PSO.
- Because the update process in PSO is based on simple equations, PSO can be efficiently used on large data sets.
- PSO has been successfully applied to many areas: function optimization, artificial neural network training, fuzzy system control and other areas where GA can be applied [35].

1.3.2. Disadvantages of PSO

A disadvantage of global PSO is that it tends to be trapped in a local optimum under some initialization conditions [35].

1.4. A survey on integration of SOM & PSO

Our proposed method is really a combination of a self-organizing map and particle swarm optimization.

First, we will represent a survey of different works that have been done in this field, and then our suggested method is represented in Section 3.

Xiao-Feng et al. [37] used mass extinction to increase the efficiency of PSO. They stated that PSO performs well in the early iterations, but has problems reaching a near optimal solution in several real-valued function optimization problems. So, they reinitialized the velocities of all particles at a predefined extinction interval, I_e, after the determined step. Of course in this method, determining I_e and the required steps for reinitiating the velocities are important and could increase or decrease the performance of the algorithm.

Xiao-Feng et al. [38] add a replacing criterion, based on the diversity of fitness between the current particle and the best historical experience. Indeed, they take off inactive particles and create new particles instead. They called their algorithm APSO. They believed that some particles may become inactive during iterations and declared that an inactive particle means that it will be only flying within quite a small space, which will occur when its position and its local best is close to the global best (if the global best has not significant change) and its velocity is close to zero (for all dimensions) [38]. In this way, they could prevent it from finding the local optimum instead of global. However, it is hard to identify the inactive particles for different problems. We extract some part of our method from APSO.

Xiang et al. [35] proposed a SOM/PSO algorithm that uses PSO to evolve the weights for SOM. In this algorithm, at the first stage, weights are trained by SOM and, at the second stage, they are optimized by PSO. In their method, each particle consists of a complete set of weights for SOM. The dimension of each particle is the number of input neurons of SOM times the number of output neurons of SOM. This increases the time and space complexity of the algorithm. In fact, their algorithm clusters input dataset by standalone SOM and then apply PSO for refining these clusters.

O'Neill and Brabazon [39] introduced a hybrid method of SOM and PSO as SOSwarm. In fact, they used PSO for updating

the weights of the neural network. In this method, the components of the mapping layer represent particles, which move according to an adapted version of the Particle Swarm algorithm.

Instead of adjusting vector values in the map space, with respect to the training input vectors alone, the particles (vectors) in the mapping layer adjust their location using a PSO update function.

They then applied their method to four benchmark classification problems from the UCI Machine Learning repository. Their results were satisfactory and indeed the basis of our method is the SOSwarm.

Using PSO for updating the weights of neurons is a good idea. However, they did not really use PSO. In fact, the SOSwarm is just a new version of the simple SOM, which instead of working just with one parameter 'weight', they update the 'velocity' parameter and then update a 'position' parameter by it; doing the same for the next input. I think it is not really a PSO.

Another problem is that the SOSwarm uses a Euclidean distance for measuring distance, but the Euclidean distance is not a good choice for complex datasets [4].

Swagatam et al. [4] created another method, based on SOM and PSO for clustering, as the Multi-Elitist PSO (MEPSO) algorithm.

One of the best things in this method is that it uses a kernel-induced similarity measure instead of the conventional sum-of-squares distance. Kernel functions make it possible to cluster data that are linearly non-separable.

MEPSO prevents accepting a local optimum instead of a global one, but because of the kind of particle representation, this method has high complexity in time and space. In MEPSO, clusters may or may not be active in some particles; this will reduce the performance of the method.

Anurag and Christian [1] proposed another hybrid algorithm. They proposed to use the PSO algorithm for finding cluster boundaries directly from the code vectors obtained from SOM. In fact, they clustered their input data set by generic SOM and then found the cluster boundary automatically from output code vectors, using generic PSO. Other methods that have been represented for this purpose are sensitive to the number of created clusters, but since PSO works individually with a particular cluster, it is insensitive to the number of clusters in the data set. One thing more is that PSO is not sensitive to noise and outlier, however, the choice of cluster centers affects its final result.

2. Definitions

2.1. Similarity measure in clustering

There are some basic methods for measuring the distance between two data points in clustering algorithms, some of which are:

- Euclidean distance (L_2)

$$D(x, y) = \sqrt{\sum_i (x_i - y_i)^2}. \tag{1}$$

- Manhattan distance (L_1)

$$D(x, y) = \sum_i |x_i - y_i|. \tag{2}$$

It is also known as city-block distance [40].

- Chebychev distance (L_∞)

$$D(x, y) = \max |x_i - y_i|. \tag{3}$$

It is also known as sup distance [40].

- Categorical data distance

$$D(x, y) = (\text{number of } x_i - \text{ number of } y_i)/N. \tag{4}$$

N: total number of categorical attributes.

- Minkowski distance
The Minkowski distance is a metricon Euclidean space, which can be considered as a generalization of both Euclidean and Manhattan distances.

$$D(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \qquad p \geq 1. \tag{5}$$

Minkowski distance is typically used with p being 1 or 2. The latter is the Euclidean distance, while the former is sometimes known as the Manhattan distance [40].

In the limiting case of p reaching infinity, we obtain the Chebyshev distance:

$$\lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} = \max |x_i - y_i|. \tag{6}$$

Mahalanobis distance

$$D(x, y) = (x_i - y_i)^T S^{-1} (x_i - y_i), \tag{7}$$

where S is the within-group covariance matrix [40].

- Kernel-based functions
Here, we explain further Euclidean distance and kernel-based functions.

2.1.1. Euclidean distance

The Euclidean distance metric, employed by most existing partitional clustering algorithms, works well with datasets in which the natural clusters are nearly hyperspherical and linearly separable. But it causes severe misclassifications when the dataset is complex, with linearly non-separable patterns [4].

The most popular way to evaluate similarity between two patterns amounts to the use of the Euclidean distance, which between any two d-dimensional patterns, $\vec{x}_I$ and $\vec{x}_J$, is given by:

$$d(\vec{x}_i, \vec{x}_j) = \sqrt{\sum_{p=1}^{d} (x_{i,p}, x_{j,p})^2} = \|\vec{x}_i - \vec{x}_j\|. \tag{8}$$

2.1.2. The kernel-based similarity measure

A kernel function measures the distance between two data points by implicitly mapping them into a high dimensional feature space, where the data is linearly separable [4].

Given a dataset, X, in the d-dimensional real space, R^d, let us consider a non-linear mapping function from the input space to a high dimensional feature space, H:

$$\varphi: R^d \to H, \qquad \vec{x}_i \to \varphi(\vec{x}_i), \tag{9}$$

where:

$$\vec{x}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,d}]^T,$$

and:

$$\varphi(\vec{x}_i) = [\varphi_1(\vec{x}_i), \varphi_2(\vec{x}_i), \ldots, \varphi_H(\vec{x}_i)]^T.$$

By applying the mapping, a dot product, $\vec{x}_i^T . \vec{x}_j$, is transformed into $\varphi^T(\vec{x}_i).\varphi(\vec{x}_j)$. Now, the central idea in kernel-based learning is that the mapping function, φ, need not be explicitly specified.

Hence, the kernelized distance measure between two patterns, $\vec{x}_i$ and $\vec{x}_j$, is given by:

$$\begin{aligned}\|\varphi(\vec{x}_i) - \varphi(\vec{x}_j)\|^2 &= (\varphi(\vec{x}_i) - \varphi(\vec{x}_j))^T(\varphi(\vec{x}_i) - \varphi(\vec{x}_j)) \\ &= \varphi^T(\vec{x}_i) - \varphi(\vec{x}_i) - 2\varphi^T(\vec{x}_i).\varphi(\vec{x}_j) + \varphi^T(\vec{x}_j).\varphi(\vec{x}_j) \\ &= K(\vec{x}_i, \vec{x}_i) - 2K(\vec{x}_i, \vec{x}_j) + K(\vec{x}_j, \vec{x}_j). \end{aligned} \quad (10)$$

The Gaussian kernel (also referred to as the Radial Basis Function) is well known, owing to its better classification accuracy over linear and polynomial kernels on many test problems. The Gaussian Kernel may be represented as:

$$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}\right), \quad \sigma > 0. \quad (11)$$

Clearly, for the Gaussian kernel, $K(\vec{x}_i, \vec{x}_i) = 1$ and, thus, Relation (10) reduces to:

$$\|\varphi(\vec{x}_i) - \varphi(\vec{x}_j)\|^2 = 2(1 - K(\vec{x}_i, \vec{x}_j)). \quad (12)$$

In fact, we used this measurement in our algorithm.

2.2. Performance measuring

One major issue in using a clustering algorithm to cluster new and unknown expression data is measuring the robustness of the clustering result.

Here, we will see some methods for measuring the performance of clustering methods:

- The general criterion of good partitioning is that objects in the same cluster are "close" or related to each other, whereas objects of different clusters are "far apart" or very different. Some popular algorithms are k-means and k-medoids [1].
- One method for this is the resampling technique. This technique is based on the simple idea that stipulates that, if the algorithm is applied to a randomly selected subset of the original set, then patterns that are in the same cluster in the original clustering should also be in the same cluster in the clustering result obtained for the subset, if the result is robust. Multiple subsets can be selected randomly, and the results of clustering these subsets can be compared to the original clustering result in order to measure the robustness of the clusters obtained. The difference between the clustering based on the randomly selected subset and the original clustering result is measured using a merit function, which is expressed as follows:

$$\text{merit} = \sqrt{\frac{\sum_j \sum_i \left(T_{ij}^{(\mu)} - T_{ij}\right)^2}{\text{No. of patterns in the selected subset}}}. \quad (13)$$

$T_{ij}^{(\mu)}$ is an element in the original similarity matrix and T_{ij} is an element in the resampled similarity matrix. A similarity matrix is constructed as follows:

$$T_{ij} = \begin{cases} 1 & \text{pattern } i \text{ and } j \text{ are in the same cluster} \\ 0 & \text{pattern } i \text{ and } j \text{ not in the same cluster.} \end{cases} \quad (14)$$

The smaller the value of the merit, the more robust the algorithm is. The method can also be used to estimate the number of clusters needed for a given dataset. Given an unknown data set, several runs of a given clustering algorithm, under varying input parameters, can be performed. If resampling is used with each run, the clustering result of choice is the one with the lowest merit value. This can be used to choose an adequate number of clusters when running a clustering algorithm on an unknown data set. One major drawback of the resampling technique is that it is computationally expensive [35].

- One criterion for the quality of the clustering involves measuring the degree of difference between clusters.

  Inspecting the average values of variables across different clusters is one simple method for measuring the differentiation ability between clusters. It is preferable to have clusters whose profiles are statistically different from each other [34].
- Ward's minimum-variance method: for computation of the distance between two clusters;

$$D_w(A, B) = \frac{N_A N_B D_C(A, B)}{(N_A + N_B)} \quad (15)$$

  where:

N_A	number of objects in A
N_B	number of objects in B
$D_C(A, B)$	the centroid distance between the two clusters, computed as the squared Euclidean distance between the centroids.

- The clustering results can be judged using Huang's accuracy measure

$$r = \frac{\sum_{i=1}^{k} n_i}{n}, \quad (16)$$

  where n_i is the number of data occurring in both the ith cluster and its corresponding true cluster, and n is the total number of data points in the data set. According to this measure, a higher value of r indicates a better clustering result with perfect clustering yielding a value of $r = 1$.

3. Suggested method

In this method, we have a layer of neurons, but we will treat each neuron just like a particle. In fact, we have a network of particles whose network is based on the idea of a self-organizing map, but each particle of which will work according to the general PSO algorithm. (It is like using PSO for updating the weights' of SOM, although with some differences). One can see the pseudo-code of our method in Figure 1.

We define a group of particles. The position of particle i is represented as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and the position of each particle is equal to the weight of the neurons. In other words, the dimension of each particle is equal to the dimension of each neuron, and each dimension shows the amount of equal weight. At first, we set the positions of all particles randomly.

Each particle also maintains a memory of its previous best position, represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$. The global best is also represented by $P_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$.

Each particle has a velocity, which can be represented as:

$V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$.

At first, we set the velocity vector of each particle to be 0.0001, and then in each iteration, we update the velocity and position of the ith particle by Eqs. (17) and (18), respectively:

$$V_i(t+1) = V_i(t) + c_1 * (P_i - X_i(t)) + c_2 * (P_g - X_i(t)), \quad (17)$$

$$X_i(t+1) = X_i(t) + V_i(t+1). \quad (18)$$

In each iteration, P_g for each dimension is the global best position for that dimension found so far.

For each input, at first we set the P vector of each particle to be its current position, and set P_g to the value of the winning particle. For finding the winning particle, we need to measure similarity of each particle with the input vector.

```
Initial all particles randomly
For (each input vector)
{
    Find particle which is most similar (according to our kernel function)
    to input vector,
    If (learning-radius is zero)
        Update the position of each dimension of winner particle (just
        like normal SOM),
 Else {
        Set P vector of each particle to be its current position,
        Set P_g to the value of winner particle,
        For (max number of iteration),
          {
            Update winner particle's velocity and position vectors,
            Update velocity and position vectors of neighbors
            of winner particle according to learning-radius,
            Update P vector of each particle,
            Update P_g,
          }
      }
}
Assign each input to one cluster according to its winner particle,
Assign each cluster a suitable label according to its members,
Categorize all inputs in three categories according to their cluster's label
and mean of each cluster,
Calculate clustering accuracy using test data.
End
```

Figure 1: Pseudo-code of suggested method.

For similarity measuring, we use the kernel function introduced in the section: 'the kernel-based similarity measure'. For each particle vector, we compute its similarity with the input vector obtained by Eq. (5)

$$\|\varphi(\vec{x}_i) - \varphi(\vec{x}_j)\|^2 = 2(1 - K(\vec{x}_i, \vec{x}_j)), \tag{19}$$

which:

$$K(\vec{x}_i, \vec{x}_j) = \exp\left(-\frac{\|\vec{x}_i - \vec{x}_j\|^2}{2\sigma^2}\right), \quad \sigma > 0,$$

and:

$$\|\vec{x}_i - \vec{x}_j\|^2 = \sum_{p=1}^{d} (x_{i,p}, x_{i,p})^2.$$

Smaller values of this function show more similarity, so in each iteration, the winning particle will be that which has the smallest value of this function.

Because of the difficulty in comparing two small values, we multiplied the value of the similarity function by the power of ten. This created more accurate answers.

In contrast with generic PSO, we update the velocity and position vectors of the neighbors of the winning particle, in addition to those of the winning particle itself. For its implementation, according to the learning-radius, we use some parameters that show the distance between particles, according to their topology.

Finally, when all inputs are assigned to one particle, we will have as many clusters as the total number of particles. Now we should categorize these clusters. Because our purpose is to find anomalies in the input data, we should define three categories. One is 'anomalous' data, another is 'probable anomalous' data and the other is 'normal' data. One major method for categorizing clusters is to investigate their members. Clusters with few members are good candidates for 'anomalous' clusters. We use the mean of members of all clusters for this categorization. For clusters with many members, we also use one other method for categorization. This part is based on the distance between the mean of investigating the cluster and its members. Members who are so far from the mean of their cluster are not normal.

In order to measure the accuracy of our method, in addition to a 'false positive' rate and a 'false negative' rate, here we define two new parameters, which are 'correct probable' and 'false probable'.

- A false positive is the rate of 'normal' cases realized as 'anomalous' incorrectly by the algorithm.
- A false negative is the rate of 'anomalous' cases realized as 'normal' incorrectly by the algorithm.
- A correct probable is the total number of cases that are anomalies, but instead of announcing them as 'anomalous' data, the method announce them as 'probable anomalous'.
- A false probable is the total number of cases that are normal, but instead of announcing them as 'normal' data, the method announces them as 'probable anomalous'.

For measuring the performance of our algorithms, we also use two other parameters: Huang's accuracy measure and Ward's minimum-variance, which were introduced in previous sections. As mentioned before, whichever algorithm whose Huang's accuracy measure is nearest to one is the best, and the greatest Ward's minimum-variance is desirable.

3.1. Advantages of suggested method

- One advantage of the proposed method is its low time and space complexity, such that one can run it in a few minutes and at some mega byte memory for tens of thousands of data record entries.
- The algorithm is simple; it has no complex and difficult computation.
- The method can be applied to different and variant domains of anomaly detection; in fact, it is a generic method for anomaly detection and needs few changes for implementation in different domains.
- It is an unsupervised method and because of the lack of labeled data, unsupervised methods are more useful. Besides, supervised methods can only detect anomalies with recognized patterns.
- There is no need to be familiar with fields of used dataset. We want to find anomalous data in a set of related data, without knowing anything about them.
- Sources of information are often linked with some sort of dependence in real life [14–16]. While most methods process one source or those that spot different sources assume that they are statistically independent from each other, this assumption does not always hold true. Our method assumes a dependency between fields of a record and processes them, en bloc.

4. Case study: forest fire detection

Forest fire is a major environmental issue, creating economical and ecological damage while endangering human lives. Fast detection is a key element in controlling such phenomenon.

Each year, millions of forest hectares (ha) are destroyed all around the world. In particular, Portugal is highly affected by forest fires. From 1980 to 2005, over 2.7 million ha of forest area (equivalent to the area of Albania) were destroyed. The 2003 and 2005 fire seasons were especially dramatic, affecting 4.6% and 3.1% of the territory, with 21 and 18 human deaths, respectively [41].

Fast detection is a key element in controlling such phenomenon and traditional methods are no longer useful, so there

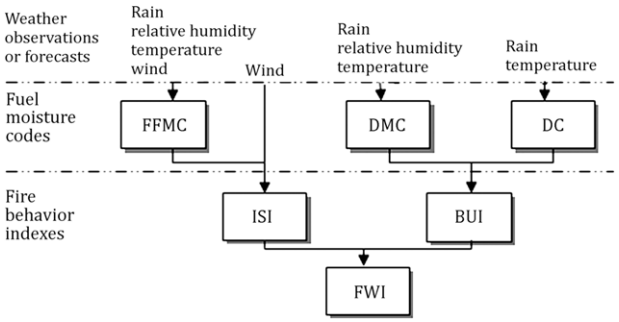

Figure 2: The fire weather index structure.

has been an emphasis on developing automatic solutions. Different new solutions can be categorized into three groups:

- Satellite-based.
- Infrared/smoke scanners.
- Local sensors (e.g. meteorological).

Satellites have acquisition costs, localization delays and their resolution is not adequate for all cases [41]. Moreover, scanners have high equipment and maintenance costs.

Local sensors seem to be the best option here, because of their low cost, and on the other hand the plurality of meteorological stations all over the world cause the simplicity of measuring parameters, such as weather conditions (like temperature and air humidity), which are known to affect fire occurrence.

4.1. Data

The forest Fire Weather Index (FWI) is the Canadian system for rating fire danger and it includes six components (Figure 2) [41]: the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), the Drought Code (DC), the Initial Spread Index (ISI), the Buildup Index (BUI) and FWI. The first three are related to fuel codes. The FFMC denotes the moisture content of surface litter and influences ignition and fire spread, while the DMC and DC represent the moisture content of shallow and deep organic layers, which affect fire intensity. The ISI is a score that correlates with fire velocity spread, while BUI represents the amount of available fuel. The FWI index is an indicator of fire intensity and it combines the two previous components. Although different scales are used for each of the FWI elements, high values suggest more severe burning conditions. Also, the fuel moisture codes require a memory (time lag) of past weather conditions: 16 h for FFMC, 12 days for DMC and 52 days for DC.

This study will consider forest fire data from the Montesinho natural park in the Tr'as-os-Montes, the northeast region of Portugal. These data were collected from January 2000 to December 2003, and were built using two sources; the inspector that was responsible for the Montesinho fire occurrences and by the Braganca Polytechnic Institute, containing several weather observations located in the center of the Montesinho Park, which were then integrated into a single dataset with a total of 517 entries. This data is available at [42]: http://www.dsi.uminho.pt/~pcortez/forestfires/.

Attribute information:

1. X: x-axis spatial coordinate within the Montesinho park map: 1–9;
2. Y: y-axis spatial coordinate within the Montesinho park map: 2–9;
3. Month: month of the year: "Jan" to "Dec";
4. Day: day of the week: "Mon" to "Sun";
5. FFMC: FFMC index from the FWI system: 18.7–96.20;
6. DMC: DMC index from the FWI system: 1.1–291.3;
7. DC: DC index from the FWI system: 7.9–860.6;
8. ISI: ISI index from the FWI system: 0.0–56.10;
9. Temp: temperature in Celsius degrees: 2.2–33.30;
10. RH: relative humidity in %: 15.0–100;
11. Wind: wind speed in km/h: 0.40–9.40;
12. Rain: outside rain in mm/m^2: 0.0–6.4;
13. Area: the burned area of the forest (in ha).

Table 1: Results of comparing different methods on simulated data. Total number of normal cases is 14 978 and total number of abnormal cases is 104.

Algorithm	Parameter			
	False negative	False positive	Huang's accuracy measure	Ward's minimum variance
SOM	49	2484	0.832	12.336
Bayesian estimation	30	890	0.939	90.309
Dempster–Shafer (statistical)	80	4601	0.689	68.901
Dempster–Shafer (Chen)	65	4368	0.706	98.931
Our suggested method	48	249	0.99	342.12

Table 2: Results of comparing different methods on real data. Total number of normal cases is 422 and total number of abnormal cases is 95.

Algorithm	Parameter			
	False negative	False positive	Huang's accuracy measure	Ward's minimum variance
SOM	78	66	0.721	11.702
Bayesian estimation	39	212	0.514	146.873
Dempster–Shafer (statistical)	63	158	0.572	129.695
Dempster–Shafer (Chen)	77	67	0.721	254.265
Our suggested method	55	87	0.832	360.579

4.2. Implementation

In fact, we have two phases for our anomaly detection. First, we should use test data and find suitable input parameters and then apply the program to unlabeled data using suitable input parameters (suitable input parameters are those that cause the best results). So, our anomaly detection system uses a training process to derive input parameters from the test data, and detects an entry as normal or abnormal.

It is necessary to say that our method has been implemented completely by visual studio.net 2005 (C# language).

5. Discussion

Results of comparing different methods on simulated data are presented in Table 1, and on real data in Table 2. Note that these results are the best of each method, and all were implemented using the same language and run under the same conditions.

Our simulated dataset is constructed of 14,987 normal cases and 104 abnormal cases, and our real dataset is constructed of 422 normal cases and 95 abnormal cases.

In each row, we show the result parameters of each compared method (we have explained these parameters before).

Self-Organizing Map results are shown in the first row. It is constructed of a hundred neurons in the hidden layer. We try

different input parameters for it (like learning rate and learning radius) and save its best results. In Table 1, however, its False Negative is comparable with our method, but its False Positive is very bad. Thus, its Huang's Accuracy Measure is much lower than our method. One can see its Ward's Minimum Variance is also very low.

We think that one reason for this bad result is the use of the Euclidean Distance for measuring the distance in generic SOM. As mentioned before, the Euclidean Distance is only suitable for linear separable data, but in our discussed domain, the data is complex. So, we do not use the Euclidean Distance in our method. In our opinion, Kernel Based functions are the best choice for measuring the distance here. The other reason is an insufficient approach for updating the weights of neurons in generic SOM. In fact, we use PSO for solving this problem in our method.

Bayesian Estimation (second row) is the most comparable method with our approach. We use a statistical method on part of the labeled data for finding the probabilities of this method. In Table 1, however, its False Negative is lower than our method, but because of its relative high False Positive, in total, the Huang's Accuracy Measure of our method is higher than that of the Bayesian Estimation. Its Ward's Minimum Variance is not too bad.

For Dempster–Shafer, we need some mass values for each hypothesis. In order to find them, we use two approaches and implement them as two Dempster–Shafer methods. One is Dempster–Shafer (statistical) that, like Bayesian Estimation, uses a statistical method on part of the labeled data for finding mass values, and the second is Dempster–Shafer (Chen) that assigns mass values according to the Chen and Aickelin [5] approach (it is based on some thresholds).

In fact, in both methods, the mass values for each hypothesis are generated and sent to the Dempster–Shafer combination component. This component uses the Dempster rule of combination to combine all mass values and generate the overall mass values for each hypothesis. If the mass value of the 'abnormal' hypothesis is bigger than the mass value of the 'normal' hypothesis, then it is classified as abnormal; otherwise it is classified as normal.

In Table 1, Dempster–Shafer (statistical) has the worst result of three parameters, only its Ward Minimum variance is higher than SOM. However, results of Dempster–Shafer (Chen) are a little better than Dempster–Shafer (statistical), but are not sufficient at all.

Results of both Dempster–Shafer and Bayesian approaches are not acceptable because:

- Both theories have a certain initial requirement; Dempster–Shafer requires masses to be assigned and Bayesian Estimation requires prior probability. Both are highly sensitive to this assignment. We need to assign these values using an expert or a computing or intelligence method and this could be time consuming and unfruitful.
- The computation complexity.
- Dempster–Shafer assumes that the pieces of evidence are statistically independent from each other. Since sources of information are often linked with some sort of dependence in real life situations, this assumption does not always hold true [14–16]

Our method does not have any of these problems. It is an unsupervised method that does not need an initial assignment, like Dempster–Shafer and Bayesian estimation. It is a simple method and does not have any complex computation. Finally, our method assumes a dependency between fields of a record and processes them, en bloc.

The same can be seen in Table 2, that our suggested method has the best Huang's Accuracy Measure and Ward's Minimum Variance (although its false positive and false negative are higher than others in some cases, this has not occurred at the same time for both and so Huang's Accuracy Measure and Ward's Minimum Variance are best). Of course, these are not as good as the results of Table 1. The major reason is that our method supposes anomalous cases to be rare (Anomaly detection is orthogonal to misuse detection. It hypothesizes that abnormal behavior is rare and different from normal behavior. Hence, it builds models for normal behavior and detects anomalies in observed data by noticing deviations from these models [2].) The proportion of the number of abnormal cases to the total number in this dataset is 0.184, i.e. a large number (it seems that we did not choose a suitable dataset, although we did not have much choice). Of course, we believe that this is not a shortcoming in our method, since, usually, normal cases are many more than abnormal cases in our interested datasets. Almost all anomaly detection systems are based on this assumption and this means that we can use these methods for detecting abnormal cases in nature monitoring datasets.

6. Conclusions

Today anomaly detection methods are of major interest to the world and are used in very different and various domains like computer intrusion detection, credit card and telephone fraud detection, spam detection, and so on. Here, we have introduced a new unsupervised method for anomaly detection, based on a combination of a Self-Organizing Map and Particle Swarm Optimization that fuse information from various sources. It is a simple, time and space consuming method that can be used in different domains. In this paper, we wished to implement it for crisis management, so we chosed forest fires and their detection. In comparison with some other methods, like the Self-Organizing Map, Dempster–Shafer and Bayesian Estimation, we obtained good results. Like other anomaly detection methods, when abnormal cases are rare, our suggested method has better results than when they are not.

We have implemented this method in various domains and wish to investigate its results in some others.

Acknowledgments

This work was supported by Tehran University. Our special thanks go to the University of Tehran, Faculty of Engineering, Department of Algorithms and Computations for providing all necessary facilities for successfully conducting this research. Also, we would like to thank the Center of Excellence, Geomatic Engineering and Disaster Management for partial support of this research.

This paper was prepared while the first two authors were visiting the Institute for Studies in Theoretical Physics and Mathematics (IPM). It is a pleasure to thank IPM for its hospitality and facilities.

References

[1] Anurag, S. and Christian, W.O. "Performance comparison of particle swarm optimization with traditional clustering algorithms used in self-organizing map", *International Journal of Computational Intelligence*, 5(1), pp. 32–41 (2009).

[2] Shelly Xiaonan, Wu and Banzhaf, W. "The use of computational intelligence in intrusion detection systems: a review", *Review Article Applied Soft Computing*, 10(1), pp. 1–35 (2010).

[3] Xiaojin, Zhu "Semi-supervised learning literature survey", Computer Sciences TR 1530, University of Wisconsin–Madison, Last modified on July 19 (2008).

[4] Swagatam, D., Ajith, A. and Amit, K. "Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm", *Pattern: Recognition Letters*, 29(5), pp. 688–699 (2008).
[5] Chen, Q. and Aickelin, U. Dempster–Shafer for anomaly detection, *Proceedings of the International Conference on Data Mining DMIN 2006*, Las Vegas, USA, pp. 232–238 (2006).
[6] Wang, Gang, Hao, Jinxing, Ma, Jian and Huang, Lihua "A new approach to intrusion detection using artificial neural networks and fuzzy clustering", *Original Research Article Expert Systems with Applications*, 37(9), pp. 6225–6232 (2010).
[7] Rouil, R., Chevrollier, N. and Golmie, N. "Unsupervised anomaly detection system using next-generation router architecture", *Military Communication Conference MILCOM*, USA (2005).
[8] Eskin, E., Arnold, A., Prerau, M., Portnoy, L. and Stolfo, S. "A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data", In *Data Mining for Security Applications*, Kluwer (2002).
[9] Zakia, F. and Akira, M. "Unsupervised outlier detection in time series data", *Proceedings of the Second International Special Workshop on Databases for Next-Generation Researchers SWOD2006*, Atlanta, GA, pp. 51–56 (Apr. 2006).
[10] Guthrie, D., Guthrie, L., Allison, B. and Wilks, Y. "Unsupervised anomaly detection", *IJCAI 2007*, pp. 1624–2162 (2007).
[11] Chatzigiannakis, V., Androulidakis, G., Pelechrinis, K., Papavassiliou, S. and Maglaris, V. "Data fusion algorithms for network anomaly detection: classification and evaluation", *Proceedings of the Third International Conference on Networking and Services*, pp. 50–51 (2007).
[12] Yu, D. and Frincke, D. "Alert confidence fusion in intrusion detection systems with extended Dempster–Shafer theory", ACM-SE 43: *Proceedings of the 43rd Annual Southeast Regional Conference*, 2, pp. 142–147 (2005).
[13] Te-Shun, C., Sharon, F., Wei, Z., Jeffrey, F. and Asad, D. "Intrusion aware system-on-a-chip design with uncertainty classification", *The 2008 International Conference on Embedded Software and Systems-ICESS* (2008).
[14] Siaterlis, C., Maglaris, B. and Roris, P. "A novel approach for a distributed denial of service detection engine", National Technical University of Athens, Athens, Greece (2003).
[15] Siaterlis, C. and Maglaris, B. "Towards multisensor data fusion for DoS detection", *Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 439–446 (2004).
[16] Siaterlis, C. and Maglaris, V. "One step ahead to multisensor data fusion for DDoS detection", *Journal of Computer Security*, 13, pp. 779–806 (2005).
[17] Ambareen, S., Susan, M. and Bridges, R.B.V. "Fuzzy cognitive maps for decision support in an intelligent intrusion detection system", National Science Foundation Grant# CCR-9988524 and the Army Research Laboratory Grant # DAAD17-01-C-0011.
[18] Maselli, G., Deri, L. and Suin, S. "Design and implementation of an anomaly detection system: an empirical approach", *Proceedings of Terena Networking Conference TNC 03*, Zagreb, Croatia (May 2003).
[19] Greensmith, J., Aickelin, U. and Tedesco, G. "Information fusion for anomaly detection with the dendritic cell algorithm", *Information Fusion*, pp. 21–34 (2010).
[20] Twycross, J. and Aickelin, U. "An immune-inspired approach to anomaly detection", In *Handbook of Research on Information Assurance and Security*, J.N.D. Gupta and S.K. Sharma, Eds., pp. 109–121, IGI Global, New York (Chapter 10) (2009).
[21] Brause, R., Langsdorf, T. and Hepp, M. "Credit card fraud detection by adaptive neural data mining", *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, pp. 103–106 (1999).
[22] Hassibi, K. "Detecting payment card fraud with neural networks", In *Business Applications of Neural Networks*, P.J.G. Lisboa, A. Vellido and B. Edisbury, Eds., World Scientific, Singapore (2000).
[23] Dorronsoro, J.R., Ginel, F., Sanchez, C. and Cruz, C.S. "Neural fraud detection in credit card operations", *IEEE Transactions on Neural Networks*, 8, pp. 827–834 (1997).
[24] Syeda, M., Zhang, Y.Q. and Pan, Y. "Parallel granular neural networks for fast credit card fraud detection", *Proceedings of the 2002 IEEE International Conference*, 1, pp. 572–577 (2002).
[25] Anderson, D., Rivold, T., Tamaru, A. and Valdes, A. Intrusion detection expert system nides, Next Generation, Software Users Manual, Beta-Update Release, Technical ReportSRI-CSL-95-07, Computer Science Laboratory, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025-3493 (May 1994).
[26] Jake, R., Meng-Jang, L. and Risto, M. "Intrusion detection with neural networks", In *Advances in Neural Information Processing Systems 10*, M.I. Jordan, M.J. Kearns and S.A. Solla, Eds., MIT Press (1998).
[27] Ghosh, A.K. and Schwartzbard, A. "A study in using neural networks for anomaly and misuse detection", *Proceedings of the 8th USENIX Security Symposium*, Washington D.C. (1999).
[28] Garvey, T.D. and Lunt, T.F. "Model based intrusion detection", *Proceedings of the 14th National Computer Security Conference*, October (1991).
[29] Kumar, S. and Spafford, E.H. "A pattern matching model for misuse intrusion detection", *Proceedings of the 17th National Computer Security Conference*, pp. 11–21 (1994).
[30] Lee, V. and Stolfo, S. "Data mining approaches for intrusion detection", *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX (1998).
[31] Chun, W.C.P. "Investigative data mining in fraud detection", A Thesis, submitted in partial fulfillment of the requirement for the Degree of Bachelor of Business Systems Honours, School of Business Systems, Monash University November, 2003.
[32] Clifton, P., Vincent, L., Smith, K. and Gayler, R. "A comprehensive survey of data mining-based fraud detection, Research", Final version 2: 9/02/2005.
[33] Yufeng, K., Chang-Tien, L., Sirirat, S. and Yo-Ping, H. "Survey of fraud detection techniques", *Proceedings of the 2004 International Conference on Networking, Sensing and Control*, pp. 749–754 (March 2004).
[34] Smith, K.A. "Introduction to Neural Networks and Data Mining for Business Applications", Eruditions Publishing 1999. viii, 155 p.: ill.; 25 cm. ISBN 1864910046: 1864910046 Pbk 18649100046 (chapter one) (1999).
[35] Xiang, X., Ernst, R.D., Russell, E., Zina, B.M. and Robert, J.O. "Gene clustering using self-organizing maps and particle swarm optimization", *Ipdps, International Parallel and Distributed Processing Symposium IPDPS'03*, p. 154b (2003).
[36] Abdull, H. and Haza, N. "Particle swarm optimization for neural network learning enhancement", Master Thesis, University Teknology Malaysia (2006).
[37] Xiao-Feng, X., Wen-Jun, Z. and Zhi-Lian, Y. "Hybrid particle swarm optimizer with mass extinction", *International Conference on Communication, Circuits and Systems ICCCAS*, Chengdu, China, pp. 1170–1173 (2002).
[38] Xiao-Feng, X., Wen-Jun, Z. and Zhi-Lian, Y. "Adaptive particle swarm optimization on individual level", *International Conference on Signal Processing ICSP*, Beijing, China, pp. 1215–1218, (2002).
[39] O'Neill, M. and Brabazon, A. "Self-organising Swarm SoSwarm", *Soft Computing*, 12(11), pp. 1073–1080 (2008).
[40] Rui, X. and Donald, W. "Survey of clustering algorithms", *IEEE Transactions on Neural Networks*, 16(3), pp. 645–678 (2005).
[41] Cortez, P. and Morais, A. "A data mining approach to predict forest fires using meteorological data", *Proceedings of the 13th EPIA 2007 – Portuguese Conference on Artificial Intelligence*, December, Guimaraes, Portugal, APPIA, ISBN-13 978-989-95618-0-9, pp. 512–523 (2007).
[42] Asuncion, A. and Newman, D.J. "UCI Machine Learning Repository" http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, School of Information and Computer Science (2007).

Maryam Lotfi Shahreza received a B.S. degree in Computer Engineering from Isfahan University of Technology and an M.S. in Computer Engineering, Algorithm and Computation from Tehran University in 2009. Her research interests are crisis management, data fusion, bioinformatics and data mining. She is currently teaching in Isfahan University of Technology.

Dara Moazzami received B.S. and M.S. degrees in Mathematics from Quebec University, Montreal, Canada, and a Ph.D. in Mathematics from Boston University, USA. He is Associate Professor of Graph Theory and Discrete Mathematics. He is Member of the Editorial Board of the Tehran University Engineering Journal "Fani", Editor-in-Chief of the Journal of Algorithms and Computation, and Member of the Board of the Center of Excellence in Geomatic Engineering and Disaster Management. His research interests include vulnerability and reliability in graphs, stability of communication networks, success tree method and fuzzy set theory for evaluation of uranium resources, and analysis of vulnerability measure in networks.

Behzad Moshiri was born in Tehran, Iran, in 1959. He received a B.S. degree in Mechanical Engineering from Iran University of Science and Technology (IUST) in 1984. He pursued his higher studies leading to M.S. and Ph.D. degrees in Control Systems Engineering from the University of Manchester, Institute of Science and Technology (UMIST), UK, in 1987 and 1991, respectively. He was a member of ISA from 1991–1992. He joined the Department of Electrical and Computer Engineering at the University of Tehran in 1992. He is currently Professor of Control Engineering and has been head of the Machine Intelligence & Robotics Division at the school of ECE. He received the Distinguished Researcher Award from Tehran University in 2003 and the Distinguished Alumnus Award from IUST in 2004. He has published more than 260 papers in journals and conferences. Professor Moshiri has been elevated to the grade of Senior Member in the IEEE since June 2006. His research interests include advanced industrial control design, advanced instrumentation design, sensor data fusion, mechatronics and bioinformatics.

Mahmoud Reza Delavar obtained his Ph.D. in Geomatic Eng.-GIS from the University of New South Wales, Sydney, Australia in 1997, and is now working as Assistant Professor in the Department of Surveying and Geomatic Engineering College of Engineering at Tehran University. He is founder of the Iranian Society of Surveying and Geomatic Engineering, Director of GIS, Center of Excellence in Geomatic Engineering and Disaster Management at Tehran University, National representative of the Urban Data Management Society/Symposium (UDM.S.) and has been the Scientific Secretary of ISPRS WG II/4 (uncertainty modeling and quality control for spatial data) from 2008–2012. His research interests are GIS spatial data quality, spatio-temporal GIS and land administration.