State space model-based trust evaluation over wireless sensor networks: an iterative particle filter approach

Bin Liu[1], *Shi Cheng*[2]

[1]*School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, People's Republic of China*
[2]*School of Computer Science, Shaanxi Normal University, Xi'an 710062, People's Republic of China*
E-mail: bins@ieee.org

Published in *The Journal of Engineering*; Received on 16th December 2016; Accepted on 9th March 2017

Abstract: In this study, the authors propose a state space modelling approach for trust evaluation in wireless sensor networks. In their state space trust model (SSTM), each sensor node is associated with a trust metric, which measures to what extent the data transmitted from this node would better be trusted by the server node. Given the SSTM, they translate the trust evaluation problem to be a non-linear state filtering problem. To estimate the state based on the SSTM, a component-wise iterative state inference procedure is proposed to work in tandem with the particle filter (PF), and thus the resulting algorithm is termed as iterative PF (IPF). The computational complexity of the IPF algorithm is theoretically linearly related with the dimension of the state. This property is desirable especially for high-dimensional trust evaluation and state filtering problems. The performance of the proposed algorithm is evaluated by both simulations and real data analysis.

1 Introduction

Wireless sensor networks (WSNs) are networked systems that consist of autonomous nodes collaborating to perform an application task. The nodes of a networked system are usually spatially distributed and equipped with limited sensing, computing and communication capabilities. The research on WSN has gained significant concern in the last decade. The related application domains include but are not limited to health care [1], energy security [2], environmental monitoring [3, 4] and military information integration [5, 6].

The performance of WSN depends on collaboration among distributed sensor nodes, while those nodes are often unattended with severe energy constraints and limited reliability. In such conditions, it is important to evaluate the trustworthiness of participating nodes since trust is the major driving force for collaboration. The focus of this paper is to propose a state space trust model (SSTM) along with a corresponding trust evaluation algorithm in the context of WSN.

The research on trust evaluation has been extensively performed in the context of several diverse domains such as security [7, 8], electronics commerce [9, 10], peer-to-peer networks [11, 12] and *ad hoc* and sensor networks [4, 13–15]. The main objective of the trust evaluation module is to expose an output metric that can be used as a representative of the subjective expectation of the sensor nodes' future behaviours. This trust metric can be used in several ways. For example, the trust value of each node can be used as a weight for a data reading reported by this node. Then, the data fusion can be performed on these weighted data readings, thereby reducing the impact of untrustworthy nodes [14]. In addition, the evolution of trust over time can facilitate online detection of misbehaving nodes. Last but not least, the trust value can be used as a decision making criteria for the end-user to take appropriate measures such as replacing detected faulty nodes. Although various trust models and trust evaluation approaches are available [10, 16–22], there are still many challenges that need to be addressed. It is not clear what are the fundamental rules the trust models must follow; therefore, there is neither a consensus on the definition of trust, nor a common rule for specifying an appropriate trust metric for a given problem. As a result, the design of trust models is still at the empirical stage.

Motivated by the lack of a unified theoretical framework to build up trust models, in this paper, we introduce a generic theoretical model, namely SSTM, in the context of WSN data analysis. We also propose a novel trust evaluation algorithm, termed iterative particle filter (IPF), based on the framework of SSTM. We show that the SSTM framework is extensible and generic, and can include related existing trust evaluation approaches, e.g. the Bayesian dynamic model-based PF (BDMPF) [4], as a special case.

The remainder of this paper is organised as follows. In Section 2, we describe the SSTM. In Section 3, we introduce the proposed IPF algorithm in detail. In Section 4, we report the simulation and real data analysis results in applying IPF for trust evaluations over WSN. Finally, in Section 5, we conclude this paper.

2 Network topology model

We focus on the network topology model as shown in Fig. 1. This model was considered in [4]. The sensor nodes are arranged to sense the environmental parameters and report them to the relay node in real time. The relay node receives the sensor readings from the sensor nodes, and then sends them to a base station that is communicated with a server computer node. All the sensor readings are gathered and analysed at the server computer node. The server computer node is connected with Internet, such that the result of real-time data analysis can be checked remotely by the end-user of the WSN system. Every sensor reading consists of the sensed environmental parameter values and the corresponding sensor ID. Therefore, at the server computer node, we can easily find out the corresponding source sensor node for each sensor reading. The controller nodes receive feedback signal from the computer node, and then control several apparatuses in order to tune the environmental parameters.

3 State Space Trust Model

In this section, we describe the SSTM in detail. We show that, based on the SSTM, trust evaluation over WSN can be formulated as a non-linear state filtering problem.

In SSTM, the trustworthiness of each sensor node is modelled by a trust index, to measure to what extent the data transmitted from this node would better be trusted by the server computer node. The state vector $\boldsymbol{x}_k$ in the SSTM is defined as follows

$$\boldsymbol{x}_k \triangleq [x_{k,1}, x_{k,2}, \ldots, x_{k,\mathrm{d}}] \tag{1}$$

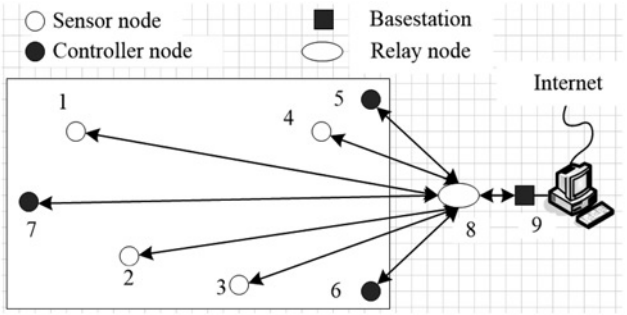

Fig. 1 *WSN network topology model under consideration*

where the element $x_{k,j}$ denotes the trust value of the jth node at the kth time step, and d is the dimension of $\boldsymbol{x}_k$, corresponding to the number of sensor nodes under consideration. The value space of the trust metric $x_{i,j}$ is [0,1], whereby the extreme value 1 means 'fully trusted', and 0 indicates the opposite, that is 'totally un-trusted'.

The trust propagation law over time is modelled by the ageing mechanism as follows [14, 23]

$$\boldsymbol{x}_{k+1} = \alpha \boldsymbol{x}_k + \boldsymbol{v}, \quad \boldsymbol{v} \sim \mathcal{N}(0,\ Q) \tag{2}$$

where $0 < \alpha < 1$ denotes the ageing parameter, $\boldsymbol{Q}$ denotes a diagonal matrix and $\mathcal{N}(0,\ \boldsymbol{Q})$ is a zero-mean Gaussian distribution with covariance $\boldsymbol{Q}$. In [14], the value of α is set to be 0.95. The value of α can also be found out by comparing the evolution of the trust in a system with and without ageing weight, respectively [23]. Note that if the value of $\boldsymbol{x}_{k+1}$ obtained with (2) falls outside of the range [0,1], then we draw a new value of $\boldsymbol{x}_{k+1}$ via (2) until it falls within the range [0,1].

A *generative* model of the sensor readings, which relates the trust metric with the real sensor readings, is specified by the likelihood function. Let $\boldsymbol{y}_k$ denote the data collected by the server computer node at the kth time step. We have $\boldsymbol{y}_k = [y_{k,1},\ y_{k,2},\ \ldots,\ y_{k,\mathrm{d}}]$, where $y_{k,j}$ denotes the data reported by the jth node at the kth time step. The likelihood function is designed to be

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k) = \exp\left(\frac{-\sum_{j=1}^{d} |\boldsymbol{x}_{k,j} - V\left(\{\boldsymbol{x}_{k,n}\}_{n\in\{1:d\}\backslash j},\ \boldsymbol{y}_k,\ j\right)|}{\beta}\right) \tag{3}$$

where $\{1{:}d\}\backslash j$ denotes $\{1,\ \ldots,\ j-1,\ j+1,\ \ldots,\ d\}$, $|A|$ denotes the absolute value of A, $0 < \beta < 1$ is a free parameter and

$$V\left(\{x_{k,n}\}_{n\in\{1:d\}\backslash j},\ y_k,\ j\right) \triangleq \frac{\sum_{n\in\{1:d\}\backslash j} x_{k,n} U(n,\ j,\ y_k)}{\sum_{n\in\{1:d\}\backslash j} x_{k,n}} \tag{4}$$

Equation (4) describes the computation of the voting metric of the jth node, given by the other nodes. The item $U(n,\ j,\ y_k)$ in (4) denotes the voting result node n gives to j, and is defined to be

$$U(n,\ j,\ y_k) = \begin{cases} 1 & \text{if } |y_{k,n} - y_{k,j}| < r \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where r denotes a preset threshold for determining whether a pair of nodes reports sensor readings with permissible differences. The underlying assumption adopted in defining $U(n,\ j,\ y_k)$ is that sensor readings reported by mutually trusted sensor nodes should not have significant difference. This assumption is reasonable for many WSN applications, wherein the nodes are mutual spatial neighbours among with each other and the trusted sensor readings should be spatially correlated with each other. This assumption is also in analogy with the social trust, wherein mutually trusted social entities report similar opinions (data) on an object or event in an *ad hoc* context. In (4), $U(n,\ j,\ y_k)$ is weighted by $(x_{k,n}/\sum_{i\in\{1:d\}\backslash j} x_{k,i})$, for each $n \in \{1{:}d\}\backslash j$. In such a way, the impact of each node is adjusted according to its trustworthiness, and thus the impact of untrustworthy nodes is reduced, in generating the final voting metric of node j, i.e. $V\left(\{x_{k,n}\}_{n\in\{1:d\}\backslash j},\ y_k,\ j\right)$

Given the data collected by the server computer node until the kth time step, denoted by $\boldsymbol{y}_{1:k} \triangleq \{\boldsymbol{y}_1,\ \ldots,\ \boldsymbol{y}_k\}$, we are right now concerned with the calculation of the *posterior* probability density function (pdf) $\pi_k = p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$ in a Bayesian inference framework.

According to Bayesian philosophy [24], given $\boldsymbol{y}_{1:k}$, all the information on x_k is encoded by the *posterior*, as long as the prior pdf and the likelihood function are specified appropriately. Particle filters (PFs), a.k.a. sequential Monte Carlo methods, are recognised as a general approach to address such a Bayesian state estimation problem [25–27]. In comparison to other state filtering algorithms such as the Kalman filter and its variants, PFs have striking advantages in coping with non-linearities and/or non-Gaussian noises in the model [26, 28, 29]. However, as a Monte Carlo method, the PF algorithm inevitably suffers from the well-known curse of dimensionality, that is, the PF may collapse in case of high-dimensional state vector [30–32].

Regarding our problem at hand, the dimension of the state, i.e. d, is equal to the number of sensor nodes under consideration. If the network of our concern consists of massive sensor nodes densely arranged, the corresponding state will become high dimensional, thus the conventional PF algorithms may become invalid. We propose in Section 4 a novel PF algorithm, namely IPF, to get around of the above computation problem caused by high dimensionality.

4 Iterative PF

In this section, we introduce the proposed IPF algorithm in detail. To begin with, we give a brief review on a conventional PF algorithm, termed bootstrap PF, to fix the notations.

4.1 Bootstrap PF

The bootstrap PF is a general practical non-linear state filter, which typically proceeds by Monte Carlo approximation. This algorithm has a recursive structure in its implementation, thus it allows the state filter to be computed online over a long time horizon. The recursion is at the level of probability measures, and the target distribution π_k is approximated by the empirical distribution $\hat{\pi}_k$. The distribution $\hat{\pi}_k$ is then computed by the recursion

$$\hat{\pi}_0 = \pi_0, \quad \hat{\pi}_k = F_k \hat{\pi}_{k-1} \tag{6}$$

where π_0 denotes a *prior* belief on the state and F_k denotes an operator that consists of two steps

$$\hat{\pi}_{k-1} \xrightarrow{\text{prediction}} \hat{\pi}_{k-} \xrightarrow{\text{correction}} \hat{\pi}_k \tag{7}$$

The empirical distribution $\hat{\pi}_{k-1}$ is the output of the algorithm at the $k-1$th ($k>1$) time step, and is represented as

$$\hat{\pi}_{k-1} = \frac{1}{N}\sum_{i=1}^{N} \sigma_{\boldsymbol{x}_{k-1}^i} \tag{8}$$

where $N \geq 1$ is the number of particles used in the algorithm, $(\boldsymbol{x}_{k-1}^i)_{i=1,\ldots,N}$ are independent identically distributed (i.i.d.) samples from $\hat{\pi}_{k-1}$, and σ_x denotes the delta function located at x.

In the prediction step, a set of new particles $\{\boldsymbol{x}_{k-}^i\}_{i=1,\ldots,N}$ is generated according to the state transition law, i.e. (2) for the problem

Algorithm 1

For $i = 1, \ldots, N$, sample an index $j(i)$ distributed according to the discrete distribution with N elements satisfying $\Pr\{j(i) = l\} = w^i$;
For $i = 1, \ldots, N$, let $\boldsymbol{x}_k^i = \boldsymbol{x}_{k-}^{j(i)}$, $w_i = 1/N$;
Output $\{\boldsymbol{x}_k^i\}_{i=1,\ldots,N}$.

Fig. 2 *Resampling algorithm with input* $\{\boldsymbol{x}_{k-}^i,\ w^i\}_{i=1,\ldots,N}$

Algorithm 2

Let $\hat{\pi}_0 = \pi_0$;
for $k = 1, \ldots, K$ **do**
  Sample i.i.d. $\boldsymbol{x}_{k-1}^i$, $i = 1, \ldots, N$ from the distribution $\hat{\pi}_{k-1}$;
  Sample $\boldsymbol{x}_{k-}^i \sim p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^i)$, $i = 1, \ldots, N$ using Eqn.(2);
  Calculate the importance weights $w^i, i = 1, \ldots, N$, using Eqn.(10);
  Run **Algorithm 1** with input $\{\boldsymbol{x}_{k-}^i, w^i\}_{i=1,\ldots,N}$, and get $\{\boldsymbol{x}_k^i\}_{i=1,\ldots,N}$;
  Let $\hat{\pi}_k = \frac{1}{N}\sum_{i=1}^N \sigma_{\boldsymbol{x}_k^i}$;

Fig. 3 *Bootstrap PF*

of our concern. Specifically, we have

$$\boldsymbol{x}_{k-}^i = \alpha \boldsymbol{x}_{k-1}^i + \boldsymbol{v},\ \boldsymbol{v} \sim \mathcal{N}(0,\ \boldsymbol{Q}),\ i = 1,\ \ldots,\ N \tag{9}$$

Note that the value of $\boldsymbol{x}_{k-}^i$ needs to be bounded within [0,1]. Provided that its value jumps outside of the bounded space [0,1], we just generate a new value for $\boldsymbol{x}_{k-}^i$ using (2).

These particles $\{\boldsymbol{x}_{k-}^i\}_{i=1,\ldots,N}$ are then weighted in the correction step. The weights are termed importance weights in the context of PF, and are calculated as follows:

$$w^i = \frac{\bar{w}^i}{\sum_{i=1}^N \bar{w}^i},\ i = 1,\ \ldots,\ N \tag{10}$$

where

$$\bar{w}^i = p(y_k|\boldsymbol{x}_{k-}^i),\ i = 1,\ \ldots,\ N \tag{11}$$

Then let $\hat{\pi}_k = \sum_{i=1}^N w^i \delta_{\boldsymbol{x}_{k-}^i}$. In bootstrap PF, a resampling procedure is included to prevent the phenomenon of particle degeneracy, that is, more and more particles get zero weights and are lost. The basic operations of resampling are described in Fig. 2.

A summarisation of the bootstrap PF is described in Fig. 3, where K denotes the total number of time steps under consideration.

It is shown that the empirical distribution $\hat{\pi}_k$ converges to the exact target distribution π_k as $N \to \infty$ [33, 34]. We see [26] for a detailed overview of PF algorithms and the related analysis.

4.2 Derivation of the IPF algorithm

Since conventional PF algorithms such as the bootstrap PF presented in Section 4.1 have inevitable drawbacks in dealing with filtering problems with high-dimensional state vectors [30–32], here we derive a novel IPF algorithm, based on the specific structure of our model, to get around of the obstacles resulted from high dimensionality.

Observe that the likelihood function constructed in (3) can be factorised as follows

$$p(\boldsymbol{y}_k|\boldsymbol{x}_k) = \prod_{j=1}^d \exp\left(\frac{-|\boldsymbol{x}_{k,j} - V\left(\{\boldsymbol{x}_{k,n}\}_{n\in\{1:d\}/j},\ \boldsymbol{y}_k,\ j\right)|}{\beta}\right) \tag{12}$$

Therefore, conditional on $\{\boldsymbol{x}_{k,n}\}_{n\in\{1,\ldots,d\}/j}$, we can calculate the likelihood of $\boldsymbol{x}_{k,j}$ as follows

$$p(\boldsymbol{y}_k|\boldsymbol{x}_{k,j}|\{\boldsymbol{x}_{k,n}\}_{n\in\{1,\ldots,d\}/j}) = \exp\left(\frac{-|\boldsymbol{x}_{k,j} - V\left(\{\boldsymbol{x}_{k,n}\}_{n\in\{1:d\}/j}, \boldsymbol{y}_k, j\right)|}{\beta}\right) \tag{13}$$

On the basis of (2), the state transition law of $\boldsymbol{x}_{k,j}$ can be shown to be

$$\boldsymbol{x}_{k+1,j} = \alpha \boldsymbol{x}_{k,j} + v_j,\ v_j \sim \mathcal{N}(0, \boldsymbol{Q}_{jj}) \tag{14}$$

where $Q_{j,j}$ denotes the jth diagonal element of matrix $\boldsymbol{Q}$.

Given the component-wise likelihood and state transition function, specified by (13) and (14), respectively, we can accordingly calculate the component-wise *posterior*, which is only a one-dimensional (1D) distribution and thus is very easy to be sampled from.

The basic idea of IPF is that, at each time step, instead of sampling straightforwardly from the high-dimensional *posterior* (such as in conventional PF), we perform component-wise inferences by sampling from a set of component-wise *posterior* pdfs, and then update the estimate of the trust iteratively. Specifically, the component-wise inference operations are described in Fig. 4, wherein $\|\boldsymbol{A} - \boldsymbol{B}\|$ denotes the Euclidean distance between the two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$, and $\boldsymbol{T}_x$ denotes a preset threshold for determining if values of a pair of trust vectors have significant difference with each other.

Finally, the IPF is described in Fig. 5. Although the proposed IPF algorithm has an iterative component, our experiments in Section 5 (see Fig. 8) show that it needs just a few iterations in order to converge.

4.3 Connections to existing work

The proposed IPF algorithm has a close connection to the BDMPF algorithm [4]. Both algorithms are developed within the Bayesian state filtering framework, while their essential difference lies in the design of the model. In BDMPF, the voting metric of the jth node, given by the other nodes, is computed as follows

$$V\left(\{\boldsymbol{x}_{k,n}\}_{n\in\{1:d\}\setminus j},\ \boldsymbol{y}_k,\ j\right) \triangleq \frac{\sum_{n\in\{1:d\}\setminus j} U(n,\ j,\ \boldsymbol{y}_k)}{d-1} \tag{15}$$

In comparison with (4), we see that (15) is equivalent to (4) in case of $\boldsymbol{x}_{k,n} = 1$ for any $n \in \{1{:}d\}\setminus j$. In another word, in calculating the voting metric of node j, BDMPF assumes that all the other sensor nodes are all completely trusted. Clearly, such an assumption is easy to be violated in practise.

In addition, regarding BDMPF and IPF, the difference in their model structures leads to a corresponding difference in the related inference algorithms. In the inference process, the IPF algorithm employs the fact that $x_{k,1}, x_{k,2}, \ldots, x_{k,d}$ are correlated with each other and thus should be estimated jointly, while the BDMPF assumes that $x_{k,1}, x_{k,2}, \ldots, x_{k,d}$ are independent with each other, thus are estimated separately.

Algorithm 3

Input: $\hat{\pi}_{k-1}$;
Initialise X_o to be a d dimensional vector with all elements being 0;
Let $m = 1$ (m denotes the iteration index);
while $m = 1$ *or* $\sqrt{\|\hat{\boldsymbol{x}}_k - X_o\|/d} > T_x$ *(T_x denotes a preset threshold)* **do**
  if $m > 1$ **then**
    Let $X_o = \hat{\boldsymbol{x}}_k$;
  for $j = 1, \ldots, d$ **do**
    Sample i.i.d. $x^i_{k-1,j}, i = 1, \ldots, N$ from the distribution $\hat{\pi}_{k-1,j}$;
    Sample $x^i_{k-,j} \sim p(x_{k,j}|x^i_{k-1,j}), i = 1, \ldots, N$ using Eqn.(14);
    Calculate the importance weights $\bar{w}^i = p(\mathbf{y}_k|x^i_{k-,j}|\{\hat{\boldsymbol{x}}_{k,n}\}_{n\in\{1,\ldots,d\}/j}), i = 1, \ldots, N$, using Eqn.(13);
    Normalise the importance weights by $w^i = \bar{w}^i / \sum_{u=1}^N \bar{w}^u, i = 1, \ldots, N$;
    Run **Algorithm 1** with input $\{x^i_{k-,j}, w^i\}_{i=1,\ldots,N}$, and get $\{x^i_{k,j}\}_{i=1,\ldots,N}$;
    Let $\hat{\pi}_{k,j} = \frac{1}{N}\sum_{i=1}^N \sigma_{x^i_{k,j}}$;
    Update the jth dimension of $\hat{\boldsymbol{x}}_k$ by $\hat{\boldsymbol{x}}_{k,j} = \frac{1}{N}\sum_{i=1}^N x^i_{k,j}$;
  Let m=m+1;
Output $\hat{\pi}_k$ and $\hat{\boldsymbol{x}}_k$.

Fig. 4 *Iterative component-wise inference within the IPF at time step k*

Therefore, the proposed IPF algorithm is preferable to BDMPF for WSN applications, wherein the trustworthy sensor readings are statistically correlated with each other and are independent with those yielded by un-trusted nodes. The empirical results presented in Section 5 are consistent with the above analysis.

5 Performance evaluation

In this section, we present performance evaluation results of the proposed IPF algorithm based on simulations and real data analysis.

5.1 Simulation results

We tested our algorithm based on the simulation case that was used in [4].

5.1.1 Simulation setting: In this case, we have ten sensor nodes involved, each of which reports its sensor reading to the server computer node at 100 discrete time steps. The network topology related with this simulation case is the same as shown in Fig. 1. The values of trustworthy sensor readings are simulated to be normally distributed centring at 20°C at each time step. Among the sensor nodes, seven of them are trustworthy as they transmit normal sensor readings from beginning to end. The remaining nodes, indexed by 'Sensor A', 'Sensor B' and 'Sensor C', have different types of unreliability in their behaviour. Specifically, 'Sensor A' is simulated to be unreliable from 31st to 70th time steps, during which the sensor reading value it transmits rises gradually from 20 to 40°C between the 30th and 50th time steps, and then falls back gradually to 20°C between the 50th and 70th time steps. The sensor reading of 'Sensor B' is simulated to be uniformly distributed between 0 and 100°C at each time step. 'Sensor C' is simulated to work normally from 1st to 50th time step and then stop reporting any values afterwards. This phenomenon is termed as 'Sleeper Attack' in [14].

5.1.2 Performance comparison with the BDMPF algorithm: We compared our IPF algorithm with the BDMPF algorithm proposed in [4] by Monte Carlo simulations. We ran 100 times of independent Monte Carlo runs of the IPF algorithm and the BDMPF algorithm. These two algorithms were initialised by the same parameter setting as shown in Table 1. At the beginning, the trust metric of every sensor node was set to be 0.5.

The estimated traces of the trust metric of 'Sensor A', 'Sensor B', 'Sensor C' and an always-trustworthy sensor node, termed 'Sensor D' here, are plotted in Figs. 6–9, respectively.

First, let us analyse the estimation result on 'Sensor A'. According to the simulation setting described in Section 5.1.1, the trust metric of 'Sensor A' takes the value of 1 in two time periods, corresponding to the 1–30 and 71–100 time steps, and it takes the value of 0 at the other, namely the 31–70 time steps. In Fig. 2, we see that within the first ten time steps, the estimated trust metric of 'Sensor A' given by the IPF algorithm converges to the expected value 1 quickly, while the estimated trust metric given by the BDMPF algorithm converges to 0.8. During the 31–70 time steps, the estimate given by the IPF algorithm converges again to the expected value 0 quickly, while the BDMPF algorithm converges to a value close to 0.1. Regarding the last 30 time steps, it is shown that the IPF algorithm can still output much accurate estimate on the trust metric, while the performance of BDMPF deteriorates much more, since the gap between its estimate and the true answer is broadened. The similar result of that the estimate given by IPF is much more accurate than that given by BDMPF can also be found in Figs. 3–5, for 'Sensors B, C and D', respectively. Furthermore, we can see that the performance gap between the BDMPF and the IPF algorithms is broadened when more sensor nodes become untrustworthy. For example, in Fig. 5, we see that, for this always-trustworthy node 'Sensor D', the estimated trust metric given by BMDPF worsens along with the increase in the number of existing untrustworthy nodes. Specifically, we can see that at 51–70 time steps, during which 'Sensors A, B and C' are all untrustworthy, the BDMPF algorithm gives the worst estimate

Algorithm 4

Let $\hat{\pi}_0 = \pi_0$;
for $k = 1, \ldots, K$ **do**
  Run Algorithm 3 with input $\hat{\pi}_{k-1}$. Output $\hat{\pi}_k$ and $\hat{\boldsymbol{x}}_k$.

Fig. 5 *The proposed IPF algorithm*

Table 1 Parameter setting of the IPF algorithm in the Monte Carlo simulation test

N	α	Q	β	r	T_x
100	0.85	diag([0.01, …, 0.01])	0.1	0.6	1×10^{-5}

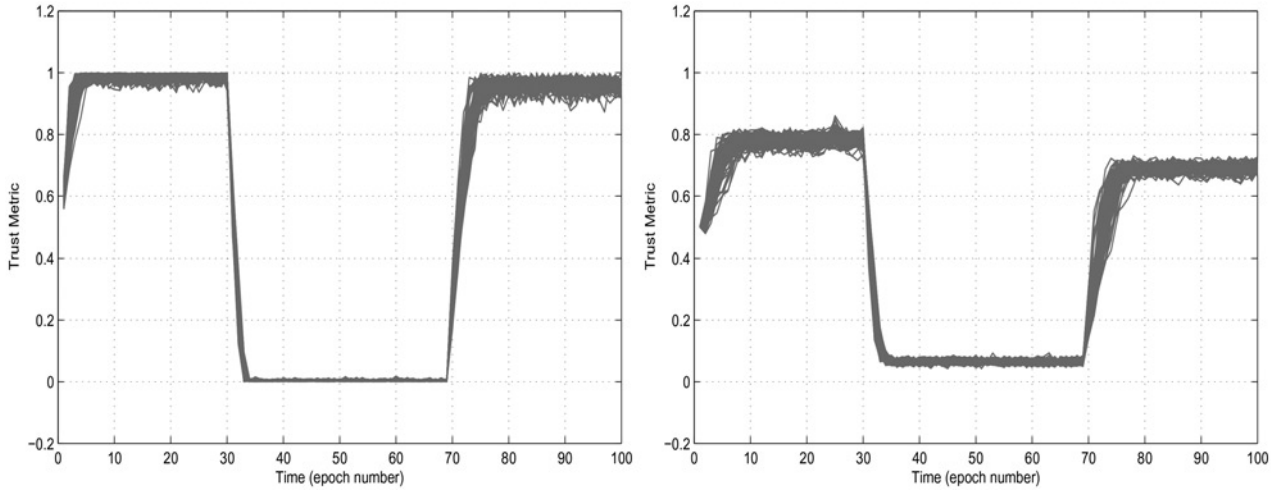

Fig. 6 *Left: traces of the estimated trust metric of 'Sensor A' in 100 independent Monte Carlo runs of the IPF algorithm. Right: traces of the estimated trust metric of 'Sensor A' in 100 independent Monte Carlo runs of the BDMPF algorithm*

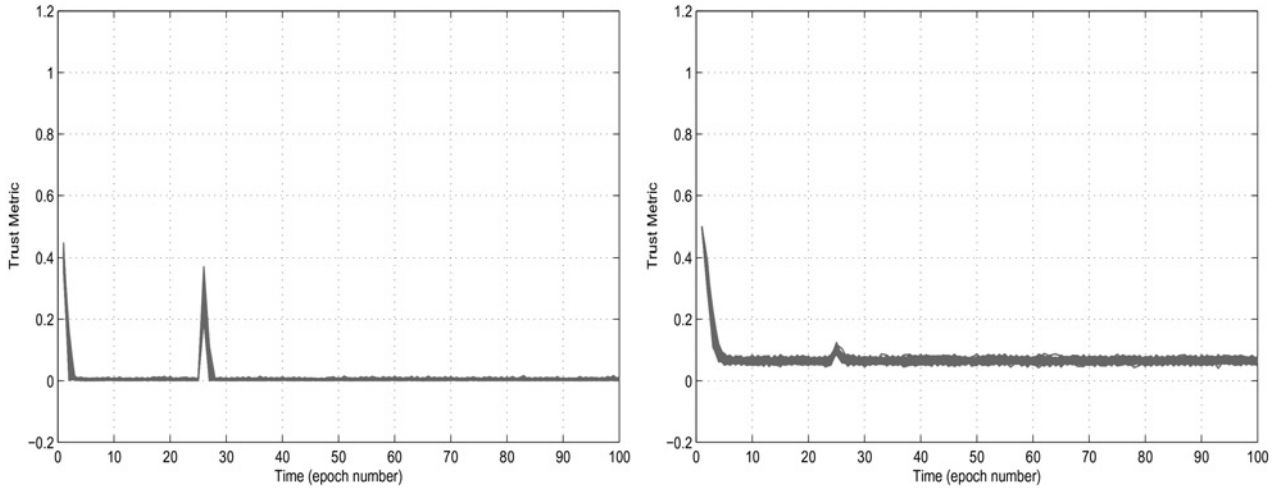

Fig. 7 *Left: traces of the estimated trust metric of 'Sensor B' in 100 independent Monte Carlo runs of the IPF algorithm. Right: traces of the estimated trust metric of 'Sensor B' in 100 independent Monte Carlo runs of the BDMPF algorithm*

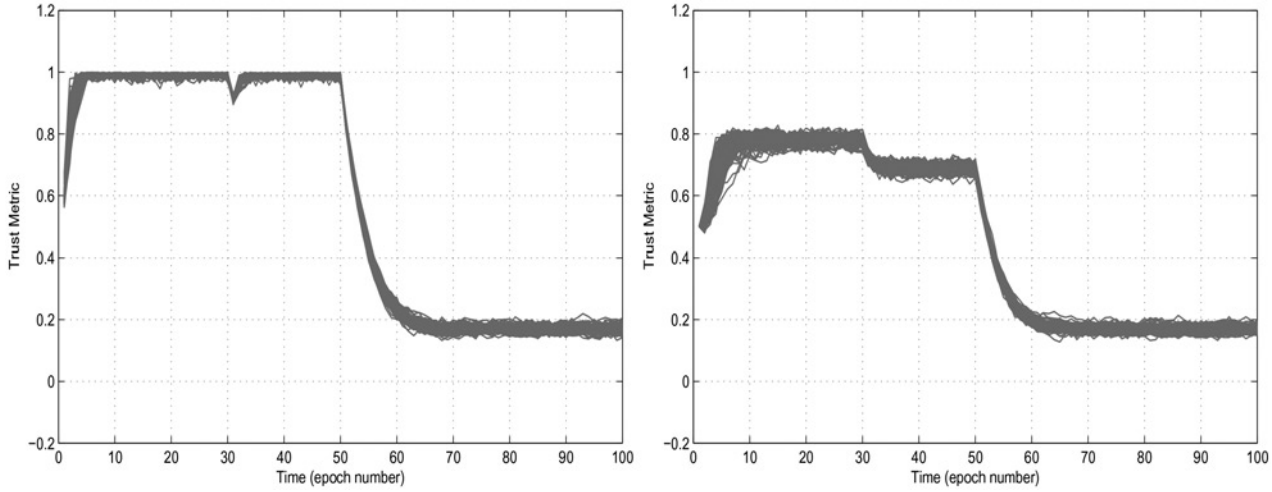

Fig. 8 *Left: traces of the estimated trust metric of 'Sensor C' in 100 independent Monte Carlo runs of the IPF algorithm. Right: traces of the estimated trust metric of 'Sensor C' in 100 independent Monte Carlo runs of the BDMPF algorithm*

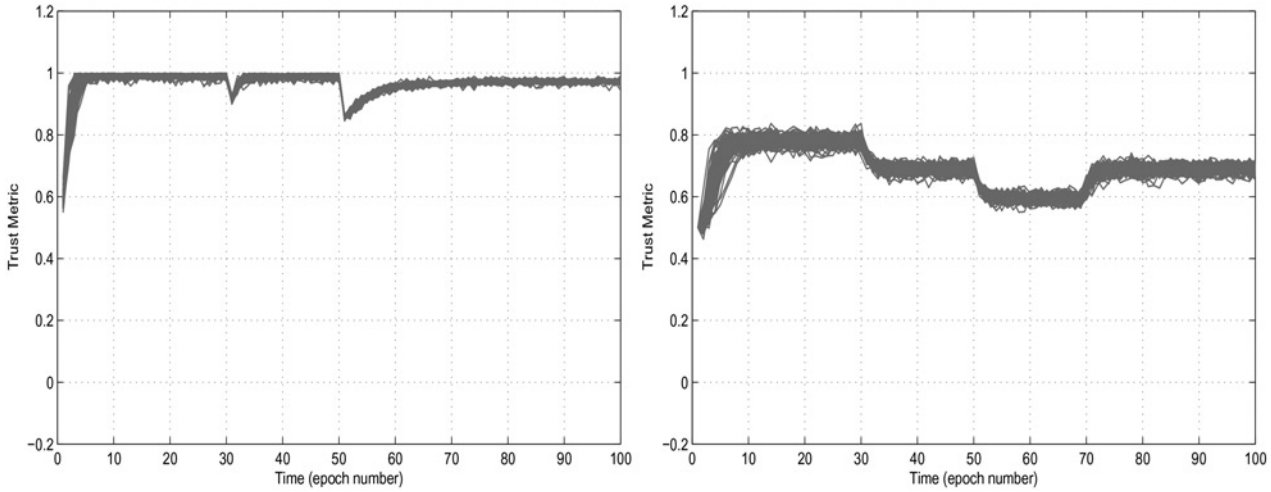

Fig. 9 *Left: traces of the estimated trust metric of 'Sensor D' in 100 independent Monte Carlo runs of the IPF algorithm. Right: traces of the estimated trust metric of 'Sensor D' in 100 independent Monte Carlo runs of the BDMPF algorithm*

of the trust metric compared with the other periods. In contrast with BDMPF, the proposed IPF algorithm always provides an accurate estimate on the trust metric of 'Sensor D' in Fig. 5. In another words, the IPF algorithm is shown to be remarkably much more robust than BDMPF in case of untrustworthy nodes being involved. The above result is consistent with the theoretical analysis on the connections between the IPF and the BDMPF algorithms as described in Section 4.3.

5.1.3 Numerical performance evaluation: For ease of quantitative performance evaluation, we used the root mean square error (RMSE) to measure the gap between the estimate provided by an algorithm and the true answer. The RMSE regarding node j at time step k is defined to be

$$\mathrm{RMSE}_{k,j} \triangleq \sqrt{\frac{\sum_{m=1}^{M}(\hat{x}_{k,j}^{m} - x_{k,j})^2}{M}} \tag{16}$$

where M denotes the total number of independent runs of the algorithm of our concern in the Monte Carlo simulation test, $\hat{x}_{k,j}^{m}$ denotes the estimate of $x_{k,j}$ yielded in the mth independent run of the algorithm. In what follows we set $M = 100$.

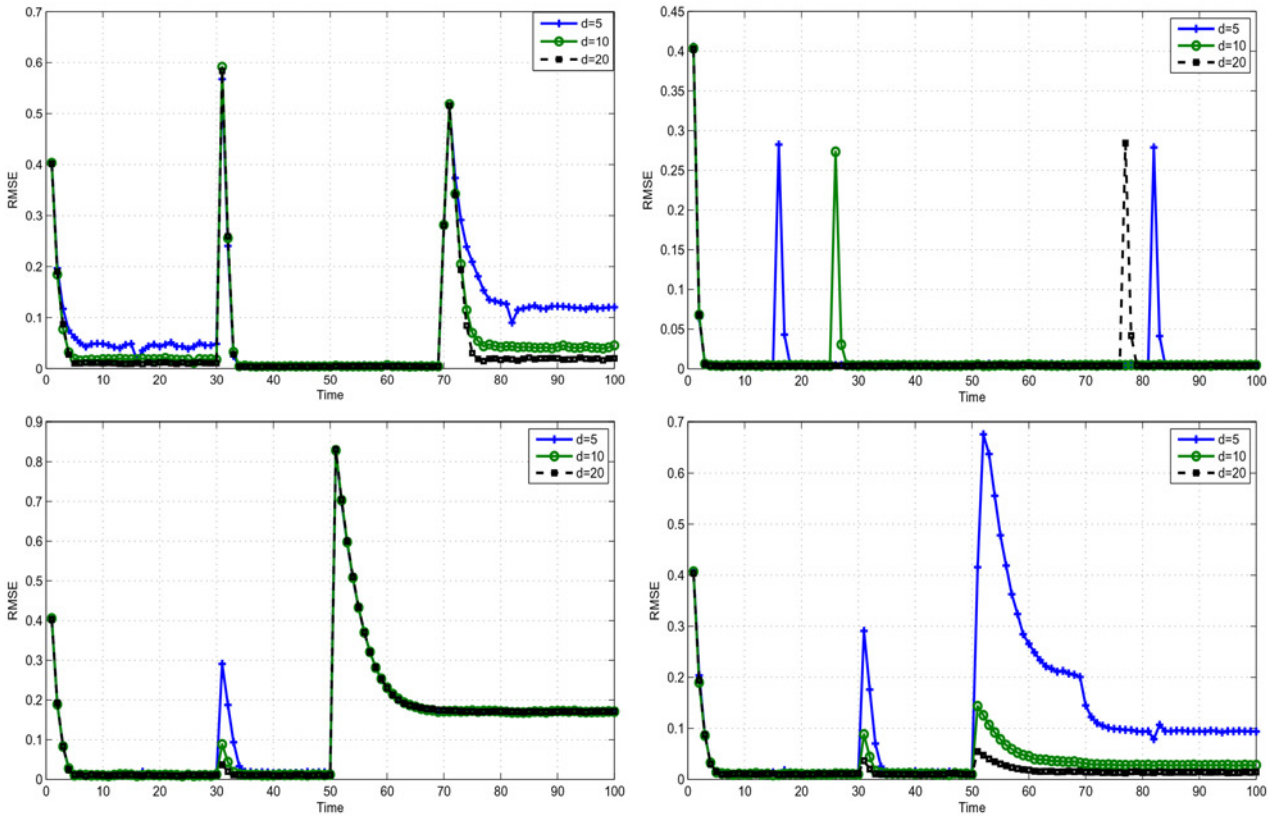

Fig. 10 *RMSE calculated based on simulation results obtained from 100 independent Monte Carlo runs of the IPF algorithm. Here, d denotes the total number of sensor nodes under consideration. The top left, top right, bottom left and bottom right sub-figures correspond to 'Sensors A, B, C and D', respectively*

doi: 10.1049/joe.2016.0373

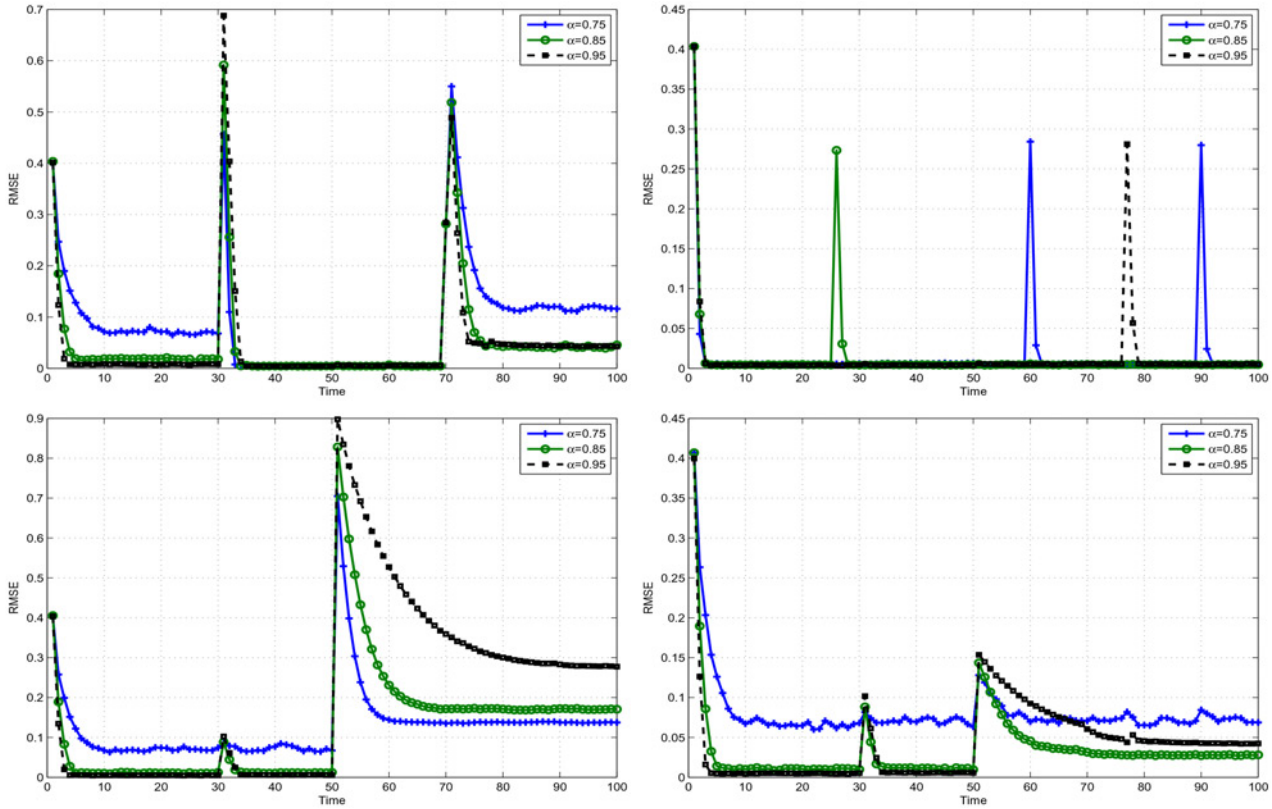

Fig. 11 *RMSE calculated based on simulation results obtained from 100 independent Monte Carlo runs of the IPF algorithm under cases with different α values. The top left, top right, bottom left and bottom right sub-figures correspond to 'Sensors A, B, C and D', respectively*

We investigated how the performance of IPF changes along with the dimension of the state d. We considered three cases, corresponding to $d = 5$, 10 and 20, respectively. 'Sensors A, B and C' with the same setting as before are involved for all cases. For each specific d value, we ran 100 independent Monte Carlo runs of the IPF algorithm, and then calculated the corresponding RMSE. The result is shown in Fig. 10, where we use 'Sensor D' to denote a representative always-trustworthy node as before. It is shown that, as d gets bigger, the RMSE gets smaller, and, even in case of $d = 5$, most of the time the RMSE does not exceed 0.12.

We also investigated the influence of the ageing parameter α in (2) on the performance of the IPF algorithm. We considered three cases corresponding to $\alpha = 0.75$, 0.85 and 0.95. For each case, we set $d = 10$, and ran 100 independent Monte Carlo runs of the IPF algorithm. The results are shown in Fig. 11. We see that, most of the time, the RMSE corresponding to $\alpha = 0.85$ and 0.95 is smaller than that corresponding to $\alpha = 0.75$, for all the sensor nodes under consideration. In the bottom left sub-figure, we see that 0.85 is more preferable to 0.75 in initialising α. Actually 0.85 is selected empirically as the default value of α in our algorithm.

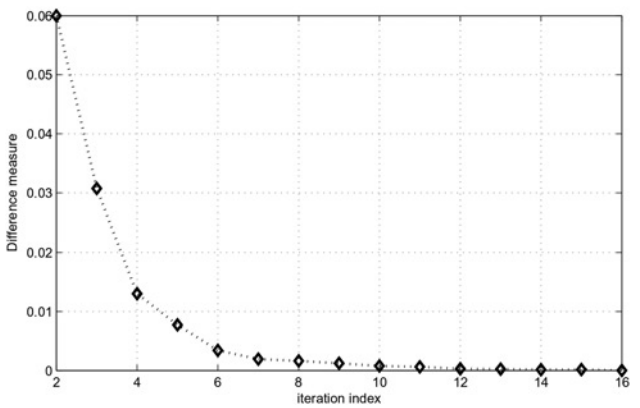

Fig. 12 *Convergence of the component-wise inference procedure in the IPF algorithm. The X and Y labels of the figure denote the iteration index m and* $\sqrt{\|\hat{x}_k - X_o\|/d}$ *in Fig. 4, respectively*

5.1.4 Investigation on the iterative component and the computational burden of the IPF algorithm: The proposed IPF algorithm includes an iterative process, namely the component-wise inference procedure, as shown in Fig. 4, while our experiments show that it needs just a few iterations in order to converge, see Fig. 12. The computational time of the IPF algorithm in three cases, corresponding to $d = 5$, 10 and 20, respectively, is presented in Table 2. So experimentally, we see that the computational burden of the IPF algorithm is linearly related with the dimension of the state d. Such a good scaling property of our algorithm is especially desirable when dealing with high-dimensional cases.

5.2 *Real data analysis results*

Here we describe an evaluation performed on the Intel Lab Data [35], a public data set collected from 54 sensors deployed in the

Table 2 Elapsed time of an independent run of the IPF algorithm. T_{elapsed} denotes the real value of the elapsed time. T_{scaled} denotes the scaled version of T_{elapsed}, calculated on the basis of the 5D case

d	5	10	20
T_{elapsed}, s	71.5	124.6	281.3
T_{scaled}	1	1.7	3.9

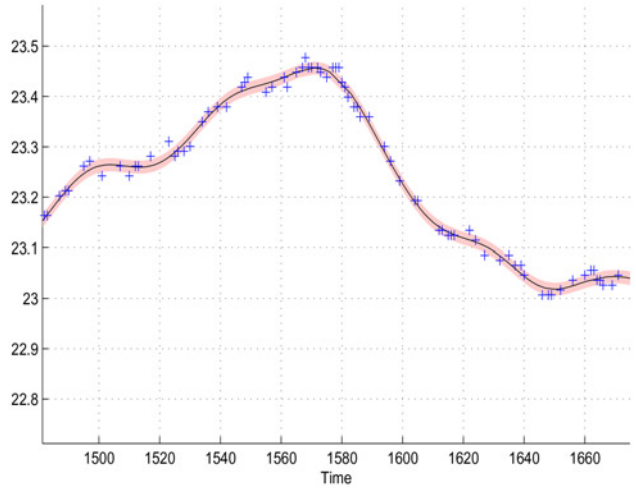

Fig. 13 *Fitting sensor readings with Gaussian process regression. The shadow depicts the one standard error uncertainty associated with the fitted curve based on the sensor readings, represented by the plus signs*

Intel Berkeley Research laboratory between 28th February and 5th April 2004. In our experiment, we chose the whole day's data from February 28th, remaining only the sensor reading attribute of original data set, i.e. humidity, temperature and light. We selected a spatial neighbour set of sensors 9, 10, 11, 12 and 13 for analysis. As the sampling time of the sensor readings reported by different sensors is not synchronous, we performed Gaussian process regression [36] to fit the readings of each sensor. A snapshot of the fitting effect is depicted in Fig. 13.

We adopted the fault models described in [14] to simulate faulty sensor readings, which are then injected into the original data. The purpose is to evaluate whether the IPF algorithm can detect such faults in time through estimating the trust metric of each sensor online. Specifically, for the first node under consideration, we removed its reported data between the 500th and 700th epoch to simulate the phenomenon termed 'Sleeper Attacks' [14]. For the second node, we modified its sensor readings between the 300th and 400th epoch to be a constant 100. This phenomenon is called 'Stuck-at Fault' in [14]. For the third node, we added a zero-mean Gaussian noise with standard error 20, to each of its sensor readings between the 200th and 250th epoch, and this is the so-called 'variance degradation fault' described in [14]. For the fourth node, we added an offset value, 100, to its pre-fault measurement values between the 100th to the 150th epoch with a probability 0.5. This type of fault is termed 'offset fault' in [14]. The IPF algorithm is initialised by the same parameter values as shown in Table 1, except that we empirically set $r = 2$ here.

The estimated trust metric for the above-mentioned sensor nodes is graphically presented in Fig. 14. As is shown, the estimated trust metric, given by our IPF algorithm, can accurately reflect the existence of different types of faults online. Thus, the IPF algorithm can be regarded as an efficient fault detection tool.

6 Conclusions

In this paper, we propose a theoretical data-driven modelling framework to address the problem of trust evaluation over WSN. The basic idea is to treat the problem of trust evaluation from the perspective of non-linear state filtering. In particular, we design a generic trust model, termed SSTM. Then, making use of the information on the model structure, we design a corresponding state

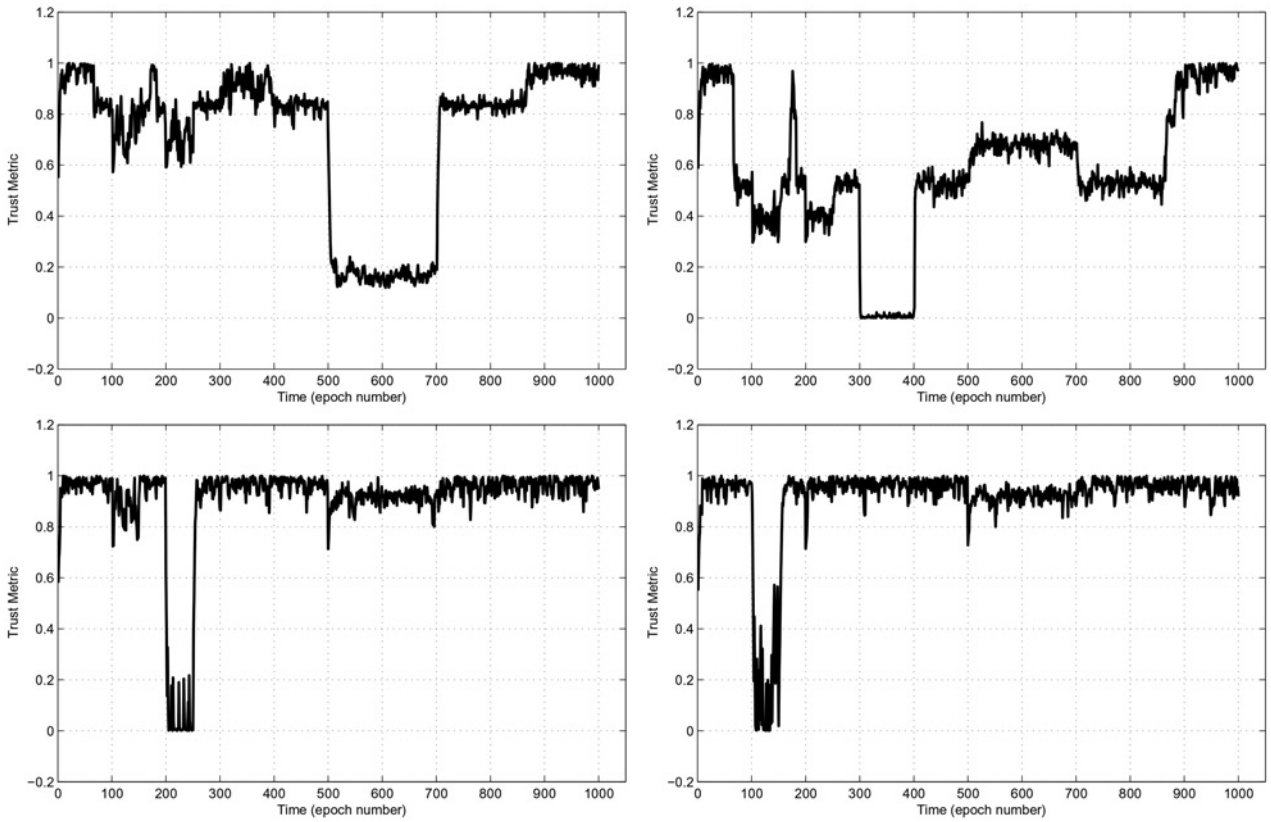

Fig. 14 *Trust evaluation in the presence of faults in the Intel lab data. The top left, top right, bottom left and bottom right sub-figures correspond to the first sensor node with 'Sleeper Attacks' between the 500th and 700th epoch, the second sensor node with 'Stuck-at Fault' between the 300th and 400th epoch, the third sensor node with 'variance degradation fault' between the 200th and 250th epoch and the fourth sensor node with 'offset fault' between the 100th and 150th epoch*

filtering algorithm, termed IPF. Through both extensive simulation studies and real data analysis, we evaluated the performance of the IPF algorithm. The results show that it can yield accurate estimate on the trust metric of the sensor nodes online, even in complex environments, wherein different types of non-trustworthy nodes exist and report different types of faulty measurements to the server node. The computational complexity of the proposed algorithm is shown to be linearly related with the dimension of the state. Such a scaling property makes our algorithm easy to meet the practical constraints in energy, memory and computation power, especially when we have a lot of sensor nodes waiting to be evaluated concurrently. The future work is planned to compare the proposed algorithm with alternatives in the aspects of consumptions in energy, memory and computation power. In addition, by virtue of Bayesian decision making theory, the proposed framework here can be generalised to deal with risk analysis and decision making issues.

7 Acknowledgments

This work was partly supported by the National Natural Science Foundation of China (NSFC) under grant no. 61571238, the China Postdoctoral Science Foundation under grant nos. 2015M580455 and 2016T90483, the China Postdoctoral International Academic Exchange Program and National NSFC under grant no. 91646116, Scientific Research Program funded by Shaanxi Provincial Education Department, P. R. China, under Grant 15JK1310, and the Fundamental Research Funds for the Central Universities under Grant GK201703062.

8 References

[1] Otto C., Milenkovic A., Sanders C., *ET AL.*: 'System architecture of a wireless body area sensor network for ubiquitous health monitoring', *J. Mob. Multimedia*, 2006, **1**, (4), pp. 307–326

[2] León R.A., Vittal V., Manimaran G.: 'Application of sensor network for secure electric energy infrastructure', *IEEE Trans. Power Deliv.*, 2007, **22**, (2), pp. 1021–1028

[3] Werner-Allen G., Lorincz K., Ruiz M., *ET AL.*: 'Deploying a wireless sensor network on an active volcano', *IEEE Internet Comput.*, 2006, **10**, (2), pp. 18–25

[4] Liu B., Xu Z., Chen J., *ET AL.*: 'Toward reliable data analysis for Internet of things by Bayesian dynamic modeling and computation'. Proc. of IEEE China Summit and International Conf. on Signal and Information Processing (ChinaSIP), 2015, pp. 1027–1031

[5] Lee S.H., Lee S., Song H., *ET AL.*: 'Wireless sensor network design for tactical military applications: remote large-scale environments'. Proc. of IEEE Military Communications Conf. (MILCOM), 2009, pp. 1–7

[6] Diamond S.M., Ceruti M.G.: 'Application of wireless sensor network to military information integration'. Proc. of Fifth IEEE International Conf. on Industrial Informatics, IEEE, 2007, vol. 1, pp. 317–322

[7] Blaze M., Feigenbaum J., Ioannidis J., *ET AL.*: 'The role of trust management in distributed systems security', in Vitek J., Jensen, C.D. (Eds.): 'Secure Internet Computing: Security Issues for Mobile and Distributed Objects' (Springer, Berlin and Heidelberg, Germany, 1999), pp. 185–210

[8] Blaze M., Feigenbaum J., Lacy J.: 'Decentralized trust management'. Proc. of IEEE Symp. on Security and Privacy, 1996, pp. 164–173

[9] Jøsang A., Ismail R., Boyd C.: 'A survey of trust and reputation systems for online service provision', *Decis. Support Syst.*, 2007, **43**, (2), pp. 618–644

[10] Resnick P., Zeckhauser R.: 'Trust among strangers in Internet transactions: empirical analysis of ebays reputation system', *Econ. Internet E-comm.*, 2002, **11**, (2), pp. 23–25

[11] Selcuk A.A., Uzun E., Pariente M.R.: 'A reputation-based trust management system for p2p networks'. Proc. of IEEE International Symp. on Cluster Computing and the Grid (CCGrid), 2004, pp. 251–258

[12] Singh A., Liu L.: 'Trustme: anonymous management trust relationships in decentralized p2p systems'. Proc. Third International Conf. on Peer-to-Peer Computing, 2003, pp. 142–149

[13] Michiardi P., Molva R.: 'Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks', in Jerman-Blazic B., Klobucar T. (Eds.): 'Advanced communications and multimedia security' (Springer, New York, NY, USA, 2002), pp. 107–121

[14] Ganeriwal S., Balzano L.K., Srivastava M.B.: 'Reputation-based framework for high integrity sensor networks', *ACM Trans. Sens. Netw. (TOSN)*, 2008, **4**, (3), p. 15

[15] Wang J., Liu B.: 'Online fault-tolerant dynamic event region detection in sensor networks via trust model'. Proc. of IEEE Wireless Communications and Networking Conf. (WCNC), 2017

[16] Gambetta D.: 'Can we trust trust?', 'Trust: Making and breaking cooperative relations', (2000), **13**, pp. 213–237

[17] Theodorakopoulos G., Baras J.S.: 'Trust evaluation in ad-hoc networks'. Proc. of the Third ACM Workshop on Wireless Security, 2004, pp. 1–10

[18] Jøsang A.: 'An algebra for assessing trust in certification chains', *NDSS*, 1999, **99**, (6), pp. 80–89

[19] Manchala D.W.: 'Trust metrics, models and protocols for electronic commerce transactions'. Proc. 18th International Conf. on Distributed Computing Systems, 1998, pp. 312–321

[20] Jøsang A., Ismail R.: 'The beta reputation system'. Proc. 15th BLED Electronic Commerce Conf., 2002, pp. 41–55

[21] Jøsang A., Haller J.: 'Dirichlet reputation systems'. Proc. of Second International Conf. on Availability, Reliability and Security, 2007, pp. 112–119

[22] Nielsen M., Krukow K., Sassone V.: 'A Bayesian model for event-based trust', *Electron. Notes Theor. Comput. Sci.*, 2007, **172**, pp. 499–521

[23] Buchegger S., Le Boudec J.-Y.: 'Coping with false accusations in misbehavior reputation systems for mobile ad-hoc networks'. Technical Report, 2003

[24] Berger J.O.: 'Statistical decision theory and Bayesian analysis' (Springer Science & Business Media, 2013)

[25] Gordon N.J., Salmond D.J., Smith A.F.: 'Novel approach to nonlinear/non-Gaussian Bayesian state estimation', *IEE Proc. F (Radar Signal Process.)*, 1993, **140**, (2), pp. 107–113

[26] Arulampalam M.S., Maskell S., Gordon N., *ET AL.*: 'A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking', *IEEE Trans. Signal Process.*, 2002, **50**, (2), pp. 174–188

[27] Liu B., Ji C., Zhang Y., *ET AL.*: 'Multi-target tracking in clutter with sequential Monte Carlo methods', *IET Radar Sonar Navig.*, 2010, **4**, (5), pp. 662–672

[28] Smith A., Doucet A., de Freitas N., *ET AL.*: 'Sequential Monte Carlo methods in practice' (Springer Science & Business Media, 2013)

[29] Liu B., Ma X.-C., Hou C.-H.: 'A particle filter using SVD based sampling Kalman filter to obtain the proposal distribution'. Proc. of IEEE Conf. on Cybernetics and Intelligent Systems, 2008, pp. 581–584

[30] Snyder C., Bengtsson T., Bickel P., *ET AL.*: 'Obstacles to high-dimensional particle filtering', *Mon. Weather Rev.*, 2008, **136**, (12), pp. 4629–4640

[31] Bengtsson T., Bickel P., Li B., *ET AL.*: 'Curse-of-dimensionality revisited: collapse of the particle filter in very large scale systems'. Probability and Statistics: Essays in honor of David A. Freedman, Institute of Mathematical Statistics, 2008, pp. 316–334

[32] Rebeschini P., Van Handel R.: 'Can local particle filters beat the curse of dimensionality?', *Ann. Appl. Probab.*, 2015, **25**, (5), pp. 2809–2866

[33] Hu X.-L., Schon T.B., Ljung L.: 'A basic convergence result for particle filtering', *IEEE Trans. Signal Process.*, 2008, **56**, (4), pp. 1337–1348

[34] Crisan D., Doucet A.: 'A survey of convergence results on particle filtering methods for practitioners', *IEEE Trans. Signal Process.*, 2002, **50**, (3), pp. 736–746

[35] Samuel M.: 'Intel lab data'. Available at http://www.db.csail.mit.edu/labdata/labdata.html/, June 2004

[36] Rasmussen C.E., Nickisch H.: 'Gaussian processes for machine learning (GPML) toolbox', *J. Mach. Learn. Res.*, 2010, **11**, pp. 3011–3015