Machine Learning-Driven Trust Prediction for MEC-based IoT Services

Prabath Abeysekara
School of Science
RMIT University
Melbourne, Australia
Email: s3693452@student.rmit.edu.au

Hai Dong
School of Science
RMIT University
Melbourne, Australia
Email: hai.dong@rmit.edu.au

A.K. Qin
School of Software and Electrical Engineering
Swinburne University of Technology
Melbourne, Australia
Email: kqin@swin.edu.au

***Abstract*—We propose a distributed machine-learning architecture to predict trustworthiness of sensor services in Mobile Edge Computing (MEC) based Internet of Things (IoT) services, which aligns well with the goals of MEC and requirements of modern IoT systems. The proposed machine-learning architecture models training a distributed trust prediction model over a topology of MEC-environments as a Network Lasso problem, which allows simultaneous clustering and optimization on large-scale networked-graphs. We then attempt to solve it using Alternate Direction Method of Multipliers (ADMM) in a way that makes it suitable for MEC-based IoT systems. We present analytical and simulation results to show the validity and efficiency of the proposed solution.**

***Keywords*-Trust, Internet of Things, Mobile Edge Computing, Machine Learning.**

I. Introduction

Trust in IoT systems is an indispensable element. Not only does it influence user acceptance towards IoT, but also is regarded as a measure of privacy and security of such systems. IoT trust is often influenced by a multitude of distinct factors such as ubiquity, heterogeneity and dynamism. Hence, ensuring the trustworthiness of services operating in IoT systems innately presents numerous challenges [1].

Furthermore, the prevalence of sensors and smart-devices in such environments also pose a number of challenges in the context of trust information collection, processing, and decision inference. Amongst them are the *ever-growing stress on current networking infrastructure* [2] caused by the sheer volume of trust information generated and sent to existing centralized cloud-based infrastructure for persistence, as well as the need for *enormous amounts of centrally-located storage and computing power to derive trust decisions* through analytics.

Mobile Edge Computing (MEC) is an emerging paradigm aiming to provide compute and storage resources in close proximity to mobile devices and sensors [2]. Characterized by the pooled compute resources sitting in the base stations of cellular networks, the concept of MEC was proposed in light of the demanding need for higher network resource utilization and low latencies. Therefore, it can be viewed as a seemingly befitting concept capable of addressing the challenges in conventional cloud-centric IoT systems [3].

However, a mere shift towards MEC infrastructures alone cannot fully address the challenges of trust computing in IoT. Instead, novel trust information processing techniques that complement such infrastructures too are essential to overcome the challenges posed by typical IoT systems. For instance, in MEC-based IoT systems, trust information is generally accumulated in *large volumes* and *distributed fashion* within each MEC environment creating a distributed topology of data silos. To effectively process and derive meaningful insights from such distributed silos of data, machine learning algorithms based on distributed optimization techniques naturally stand out as a good fit. In fact, such algorithms inherently allow *parallel processing of distributed datasets* and provide an elegant formula to take on trust information processing challenges in such a setting.

Motivated by the setting described above, this paper aims to propose *a distributed machine learning architecture to effectively predict trustworthiness of sensor service providers in MEC-based IoT systems*. The proposed architecture models a distributed trust prediction model over a topology of MEC environments as a Network Lasso problem. It then attempts to train the aforesaid distributed trust prediction model *collaboratively* and in parallel using Alternating Method of Multipliers (ADMM) by sharing knowledge with each other for efficient model training.

II. Problem Formulation

Trust between a sensor service provider and consumer in an IoT system is composed of multiple distinct factors [4, 5], which we identify as *trust features*. These trust features are combined in multiple different ways to predict trust between a service provider and consumer in such systems, as well. We suppose a simple mathematical representation could be derived out of them, which can represent every existing trust prediction model generically, as below.

Given a set of arbitrary trust features $x_i \in R^D$, the impact of each trust feature towards the overall trust value denoted as elements of a weight vector $w \in R_D$, and a mapping function f_i defining how each trust feature and their respective weight coefficients can be consolidated to come up with an overall trust value y_i, any arbitrary trust model could be represented, as below.

$$\mathrm{y}_i = f_i(x_i; w) \text{ where } f : R^D \times R \Rightarrow R \tag{1}$$

Then, by applying the basic machine learning theory, and given a labelled trust dataset $P = (x_i, y_i), i = 1, \ldots, n$, the problem of deriving a trust prediction model can be introduced as inferring the *best* set of values for w from P that minimizes the cumulative deficit between the observed trust y_i and output of $f(x; w)$ against every training example in P. For mathematical convenience, let us denote the aforementioned deficit in the form of a loss function $l(x_i, y_i; w)$. Then, finding the best set of values for w can *generally* be defined in the form of an optimization problem, as below.

$$\underset{w \in R^D}{\text{minimize}} \quad \sum_{i=1}^{n} l(x_i, y_i; w) = \underset{w \in R^D}{\text{minimize}} \quad F(w) \tag{2}$$

Most existing machine learning based trust prediction models attempt to solve problem (2) on top of datasets accumulated *centrally* in cloud-based data centres. However, the aforementioned assumption is inherently obsolete and too restrictive in the context of MEC-based IoT environments. In such environments, trust information generated from transactions between service producers and consumers is generally persisted and processed in an entirely decentralized manner within MEC-local data-centres. This demands us to re-formulate problem (2) to fit into a distributed setting.

$$\underset{w \in R^D}{\text{minimize}} \quad \sum_{i=1}^{m} \sum_{j=1}^{n_i} l_i(x_j, y_j; w_i) = \underset{w_i \in R^D}{\text{minimize}} \quad \sum_{i=1}^{m} F_i(w_i) \tag{3}$$

where MEC environments within a given MEC topology are indexed with $i \in (1, .., m)$, l_i and F_i denote the loss and cost functions used to train a trust model in i^{th} MEC environment within a given MEC topology, w_i denotes the weight vector associated with F_i and l_i, (x_{ij}, y_{ij}) denotes the dataset used to train the trust model used by the i^{th} MEC environment defined as $P_i = (x_{ij}, y_{ij});\ j = 1, , n$.

The minimization problem (3) implies that the total prediction loss over the entire MEC topology is computed and minimized as a finite sum of the prediction losses incurred by each individual MEC environment. Solving a problem of this form can inherently be componentized into multiple sub problems, each of which can be solved in parallel and isolation within individual MEC environments. In other words, each sub problem of the form (2), represented by i^{th} index in problem (3) can conveniently be solved at the i^{th} MEC environment in a given MEC topology on top of dataset accumulated within itself. In addition to the inherent parallelism enforced by the aforesaid approach, it also minimizes the movement of data across multiple MECs as well as between MEC and Cloud layers resulting in significantly less network overhead on the underlying network infrastructures.

However, it is important to allow collaboration among multiple MEC environments so that *similarly-poised* MEC environments can derive more accurate trust prediction models by *borrowing strength from each other*. In such a setting, we assume that MEC environments accumulating trust information that exhibit similar trust features would want to train trust prediction models collaboratively while avoiding those that use different trust features. Consequently, problem (3) can be further modified as a *regularization problem* in which trust models carrying significantly different trust features are penalized while incentivizing those that carry similar trust features, as below.

$$\underset{w_i \in R^D}{\text{minimize}} \quad \sum_{i=1}^{m} F_i(w_i) \quad + \quad \lambda \sum_{j \in N(i)} R(w_i, w_j) \tag{4}$$

where R denotes a penalty function that incentivizes similarly-poised models and penalizes dissimilar models and (w_i, w_j) are weight vectors corresponding to trust prediction models trained by two adjacent MEC environments.

The penalty term introduced in problem (4), in its default form, pollutes the parallelism enforced by problem (3) as it involves computing a global sum of differences between weight vectors learnt by adjacent MEC environments across the MEC topology, even though the objective and regularization terms can *individually* be solved in parallel. These differences are computed in a pair-wise manner between two adjacent MEC environments over the entire MEC network. As a result, we can no longer isolate solving an independent sub-problem at i^{th} MEC environment. We, therefore, deem problem (4) is of the exact form of a typical Network Lasso problem, and therefore, attempt to solve it using Alternating Method of Multipliers (ADMM) to be able to transform it to a form that can be solved in parallel.

III. Solution

In a typical MEC topology, individual MEC environments tend to operate independently from others within their own network boundaries hindering knowledge sharing with each other. This is when we can fully utilize the centralized cloud, which all MEC environments possibly connect to, in order to establish a *logical* MEC network allowing them to communicate indirectly with each other (see Figure 1). In such a setting, each MEC environment can be *logically* connected to multiple neighbouring MEC environments via the centralized cloud for collaborative training of trust prediction models. There are multiple ways we can model a network of MEC environments exploiting various characteristics and properties of different application contexts [4]. However, for simplicity, we used a simple model where each MEC environment connects to a set of other MEC environments determined based on proximity. We map the Network Lasso problem to the graph resulting from this topology (see Figure 2), and ADMM is applied to derive a parallel solution to train a distributed trust prediction model.

Given a MEC topology modelled as an undirected networked graph, i.e. $G = (V, E)$ in which individual MEC environments are are denoted by i.e.

$V = \{1, \dots, N\}$, and their *logical* connectivity with neighbouring MEC environments is denoted by the edges i.e. $E = \{(v_1, v_2) : v_1, v_2 \in V, v_1 \neq v_2\}$, the Network Lasso problem over a MEC topology can be mathematically expressed, as below.

$$\text{minimize} \quad \sum_{i \in V} f_i(w_i) + \lambda \sum_{(j,k) \in E} a_{jk} ||w_j - w_k||_2. \tag{5}$$

In this optimization problem, $w_i \in \mathbb{R}^n$ represents model parameters of a loss function $f_i = \{(w_i, f(w_i)) : w_i \in \mathbb{R}^n\}$ corresponding to the trust model trained at i^{th} MEC environment denoted by $v_i \in V$. Each loss function f_i is defined over the input-output space $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is local to a node $v_i \in V$ in the graph G. Meanwhile, λ is a regularization parameter that scales the edge objectives relative to the node objectives, a_{jk} represents an impact factor of a particular edge i.e. (v_j, v_k) on the finite-sum problem computed over the loss functions of all nodes participating in the optimization problem. It is also noteworthy that w_j and w_k correspond to the parameters of the models associated with two adjacent nodes v_j, v_k in the graph, respectively.

ADMM algorithm allows a suitable convex optimization problem to be decomposed into multiple sub-problems and solve them iteratively in a coordinated manner in parallel [5]. It has been priorly used to decompose the Network Lasso problem into multiple sub-problems that can be solved in parallel [6]. Therefore, to convert problem (5) into a form, which allows us to apply ADMM, copies of both w_i and w_j (i.e. z_{ij} and z_{ji}, respectively) are introduced at every edge v_j and v_k in the graph as per [6]. The transformed problem can now be viewed as,

$$\begin{aligned}&\text{minimize} \quad \sum_{i \in V} f_i(w_i) + \lambda \sum_{(j,k) \in E} a_{jk} ||z_{jk} - z_{kj}||_2. \\ &\text{subject to} \quad w_j = z_{jk}\end{aligned} \tag{6}$$

By applying ADMM against (6), we then decompose the aforementioned optimization problem into 3 sub-problems, as below.

$$\begin{aligned}
w_i^{k+1} &= \underset{w_i}{\text{argmin}} \Big(f_i(w_i) + \sum_{j \in N(i)} \frac{\rho}{2} ||w_i - z_{ij}^k + u_{ij}^k||_2^2 \Big) \\
z_{ij}^{k+1}, z_{ji}^{k+1} &= \underset{z_{ij}, z_{ji}}{\text{argmin}} \Big(A_{ij} ||z_{ij} - z_{ji}||_2 + \\
&\quad \frac{\rho}{2} (||w_i^{k+1} - z_{ij} + u_{ij}^k||_2^2 + ||w_i^{k+1} - z_{ji} + u_{ji}^k||_2^2) \Big) \\
u_{ij}^{k+1} &= u_{ij}^k + (w_i^{k+1} - z_{ij}^{k+1})
\end{aligned} \tag{7}$$

where, u is a scaled dual variable used for mathematical convenience and $\rho\ (> 0)$ is the penalty parameter, while k denotes the iteration number. We refer the reader to [5], [6] for further reading on the mathematical proof of the above.

In such a topology, the sub-problems denoted by (7) can be solved at different layers between local MEC and centralized cloud infrastructures (see Figure 2) with different semantics, as elaborated below.

Algorithm 1 Network-Lasso for MEC-based IoT systems

```
1: parameters: $M$-MEC environments, $E_p, E_d$-Expected primal & dual residuals, $\rho$-Penalty parameter, $K$-Maximum allowed iterations
2: for all $m \in M$ do    ▷ Loop over MECs in Cloud layer
3:     Send initial $z_{ij}$, $z_{ji}$ and $u_{ij}$ to $m$
4: Initialize $k$, $\left\|res_p^k\right\|_2$ and $\left\|res_d^k\right\|_2$
5: while $\left\|res_p^k\right\|_2 \geq E_p$; $\left\|res_d^k\right\|_2 \geq E_d$; $k < K$ do
6:     for all $m \in M$ do    ▷ Distributed loop over MECs
7:         $w_i^{k+1} \leftarrow \underset{w_i}{\text{argmin}} \Big( f_i(w_i) + \sum_{j \in N(i)} \frac{\rho}{2} \left\|w_i - z_{ij}^k + u_{ij}^k\right\|_2^2 \Big)$
8:         Send $w_i^{k+1}$ to cloud layer
9:     for all $m \in M$ do    ▷ Loop over MECs in Cloud
10:        $z_{ij}^{k+1}, z_{ji}^{k+1} \leftarrow \underset{z_{ij}, z_{ji}}{\text{argmin}} \Big( A_{ij} \left\|z_{ij} - z_{ji}\right\|_2 + \frac{\rho}{2}(\left\|w_i^{k+1} - z_{ij} + u_{ij}^k\right\|_2^2 + \left\|w_i^{k+1} - z_{ji} + u_{ji}^k\right\|_2^2) \Big)$
11:        $u_{ij}^{k+1} \leftarrow u_{ij}^k + (w_i^{k+1} - z_{ij}^{k+1})$
12:    Compute $\left\|res_p^k\right\|_2$ and $\left\|res_d^k\right\|_2$
13:    $k \leftarrow k + 1$
```

w_i-update: As the first of the aforementioned three sub-problems, w_i-update is separable across each local MEC environment in the MEC topology and solved in parallel. A global iteration coordinator system will provide each and every local MEC-based models the z_{ij}- and u_{ij}-updates cached from the previous iterations so that each local MEC layer will then be able to independently train their own local model with sufficient accuracy at each iteration (see Figure 2(a)). At the end of each local update, the resulting vector of model parameters w_i is synchronized with the global model coordinator (see Figure 2(b)). This will assist the global model coordinator to maintain a pseudo graph of the MEC topology (together with the model parameters of each local MEC layer), and carry out all subsequent steps against the aforementioned graph.

z_{ij}-, z_{ji}- and u_{ij}-updates: In contrast to w_i-update, z_{ij}-, z_{ji}- and u_{ij}-updates are separable across edges that exist over all nodes in the pseudo MEC topological graph maintained by the global model coordinator. These updates will be computed solely at the centralized cloud infrastructure to minimize extensive back-and-forth communication between the local MEC networks and centralized cloud infrastructures (see Figure 2(c)).

The complete algorithm that consolidates all the aforementioned steps is depicted in Algorithm 1. We used a soft-margin Support Vector Machine (SVM) [7] to train trust

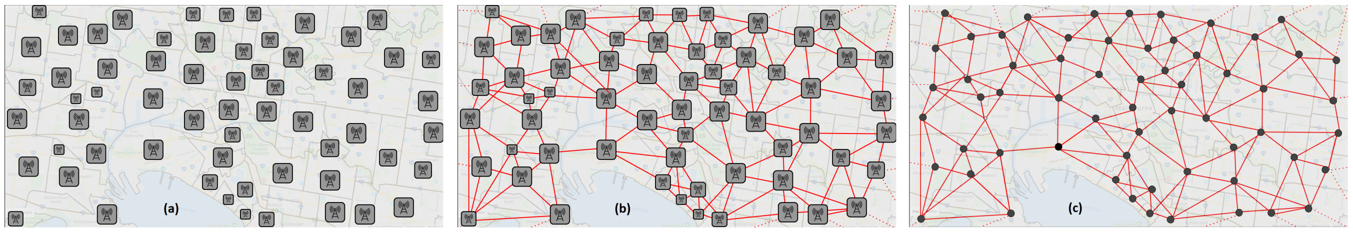

Figure 1: A hypothetical deployment of MEC environment, which shows how the neighbouring MEC environments are linked based on proximity forming a partial mesh network.

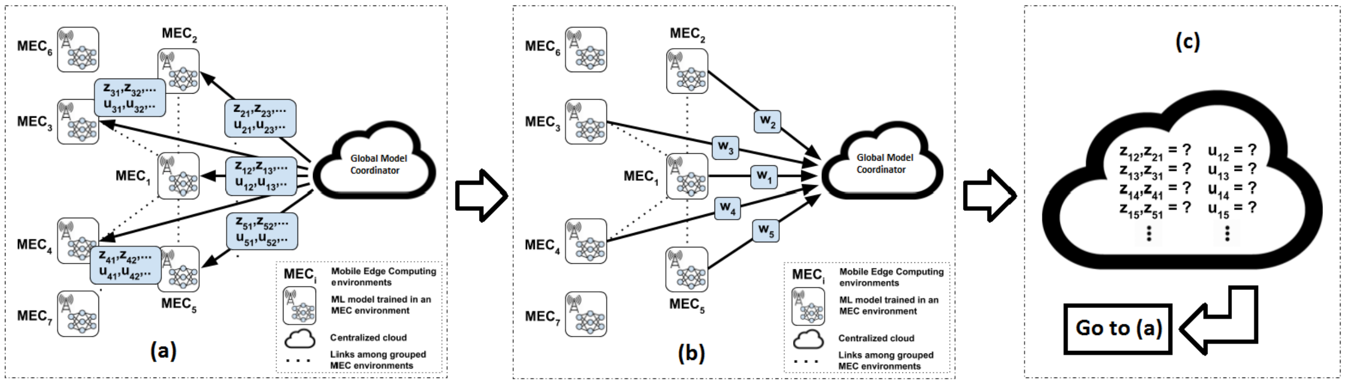
MEC$_6$
MEC$_2$
MEC$_3$
MEC$_1$
MEC$_4$
MEC$_5$
MEC$_7$
$z_{31}, z_{32}, \dots$ $u_{31}, u_{32}, \dots$
$z_{21}, z_{23}, \dots$ $u_{21}, u_{23}, \dots$
$z_{12}, z_{13}, \dots$ $u_{12}, u_{13}, \dots$
$z_{51}, z_{52}, \dots$ $u_{51}, u_{52}, \dots$
$z_{41}, z_{42}, \dots$ $u_{41}, u_{42}, \dots$
Global Model Coordinator
MEC$_i$ Mobile Edge Computing environments
ML model trained in an MEC environment
Centralized cloud
Links among grouped MEC environments
(a)
w_2
w_3
w_1
w_4
w_5
(b)
(c)
z_{12}, z_{21} = ? u_{12} = ?
z_{13}, z_{31} = ? u_{13} = ?
z_{14}, z_{41} = ? u_{14} = ?
z_{15}, z_{51} = ? u_{15} = ?
Go to (a)

Figure 2: A visualization of Network Lasso for MEC-based IoT systems.

prediction models in each individual MEC environment. SVM had already been widely used and shown to work well in prior trust research for developing classification- and regression-based trust prediction models across multiple context including IoT systems [8], [9]. This background provided us with a rational basis to adapt SVM as the MEC-local trust prediction problem to be solved as part of each sub-task running in MEC environments of the reference implementation. In that, each local MEC environment trains its own SVM-based binary classifier to predict untrustworthy IoT services.

IV. Evaluation

Experiments: We conducted a series of experiments to evaluate the suitability of the proposed approach for MEC-based IoT environments. The primary objective of these experiments was to observe the prediction accuracy of the proposed approach by comparing the results it produces against two baseline models. These baseline models included

- 100 local binary SVMs trained atop the same splits of dataset upon which the Network Lasso model is trained, representing a distributed yet a non-communicative and isolated set of prediction models,
- a global binary SVM classifier trained atop the same dataset to represent a centralized prediction model.

Network Lasso-based proposed model was trained over multiple progressively incremented values of $\lambda \in \{0, 0.001, 0.01, 0.1, 1, 10, 100\}$ and ρ=1.0 to determine the point where it achieves maximum accuracy. A confusion matrix representing Accuracy, Precision, Recall, F1-Score was used to compare the performance of each binary SVM classifier trained during the experiments.

Experiment Set-up: The simulation took into account a hypothetical scenario where MEC can be used in a smart-city set-up surrounding 100 suburbs in the Melbourne City Council area. We marked every MEC environment pertaining to a particular suburb as a node in a graph laid on top of a map of Melbourne City Council (see Figure 2). The MEC topology depicted in the form a complex graph resembles a partial mesh network in which every node is connected to five other nodes based on proximity. In addition, all simulations and algorithms used were implemented in Python.

Dataset: All experiments were conducted on a population of training examples extracted from UNSW-NB15 [10], a public dataset available for simulations of intrusion detection systems. The dataset consisted of examples (each containing 49 numerical and categorical features) collected from benign transactions and nine types of attacks scenarios, labelled under two classes accordingly. This dataset was first normalized, and then divided into 100 splits carrying independent and randomly-sized ($n \in [200, 2000]$) datasets forming an aggregate of 110892 examples. Random noise was also added to each data split via flipping the labels of randomly-picked samples to mimic a real-world dataset that carries noise. The resulting splits and an aggregate of them (with a training-to-test split ratio of 70:30) were used to train local SVMs for each simulated MEC-environment and the global SVM, respectively.

A. *Results and Discussion*

Results of our experiments (see Figure 3) showed that the binary SVM classifiers trained by the Network Lasso based model recorded a promising maximum average accuracy of 97.79% (when λ=10) whereas non-collaborative local SVMs and the global SVM we used as baseline models recorded an maximum average accuracy of 81.36% and 98.81%, respectively. In addition, the average precision, recall and F1-score recorded by all three approaches against the 100 simulated MEC environments via multiple trials too showed similar results as depicted in Figure 3.

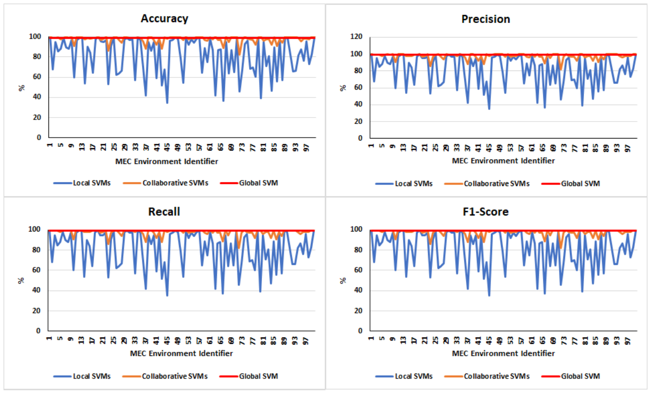

Figure 3: Comparison of average accuracy, precision, recall and F1-score of local, global and collaborative SVMs.

Based on the results, it is apparent that the proposed Network Lasso based machine learning architecture significantly outperforms the non-collaborative local SVM models trained within each simulated MEC environment. The significant increase in accuracy recorded by Network Lasso based SVMs against non-collaborative local SVMs can be attributed to the ability of the former to incentivize knowledge sharing among prediction models trained by the nodes simulating MEC environments. This affirms the *effectiveness* of the proposed architecture as it encourages training prediction models with higher accuracies even amidst distributed, noisy and unbalanced datasets via collaboration. Therefore, the proposed collaborative approach can be deduced significantly better compared to each different MEC environments attempting to train trust prediction models on top of their locally accumulated datasets in isolation.

Meanwhile, the difference between the maximum average accuracies shown by collaborative and global SVM models was observed to be very insignificant. The superior accuracy shown by the global SVM model can be attributed to it being trained on a much larger dataset compared to the smaller splits used by the MEC-local SVMs, as the former sees a better view of the dataset during model training. However, the above accuracy was achieved at the expense of a total of 110892 communication iterations (i.e. the cost of transmitting all training data to the centralized cloud) via the simulated core network layer of the mobile network providers. In comparison, the proposed approach showed comparably superior accuracy with significantly reduced number of communication iterations (n=29000), also offering controllability to achieve a trade-off between accuracy and communication cost via tuning the hyperparameters. These communication iterations were resulted in from carrying out the w_i updates (see Section III) against the 100 nodes used in the simulated MEC topology. Consequently, we can deduce that the proposed approach promises a magnitude smaller network-stress on the core networks of mobile network providers. This affirms the *adaptability* of the proposed model to MEC-based IoT systems, and its close alignment with the goals of the MEC paradigm [11].

V. Conclusion and Future Work

This paper proposes a distributed machine learning architecture based on the *Network Lasso* problem and *Alternating Direction Method of Multipliers* algorithm for trust prediction in MEC-based IoT services. The proposed architecture allows training a distributed trust prediction model collaboratively across a given topology of MEC environments by sharing knowledge with each other. The proclaimed benefits of our approach were validated via a suitable simulation environment on top of a public dataset. Our future work includes evaluating how the proposed model fares against the scalability requirements and heterogeneity of trust information available in typical MEC-based IoT systems.

References

[1] Z. Yan, P. Zhang, and A. V. Vasilakos, "A survey on trust management for internet of things," *Journal of network and computer applications*, vol. 42, pp. 120–134, 2014.

[2] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *AFIN*, 2014, Conference Proceedings, pp. 48–55.

[3] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[6] D. Hallac, J. Leskovec, and S. Boyd, "Network lasso: Clustering and optimization in large graphs," in *ACM SIGKDD*, 2015, Conference Proceedings, pp. 387–396.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] J. Lopez and S. Maag, "Towards a generic trust management framework using a machine-learning-based trust model," in *Trustcom/BigDataSE/ISPA*, vol. 1, 2015, pp. 1343–1348.

[9] U. Jayasinghe, G. M. Lee, T.-W. Um, and Q. Shi, "Machine learning based trust computational model for iot services," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 39–52, 2018.

[10] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *MilCIS*, 2015, pp. 1–6.

[11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.