The ISC Int'l Journal of Information Security
ISeCure

Manuscript template of ISeCure Journal (pp. 1–8)
http://www.isecure-journal.org

A Budget-Constrained Placement of Controller Nodes for Maximizing the Network Performance in SDN-Enabled WSNs

Seyedeh Kosar Mousavi [1,*], Saber Fazliahmadi [2], Nayereh Rasouli [3], Hamid Reza Faragardi [4,*], Hossein Fotouhi [5], Thomas Fahringer [6]

[1] *Department of Computer Science, Islamic Azad University, Ramsar, Iran*
[2] *Department of Information Technology Management, Science and Research Branch, Islamic Azad University, Tehran, Iran*
[3] *Department of Computer Engineering, Karaj Branch, Technical and Vocational University, Alborz, Iran*
[4] *Department of Computer Science and Communication, KTH Royal Institute of Technology, Stockholm, Sweden*
[5] *Mälardalen University, Västerås, Sweden*
[6] *School of Computer Science, The University of Innsbruck, Innsbruck, Austria*

ARTICLE INFO.

Keywords:
Wireless sensor networks, Software defined networking, Node placement, Integer Linear Programming, Optimization.

Abstract

Software Defined Networking (SDN) is a novel technique to provide network reconfigurability in Wireless Sensor Networks (WSNs). SDN is highly suitable to be applied in WSNs where high scalability and high reliability are required. To realize the SDN concept, a set of additional nodes, referred to as SDN-controller nodes (or controllers for short), are integrated into the network. Controllers are responsible to advertise routing rules dynamically based on network and link changes. Emerging controllers rises a new research challenge to determine the number and location of controller nodes in a WSN to maximize the network performance subject to both reliability and budget constraints. The budget constraint restricts the maximum number of controller nodes deployed in a WSN. In this paper, we first deal with the challenge to place SDN-controller nodes by introducing an ILP model for the problem which then is solved using the CPLEX ILP solver. We evaluate the results of the proposed method through comparison with the state-of-the-art method. Extensive experiments demonstrate that the proposed method reduces the maximum distance between sensors and controllers by 13% in average in comparison with the state-of-the-art method.

1 Introduction

Software Define Network (SDN) [1] is a network technology initially proposed for wired networks [2] providing the possibility of network reconfiguration through on-the-fly programming. SDN provides fine grained information to select the best forwarder to form global resource optimization rather than ad-hoc networks where the next-hop to forward data towards destination is selected by each node independently. Recently, using SDN in Wireless Sensor Networks (WSNs) becomes an increasingly important trend to improve flexibility and scalability of the network. Dynamic reconfiguration of the network is an useful feature specially in harsh environments where wireless links could be highly unreliable, and thus network routing should be updated frequently.

Since the SDN controller demands high storage

* Corresponding author.
Email addresses: sara.heidarian@gmail.com, saberfazliahmadi@gmail.com, nrasuly@yahoo.com, hrfa@kth.se, hossein.fotouhi@mdh.se, tf@dps.uibk.ac.at
ISSN: 2008-2045

and large computational power, the limited computation and storage capacity of available WSN devices hinders the implementation of the SDN controller. A promising solution to implement the SDN controller is to integrate additional node(s) into WSNs which are responsible to execute the SDN controller software. Although the notion behind the control plane in software-defined networks is based on a centralized fashion, the SDN controller must be physically distributed among multiple nodes to achieve higher performance, scalability and fault tolerance [3, 4]. This means that the network state must be synchronized among the different controller nodes. In [3], the synchronization cost between multiple controllers was discussed and the results indicate the feasibility of multi-controller deployments in terms of network latency.

If we adopt a distributed version of the SDN controller in WSNs, then the question is how to deploy the SDN controller nodes in the network. How many nodes are required and where should they be placed? Two important Quality of Service (QoS) elements should be taken into account when it comes to deployment of multiple controller nodes in a WSN, namely, reliability and performance.

Reliability is a challenging issue in WSNs since the failure of the key entities can degrade the quality of service resulting in missing the application deadlines. Node failure in WSNs may be caused by communication errors, unstable connectivity and faulty sensing/actuating/controlling modules [5]. The controller failure will abandon the network with no proper control commands which results in performance degradation. A practical solution to design a reliable network is to avoid a single point of failure, which can be fulfilled using node replication. Hence, in the network architecture, multiple controllers should be accommodated to achieve a higher reliability.

Performance is the other parameter that can be considerably affected by the placement of controller nodes. A tactful placement of controller nodes can reduce the distance between a sensor and the controller(s) covering the sensor. A shorter distance not only reduces the network latency to exchange control messages, but also decreases the network traffic and saves energy consumption.

Although using multiple controllers can improve both reliability and the performance of the network, it increases the deployment cost of the network. In this paper, we intend to find an optimal controller placement to maximize the network performance subject to (i) a certain budget limiting the maximum number of controllers, and (ii) the reliability constraint. We formulate the problem as an optimization problem which is then modeled as an Integer Linear Programming (ILP). The proposed ILP model is solved by CPLEX ILP Solver and the results are compared with the recently proposed method in the literature, dubbed as MSCP [4].

A. Contributions. Our major contributions are listed as follows:

(1) Specifying the optimal placement of controller nodes in WSNs.
(2) Taking into account the location of both sensors and sink nodes to find an optimal deployment of SDN controllers.
(3) Targeting both reliability and performance (in terms of the number of hops between controllers and other nodes of the network) in the deployment phase of SDN-enabled WSNs.

B. Organization of the paper. The paper is organized as follows: In Section 2, a comprehensive review of related work is presented. In Section 3, we describe the problem and assumptions, which is then followed by our proposed ILP model. In Section 4, the performance of the proposed method is investigated. Finally, in Section 5, we conclude our paper by providing a summary along with future directions.

2 Related Work

In WSNs, a large number of nodes including sensors, sink nodes, relays, and controller nodes should be distributed in the environment while their locations may not be known in advance. The location of the nodes can change the structure of the network and noticeably impacts on the network performance. Therefore, it is an important research challenge that 'how the nodes in a WSN should be placed to achieve the maximum efficiency and performance?'

There are a huge number of studies focusing on the node placement in WSNs and in IoT systems. They concentrate on different types of nodes and different performance metrics in their node placement strategies. For example, in [6], the authors targeted coverage and connectivity metrics by solving the sensor placement problem. This work formulated the sensor placement problem as a constrained optimization problem. In [7], mobile sinks are employed to reduce the problem of nodes' battery depletion closer to sinks. It presents a Mixed Integer Linear Programming (MILP) formulation for finding optimal positioning of mobile sinks while CPLEX is used to address the MILP problem. Indeed, it targets mobile sink placement in order to increase network lifetime. In [8], a Joint Sink Deployment and Association (JSDA) in a multiple sink wireless camera network is formulated as a mixed integer linear programming problem.

In [9], four algorithms are proposed, namely, GreedyMSP and GRASP-MSP to solve the problem of multiple sink placement, and Greedy-MSRP and GRASP-MSRP for the problem of multiple sink and relay placement. Greedy-MSP and GRASP-MSP were designed to minimize the deployment cost, while ensuring double-coverage of each sensor node in the network. In [10], a Genetic Algorithm (GA) is proposed to deploy multiple sinks in order to minimize the worst-case delay in WSNs, while in [11], a Particle Swarm Optimization (PSO) is employed to calculate the number of sinks and their locations in order to reduce the path length between sensors and sinks. In the same context, [12] formulates the k-sink placement problem as an optimization problem, so that the data latency from a sensor with its nearest sink is minimized, while in [13], a PSO-based algorithm is proposed to prolong the network lifetime by considering the Euclidean distance and hop count. In [12], the placement of multiple sinks in order to minimize worst-case delay is taken into account and an approximation algorithm called GREEDY-k-SPP is proposed while [13] focuses on reducing energy consumption in WSNs in a random deployment of a grid topology network.

In [14], the authors propose a Deployment Strategy for Multiple Types of Requirements (DSMTRs) that targets full coverage, guaranteeing connectivity, and satisfying different critical requirements, including deployment cost, transmission delay, and network lifetime. The main contribution of DSMTRs in terms of nodes' placement is to devised a Cost-Based Deployment Algorithm (CBDA) to reduce the deployment cost.

The controller placement problem for wired networks has been addressed in [15]. The authors explore the trade-offs when optimizing for minimum latency between nodes and controllers. [16] is an extension of [15] which is referred to as Pareto-Optimal Controller placement (POCO) considering additional aspects other than network latency.

Although placing controller nodes in WSNs is already addressed by multiple research works such as [4, 17], in designing the network, they mainly focus on the deployment cost of the network as the objective function rather than network performance. Indeed, browsing the related works manifests that none of the research works target the problem of multiple controller placement while considering both reliability and the performance metrics.

3 Problem Modeling

In this paper, we assume an SDN-enabled sensor network, where sensor nodes collect information and forward it towards sink nodes. We assume that the placement of the sink nodes has been already accomplished using the state-of-the-art methods such as [4, 17], and here, we focus only on the placement of controllers nodes with respect to the given location of sinks and sensors. Reliability and timeliness are two major requirements in the design of the system which are considered in our model through the following items:

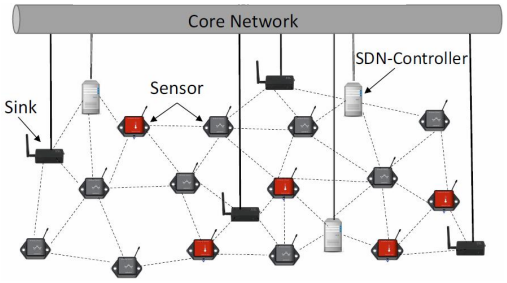

Figure 1. Network topology [4].

(1) Covering each sensor by k (k>1) controllers to avoid a single point of failure.
(2) Minimizing the maximum number of hops between sensors and controllers.
(3) Limiting the maximum number of hops between controllers and sink nodes.

where the first item implies the reliability constraint and the two latter items come from performance aspects. These items will be formulated in the following of this section. Fig. 1 illustrates a network, where sensor nodes construct a mesh network connecting to multiple sinks and controllers.

A WSN is represented by an undirected graph, where vertices are partitioned into a set of sensors T, sinks S and controllers C. Hence, in the graph representation of a WSN, $V = T \cup S \cup C$. An edge of the graph indicates a wireless connection between a pair of nodes. The total number of sensors is N; i.e., $N = |T|$. A pair of nodes connected with an edge are called, *neighbor*. A path with the length l from node v to v' is a sequence of nodes $v, v_1, v_2, ..., v_l$ – where $v_l = v'$– connecting node v to v', such that v_i, v_{i+1} are neighbors. We define A_C, denoting a set of candidate controllers where each candidate controller is associated to a single candidate location to place a controller, thereby, the term 'candidate controller' corresponds to 'a candidate location to place a controller node'. Hence, $C \subseteq A_C$. For example, as shown in Fig. 2(a), the set of candidate controllers A_C is equal to $\{1, 2, 3\}$, and as shown in Fig. 2(b), the set of selected controllers C is equal to $\{1, 3\}$. In this paper, we assume that the sets A_C is known in advance, similar to [4, 10, 17], however, there are multiple methods in the literature discussing the number of candidate locations to place nodes such as [11, 18].

Fig. 2(a) illustrates an example of our network assumption with three candidate controllers. It means that initially we have the possibility to place controllers in all these three locations. Fig. 2(b) shows the case where we applied our proposed controller placement strategy, and the network has two controller place with ID 1 and 3.

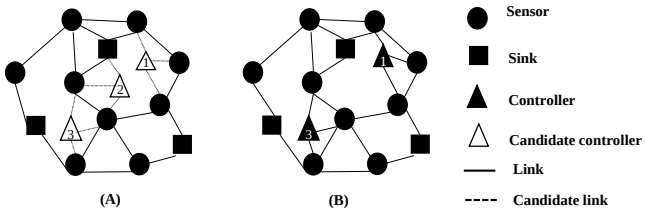

Figure 2. An illustrating example for the controller selection [4].

The amount of data exchange between sensors and sinks is often significantly greater than that for sensors and controllers [4]. The reason is that, all the gathered information by the sensors is sent to sinks, while only routing and other network managing messages are exchanged between sensors and controllers. Section 4 indicates that the number of required controllers to cover a WSN is significantly smaller than the number of required sinks.

In order to represent the problem, we use a binary vector X which determines whether a candidate controller has been selected or not.

$$X_i = \begin{cases} 1 & \text{if the candidate controller } i \text{ is chosen} \\ 0 & \text{else} \end{cases} \tag{1}$$

3.1 Network Reliability

A sensor is *controller-covered* if and only if it has at least one path with length $\leq l_{max}$ to one of the controllers in C [4]. If a sensor is not controller-covered, it is uncovered. Generally, a sensor is *k-controller-covered* if and only if it has at least k paths of length $\leq l_{max}$ to k controllers in C (k is an integer number ≥ 1). We define a WSN as k-controller-covered if each sensor in T is k-controller-covered. To model the k-controller-covered constraint, we first define a binary matrix Y as follows.

$$Y_{i,j} = \begin{cases} 1 & l^*(v_i, c_j) \leq l_{max} \\ 0 & \text{else} \end{cases} \tag{2}$$

where $l^*(v_i, c_j)$ is the shortest path (in terms of the number of hops) between node v_i and controller c_j, which can be calculated by the Dijkstra algorithm. We expect that each sensor is covered by $K > 1$ controller to avoid a single point of failure and to have a reliable network. Now we can formulate the k-controller-covered constraint as follows:

$$\sum_{\forall c_j \in A_C} Y_{i,j} X_j \geq K \qquad \forall v_i \in T \tag{3}$$

where v_i denotes the ith sensor.

3.2 Network Performance

In this paper we focus on two performance metrics. The first performance metric is one of the constraints of the problem, dubbed as *locality constraint*, restricting the maximum distance between controllers and sinks, whereas, the second metric is the objective of the optimization problem aiming at keeping the number of hops between sensors and controllers as short as possible.

3.2.1 Locality constraint

Placing a controller in the vicinity of a sink node reduces network overhead in terms of control message exchanges during run-time. Generally, a controller node is supposed to collect network updates by broadcasting control messages to all nodes. Establishing a connection between sink and controller would provide the opportunity of reducing the control message exchanges drastically as the sink node has already received the network update information through a normal data message. The controller may need to probe only low frequency and less active nodes. Accordingly, it is desirable that each controller is covered by at least one sink such that the length of the shortest path between the controller and the corresponding sink is not longer than l'_{max}. This assumption is called *locality assumption*. Fig. 3 illustrates an example of applying the locality assumption on the selection of the controller. The candidate controller 2 will not be selected as it has been located too far away from the given acceptable distance defined in the locality assumption.

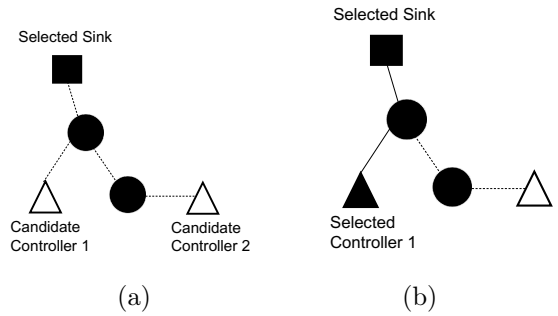

Figure 3. An example of controller selection with respect to a selected sink; (a) a scenario with two candidate controllers, and (b) the selected controller with respect to the $l'_{max} \leq 2$ constraint [4].

To formulate the locality constraint, the following inequality should hold for all selected controllers assuring that the number of hops between the closest sink and a controller is not more than l'_{max}.

$$\min_{\forall sink\ s_k} \{X_j l^*(s_k, c_j)\} \leq l'_{max} \qquad \forall c_j \in A_C \qquad (4)$$

where $l^*(s_k, c_j)$ shows the shortest path (i.e., the minimum number of hops) between the sink s_k and the controller c_j. The value of l'_{max} is determined according to the expected performance by the network designer. A lower value for l'_{max} cloud result in higher performance, but on the other hand, may increase the number of required controllers, i.e., a higher deployment cost.

3.2.2 Objective function

Since the controller needs to keep in touch with sensors to dynamically manage the routing decisions, the number of hops between sensors and the controller(s) considerably affects the network performance. A long path between sensors and the controller(s) can increase not only the network traffic but also the delay of exchanging control messages. Accordingly, we would like to place the controller nodes such that the farthest controller that covers a sensor becomes as close as possible to the sensor. To reflect this demand, Eq. 5 and 6 are applied.

$$L^*_{v_i} = \max_{\forall c_j \in A_C} \{Y_{i,j} X_j l^*(v_i, c_j)\} \qquad (5)$$

where $L^*_{v_i}$ denotes the furthest selected controller to the sensor v_i that covers the sensor. Accordingly, to minimize the maximum distance between sensors and controllers we should minimize $L^*_{v_i}$.

$$\textbf{Minimize:} \max_{\forall v_i \in T} \{L^*_{v_i}\} \qquad (6)$$

3.3 Budget Constraint

We assume all the controllers have the same type and the same price denoted by $Price^{controller}$. We have a certain budget to purchase controller nodes which is denoted by $Budget$. Obviously, the purchasing cost of controllers should not exceed the given budget.

$$\sum_{\forall c_j \in A_C} X_j \leq \left\lfloor \frac{Budget}{Price^{controller}} \right\rfloor \qquad (7)$$

Apparently, when the given budget is high enough to choose all the candidate controllers, we simply take all the candidate controllers and the problem is solved without further actions required. However, most often, the budget is limited which prevents taking all the candidate controllers.

3.3.1 Optimization problem formulation

The optimization problem is formulated as follows:

$$\textbf{Minimize}: \quad \max_{\forall v_i \in T} \{L^*_{v_i}\}, \qquad (8a)$$

$$\textbf{Subject to}: (3), (4), (7). \qquad (8b)$$

The problem has been modeled as an Integer Linear Programming (ILP) problem, particularly as an *{0,1} ILP* which is a subset of the ILP category where the domain of optimization parameters is only $\{0, 1\}$.

4 Performance Evaluation

In this section, we evaluate the performance of the proposed method with extensive experiments. We consider four WSNs with different sizes, from a small scale to a very large scale. The first WSN is derived from [17, 19] with 100 sensors and 20 sinks. We refer to this system as the small scale test-case. The number of candidate locations to place controllers is assumed to be 14. In the second test-case, referred to as a medium scale test-case, all system parameters of the small test-case are multiplied by two to provide a bigger WSN. Therefore, the number of sensors is 200, and the number of candidate locations to place controllers is 28. In the third test-case, referred to as the large scale test-case, all system parameters of the medium benchmark are multiplied by two to provide a bigger WSN. In the last test-case, referred to as the very large test-case, all system parameters of the large test-case are multiplied by two.

Other system parameters are listed in Table 1. It should be mentioned that the shortest paths between each pair of nodes are generated randomly using a uniform distribution in the range mentioned in the *Shortest Distance* column of the table.

4.1 Comparison Benchmark

To investigate and evaluate the performance of the proposed method against other methods introduced in the literature, we use MSCP proposed by Faragardi et al. in 2018 [4] as the benchmark. The reason to opt MSCP is that it is the state-of-the-art-method for placing controllers in WSNs. However, there are multiple differences in both the objective function and the constraints of that paper compared the the work described in this paper. Therefore, we should first modify MSCP to make it capable of being used as a comparison benchmark for our paper. First of all, as MSCP focuses on both sink and controller placement, we modify it such that it ignores the sink placement and only focuses on controllers. Moreover, the objective function of MSCP is to minimize the de-

Table 1. The test-cases used in the experiments

	No. Sensors	No. Sinks	No. Candidate Controller	Shortest Distance	k-covered	lmax	l'max
Small	100	20	14	u(1,15)	2	10	6
Medium	200	40	28	u(1,30)	2	10	6
Large	300	60	42	u(1,45)	2	10	6
Very large	400	80	56	u(1,60)	2	10	6

ployment cost of the network rather than minimizing the farthest controllers to sensors, thus, we first run MSCP for various values of l_{max} and the generated cost by MSCP is given to our proposed method as the budget in Eq. 7. We then run the ILP solver. At the end, the value of the key parameters (i.e., the furthest controllers to the sensors) generated by the proposed method and MSCP are compared with each other.

Table 2. The execution time of the ILP solver vs MSCP.

	Small	Medium	Large	Very large
Proposed Method	33 Sec	163 Sec	348 Sec	427 Sec
MSCP	92 Sec	106 Sec	120 Sec	143 Sec

4.2 Results

For each size of the system (small, medium, large and very large), both MSCP and the proposed method were implemented in C and executed on a PC with 8 cores and 10GB memory, running Ubuntu. Moreover, IBM CPLEX APIs were used to solve the model.

The results achieved by the two methods in terms of the objective value (i.e., the maximum number of hops between the farthest controller covering a sensor) for all test-cases are illustrated in Fig. 4. It should be noted that in all experiments, we assume that the price of controllers is equal to one, meaning that the budget value indicates the maximum number of selected controllers. As shown in the figure, in the small-scale test-case when the budget is set to three, the farthest controller from a sensor is located 10 hops far away from the location of the sensor in both MSCP and the proposed method. Rising the budget increases the flexibility in choosing controllers which results in reduction of the objective value, e.g., when the budget rises to 11, the objective value in the proposed method significantly improves and reaches the value 6. By increasing the size of the network, the preference of the proposed method against MSCP becomes more and more considerable. In the small-scale test-case, our method improves the objective value against MSCP by 6.7%. This preference for other test-cases including the medium-scale, large-scale, and very large-scale networks reaches 15.9%, 13.4%, and 16%, respectively.

Now let us have a look at the average execution time of CPLEX to solve the model versus that of MSCP which are listed in Table 2. The results of Table 2 demonstrate that by doubling the size of the network, the execution time of the proposed method does not exponentially scale up. On the other hand, the execution time of the proposed method for the largest system is less than 8 minutes, which is acceptable when it comes to deciding about the number and location of the required controllers in the deployment phase. This indicates the feasibility of the proposed method for real-world SND-enabled WSNs, however, for extremely large scale WSNs (e.g., a WSN with a few thousands of nodes), the ILP method is not feasible anymore.

5 Conclusion

In this paper, we have investigated the placement problem for multiple controllers in WSNs in order to improve the quality of the service, while considering both reliability and network performance. We formulated the problem as an ILP. CPLEX ILP solver was used to find an optimal solution for the ILP problem. To assess the performance and efficiency of the proposed method, various experiments were conducted on four WSNs with different sizes. The experimental results revealed that the proposed method substantially surpasses the state-of-the-art-method in terms of the maximum distance between sensors and controllers which leads to better network performance. For a WSN with 400 senors, the preference of our method reached up to 16% in comparison with the recently proposed method. As a continuation of this research work, we plan to introduce a new heuristic algorithm for the considered problem to resolve the scalability problem of the ILP method for huge WSNs with a few thousands of nodes.

ACKNOWLEDGMENT

The work presented in this paper is supported by the Swedish Foundation for Strategic Research via

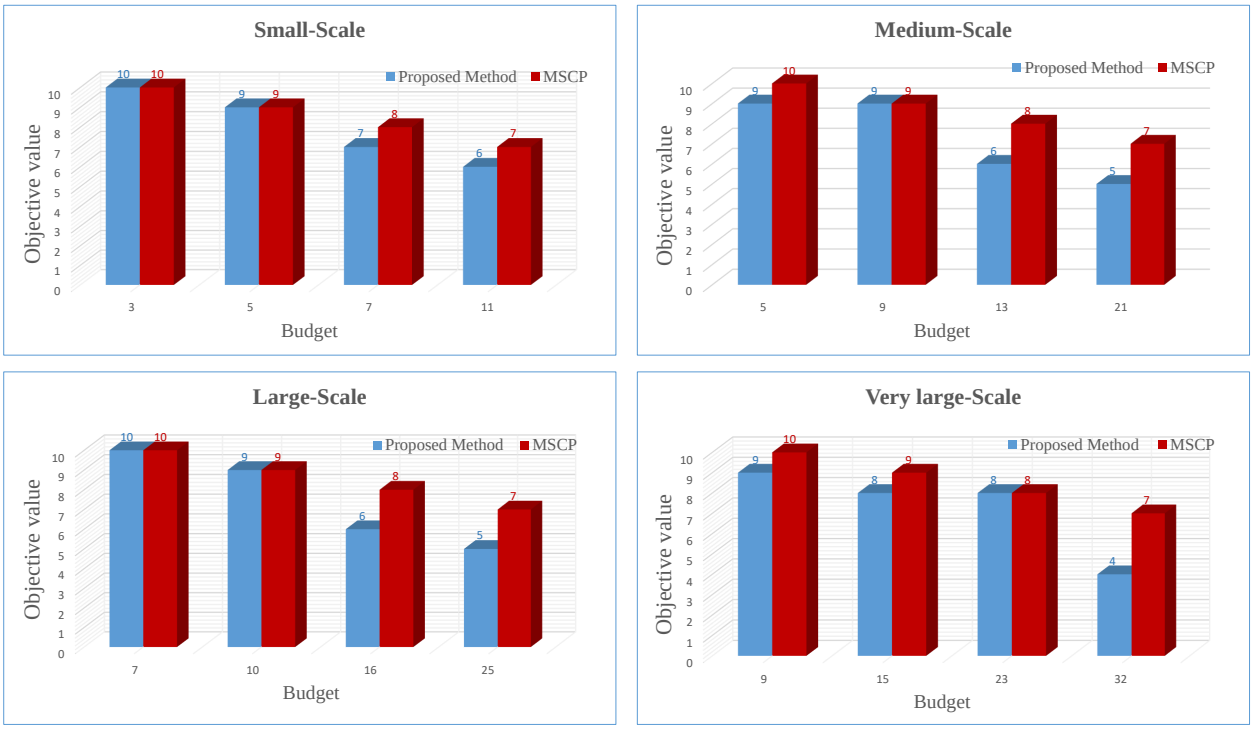

Figure 4. The results of the proposed method in comparison with the state-of-the-art method.

the project Future Factories in the Cloud (FiC), and by the Swedish Research Council (Vetenskapsrådet), through starting grant no. 2018-04582 via the project MobiFog: mobility management in Fog-assisted IoT networks.

References

[1] White paper: Software defined networking: the new norm for networks. Technical report, Open Networking Foundation, April 2015.

[2] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[3] Fouad Benamrane, Francisco J Ros, and Mouad Ben Mamoun. Synchronisation cost of multi-controller deployments in software-defined networks. *International Journal of High Performance Computing and Networking*, 9(4):291–298, 2016.

[4] Hamid Reza Faragardi, Maryam Vahabi, Hossein Fotouhi, Thomas Nolte, and Thomas Fahringer. An efficient placement of sinks and sdn controller nodes for optimizing the design cost of industrial iot systems. *Software: Practice and Experience*, 48(10):1893–1919, 2018.

[5] Md Zakirul Alam Bhuiyan, Guojun Wang, Jiannong Cao, and Jie Wu. Deploying wireless sensor networks with fault-tolerance for structural health monitoring. *IEEE Transactions on Computers*, 64(2):382–395, 2015.

[6] Huping Xu, Jiajun Zhu, and Bang Wang. On the deployment of a connected sensor network for confident information coverage. *Sensors*, 15(5):11277–11294, 2015.

[7] Leila Ben Saad and Bernard Tourancheau. Towards an optimal positioning of multiple mobile sinks in wsns for buildings. *International Journal on Advances in Intelligent Systems*, 2(4):411–421, 2009.

[8] Michael Chien-Chun Hung and Kate Ching-Ju Lin. Joint sink deployment and association for multi-sink wireless camera networks. *Wireless Communications and Mobile Computing*, 16(2):209–222, 2016.

[9] Lanny Sitanayah, Kenneth N Brown, and Cormac J Sreenan. Planning the deployment of multiple sinks and relays in wireless sensor networks. *Journal of Heuristics*, 21(2):197–232, 2015.

[10] Wint Yi Poe and Jens B Schmitt. Placing multiple sinks in time-sensitive wireless sensor networks using a genetic algorithm. In *MMB*, pages 1–15. VDE, 2008.

[11] Haidar Safa, Wassim El-Hajj, and Hanan Zoubian. A robust topology control solution for the sink placement problem in wsns. *Journal of Network and Computer Applications*, 39:70–82, 2014.

[12] Donghyun Kim, Wei Wang, Nassim Sohaee, Changcun Ma, Weili Wu, Wonjun Lee, and Ding-Zhu Du. Minimum data-latency-bound k-sink placement problem in wireless sensor networks. *TON*, 19(5):1344–1353, 2011.

[13] Sirnivas Rao, Haider Banka, and Prasanta Jana. PSO-based multiple-sink placement algorithm for protracting the lifetime of wireless sensor networks. In *Proceedings of the Second International Conference on Computer and Communication Technologies*, pages 605–616. Springer, 2016.

[14] Xuxun Liu. A deployment strategy for multiple types of requirements in wireless sensor networks. *IEEE Transactions on Cybernetics*, 45(10):2364–2376, 2015.

[15] Brandon Heller, Rob Sherwood, and Nick McKeown. The controller placement problem. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 7–12. ACM, 2012.

[16] David Hock, Steffen Gebert, Matthias Hartmann, Thomas Zinner, and Phuoc Tran-Gia. Poco-framework for pareto-optimal resilient controller placement in sdn-based core networks. In *Network Operations and Management Symposium (NOMS)*, pages 1–2. IEEE/IFIP, 2014.

[17] Hamid Reza Faragardi, Hossein Fotouhi, Thomas Nolte, and Rahim Rahmani. A cost efficient design of a multi-sink multi-controller wsn in a smart factory. In *High Performance Computing and Communications; IEEE 15th International Conference on Smart City; IEEE 3rd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2017 IEEE 19th International Conference on*, pages 594–602. IEEE, 2017.

[18] Haidar Safa, Wassim El-Hajj, and Hanan Zoubian. Particle swarm optimization based approach to solve the multiple sink placement problem in wsns. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5445–5450. IEEE, 2012.

[19] Lanny Sitanayah, Kenneth N Brown, and Cormac J Sreenan. Multiple sink and relay placement in wireless sensor networks. In *Proceedings of the 1st Workshop Artificial Intelligence for Telecommunications and Sensor Networks (WAITSâĂŹ12), 20th European Conference on Artificial Intelligence (ECAIâĂŹ12)*, pages 18–23, 2012.