Optimal Device Management Service Selection in Internet-of-Things

Weiling Li[1], Yunni Xia[1(✉)], Wanbo Zheng[2,3(✉)], Peng Chen[4], Jia Lee[1], and Yawen Li[1]

[1] School of Computer Science, Chongqing University, Chongqing, China
weilinglicq@outlook.com, xiayunni@hotmail.com, lijia@cqu.edu.cn
[2] Data Science Research Center, Kunming University of Science and Technology, Kunming, China
zwanbo2001@163.com
[3] Faculty of Science, Kunming University of Science and Technology, Kunming, China
[4] School of Computer and Software Engineering, Xihua University, Chengdu, China
Chenpeng1980@163.com

Abstract. In Internet-of-Things (IoT), IoT device management is a challenge for device owners considering the huge amount of devices and their heterogeneous quality of service (QoS) requirements. Recently, IoT device management service (MS) providers are arising to serve device owners. Device owners can now easily manage their devices by using IoT device MSs. It is critical to select suitable MSs from numerous candidates for devices. An optimal service selection must maximize the number of MS managed devices and minimize the total cost while ensuring the QoS requirements of IoT system. To optimize the IoT Device Management Service Selection problem, we propose IDMSS, a Lexicographic Goal Programming (LGP) based approach. However, due to the high computational complexity of the IoT Device Management Service Selection problem, an alternative heuristic-based approach called GA4MSS is proposed. Two series of experiments have been conducted and the experimental results show the performance of our approaches.

Keywords: Internet-of-Things (IoT) · IoT Device Management · Vector bin packing problem · Genetic Algorithm (GA)

This work was supported in part by the International Joint Project through the Royal Society of the U.K., in part by the National Natural Science Foundation of China under Grant 61611130209, in part by the National Science Foundations of China under Grants 61472051/61702060, in part by the Science Foundation of Chongqing under Grant cstc2017jcyjA1276, in part by the China Postdoctoral Science Foundation under Grant 2015M570770, in part by the Natural Science Foundation of Chongqing under Grant cstc2016jcyjA1315, and in part by the National Key R&D Program of China under Grant 2018YFD1100304.

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2019
Published by Springer Nature Switzerland AG 2019. All Rights Reserved

1 Introduction

1.1 Background

Internet-of-things (IoT) [1], which integrates distributed smart objects, burgeoning technologies and communications solutions [2], e.g., tracking technologies and enhanced communication protocols, has become a promising paradigm for smart systems such as smart cities and healthcare [3].

IoT is the network of devices, e.g., sensors and actuators. The devices sense the physical world and take reactions to specific scenarios. To achieve a smart system, the system builder should (a) deploy sufficient and specific designed IoT devices to specific environment or space, (b) interconnect deployed devices by some cores. However, due to the high difficulty of owning all-round management techniques and resources [4,5], it is not a easy job for many IoT device owners to maintain such huge amount of devices. In such condition, IoT management service providers (MSP) are arising to catch business opportunities. IoT MSPs usually provide rules engine for users which makes it possible to build IoT applications without managing any infrastructure. They also support a wide range of communication protocols and even allow IoT devices to communicate with each other while they are using different protocols. For example. Amazon provides MSs called AWS IoT [6], which provides all aforementioned features and extensions like device shadow for device owners.

1.2 Motivating

According to Ericsson's Mobility Report [7], by 2023, there will be around 32 billion connected devices. Considered to be a growing market, its great economic benefit attracts many organizations to provide their own MS. It is predictable that there will be more and more MSPs. By utilizing cloud computing and edge computing, the MSs are usually convenient, reliable and economical efficiency. Adequate utilization of IoT device MS allows device owners to improve their own business. However, in practical scenarios, the heterogeneous QoS requirements of devices and the heterogeneous capacities of services make IoT device owners more difficult to work out a plan for device management optimization.

Motivated by this need, in this study, our objectives are modeling the problem and providing approaches to solve it. First of all, the constraints in the problem are investigated.

To build a large-scale IoT application, such as smart grid [8], a large amount of IoT devices should be deployed. These devices, such as sensors and monitors are spread among a large space which makes it hard to connect them to one MS due to the nonfunctional requirements like Quality-of-service (QoS) [9].

QoS requirement of an IoT device is multiple dimensional. It is naturally, for example, IoT devices for remote health monitoring and emergency notification systems have stringent demands on performance and reliability for real-time communication, and in smart grid, to satisfy certain security requirements, data collected by video monitors should be transmitted to MS for analysis within a limited time frame to detect potential threats, which demands sufficient throughput. Most of these QoS requirements are quantifiable.

According to the European Telecommunications Standards Institute (ETSI), the typical response times of different IoT functions should meet the values in Table 1.

Table 1. Typical response times of different IoT functions

Function	Response time
Protection	1 to 10 ms
Control	100 ms
Monitoring	1 s
Metering/Billing	1 h to 1 day
Reporting	1 day to 1 year

For any IoT device, there might be list of satisfiable and selectable MSs. However, a MS cannot bear all application devices due to the capacity limitation. It is clear that in the era of edge computing, a MSP can deploy MS on edge servers, whose computing resource such as CPU, bandwidth or memory are limited.

Number of devices, service capacity, QoS requirement of a certain device and the QoS prediction data between any device and service can be obtained or calculated. Based on this information, while fulfilling the above constraints, the number of devices managed by MS must be maximized. Due to the aforementioned constraints, there might be a number of devices that cannot be assigned to MS. Those devices will managed by device owners with extra resource. Additionally, minimize the total cost of renting management service is another optimization objective.

1.3 Our Work

In this study, we refer to the above problem as a Constraint Optimization Problem (COP). The IoT Device Management Service Selection problem is proven to be $\mathcal{NP}$-hard. Two approaches have been proposed to solve it. The main contributions of this work are as follows:

- The IoT Device Management Service Selection problem is modeled as a COP and we have proven its $\mathcal{NP}$-hard;
- we have developed an optimal approach for solving the COP problem using the Lexicographic Goal Programming technique;
- A genetic algorithm (GA)-based method has been proposed as an alternative approach to solve the COP problem.
- we have evaluated our approaches against a baseline approach with experiments to demonstrate their performance.

The remainder of the paper is organized as follows. Section 2 models the problem. In Sect. 3, we prove proposed COP problem is $\mathcal{NP}$-hard and provide a solution. Section 4 proposes an alternative approach. Section 5 evaluates proposed approaches and Sect. 6 concludes this paper.

2 System Modeling

Let $V_1 = \{d_1, d_2, ..., d_m\}$ represent a set of IoT devices, where $m = |V_1|$ is the size of set V_1 and $V_2 = \{s_1, s_2, ..., s_n\}$ a set of candidate MSs, where $n = |V_2|$ is the size of set V_2. Obviously, the vertex sets V_1 and V_2 are two disjoint sets that $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$. The potential assignment between IoT devices and services can be presented by a set of edges $E \in V_1 \times V_2$, such that every edge $e \in E$ has one vertex in V_1 and the other in V_2. Therefore, the relationship between IoT devices and device management services can be presented by a Bipartite Graph $G = (V_1, V_2, E)$. Then, the solution space of concerned problem can be presented as a $0-1$ matrix.

For any IoT device d_i in V_1, there are totally n potential services and naturally $e_{ij} = 1$ if device i is managed by service j.

The infrastructure of management service can be Cloud data center, 5G base station or other computational resources. Their scales are obviously heterogeneous. Consequentially, the maximum capacity of management services are many and varied. In this study, the maximum connectivity, e.g., c_j, determines the service capacity. Specifically, for a service s_j, the number of connected devices is limited to c_j.

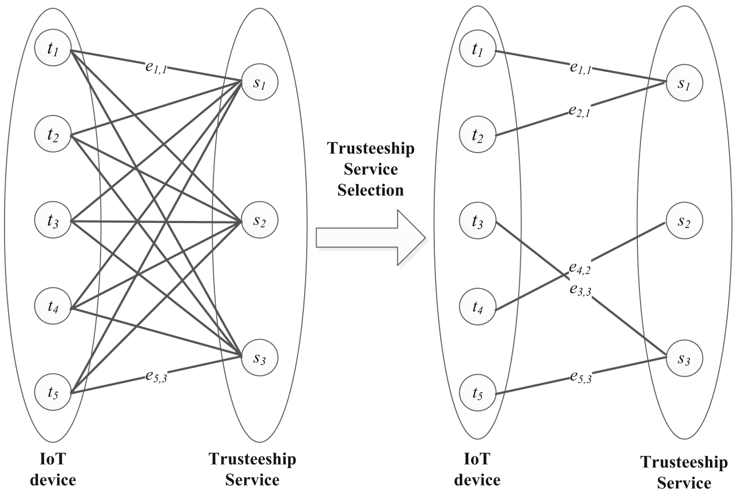

Fig. 1. A breif example of proposed COP problem.

To fulfil its function smoothly, an IoT system requires its components follow the QoS requirements. Therefore, selecting qualified MSs is critical. As aforementioned, different IoT devices have different QoS requirements. Before allocating an IoT device to a management service, the QoS data is required first to support such decision. However, in IoT scenarios, the device-side QoS performance, e.g., response-time and throughput, are highly different due to many factors such as network conditions and deployment environment. Considering the large number of IoT devices and candidate services, the high expense of taking real-world service evaluations is unacceptable. It is a commonly dilemma for QoS-based service selection approaches. Fortunately, these QoS data can be predicted by using QoS prediction techniques, e.g., collaborative filtering (CF)-based approaches [12,13]. Especially, the latent factor (LF)-based predictors [14–16] are proven to be highly accurate.

Suppose an IoT device has z independent QoS requirements. The source data for the QoS predictors is a 3-dimensional matrix, e.g., $H^{m\times n\times z}$ which contains numerous missing entries. The QoS predictors firstly separate $H^{m\times n\times z}$ into z $H^{m\times n}$ matrices, and implement the QoS prediction to complete them. For any QoS $H_k^{m\times n}, 1 \geq k \leq z$, the entry q_{ij}^k indicates the k^{th} predicted QoS data between device i and service j. Suppose k^{th} dimension Qos requirement of device i is $\hat{q_i^k}$. Comparing each q_{ij}^k with $\hat{q_i^k}$ by a specific rule, e.g., $<$ or $>$, we can obtain a selectable service list notated by $g_k(i)$. The intersection of z $g_k(i)$, e.g., $g(i)$, is a list of selectable services for device i.

The price of device management service depends on both device and service. According to AWS IoT Core pricing, the price is determined by Connectivity, e.g. number of devices and duration, Messaging, e.g., message number and message size, and Rules Engine. In this study, a matrix $P^{m\times n}$ contains the price information and p_{ij} denotes the money cost by device i on service j in a unit time.

Additionally, in this study we suppose that any IoT device is only managed by one service, then we have the condition (Fig. 1),

$$\sum_{j=1}^{n} e_{ij} = 1, \forall i \tag{1}$$

Until now, all constraints of our optimization problem are clear. As aforementioned, the optimization has two objectives: (1) maximizing the number of devices connected to MS and (2) minimizing the total cost, while satisfying the capacity constraint and QoS constraint. Then we have modeled the IoT Device Management Service Selection problem as a constraint optimization problem (COP):

$$Maximize.F = \sum_{i=1}^{m}\sum_{j=1}^{n} e_{ij} \tag{2}$$

$$Minimize.R = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}e_{ij} \tag{3}$$

subject to

$$\sum_{j=1}^{n} e_{ij} = 1, \forall i \in \{1, ..., m\} \tag{4}$$

$$\sum_{j \in g(i)} e_{ij} = 1, \forall i \in \{1, ..., m\} \tag{5}$$

$$\sum_{i=1}^{m} e_{ij} \leq c_j, \forall j \in \{1, ..., n\} \tag{6}$$

$$e_{ij} \in \{0, 1\}, \forall i \in \{1, ..., m\}; j \in \{1, ..., n\} \tag{7}$$

where:

$e_{ij} = 1$ if device d_i is allocated to service s_j.

g_j is provided by CF-based QoS predictor.

c_j is provided by MSP.

The objective function (2) maximizes the number of devices that are assigned to management service. The objective function (3) minimizes the total cost of the management. Note that objective (2) has the higher rank compared to objective (3). Constraint family (4) ensures each device is allocated to at most one management service. family (5) ensures the QoS requirement of any device based on the result of QoS predictor. Constraint family (6) ensures the number of devices allocated to any service won't exceed its capacity.

3 IDMSS Approach

In this section, we analyse the aforementioned COP problem and proposed an approach to solve it directly. First, we introduce the definitions of Bin packing (BP) Problem and Vector Bin Packing (VBP) Problem, which are similar with our COP.

Definition 1. ***Bin Packing (BP) Problem.*** *Given an infinite supply of identical bins $B = \{b_1, b_2, ..., b_m\}$ and the maximum capacity of any bin b_i, C_i, equals 1. And a set of n items $U = \{u_1, u_2, ..., u_n\}$. The size of a item u_j, w_j, satisfies $0 < w_j \leq C$. The objective of **BP** is to pack all the given items into the fewest bins such that the total item size in each bin must not exceed the bin capacity C.*

Definition 2. ***Vector Bin Packing (VBP) Problem.*** *Given a set of items $U = \{u_1, u_2, ..., u_n\}$, the size of an item u_j is denoted as a k-dimensional vector $w_j =< w_j^1, w_j^2, ..., w_j^k >, w_j \in [0, 1]$. given an infinite supply of identical bins $B = b_1, b_2, ..., b_m$ with maximum capacity $C =< 1^1, 1^2, ..., 1^d >$. The objective is to pack the set U into a minimum number of bins.*

BP problem is known to be an $\mathcal{NP}$-hard combinatorial optimization problem [10]. The size of an item is presented as a single aggregate measure in BP problem. By contrast, the size of an item in the VBP problem is associated with a multi-dimensional vector and the VBP problem is also known as multi-capacity BP problem in some literatures, which is $\mathcal{NP}$-hard.

Proof. In our problem, there is a set of devices $D = \{d_1, d_2, ..., d_m\}$ and a set of management services $S = s_1, s_2, ..., s_i$, the selectable services of a device d_i is denoted as a n-dimensional vector $e_i =< e_{i1}, e_{i2}, ..., e_{in} >, e_{ij} \in 0, 1$. The maximum service capacity $C =< c_j * e_{1j}, c_j * e_{2j}, ..., c_j * e_{mj} >$. The objective is to pack maximum elements in set D into a fix number of bins. VBP requires $d > 2$, in our COP, m is a huge number and $\sum_{i=1}^{M} e_{ij} \gg d$, so our COP is equivalent to VBP.

To solve the Proposed COP, Lexicographic Goal Programming (LGP) [11] technique is a valid solution. LGP is suitable to solve multi-objective optimization problem whose optimization objectives are ranked by their levels of importance, or priorities. The LGP solver first finds optimal solutions satisfied the primary objective and then proceeds to find solutions for the next objective without deteriorating the previous objective(s).

An LGP program can be solved as a series of connected integer linear programs, which can be easily solved by commercial computing tools such as IBM CPLEX Optimizer. This direct approach is named as IDMSS in this study.

4 GA-based Approach

As proven in Sect. 3, the proposed COP problem is $\mathcal{NP}$-hard. The traditional methods are exhausted considering the solution space of proposed COP problem can be very big in practical. Therefore, we proposed an alternative approach called GA4MSS, a Genetic Algorithm-based approach for MS Section.

Genetic algorithms (GAs) [17,18] belongs to a subset of heuristic algorithms. They are inspired by the natural biological evolution and proven to be robust and performed well in most cases. The GAs operate the solution, which is in the form of chromosome, with genetic operators such as crossover and mutation to boost the evolution. This process may create solutions approximate the optimum. Basic GA components and their relationship are shown in Fig. 2.

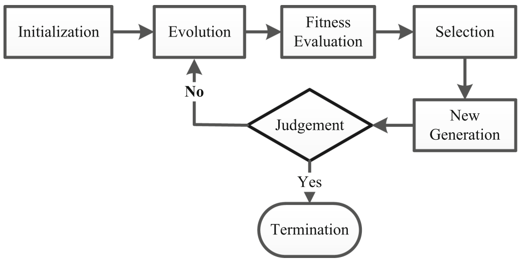

Fig. 2. A flow chart of basic genetic algorithms.

In GAs, solutions are encoded as chromosomes. GA4MSS encodes a possible solution as a single string, as shown in Fig. 3. The index of the string element denotes the device id, e.g., index 1 is device d_1. The values in each position denotes the id of a service. The encoding method promises that each device can only select one service, which makes GA4MSS satisfies the constraint family (4). Moreover, to satisfies the constraint family (5), the value in each position of the chromosome can only choose from the selectable set, e.g., $g(i)$. According to such encoding method, GA4MSS covers all possible QoS-aware solutions and Z solutions are generated as the Initial Population.

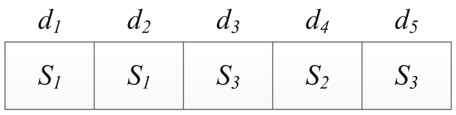

Fig. 3. GA4MSS encoding.

To evaluate the fitness of a solution, a fitness function is required. GA4MSS combines two objectives by punishing the unconnected devices. The punishment is implemented by adding an extra cost to the solution as follows:

$$T = \sum_{i=1}^{m} (\lambda \times p_{ij} \times \hat{e_i}), \forall j \tag{8}$$

where $\hat{e_i} \in \{0, 1\}$ denotes management state of a device, $\hat{e_i} = 1$ if device d_i connects no service, and λ is a $(1, \infty)$ parameter to estimate the potential cost of managing the devices by owners themselves. It is reasonable because if there is no qualified management service, the device owner should manage these devices by themselves, which leads to a higher cost. Moreover, GA4MSS assimilates the capacity constraint into the fitness function by punishing the exceeding devices. The fitness function of GA4MSS is as follows,

$$Minimize.R = \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} e_{ij} + \sum_{i=1}^{m} (\lambda \times p_{ij} \times \hat{e_i}) \tag{9}$$

Given the fitness function, the GA4MSS can iteratively approximate the optimum by operating the gene pool. GA4MSS has 3 basic operators, namely crossover, mutation and selection.

Crossover operator is designed to encourage the recombination of individual features in current population in order to produce better offsprings. In GA4MSS, it consists of 4 steps, which are explained in what follows with the help of Fig. 4:

1. generate two replicas of two individuals which are randomly chosen from current population.
2. Randomly generate a 0–1 crossover indicator.
3. swap the i^{th} element of two replicas if i^{th} value in crossover indicator is 1.
4. repeat 1–3 until a number of individuals, e.g., $2Z$, in the population.

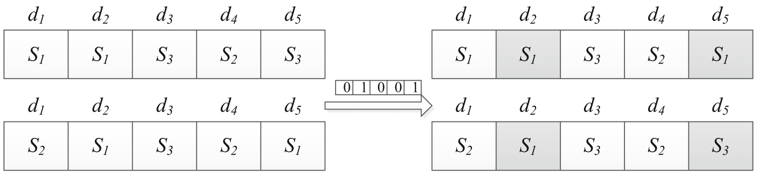
d_1 d_2 d_3 d_4 d_5
S_1 S_1 S_3 S_2 S_3
S_2 S_1 S_3 S_2 S_1
0 1 0 0 1
S_1 S_1 S_3 S_2 S_1
S_2 S_1 S_3 S_2 S_3

Fig. 4. GA4MSS crossover operation

Mutation operator allows an offspring to obtain features which are not owned by its parents. GA4MSS randomly exchange a value in d_i with a element in $g(i)$.

Selection operator works as a filter which allows a part of the operated population into next generation. There are Kinds of selection schemes, GA4MSS implements the truncation. The individuals in current population after crossover and mutation is sorted by their fitness value, and the first Z individuals survive.

5 Experimental Evaluation

5.1 Experiment Settings

We have conducted a range of experiments aiming at evaluating the effectiveness and efficiency of proposed approaches. In this study, all experiments are conducted on a Windows machine with Intel(R) Xeon(R) E5-CPU and 32 GB RAM. The IDMSS approach in Sect. 3 was implemented using IBM ILOG CPLEX Optimizer and the GA4MSS was implemented using Java SE.

Our approaches will be benchmarked against a baseline approach called RANDOM. In RANDOM, each IoT device will be assigned to a random service. A device won't be account to be a MS managed device if it violates the QoS requirement or it exceeds the capacity of assigned service.

Experiment Data. A QoS data set collected by the WS-Dream system is utilized as the experiment data. The data set contains 1873838 response-time data by 339 users on 5825 real world Web-services. To implement our experiment, we first employed INLF [16], a non-negative latent factor QoS predictor to predict the missing QoS data. The price of managing a device by a MS is randomly generated based on the basic price and a random factor τ. In this study, the basic price is 0.10 dollar, which is a standard fee charged by AWS GovCloud for one connection.

Performance Metrics. We evaluate three approaches, namely IDMSS, GA4MSS and the RANDOM baseline approach by following metrics: (1) the percentage of MS managed IoT devices of all IoT devices, the higher the better; (2) total cost including the estimating cost of the devices managed by device owner himself, the lower the better; and (3) the execution time (CPU time), the lower, the better.

Given the data and the experimenting parameters, we conduct two sets of experiments. For each set, we vary one parameter and keep the other fixed to observe the impact of each parameter. The evaluation metrics are as shown in Table 2.

Table 2. Parameter Settings

Set	capacity	Basic unit Price	τ	λ	Number of devices	Number of services
#1	20	0.10	(1, 1.5]	2	100, 200,...,1000	100
#2	20	0.10	(1, 1.5]	2	1000	20, 40,...,100

5.2 Experimental Results

Effectiveness: Figures 5 and 6 show the results obtained from the experiment set 1 and set 2, respectively. The three performance metrics are depicted in each sub-figure: (a) percentage of MS managed devices, (b) total cost, and (c) gives the execution time of each approach.

Figure 5 shows that in experiment set 1, when the number of IoT devices increases from 100 to 1000, the random approach performs poorly in terms of trusteeship device percentage (only 38%–40% of the IoT devices are assigned to the MS) compared to our approaches. Proposed approaches perform approximately equal that most devices having been managed by MS. Comparing to GA4MSS, IDMSS can always find a better solution, although the trusteeship device percentage keep decreasing as the number of devices increases, the decreasing of IDMSS is obviously slower note that the gap between IDMSS and GAMSS is approximately growing from 0.05% to 3.7%.

In experiment set 2, we change the number of candidate services. As depicted in Fig. 6(a), trusteeship device percentage increases while the number of services increases from 20 to 100. IDMSS continues to achieve the best performance. Regarding to Fig. 6(b), the total cost decreasing largely as the service capacity increases.

Efficiency: Experiment set 1 shows that the computation time of IDMSS approach increases considerably while we increasing the number of IoT devices. As shown in Fig. 5(a), when there are 1000 devices, the GA4MSS and random approaches take only approximately 4.3 and 1.4 s while IDMSS takes around 59.0 s to solve an instance of the IoT device management selection problem. In experiment sets 2, where we increase the capacity of service, the IDMSS approach consumes more time to make a decision. Since proposed COP problem is $\mathcal{NP}$-hard, it is expected that IDMSS approach, which optimally solves the problem, will take the most time as opposed to the other approaches.

Increasing number of IoT devices or number of services will increase the complexity of proposed COP problem, which is $\mathcal{NP}$-hard, and thus take more time

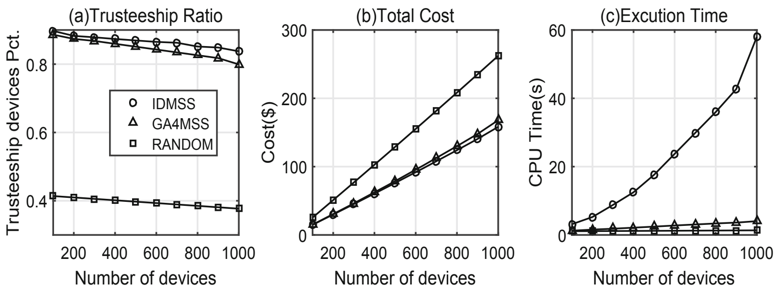

Fig. 5. Results of set #1 (number of devices changing)

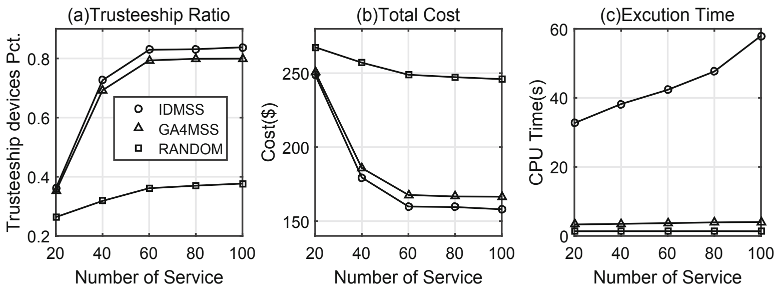

Fig. 6. Results of set #2 (number of services changing)

to find an optimal solution. Our experimental results show that the random approach is not able to optimize the optimization objectives as it can only connect around only 40% of all the IoT devices in the experiments by chance. IDMSS and GA4MSS have similar effectiveness; they are able to assign a similar number of IoT devices to MS. However, as shown in Figs. 5(c) and 6(c), GA4MSS outperforms IDMSS in performance especially while the COP problem is scaling up.

6 Conclusion

IoT device management is a challenge to many device owners considering the huge amount of devices and their heterogeneous characteristics. Recently, IoT device management service (MS) providers raise to serve device owners, e.g., AWS IoT core. IoT device owners can easily do their management work by using IoT device MS. However, it is not a easy job to select suitable service for each devices considering the constraints, e.g., QoS constraint and capacity constraint. An optimal service selection must maximize the number of MS managed devices and minimize the total cost while ensuring the required QoS of devices. To address this problem, we model the IoT Device Management Service Selection problem as a COP and solve it by a Lexicographic Goal Programming (LGP) approach. At the mean time, an alternative approach, GA4MSS, is proposed to find the approximate optimal solution within shorter time. Our experimental results show that our approaches significantly outperforms the baseline random approach and each of the approach has its own advantages. This research has established a basic foundation for the IoT device management service selection problem and in the future we will (a) improve the performance of GA4MSS, and (b) consider more scenarios in this problem, such as IoT devices' mobility and service price volatility.

References

1. Gubbi, J., Buyya, R., Marusic, S., et al.: Internet of Things (IoT): a vision, architectural elements, and future directions. Future Gener. Comput. Syst. **29**(7), 1645–1660 (2013)
2. Han, K.H., Bae, W.S.: Proposing and verifying a security-enhanced protocol for IoT-based communication for medical devices. Cluster Comput. **19**(4), 1–7 (2016)
3. Amendola, S., et al.: RFID technology for IoT-based personal healthcare in smart spaces. IEEE Internet Things J. **1**(2), 144–152 (2014)
4. Perumal, T., Datta, S.K., Bonnet, C.: IoT device management framework for smart home scenarios. Consum. Electron. (2016)
5. Guo, C., et al.: A social network based approach for IoT device management and service composition. IEEE World Congr. Serv. (2015)
6. AwS IoT Core Homepage. https://aws.amazon.com/iot-core/. Accessed 4 Mar 2019
7. Heuveldop, N.: Ericsson Mobility Report. Technical report, Ericsson, November 2017
8. Yun, M., Bu, Y.: Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid. In: International Conference on Advances in Energy Engineering (2010)
9. Aazam, M., et al.: MeFoRE: QoE based resource estimation at Fog to enhance QoS in IoT. In: International Conference on Telecommunications (2016)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
11. Kwak, N.K., Schniederjans, M.J.: An alternative solution method for goal programming problems: the lexicographic goal programming case. Socio Econ. Plann. Sci. **19**(2), 101–107 (1985)
12. Zheng, Z., Zhang, Y., Lyu, M.R.: Distributed QoS evaluation for real-world Web services. In: IEEE International Conference on Web Services (2010)
13. Luo, X., et al.: Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models. IEEE Trans. Neural Netw. Learn. Syst. **27**(3), 524–537 (2016)
14. Zheng, Z.B., Ma, H., Lyu, M.R., King, I.: Collaborative web service QoS prediction via neighborhood integrated matrix factorization. IEEE Trans. Services Computing. **6**(3), 289–299 (2012)
15. Luo, X., Zhou, M., Xia, Y., et al.: An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans. Industrial Informatics. **10**(2), 1273–1284 (2014)

16. Luo X., et al.: An inherently non-negative latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE Trans. Ind. Inf. (2017)
17. Anderson-Cook, C.M.: Practical genetic algorithms. Publ. Am. Stat. Assoc. **100**(471), 1099 (2004)
18. Canfora, G., et al.: An approach for QoS-aware service composition based on genetic algorithms. In: Conference on Genetic & Evolutionary Computation (2005)