Delay-Based Dynamic Clustering Method for the IoT Cluster

1st Seo Jin Chang
Dept. of Computer Science
University of California, Irvine
Irvine, California
schangeojin@gmail.com

2nd Boyeon Kim
Dept. of R&D
Sekyung BMT Co.
Seoul, Republic of Korea
bykim9988@gmail.com

3rd Yunseok Chang
Dept. of Computer Engineering
Daejin University
Pocheon, Republic of Korea
cosmos@daejin.ac.kr

***Abstract*—A dynamic clustering method can reconstruct IoT nodes into logical group units when using several groups of IoT nodes. However, massive control at a logical group level does not guarantee optimal control performance. To efficiently deliver and execute a massive control command within a certain communication delay, it is necessary to cluster each logical group into subgroups and execute massive control commands by each subgroup. This paper proposes the DBDC(Delay-Based Dynamic Clustering) method, a clustering method based on DBSCAN that can optimize the communication delay of a massive control within a logical group. Through several simulations, the parameters of the DBSCAN algorithm create different clustering cases, and the DBDC method can find the optimal parameter value that satisfies the optimization condition. Therefore, this study shows that the DBDC method can effectively design a cluster system so collective IoT clusters have the best massive control performance.**

***Index Terms*—Delay Based Dynamic Clustering, IoT cluster, DBDC method, DBSCAN algorithm, Logical group**

I. Introduction

The development of IoT technology has primarily contributed to the remote control field that controls various devices through the Internet. Such IoT technology is adopted on a diverse scale: a small scale of IoT usage includes households and offices that use several to dozens of devices, and a large scale includes architectures such as smart buildings, smart factories, and smart campuses. Recently, there have been increasing cases where IoT technologies based on cutting-edge smart systems are utilized in large-scale areas. Generally, large-scale areas are designed and divided into subspaces by their functions and purposes. In such spaces and subspaces, various IoT devices are installed by their purposes and usages and controlled by the smartphone or smart pad through cutting-edge IoT control systems that collectively control IoT devices through central control or distributed control [1]. Collective IoT control technology has an extensive application field, for example, in the case of architectures like smart buildings. There has been an increasing need for the collective operation of IoT devices and sensors in smart cities that employ many environmental IoT sensors, such as air pollution, temperature, and humidity [2].

In an IoT cluster environment that uses collective IoT devices, IoT devices are installed in specified spaces by clustering method and perform massive control by segment and group [3]. Static clustering and dynamic clustering are typically used to cluster numerous IoT devices. In the static clustering method, the cluster structures are designed and installed statically in the system planning stage. Since the static clustering method clusters IoT devices by the physical architecture and arrangement of their network equipment and cables, it requires more costs and effort when reinstalling IoT devices and systems or modifying segments of IoT clusters. Also, as the users' requirements or the uses of subspace change, it is difficult to accommodate those changes to IoT clusters. On the other hand, dynamic clustering uses a method that logically composes IoT devices connected by a cluster regardless of their physical network connection and performs massive control by their logical groups [4]. Compared to static clustering, dynamic clustering has a flexible system reconfiguration capability by freely constructing groups, which is the basic unit of massive control, without a change in physical network architecture, thereby efficiently providing flexibility on changes in the architecture of the space where IoT is applied [5].

Static clustering techniques can organize massive control groups so that network communication delays can be physically optimized when collective control is performed. On the other hand, in a dynamic clustering method with logical groups, optimizing communication delay for IoT devices belonging to the same group is difficult because the IoT devices are clustered according to the user's logical requirements. In the dynamic clustering method, IoT devices belonging to a specific logical group would belong to the different physical networks. So, the communication delay while the same collective control command is delivered to IoT devices belonging to the logical group may vary for each device. For this reason, communication delays in the dynamic clustering method using logical groups would be more significant than those in the static clustering method, which degrades the performance of the entire IoT cluster. To improve this problem, it is necessary to break a logical group into several subgroups in the dynamic clustering method and perform collective control on these subgroups in parallel. In other words, to improve the overall collective control performance by dividing the collective control of one logical group into several subgroups, it is necessary to re-cluster IoT devices in the logical group

979-8-3503-2028-2/22/$31.00 ©2022 IEEE
DOI 10.1109/CSCI58124.2022.00270

to optimize communication delays in each subgroup.

This paper presents a DBDC(Delay-Based Dynamic Clustering) method based on the DBSCAN algorithm to optimize collective IoT control communication delay and simulates whether the proposed method can effectively improve the communication delay within the logical group.

II. Related Works

A. Clustering Algorithm

Clustering aims to group data with high similarities in the same class and separate data with low similarities in separate classes[6]. In the clustering process, deciding how many groups to create and how to define the data similarity is a critical problem. The K-Means algorithm is a well-known method that resolves such a problem [7].

The K-Means algorithm finds the expected clusters or number of clusters, the minimum value of the mean distance from each data, and the center of the cluster that the data belongs to from the given data set. Therefore, the K-Means algorithm randomly locates K centroids and forms clusters by assigning data to the closest centroid. Then, based on the data assigned to specific clusters, it repeatedly updates the centroid of each cluster until the centroids are no longer updated. As centroids are wholly updated, it can decide which cluster the data should be assigned to when new data is entered. Indeed, to properly perform this process, it is essential to decide the appropriate number of clusters that should be created from the entire dataset. If correctly clustered, samples in each cluster will be clustered by their density at a close distance. The density of data in clusters can be computed by the inertia, the distance from each data, and the centroid of the cluster it belongs to. We can say the data is suitably clustered if the inertia is low.

The K-Means algorithm can converge fast and powerfully interpret a model. However, if the dataset is non-convex, it is difficult to converge and often finds just a locally optimal solution. Another clustering method that redeems those issues is the Hierarchical clustering algorithm(HCA) [8]. Since HCA can easily define the distance and similarity without setting the number of clusters, it can find the hierarchical relationship among the clusters in various shapes. However, HCA often requires a high computational complexity and is sensitive to single values.

B. DBSCAN Algorithm

DBSCAN(Density-Based Spatial Clustering of Application with Noise) algorithm performs based on the density of the data distribution, while K-Means and HCA cluster by the distances among data and clusters groups where the density of data is high. Therefore, when the data is represented as a distribution of 2D or 3D distances, based on specific data of a cluster, if there are n points(data) within radius r, it groups them in a cluster [9] [10]. To apply the DBSCAN algorithm, specific data that acts as a benchmark, the value of r, and the number of $minPts$ that should be included within that radius should be decided. The candidate point of the core point is randomly selected, and if the number of points within radius r from the candidate point k is greater than or equal to $minPts$, they become a cluster, and point k becomes the core point of this cluster. However, if the number of points within radius r from the candidate point$k$ is less than $minPts$, the candidate point k cannot be a core point. In this case, another random point is selected as a candidate point and determined if it can form a cluster. Repeating this process, DBSCAN clusters all points. In the case of massive IoT clustering, a point is an IoT node, and a cluster is a subgroup of a logical group. Consequently, DBSCAN can be applied when re-clustering IoT nodes included in logical groups that form massive IoT clusters into subgroups.

III. Delay-Based Dynamic Clustering Method

A. Logical Group Re-clustering

We can use the existing clustering algorithms, such as K-Means and HCA, to break up a single logical group in IoT clusters consisting of massive IoT nodes into multiple subgroups. However, IoT nodes or network tools that form IoT clusters are installed in specific areas and frequently have an inhomogeneous density distribution instead of a uniform density distribution. Also, the communication delay from data transmission between IoT nodes and network equipment is directly related to their physical space and arrangement. Therefore, when re-clustering a single logical group that consists of dozens to thousands of nodes into several subgroups, DBSCAN, which performs clustering based on density, is more appropriate to optimize the communication delay compared to other clustering methods. Accordingly, this research presents a new clustering method based on DBSCAN. It confirms its appropriateness to improving the communication delay in IoT clusters and the advances in communication delay through experiments.

B. Basic concept of the Delay-Based Dynamic Clustering

As shown in Fig. 1, Dynamic clustering splits the entire system into several logical groups, such as logical group A, logical group B, and logical group C and conducts massive control by logical group unit. IoT devices in each logical group are called nodes. If there are N nodes in a random logical group, one of the nodes becomes a core node(CN_{org}). The cluster control app sends a massive control command to the core nodes of specific logical groups, and the core nodes perform massive control by spreading the command to each node in the same logical group.

Assume the time each node takes to complete an individual control command is the same. Suppose the communication delay that each node i takes to perform its control command from the core node CN_{org} is represented as $delay(CN_{org}, i)$. Therefore, the time a logical group completes a massive control, $Delay_{org}$, is the sum of the time the command

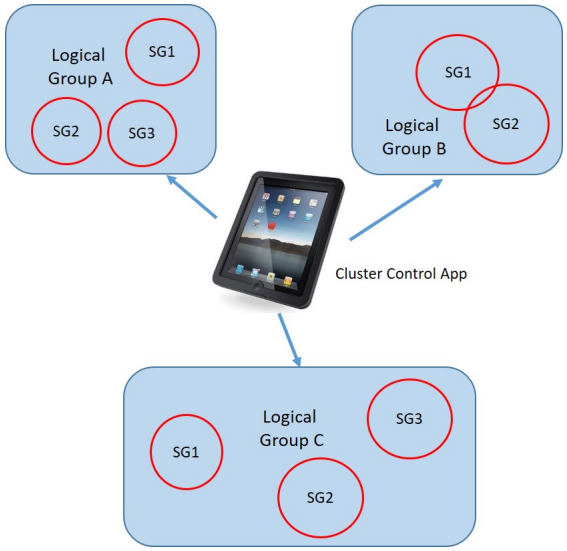

Fig. 1. A large scale IoT control concept with logical groups

is spread from the core node to each node, as defined in Formula (1):

$$Delay_{org} = sum(delay(CN_{org}, 0),\ delay(CN_{org}, 1),\ \ldots,\ delay(CN_{org}, n\text{–}1)) \quad (1)$$

If there are more than 2 subgroups SG_i $(i = 0, 1, ..., m-1, m > 2)$ in a logical group, a core node CN_i is created for each subgroup SG_i. If the communication delay when a core node sends a control command is defined as $delay(CN_i, j)$, the communication delay from each subgroup SG_i is defined as Formula (2), where k_i is the number of nodes in SG_i. When a single logical group is clustered into several subgroups, the control command is sent to each node through the core node CN_i of each subgroup instead of the original core node CN_{org} of a logical group. Accordingly, the total communication delay, $Delay_{total}$, to send the massive control is the sum of the communication delay from the original core node to the core nodes of each subgroup, $delay(CN_{org}, CN_i)$, and the delay from the core nodes of each subgroup to the nodes in the subgroup, $delaySG(i)$. The communication delay that each subgroup SG_i takes to send a massive control command to its nodes, $delaySub(i)$ can be represented as Formula (3).

$$delaySG(i) = sum(delay(CN_i, 0), ..., delay(CN_i, j)) \quad (2)$$

$$delaySub(i) = delay(CN_{org}, CN_i) + delaySG(i) \quad (3)$$

Therefore, when a logical group is clustered into m subgroups, the total communication delay $Delay_{total}$ is defined in Formula (4).

$$Delay_{total} = sum(delaySub(0), delaySub(1), ..., delaySub(m\text{–}2), delaySub(m\text{–}1)) \quad (4)$$

The result of re-clustering a logical group of IoT clusters into several subgroups must satisfy the optimization condition, $Delay_{total} < Delay_{org}$. The $DelayImprovementRatio(DIR)$ of the result that satisfies such a condition is calculated as (5), and the communication delay of a logical group can be optimized by re-clustering it with the highest DIR value.

$$DIR(\%) = \frac{(Delay_{org} – Delay_{total})}{Delay_{org}} \quad (5)$$

In order to optimize the communication delay, we can split a logical group into as many subgroups as possible. However, if there are too many subgroups, the average $delaySub(i)$ converges to the average $delay(CN_{org}, k)$. Therefore, a clustering method that can produce a proper number of clusters should be selected.

In the case of architectures or facilities that apply smart technologies, IoT devices are used to be installed in specific areas with non-uniform distribution. Therefore, among unsupervised clustering algorithms, the DBSCAN algorithm can meet the optimization condition when it clusters a logical group into proper size and number of subgroups and uses the r and $minPts$ values of the DBSCAN method that can optimize DIR to cluster IoT nodes. When clustering a logical group into subgroups, SG_i should be created so that each subgroup gets the minimum $delaySG(i)$. To accomplish this task, selecting a subgroup's core nodes is important. If there are too many subgroups, the communication delay increases as the communication traffic by the core nodes increases. Thus, we should create minimal subgroups and select core nodes so that $delaySG(i)$ can be minimized. This study called this clustering method Delay-Based Dynamic Clustering(DBDC).

The DBSCAN algorithm is based on 2-dimensions or 3-dimensions in the distance to apply the DBDC method, each node in a logical group must be represented as a 2-dimensional coordinate. Similarly, the DBDC method processes the communication delay of IoT nodes as 2-dimensional distance values. Although the communication delay among nodes is determined by communication devices such as switches and nodes, the physical distance of the communication link almost affects the delivery time by the spatial location. To simplify the problem, we assume each node's delay time corresponds to the spatial distance. Like the DBSCAN algorithm, the DBDC method selects a random node in a logical group as a candidate node. It expands to other nodes within a communication delay d from the candidate node. If the number of explored nodes is greater than the size of a cluster, $minPts$, the nodes are grouped as a new subgroup, and the candidate node becomes the core node of the subgroup. After clustering with the DBSCAN algorithm, the nodes classified as noise are assigned to subgroups whose core node is the closest to those nodes.

Fig. 2 is an example that explains the DBDC method, where it clusters a logical group A that consists of 13 nodes with the $minPts$ of 3 and $r = d$ into three subgroups: SG_0, SG_1, and SG_2. Here, nodes 1, 6, and 8 are the core nodes of each subgroup SG_0, SG_1, and SG_2, represented by CN_0, CN_1, and CN_2, and more than three nodes that satisfy $delay(CN_i, j) < d$ are clustered into corresponding

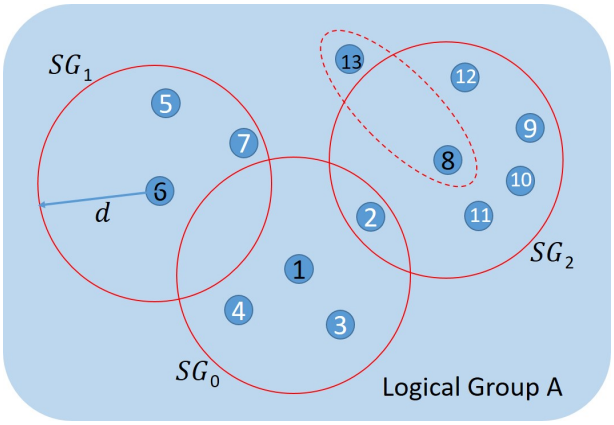

Fig. 2. An IoT example of the clustering with the DBDC method

subgroups. In the case of node 2, while it can be assigned to either SG_0 or SG_2, it is assigned to a subgroup with less $delay(CN_i, j)$. Node 13 is not included in any subgroup called noise in the DBSCAN algorithm, and this node is excluded from clustering. However, the DBDC method assigns such noises to subgroups whose communication delay with the core node is the smallest, represented as dotted lines, which is SG_2 in this case.

IV. Performance Evaluation and Clustering Optimization

Since DBDC is a method that modifies the DBSCAN algorithm, the resulting clusters differ by the value of the parameters such as $minPts$, the size of a cluster, and d, the radius of a cluster. We must decide the parameters properly to get clustering results that satisfy the optimization condition. In this work, we constructed a simulation program and evaluated the performance to assess the performance of the DBDC method for each parameter value. The major target of this performance evaluation is the total communication delay $Delay_{total}$ after clustering and the rate of communication delay improvement DIR if it satisfies the optimization condition. Through a simulation, we evaluated the performance using those two metrics and analyzed how effective the method presented in this article is for massive control of IoT cluster systems.

A. Simulation environments

The simulation program for DBDC method evaluation is written in Python and executed by the Anaconda environment in $JupyterNotebook$ using $iMac$ with Intel $Corei5$ CPU. The simulation input is the communication delay value calculated by the x, y coordinate representing the spatial location of the nodes of a logical group. The simulation results include the number of subgroups after clustering, the nodes in each subgroup, and the performance evaluation.

Fig. 3 graphically illustrates the IoT devices in IoT clusters used in the simulation, which are the spatial locations of the nodes. This data represents the locations of 228 IoT devices and network equipment on the 3rd and 4th floors in the University Research Center where this study is conducted. For convenience, two floors are on the same plane, and the nodes between two floors represent the nodes attached between the floors or on the ceiling. The nodes outside the border of the building represent nodes installed outside the building.

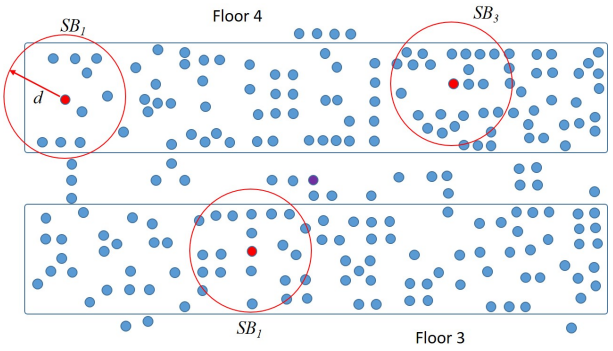

Fig. 3. IoT node distribution sample

In Fig. 3, three different subgroups with different densities SB_1, SB_2, and SB_3 are sampled among all possible subgroups that can be formed with the DBDC method in the simulation. Each subgroup contains nodes within a communication delay d from the core nodes in red at the location where the density is the highest. If no more subgroups can be created, other nodes not in any subgroups are included in a subgroup whose core node is the closest to them. Clustering completes when all nodes are assigned to subgroups and creates a list of the total number of subgroups with the list of nodes.

The basic algorithm implemented in DBDC is the same as DBSCAN. Table I shows the logical groups used in the simulation and the value and range of parameters of the DBSCAN algorithm. In the simulation, a subgroup's radius d and $minPts$ are given as inputs. The simulation program executes the clustering method by repeatedly changing the radius and $minPts$ and calculating $Delay_t otal$.

TABLE I
Simulation Parameters

Parameters	Values(Ranges)
Amount of nodes	228
radius(d)	1.6 2.4 ns
minPts	5, 8, 10, 12, 15
Communication delay	0.2 25 ns

B. Simulation results and analyses

The objective of the simulation is to find the optimal result that gives the highest DIR when clustering a logical group into more than two subgroups with a fixed CN_{org}. Although it is better to simulate as many results as possible, to verify the effectiveness of the DBDC method, this experiment uses

five values for $minPts$ and three for d. Table II shows the results of the simulation.

TABLE II
SIMULATION RESULTS WITH THE DBDC METHOD

d	$minPts$	**# of subgroups**	$Delay_{total}(ns)$	$DIR(\%)$
1.5	5	11	776.7	58.5
1.5	8	2	1,255.7	32.9
1.5	10	1	1,870.9	0.0
1.5	12	1	1,870.9	0.0
1.5	15	1	1,870.9	0.0
1.8	5	3	1,630.9	12.8
1.8	8	7	709.6	62.1
1.8	10	3	1,137.3	39.2
1.8	12	1	1,870.9	0.0
1.8	15	1	1,870.9	0.0
2.1	5	1	1,870.9	0.0
2.1	8	1	1,870.9	0.0
2.1	10	4	1,023.4	45.3
2.1	12	4	967.6	48.3
2.1	15	1	1,870.9	0.0
2.4	5	1	1,870.9	0.0
2.4	8	1	1,870.9	0.0
2.4	10	1	1,870.9	0.0
2.4	12	2	1,599.9	14.5
2.4	15	3	1,145.7	38.8

In the simulation, $Delay_{total}$ in the case of the single subgroup can substitute the $Delay_{org}$ in Table I. Fig. 4 shows one of the clustering results when d = 1.8 and $minPts$ = 8. There are seven subgroups created by the DBDC method, and each subgroup is marked as a specific color. The red dot represents the centroid of each subgroup.

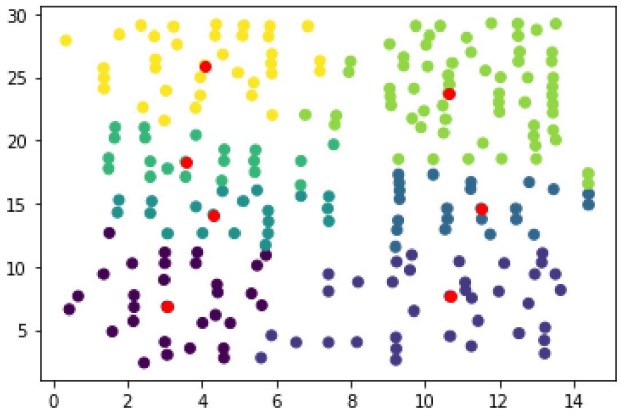

Fig. 4. An example of the clustering with the DBDC method

Fig. 5, Fig. 6, and Fig. 7 show the results of Table II that, performed the simulation by applying the DBDC technique in a graph. In Fig. 5, the graph's x-axis represents the $minPts$ value, and the y-axis represents the DIR. Within the factor range given in Table II, DIR has the largest value when d = 1.8 and $minPts$ = 8, indicating that communication delay can be improved the most when clustering is performed with the corresponding factor value. However, for all experiments, DIR is not improved. The simulation shows that if the d value is 1.8 or 2.1, the DIR value increases by clustering only if the $minPts$ value is within a certain range. It shows that when the d value is determined in clustering, the improvement rate DIR of the communication delay varies greatly depending on the $minPts$ value. Therefore, it can be confirmed through this simulation that the DBDC technique can determine the $minPts$ value that can derive the maximum DIR if an appropriate d value is determined when clustering one logical group into a subgroup.

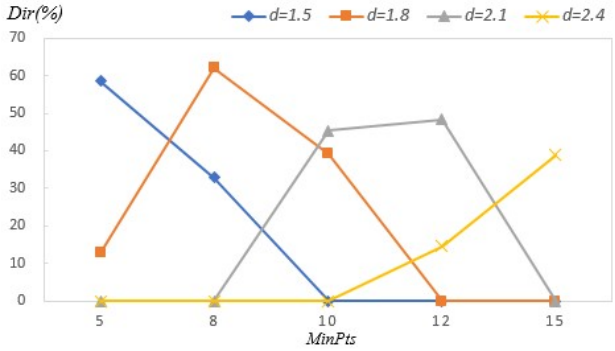

Fig. 5. Simulation result: DIR vs. d and $minPts$

Fig. 6 shows the sum of the communication delays from clustering by the overall DBDC technique. As in Fig. 5, the communication delay time varies greatly by the $minPts$ value when the d value is determined.

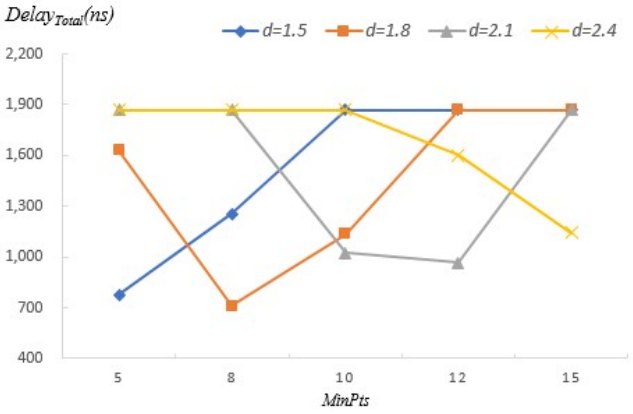

Fig. 6. Simulation result: $Delay_{total}$ vs. d and $minPts$

Fig. 7 shows the number of subgroups created as a result

of clustering. Like Fig. 5 and Fig. 6, the number of subgroups depends on the $minPts$ value for the value of d. Since the number of subgroups and nodes included in the subgroup is related to overall communication traffic, clustering into appropriate subgroups can prevent further communication delays. Simulation results also show that the most frequently generated subgroup in Fig. 7 (d = 1.5, $minPts$ = 5) does not correspond to the highest DIR in Fig. 5 (d = 1.8, $minPts$ = 8). Therefore, creating a large number of subgroups does not necessarily increase DIR. It shows that the DBDC technique presented in this study can effectively provide design elements that can perform effective clustering for clusters with concentrated regional densities.

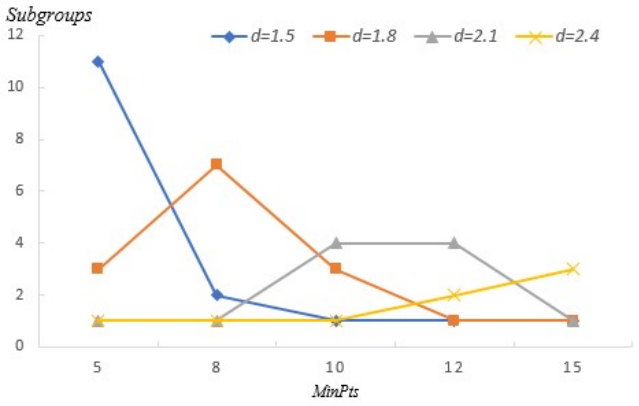

Fig. 7. Number of subgroups vs. d and $minPts$

V. Conclusion

To enhance the performance of massive control in a large-scale IoT system that uses a dynamic clustering method, it must minimize the communication delay in each logical group. To this end, this study employs DBSCAN, one of the unsupervised learning methods, to implement the DBDC method that clusters a logical group into several subgroups. The suggested method can reduce the general communication delay of a massive control command for a specific logical group by splitting it into multiple subgroups and having them execute the command in parallel. However, if too many subgroups exist, a cluster's massive control performance will decrease as the communication traffic increases. Based on the parameter values of the DBSCAN algorithm, the DBDC method can cluster logical groups into subgroups and measure the communication delay for the logical groups after clustering. The outcome of the simulation shows that the DBDC can effectively produce clustering results with the communication delay that meets the optimization condition and bring out the best clustering result with the greatest DIR among the others. Such a result can be utilized when constructing logical groups with the optimal communication delay for a massive IoT cluster and as a useful technology that can improve IoT clustering.

The DBDC method introduced in this study is a kind of hybrid technology of designing a massive, large-scale IoT cluster with improved efficiency by combining the DBSCAN algorithm, an unsupervised learning method often applied in the humanities and sociology, and the dynamic clustering method in IoT, one of the major technology in the fourth Industrial Revolution. Such a hybrid technology can provide a basis for a new technological advance to emerge by combining the advantages of different fields. Although several constraints and assumptions limit the technique and results of the experiment in this study, the effectiveness of the DBDC method is proven enough. Hence, further research will apply a meaningful scale of clusters and concentrate on a method that can accurately measure the communication delay between nodes. This research will be applied in modern IoT system design, which will be enlarged daily, maintenance technology development, and management.

References

[1] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, B. Sikdar, "A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures," IEEE Access, vol. 7, pp. 82721–82743, 2019.

[2] M. Asad, D. Aslam, Y. Nianmin, N. Ayoub, E. Munir, "IoT enabled adaptive clustering based energy efficient routing protocol for wireless sensor networks," International Journal of Ad Hoc and Ubiquitous Computing, vol. 32 no. 2, pp. 133–145, 2019.

[3] R. Morabito, I, Farris, A. Iera, T. Taleb, "Evaluating Performance of Containerized IoT Services for Clustered Devices at the Network Edge," IEEE Internet of Things Journal, vol. 4 no. 4, pp. 1019–1030, 2017.

[4] X. Shao, C. Yang, D. Chen, N. Zhao, F. Yu, "Dynamic IoT Device Clustering and Energy Management With Hybrid NOMA Systems," IEEE Transactions on Industrial Informatics, vol. 14 no. 10, pp. 4622–4630, 2018.

[5] A. Mukherjee, P. Goswami, L. Yang, Z. Yan, M. Daneshmand, "Dynamic clustering method based on power demand and information volume for intelligent and green IoT," Computer Communications, vol. 152, pp. 119–125, 2020.

[6] P. Dhas and B. Gomathi, "A novel clustering algorithm by clubbing GHFCM and GWO for microarray gene data." The Journal of Supercomputing, vol. 76 no. 8, pp. 5679–5693, 2020.

[7] W. Gong, L. Pang, J. Wang, M. Xia, and Y. Zhang. "A social-aware k means clustering algorithm for d2d multicast communication under sdn architecture." AEU-International Journal of Electronics and Communications, vol. 132, pp. 153–610, 2021.

[8] F. Murtagh and P. Contreras. "Algorithms for hierarchical clustering: an overview." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 2 no. 1, pp. 86–97, 2012.

[9] S. Raj and D. Ghosh. "Improved and optimal dbscan for embedded applications using high-resolution automotive radar." In 2020 21st International Radar Symposium (IRS), pp. 343–346, 2020.

[10] K. Mardani and K. Maghooli. "Enhancing retinal blood vessel segmentation in medical images using combined segmentation modes extracted by dbscan and morphological reconstruction." Biomedical Signal Processing and Control, vol. 69:102837, 2021.