JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 35, 1075-1097 (2019)
DOI: 10.6688/JISE.201909_35(5).0009

A Cost-Efficient Virtual Sensor Management Scheme for Manufacturing Network in Smart Factory

CONG GAO[1,2], ZHENZHOU TIAN[1,2], YANPING CHEN[1,2] AND ZHONGMIN WANG[1,2]
[1]*School of Computer Science and Technology*
[2]*Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing*
Xi'an University of Posts and Telecommunications
Xi'an, 710121 P.R. China
E-mail: {cgao; zztian; chenyp; zmwang}@xupt.edu.cn

The information comes from physical shop floor and manufacturing process is closely monitored and coordinated under the framework of cyber-physical system in Industry 4.0. Wireless sensor networks are deployed to collect the massive amounts of data generated in a smart factory. Researchers employ sensor cloud to facilitate the management of a large scale deployment of wireless sensor nodes. Facing with lots of concurrent sensing demands of users administrators of a manufacturing network need to deal with the mapping of physical sensors and virtual sensors. We propose a cost-efficient virtual sensor management scheme which is able to accord an overall virtual sensor instantiation result for the whole manufacturing network. Both the architecture of the manufacturing network and the application scenario are modeled by entities, actions, and messages. The key component of the proposed model is called *k* resource scheduler. Different resource scheduling algorithms could be applied to the *k* resource scheduler, and thus make our model flexible. Three resource scheduling algorithms are devised to tackle the problem of virtual sensor management. The effectiveness of the proposed model is verified by simulation experiments and a comprehensive analysis of the experimental results is provided.

Keywords: manufacturing network, cyber-physical system, Industry 4.0, wireless sensor networks, virtual sensor, *k*-median

1. INTRODUCTION

Recent years, the booming development of information communication technology has enlightened an evolution which transforms traditional manufacturing industry to the next generation, namely Industry 4.0. Smart factory is the primary application entity which holds up Industry 4.0. Modern production lines and manufacturing processes contain various data, such as temperature, pressure, displacement, thermal energy, vibration, and noise. Multiple forms of analysis can be conducted based on these data, such as device diagnostics, energy consumption, quality assurance, and automated logistics. A brief comparison between today's factory and an Industry 4.0 factory is presented in [1]. The utilization of information communication technologies such as Internet of things, sensor networks, and cloud computing has significantly accelerated the generation of large amounts of data. The rapid growing amounts of data in smart factory pose a critical challenge to data acquisition, data storage, data processing, data analysis, *etc.* In such circumstances, pioneers in both the academic and industrial fields advocate to bring in the cyber-physical system (CPS). Currently, there is no a unanimous definition of CPS,

Received October 29, 2018; revised December 5, 2018; accepted December 21, 2018.
Communicated by Xiaohong Jiang.

however. An exact definition depends on the underlying ideas and technologies. One of the commonly accepted definition is: the CPS is defined as a collection of innovative technologies for regulating interconnected systems between their physical assets and computing power [2]. Thus, the deployment of CPS enables a comprehensive linkage among mechanical equipment, physical assets, manufacturing process monitor systems, *etc.* These entities are connected with each other by specialized network infrastructures. The information comes from physical shop floor and manufacturing process is closely monitored and coordinated under the framework of CPS. For a manufacturing plant, multitudinous control requirements and operations are the major routines. Generally speaking, the corresponding control mechanisms are distributed among several entities. The cooperation among these mechanisms facilitates a self-organized production system. The high flexibility and reconfigurability of this production system could handle the dynamic nature of production line and manufacturing process in a smart factory.

The application of CPS in Industry 4.0 contains two aspects [3]: wireless sensor network and cloud computing.

Wireless sensor network focuses on delivering collected data with various types of sensors and is currently bridging the gap between the cyber and physical systems [4]. The data obtained by sensors play a crucial role in machine behavior modeling and manufacturing process optimization. In other words, the sensor data possess vital potential information which can be applied to manufacturing status prediction and adjustment, production line optimization, device diagnostics, *etc.* To enable a highly flexible and reconfigurable smart factory and conduct the operations mentioned above, data should be obtained appropriately and efficiently in the first place. In a smart factory, there are various kinds of data requesters such as A/C controller, energy analyzer, LED panel, human operator, *etc.* For simplicity, we refer to them collectively as *users*. Traditional data collection methods use the physical sensors in a direct way. However, this approach has a major drawback: with a large scale deployment of wireless sensor nodes, users are faced with the problem of choosing a functionally matched sensor node with minimum overhead.

To address this problem, researchers introduced cloud computing technologies. Cloud computing provides shared computing resources and data in the light of user demands. Thus, the concept *sensor cloud* [5] sprang up with the combination of wireless sensor network and cloud computing. As the name implies, sensor cloud refers to an infrastructure within which physical sensors are connected to cloud for management. It provides users with cloud service instances in an automated way. The cloud service instance is called *virtual sensor*. A virtual sensor is an emulation of a physical sensor and its data are obtained from underlying physical sensors [6]. The term *virtual* means transparency to users. That is to say, there is no difference between a cloud service instance and other physical resources in the system in terms of user experience. To be specific, virtual sensors provide user with customized views by conducting distribution transparency and location transparency. Before the concept of sensor cloud appears, the real-time communication of cloud computing has been discussed [7, 8]. Then it comes up with extensive studies on the integration of sensor with cloud framework. In [5], a clear picture of the establishment and opportunities of sensor cloud architecture was provided. In [9], a detailed review of sensor cloud was given, including concepts, inherent natures, and application advantages. In addition, a comparison among the message types involved

in different models was also conducted. In [10], the integration of sensor with cloud scheme was applied to health monitoring. An optimal gateway selection model is proposed for the purpose of maximizing bandwidth for health data transmission. In [11], the authors listed the challenges in front of the integration of wireless sensor network and cloud, and proposed a dedicated sensor cloud framework for Software-as-a-Service applications. A similar work was provided in [12], the authors discussed the challenges for comprehension of sensor cloud diversification, implementation of scalable functions, privacy protection, *etc.* Moreover, a baseline for studying the above issues was also given. In [13], a simple virtual wireless sensor network infrastructure was proposed. The scheme is independent of the underlying protocols, and is able to combine with popular routing protocols and data aggregation protocols. In [14], a topology virtualization model was implemented by node self-organization for underwater sensor network.

Most of the existing work discussed the benefits of sensor cloud and concomitant challenges. However, an efficient sensor management scheme dealing with concurrent sensing demands in sensor cloud is missing. Specifically, two key problems need to be investigated: (1) how concurrent sensing requests of a group of users are mapped to the sensor cloud; and (2) which physical sensor is used to instantiate a specific virtual sensor.

In this work, we address the problem of virtual sensor management in a manufacturing network. The basic idea is introducing a resource scheduler and formulating it as the famous facility location problem (FLP). Our proposal concentrates on obtaining an overall cost-efficient virtual sensor instantiation result for concurrent sensing demands. For k simultaneously active virtual sensors in the sensor cloud, we devise three algorithms to pursuit an optimal solution which minimizes the total connection cost of the manufacturing network.

The remainder of the paper is organized as follows: Section 2 reviews the facility location problem and introduces the k-median problem. Section 3 proposes a k resource scheduler model. Three virtual sensor management algorithms are elaborated with detailed theoretical analysis. In Section 4, we evaluate our proposal by simulation and present a comprehensive analysis of the simulation results. Section 5 draws the conclusions and possible future extensions for the proposed method.

2. FACILITY LOCATION PROBLEM

The theory of facility location problem (FLP) was created by Cooper [15] in 1963. As the adoption of FLP is widespread both in scientific area and everyday life, it has been a research hotspot for nearly sixty years [16]. Originally, the facility location problem focuses on the modeling of facility location. It is a branch of operations research and also known as location analysis. Theoretically, the facility location problem refers to determine a series of optimal locations for several facilities that will provide services from a given set of points.

2.1 FLP Overview

Assume there are n potential facility locations and m users in the region of interest. These potential facility locations are denoted by set $FL = \{fl_1, fl_2, \ldots, fl_n\}$. We denote the

opening cost of facility location fl_i by f_i, where $f_i \geq 0$. By *opening cost* of a facility location, we mean a catch-all term which includes the facility construction/renting cost, daily operation cost, and equipment maintenance/upgrading cost. For each location $fl_i \in FL$, we denote the corresponding opening costs by set $F = \{f_1, f_2, ..., f_n\}$. Provided a facility location is open, it is able to offer services to a user. For a single user to be served, one facility is sufficient.

We denote all possible users by set $U = \{u_1, u_2, ..., u_m\}$. The connection cost for a user $u_j \in U$ to use the service provided by a facility at facility location fl_i is denoted by $d(fl_i, u_j)$, where $d(fl_i, u_j) \geq 0$. The facility locations and the users in the region of interest constitute a point set $V = FL \cup U$. We assume that $FL \cap U = \varnothing$. Thus, $V = \{v_1, v_2, ..., v_i\}$, where $1 \leq i \leq n + m$. For $v_i = fl_i \in FL$ and $v_j = u_j \in U$, the total cost for the user u_j to be served by a facility located at facility location fl_i can be calculated as $C(v_i, v_j) = f_i + d(v_i, v_j)$. The solution to a facility location problem is a subset $S \subseteq FL$ whose facilities are able to serve all users in set U. Besides, the subset S minimizes the overall cost

$$C(S) = \sum_{v_i \in S} f_i + \sum_{v_j \in U} d(v_i, v_j). \tag{1}$$

As stated, there only needs one facility for an individual user. We denote the coexisted facilities at facility location fl_i by set $F(fl_i)$. For a facility at facility location fl_i, the concurrent users of the facility are denoted by set $U(fl_i)$. If there are no upper bounds for both the number of elements in sets $F(fl_i)$ and $U(fl_i)$, this specific facility location problem is called Uncapacitated Facility Location Problem (UFLP) [17]. The UFLP was proved to be NP-hard [18]. As the facility locations and the users in the region of interest constitute a point set V, we define a unified metric space Θ for the elements in V. The distance metric in Θ is denoted by d. Consider three arbitrary points $v_i, v_j, v_k \in \Theta$, we have $d(v_i, v_j) \geq 0$, $d(v_i, v_i) = 0$, and $d(v_i, v_j) = d(v_j, v_i)$. A sufficient condition of $v_i = v_j$ is $d(v_i, v_i) = 0$. Furthermore, the inequality $d(v_i, v_j) + d(v_j, v_k) \geq d(v_i, v_k)$ holds.

However, in real applications, restrictions are imposed on a plain application of the UFLP. For example, the number of users concurrently served by a specific facility is restricted due to various limits of resources. In addition, the number of facilities concurrently opened at the same facility location is also limited. Thus, the evaluation of the overall cost requires an analysis of facility location selection and the number of facilities opened at each location. The overall cost consists of opening cost and connection cost. This kind of problem is referred to as Capacitated Facility Location Problem with Hard Capacities (HCFLP) [19]. Generally, there is only one opened facility at each facility location in the real world. That is to say, for facility location fl_i, the number of coexisted facilities $|F(fl_i)| = 1$. This results in the k-median problem [20].

2.2 *K*-median Problem

The k-median problem is the most popular variant of the original FLP. Generally speaking, it is categorized as a method for clustering data [21]. It differs from the UFLP that a parameter k is introduced, where $0 < k < |FL|$. In the realm of clustering, if the number of clusters is larger than k, a solution to the k-median problem keeps merging the existing clusters until there are exactly k clusters. Besides the additional parameter k, there are three major differences between the k-median problem and the original UFLP:

- Only one facility can be opened at each facility location.
- The number of open facilities is no more than k.
- There is no opening cost for facility locations.

To solve the k-median problem, we should obtain a subset $S \subseteq FL$, where $|S| \leq k$ such that all users are served by facilities at locations in the set S. The obtained set S minimizes the overall cost. Since there is no opening cost for facility locations in the k-median problem, the overall cost is

$$C(S) = \sum_{v_j \in U} d(v_i, v_j). \tag{2}$$

Now that the number of coexisted facilities at facility location fl_i is $|F(fl_i)| = 1$, a facility location can be considered as equivalent to a facility. In the reminder of of this section, we use the facility location set FL to denote the facilities interchangeably. We confine the number of concurrent users of a facility in the range of [0, m], namely $0 \leq |U(fl_i)| \leq m$, where $m = |U|$. The ideal solution of the k-median problem is that each user is served by the facility whose connection cost is minimum. For $v_i = fl_i \in FL$ and $v_j = u_j \in U$, the ideal solution of the k-median problem can be formulated as

$$\forall v_j \in V, \hat{d}(v_i, v_j) = \min_{\forall v_k \in FL} \{d(v_k, v_j)\}. \tag{3}$$

By Eq. (3), each $\hat{d}(v_i, v_j)$ is minimized. Thus, the overall cost $C(S)$ in Eq. (2) is also minimized.

3. A COST-EFFICIENT VIRTUAL SENSOR MANAGEMENT SCHEME

The vertical integration of manufacturing network in a smart factory enables a flexible and reconfigurable manufacturing system which is composed of several hierarchical subsystems. As the number and scale of subsystems keep growing, the numbers of physical sensors and users in a smart factory have increased exponentially. Thus, the CPS administrators in a smart factory are confronted with a problem of how to reach an overall virtual sensor instantiation result for the whole manufacturing network. In this section, we propose a k resource scheduler to address the issue of virtual sensor management in a manufacturing network.

3.1 *k* Resource Scheduler

The application scenario of our proposal is illustrated in Fig. 1. Specifically, Fig. 1 depicts a layered architecture of the manufacturing network, which is divided into four layers.

(1) Cyber-physical subsystem layer. A smart factory possesses several cyber-physical subsystems, such as cooperative planning system, mechanical driving system, signal sen-

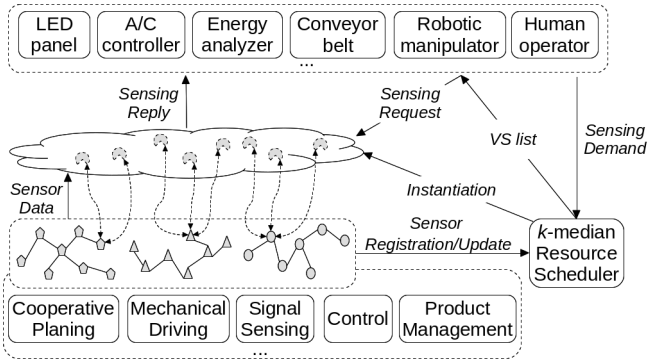

Fig. 1. Manufacturing network.

sing system, control system, and production management system. Generally, a cooperative planning system possesses three key features: the timeliness of information feedback, the coexistence of synchronous operation and asynchronous operation, and the reliability of information transfer. The role of a mechanical driving system is the delivery of motion and force. Common driving types are toothed wheel, worm wheel, chain drive, and gear train. A signal sensing system contains several kinds of sensors. The underlying cooperation among different components with a device relies on signal sensing. A control system consists of control subject, control object, and control medium. It is a management mechanism which has clearly defined goals. A production management system dedicates to establish a real-time production database. The database is efficiently managed in order to monitor the manufacturing processes. Conventionally, a system within this layer is independent. In other words, this layer represents the physical world.

(2) Wireless sensor network layer. This layer contains various wireless sensor networks. For different cyber-physical subsystems in a smart factory, there usually exist several heterogeneous wireless sensor networks. The sensors of these networks aim at intuitive physical phenomena such as temperature, pressure, displacement, thermal energy, vibration, and noise. Due to lack of standardized data exchange formats among heterogeneous wireless sensor networks. Popular ways to handle this problem include introducing a certain description model, a middle middleware, and an upper layer communication protocol, *etc.* For instance, the extensible markup language (XML) is suitable to establish a framework which provides a unified data exchange format to facilitate the communication among heterogeneous wireless sensor networks.

(3) Sensor cloud layer. This layer deals with the virtualization of physical sensors and other resources in the cloud. Similar to common cloud systems, the virtualization provides users with cloud-based sensing services. The overall virtual sensor instantiation result is given by the k resource scheduler. The instantiated virtual sensors are able to process sensing requests and send sensing replies. In general, a virtual sensor is able to simultaneously handle several sensing requests from multiple users. Sensor data collected by the physical sensors in the wireless sensor networks are transmitted to the corresponding virtual sensors in the sensor cloud. Users in the manufacturing network issue sensing requests to the sensor cloud. Then, virtual sensors in the sensor cloud contact

specific physical sensors in the wireless sensor networks to collect potential data. At last, the sensor cloud sends sensing replies to users based on the obtained sensor data. In [6], the authors implemented virtual sensors in four different configurations: one-to-many, many-to-one, many-to-many, and derived configurations. For simplicity, the relation between virtual sensors and physical sensors illustrated in Fig. 1 is *one-to-many*. Although our work is presented in this setting, the key idea of the proposed scheme applies to the other three configurations as well.

(4) User layer. This layer contains all potential entities which need data. To a great degree, the operation of a smart factory and the detailed production status are presented by data contained in the production lines and manufacturing processes. To conduct data visualization and data analysis, relevant data should be obtained in the first place. The term *user* is a catch-all term which encloses system modules and human beings. Common users in the manufacturing network are LED panels, A/C controllers, energy analyzers, conveyor belts, robotic manipulators, and human operators. When a user needs some data, it issues a sensing demand to the *k* resource scheduler. For the concurrent sensing demands in the manufacturing network, the *k* resource scheduler generates an overall virtual sensor instantiation result. Then, the users send sensing requests to the sensor cloud based on the virtual sensor instantiation result.

To facilitate the presentation of our proposal, we make the following formal definitions. A manufacturing network (MN) is composed of eight types of entities: CPS administrator (CA), *k* resource scheduler (RS), manufacturing system (MS), physical sensor (PS), wireless sensor network (WSN), virtual sensor (VS), sensor cloud (SC), and user (U). The definitions of the eight entities are shown in Table 1.

Table 1. Entity definitions.

Entity	Components
CA	⟨*ca_id*⟩ ⟨*algorithm*⟩
RS	⟨*rs_id*⟩ ⟨CA⟩ ⟨*sensor_db*⟩ ⟨*k*⟩
MS	⟨*ms_id*⟩
PS	⟨*ps_id*⟩ ⟨*type*⟩ ⟨VS…⟩ ⟨max_N_v⟩ ⟨cur_N_v⟩
WSN	⟨*wsn_id*⟩ ⟨*prot*⟩ ⟨PS⟩ {PS…}
VS	⟨*vs_id*⟩ ⟨PS⟩ ⟨*s_cost*⟩ ⟨max_N_u⟩ ⟨cur_N_u⟩
SC	⟨*sc_id*⟩ ⟨VS⟩ {VS …}
U	⟨*u_id*⟩ ⟨*u_cost*⟩ {VS…}

There are two kinds of brackets used in the definitions of entities: angle brackets (⟨⟩) and braces ({}). The variables and values within angle brackets are integral parts of a valid definition whereas a pair of braces represents a set. Even if the set is empty, the definition is still valid. For the sake of simplicity, the MS just contains one property for identification, namely *ms_id*. Though a physical sensor possesses multiple attributes, we in this paper use five properties to portray it. The first one is an identification property *ps_id*. The second property *type* indicates the physical phenomenon a physical sensor targeted at. As stated, a physical sensor may be shared by several virtual sensors. The third property of a physical sensor is a set of virtual sensors. If a physical sensor is not

associated with any virtual sensor, the third property is an empty set. The fourth and the fifth properties are the maximum number and the current number of virtual sensors associated with it, respectively.

A wireless sensor network is composed of multiple physical sensors, thus the WSN contains one PS at least. The second property of the WSN indicates the communication protocol used within the wireless sensor network. Similar to a wireless sensor network, a sensor cloud contains one virtual sensor at least. Each virtual sensor is instantiated with a specific physical sensor. Thus, the VS contains an identification *vs id* and a PS. The third property of the VS is the connection cost for the VS to communicate with its corresponding PS. The fourth and the fifth properties are the maximum number and the current number of users associated with it, respectively.

The RS contains the CPS administrator and a database of all physical sensors available in a manufacturing network. For ease of an efficient operation and administration, there needs an upper bound for the number of virtual sensors simultaneously active in the sensor cloud. This upper bound is denoted by the fourth property k of the RS. The second property of the CPS administrator CA is *algorithm*. The role of this algorithm is to obtain an overall virtual sensor instantiation result for the whole manufacturing network. This operation is conducted by the RS. For each user U, the second property u cost denotes the connection cost for the U to communicate with the border gateway of the SC. The third property of the U is the set of virtual sensors assigned to the user.

As shown in Fig. 1, the interactions among the four layers are represented by black line arrows. There are two kinds of connections: a dotted line with bidirectional arrows and a solid line with unidirectional arrow. The former connection is merely appeared between the wireless sensor network layer and the sensor cloud layer. As stated before, the relation between virtual sensor and physical sensor is *one-to-many*. Namely, one physical sensor may be shared by several virtual sensors. The corresponding relation between physical sensors and virtual sensors is determined by the k resource scheduler. Unlike the former connection, there are totally seven cases of the latter connection. They are sensor data, sensor registration/update, sensing demands, instantiation, VS list, sensing request, and sensing reply. We refer to the above seven connection as *actions*. As shown in Table 2, an action has three attributes: *sender*, *receiver*, and *type*. Moreover, all actions are unidirectional, which contributes to a simple and clear design. Sensing demands are sent to the RS then the RS gives an instantiation result to the SC and responses to the users with a VS list. A sensor registration/update is a request sent to the RS, which means it is the WSN who maintains the physical sensors database proactively. The RS just updates the database upon being requested and reads the database when it works on an instantiation result. Namely, the management of physical sensors is handled by the WSN, and the RS just uses the records of physical sensors and concentrates on formulating the instantiation result. Upon receiving the corresponding VS list of the instantiation result, each U makes a sensing request towards the virtual sensor assigned to it. Then, the relevant virtual sensors in the SC communicate with the corresponding physical sensors. The sensor data collected by physical sensors are sent to the SC. At last, the SC sends sensing replies containing the sensor data to the U.

On the whole, when a user has a potential need of data, it generates a sensing demand and then sends it to the resource scheduler. For a manufacturing network which contains considerable number of users, it is probably that there turn up massive sensing

Table 2. Actions.

Action	Sender	Receiver	Type
Sensor Data	WSN	SC	Result
Sensor Registration/Update	WSN	RS	Request
Sensing Demands	U	RS	Request
Instantiation	RS	SC	Result
VS List	RS	U	Result
Sensing Request	U	SC	Request
Sensing Reply	SC	U	Result

demands concurrently. The resource scheduler is designed to formulate an overall virtual sensor instantiation result which meets all sensing demands based on the current information of available physical sensors. The overall virtual sensor instantiation result is obtained according to the specific algorithm stipulated by the CPS administrator. The resource scheduler sends the information contained in the VS list point to point in response to each sensing demand. Once a user receives the information of a virtual sensor, it may issue a sensing request to the specific virtual sensor in the sensor cloud. For the communication between a user and a virtual sensor, there is a connection cost *u cost*. Similarly, for the communication between a virtual sensor and a physical sensor, there is a connection cost *s_cost*. In practice, the CPS administrator does not only take the total connection cost into account, but also consider the number of virtual sensors simultaneously active in the sensor cloud. For ease of an efficient operation and administration, we denote the upper bound of this number by k.

The interactions described above are conducted with eight types of messages: sensor registration, sensor update, sensing demand, instantiation, VS result, sensing request, sensor data, and sensing reply. The definitions of the eight messages are shown in Table 3.

Table 3. Message definitions.

Message	Components
Sensor Registration	⟨*wsn_id*⟩ ⟨*ps_id*⟩ ⟨*type*⟩ ⟨*max_*N_v⟩
Sensor Update	⟨*modify*⟩ ⟨*ps_id*⟩ ⟨*cur-*N_v⟩ | ⟨*del*⟩ ⟨*ps_id*⟩
Sensing Demand	⟨*rs_id*⟩ ⟨*u_id*⟩ ⟨*type*⟩
Instantiation	⟨*vs_id*⟩ ⟨*ps_id*⟩
VS Result	⟨*u_id*⟩ ⟨*sc_id*⟩ ⟨*vs_id*⟩ ⟨*rs_id*⟩
Sensing Request	⟨*sc_id*⟩ ⟨*vs_id*⟩ ⟨*u_id*⟩
Sensor Data	⟨*wsn_id*⟩ ⟨*ps_id*⟩ ⟨*s_data*⟩ ⟨*vs_id*⟩
Sensing Reply	⟨*sc_id*⟩ ⟨*vs_id*⟩ ⟨*data*⟩ ⟨*u_id*⟩

The basic information of a physical sensor is registered with the resource scheduler by sensor registration message. Since the operation of registration is conducted by a new physical sensor, the current number of virtual sensors associated with it is zero. Thus, the property *cur_*N_v is omitted. The sensor update message is sent to the resource scheduler by a wireless sensor network. It has two forms: *modify* and *del*. The former is used to update the current number of virtual sensors associated with a physical sensor. While the latter is used to remove the entry of a physical sensor from the database of the resource

scheduler. A sensing demand message is sent to the resource scheduler by a user. For each sensing demand, the resource scheduler sends a VS result message to the corresponding user according to an overall virtual sensor instantiation result. Upon receiving a VS result message, a user may send a sensing request message to the sensor cloud. The virtual sensor specified in the sensing request message communicates with the associated physical sensor to collect sensor data. Both the sensing request message and the sensing reply message contain *sc_id*, *vs_id*, and *u_id* to indicate source and destination.

For a manufacturing network which has m users, we assume that the sensor cloud in the manufacturing network is able to accommodate a maximum of n virtual sensors. We denote the users and the virtual sensors by sets $U = \{u_1, u_2, ..., u_m\}$ and $VS = \{vs_1, vs_2, ..., vs_n\}$, respectively.

Suppose the number of concurrent sensing demands sent to the resource scheduler is $m' \le m$. The maximum number of virtual sensors active in the sensor cloud is k, where $k \le n$. We make the assumption that there are enough physical sensors deployed in wireless sensor networks such that arbitrary sensing demands can be served. For the m' concurrent sensing demands, we denote a solution to the virtual sensor instantiation problem by a set $R = \{r_1, r_2, ..., r_{m'}\}$. Thus, the total connection cost for the manufacturing network is

$$C(R) = \sum_{i=1}^{m'} (r_i.\mathrm{VS}.s\_\cos t + r.\mathrm{U}.u\_cost). \tag{4}$$

Note that in Eq. (4), "VS" and "U" merely denote the virtual sensor and the user of r_i, respectively. They are not set symbols. Though there are m' terms in Eq. (4), the number of virtual sensors involved in R is no more than k. These involved virtual sensors constitute a set V, where $|V| \le k$. We make the assumption that only one virtual sensor is needed to meet a sensing demand. Thus, we have $|V| \le m'$. Our goal is to minimize the total connection cost $C(R)$. Based on the above discussion, it can be formulated as a k-median problem.

In our application scenario, the actual interactions among the four layers within the manufacturing network are of high volume. During the operation of the manufacturing network, a lot of virtual sensor instantiation records which list pairs of a sensing request and a physical sensor are readily available. Thus, to concentrate on the elaboration of our model, we make a premise that there exists a function $FS(VS)$ which is able to give a feasible solution for m' concurrent sensing demands based on the available virtual sensors. The solution includes two parts: V and R. The former one is a set of involved virtual sensors and contains exactly k elements; the latter one is a set of records and contains exactly m' elements, each record resolves one sensing demands. In addition, we denote the concurrent sensing demands by set D, where $|D| = m'$.

3.2 Progressive Swap Algorithm

The progressive swap algorithm mainly consists of three steps.

(1) A feasible solution is generated by $FS(VS)$, the records which resolve the m' sensing demands constitute a set R, where $|R| = m'$. The virtual sensors involved in set R constitute a set V, where $V \subset VS$ and $|V| = k$.

(2) To get a reduction of $C(R)$, the elements of set V are modified for the purpose of changing the elements of set R. We introduce a plain swap operation $s(V)$ to modify set V. The operation $s(V)$ is conducted between elements in V and $VS \setminus V$. The elements of V is progressively modified in order to reduce $C(R)$.

(3) As the number of swaps increases, $C(R)$ is progressively reduced. When there is no $s(V)$ which is able to get $C(R)$ reduced, the algorithm terminates. At this point, $C(R)$ is minimized. However, in the real world, a thorough execution which leads to a minimized result is always not permissible or practical due to various limitations. Thus, to control the number of swap operations, we introduce a variable *cap* as the upper bound which terminates the execution of the algorithm. In most cases, an execution terminated by *cap* does not lead to a minimized result, for there still exists one or more $s(V)$ which can reduce $C(R)$. In this case, we refer to the set R as a partial optimal solution. The progressive swap algorithm (PSA) is detailed in Algorithm 1.

Algorithm 1: Progressive Swap Algorithm

Require: VS, D, k.
1: $V, R \leftarrow FS(VS)$
2: $s := s(V) = V \setminus \{vs_i\} \cup \{vs'_i\}$
3: $V' := V + s(V)$
4: $R' := R + V'$
5: $c \leftarrow 0$
6: **repeat**
7: **if** $\exists s$ allows $C(R') < C(R)$ **then**
8: $V \leftarrow V'$
9: $R \leftarrow R$
10: **end if**
11: $c \leftarrow c + 1$
12: **until** (no s can reduce $C(R)$) $|(c == cap)$
13: **return** V, R

Let R^* be a partial optimal solution. For a feasible solution R, the involved virtual sensors are denoted by set V. We make the assumption that there exists a swap $s(V)$ which allows

$$C(R') \le \left(1 - \frac{\varepsilon}{P(n, m')}\right) \cdot C(R) \tag{5}$$

holds, where $\varepsilon > 0$ and $p(n, m')$ is a polynomial of n and m'. The number of swap operations is at most

$$\log(C(R)/C(R^*))/\log\frac{1}{1-\frac{\varepsilon}{p(n,m')}}. \tag{6}$$

Since the input size of $\log(C(R))$ is polynomial and the time complexity of operation s is also polynomial, the time complexity of Algorithm 1 is polynomial [22].

For any feasible solution R, the corresponding set of involved virtual sensors V possesses two properties: $V \subset VS$ and $|V| = k$. The minimization process of $C(R)$ is conducted by swapping elements between set V and set $VS \setminus V$. A small k indicates more alternative virtual sensors in set $VS \setminus V$. While a large k leads to less room for the swap operation. Thus, a variation of a small k is more influential to the total connection cost (or the average connection cost) of the manufacturing network than that of a large k.

3.3 Greedy Algorithm

In most cases, an algorithm called "greedy" indicates that the expected return keeps being maximized during each step of execution. For our application scenario, since the sensor cloud in the manufacturing network is able to accommodate a maximum of n virtual sensors, the initial state of a greedy strategy takes n virtual sensors into account. On this premise, we derive the corresponding initial solution. In the first place, the number of current users of each virtual sensor in set VS is set to zero. We denote the current users of a virtual sensors vs_i by set $U(vs_i)$. Initially, $|U(vs_i)| = 0$, where $1 \le i \le n$. The total connection between virtual sensor vs_i and user u_j is

$$d(vs_i, u_j) = vs_i.s\_cost + u_j.u\_cost. \tag{7}$$

Since n virtual sensors are considered initially, for each $j \in [1, m']$, $d(vs_i, u_j)$ can be minimized. This is the ideal solution. Then, for each $j \in [1, m']$, we add user u_j to the corresponding set of current users $U(vs_i)$. At this point, we denote the number of elements in set V by n'. As the number of involved virtual sensors should be no more than k, a value of n' which is larger than k is invalid. When this happens, the virtual sensors whose current users are few should be removed from set V. A user formerly served by a removed virtual sensor is reassigned to another virtual sensor in set V. When there are exactly k virtual sensors in set V, the algorithm terminates and the set R is a partial optimal solution. The greedy algorithm (GA) is illustrated in Algorithm 2.

In Algorithm 2, there are three parts of calculation. They possess different time complexities; (1) The number of current users of n virtual sensors: $O(\max(n, m'))$; (2) The searching of the virtual sensor whose current users is the fewest: $O(n)$; (3) The reassignment of orphan users: $O(n \times m')$. So the time complexity of Algorithm 2 is $O(n^2 \times m')$.

The greedy algorithm initially considers n virtual sensors. When the ideal solution is derived, there are n' virtual sensors involved, where $|V| = n'$ and $n' \le n$. However, a necessary condition of a valid solution is $n' \le k$. For $n' > k$, the virtual sensors whose current users are few should be removed from set V. A user formerly served by a removed virtual sensor is reassigned to another virtual sensor in set V. This process repeats until $n' = k$, namely there are k elements in set V. For a small k, the number of virtual sensors in set V is also small at the ending part of the execution of the algorithm. It is probably that a reassignment of a user merely leads to a slight difference between its former connection cost and the new one, since all virtual sensors in set V could be considered far away from the user. When k is large, there are more alternative virtual sensors. For a reassignment of a user, the former virtual sensor is considered near to the user, while the new virtual sensor could be considered far from the user. Thus, a variation of a large k is more influential to the total connection cost (or the average connection cost) of the manufacturing network than that of a small k.

Algorithm 2: Greedy Algorithm

Require: VS, D, k.
1: **for** $i = 1$ **to** $|VS|$ **do**
2: $U(vs_i) \leftarrow \varnothing$
3: **end for**
4: **for** $i = 1$ **to** $|VS|$ **do**
5: **for** $j = 1$ **to** $|D|$ **do**
6: $d_{ij} \leftarrow d(vs_i, u_j)$
7: **end for**
8: **end for**
9: **for** $j = 1$ **to** $|D|$ **do**
10: $d_{kj} = \min_{1 \le i \le |V|} \{d_{ij}\}$
11: $U(vs_k) \leftarrow U(vs_k) \cup \{u_j\}$
12: **end for**
13: **repeat**
14: $vs_r = \left\{ vs_i \mid \min_{1 \le i \le |V|} U(vs_i) \right\}$
15: **for** $j = 1$ **to** $vs_r.cur\_N_u$ **do**
16: $dk_{kj} = \min\{\{d_{1j}, d_{2j}, \ldots, d_{|V|j}\} \setminus \{d_{rj}\}\}$
17: $U(vs_k) \leftarrow U(vs_k) \cup \{u_j\}$
18: **end for**
19: $V \leftarrow V | \{vs_r\}$
20: **until** $|V| == k$
21: **return** V, R

3.4 RK Algorithm

The RK algorithm is a hybrid of the progressive swap algorithm and the greedy algorithm. To some degree, the progressive swap algorithm and the greedy algorithm show two opposite strategies, respectively. The former one initially considers k virtual sensors which meet the m' sensing demands then adjusts the solution by swapping virtual sensors. The status it starts at is random then the algorithm proceeds with optimization. The initial solution merely contains k virtual sensors. Thus, the benefit to cost ratio of the subsequent swapping process is low. While, the latter one initially considers all virtual sensors available, then adjust the solution by deleting virtual sensors. The status it starts at is the ideal case, then the algorithm proceeds with degradation. The initial solution contains n' virtual sensors. In most cases, $n' \gg k$ holds. Hence, the subsequent degradation process is expatiatory.

In view of the above, we develop the RK algorithm which combines the progressive swap algorithm and the greedy algorithm. The RK algorithm consists of two parts. In the first part (lines: 1-7), $r \cdot k$ virtual sensors are eventually obtained which meet the m' sensing demands. The parameter r is a coefficient for the candidate virtual sensors, where $r = 1, 2, 3, \ldots, \left\lfloor \frac{|VS|}{k} \right\rfloor$. Specifically, a feasible solution is generated by $FS(VS)$ in the first place, and the virtual sensors involved constitute set V. Then, another feasible solution is generated by $FS(VS \setminus V)$, and the virtual sensors involved constitute set V^n. We let the virtual sensors contained in set V and set V^n constitute set V. This process continues until $|V| =$

$r \cdot k$. In the second part (line 8), the greedy algorithm is executed. At this point, the virtual sensors initially considered are denoted by set V. The detailed RK algorithm is showed in Algorithm 3. The parameter r indicates the number of feasible solutions generated. In the first part of the RK algorithm, several feasible solutions are successively generated. For instance, when $r = 2$, two feasible solutions are successively generated. After the first feasbile solution is generated, the virtual sensors involved are not considered in the generation of the second feasible solution. This exclusion is carried out by the set operation $VS \setminus V$. Thus, the value of r is directly proportional to the number of elements in set V. The operation of the first part ends up with $|V| = r \cdot k$. The second part is the very greedy algorithm. However, the set of virtual sensors initially considered is V, instead of VS. By the greedy algorithm, we know that as long as set VS is determined, the final solution (V and R) is also determined. Generally speaking, the greater the number of virtual sensors in set VS is, the better the performance of the greedy algorithm is. However, the more the number of virtual sensors initially considered, the more time consuming the greedy algorithm is. Thus, the first part of the RK algorithm introduces a parameter r to control the number of virtual sensors passed to the second part.

Algorithm 3: RK Algorithm

Require: VS, D, k, r.
1: $V \leftarrow \varnothing$, $V^t \leftarrow \varnothing$
2: $V \leftarrow FS(VS)$
3: **for** $i = 1$ **to** $r - 1$ **do**
4: $\quad V^t \leftarrow FS(VS \setminus V)$
5: $\quad V \leftarrow V \cup V^t$
6: $\quad V^t \leftarrow \varnothing$
7: **end for**
8: **exec** GA with V, D, k.

4. SIMULATION RESULTS

We conduct a set of simulations to evaluate the performance of our proposal. In these simulations, we show (1) the relation between the performance and the number of iteration for the PSA and the GA; (2) the ranges of the number of iteration for the solutions and the corresponding values for optimal solutions for the PSA and the GA; and (3) the variation of the average connection cost for different numbers of virtual sensors simultaneously active and different numbers of concurrent sensing demands for the PSA, the GA, and the RKA. The simulations are conducted on a 2.50 GHz Intel Core i5 processor and 8GB RAM under Debian Stretch 9.4.0 [23]. The simulation platform is developed based on the NS-3 [24] simulator framework. The entities and the associated actions are implemented in the light of Section 3.

4.1 Simulation Settings

For simplicity, we consider an MN with one WSN and one SC. We employ three types of physical sensors: temperature sensors (TPS), humidity sensors (HPS) and pres-

sure sensors (PPS). Each type of sensor consists of 100 nodes. The SC is able to accommodate a maximum of $n = 150$ virtual sensors and the number of users is $m = 600$. The physical sensors are randomly distributed in an 80×80 region, while the users are not restricted in this area. The values of *u_cost* and *s_cost* are uniformly distributed in the range of (0, 80] and (0, 600], respectively.

Due to limited resources and real-time requirements, the chances for a complete execution of an algorithm which achieves an optimal solution is slim to none. Thus, we introduce a parameter *iter* to indicate the number of iterations an algorithm is executed. For the progressive swap algorithm, as long as the value of *cap* is large enough, the total connection cost $C(R)$ will eventually be minimized. In other words, if the value of *iter* is large enough, the progressive swap algorithm will obtain an optimal solution. For the greedy algorithm, when $n' > k$, there are $n' - k$ virtual sensors to be removed from set V before the minimized $C(R)$ is obtained. When *iter* is greater than or equal to $n' - k$, the greedy algorithm will accord an optimal solution. In Table 4, we identified five key parameters for our simulation: k, m', *cap*, $n' - k$, and r. k is the number of virtual sensors simultaneously active in the sensor cloud. m' is the number of concurrent sensing demands sent to the resource scheduler. *cap* is the upper bound which terminates the execution of the progressive swap algorithm. $n' - k$ is the number of virtual sensors which should be removed from set V.

Table 4. Key parameters.

Parameter	Explanation
k	# of VS simultaneously active in the SC
m'	# of concurrent sensing demands sent to the RS
cap	the upper bound of the *iter* for the PSA
$n' - k$	# of VS which should be removed from set V
r	# of feasible solutions generated in the RKA

4.2 Simulation Results and Analysis

To investigate the performance of our proposal, we conducted extensive simulations. There are three cases of the relation between *cap* and $n' - k$: $cap < n' - k$, $cap = n' - k$, and $cap > n' - k$. The corresponding simulation results are depicted in Figs. 2-4. As shown in Figs. 2-4, the total connection cost of the progressive swap algorithm monotonically decreases with the increase of *iter*, while the total connection cost of the greedy algorithm monotonically increases with the increase of *iter*. The performance of the progressive swap algorithm is bounded within the curves PSA_w and PSA_b. And specifically, the curve PSA_w stands for the worst case, while the curve PSA_b indicates the best case. The curve *GA* is a representative which shows the performance of the greedy algorithm. At the beginning of the greedy algorithm (*i.e.*, $iter = 0$), the ideal solution is achieved and there are n' virtual sensors involved. The total connection cost increases with the increase of *iter*. When the greedy algorithm terminates, an optimal solution is obtained. We denote the corresponding coordinates for the curve *GA* by $(iter_g, c_g)$. The termination of a progressive swap algorithm also accords an optimal solution. We denote the corresponding coordinates for the curves PSA_w and PSA_b by $(iter_w, c_w)$ and $(iter_b, c_b)$, respec-

tively. Trivially, $iter \in [0, +\infty]$. However, there is a range of *iter* within which an algorithm is able to produce feasible solutions. Table 5 lists the ranges of *iter* for the solutions and the corresponding values for the optimal solutions.

Table 5. Solutions and optimal solutions.

	$cap < n' - k$		$cap = n' - k$		$cap > n' - k$	
	Solution	Optimal	Solution	Optimal	Solution	Optimal
PSA	$iter \geq 0$	$iter = 40$	$iter \geq 0$	$iter = 50$	$iter \geq 0$	$iter = 50$
GA	$iter \geq 50$	$iter = 50$	$iter \geq 50$	$iter = 50$	$iter \geq 40$	$iter = 40$

The case of $cap < n' - k$ is depicted in Fig. 2, where *cap* and $n' - k$ are 40 and 50, respectively. When $iter = n' - k = 50$, the optimal solution of the greedy algorithm is obtained, namely $iter_g = 50$. Similarly, as $cap = 40$, the curves PSA_w and PSA_b end at $iter = 40$. Thus, $iter_w = iter_b = 40$. For $iter \geq 0$, the progressive swap algorithm is able to give a solution; while the greedy algorithm is able to give a solution when $iter \geq 50$.

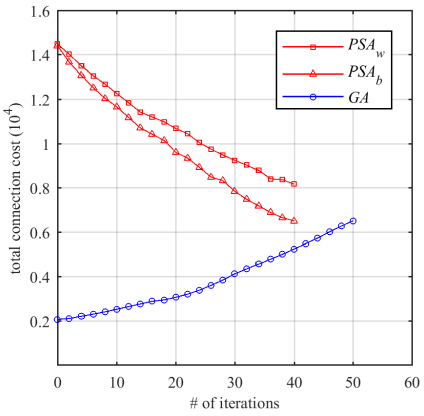

Fig. 2. $cap < n' - k$.

The case of $cap = n' - k$ is depicted in Fig. 3, where both *cap* and $n' - k$ are 50. When $iter = 50$, the optimal solution of the progressive swap algorithm is obtained, namely $iter_w = iter_b = 50$. Analogously, $iter_g = 50$. For $iter \geq 0$, the progressive swap algorithm is able to give a solution; while the greedy algorithm is able to give a solution when $iter \geq 50$.

The case of $cap > n' - k$ is depicted in Fig. 4, where *cap* and $n' - k$ are 50 and 40, respectively. When $iter = n' - k = 40$, the optimal solution of the greedy algorithm is obtained, namely $iter_g = 40$. Similarly, as $cap = 50$, the curves PSA_w and PSA_b end at $iter = 50$. Thus, $iter_w = iter_b = 50$. For $iter \geq 0$, the progressive swap algorithm is able to give a solution; while the greedy algorithm is able to give a solution when $iter \geq 40$.

For the progressive swap algorithm, if the number of swap operations is large enough, namely the variable *cap* is large enough, the execution of the progressive swap algorithm will terminate with an optimal solution. For the greedy algorithm, $n' - k$ virtu-

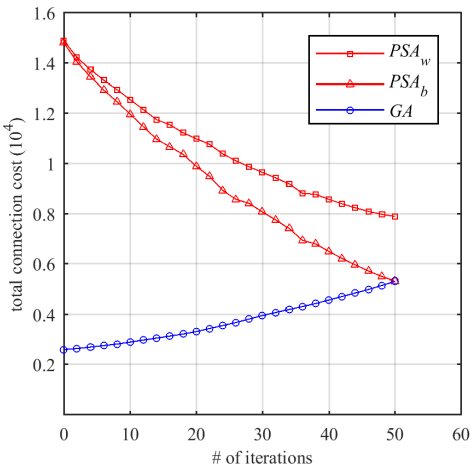

Fig. 3. $cap = n' - k$.

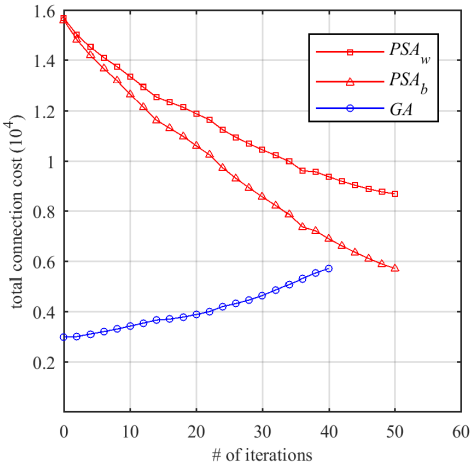

Fig. 4. $cap > n' - k$.

al sensors should be removed from set V before the optimal solution is obtained. As shown in Fig. 3, when the greedy algorithm obtains the optimal solution, we have $c_g = c_b < c_w$. In other words, the performance of the greedy algorithm is better than the progressive swap algorithm when $cap = n' - k$. For the purpose of investigating the parameters k and m', we set $cap = n' - k$. All things being equal, the total connection cost for the MN increases with the number of concurrent sensing demands sent to the resource scheduler. Thus, we concentrate on the average connection cost for m' concurrent sensing demands and k virtual sensors simultaneously active in the sensor cloud. For $m' = 200$ and 400, we consider a confined range of k as $k \in [50, 100]$. For $k = 30$ and 60, we consider a confined range of m' as $m' \in [100, 500]$. The simulation results for k and m' are depicted in Figs. 5 and 6, respectively.

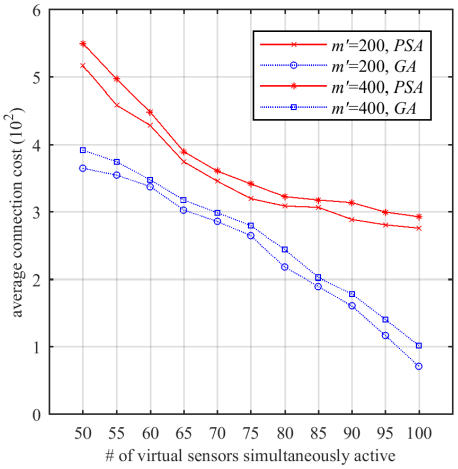

Fig. 5. Variation of cost with different k for PSA and GA.

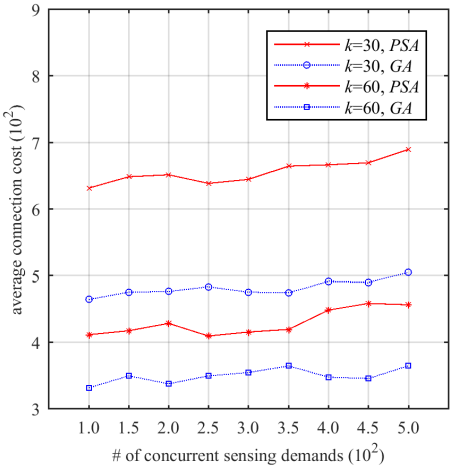

Fig. 6. Variation of cost with different m' for PSA and GA.

As shown in Fig. 5, the average connection costs of both algorithms monotonically decrease with the increase of k. The overall performance of the *GA* is superior to the *PSA*. For the *PSA*, the variation of the average connection cost around a small k is more significant than that of a large k, and the inflection point is around $k = 75$. In other words, the curves of the *PSA* are concave. For the *GA*, there also exists an inflection point which is around $k = 75$. However, the curves of the *GA* are convex. In other words, the variation of the average connection cost around a large k is more significant than that of a small k.

As shown in Fig. 6, the fluctuation of the average connection costs of both algorithms are slight with the variation of m'. Namely, the average connection cost is relatively steady. Though the average connection cost as a whole shows an increasing trend, the variation of the average connection cost is not monotonical. This indicates that the characteristics of the relation between m' and the the average connection cost is not as obvious as that of k. For simplicity, we consider two cases of $k = 30$ and 60. With the increase of k, there are more virtual sensors which could be used in the sensor cloud. Namely, there are more room for the solving process and the final solution of both algorithms. Thus, when $k = 60$, the overall performances of both algorithms are better than $k = 30$.

To facilitate the operation and management of the MN, the number of virtual sensors simultaneously active in the sensor cloud should be as small as possible. The above two algorithms pursue a minimization of the total connection cost. Namely, both k and $C(R)$ should be as small as possible. Thus, it is necessary to investigate the relation between k and $C(R)$. As shown in Fig. 5, the average connection cost monotonically decreases with the increase of k. Thus, the total connection cost also monotonically decreases with the increase of k. If the CPS administrator shows more concern about the total connection cost, a larger k (*e.g.*, $k \in [75, 100]$) is preferred. On the contrary, if the CPS administrator pays more attention to the number of virtual sensors simultaneously active in the sensor cloud, a smaller k (*e.g.*, $k \in [50, 75]$) is preferred.

To make an analysis of the Algorithm 3, we set $m' = 300$ and observe how the coefficient for the candidate virtual sensors affects the average connection cost of the whole manufacturing network. The setting of $m' = 300$ is in concert with the experimental results depicted in Fig. 5. This setting facilitates the performance evaluation of the RKA in comparison with the PSA and the GA. As shown in Fig. 7, the average connection cost for the RKA monotonically decreases with the increase of the coefficient for the candidate virtual sensors. In other words, the average connection cost monotonically decreases with the increase of the number of candidate virtual sensors. This characteristic coincides with the PSA and the GA. In addition, the greater the coefficient for the candidate virtual sensors, the greater the rate of decreasing.

When $r = 1$, the first part of the RKA ends up with a set V which contains exactly k virtual sensors. Thus, the second part does nothing. In this case, the final solution is very set V which is obtained by $FS(VS)$ in the first part. There is no further modifications imposed on the original feasible solution generated by $FS(VS)$. Thus, the average connection cost is quite large($C(R) = 850$). By the experimental results showed in Fig. 6, when $k = 30$, the average connection costs for the PSA and the GA are 645 and 476, respectively. By the experimental results showed in Fig. 6, the performance of the GA is superior to the PSA for a fixed k. We use a red dashed line and a blue dashed line to delimit the performance of the PSA and the GA when $k = 30$. The number of virtual sensors consid-

ered by the second part of the RKA is growing in direct proportion to the coefficient for the candidate virtual sensors. For k = 1, 2, 3, 4, 5, 6, the performance of the PSA is superior to that of the RKA. For k = 7, 8, the performance of the RKA is better than that of the PSA. However, the performance of the GA is still superior to that of the RKA. When r = 2, the first part of the RKA generates two disjoint feasible solutions. Namely, the corresponding sets of involved virtual sensors for the two feasible solutions are disjoint. Thus, the first part of the RKA ends up with a set V which contains exactly $2k$ virtual sensors. Since the number of candidate virtual sensors increases, the second part of the RKA is able to obtain a better solution than that of r = 1. For r = 1 and r = 2, the numbers of involved virtual sensors of the final solutions are both k = 30. The performance improvement is due to the increase of the number of candidate virtual sensors processed by the second part of the RKA. For k = 60, the performances of both the PSA and the GA improve significantly. These two points are marked along with the case of k = 30 for a clear comparison.

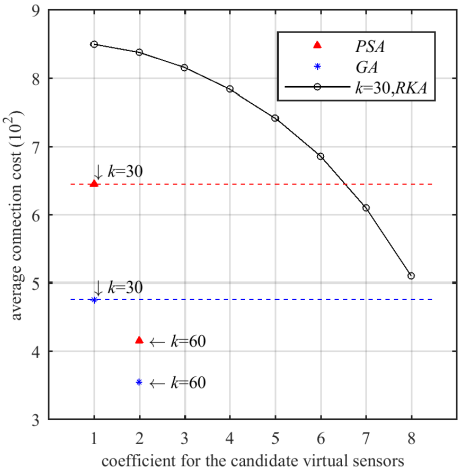

Fig. 7. Variation of cost with different r for RKA.

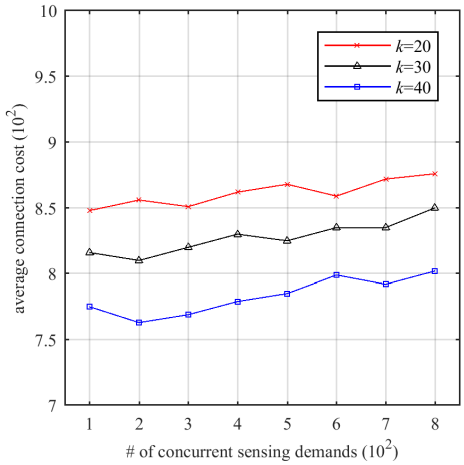

Fig. 8. Variation of cost with different m' for RKA.

To further investigate the RKA, we set the number of feasible solutions generated to three, namely r = 3. The number of virtual sensors simultaneously active in the sensor cloud is k = 20, 30, 40. The relation between the number of concurrent sensing demands and the average connection cost is depicted in Fig. 8. With the increase of m', the average connection cost of the whole network shows a rising trend. It is obvious the total connection cost of the whole network is monotonically increasing with the increase of the number of concurrent sensing demands. For the average connection cost of the whole network, the overall trend is also increasing. However, the way of increasing is not monotonic. Thus, the increasing of the number of concurrent sensing demands has the effect of degrading the performance of the RKA. Besides, the degree of degradation is not directly proportional to the increasing of the number of concurrent sensing demands. The average connection costs for k = 20, 30, 40 gradually decrease in a steady way. Thus, the increase of the number of virtual sensors simultaneously active in the sensor cloud is able to improve the performance of the RKA. This is similar to the cases of the PSA and the GA.

5. CONCLUDING REMARKS

We studied the problem of virtual sensor management in a manufacturing network. The vertical integration of a manufacturing network was demonstrated as four layers. We described the components and functions of each layer. In addition, we introduced eight entities and seven actions to model the operation of a manufacturing network. The interactions among the eight entities were conducted with eight messages. Based on the above building blocks, we formulated the issue of virtual sensor instantiation as a *k*-median problem and proposed a cost-efficient virtual sensor management scheme. Our proposal provided three resource scheduling algorithms: progressive swap algorithm, greedy algorithm, and RK algorithm. Simulations were conducted for five key parameters to evaluate the performance of the above three algorithms. The simulation results show that the performance differences between the progressive swap algorithm and the greedy algorithm are closely related to the upper bound which terminates the execution of the progressive swap algorithm and the number of virtual sensors which should be removed from the set of involved virtual sensors. Moreover, the value of k should be prudently determined based on different emphases on the number of virtual sensors simultaneously active in the sensor cloud and the total connection cost. To some degree, the progressive swap algorithm and the greedy algorithm represent two opposite strategies, respectively. A hybrid solution which combines the progressive swap algorithm and the greedy algorithm is formulated by the RK algorithm. The parameter r which indicates the number of feasible solutions generated by the first part of the RK algorithm is a key tuner of the performance. Besides, there is still room for improvement. To approach the operations in a real-world scenario, details such as message delay, physical sensor node failure, and energy conservation in wireless sensor networks should be introduced.

ACKNOWLEDGMENT

This work is partly supported by the International Science and Technology Cooperation Program of the Science and Technology Department of Shaanxi Province, China (Grant No. 2018KW-049), the Special Scientific Research Program of Education Department of Shaanxi Province, China (Grant No. 17JK0711), the National Science Foundation of China (Grant No. 61702414), the Shaanxi Science and Technology Coordination & Innovation Project (No. 2016KTZDGY04-01), and the Communication Soft Science Program of Ministry of Industry and Information Technology, China (Grant No. 2019-R-29). The authors would like to thank the anonymous reviewers whose comments and suggestions greatly helped us improve the quality and presentation of this paper.

REFERENCES

1. J. Lee, "Industry 4.0 in big data environment," *German Harting Magazine*, 2013, pp. 8-10.
2. R. Baheti and H. Gill, "Cyber-physical systems," *The Impact of Control Technology*, Vol. 12, 2011, pp. 161-166.

3. J. C. B. Fernandes, "Industrial mobile app for a sensor cloud," MS Thesis, Faculty of Engineering, University of Pedro, 2017.
4. A. Darwish and A. E. Hassanien, "Cyber physical systems design, methodology, and integration: the current status and future outlook," *Journal of Ambient Intelligence and Humanized Computing*, 2017, pp. 1-16.
5. M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing," in *Proceedings of IEEE 13th International Conference on Network-Based Information Systems*, 2010, pp. 1-8.
6. S. Madria, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors," *IEEE Software*, Vol. 31, 2014, pp. 70-77.
7. C.-F. Lai, H. Wang, H.-C. Chao, and G. Nan, "A network and device aware qos approach for cloud-based mobile streaming," *IEEE Transactions on Multimedia*, Vol. 15, 2013, pp. 747-757.
8. C.-F. Lai, H.-C. Chao, Y.-X. Lai, and J. Wan, "Cloud-assisted real-time translating for http live streaming," *IEEE Wireless Communications*, Vol. 20, 2013, pp. 62-70.
9. A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A survey on sensor-cloud: architecture, applications, and approaches," *International Journal of Distributed Sensor Networks*, Vol. 9, 2013.
10. S. Misra, S. Bera, A. Mondal, R. Tirkey, H.-C. Chao, and S. Chattopadhyay, "Optimal gateway selection in sensor-cloud framework for health monitoring," *IET Wireless Sensor Systems*, Vol. 4, 2013, pp. 61-68.
11. M. M. Hassan, B. Song, and E.-N. Huh, "A framework of sensor-cloud integration opportunities and challenges," in *Proceedings of the 3rd ACM International Conference on Ubiquitous Information Management and Communication*, 2009, pp. 618-626.
12. M. Eggert, R. Häußling, M. Henze, L. Hermerschmidt, R. Hummen, D. Kerpen, A. N. Pérez, B. Rumpe, D. Thißen, and K. Wehrle, "Sensorcloud: Towards the interdisciplinary development of a trustworthy platform for globally interconnected sensors and actuators," *Trusted Cloud Computing*, H. Krcmar *et al.*, (ed.), Springer, Switzerland, 2014, pp. 203-218.
13. S. Olariu, A. Wada, L. Wilson, and M. Eltoweissy, "Wireless sensor networks: leveraging the virtual infrastructure," *IEEE Network*, Vol. 18, 2004, pp. 51-56.
14. T. Ojha, M. Khatua, and S. Misra, "Tic-tac-toe-arch: a self-organising virtual architecture for underwater sensor networks," *IET Wireless Sensor Systems*, Vol. 3, 2013, pp. 307-316.
15. L. Cooper, "Location-allocation problems," *Operations Research*, Vol. 11, 1963, pp. 331-343.
16. R. Z. Farahani and M. Hekmatfar, *Facility Location: Concepts*, *Models*, *Algorithms and Case Studies*, Springer, Berlin, 2009.
17. G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacitated facility location problem," Technical Report No. 605, Department of Management Sciences, Carnegie-Mellon University, 1983.
18. J. Krarup and P. M. Pruzan, "The simple plant location problem: survey and synthesis," *European Journal of Operational Research*, Vol. 12, 1983, pp. 36-81.

19. K. Aardal, P. L. van den Berg, D. Gijswijt, and S. Li, "Approximation algorithms for hard capacitated *k*-facility location problems," *European Journal of Operational Research*, Vol. 242, 2015, pp. 358-368.
20. J. Chuzhoy and Y. Rabani, "Approximating k-median with non-uniform capacities," in *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005, pp. 952-958.
21. A. K. Jain and R. C. Dubes, "Algorithms for clustering data," 1988.
22. V. Arya, N. Garg, R. Khandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local search heuristics for *k*-median and facility location problems," *SIAM Journal on Computing*, Vol. 33, 2004, pp. 544-562.
23. https://www.debian.org/, 2018.
24. https://www.nsnam.org/, 2018.

Cong Gao (高聪) received the Ph.D. degree in computer architecture from Xidian University, Xi'an, China, in 2015. Currently, he is an Assistant Professor in the School of Computer Science and Technology at Xi'an University of Posts and Telecommunications, Xi'an, China. His current research interests include data sensing and fusion, network and information security, and service computing.

Zhenzhou Tian (田振洲) received the B.S. degree and Ph.D. degree in Computer Science and Technology from Xi'an Jiaotong University, Xi'an, China, in 2010 and 2016, respectively. He is currently a Lecturer in the School of Computer Science and Technology at Xi'an University of Posts and Telecommunications. His research interests include trustworthy software, software plagiarism detection and software behavior analysis.

Yanping Chen (陳彦萍) received the Ph.D. degree from Xi'an Jiaotong University, Xi'an, China, in 2007. Currently, she is a Professor in the School of Computer Science and Technology at Xi'an University of Posts and Telecommunications, Xi'an, China. Her current research interests include service mining, service computing, and network management.

Zhongmin Wang (王忠民) received the Ph.D. degree from Beijing Institute of Technology, Beijing, China, in 2000. Currently, he is a Professor in the School of Computer Science and Technology at Xi'an University of Posts and Telecommunications. His current research interests include embedded intelligent perception, big data processing and application, and affective computing.