Received September 30, 2017, accepted October 24, 2017, date of publication November 1, 2017, date of current version December 5, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2768544

Energy-Efficient Composition of Configurable Internet of Things Services

MENGYU SUN[1], ZHENSHENG SHI[2], SHENGJUN CHEN[3], ZHANGBING ZHOU[1,4], AND YUCONG DUAN[5]

[1]School of Information Engineering, China University of Geosciences, Beijing 100083, China
[2]PetroChina Research Institute of Petroleum Exploration and Development, Langfang 065007, China
[3]Business School, University of International Business and Economics, Beijing 100029, China
[4]Computer Science Department, Telecom SudParis, 91011 Evry, France
[5]College of Information Science and Technology, Hainan University, Haikou 570228, China

Corresponding author: Zhangbing Zhou (zhangbing.zhou@gmail.com)

ABSTRACT Along with the adaptation of Internet of Things (IoT) to support various industrial applications, the cooperation and coordination of smart things is a promising strategy for satisfying requirements that are beyond the capacity of any single smart thing. To address this challenge, a two-tier IoT service framework is proposed, where the functionalities provided by smart things are encapsulated into IoT services, which are categorized into service classes. The service network is constructed by considering the invocation possibility between service classes, and service class chains are generated using traditional Web service composition techniques, where the functional specification of certain requirements is considered. Considering factors, such as spatial and temporal-constraints, energy efficiency, and the configurability of IoT services, selecting IoT services for the instantiation of service classes contained in chains is reduced to a multiobjective and multiconstrained optimization problem. Heuristic algorithms, such as the genetic algorithm (GA), ant colony optimization (ACO) and particle swarm optimization (PSO), are adopted to search for optimal IoT service compositions. An experimental evaluation shows that PSO performs better than the GA and ACO in searching for approximately optimal IoT service compositions and reduces the energy consumption, thus prolonging the network lifetime.

INDEX TERMS IoT services, service composition, reconfiguration, energy efficiency.

I. INTRODUCTION

With the advent and rapid development of the Internet of Things (IoT), smart things communicate with each other and are interconnected via networks to facilitate their collaboration and cooperation to support various industrial applications [1], [2]. As a special example of IoT, wireless sensor networks (WSNs) have been applied in various domains including military surveillance, industry applications, and smart homes [3]–[5]. Smart things are usually heterogeneous and may vary to a large extent in their capacities. Mostly, a single smart thing is weak, or at least not strong, in terms of its functionalities and computational and storage capacities and has limited energy, especially when the smart thing is battery-powered. Therefore, a smart thing, such as a (mobile) sensor node or a smart phone, can usually provide a limited number of functionalities. Considering the limitations, including the capacity constraint and the reduction of energy consumption, the functionalities provided by contiguous smart things should be composed, especially when a relatively complex requirement must be fulfilled [6]. Because conflicts potentially exist between the functionalities co-hosted by a certain smart thing, these functionalities should be configurable as required, which indicates that they should be turned on or switched off online with respect to certain requirements after a certain time duration [7]. For instance, a sensor node cannot sense humidity and chemical gases (like N_xO_y) simultaneously, as certain chemical gases may dissolve in water. On the other hand, a sensor node can always sense humidity and temperature at the same time. Moreover, when a sensor node is short of energy, a contiguous sensor node with relatively abundant energy should be adopted alternatively to provide a certain functionality to balance the energy consumption of all sensor nodes and prolong the network lifetime. Consequently, combining the functionalities provided by smart things, while considering their spatial and temporal constraints, the energy efficiency at the network level, and the configurability of their functionalities, is a challenge to be addressed, especially when the requirement is relatively complex, and hence is beyond the capacity of any single smart thing.

By adopting machine-to-machine communication techniques [8], Currently, most domain applications can be supported by integrating complemental functionalities provided by smart things in industrial IoT environments [9]. In fact, service discovery and composition are long-standing research topics, and many techniques have been developed in the Web/REST service domain [10], [11], whereas these techniques take the functional and non-functional properties of services as first-class citizens. Along with the development of service-defined everything [12], [13], smart things are represented in terms of IoT services [14], which correspond to the set of their co-hosted functionalities [15]. Consequently, the cooperation and collaboration between smart things are reduced to the composition of IoT services [16]–[20]. To summarize, current techniques adopt service-oriented architectures to support IoT service discovery and composition, where the quality of services and the middleware development are the main concerns. Note that spatial and temporal- constraints, as well as the energy efficiency, are generally not considered as essential ingredients by the majority of current techniques. To mitigate this problem, we have developed a service-oriented mechanism for composing WSN services, which are encapsulated from various functionalities provided by sensor nodes [15]. Specifically, a sensor node is assumed to provide a certain kind of functionality, which is represented as the corresponding WSN service. Spatial and temporal- constraints are considered when the composition of WSN services is instantiated, and the energy- efficiency is also an important factor to consider for prolonging the network lifetime. It is worth mentioning that a smart thing such as a sensor node can usually host multiple services upon the relatively powerful hardware device, which should be configurable, considering potential conflicts between functionalities and the balancing of energy consumption. To address this challenge, this article proposes an energy-efficient mechanism for composing IoT services, where these services are configurable on the co-hosting smart things when necessary. The main contributions of this article are summarized as follows:

- A two-tier IoT service framework is proposed, as shown in Fig. 1, where the functionalities provided by smart things are encapsulated into IoT services by adopting, for example, the Devices Profile for Web Services (DPWS) standard [21]. These IoT services are categorized into service classes according to their functionalities, while the configurability of these IoT services is not considered when these services are co-hosted by certain smart things. A service network is constructed while considering the invocation possibility between service classes, and service class chains are generated using traditional Web service composition techniques, where the functional specification of certain requirements is considered.
- Service classes chains are instantiated by discovering appropriate IoT services for service classes in chains. Spatial and temporal- constraints of IoT services inherited from smart things, the balancing of energy consumption, and the configurability of IoT services are the constraints to be considered. The composition of IoT services can be reduced to a multiobjective and multiconstrained optimization problem, where optimal solutions are derived by adopting heuristic methods, including the genetic algorithm (GA), ant colony optimization (ACO), and particle swarm optimization (PSO).

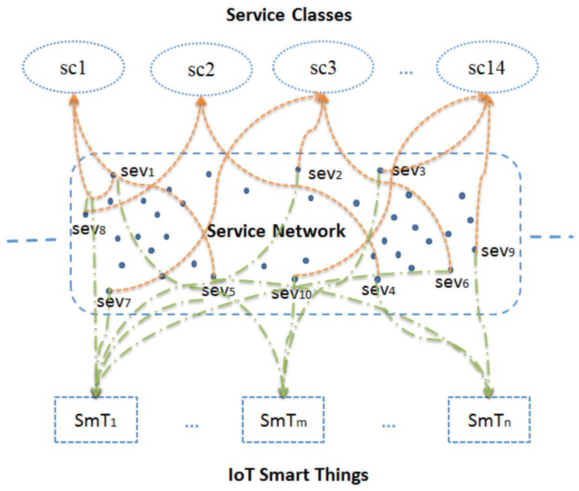


FIGURE 1. Smart things (SmT_i), IoT services (sev_j), and service classes (sc_k), where a smart thing can co-host one or multiple IoT services, and an IoT service corresponds to a certain service class according to the functionality it can provide.

Extensive evaluations are conducted to evaluate the accuracy and effectiveness of the IoT service composition technique developed in this article. The results show that approximately optimal IoT service compositions can be derived and that PSO performs better than the GA and ACO regarding the fitness and energy consumption.

The rest of this article is organized as follows. Section II introduces related concepts and the energy model. Section III presents the mechanism for service class chains recommendation. Section IV develops the instantiation technique for supporting IoT service composition. Section V shows the result of our experimental evaluation. Section VI discusses and compares relevant techniques. Finally Section VII concludes this work.

II. PRELIMINARIES

This section defines certain concepts and introduces the energy model, all of which are used in the following sections.

A. CONCEPT DEFINITION

A smart thing should co-support one or several functionalities, which are represented in terms of IoT services. Generally:

Definition 1 (Smart Thing): A smart thing SmT_i is a tuple $(id, nm, sev_{list}, eng, spt, tpr)$, where:

- id is the unique identifier of SmT_i.
- nm is the name of SmT_i.
- sev_{list} is the set of IoT services co-hosted by SmT_i.
- eng is the remaining energy of SmT_i.
- spt is the spatial constraint of SmT_i.
- tpr is the temporal constraint of SmT_i.

Note that spt is defined by the geographical location and communication radius of SmT_i. tpr is the time duration during which SmT_i is active.

Definition 2 (IoT Service): An IoT service sev_m is a tuple (nm, dsc, op, SmT), where:

- nm is the name of sev_m.
- dsc is the text description of sev_m.
- op is an operation of sev_{sn}.
- SmT is the smart thing hosting sev_m.

Intuitively, smart things can support certain types of IoT services, although they are different regarding their spatial and temporal- constraints, and their remaining energy. To facilitate our two-tier service framework, the concept of service class is defined to specify IoT services with a certain functionality:

Definition 3 (Service Class): A service class sev_{cl} for a certain IoT service sev_m is a tuple (nm, dsc, op), where nm, dsc, and op are the same as those specified for sev_m.

A service class should be instantiated by the appropriate IoT service during the service composition procedure. Service classes are chained before the instantiation of service classes, and this procedure is conducted by leveraging the service network:

Definition 4 (Service Network): A service network sn is a tuple (sc, lnk, ivp), which is represented in terms of a directed graph, where:

- sc is the set of service classes, corresponding to the vertexes contained in sn.
- lnk is the set of invocation possibility relationships between service classes corresponding to the directed edges contained in sn.
- ivp is the set of invocation possibilities between service classes, corresponding to the weights specified on the edges contained in sn.

As presented in our previous work [22], the construction of a service network is mainly carried out to compute the invocation possibility between service classes and to prune the links that suggest a relatively low invocation possibility. The invocation possibility is computed by aggregating the semantic similarity of the name, the text description, and the operation of a certain pair of service classes. We refer the reader to [22] for the algorithm. Given an invocation possibility value between 0 and 1 for two service classes, the larger the value is, the higher the possibility is for the invocation between these service classes. When this value is smaller than a prespecified threshold, which means that it is hardly possible for an invocation to occurred between the given service classes, the corresponding link is pruned. Consequently, a service network is constructed and represented in terms of a weighted directed graph, in which the vertexes are service classes and the weights on the directed edges represent the invocation possibilities.

A sample service network is shown in Fig. 2, which includes 14 sample service classes (denoted as sc_j, where $j \in \{1, 14\}$) that are also presented in Table 3. The threshold is set to 0.511 according to the experiments in this article, and 101 directed edges are reserved in this figure. Note that this threshold can be changed to another appropriate value when necessary according to the certain domain requirements. The thickness of the directed edges reflects the value of the invocation possibility such that the thicker the directed edges are, the larger the invocation possibility is.

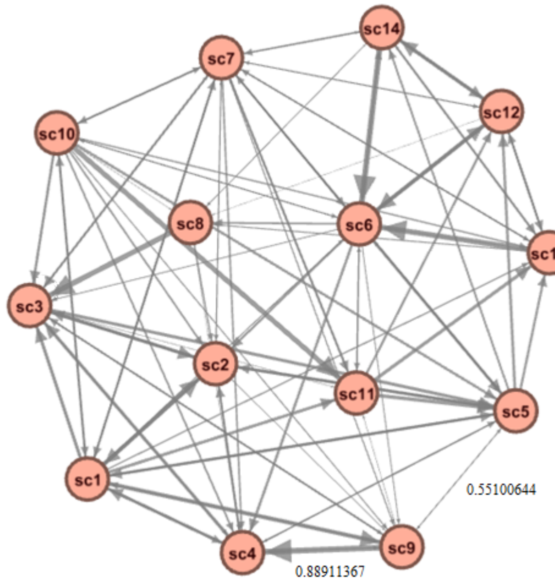


FIGURE 2. Sample service network with 14 service classes, where sc_j represent service classes and $j \in \{1, 14\}$.

B. ENERGY MODEL

Since sensor nodes can be regarded as the most common smart things, we adjust the first-order radio model [23], which is the often adopted in the WSN domain, to calculate the energy consumption of smart things. Table 1 presents the parameters of this model. Specifically, $E_{Tx}(k, d)$ represents the energy consumed when transmitting k bits within a distance d, while $E_{Rx}(k)$ represents the energy consumed when receiving a k bit packet, the formulae are as follows:

$$E_{Tx}(k,d) = E_{elec} \times k + \epsilon_{amp} \times k \times d^n \quad (1)$$

$$E_{Rx}(k) = E_{elec} \times k \quad (2)$$

where E_{elec} is the energy consumption constant of the transmission and receiver electronics and ϵ_{amp} is the energy consumption constant of the transmission amplifier. Therefore, the energy consumed when transmitting a packet

TABLE 1. Parameters in the energy model.

Name	Description
E_{elec}	Energy consumption constant of the transmit and receiver electronics.
ϵ_{amp}	Energy consumption constant of the transmit amplifier.
k	The number of bits in one packet.
d	The distance of transmission.
n	The attenuation index of transmissions.
r	The communication radius of smart things.
$E_{Tx}(k,d)$	The energy consumed to transmit a k bit packet to a distance d.
$E_{Rx}(k)$	The energy consumed to receive a k bit packet.
$E_{ij}(k)$	Energy consumption for transmitting a k bit packet from a smart thing SmT_i to a neighboring smart thing SmT_j
$E_{inv}(SmT_i.sev_m)$	Energy consumption for activating an instantiation of a service sev_m in a smart thing SmT_i.
$E_{st}(SmT_i)$	The energy consumption of waiting time in conflict per unit in a smart thing SmT_i.

of k bits from a smart thing SmT_i to one of its neighboring smart things SmT_j, denoted as $E_{ij}(k)$, is calculated as follows:

$$E_{ij}(k) = E_{Tx}(k,d) + E_{Rx}(k) \tag{3}$$

The cost of transmitting a packet from a smart thing to a neighboring smart thing or to a sink node (denoted as SN) is different, as an SN is assumed to have unlimited energy. Therefore, an SN has no energy constraint, and the cost of receiving packets is not considered in this model. Consequently, $E_{ij}(k)$ is as follows:

$$E_{ij}(k) = \begin{cases} E_{elec} \times k + \epsilon_{amp} \times k \times d^n & \text{if } j \text{ is SN} \\ 2 \times E_{elec} + \epsilon_{amp} \times k \times d^n & \text{otherwise} \end{cases} \tag{4}$$

where the parameter d represents the distance between the smart things SmT_i and SmT_j (or SN). Assume that the energy needed to transmit a packet from SmT_i to SmT_j is the same as that needed to transmit a message from SmT_j to SmT_i. The parameter n is the attenuation index of transmission, which is influenced by the surrounding environments. Generally, if smart things in the IoT network are barrier-free when forwarding packets, n is set to 2. Otherwise, n is set to a value between 3 to 5.

III. SERVICE CLASS CHAINS RECOMMENDATION

Leveraging the service network constructed in the previous section, we introduce the service class chains generation and recommendation. Regarding the technical details, we refer the reader to our previous work [22] for the algorithms. Generally, end- users may not be domain experts, and the requirement of certain applications can hardly be specified in a very clear and precise fashion. In fact, this is quite common, especially when the task to be solved is relatively vague in the initial stage. On the other hand, end- users can describe their requirement in plain text regarding the initial and ending states (denoted as sta_{ini} and sta_{end} respectively), and possibly the input and output parameters of sta_{ini} and sta_{end}. In this setting, by calculating the similarity of service classes contained in the service network with respect to sta_{ini} (or sta_{end}) via the consideration of their plain-text description and input and output parameters, candidate service classes for sta_{ini} (or sta_{end}) can be determined and ranked. Without loss of generality, service class sev_{ini} (or sev_{end}) with the largest similarity is selected as the initial (or ending) state.

After determining the service classes of the initial state and ending state, candidate service class chains should be retrieved from the service network using the depth-first graph search algorithm with a prespecified limitation on depth [22]. In this article, the depth (denoted as $dept_{lmt}$) is set to 5, and $dept_{lmt}$ can be changed to another appropriate value according to the requirements.

When several service class chains are retrieved, these chains are ranked according to the average weight specified on the links. Specifically, the average weight is computed using the following parameters: (i) the sum of weights on all direct links, (ii) the length of the service class chain. Without loss of generality, the service class chain with the largest weight on average should be selected as the most appropriate candidate.

For instance, assume that the initial and ending states are described in plain- text as follows:

- The initial state: "*Gathers data from multiple sensor nodes in a wireless sensor network, which may include temperature, humidity, wind direction, wind power, ambient smog, light, accelerometer, and other sensors commonly used for monitoring fire alarm, and so on. Public reports to generate the accurate location of the fire in the firehouse.*"
- The ending state: "*According to the gathered data from corresponding sensors, such as temperature, humidity, ambient smog, wind direction, wind power, light sensors and so on, to generate the current fire region and forecasts of fire spreading in the following time duration. Properly identify the incident, raise the occupant alarm, and then notify emergency response professionals timely.*"

By calculating the similarity between the above description and the text description of service classes as presented in Table 3, *sc4* and *sc10* are determined as the initial state and the ending state, respectively. Candidate service class chains are derived from the service network as shown in Fig. 2. These five chains are listed below, where the sequence represents the invocation order between service classes.

- The first candidate service class chain, the length of which is 5:
  1) *sc4*: *Light sensing service*
  2) *sc9*: *Ambient smog sensing service*
  3) *sc1*: *Ambient temperature sensing service*
  4) *sc3*: *Relative humidity sensing service*
  5) *sc10*: *Wind direction sensing service*

  The weight on average is 0.610681398.
- ...
- The third candidate service class chain, the length of which is 4:
  1) *sc4*: *Light sensing service*
  2) *sc1*: *Ambient temperature sensing service*
  3) *sc2*: *Temperature sensing service*
  4) *sc10*: *Wind direction sensing service*

  The weight on average is 0.591111399.
- ...
- The fifth candidate service classes chain, the length of which is 5:
  1) *sc4*: *Light sensing service*
  2) *sc2*: *Temperature sensing service*
  3) *sc1*: *Ambient temperature sensing service*
  4) *sc5*: *Pressure sensing service*
  5) *sc10*: *Wind direction sensing service*

  The weight on average is 0.559457158.

When candidate service class chains (denoted as *CHN*) are generated, the service classes in these chains should be instantiated by discovering and selecting appropriate IoT services, this IoT service composition procedure is presented in the following section.

IV. IoT SERVICE COMPOSITION

This section presents our IoT service composition mechanism, which discovers and selects IoT services for the instantiation of service classes in chains, where spatial and temporal- constraints, potential conflicts between IoT services co-hosted by certain smart things, and the energy efficiency are the main concerns.

A. IoT SERVICE CONSTRAINTS

1) SPATIAL AND TEMPORAL CONSTRAINTS

Smart things can usually work well under certain spatial and temporal constraints. Intuitively, the spatial- constraint of a certain smart thing SmT_i is specified according to the physical location and the communication radius and can be represented as follows:

$$spt(SmT_i) = (p_{SmT_i}, r_{SmT_i}) \tag{5}$$

where p_{SmT_i} is the geographical location of the smart thing SmT_i, which is represented by its latitude and longitude, while r_{SmT_i} is the communication radius, which represents the maximum transmission distance of SmT_i. Note that a user requirement rq has the spatial- constraint as well, which can be specified as follows:

$$spt(rq) = (p_{rq}, r_{rq}) \tag{6}$$

Therefore, the spatial relevancy between rq and SmT_i is computed as follows:

$$spt(rq, SmT_i) = (spt(SmT_i) \cap spt(rq)) \div spt(rq) \tag{7}$$

where $spt(rq,\ SmT_i)$ represents the degree of overlap between $spt(SmT_i)$ and $spt(rq)$. We do not consider smart things that are beyond the scope of the requirement.

The temporal- constraint is specified in a similar fashion. The user requirement rq usually needs to be carried out within a certain time duration $tpr(rq)$. Similarly, an IoT service hosted by a smart thing SmT_i is not operated within all time duration to reduce the energy consumption, and it should be available only for the prespecified time duration $tpr(SmT_i)$. Consequently, the temporal relevancy for rq and SmT_i is computed as follows:

$$tpr(rq, SmT_i) = (tpr(SmT_i) \cap tpr(rq)) \div tpr(rq) \tag{8}$$

Intuitively, $tpr(rq, SmT_i)$ represents the common time duration for the user requirement and the smart thing, which indicates that the smart thing can provide these time durations to support the user requirement.

2) CONFLICTS BETWEEN IoT SERVICES

A smart thing can usually host multiple IoT services, which should be configurable when these IoT services functionally conflict with each other. For a service class chain, we should consider whether there is a delay between two consecutive service classes (sc_m and sc_{m+1}, $0 \leq m \leq k$, where k represents the number of service classes) when these two IoT services are instantiated in the same smart thing. That is, when the previous IoT service is completed, can the smart thing start the next service immediately without any halt. Taking this factor into account, the priority coefficient is proposed in the selection of instantiating every single service class. For each service class in chains, we build a set of aggregated alternative smart things, which contains certain functionalities, and calculate the priority coefficient for each smart thing in the aggregation. Considering two consecutive IoT services that are co-hosted in the same smart thing and have conflicting functionalities, there are certain influences on energy consumption in the service network. Energy consumption occurs during the delay time of two consecutive IoT services. However, no data packets are transmitted and received when IoT services function in the same smart thing.

Given two constant IoT services, the delay time in a certain smart thing SmT_i is denoted as $delay(SmT_i.sev_m, SmT_i.sev_{m+1})$, and the priority coefficient of a certain smart thing for a certain IoT service (denoted as $pri(SmT_i.sev_m)$) is computed as follows:

$$pri(SmT_i.SmT_m) = \begin{cases} E_{inv}(SmT_i.sev_m) \\ \dfrac{E_{TR} + E_{st} * delay(SmT_i.sev_m, SmT_i.sev_{m+1})}{E_{TR}} \\ *E_{inv}(SmT_i.sev_m) \end{cases} \tag{9}$$

where E_{TR} is the sum of $E_{Tx}(k,d)$ and $E_{Rx}(k)$. The first equation represents the situation where sev_m has no conflict with the previous IoT service. The second one indicates that the next IoT service selection needs to take the previous one into account and checks whether the two IoT services have a delay time if they are co-hosted by the same smart thing. Each time an IoT service is selected, the coefficient of priority is calculated for the next IoT service in the alternative smart things aggregation.

3) ENERGY CONSTRAINTS

To prolong the network lifetime, IoT services should be configured properly, considering the load- balancing of smart things and avoiding the excessive consumption of any smart thing. Generally, IoT services supported by smart things with relatively large amounts of remaining energy (denoted as $reg(SmT_i)$) should be chosen for the instantiation of certain service classes (denoted as $E_{cst}(SmT_i.sev_m)$). Therefore, the energy consumption for a certain IoT service adopted in the service composition is computed as follows:

$$E_{cst}(SmT_i.sev_m) = E_{inv}(SmT_i.sev_m) * t(SmT_i.sev_m) \quad (10)$$

where $t(SmT_i.sev_m)$ is the invocation time for a certain IoT service hosted by a certain smart thing.

The total energy consumption of a certain smart thing containing all the IoT services is computed as follows:

$$E_{cst}(SmT_i) = \Sigma_{m=1}^{ns(SmT_i)} E_{cst}(SmT_i.sev_m) \quad (11)$$

where $ns(SmT_i)$ is the number of IoT services that a certain smart thing SmT_i can configure.

To avoid the over-consumption of energy for any smart thing, we should try to prevent a smart thing with relatively low remaining energy from completing an IoT service requiring a relatively large amount of energy consumption. Therefore, an energy- consumption ratio is proposed, which is computed as follows:

$$ecr(SmT_i.sev_m) = E_{inv}(SmT_i.sev_m) \div reg(SmT_i) \quad (12)$$

To estimate the usability of smart things, the following threshold is proposed:

$$thrd = \Sigma_{i=1}^{nh(sev_m)} \frac{E_{inv}(SmT_i.sev_m)}{reg(SmT_i)} \div nh(sev_m) \quad (13)$$

where $nh(sev_m)$ is the number of smart things that can be chosen to support a certain IoT service sev_m. $thrd$ is the average energy- consumption ratio for all smart things. A load-balancing factor (denoted as lbf) leveraging $ecr(SmT_i.sev_m)$ and $thrd$ is defined to examine whether a certain IoT service hosted on a certain smart thing is optimal, where $lbf(SmT_i.sev_m)$ is calculated as follows:

$$lbf(SmT_i.sev_m) = \frac{ecr(SmT_i.sev_m)}{thrd} \quad (14)$$

When the value of lbf is relatively large, the energy consumption required to complete an IoT service has a significant proportion with respect to the residual energy of a certain smart thing. Therefore, the IoT service supported by a certain smart thing should be less than that of its rivals to balance the energy consumption of the whole network.

B. ENERGY CONSUMPTION OF IoT SERVICE COMPOSITION

The energy in an IoT services network is not infinite. To prolong the lifetime of a network, the energy consumption required to compose these service classes should be as low as possible. For a service class chain $chn = sc_1 \rightarrow sc_2 \rightarrow sc_3 \rightarrow \ldots \rightarrow sc_k$, assuming that sev_m is an instance in the service class sc_m, the composition of IoT services is instantiated as $cp(chn) = sev_1 \rightarrow sev_2 \rightarrow sev_3 \rightarrow \ldots \rightarrow sev_k$. The energy consumption for a certain composition $cp(chn)$ includes the following parts [24]:

- The energy consumption required to activate the instantiation of IoT services in the service class chain $cp(chn)$ is computed as follows:

$$E_{inv}(cp(chn)) = \Sigma E_{cst}(SmT_i) \quad (15)$$

  In fact, this is the total energy consumption for all smart things that invoke all the IoT services in $cp(chn)$.
- The energy consumption for the standby-state in the whole $cp(chn)$ is computed as follows:

$$E_{st}(cp(chn)) = \Sigma delay(SmT_i.sev_m, SmT_i.sev_{m+1}) * E_{td} \quad (16)$$

  Note that the selection procedure has two consecutive service restrictions. Specifically, when finishing the first IoT service and waiting to launch the next one, the smart thing that is selected operates in the standby- stage, this process also consumes energy. Therefore, $E_{st}(cp(chn))$ represents the total energy consumption for the whole chain in this case.
- The energy consumed to transmit and receive data packets in the $cp(chn)$ is computed as follows:

$$E_{TR}(cp(chn)) = (leg(chn) - time_{sa} - 1) * (E_{Tx}(k,d) + E_{Rx}(k)) \quad (17)$$

  where $leg(chn)$ is the length of the service class chain contained in chn. $time_{sa}$ is the amount of time needed for two consecutive services hosted on the same smart thing. When there are no packets that need to be transmitted between two smart things, $E_{Tx}(k, d)$ and $E_{Rx}(k)$ are set to 0. This situation occurs when two consecutive services are deployed on the same smart thing or the IoT service sev_m is the last one in the service chain. Note that the first IoT service sev_i in chn only needs to transmit a data packet, and the last IoT service sev_k only needs to receive a data packet. Therefore, the total energy consumption for a certain service composition $cp(chn)$ is calculated as follows:

$$E(cp(chn)) = E_{inv}(cp(chn)) + E_{st}(cp(chn)) + E_{TR}(cp(chn)) \quad (18)$$

C. IoT SERVICE COMPOSITION

Given a service class chain $chn = sc_1 \rightarrow sc_2 \rightarrow sc_3 \rightarrow \ldots \rightarrow sc_k$, each sc_m should be instantiated by a certain IoT service while taking the constraints into account. This process can be formulated as a multiobjective and multiconstrained optimization problem as follows.

- *Input Parameters*:
  1) *CHN*: the set of service class chains *chn* constructed and recommended in Section III.
  2) $spt(rq)$: the spatial constraint of a user requirement rq.
  3) $spt(SmT_i)$: the spatial constraint of a certain smart thing SmT_i.
  4) $tpr(rq)$: the temporal constraint of a user requirement rq.
  5) $tpr(SmT_i)$: the temporal constraint of a certain smart thing SmT_i.
  6) $E_{Tx}(k,d)$: the energy consumed when transmitting a k bit packet within a distance d.
  7) $E_{Rx}(k)$: the energy consumed when receiving a k bit packet.
  8) $E_{ij}(k)$: the energy consumed when transmitting a k bit packet from a smart thing SmT_i to a neighboring smart thing SmT_j.
  9) $E_{inv}(SmT_i.sev_m)$: the energy consumed when activating an instantiation of an IoT service sev_m in a smart thing SmT_i.
  10) $E_{st}(SmT_i)$: the energy consumption of stand-by time per unit in a smart thing SmT_i.
  11) $delay(SmT_i.sev_m, SmT_i.sev_{m+1})$: for two contiguous IoT services sev_m and sev_{m+1}, the time of delay in a certain smart thing SmT_i.
  12) $t(SmT_i.sev_m)$: the invocation time for a certain IoT service sev_m configured in a certain smart thing SmT_i.
  13) $ns(SmT_i)$: the number of IoT services that a certain smart thing SmT_i can configure.
  14) $nh(sev_m)$: the number of smart things that can be chosen for a certain IoT service sev_m.
  15) $reg(SmT_i)$: the residual energy of a certain smart thing SmT_i.
- *Output Parameters*:
  1) $cp(chn)$: the optimal composition of IoT services in the *CHN*.
- *Multiobjective Functions*:
  1) Minimize:
     $f_1 = E(cp(chn))$
     $f_2 = \alpha * pri(cp(chn)) + \beta * lbf(cp(chn))$
  2) Maximize:
     $f_3 = \varphi * spt(cp(chn)) + \delta * tpr(cp(chn))$

  where φ, δ, α and β are positive constant factors, and $\varphi + \delta = 1, \alpha + \beta = 1$.
  Thus:
  1) $pri(cp(chn)) = \sum pri(SmT_i.sev_m) \div leg(chn)$
  2) $lbf(cp(chn)) = \sum lbf(SmT_i.sev_m) \div leg(chn)$
- *Constraints*:
  1) $reg(SmT_i) \geq E_{cst}(SmT_i)$
- *Fitness Function*:

$$\begin{aligned} & fit(cp(chn)) \\ & \quad = w_1 * f_1 + w_2 * f_2 - w_3 * f_3 \\ & \quad = w_1 * E(cp(chn)) + w_2 * (\alpha * pri(cp(chn)) + \beta \\ & \qquad * lbf(cp(chn))) - w_3 * (\varphi * spt(cp(chn)) \\ & \qquad + \delta * tpr(cp(chn))) \end{aligned}$$

where w_1, w_2 and w_3 are positive constant factors representing the importance of the functions f_1, f_2 and f_3, respectively, and $w_1 + w_2 + w_3 = 1$. The fitness function measures whether the IoT service class chain composition $cp(chn)$ can satisfy the objective functions and constraint functions aforementioned, the smaller the value of the fitness function is, the better the composition is.

D. OPTIMIZATION ALGORITHM IN IoT SERVICE COMPOSITION

To solve the multiobjective and multiconstrained optimization problem, three optimization algorithms, including the genetic algorithm (GA), ant colony optimization (ACO), and particle swarm optimization (PSO), are adopted, their procedures are presented below:

1) GENETIC ALGORITHM

The GA is a heuristic algorithm used to search for the optimal solution by simulating the natural evolutionary process, it includes four steps: *inheritance*, *selection*, *mutation* and *crossover*.

In this article, a chromosome represents a service class chain, and the aggregation of service class chains *CHN* comprises a population (denoted as p_i), where i ranges from 1 to the size of the *CHN*. The genes represent service classes in chains, and the fitness function of chromosomes represents the individual fitness value of the population. We calculate the fitness function value via the method presented in Section IV-C, the smaller the fitness function value of $cp(chn)$ is, the better the $cp(chn)$ is in the population.

The procedure of selecting the optimal individual and eliminating the inferior individuals is called *selection*. For each population, the fitness function values are calculated for all the chromosomes, and only a proportion of the population is selected to be carried over into the next generation. The next generation is generated via the combination of *crossover* and *mutation*. *Crossover* occurs for two chromosomes chosen in a population, selecting a position as a crossover point and exchanging the genes of those separated from the point while obtaining two new chromosomes in the next generation. *Mutation* is a random change in a gene sequence. It randomly chooses a gene from the chromosome and alters the gene. The crossover and mutation probabilities are set according to specific examples. These processes aim to obtain the most appropriate IoT service composition in the *CHN*.

Algorithm 1 IoT Service Composition via GA

```
Require:
    - $MAX\_GEN$: the maximum number of generations.
    - $CHN$ : the aggregation of service class chains.
Ensure:
    - an approximately optimal chromosome.
1: while $looptime < MAX\_GEN$ do
2:     choose the chromosome that has the minimum fitness value in the current generation and add it to the new generation.
3:     for $i < |CHN| - 1$ do
4:         select chromosomes randomly via roulette wheel selection, a chromosome that has a smaller fitness function value is more likely to be chosen.
5:     end for
6:     for $i < |CHN| - 1$ do
7:         apply the crossover and mutation methods to evolve chromosomes for the next generation.
8:     end for
9:     find an approximately optimal chromosome.
10: end while
```

Algorithm 1 is the procedure carried out when using the genetic algorithm. Specifically, after generating the initial population, the fitness function values are computed for all chromosomes, and the chromosomes with the minimum values are reserved for the next generation (line 2). Roulette wheel selection is adopted to obtain the remaining chromosomes (line 4), and all new chromosomes are put into the new generation. Thereafter, two chromosomes in the new generation are chosen to implement the crossover operation to obtain two new chromosomes and replace the old chromosomes (line 7). In addition, the method of mutation is also applied for chromosomes selected randomly to obtain new chromosomes (line 7). These procedures are iterated until the maximum number of generations has been reached or an optimal fitness function value has been derived. Consequently, the chromosome with the minimum fitness function value is chosen as the candidate.

2) ANT COLONY OPTIMIZATION

ACO is an optimization algorithm inspired by ants for looking for food. The main steps involve selecting the next service vertexes and updating the pheromone matrix. Regarding the selection procedure, ants are scheduled to choose a service randomly as the starting service vertex and then move from one service to another. A pheromone matrix is constructed to contain the pheromone value between every two services. For every service chain that has been selected, the pheromone matrix is updated with the fitness function value of the service chain for any two continuous services. In the following procedure, for the service vertices, the larger the pheromone value is, the larger the possibility that a service is selected by the ants.

Algorithm 2 Service Composition via ACO

```
Require:
    - $MAX\_GEN$: the maximum number of iterations.
    - $NUM$: the number of ants.
    - $N_s$: the number of IoT services.
    - $CHN$: the aggregation of service class chains.
Ensure:
    - an approximately optimal service chain.
1: for $looptime < N_s$ do
2:     set initial pheromone matrix.
3: end for
4: for $looptime < MAX\_GEN$ do
5:     for $i < NUM$ do
6:         choose a starting service vertex randomly.
7:         for $j < |CHN|$ do
8:             choose the following service vertexes according to the pheromone matrix.
9:         end for
10:        update the pheromone matrix with the value of the fitness function.
11:    end for
12:    find an approximately optimal service chain.
13: end for
```

Algorithm 2 presents the general service composition procedure involving ant colony optimization. First, we set the initial value for the pheromone and form a matrix N_s (line 2). Then, a starting vertex is chosen randomly, as the pheromone value is the same at this moment (line 6). Regarding subsequent vertexes, the previous vertex is chosen in the pheromone matrix according to the row or the column vertex, and the next vertex is selected via roulette wheel selection in terms of the pheromone (line 8). Whenever an ant finishes the path selection, the pheromone matrix is updated by adopting the last fitness value (line 10). Finally, a service route that has the minimum fitness function value is found. This procedure iterates until the iteration threshold is reached.

3) PARTICLE SWARM OPTIMIZATION

PSO is one of the evolutionary algorithms inspired by the bird flock-prey behavior. In PSO, the solution to each optimization problem is a bird in the search space, and a solution is called a particle. All particles have a fitness value determined by the optimized function, and they follow the current optimal particle to search for the best solution.

In our context, each service class chain chn corresponds to a particle p_i, and p_i is the number of particles i defined as $p_i = (p_{i1}, p_{i2}, \ldots, p_{ij}, \ldots)$, where p_{ij} is a certain service in the service class sc_j in chn. The particle position changes when a service sev_i is substituted with another service in chn. $cp(chn)$ corresponds to an instantiated service of a certain particle. There are two values that require special attention: (i) $gbest$: the most appropriate position for all particles in the swarm at present, and (ii) $pbest$: the most appropriate position

TABLE 2. Parameters settings in the experiments.

Parameters Name	*Value*
Network region	$500m \times 500m$
Number of sensor nodes	400
Skewness degree	40%
Number of service classes	14
Number of hardware devices	30
Initial energy of hardware devices	0.5J
Total number of iterations	50
Communication radius	50m
Energy consumption constant for the transmit and receiver electronics	50nJ/bit
Energy consumption constant for the transmit amplifier	$100pJ/(bit \times m2)$
Energy consumption of stamp-by time	$100nJ/unit$

that has been found. The particles update their speeds and new locations according to the following formulas:

$$v(t{+}1) = w * v(t) + c1 * r1 * (pbest(t) - x(t)) + c2 * r2 * (gbest(t) - x(t)) \quad (19)$$

$$x(t{+}1) = x(t) + v(t{+}1) \quad (20)$$

where w is the inertia weight, $v(t)$ is the speed of the particles at present, $c1$ and $c2$ are coefficients that are positive constant variables, $r1$ and $r2$ are random variables between 0 and 1.

Algorithm 3 Service Composition via PSO

```
Require:
  - MAX_GEN: the maximum number of iterations.
  - NUM: the number of particles.
  - CHN: the aggregation of service class chains.
Ensure:
  - an approximately optimal service chain.
1: for looptime < MAX_GEN do
2:   for looptime < NUM do
3:     initialize particles.
4:   end for
5:   for looptime < |CHN| do
6:     calculate fitness function value.
7:     if the fitness function value is less than pBest then
8:       set the current fitness value as the new pBest.
9:     end if
10:  end for
11:  choose the particle with the minimum fitness value of
     all the particles as gBest.
12:  if the fitness function value is less than gBest then
13:    set the current fitness value as the new gBest.
14:  end if
15:  for looptime < NUM do
16:    calculate particle velocity according to formulas 19.
17:    update particle position according to formulas 20.
18:  end for
19:  find an approximately optimal service chain.
20: end for
```

Algorithm 3 shows the general service composition procedure involving particle swarm optimization. We initialize particles to set values for the start position and initial velocity (line 3). The fitness function values are calculated for all particles (line 6), and they are compared with *pbest*. If a fitness value is less than *pbest*, the fitness value is selected as the new *pbest* (line 8). Then, the particle that has the minimum fitness value is compared with *gBest*, and *gBest* is updated accordingly (line 13). The position and velocity are updated according to formulas 19 and 20 (line 16 and line 17). This procedure terminates when the maximum number of iterations has been reached.

TABLE 3. Sample for service classes specified at https://developer.android.com/guide/topics/sensors/sensors_overview.html.

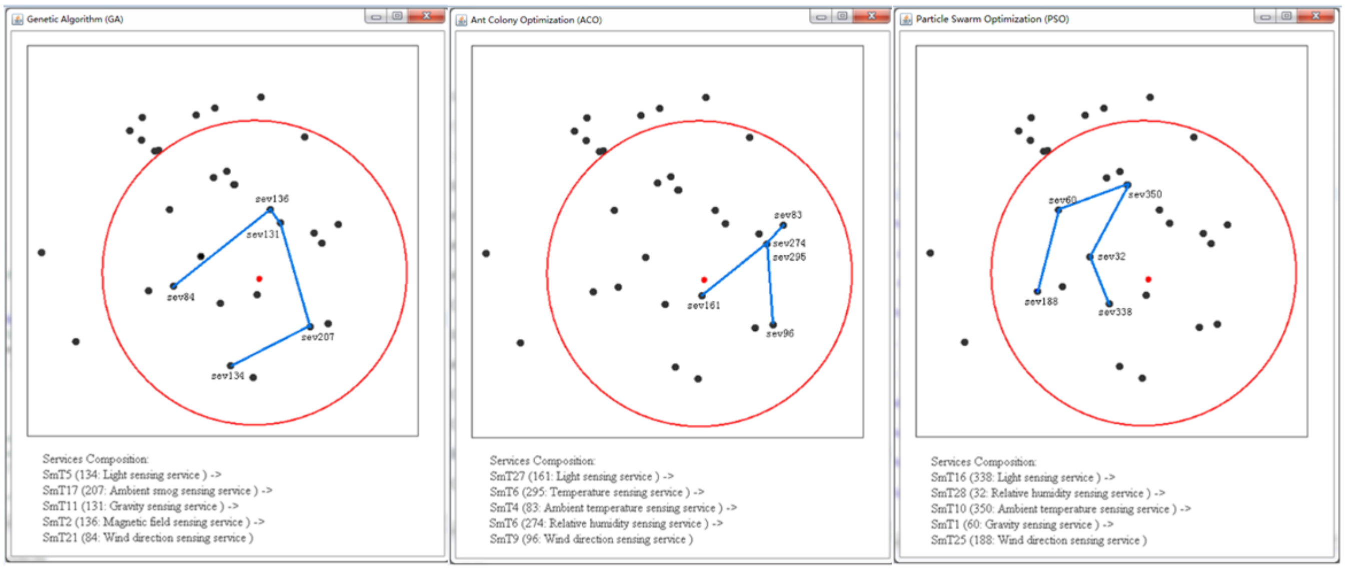
SC Id	*SC Name*
sc1	Ambient temperature sensing service
sc2	Temperature sensing service
sc3	Relative humidity sensing service
sc4	Light sensing service
sc5	Pressure sensing service
sc6	Gravity sensing service
sc7	Magnetic field sensing service
sc8	Proximity sensing service
sc9	Ambient smog sensing service
sc10	Wind direction sensing service
sc11	Wind power sensing service
sc12	Accelerometer sensing service
sc13	Rotation vector sensing service
sc14	Gyroscope sensing service

V. IMPLEMENTATION AND EVALUATION

The prototype has been implemented using a Java program, and experiments have been conducted on a desktop with an Intel i7-6700 CPU at 3.4GHz, 8-GB of memory and a 64-bit Windows 7 system. The experiment settings and evaluation results are presented in the following.

A. EXPERIMENT SETTINGS

The parameter settings for our experiments are presented in Table 2. Our experiments are set in a region with 30 smart things that co-host 400 IoT services. The geographical range is $500m \times 500m$, in which smart things are deployed with a skewness degree of 40%. Generally, a skewness degree (denoted sd) represents the distribution of unevenness for the distribution of smart things, and is computed using the following rule: $sd = (dn - sd) \div N$, where (i) dn is the number of smart things in dense subregions, (ii) sn is the number of



```
Services Composition:
SmT5 (134: Light sensing service ) ->
SmT17 (207: Ambient smog sensing service ) ->
SmT11 (131: Gravity sensing service ) ->
SmT2 (136: Magnetic field sensing service ) ->
SmT21 (84: Wind direction sensing service )
```

```
Services Composition:
SmT27 (161: Light sensing service ) ->
SmT6 (295: Temperature sensing service ) ->
SmT4 (83: Ambient temperature sensing service ) ->
SmT6 (274: Relative humidity sensing service ) ->
SmT9 (96: Wind direction sensing service )
```

```
Services Composition:
SmT16 (338: Light sensing service ) ->
SmT28 (32: Relative humidity sensing service ) ->
SmT10 (350: Ambient temperature sensing service ) ->
SmT1 (60: Gravity sensing service ) ->
SmT25 (188: Wind direction sensing service )
```

FIGURE 3. Sample IoT service compositions generated by applying GA, ACO, and PSO, respectively.

smart things in sparse subregions, and (iii) N is the sum of dn and sn [25]. As mentioned in Section II-A, we construct 14 service classes. Their specific information was presented in our previous work [15]. Here, we only list each one's identification and name as shown in Table 3. As shown in Fig. 2, the service network is constructed as a weighted directed graph, where the links, the weight of which is smaller than 0.511 (the prespecified threshold $thrd_{ip}$), are pruned.

Generally, a smart thing can co-host one or multiple IoT services, and IoT services with a certain kind of functionality can be hosted by multiple smart things. Without loss of generality, smart things are set to have the same amount of initial energy. The spatial constraint of a certain smart thing is set according to the geographical location and the communication radius, and the temporal constraint is set to contiguous time slots, where a time slot represents a 2-hour time duration in the 24-hour format. The conflict constraint for a certain IoT service in a certain smart thing is set in terms of the IoT services of continuity. The selection of IoT services is carried out according to the energy consumed when activating an instantiation of the IoT service and the proportion of energy consumed during the delay time and instantiation of the IoT service. The energy constraint is specified by the ratio of residual energy for smart things to the energy consumed for an IoT service in the smart things.

The parameters for the GA are set as follows: (i) the possibility of crossover $pc = 0.8$, and (ii) the possibility of mutation $pm = 0.1$. Crossover is an operation that is mostly used when generating new generations. On the other hand, mutation is to be applied randomly. The parameters for PSO are set as follows: (i) the acceleration coefficient for the velocity of particles $c1$ is set to 2, (ii) the acceleration coefficient for the position of particles $c2$ is set to 2, and (iii) the inertia weight factor w is set to 0.5, which shows the impact of previous values of particle velocities on the current values. The parameters for ACO are set as follows: (i) the pheromone volatilization factor rho is set to 0.05. The number of iterations for algorithms is set to 50, and it can be adjusted according to the particular experimental situation. The parameters for the fitness function are set as follows: $w1 = 0.2$, $w2 = 0.3$, $w3 = 0.5$, $\alpha = 0.4$, $\beta = 0.6$, $\varphi = 0.5$, and $\delta = 0.5$. Note that these parameters can be set to other appropriate values depending on the requirements of the particular domain application.

B. EVALUATION RESULTS

A sample of starting and ending states is shown in Section III. The geographical region in which a user is interested in is specified by the red circle shown in Fig. 3, where the black solid points represent the positions of smart things on which the IoT services are hosted. These smart things and the corresponding IoT services are not considered for the service composition when they cannot satisfy the temporal and spatial constraints of certain requirements. The GA, ACO, and PSO are applied to generate IoT service compositions.

As shown in Fig. 3, the recommended IoT service composition generated using the GA is $cp(chn_{ga}) = sev_{134}(SmT_{21}) \rightarrow sev_{207}(SmT_2) \rightarrow sev_{131}(SmT_{11}) \rightarrow sev_{136}(SmT_{17}) \rightarrow sev_{84}(SmT_5)$. Correspondingly, the service class chain is $chn_{ga} = sc4 \rightarrow sc1 \rightarrow sc6 \rightarrow sc7 \rightarrow sc10$, where the weight on average for $cp(chn_{ga})$ is 0.475560817. The recommended service composition generated via ACO is $cp(chn_{aco}) = sev_{161}(SmT_{27}) \rightarrow sev_{295}(SmT_6) \rightarrow sev_{83}(SmT_4) \rightarrow sev_{274}(SmT_6) \rightarrow sev_{96}(SmT_9)$. Correspondingly, the service class chain is $chn_{aco} = sc4 \rightarrow sc2 \rightarrow sc1 \rightarrow sc3 \rightarrow sc10$, where the weight on average for $cp(chn_{aco})$ is 0.501535008. A sample IoT service composition generated via PSO is $cp(chn_{pso}) = sev_{338}(SmT_{16}) \rightarrow sev_{32}(SmT_{28}) \rightarrow sev_{350}(SmT_{10}) \rightarrow sev_{60}(SmT_{1}) \rightarrow sev_{188}(SmT_{25})$. Correspondingly, the service class chain is $chn_{pso} = sc4 \rightarrow sc3 \rightarrow sc1 \rightarrow sc6 \rightarrow sc10$, where the weight on average for $cp(chn_{pso})$ is 0.520225996. These results show that PSO can generate an optimal IoT service composition in comparison with the GA and ACO. However, the difference is not much and is mainly caused by the contingency and randomness of these algorithms.

Fig. 4 shows fitness values for the GA, ACO, and PSO for 30 contiguous time slots. These values are used as the criteria for measuring the optimum of the corresponding IoT service compositions. This figure shows that the fitness values for PSO increase gradually and almost linearly, with a slope of approximately -0.6. The fitness values for the GA and ACO exhibit a slight ascending trend, but their contingency is larger. This fact indicates that under the current parameter settings, PSO should yield an appropriate IoT service composition in most situations. Actually, as the number of iterations increases, more and more energy is consumed by the smart things, and the proportion of energy consumed to instantiate a service as well as the residual energy is increased, which contributes to the increase in fitness values.

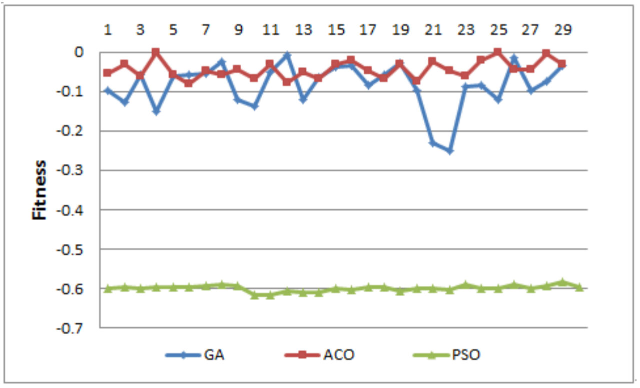


FIGURE 4. **The fitness value for the GA, ACO and PSO when the algorithms are executed in 30 contiguous time slots.**

Fig. 5 shows the minimum residual energy of smart things for 30 contiguous time slots. This figure shows that the curve for PSO is smoother and that the minimum residual energy for the GA and ACO decreases quickly. The initial energy of smart things is set to the same amount used in our experiment. The network lifetime is set to the time slots during which the first smart thing exhausts its energy. In this setting, a smart thing, with excessive energy consumption should not be selected to support a certain IoT service when a neighboring smart thing can provide the same IoT service and is relatively abundant in energy. This strategy is beneficial for balancing the energy consumption of smart things, thus prolonging the network lifetime. This figure shows that PSO outperforms the GA and ACO in preserving the minimum residual energy for smart things when selecting appropriate IoT services for instantiating service classes in chains.

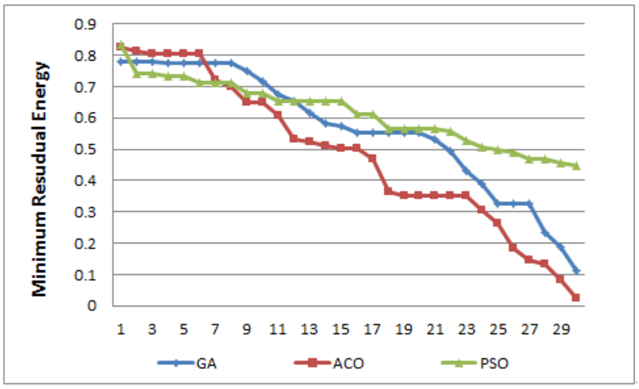


FIGURE 5. **The minimum residual energy of smart things when the algorithms are executed in 30 contiguous time slots.**

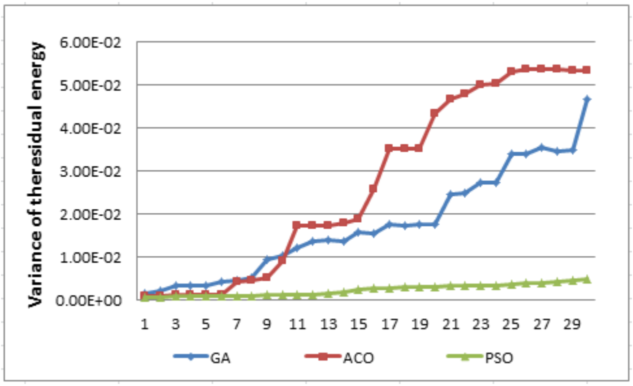


FIGURE 6. **The variance of the residual energy of smart things when the algorithms are executed in 30 contiguous time slots.**

Fig. 6 shows the variance of the residual energy of smart things for 30 contiguous time slots when the GA, ACO and PSO are adopted. Generally, this variance indicates the balancing of the energy consumption of smart things. The larger the value is, the more uneven the energy consumption of smart things is. This figure shows that PSO outperforms the GA and ACO in improving the balancing of the energy consumption, thus leading to a longer network lifetime.

VI. RELATED WORKS AND COMPARISON

Service composition techniques are well- developed and widely adopted for supporting relatively complex domain applications, where the requirement is beyond the capacity of any single smart thing. Generally, traditional techniques compose services in a certain order to achieve a certain goal [26]. Leveraging the semantic similarity of Web services for the construction of a service network, which represents the invocation possibilities between operations contained in Web services, a graph searching technique is adopted to discover service chains with respect to the determined initial and ending states. Note that the service composition technique proposed in [22] is adopted in this article to generate candidate service class chains. However, this technique mainly concentrates on the discovery and recommendation of appropriate service chains, while the relationship between services and sensing devices is not explored. The technique proposed in [27] is used to search for the compositions of IoT services in this article. A set of fundamental control-flow patterns was analyzed in the context of stateless compositions of REST services to support the composition of Web and REST services, while the characteristics of IoT services, including the spatial and temporal- constraints and the energy-awareness, are not explored extensively.

Service-oriented IoTs have been developed to support IoT service compositions. Service-Oriented Architecture (SOA) is a flexible architectural pattern used for system integration, and SOA-based IoT focuses on designing and developing IoT devices as services. Therefore, these IoT devices can be connected and composed via service composition techniques. This strategy facilitates the collaboration and cooperation of IoT devices. Cheng *et al.* [28] propose a situation-aware IoT service coordination platform based on the event-driven service-oriented architecture paradigm, which integrates the advantages of SOA effectively to support the coordination of IoT services. Ko *et al.* [18] propose the service-oriented IoT, as well as a user-centric IoT-based service framework. Generally, this framework integrates services that utilize IoT resources in an urban computing environment. The requirements of certain users can be fulfilled in terms of composing IoT services. During the service composition, the social, spatial, and temporal aspects in the environment are considered as constraints.

As a key pillar of IoT, in recent years, WSN sensor nodes or other intelligent equipment have been encapsulated in terms of services to facilitate the functional integration and collaboration of these devices. Shah *et al.* [29] propose a service-oriented system for WSNs, which is capable of practicing service configuration under various geo-spatial and resource constraints. A service has a spatial relevance with a certain region, which is overlapped by the region of interest prescribed by the requirement. The configuration mechanism should reconfigure the service according to the requirement. The impact of the geo-spatial and cost constraints of sensor services is considered, whereas temporal constraints, energy efficiency and conflict constraints, which are the main concerns of our technique, are not considered. Generally, this technique complements our technique, as it provides a more appropriate approach to WSN service composition. Zhou *et al.* [15] propose a three-tier serviced-oriented framework, where each sensor node is encapsulated as a WSN service, which is classified into a service class according to its functionality. Service classes are chained to fulfill the requirement, and the instantiation of service compositions should consider spatial and temporal constraints, and energy-efficiency. As the foundation of the technique developed in this article, IoT smart things are encapsulated and represented as the aggregation of IoT services, and each smart thing contains multiple IoT services. Therefore, in the process of instantiating service compositions, the conflict constraints and load-balanced energy consumption of smart things are also the constraints to be considered. To balance the disparity between the high-level requirements and the low-level equipment in an IoT environment, different middleware architectures have been proposed in WSN and IoT in recent years [16]. Issarny *et al.* [30], the authors explore the mechanism for adopting the service-oriented architecture to address challenges posed by the IoT for the development of distributed applications. The evolution of the supporting middleware solutions, including the probabilistic protocols used to address the scale, the cross-paradigm interaction used to address the heterogeneity, and streaming-based interactions used to support the inherent sensing functionality, have been discussed.

The service-oriented paradigm has been applied in a variety of IoT contexts. In [17], trust management for supporting service composition applications in a SOA-based IoT system is proposed. Heterogeneous IoT devices in physical networks are virtually connected via social networks. Direct and indirect trusts are dynamically combined to minimize the convergence time and trust estimation bias to perform opportunistic services. Formulating the service distribution problem in IoT-cloud networks mathematically, a minimum cost mixed-cast flow problem is proposed in [20], which can be efficiently solved via linear programming. A flexible mathematical model was introduced for IoT-cloud networks. This model characterizes the capacity, efficiency, and reliability of sensing, computing, and transmission resources across end devices, access, and cloud layers. A generic IoT service is characterized, and it encodes the relationship between service functions that must be delivered to users. A smart IoT communication system manager design, which is used as a low-cost irrigation controller, is proposed in [31]. This work presents the design and development of a multimedia platform for precision agriculture and integrates an intelligent IoT irrigation system based on a mesh network. The aerial mapping sensor included in AR Drones with HD cameras is used to monitor an area used to grow crops. In [32], an architecture that supports Web objects based on energy-efficient for smart home IoT services is proposed, which minimizes the energy consumption while satisfying user living comfort. Generally, the integration of IoT devices is mostly achieved via the service-oriented mechanism, where spatial and temporal- constraints, energy efficiency, and the configurability of IoT services should be considered as main concerns.

VII. CONCLUSIONS

Along with the wide adaptation of IoT in various industrial applications, the integration of IoT smart things for facilitating the cooperation and coordination of these smart things is a challenge, especially when the requirement to be achieved is beyond the capacity of any single smart thing. In this setting, service-oriented IoT is proposed, where the functionalities provided by smart things are encapsulated into IoT services, which are composed into value-added services when necessary. Intuitively, a smart thing can co-host multiple IoT services with different functionalities. On the other hand, a certain functionality, which can be represented by a service class, can be supported by one or multiple smart things. To address the challenge of IoT service composition, service class chains are generated with respect to the requirement by adopting traditional Web service composition techniques. Considering the factors, including spatial and temporal- constraints, energy efficiency, and the configurability of IoT services, IoT service selection for the instantiation of service classes contained in chains is reduced to a multiobjective and multiconstrained optimization problem. Heuristic algorithms, such as the genetic algorithm (GA), ant colony optimization (ACO) and particle swarm optimization (PSO), are adopted to search for optimal IoT service compositions. The experimental evaluation mainly considers the fitness, minimum residual energy of smart things, and the variance of residual energy in smart things. The results show that PSO performs better than the GA and ACO in searching for approximately optimal IoT service compositions and reduces and balances the energy consumption, thus prolonging the network lifetime.

REFERENCES

[1] H. Wang, D. Xiong, P. Wang, and Y. Liu, "A lightweight XMPP publish/subscribe scheme for resource-constrained IoT devices," *IEEE Access*, vol. 5, pp. 16393–16405, 2017.

[2] T. Qiu, R. Qiao, and D. O. Wu, "Eabs: An event-aware backpressure scheduling scheme for emergency Internet of Things," *IEEE Trans. Mobile Comput.*, to be published, doi:10.1109/TMC.2017.2702670.

[3] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: A survey on recent developments and potential synergies," *J. Supercomput.*, vol. 68, no. 1, pp. 1–48, 2014.

[4] S. Cheng, Z. Cai, J. Li, and H. Gao, "Extracting kernel dataset from big sensory data in wireless sensor networks," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 4, pp. 813–827, Apr. 2017.

[5] T. Mekonnen, P. Porambage, E. Harjula, and M. Ylianttila, "Energy consumption analysis of high quality multi-tier wireless multimedia sensor network," *IEEE Access*, vol. 5, pp. 15848–15858, 2017.

[6] S. Wang, A. Zhou, M. Yang, L. Sun, C.-H. Hsu, and F. Yang, "Service composition in cyber-physical-social systems," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2017.2675479.

[7] R. Helal and A. ElMougy, "An energy-efficient service discovery protocol for the IoT based on a multi-tier WSN architecture," in *Proc. IEEE 40th Local Comput. Netw. Conf. Workshops*, Oct. 2015, pp. 862–869.

[8] A. Ali, G. A. Shah, and J. Arshad, "Energy efficient techniques for M2M communication: A survey," *J. Netw. Comput. Appl.*, vol. 68, pp. 42–55, Jun. 2016.

[9] Z. Meng, Z. Wu, C. Muvianto, and J. Gray, "A data-oriented M2M messaging mechanism for industrial IoT applications," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 236–246, Feb. 2017.

[10] M. Garriga, C. Mateos, A. Flores, A. Cechich, and A. Zunino, "Restful service composition at a glance: A survey," *J. Netw. Comput. Appl.*, vol. 60, pp. 32–53, Jan. 2016.

[11] H. Naim, M. Aznag, M. Quafafou, and N. Durand, "Probabilistic approach for diversifying Web services discovery and composition," in *Proc. IEEE Int. Conf. Web Services*, Jun./Jul. 2016, pp. 73–80.

[12] R. Bawa and R. Clark, "Software-defined everything—Breaking virtualization's final frontier," Deloitte, New York, NY, USA, Tech. Rep. 75-90, 2015.

[13] A. Malik, B. Aziz, M. Adda, and C.-H. Ke, "Optimisation methods for fast restoration of software-defined networks," *IEEE Access*, vol. 5, pp. 16111–16123, 2017.

[14] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for Internet of Things services," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 16–24, Mar./Apr. 2017.

[15] Z. Zhou, D. Zhao, L. Liu, and P. C. K. Hung, "Energy-aware composition for wireless sensor networks as a service," *Future Generat. Comput. Syst.*, to be published, doi: 10.1016/j.future.2017.02.050.

[16] R. Morabito, I. Farris, A. Iera, and T. Taleb, "Evaluating performance of containerized IoT services for clustered devices at the network edge," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 1019–1030, Aug. 2017.

[17] I.-R. Chen, J. Guo, and F. Bao, "Trust management for SOA-based IoT and its application to service composition," *IEEE Trans. Serv. Comput.*, vol. 9, no. 3, pp. 482–495, May/Jun. 2016.

[18] I.-Y. Ko, H.-G. Ko, A. J. Molina, and J.-H. Kwon, "SoIoT: Toward a user-centric IoT-based service framework," *ACM Trans. Internet Technol.*, vol. 16, no. 2, p. 8, 2016.

[19] E. Karmouch and A. Nayak, "Capability reconciliation for a CSP approach to virtual device composition," *IEEE/ACM Trans. Netw.*, to be published, doi: 10.1109/TNET.2012.2190296.

[20] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "IoT-cloud service optimization in next generation smart environments," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 4077–4090, Dec. 2016.

[21] S. N. Han, S. Park, G. M. Lee, and N. Crespi, "Extending the devices profile for Web services standard using a rest proxy," *IEEE Internet Comput.*, vol. 19, no. 1, pp. 10–17, Jan./Feb. 2015.

[22] Z. Zhou, Z. Cheng, K. Ning, W. Li, and L.-J. Zhang, "A sub-chain ranking and recommendation mechanism for facilitating geospatial Web service composition," *Int. J. Web Services Res.*, vol. 11, no. 3, pp. 52–75, 2014.

[23] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proc. 33rd Annu. Hawaii Int. Conf. Syst. Sci.*, 2000, pp. 1–10.

[24] S. C. Geyik, B. K. Szymanski, and P. Zerfos, "Robust dynamic service composition in sensor networks," *IEEE Trans. Serv. Comput.*, vol. 6, no. 4, pp. 560–572, Oct./Dec. 2013.

[25] Z.-B. Zhou, D. Zhao, L. Shu, and H.-C. Chao, "Efficient multi-attribute query processing in heterogeneous wireless sensor networks," *J. Internet Technol.*, vol. 15, no. 5, pp. 699–712, 2014.

[26] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, and R. Buyya, "An energy-aware service composition algorithm for multiple cloud-based IoT applications," *J. Netw. Comput. Appl.*, vol. 89, pp. 96–108, Jul. 2017.

[27] J. Bellido, R. Alarcón, and C. Pautasso, "Control-flow patterns for decentralized restful service composition," *ACM Trans. Web*, vol. 8, no. 1, p. 5, 2013.

[28] B. Cheng, D. Zhu, S. Zhao, and J. Chen, "Situation-aware IoT service coordination using the event-driven SOA paradigm," *IEEE Trans. Netw. Serv. Manag.*, vol. 13, no. 2, pp. 349–361, Jun. 2016.

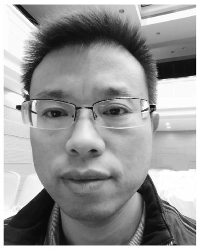
[29] S. Y. Shah, B. K. Szymanski, P. Zerfos, and C. Gibson, "Towards relevancy aware service oriented systems in WSNs," *IEEE Trans. Serv. Comput.*, vol. 9, no. 2, pp. 304–316, Mar./Apr. 2016.

[30] V. Issarny, G. Bouloukakis, N. Georgantas, and B. Billet, "Revisiting service-oriented architecture for the IoT: A middleware perspective," in *Proc. 14th Int. Conf. Service-Oriented Comput.*, 2016, pp. 3–17.

[31] C. Cambra, S. Sendra, J. Lloret, and L. Garcia, "An IoT service-oriented system for agriculture monitoring," in *Proc. IEEE Int. Conf. Commun.*, May 2017, pp. 1–6.

[32] M. G. Kibria, M. A. Jarwar, S. Ali, S. Kumar, and I. Chong, "Web objects based energy efficiency for smart home IoT service provisioning," in *Proc. 9th Int. Conf. Ubiquitous Future Netw.*, Jul. 2017, pp. 55–60.

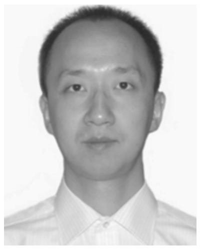
MENGYU SUN is currently pursuing the master's degree with the School of Information Engineering, China University of Geosciences, Beijing. Her research interests include wireless sensor networks.



ZHENSHENG SHI is currently a Senior Engineer with the Research Institute of Petroleum Exploration and Development, CNPC, China. His research interests include wireless sensor networks, sedimentology, sequence stratigraphy, and reservoir geology.

SHENGJUN CHEN is currently an Associate Professor with the Business School, University of International Business and Economics. His research interests include wireless sensor networks and human resource management.

ZHANGBING ZHOU is currently a Professor with the China University of Geosciences, Beijing, China, and an Adjunct Associate Professor with the Telecom SudParis, France. His research interests include wireless sensor networks, services computing, and business process management.



YUCONG DUAN is currently a Professor with the College of Information Science and Technology, Hainan University. His research interests include wireless sensor networks.