Energy-efficient IoT service composition for concurrent timed applications

Mengyu Sun, Zhangbing Zhou, Junping Wang, Chu Du, Walid Gaaloul

Version of Record: https://www.sciencedirect.com/science/article/pii/S0167739X19302274
Manuscript_a4a61bebb5f4bb51558455c1dd4dc98e

Energy-Efficient IoT Service Composition for Concurrent Timed Applications

Mengyu Sun[a], Zhangbing Zhou[a,d,*], Junping Wang[b], Chu Du[c], Walid Gaaloul[d]

[a] *School of Information Engineering, China University of Geosciences (Beijing), China*
[b] *Laboratory of Precision Sensing and Control Center, Institute of Automation, Chinese Academy of Sciences, Beijing, China*
[c] *The 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, China*
[d] *Computer Science Department, TELECOM SudParis, Evry, France*

Abstract

The *I*nternet *o*f *T*hings (*IoT*) paradigm has established an efficient platform to enable the collaboration and cooperation of self-configurable and energy-aware *IoT* nodes for supporting complex applications. Heterogeneous *IoT* nodes provide various kinds of functionalities, which can be encapsulated and represented as *IoT* services. These services can be composed to provide value-added services, while spatial-temporal constraints of *IoT* services should be satisfied, and energy consumption of *IoT* nodes should be balanced to prolong the network lifetime. Given a set of concurrent service requests, a challenge is to recommend efficient service compositions.

To address this challenge, this paper proposes to identify and share common functional components, and thus, to integrate and optimize concurrent requests, where a component corresponds to a snippet of *IoT* service compositions. Shared components in different requests should not violate their temporal dependencies and thus improving resource utilization. Consequently, composing *IoT* services with respect to concurrent requests can be reduced to a constrained multi-objective optimization problem, which can be solved by heuristic algorithms. Experimental evaluation has been performed with respect to the state-of-art's algorithms, and the results demonstrate the efficiency and performance of this technique, especially when *IoT* nodes are relatively large in number and their functionalities are highly overlapped with each other.

Keywords: *IoT* Service Composition; Temporal Constraints; Concurrent Requests; Energy Efficiency.

1. Introduction

The *I*nternet *o*f *T*hings (*IoT*) has extensively developed and widely-adopted in domain applications, and promotes the interconnection of distributed devices with sensing capabilities. These connected devices, also known as *IoT* nodes, achieve their collaboration and cooperation through integrating the functionalities of contiguous *IoT* nodes. Along with this trend, large-scale *IoT* sensing networks have been emerging, and they have become the infrastructure to support applications in various domains [1, 2]. Similar as sensor nodes in *W*ireless *S*ensor *N*etworks (*WSN*), *IoT* nodes are mostly battery-powered, and their energy may hardly be replenished in most environments, although energy harvesting techniques [3, 4] have been developed and could alleviate the energy consumption somehow in certain situations. In this setting, energy efficiency is an essential ingredient to be considered, especially when concurrent applications is relatively large in number.

With increasing complexity of requirements of domain applications, an application, also known as a service request, should be achieved through assembling the functionalities provided by multiple *IoT* nodes, which may exceed the capability of any single *IoT* node. To promote this procedure leveraging *S*ervice-*O*riented *A*rchitecture (*SOA*) [5], the functionalities provided by *IoT* nodes are encapsulated and represented in terms of respective *IoT* services, where an *IoT* node may co-host one or multiple *IoT* services [6]. Consequently, the task of assembling the functionalities of *IoT* nodes is reduced to the composition of corresponding *IoT* services. Besides functional requirements, temporal constraints of functional components are usually specified upon and mandated to be satisfied in real-world applications. Hence the temporal consistency should be examined when composing timed *IoT* services. It is worth emphasizing that an *IoT* sensing network should support concurrent applications, Intuitively, these applications can be processed independently. However, considering the fact that there may exist common components in concurrent applications, which may be processed once to satisfy some sub-requirements in multiple applications. Therefore, the strategy that common components used in a shared manner among concurrent service requests should be appropriate to promote the decreasing of energy consumption of *IoT* nodes and the network traffic. For instance, concurrent processing technology with temporal constraints is applied in distributed real-time monitoring

* *Email address:* zhangbing.zhou@gmail.com (Zhangbing Zhou)
URL: sunmengyu.cugb@gmail.com (Mengyu Sun), wangjunping@bupt.edu.cn (Junping Wang), duchu2017@126.com (Chu Du), walid.gaaloul@mines-telecom.fr (Walid Gaaloul)

systems and information audit systems by an integration and assignment of tasks to optimize system structure and improve resource utilization.

In this setting, composing timed *IoT* services which share common components to support concurrent requests is a promising research challenge. As a specific example of *IoT* sensing networks, *WSN* can be shared by sensor middlewares or architectures to ensure the network which can support multi-application simultaneously [7, 8], wherein sensor nodes and network resources are shared to provide value-added services for responding to concurrent applications. Data sharing techniques have been developed and applied to support the processing of multiple applications through integrating sampling temporal intervals by reasonably planning tasks [9]. Generally, these techniques mostly target at the concurrent-execution of multiple applications. From the temporal aspect, task allocation with overlapping temporal intervals is performed in order to promote the sharing of common sub-requirements in concurrent applications. However, temporal constraints for seperate functional components should be explored extensively to guarantee temporal consistency for optimizing *IoT* service compositions.

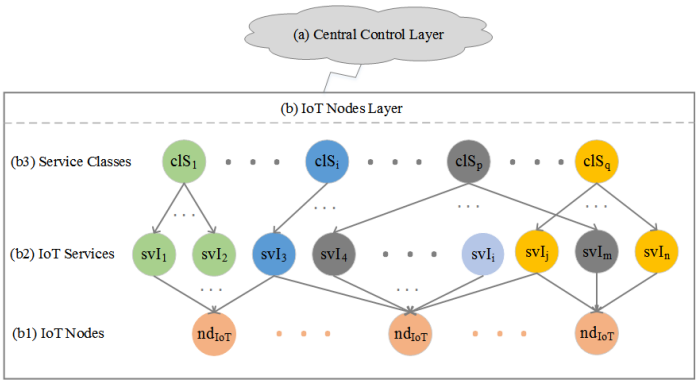

Figure 1: The framework of *IoT* sensing networks includes a Central Control Layer and an *IoT* Nodes Layer, where the *IoT* Nodes Layer is a three-tier subframe including the tiers of *IoT* Nodes, *IoT* Services, and Service Classes.

To remedy this issue, this paper proposes a framework for *IoT* sensing networks as illustrated by Figure 1, and develops a *C*oncurrent *R*equests *I*ntegration *O*ptimization (*CRIO*) mechanism to specify temporal dependencies between connected components and thus to promote the optimization model for concurrent requests. The contributions of this paper are summarized as follows:

- *IoT* nodes accompanying with multiple kinds of functionalities are virtualized in terms of service providers, where these functionalities are encapsulated and represented as respective *IoT* services.
- A service request is achieved by implementing functionalities sequentially, and temporal dependencies are examined to ensure their consistency.
- The *CRIO* mechanism is proposed to optimize shared functionalities among concurrent requests.

Extensive experiments are conducted for evaluating the feasibility and performance of our *CRIO* mechanism, and exploring the influence of various experimental parameters upon this technique. The result shows that our technique performs better than the rivals in the energy efficiency, especially when *IoT* nodes are relatively large in number, and they are largely overlapped in their functionalities.

The rest of this paper is organized as follows. Section 2 introduces correlation definitions and energy model. Section 3 introduces the *CRIO* mechanism for concurrent service requests. Section 4 presents *IoT* service composition for optimal solution which mainly concerns about spatial constraints, temporal relevancy and energy constraints. Experimental results are presented at Section 5 and Section 6 reviews and compares our approach with relevant works. Section 7 concludes our work.

2. Preliminaries

This section presents relevant concepts and the energy model for supporting the composition of concurrent application requests in the following sections.

2.1. Concept Definition

Definition 1 (*IoT* Node). *An IoT Node nd_{IoT} is a tuple (id, SvI, spt, eng), where:*

- *id is the unique identifier of this IoT node nd_{IoT}.*
- *SvI is a set of IoT services co-hosted by nd_{IoT}.*
- *spt is the spatial constraint specified upon nd_{IoT}.*
- *eng is the remaining energy of nd_{IoT}.*

As shown in Figure 1-(b1), an *IoT* node nd_{IoT} may co-host multiple *IoT* services. Due to the scarceness of computational and energy resources, most *IoT* nodes could hardly activate multiple *IoT* services concurrently. Note that *IoT* nodes are assumed to have no temporal constraints in this paper, since *IoT* nodes, which are deployed for supporting domain applications like environmental monitoring and event detection, may be in the status of *active* during their lifetime. Without loss of generality and for simplicity, we assume that an *IoT* node can support the operation of only one *IoT* service at a certain time duration. *spt* is defined by the physical attribution in terms of a circular region specified by a centre and a communication radius.

Definition 2 (*IoT* Service). *An IoT Service svI hosted by an IoT node nd_{IoT} is a tuple (id, nm, dsc, nd_{IoT}), where:*

- *id is the unique identifier of svI.*
- *nm is the name of svI.*
- *dsc is the short-text description of svI.*
- *nd^{i}_{IoT} represents the set of IoT nodes which host svI.*

Note that *IoT* services and *IoT* nodes may have a many-to-many relationship. To be specific, a certain kind of *IoT* services may be hosted by one or multiple *IoT* nodes, and a certain *IoT* node can host one or multiple kinds of *IoT* services. For instance, an *IoT* service *svI* with an *id* as 325 as shown in Table 3 is represented as follows:

- $svI.nm$: *"Relative humidity monitor sensing service"*
- $svI.dsc$: *"Measures the relative ambient humidity in percent (%). Common use is for monitoring dew-point, absolute, and relative humidity."*
- $svI.nd^i_{IoT}$: nd^1_{IoT}

As shown by Figure 1-(b2), multiple *IoT* services hosted by various *IoT* nodes may be similar in their functionality. The concept of service class is adopted to represent these *IoT* services sharing a certain functionality [6].

Definition 3 (Service Class). *A service class clS is a tuple (id, nm, dsc, SvI_{clS}), where id, nm and dsc are the identifier, name, and text description of clS, and SvI_{clS} is a set of IoT services belonging to clS and hosted by different IoT nodes.*

As shown in Figure 1-(b3), a service class takes the functional aspect of *IoT* services into concern, while their spatial constraints and energy consumption are not the focuses. Consequently, a service request is defined as follows:

Definition 4 (Service Request). *A service request rq is a tuple (id, CLS, LNK_{TpCt}, SpCt), where:*

- *id is the unique identifier of this service request rq.*
- *CLS is a set of preferring service classes according to the requirement.*
- *LNK_{TpCt} is a set of temporal links between adjacent service classes.*
- *SpCt is the spatial constraints specified upon rq.*

Generally, a service request should prescribe the preferring composition of service classes, and this composition can be discovered and generated through traditional Web service composition techniques [10, 11]. These service classes are linked according to their functional dependency relations [12]. Temporal constraints denoted as the tuple (T, C) are specified in terms of the rules in *S*imple *T*emporal *N*etworks (*STN*) [13], where (i) T is a set of time point variables, which correspond to the completion time of service classes, and (ii) C is a finite set of binary constraints specified upon these time point variables. The completion time of service classes as temporal objects are constrained by temporal statements such as points and intervals. We adopt the interval algebra [14] to define qualitative statements for relative temporal constraints of adjacent service classes, which is denoted as [T_1, T_2] to represent the alternative time interval of this service class with respect to the pervious one. To avoid the issue of cyclic temporal dependencies between service classes, a service request is assumed not containing cyclic structures. For a given service request, a reasonable solution is a complete set of *IoT* services assignments that satisfies all constraints.

2.2. Energy Model

The first order radio model proposed at [15] is adopted to calculate the energy consumption in *WSN*s. Since *IoT* nodes can be regarded as a certain kind of sensor nodes in *WSN*, we calibrate the first order radio model to support the *IoT* sensing network, which enhances the model for adapting to the architecture. The energy consumption for transmitting and receiving a k bit data packet within the distance d are denoted as $E_{Tx}(k,d)$ and $E_{Rx}(k)$ respectively, and the formulae are presented as follows:

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

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

where E_{elec} is the energy consumption constant of the transmission and receiver electronics, ϵ_{amp} is the energy consumption constant of the transmission amplifier, and the parameter p refers to the attenuation index of transmission, which is influenced by surrounding environments. Generally, if *IoT* nodes are barrier-free when forwarding data packets, p is set to 2. Otherwise, p is set to a value between 3 to 5. Therefore, the energy consumed when transmitting a packet of k bits data packet from an *IoT* node nd^i_{IoT} to one of its neighbours nd^j_{IoT} (denoted as $E_{ij}(k)$), where the formula is presented as follows:

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

Note that *B*ackbone *N*odes (denoted *BN*) in *IoT* sensing networks are assumed to have unlimited energy, since energy harvesting techniques can be adopted to replenish their energy. The cost of transmitting a data packet from an *IoT* node to a neighboring *IoT* node or to a *BN* is different. Besides, the cost of receiving a data packet by *BN* is not taken into consideration. Therefore, $E_{ij}(k)$ is calculated as follows:

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

Assume that the energy consumed to transmit a data packet from nd^i_{IoT} to nd^j_{IoT} is the same as that needed to transmit from nd^j_{IoT} to nd^i_{IoT}. Parameters in the energy model we used are shown in Table 1.

3. Concurrent Service Requests Integration

Given a set of concurrent service requests, there exists a set of common components in terms of functionality perspective. Except for that, the spatial and temporal factors are conditions that cannot be ignored for the integration in the concurrent fashion. In this setting, components with

Table 1: Parameters in the energy model

Parameters Name	Description
$E_{Tx}(k,d)$	The energy consumed to transmit a k bit data packet within a distance d.
$E_{Rx}(k)$	The energy consumed to receive a k bit data packet.
$E_{ij}(k)$	Energy consumption for transmitting a k bit data packet from an *IoT* node nd^i_{IoT} to a neighboring *IoT* node nd^j_{IoT}.
$E_{inv}(nd^i_{IoT}.svI_m)$	Energy consumption for activating an instantiation of an *IoT* service svI_m in an *IoT* node nd^i_{IoT}.
E_{elec}	Energy consumption constant of electronics.
ϵ_{amp}	Energy consumption constant of the transmit amplifier.
k	The number of bits in one data packet.
d	The transmission distance of two nodes.
p	The attenuation index of transmissions.
r	The communication radius of *IoT* nodes.

common functionalities can be shared which have compatible spatial regions and temporal intervals. The satisfaction of relevant service requests can be guaranteed by arranging reasonable schedule that do not violate any constraints. Sample concurrent service requests are shown as Figure 2, and they have the same initial time point but not necessarily the same ending time. In the specification level, each component is represented as a service class and temporal dependencies between these components are specified by directed timed edges that denote the temporal constraints between the completion of the pervious component and the latter one. Additional temporal constraints on each edge are represented as a relative temporal interval which are transformed into absolute temporal constraints based on links between service classes. Consequently, service compositions may be optimized through considering them in an integrated fashion, in order to reduce the energy consumption collectively. To solve this problem, our *CRIO* mechanism is presented by Algorithm 1 to recommend *IoT* services for concurrent service requests.

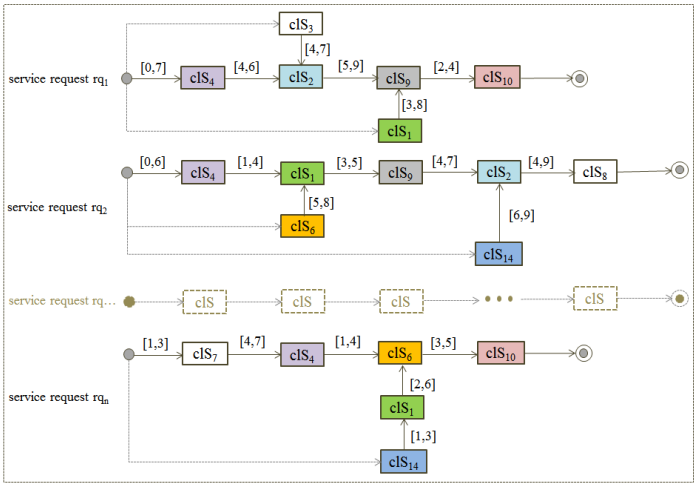

Figure 2: A sample of serval concurrent service requests, which have specific functional components and concrete temporal constraints, where several kinds of functional components are distinguished by different colors.

RQ_{con} in Algorithm 1 represents a set of concurrent service requests, CLS_{med} and RQ_{med} are the intermediate sets that store the common service classes and service

Algorithm 1 CRIO: Concurrent Requests Integration Optimization Mechanism

Require:
- RQ_{con} : concurrent service requests with same initial time

Ensure:
- CLS_{con} : a set of service classes shared by different requests
- RQ_{cls} : a set of service requests with shared service classes

1: $CLS_{med} \leftarrow \varnothing$; $RQ_{med} \leftarrow \varnothing$; $CLS_{con} \leftarrow \varnothing$; $RQ_{cls} \leftarrow \varnothing$
2: **for all** rq_u, $rq_v \in RQ_{con}$ where $u, v \in [1, |RQ_{con}|]$ **do**
3: **if** $\exists\ clS_p$ where $clS_p \in rq_u.CLS$ **and** $clS_p \in rq_v.CLS$ **then**
4: $CLS_{med} \leftarrow CLS_{med} \cup \{clS_p\}$
5: **end if**
6: **end for**
7: **for all** $clS_p \in CLS_{med}$ where $p \in [1, |CLS_{med}|]$ **do**
8: **if** $clS_p \in rq_u.CLS$ where $i \in [1, |rq_u.CLS|$ **then**
9: $RQ_{med} \leftarrow RQ_{med} \cup \{rq_u\}$
10: **end if**
11: **if** $\exists\ rq_u$ **and** $rq_v \in RQ_{med}$ where $u, v \in [1, |RQ_{med}|]$ **then**
12: $Intv(rq_u.CLS(clS_p)) \leftarrow$ ConvertAbsoluteInterval(rq_u, clS_p)
13: $Intv(rq_v.CLS(clS_p)) \leftarrow$ ConvertAbsoluteInterval(rq_v, clS_p)
14: **if** $Intv(rq_u.CLS(clS_p)) \cap Intv(rq_v.CLS(clS_p)) \neq \varnothing$ **then**
15: $CLS_{con} \leftarrow CLS_{con} \cup \{clS_p\}$
16: $RQ_{cls} \leftarrow RQ_{cls} \cup \{rq_u, rq_v\}$
17: **end if**
18: **end if**
19: **end for**

requests to which they belong, respectively. CLS_{con} represents service classes which can be shared by different service requests RQ_{cls}. For an arbitrary service request contained in RQ_{con}, if there exists a service class clS_p which is a common component of any two different service requests rq_u and rq_v, clS_p is inserted into CLS_{med} as a candidate service class (line 4).

All shared service classes candidates are found which can be used to retrieve the corresponding set of service requests RQ_{med} (line 9). For each clS_p in CLS_{med}, the absolute temporal intervals for rq_u and rq_v are calculated through Algorithm 2 (lines 12-13). If the available temporal interval of a common service class clS_p in different rq_u and rq_v has a coincident duration, clS_p is verified which can be shared by rq_u and rq_v to optimize energy consumption (line 14). clS_p is inserted into CLS_{con} (line 15), besides corresponding rq_u and rq_v are inserted into RQ_{cls} (line 16), which are used in Section 4. For instance, as shown in Figure 2, all colored service classes are stored

into CLS_{med} and these corresponding service requests to which they belong are deposited into RQ_{med}. Take clS_4 as an example, clS_4 appears in over one service requests, which can be inserted as an element to the candidate set CLS_{med} and those service requests to which it belongs are stored in RQ_{med}. The validity of elements in candidate sets are verified by temporal constraints. The absolute intervals for rq_1 and rq_2 are [0, 7] and [0, 6] respectively. As for clS_4 in rq_n, the absolute temporal interval of clS_4 is affected by its previous service class clS_7, which can be explicated through Algorithm 2. There is a common overlap time duration between these three service requests for a certain clS_4, so clS_4 is one of concurrent service classes in CLS_{con} and rq_1, rq_2 and rq_n are the corresponding service requests confirmed in RQ_{cls}.

Algorithm 2 ConvertAbsoluteInterval

Require:
- RQ_{med} : service requests corresponding to common service classes generated by Algorithm 1
- CLS_{med} : common service classes generated by Algorithm 1

Ensure:
- $Intv(rq_u.CLS(clS_p))$: the absolute temporal interval for a certain clS_p in a certain rq_u

1: $\widetilde{clS_0} \leftarrow$ get the virtual initial vertex
2: $t_0 \leftarrow$ get the initial time point
3: **for** $rq_u \in RQ_{med}$ where $u \in [1, |RQ_{med}|]$ **do**
4: **while** $\exists\ clS_p \in CLS_{med}$ where $p \in [1, |CLS_{med}|]$ **do**
5: $LNK_{tmp}\{\{\widetilde{clS_0}, \ldots, clS_p\}, \ldots, \{\{\widetilde{clS_0}, \ldots, clS_p\}\} \leftarrow$ extract all links connecting $\widetilde{clS_0}$ to clS_p
6: **for** $lnk_i \in LNK_{tmp}$ where $i = |LNK_{tmp}|$ **do**
7: $Intv_i \leftarrow$ get absolute temporal intervals related to the initial time point t_0 of $\widetilde{clS_0}$
8: $INTV_{tmp} \leftarrow INTV_{tmp} \cup \{Intv_i\}$
9: **end for**
10: $INTV_{tmp}\{Intv_1, \ldots, Intv_{|LNK_{tmp}|}\} \leftarrow$ get all absolute temporal intervals
11: **for all** $Intv_i \in INTV_{tmp}$ where $i = |INTV_{tmp}|$ **do**
12: $Intv(rq_u.CLS(clS_p)) \leftarrow Intv_1 \cap \cdots \cap Intv_i$
13: **end for**
14: **end while**
15: **end for**

Algorithm 2 is presented to convert temporal intervals related to adjacent structure into absolute temporal intervals. A virtual initial vertex $\widetilde{clS_0}$ is generated to connect these service classes without preorder vertexes (line 1), and the initial time point t_0 is obtained (line 2). We extract all links LNK_{tmp} from $\widetilde{clS_0}$ to each clS_p in CLS_{med} for different service requests (line 5). At least one connection path is found in each rq_u for each clS_p. Absolute temporal intervals are acquired according to relative intervals on corresponding connection paths (line 7). Therefore, $INTV_{tmp}$ is the set containing possible absolute temporal intervals relative to t_0 calculated according to the specification of service requests, and its corresponding relative temporal intervals based on adjacent service classes (line 10). An absolute temporal interval $Intv(rq_u.CLS(clS_p))$ for a certain clS_p in rq_u is acquired through adopting the intersection of each $Intv_i$ in $INTV_{tmp}$ (line 13). As shown in Figure 2 and taking clS_6 in rq_n for an example, where two links included in $INTV_{tmp}$ consist of $\{\widetilde{clS_0}, clS_7, clS_4, clS_6\}$ and $\{\widetilde{clS_0}, clS_{14}, clS_1, clS_6\}$. According to the link of $\{clS_0, clS_7, clS_4, clS_6\}$, the absolute temporal interval is [6, 14] based on mathematical operation of lower bound and upper bound of preorder vertexes. Thereafter, the absolute temporal interval for $\{\widetilde{clS_0}, clS_{14}, clS_1, clS_6\}$ is set to [3, 9]. Hence, the definitized absolute temporal interval of $Intv(rq_n.CLS(clS_4))$ is set to [6, 9] which is the intersection of two intervals as mentioned above.

4. IoT Service Composition

This section presents *IoT* service composition technique for concurrent service requests and recommends optimal solutions, where spatial constraints, temporal relevancy and energy efficiency are taken into consideration.

4.1. Constraints of IoT Service Composition

4.1.1. Spatial Constraints

IoT nodes should be prescribed spatial constraints since they can only work well within a certain range of areas. Generally, the spatial constraint of an *IoT* node nd_{IoT} and a certain service request rq are specified according to their geographical positions and communication radius, which can be represented as follows:

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

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

where $p_{nd_{IoT}}$ and p_{rq} are the geographical coordinates in terms of their latitude and longitude, $r_{nd_{IoT}}$(or r_{rq}) describes the communication distance of an *IoT* node (or the radius of rq interesting in). It means that the spatial constraint corresponds to a circular region where the center and radius of the circle are $p_{nd_{IoT}}$ (or p_{rq}) and $r_{nd_{IoT}}$ (or r_{rq}) respectively. Therefore, we propose to calculate the spatial relevancy between rq and nd_{IoT} as follows:

$$spt(rq, nd_{IoT}) = (spt(rq) \cap spt(nd^i_{IoT})) \div spt(rq) \tag{7}$$

where $spt(rq, nd_{IoT})$ represents the proportion of spatial coincidence with respect to rq and nd_{IoT}. The larger the value of $spt(rq, nd_{IoT})$, the larger the overlap for rq and nd_{IoT} is. We do not consider *IoT* nodes that are beyond the scope of the service request.

From the perspective of multiple fashion, *IoT* service compositions for concurrent requests are represented by $comp(allrq)$, and the spatial constraint of involved *IoT* nodes with respect to service requests is calculated as follows:

$$spt(comp(allrq)) = \frac{1}{j} \sum_{u=1}^{v} \sum_{i=1}^{j} spt(rq_u, nd^i_{IoT}) \tag{8}$$

Note that the variable v represents the number of concurrent service requests and j denotes the number of *IoT* nodes instantiated in all service compositions $comp(allrq)$.

4.1.2. Temporal relevancy of Common Components with Respect to Various Service Requests

There exists common components in concurrent service requests, which may be processed once to satisfy these requirements. These components, as individual elements, connect with others in terms of the sequential logical structure of requirements. And the temporal intervals as constraints are additional conditions for satisfying connections. Temporal intervals are relative based on the contiguity constraints of two adjacent service classes. However, relative intervals are not fully aware of the temporal relation of the common components in different service requests. Therefore, it is necessary to convert relative temporal intervals into an uniform standard in each service requests. Based on the initial time point, each relative interval is transformed into an absolute constraint shown as Alogrithm 2, which is adopted to calculating the temporal coincidence of intervals for common service classes in different service requests. Therefore, for each clS_p in CLS_{con} and its corresponding rq_u and rq_v in RQ_{cls}, the temporal relevancy is computed as follows:

$$tpr(clS_p) = \frac{Intv(rq_u.CLS(clS_p)) \cap Intv(rq_v.CLS(clS_p))}{Intv(rq_u.CLS(clS_p)) \cup Intv(rq_v.CLS(clS_p))} \quad (9)$$

The temporal relevancy is calculated through the upper bounds and lower bounds of these intervals. Intuitively, the larger the value of $tpr(clS_p)$ is, the larger the overlap interval of two service requests for this certain service class is. Therefore, common service classes in concurrent service requests have certain possibilities instantiated as the same *IoT* service and executed only once based on $tpr(clS_p)$, for reducing its energy consumption by activating a minimum number of *IoT* nodes. For concurrent service requests, the temporal relevancy of service compositions $tpr(comp(allrq))$ is affected by the proportion of relative coincidence and calculated by the number of *IoT* nodes instantiated in $comp(allrq)$.

4.1.3. Energy Consumption of IoT service composition

Due to the limited amount of energy in *IoT* nodes, the energy consumption of *IoT* service compositions for *IoT* nodes should be minimized for prolonging the lifetime of the whole network. Given service compositions $comp(allrq)$ for concurrent requests, the energy consumption for $comp(allrq)$ can be calculated including the following ingredients:

- Energy consumption for instantiating *IoT* services in an *IoT* node nd^i_{IoT} is computed as follows:

$$E_{inv}(svI_m) = t_i \times E_{inv}(svI_m) \quad (10)$$

$$E_{inv}(nd^i_{IoT}) = \sum_{m=1}^{n} E_{inv}(svI_m) \quad (11)$$

  where t_i is the invocation times for a certain *IoT* service svI_m, and E_{inv} refers to the energy consumed for activating svI_m. Therefore, the energy consumption for an *IoT* node is calculated includes all the *IoT* services hosted on it.
- Energy consumption for communication between *IoT* nodes is calculated as follows:

$$E_{comm}(nd^i_{IoT}) = \sum_{k=1}^{h} E_{Tx}(k,d) + \sum_{k=1}^{l} E_{Rx}(k) \quad (12)$$

  where i and j are the times of transmitting and receiving data packets for nd^i_{IoT}. It is worth mentioning that when nd_{IoT} do not require to transmit any packet to other *IoT* nodes, $E_{Tx}(k,d)$ is set to 0, and the same situation holds for $E_{Rx}(k)$. The details about energy consumption for the communication are presented in Section 2.2.
- Energy consumption of an *IoT* node is calculated as follows:

$$E_{cst}(nd^i_{IoT}) = E_{inv}(nd^i_{IoT}) + E_{comm}(nd^i_{IoT}) \quad (13)$$

  Note that the energy consumption of an *IoT* nodes includes instantiation and communication as presented in Formula 11 and 12.

Therefore, the energy consumption of service compositions $comp(allrq)$ for concurrent service requests is computed by all *IoT* nodes as follows:

$$E(comp(allrq)) = \sum_{i=1}^{j} E_{cst}(nd^i_{IoT}) \quad (14)$$

4.1.4. Residual Energy Constraint of IoT nodes

IoT nodes should be balanced by considering the load balancing and avoiding the excessive composition of any *IoT* node [16]. Given concurrent service requests which are divided into several kinds of service classes, each of them has an alternative list of *IoT* services. According to different *IoT* nodes they belong to, those who with relatively large amounts of remaining energy can be an instantiation. Formally, the *IoT* node nd_{IoT} should have enough residual energy (denoted E_{rsd}) than required to be consumed (denoted E_{cst}) for implementing a certain functionality:

$$E_{cst}(nd^i_{IoT}) \leq E_{rsd}(nd^i_{IoT}) \quad (15)$$

To avoid the over-consumption of any *IoT* node, a load-balancing factor is proposed to prevent *IoT* nodes with relatively low residual energy from instantiating more *IoT* services, if there are other candidate nodes. Formally,

$$lbf(nd^i_{IoT}) = (E_{rsd}(nd^i_{IoT}) - E_{cst}(nd^i_{IoT})) \div E_{rsd}(nd^i_{IoT}) \quad (16)$$

$$lbf(comp(allrq)) = \frac{1}{j} \sum_{i=1}^{j} lbf(nd^i_{IoT}) \quad (17)$$

Note that the $lbf(nd^i_{IoT})$ can avoid excessive energy consumption effectively for any of *IoT* nodes, and thus,

achieve the energy load balancing in the whole network. This strategy avoids the polarization of *IoT* nodes. When the value of $lbf(comp(allrq))$ is relatively large, which indicates that the majority of *IoT* nodes, as the instantiation of service classes contained in $comp(allrq)$ have relatively larger amount of remaining energy.

4.2. Service Composition Optimization of Concurrent Service Requests

For multiple concurrent service requests with temporal constraints, a component expressed as a service class clS can be instantiated by a set of candidate *IoT* service lists. In this setting, selecting an appropriate *IoT* service composition for each service request by composing corresponding *IoT* nodes, and thus, saving the energy of the whole network is the challenge to be addressed. The problem is reduced to a constrained multi-objective optimization problem, which is presented as follows:

- Input Parameters
  1. $RQ_{con} = \{rq_1, \ldots, rq_u, \ldots\}$: the set of concurrent service requests with temporal constraints.
  2. $CLS_{con} = \{clS_1, \ldots, clS_p, \ldots\}$: the set of service classes can be shared by different service requests.
  3. $spt(nd^i_{IoT})$: the spatial constraint of nd^i_{IoT}.
  4. $spt(rq_u)$: the spatial constraint of rq_u.
  5. $tpr(clS)$: the temporal interval relevancy of a certain service class clS in multiple concurrent service requests.
  6. t_i: the invocation times for a certain svI_m.
  7. $E_{inv}(svI_m)$: the energy consumption of activating a single *IoT* service svI_m.
  8. $E_{rsd}(nd^i_{IoT})$: the residual energy of a certain *IoT* node nd^i_{IoT}.
  9. $E_{Tx}(k, d)$: the energy consumption of transmitting a k bit data packet within a distance d.
  10. $E_{Rx}(k)$: the energy consumed for receiving a k bit data packet.
- Output Parameters
  1. $comp(allrq)$: the optimal *IoT* service compositions for concurrent service requests which can fulfill all the temporal constraints.
  2. $E(comp(allrq))$: the sum of energy consumption of concurrent service requests.
- Constraints

$$E_{rsd}(nd^i_{IoT}) \geq E_{cst}(nd^i_{IoT}) \tag{18}$$

- Multi-objective Functions
  1. Minimize

$$Z_{min} = E(comp(allrq)) \tag{19}$$

  2. Maximize:

$$Z_{max} = \alpha \cdot spt(comp(allrq)) + \beta \cdot tpr(comp(allrq)) + \gamma \cdot lbf(comp(allrq)) \tag{20}$$

where the value of α, β and γ are positive constants, and $\alpha + \beta + \gamma = 1$. Intuitively, their specific values depend on the importance of $spt(comp(allrq))$, $tpr(comp(allrq))$ and $lbf(comp(allrq))$.

Hence, the objective function is calculated as follows:

$$\begin{aligned} F_{obj}(comp(allrq)) = w_{max} \cdot Z_{max} - w_{min} \cdot Z_{min} = \\ w_{max} \cdot (\alpha \cdot spt(comp(allrq)) + \beta \cdot tpr(comp(allrq)) \\ +\gamma \cdot lbf(comp(allrq))) - w_{min} \cdot E(comp(allrq)) \end{aligned} \tag{21}$$

where w_{max} and w_{min} are objective factors of Z_{min} and Z_{max}, and $w_{max} + w_{min} = 1$. The value of objective function is the judgment form of service composition. Under the circumstances, service composition is an approximately optimal solution with relatively large $F_{obj}(comp(allrq))$.

4.3. Optimization Algorithms for Concurrent Service Composition

Two heuristic algorithms, *P*article *S*warm *O*ptimization (*PSO*) and *G*rey *W*olf *O*ptimizer (*GWO*) are adopted in this paper, where *PSO* shows more effective than other algorithms like genetic algorithm in achieving optimal *WSN* service compositions in our previous work [6]. *GWO* is a relatively novel intelligent algorithm with fast convergence. Generally, *GWO* has fewer adjustable parameters, which can alleviate the impact of subjective parameter settings. These two algorithms are implemented to solve this constrained multi-objective optimization problem proposed aforementioned.

4.3.1. Particle Swarm Optimization (PSO)

PSO is an evolutionary algorithm which inspired by the regularity of the movement of birds initially. Generally, *PSO* utilizes the information shared by the individuals in the population to make the whole population movement in the solution space produce the evolution process from disorder to order so as to obtain the optimal solution. In our context, the set of concurrent service requests refers to a particle, and the service compositions of them *comp(allrq)* correspond to a certain position of particles. At each iteration, the position and velocity of particles are updated by its most appropriate global position value of all the particles (denoted as *gbest*) and the most appropriate individual position value that each particle has acquired (denoted as *pbest*). Each particle p_i updates its velocity and position according to the following formulae:

$$\begin{aligned} v_{id}(t+1) = wv_{id}(t) + c_1 r_1 (pbest_{id}(t) - x_{id}(t)) \\ +c_2 r_2 (gbest_{id}(t) - x_{id}(t)) \end{aligned} \tag{22}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t) \tag{23}$$

where d is the number of service classes in *comp(allrq)*, w refers to an inertia weight factor which represents the effect of the current velocity value on the next iteration. $c1$ and $c2$ represent acceleration coefficients which are positive constant variables, $r1$ and $r2$ are random variables. The position of particles change when an *IoT* service are substituted by another in *comp(allrq)*.

4.3.2. Grey Wolf Optimizer (GWO)

GWO is a kind of swarm intelligence optimization method which imitates the social hierarchy and hunting mechanism of wolves in nature [17] Generally, the wolf populations are divided into four levels following their social status from high-ranking to low-ranking denoted as α wolf, β wolf, δ wolf and w wolves, respectively. α, β and δ wolves are responsible for guiding the entire swarm to catch the prey. In this setting, wolves refer to concurrent service requests, and *comp(allrq)* represents the position which is changed accordingly when services in the service composition are replaced. α, β and δ wolves refer to the optimal solution, suboptimal solution and the third optimal solution, of a population of wolves, where the rest are w wolves. The gray wolf population updates their position according to formulae as follows:

$$D_{\alpha,\beta,\delta} = \mid 2r_2 \cdot X_{\alpha,\beta,\delta}(t) - X(t) \mid \tag{24}$$

$$X_{\alpha,\beta,\delta} = X_{\alpha,\beta,\delta}(t) - (2a \cdot r_1 - a) \cdot D_{\alpha,\beta,\delta} \tag{25}$$

$$X(t+1) = \sum X_{\alpha,\beta,\delta} \div 3 \tag{26}$$

where r_1 and r_2 are two random vectors, a is a linear decreasing variable which is used to facilitate the update of the location for wolves with the number of iterations. The rest of wolves update their position according to the movement of α, β and δ wolves.

5. Implementation and Evaluation

A prototype has been implemented using Java program to evaluate experimental results, 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 as follows.

5.1. Experiment Settings

The parameter settings for our experiments are presented at Table 2 detailedly. A network with 400 *IoT* services is deployed, and these *IoT* services are assigned to *IoT* nodes randomly. The number of *IoT* nodes ranging from 30 to 70 is generated and distributed in a rectangle geographical region of $500m \times 500m$, where these *IoT* nodes are deployed unevenly with the skewness degree 0.4. Generally, the skewness degree (denoted *sd*) represents the unevenness of distribution of *IoT* nodes in the network, and it is calculated by the formula: $sd = (dn - sn) \div (dn + sn)$, where dn and sn refer to the number of *IoT* nodes in dense and sparse sub-region, respectively [18]. These *IoT* nodes are assumed to have the same initial residual energy. We adopt 14 sensor types supported by the Android platform as service classes as mentioned in our previous work [6], and assign 400 *IoT* services to respective service classes. Intuitively, an *IoT* service cannot belong to two service classes, but a service class contains multiple *IoT* services. In our experiments, an *IoT* node is assigned multiple *IoT* services randomly with various functionalities and there is no correspondence between *IoT* nodes and service classes. An experimental example shown as Figure 2, we determine the upper and lower bounds of temporal interval between two adjacent service classes by generating pairs of random numbers. Note that the upper bound of temporal intervals must be larger than the lower bound, because we only think about the positive duration of functionalities.

Table 2: Parameters Settings in Experiments.

Parameters Name	**Value**
Network region	$500\ m \times 500\ m$
Number of IoT Services	400
Number of IoT Nodes	$30 \sim 70$
Skewness degree (sd)	0.4
Attenuation index of transmission (n)	2
Number of service classes	14
Total number of iterations	100
Invocation times for a sensor node(t_i)	1
Energy consumption constant for electronics (E_{elec})	$50nJ/bit$
Energy consumption constant for the transmit amplifier (ϵ_{amp})	$100pJ/(bit \times m^2)$

Besides, the parameters for *PSO* are set as follows: (i) the acceleration coefficient for velocity $c1 = 2$, (ii) the inertia weight factor $w = 0.5$ which shows the impact of previous values of velocity for the current values, and (iii) the acceleration coefficient for the position $c2 = 2$. The parameters for *GWO* are set as follows: (i) $r1$ and $r2$ are two random vectors between 0 and 1, (ii) a decreases from 2 to 0 linearly with the increasing number of iterations. The number of iterations for *PSO* and *GWO* is set to 100. And the parameters for the objective function as specified at Formula 21 are set as follows: $w_{max} = 0.5$, $w_{min} = 0.5$, $\alpha = 0.2$, $\beta = 0.3$, $\delta = 0.5$. These parameters can be set to other values of appropriate according to the requirement of certain applications. Consequently, the emphasis of minimum and maximize objection functions are considered equivalent, and the load-balancing factor is more important in the maximize function.

The following aspects have been considered for evaluating the performance and parameters of our mechanism:

- *The performance of algorithms.* Two intelligent optimization algorithms are adopted to solve this constrained multi-objective optimization problem, and

they are evaluated by comparing the values of objective function, the minimum residual energy and the variance of residual energy in the whole network.

- *The number of IoT nodes in the network.* Intuitively, the number of *IoT* nodes deployed affects the experimental results to some extent. The fewer the number of *IoT* nodes is deployed in the network, the more *IoT* services are configured on each node. And thus the energy consumption of communication is economized in the network. In our experiments, the number of *IoT* nodes distributed in the network region is set to a value ranging from 30 to 70.
- *The number of functional-overlap in concurrent service requests.* In our experiments, the functional-coincidence is set by adjusting the number of the shared functionalities that satisfy temporal constraints in multiple service requests. The number of overlapping service classes pairs ranging from 4 to 7 gradually to verify the impact of experiments.

5.2. Experimental Evaluation

5.2.1. Experimental Result and Impact of Key Parameters

According to the experimental sample mentioned in Section 5.1, 400 *IoT* services are generated randomly, and their names and descriptions are composed by semantic keywords of these given service classes. The temporal constraints of concurrent service requests are represented as Figure 2. Firstly, *PSO* and *GWO* are adopted to compose services according to the given temporal service request graphs. Results of *IoT* services compositions and corresponding *IoT* nodes by *PSO* and *GWO* are shown in Table 3 and 4. According to Formula 21, the values of objective function corresponding to service compositions generated by *PSO* and *GWO* are 0.372021 and 0.371511, respectively, where the larger the value is, the better the composition is. The results show that appropriately optimal service compositions for concurrent service requests by adopting two heuristic algorithms *PSO* and *GWO* are mostly similar. The difference between their results is mainly due to the randomness of these experiments. And two algorithms are adopted to show the impact of different algorithms to our *CRIO* mechanism.

The variation tendency about the value of the objective function is shown in Figure 3. Evolution has been executed 50 contiguous times for reducing the impact of randomness. The figure shows that the values of objective function of both algorithms are around 0.37, they have slight variation and the results for *PSO* are more stable comparing with *GWO*. Figure 4 shows the ratio of minimum residual energy of all *IoT* nodes for 50 contiguous time slots. The curve for *PSO* is slightly smoother than *GWO* which decreases quickly as shown in the figure. The initial energy of all *IoT* nodes is set to the same amount in our experiments, and the ending of the entire network lifecycle is identified when the first *IoT* node exhausts its energy. As can be seen from the figure, *PSO* is more effective in preserving the minimum residual energy by selecting appropriate *IoT* services for instantiating service requests. Figure 5 shows the variance of the residual energy of *IoT* nodes for 50 contiguous times when *PSO* and *GWO* are adopted. Generally, the variance refers to the balance of energy consumption in the whole network, and the more uneven of the energy consumption of *IoT* nodes, the larger the value of variance is. The figure shows that *PSO* outperforms *GWO* in balancing the energy consumption in the network, thus extending the lifecycle of entire network.

Figure 6 shows the comparison of energy consumption when the number of *IoT* nodes ranges from 30 to 70. The other parameters are the same setting for this experiment. This figure shows that the energy consumption ascends gradually with the increasing number of *IoT* nodes. Note that when two adjacent services are configured on the same *IoT* node, the energy consumption of transmit-

Table 3: The results of *IoT* service compositions by *PSO*

	clS_{id}	svI_{id}	nd^{id}_{IoT}
rq1	3	325	42
	4	246	5
	2	293	17
	1	219	15
	9	164	29
	10	161	10
rq2	6	346	2
	4	246	5
	1	296	29
	9	255	48
	14	135	24
	2	124	25
	8	160	7
rq3	7	191	16
	14	263	14
	1	296	29
	4	309	24
	6	253	6
	10	214	26

Table 4: The results of *IoT* service compositions by *GWO*

	clS_{id}	svI_{id}	nd^{id}_{IoT}
rq1	3	319	43
	4	172	40
	2	138	21
	1	73	3
	9	362	33
	10	297	16
rq2	6	396	11
	4	206	33
	1	313	29
	9	164	29
	14	244	32
	2	216	2
	8	324	22
rq3	7	181	2
	14	250	7
	1	375	17
	4	100	12
	6	60	16
	10	387	6

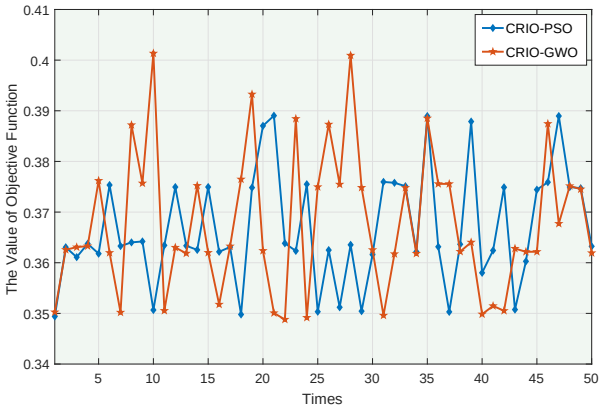

Figure 3: The value of objective function for *PSO* and *GWO* when the algorithms are executed in 50 contiguous times.

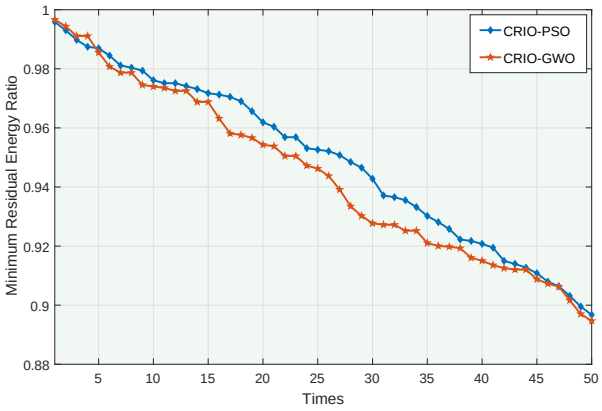

Figure 4: The ratio of minimum residual energy of all *IoT* nodes for *PSO* and *GWO* when the algorithms are executed in 50 contiguous times.

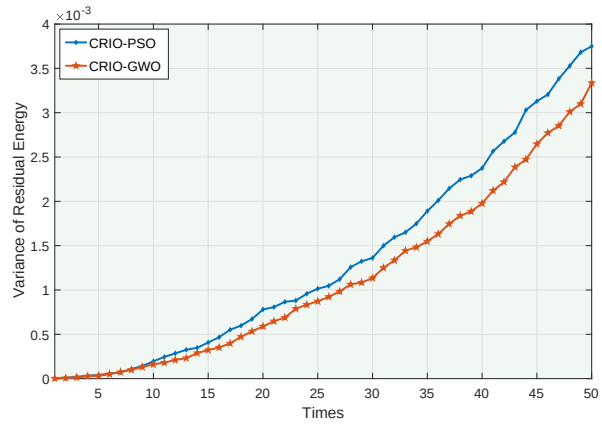

Figure 5: The variance of minimum residual energy of all *IoT* nodes in the whole network for *PSO* and *GWO* when the algorithms are executed in 50 contiguous times.

ting and receiving data packets is saved in the network, and which reduces the energy consumption of the whole network whether *PSO* or *GWO* is adopted.

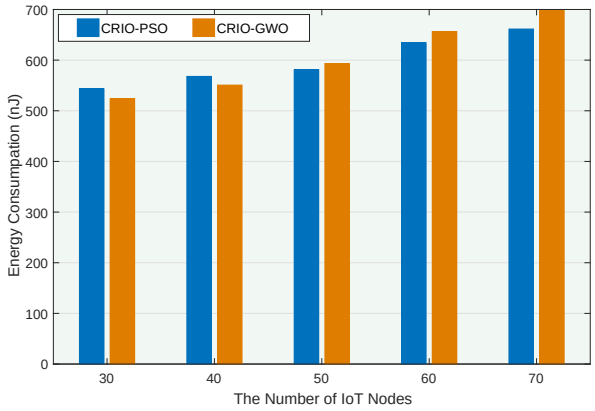

Figure 6: The energy consumption for *IoT* nodes ranging from 30 to 70 adopted to *PSO* and *GWO* respectively.

The impact of the number of functional-overlap service classes in concurrent service requests on energy consumption for *PSO* and *GWO* is shown in Figure 7 and Figure 8 respectively. Experiments have been executed 10 times, which show that the energy consumption is decreasing along with the increasing number of functional-overlap in concurrent service requests. Generally, the number of functional-overlap of service classes has a significant impact on energy consumption regardless of which algorithm is adopted, especially when the functional-overlap degree is relatively large. The overlapping of functionalities increases the possibility of service sharing to reduce energy consumption according to temporal interval coincidence.

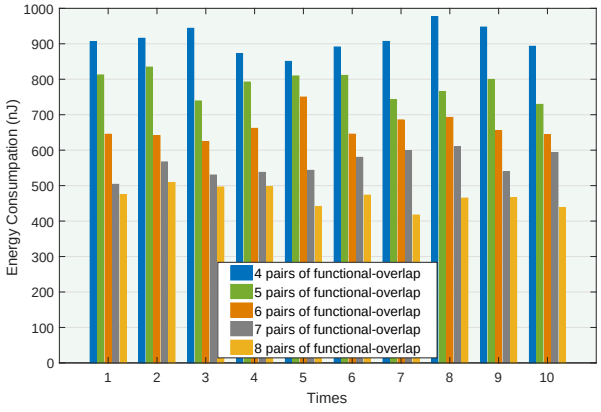

Figure 7: The energy consumption of different number of functional-overlap in concurrent service requests when *PSO* is executed in 10 times.

5.2.2. Comparison With CR for Service Composition

This section presents the comparison for our mechanism *CRIO* and the strategy that implementing and instantiating these service classes in concurrent service requests in an independent fashion is denoted as *CR*. To reduce of the influence of different algorithms on exper-

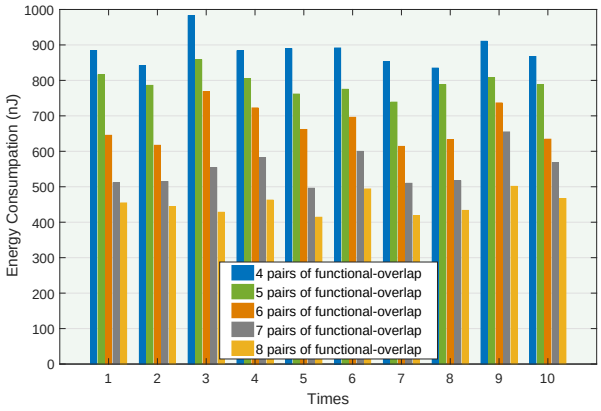

Figure 8: The energy consumption of different number of functional-overlap in concurrent service requests when *GWO* is executed in 10 times.

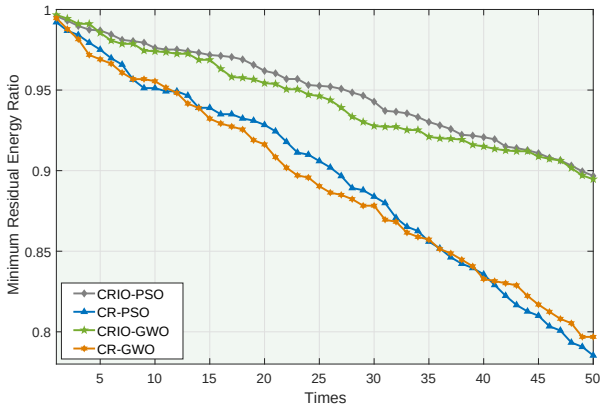

Figure 10: The ratio of minimum residual energy of *IoT* nodes in the whole network for *CRIO* and *CR* with *PSO* and *GWO* when the algorithms are executed in 50 contiguous times.

imental results, *PSO* and *GWO* were used to compare the performance of *CRIO* and *CR*, respectively. Figure 9 shows that the *CRIO* optimization mechanism adopting these two algorithms can reduce more energy consumption than *CR*. The minimum residual energy ratio for *CRIO* and *CR* is presented as Figure 10. This figure shows that the minimum residual energy of *IoT* nodes for *CRIO* is more than *CR*, which means that *CRIO* has better results in preserving energy and our mechanism is effective. Due to integration and sharing in common service classes, the energy consumption of *CRIO* for concurrent service requests is reduced comparing to the *CR* which executes each functionality independently. Therefore, for *CRIO*, the minimum residual energy ratio of the whole network is larger no matter what algorithm is adopted.

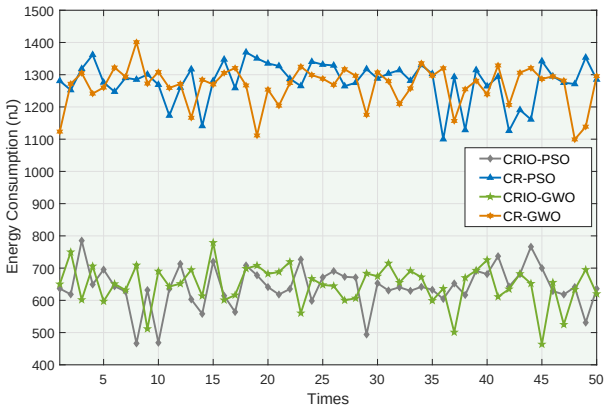

Figure 9: The energy consumption for *CRIO* and *CR* adopted to *PSO* and *GWO* when the algorithms are executed in 50 contiguous times.

5.2.3. Comparison With SSA [19] for Service Selection

This section presents the comparison of our mechanism *CRIO* and the *S*ervice *S*election *A*pproach (denoted *SSA*) proposed in [19]. As discussed in [19], *SSA* considers the best-fitted *QoS* and makes efficient usage of services among multiple requests. The approach concerns both functional and non-functional requirements, where selecting temporal availability, residual energy and transmission distance as the *QoS* attributes to adapt to our experimental scenario. For minimized and maximized objective attributes, they are processed so that they can be compared by the same principle. Each *QoS* attribute value is regarded as a vector, and the sum of vectors of all attributes for an *IoT* service is used as a standard to measure it. Load-balancing among these *IoT* nodes should be fully considered to avoid being used excessively of any one, where an *IoT* service with optimal *QoS* attributes is selected in the candidate set.

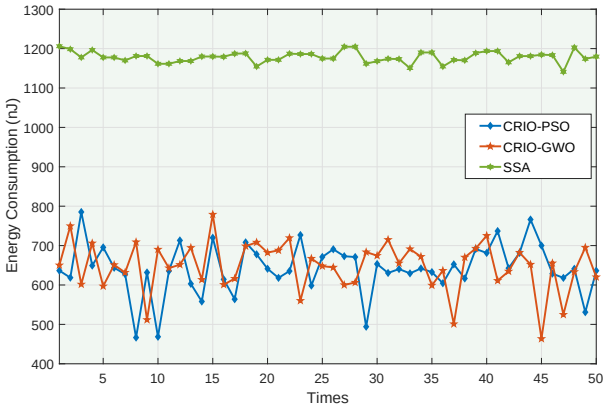

Figure 11: The energy consumption for *CRIO* adopted to *PSO* and *GWO* and *SSA* when the algorithms are executed in 50 contiguous times.

Figure 11 shows the energy consumption of our mechanism compared with *SSA* for executing 50 contiguous times. Note that *CRIO-PSO* (or *CRIO-GWO*) means that these two algorithms are adopted and optimized by the

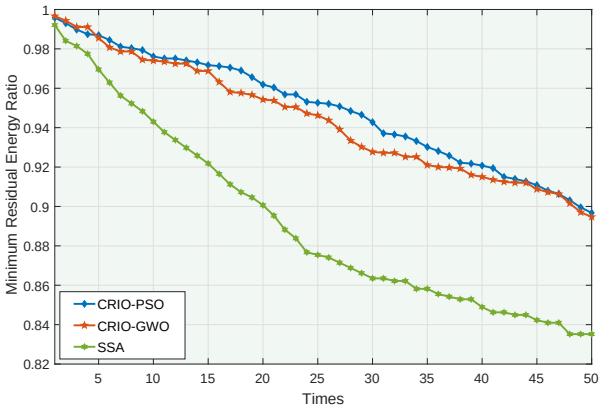

Figure 12: The ratio of minimum residual energy of all *IoT* nodes for *CRIO* adopted to *PSO* and *GWO* and *SSA* when the algorithms are executed in 50 contiguous times.

CRIO mechanism, and *SSA* is the approach designed by [19] according to our experimental scenario. The energy consumption of the *CRIO* mechanism is between 450nJ and 850nJ, which the overall average energy consumption is about 650nJ excepting some maximum or minimum values that occur in a few cases. And the energy consumption adopting *SSA* is relatively stable which is about 1200nJ. Figure 11 shows that energy consumption is much higher for *SSA* compared with *CRIO-PSO* and *CRIO-GWO*. As presented by Algorithm 1, this mechanism considers service sharing in concurrent service requests and proposes common component integration under conditions that satisfy temporal constraints. As a result, some of the energy consumed to activate services is saved, which influences the minimum residual energy in the whole network. Compared with the *CRIO* mechanism, the minimum residual energy of *SSA* decreases rapidly shown as Figure 12, which means that *CRIO* is superior in reducing energy consumption and maintaining load balancing throughout the network.

5.2.4. Comparison with MOEAQI [20] for Service Selection

This section presents the comparison for our mechanism *CRIO* and the improved multi-objective evolutionary algorithm *MOEAQI* proposed in [20], where *MOEAQI* selects optimal solutions considering both *QoS* criteria and inter-service correlations. In order to adapt to our experimental settings, we consider service selection at build time in [20]. Temporal availability, residual energy and transmission distance are chosen as the criterion to measure *QoS*. Different from business processes in [20], inter-service correlations are calculated based on the nodes hosting these *IoT* services, where energy consumption by transmitting and receiving of adjacent services is considered. The whole processes of *MOEAQI* is designed including service pruning, nondominated sorting and offspring generation. Both *QoS* constraints and inter-service correlations are considered to compute the satisfaction degree when a service composition is achieved. Classical crossover and mutation operators are adopted to generate offsprings based on evaluation strategy of service compositions. Pareto optimal service compositions are acquired when the maximum iteration is reached.

Figure 13 illustrates the energy consumption of *CRIO* compared with *MOEAQI* when both algorithms are executed in 50 contiguous times. *CRIO-PSO* (or *CRIO-GWO*) means that the *CRIO* mechanism adopted to *PSO* (or *GWO*), and *MOEAQI* is designed by [20], where some adaptive changes have been made based on our experimental environment settings. This figure shows that energy consumption of *MOEAQI* is roughly between 900nJ and 1200nJ, which is much higher than *CRIO-PSO* (or *CRIO-GWO*). Because the mechanism of *MOEAQI* does not allow for resource sharing, and does not consider the concurrency of *IoT* nodes, the optimization of concurrent services is not well addressed. The ratio of minimum residual energy of *IoT* nodes is shown by Figure 14, where the proportion of remaining energy decreases quickly due to the excessive consumption of energy in most situations. However, compared with traditional service selection methods, *MOEAQI* considers the relevance between consecutive services, which reduces a certain amount of energy consumption for sensory data transmission.

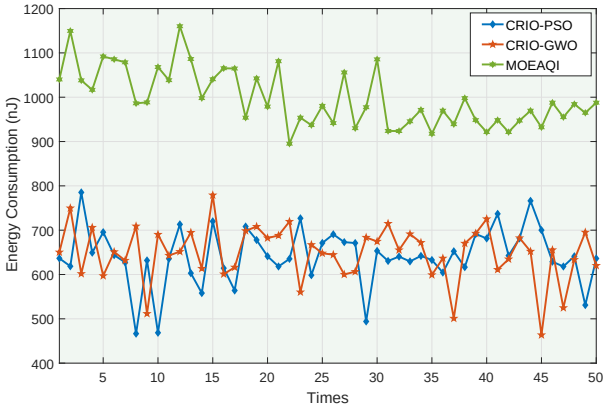

Figure 13: The energy consumption for *CRIO* adopted to *PSO* and *GWO*, and *MOEAQI*, when both algorithms are executed in 50 contiguous times.

To summarize, it can be concluded that our *CRIO* mechanism can be of significance for integrating concurrent users' requirements with temporal constraints and thus forming relatively optimized service compositions by comparing with the state-of-art approaches including *SSA* and *MOEAQI*. Two swarm intelligent algorithms, *PSO* and *GWO* show little difference in experimental results of the objective function and *PSO* performs slightly better than *GWO* in minimum and variance of residual energy. Meanwhile, the energy consumption relates to the number of functional-overlap in concurrent service requests and the number of *IoT* nodes throughout the network.

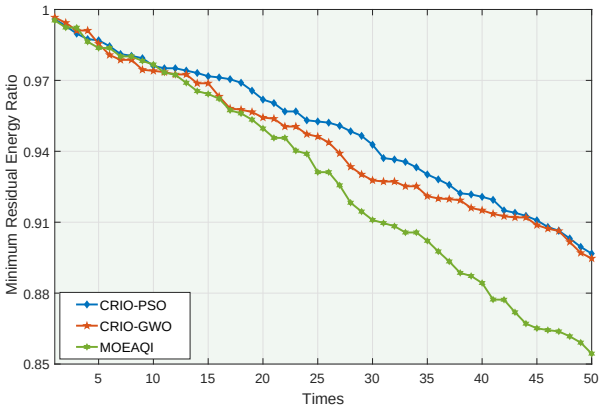

Figure 14: The ratio of minimum residual energy of all *IoT* nodes for *CRIO* adopted to *PSO* and *GWO* and *MOEAQI* when the algorithms are executed in 50 contiguous times.

6. Related Work and Comparison

In this section, we classify related techniques and approaches into the following three aspects: (i) energy-efficient service composition, (ii) multi-application sharing, and (iii) temporal constraints for service composition.

6.1. Energy-Efficient Service Composition

With the development of service defined everything [21, 22], *SOA* is adopted to achieve higher levels of interoperability, and related techniques are denoted as a component-based model which interrelates various functional services. Due to the evolution of Cloud computing, Edge computing and *IoT*, some techniques explore service compositions based on diverse infrastructure platforms. A single intelligent node can hardly fulfil relatively complex requirements. Consequently, the cooperation and collaboration between nodes is essential [23, 24, 25, 26]. These nodes can be connected and composed through traditional service composition techniques. A composite service orchestrates a set of atomic services to solve a relatively complex goal successfully in a way that adds value to the delivered composite services [27]. It is worth noting that different from traditional service composition techniques, the physical characteristics of nodes should be considered for ensuring the win-win result.

Besides, some techniques adopt software agents as potential candidates for achieving the collaboration of services. Generally, software agents deployed in Cloud architecture are implemented to fulfill requirements of applications. Energy-efficiency is a promising research challenge in service composition of infrastructure. In [28], authors developed a novel multi-cloud *IoT* service composition algorithm which is adopted to reduce a large amount of data exchange and various other operations. They created an energy-aware composition plan through discovering and integrating the least possible number of *IoT* services in order to fulfill certain requirements. A novel multi-agent approach to perform Web service composition is proposed in [29], which fulfills compositions based on user's requirements of either energy efficiency or cost-effectiveness through adopting to the efficient retrieval of distributed services and propagation of information within the agent network for reducing the amount of brute-force search. To select the most energy efficient service compositions from cloud-based data centers, a novel bin-packing service broker is proposed in [30] using integer linear programming named Cloud-SEnergy. This technique matches the user's needs and discovers the least possible number of cloud services' providers in the multi-cloud environment to accomplish the objective of energy efficiency.

Energy-aware service composition has been paid extensive attention in *WSN* in recent years. Authors in [6] presented a three-tier service-oriented framework, where sensor nodes are encapsulated as *WSN* services, and they are classified as service classes in terms of their similarity of functionalities. These service classes are selected to construct chains for fulfilling the requirements. In the process of instantiation, spatial and temporal constraints and energy efficiency are taken into consideration to select a service composition which has greater relevance and less energy consumption. As a special architecture of *WSN*, physical properties of *IoT* nodes, especially the energy consumption, is the focus of current methodological studies. In [31], authors proposed a situation-aware dynamic *IoT* services coordination approach, which effectively integrates the advantages of service-oriented architecture and event driven architecture. Energy consumption is the main challenge in composing the minimum service nodes to preserve the energy and decrease the cost. In our approach, service compositions are optimized through integrating common functional components, where taking functionality properties as the first class citizen. Besides, spatial constraints, temporal relevancy and energy consumption are non-functional factors which affect the result of service recommendation and composition.

6.2. Multi-Application Sharing

As the explosive growth of the network scale, both network traffic and bandwidth are presented by unprecedented tendency. Complex networks are adopted to fulfill more and more requirements, and the concurrency is one of the research topics of network service that supporting multiple applications in various domain. Generally, supporting concurrent applications is an approach for maximizing the resource utilization and reducing response time [32]. Serval approaches have been developed for managing the processing of multiple applications. Data sharing technique amongst multiple applications efficiently reduces energy consumption and communication cost by decreasing the number of data sampling. In [33], authors proposed a data sharing approach to schedule sampling intervals in *WSN*s. This approach adopts jointly optimizing task allocation and sampling interval scheduling to maximize data

sharing. Authors in [9] study data sharing for data collection and how to reduce the overall length of data sampling intervals among multiple applications. This problem is formulated as a non-linear nonconvex optimization problem, where a greedy approximation algorithm is adopted to improve resource utilization.

With the increasing number of requirements, sequential processing cannot satisfy efficiency demand for multiple time-sensitive applications. There are obvious advantages where an infrastructure can be shared across multiple applications. The sharing of equipments and sensing architectures in the network plays an important role in the optimization of multi-application. A data aggregation architecture which optimizes the power consumption of several applications deployed on the sensor nodes is proposed in [34]. The architecture includes two layers, the low layer is formed by the physical sensor nodes and the high layer is constituted by agents which adopts a cooperative strategy based on a single application. In [8], a platform that transforms the sensor network into an open access infrastructure that supports multiple collaborative applications is proposed, which decouples between the infrastructures and applications. In this platform, each application works in an isolated environment which consisted by a hardware abstraction layer. The innovative development can open new opportunities for efficient resource utilisation. Generally, these approaches are mainly designed for the multi-application sharing technology, whereas the time attribute of applications is not taken into account. Among the interval data sampling techniques mentioned earlier, there is a time interval involving data procurement, which have some similarities with our paper to an extent. However, those methods we mentioned do not involve temporal dependencies, which is an emphasis of our approach to implement service sharing in multiple application combinations.

6.3. Temporal Constraints for Service Composition

Temporal property is an essential ingredient to guarantee the timeliness of functionalities for each service to fulfill certain requirements. As a critical non-functional indicator, temporal attribute has not been studied extensively by most current researches in service composition. Actually, each atomic service in the composition should have its own temporal restriction. In some scenarios, requirements are limited to a certain of global temporal intervals, which can be satisfied only if all the service providers in the composition apply themselves to satisfy their own local temporal constraints. Usually, these services are ensured effectiveness, which are the antecedent conditions on the occurrence of the following [35]. Therefore, it is important to take the temporal constraints into consideration for *IoT* service composition.

Due to the sequence and temporal factors, some papers focus on the dynamic controllability in service composition. In [12], a novel proactive dynamic service selection approach is proposed to solve uncertainty during service execution. Several sudden situations may occur at runtime because of the dynamic of system. To do so, a set of thresholds is identified to characterize the trigger dynamic selection mechanism and some alternative services are set for each task, which are updated during execution based on the result of the already executed services to meet the temporal constraints of users. In order to guarantee the service processes successfully, it is important to optimize the service dynamically for guaranteeing certain temporal constraints. A two-stage approach based on dynamic optimisation of service processes is proposed in [36] based on temporal constraints. Firstly, calculating the temporal constraints by considering both the uncertainty of queue time and operation time of services in processes. And the temporal adjustment model is adopted to adjust optimal solution. Once potential temporal violations are discovered, temporal adjustment is executed to fulfill the requirement of temporal consistency.

Petri nets are widely used to model and analyze service composition technology. Petri net describes the functional transition between activities through sequence and logical structure. Authors in [37] proposed a Petri net-based algebra to model control flows. They declared that the defined algebra is a good way to represent dynamic and transient relationships among services. Authors structured service composition based on workflow Petri nets and discussed the compatibility and environments in that technique. In [38], authors presented a Petri net-based method to consider message mismatches, state-space explosion and execution paths in a modular way. Petri net is used to formalize services, which can express the temporal relation between services clearly. As another canonical model, the temporal problems addressed in the thesis are stated and analyzed based on simple temporal networks [13]. The qualitative network model of interval algebra is adopted to describe temporal constraints among the tasks to which an agent has committed itself. Generally, these approaches are mainly focus on absolute temporal relations in service composition, whereas relative temporal dependency has not been explored extensively. To remedy this issue, this paper takes temporal dependency and service sharing into concern for supporting concurrent applications, and hence, addressing the energy efficiency throughout the network.

7. Conclusion

This paper proposes an energy-efficient mechanism to optimize *IoT* service compositions for supporting concurrent requests. Specifically, an *IoT* node is encapsulated with multiple *IoT* services which correspond to various functionalities hosted by this *IoT* node. *IoT* services are classified into service classes based on the similarity of their functionalities. The requests are achieved by the composition of service classes with temporal dependencies. Compositions of concurrent requests are optimized through integrating common components. This *IoT* service composition is reduced to the constrained multi-objective opti-

mization problem, where spatial-temporal relevancy, energy efficiency of *IoT* nodes are considered. Experimental results show that this technique can improve the shareability of *IoT* services among concurrent requests, and reduce the energy consumption of the network. Note that *IoT* services and service classes may be different in their granularity. Considering the impact of granularity to *IoT* service composition is one of our future research challenges.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61772479 and 61662021).

References

[1] S. Xiong, Q. Ni, X. Wang, Y. Su, A connectivity enhancement scheme based on link transformation in iot sensing networks, IEEE Internet of Things Journal 4 (6) (2017) 2297–2308.

[2] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, L. T. Yang, Trust-based communication for the industrial internet of things, IEEE Communications Magazine 56 (2) (2018) 16–22.

[3] K. S. Adu-Manu, N. Adam, C. Tapparello, H. Ayatollahi, W. Heinzelman, Energy-harvesting wireless sensor networks (eh-wsns): A review, ACM Transactions on Sensor Networks 14 (2) (2018). doi:10.1145/3183338.

[4] C. Wang, J. Li, Y. Yang, F. Ye, Combining solar energy harvesting with wireless charging for hybrid wireless sensor networks, IEEE Transactions on Mobile Computing 17 (3) (2018) 560–576.

[5] X. Xue, S. Wang, Z. Lejun, Z. Feng, Evaluating of dynamic service matching strategy for social manufacturing in cloud environment, Future Generation Computer Systems 91 (2019) 311–326.

[6] Z. Zhou, D. Zhao, L. Liu, P. C. K. Hung, Energy-aware composition for wireless sensor networks as a service, Future Generation Computer Systems 80 (2018) 299–310.

[7] J. Maerien, S. Michiels, D. Hughes, Christophe, Seclooci: A comprehensive security middleware architecture for shared wireless sensor networks, Ad Hoc Networks 25 (A) (2015) 141–169.

[8] I. Leontiadis, C. Efstratiou, C. Mascolo, J. Crowcroft, Senshare: Transforming sensor networks into multi-application sensing infrastructures, Wireless Sensor Networks 7158 (2012) 65–81.

[9] H. Gao, X. Fang, J. Li, Y. Li, Data collection in multi-application sharing wireless sensor networks, IEEE Transactions on Parallel and Distributed Systems 26 (2) (2015) 403–412.

[10] Z. Zhou, Z. Cheng, K. Ning, W. Li, L. Zhang, A sub-chain ranking and recommendation mechanism for facilitating geospatial web service composition, International Journal of Web Service Research 11 (3) (2014) 52–75.

[11] T. Laleh, J. Paquet, S. Mokhov, Y. Yan, Constraint verification failure recovery in web service composition, Future Generation Computer Systems 89 (2018) 387–401.

[12] G. Ikbel, J. I. Al, G. Nawal, Dynamic selection for service composition based on temporal and qos constraints, in: IEEE International Conference on Services Computing, 2016, pp. 267–274.

[13] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, Artificial Intelligence (1991) 61–95.

[14] J. F.Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 26 (1983) 832–843.

[15] W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, in: 33rd Annual Hawaii International Conference on System Sciences, 2000, pp. 1–10.

[16] Y. Yun, X. Ye, Maximizing the lifetime of wireless sensor networks with mobile sink in delay-tolerant applications, IEEE Educational Activities Department 9 (9) (2010) 1308–1318.

[17] H. Faris, I. Aljarah, M. A. Al-Betar, S. Mirjalili, Grey wolf optimizer: a review of recent variants and applications, Neural Computing and Applications 30 (2) (2018) 413–435.

[18] Z. Zhou, D. Z. L. Shu, H.-C. Chao, Efficient multi-attribute query processing in heterogeneous wireless sensor networks, Journal of Internet Technology 15 (5) (2014) 699–712.

[19] J. C. Lima, R. C. A. Rocha, F. M. Costa, An approach for qos-aware selection of shared services for multiple service choreographies, in: Service-Oriented System Engineering, 2016, pp. 221–230.

[20] H. Liang, Y. Du, T. Jiang, F. Li, A comprehensive multi-objective approach of service selection for service processes with twofold restrictions, Future Generation Computer Systems 92 (2019) 119–140.

[21] M. Ali, A. Benjamin, M. Adda, K. C. Heng, Optimisation methods for fast restoration of software-defined networks, IEEE Access 5 (2017) 16111–16123.

[22] Y. Duan, W. Li, X. Fu, Y. Luo, L. Yang, A methodology for reliability of wsn based on software defined network in adaptive industrial environment, IEEE/CAA Journal of Automatica Sinica 5 (1) (2018) 74–82.

[23] I. R. Chen, J. Guo, F. Bao, Trust management for soa-based iot and its application to service composition, IEEE Transactions on Services Computing 9 (3) (2017) 482–495.

[24] K. I. Young, K. H. Gyu, M. A. Jimenez, K. J. Hyun, Soiot:toward a user-centric iot-based service framework, ACM Transactions on Internet Technology 16 (2) (2016) 1–21.

[25] K. Eric, N. Amiya, Capability reconciliation for a csp approach to virtual device composition, IEEE/ACM Transactions on Networking PP (99) (2012) 1–1.

[26] M. Asim, A. Yautsiukhin, A. D. Brucker, T. Baker, Q. Shi, B. Lempereur, Security policy monitoring of BPMN-based service compositions, Journal of Software: Evolution and Process (2018). doi:10.1002/smr.1944.

[27] P. Asghari, A. M. Rahmani, H. H. S. Javadi, Service composition approaches in iot: A systematic review, Journal of Network and Computer Applications 120 (2018) 61–77.

[28] T. Baker, M. Asim, H. Tawfik, B. Aldawsari, R. Buyya, An energy-aware service composition algorithm for multiple cloud-based iot applications, Journal of Network and Computer Applications 89 (2017) 96–108.

[29] P. Kendrick, T. Baker, Z. Maamar, A. Hussain, R. Buyya, An Efficient Multi-Cloud Service Composition Using A Distributed Multiagent-based Memory-driven Approach, IEEE Transactions on Sustainable Computing (2018). doi:10.1109/TSUSC.2018.2881416.

[30] T. Baker, B. Aldawsari, M. Asim, H. Tawfik, A Bin-Packing Based Multi-Cloud Service Broker for Energy Efficient Composition and Execution of Data-intensive Applications, Sustainable Computing: Informatics and Systems, (2018). doi:10.1016/j.suscom.2018.05.011.

[31] B. Cheng, M. Wang, S. Zhao, Z. Zhai, D. Zhu, J. Chen, Situation-aware dynamic service coordination in an iot environment, IEEE/ACM Transactions on Networking 25 (4) (2017) 2082–2095.

[32] J. Li, S. Cheng, H. Gao, Z. Cai, Approximate physical world reconstruction algorithms in sensor networks, IEEE Transactions on Parallel and Distributed Systems 25 (12) (2014) 3099–3110.

[33] Y. Zhao, D. Guo, J. Xu, P. Lv, T. Chen, J. Yin, Cats: Cooperative allocation of tasks and scheduling of sampling intervals for maximizing data sharing in wsns, ACM Transactions on Sensor Networks 12 (4) (2016). doi:10.1145/2955102.

[34] A. Sardouk, R. Rahim-Amoud, L. Merghem-Boulahia, D. Ga?ti, Data aggregation scheme for a multi-application wsn, Wired-Wireless Multimedia Networks and Services Management 5842 (2009) 183–188.

[35] X. Gao, S. P. Singh, Mining contracts for business events and temporal constraints in service engagements, IEEE Transactions on Services Computing 7 (3) (2014) 427–439.

[36] H. Liang, Y. Du, Two-stage dynamic optimisation of service processes with temporal constraints, International Journal of

High Performance Computing and Networking 9 (1-2) (2016) 116–126.
[37] R. Hamadi, B. Benatallah, A petri net-based model for web service composition, in: Australasian Database Conference, 2003, pp. 191–200.
[38] Y. Du, W. Tan, M. Zhou, Timed compatibility analysis of web service composition: A modular approach based on petri nets, IEEE Transactions on Automation Science and Engineering 11 (2) (2014) 594–606.