IoT Services Configuration in Edge-Cloud Collaboration Networks

Authors

***Abstract*—The edge-cloud collaboration networks have been applied to support delay-sensitive *I*nternet *o*f *T*hings (*IoT*) applications nowadays, where applications are represented in terms of service configurations. In this setting, *IoT* services should be configured mostly at the network edge, and are offloaded to the cloud only when the capacity of edge nodes can hardly meet the requirement. To solve this problem, this paper proposes to configure *IoT* services with temporal constraints that are discovered from event logs. Service configuration is reduced to a constrained multi-objective optimization problem, which can be solved by an improved non-dominated sorting genetic algorithm II. Experimental results demonstrate that our approach outperforms the state-of-art's techniques on delay sensitivity and energy efficiency, especially when edge nodes are relatively large in the number of hosted services.**

***Keywords*-Service Configuration; Edge-Cloud Network; Temporal Constraints; Delay Sensitive; Energy Efficient.**

I. Introduction

As the complement of cloud computing, edge computing has stimulated functional composition of smart devices at the network edge to satisfy complex requirements with certain temporal constraints, whereas a computational task may necessary to be offloaded from the edge to the cloud, when this task is computation-intensive and non-delay-sensitive [1]. This edge-cloud collaboration strategy is promising to support versatile applications in the era of *I*nternet *o*f *T*hings (*IoT*), where smart devices, also known as *IoT* nodes, are encapsulated in terms of *IoT* services to formalize their functional and non-functional properties. In this setting, given an application to be fulfilled by the composition of *IoT* services, a configuration of certain *IoT* services to be hosted by appropriate edge nodes, besides a task offloading of other *IoT* services to the cloud, is fundamental, in order to satisfy temporal constraints among *IoT* services, while promoting the network performance as the decrease of the energy consumption and traffic of the network.

Techniques have been developed to investigate service configuration in the context of cloud or edge computing paradigm [2], where on-demand resource provisioning and effort-resource balancing are their main concerns. However, these techniques do not adequately consider the rational allocation of service latency and processing location. How to complete service configuration for ensuring temporal consistency and achieving energy efficiency is a matter of inadequately explored. Consequently, combining edge computing and cloud computing is promising to support the network environment featured by (near) real-time response and energy awareness. Service configuration optimization has been preliminarily explored recently in the hybrid architecture, which often hosts partial delay-critical services on edge nodes and configures insensitive services on the cloud for achieving energy-latency tradeoff [3]. Generally, they focus mainly on atomic requests and do not draw attention to the relevance between services, which can only be appropriate for uncomplicated and inter-unrelated tasks. Task-dependency requirements are applied to accomplish relatively complex applications, where it is inevitable that service compositions are mutually invoked and implemented functional alternating. Some works promote a cost-effective way to configure services for the optimization of offloading decisions considering completion time constraint and energy-efficiency [4]. The effect of invocation or data transmission on interaction and movement is not considered in service configuration optimization, which can be used as a significant measure for the reduction of energy consumption and transmission delay of the network. Besides, these techniques mainly consider the minimization of overall execution time so that they satisfy established and predefined deadlines, whereas there are few temporal constraints between internal services to ensure the successful advancing of each procedure. Consequently, optimizing service configuration while discovering temporal constraints between separate services is a challenge to be further explored.

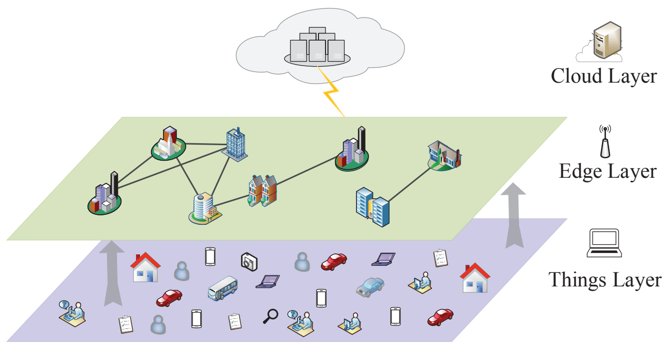


Figure 1. The framework of edge-cloud collaboration network includes cloud layer, edge layer and things layer, where *IoT* nodes lie in the things layer, and these devices carry different types of requests that are decomposed into multiple *IoT* services. The edge layer includes multiple edge nodes whose certain functionalities are configured, which should cooperate with the cloud layer for achieving requests.

To address this challenge, this paper proposes to configure services optimally with certain temporal constraints between internal services in an edge-cloud collaboration framework as shown at Figure 1, where temporal constraints are discovered from event logs through improving process mining techniques. Our contributions are summarized as follows:

- We propose a timed service-based process model, where temporal constraints between internal services are mined through our *D*iscovering *Q*ualified *T*emporal *I*ntervals (denoted *DQTI*) mechanism.
- We reduce service configuration to the constrained multi-objective optimization problem, which can be solved through our proposed *I*mproved *N*on-dominated *S*orting *G*enetic *A*lgorithm *II* (denoted *INSGA-II*), where temporal constraints between services, delay sensitivity and energy consumption are the factors to be considered.

Extensive experiments are conducted for evaluating the performance of, and investigating the effectiveness of typical parameter settings upon our proposed approach. Evaluation results demonstrate that our approach outperforms the state of art's techniques in terms of delay sensitivity and energy efficiency, especially when edge nodes are relatively large in the number of hosted services.

II. Concepts Definition

*Definition 1 (**Event**):* An event e is a tuple ($traceId$, act, tmp), where $traceId$ is the trace identifier of e, act is the activity type contained in e, and tmp is the occurrence timestamp of e.

Generally, an event is contained in a certain trace, where a trace consists of multiple events. An activity is recorded in each event, and it performs a certain type of operation. We discuss the timestamp information which is the occurrence time for each event.

*Definition 2 (**Event Log**):* An event log L is a set of traces σ_t, where $L = \{\sigma_1, \sigma_2, \ldots, \sigma_t | t \in N^+\}$, $\forall \sigma_1, \sigma_2 \in L$, $\sigma_1 \neq \sigma_2$.

An event log consists of multiple traces which correspond to a set of business process instances. Any two traces are not exactly the same in terms of activity and timestamp.

*Definition 3 (**Timed Service-based Process Model**):* A timed service-based process model *TSPM* is a tuple (*SEV*, *GTW*, *TR*, *TC*), where *SEV* is a finite set of services, *GTW* is the set of gateways of *TSPM*, *TR* and *TC* are transitive relations and temporal constraints between *SEV*.

In this paper, services are mined and extracted from event logs based on different types of activities, where a category of activities with the same/similar functionality are encapsulated into a service. A timed service-based process model is expressed as the form of business process modeling notation, which is widely adopted for model representation. Process model is produced through capturing the control-flow between services that are observed in or implied by event logs. As illustrated by Figure 2, gateways usually correspond to exclusive choice (×) and concurrency (+), which come in pairs respectively to express logical relations between services. Transitive relations connect services and gateways, where temporal constraints are attached to internal services through binding temporal relations of the previous service to the latter one.

III. Relational Information Mining

This section develops a mechanism for mining temporal constraints. Specifically, we construct a *T*emporal *D*istance *G*raph (*TDG*), discover temporal rigidity and transitive relations between internal services.

A. Temporal Constraints Mining

Temporal information reveals occurrence time and precedence order of activities, where temporal dependencies between internal services can be formed into temporal constraints that can be extracted from event logs. Chi-square test is adopted to discover temporal intervals and qualify temporal constraints as follows [6]:

$$N'_{IntvT} = \sqrt{\chi^2_{IntvT} N P_{IntvT}(1 - P_{IntvT})} + N P_{IntvT} \quad (1)$$

where N refers to the total number of association between two given services, and P_{IntvT} is the probability of theoretical value in a temporal interval $IntvT$ = [t_{min}, t_{max}].

Algorithm 1 DQTI : Discovering Qualified Temporal Intervals

```
Require:
    - $T$ : temporal lag values between two services
    - $t_{med}$ : the median of temporal lag values in $T$
    - $N_{t_i}$ : the occurrence number of each item in $T$
    - $N_{sum}$ : the total number of temporal lag values for $T$
Ensure:
    - $IntvT$ : the temporal interval for $sev_i$ and $sev_j$
1: $t_{min} \leftarrow t_{med}$, $t_{max} \leftarrow t_{med}$, $N_{IntvT} \leftarrow N_{t_{med}}$
2: $N'_{IntvT} \leftarrow$ calculated by equation (1)
3: while $N_{IntvT} < N'_{IntvT}$ do
4:     if $N_{t_{med+1}} \leq N_{t_{med-1}}$ then
5:         $t_{min} \leftarrow t_{med-1}$, $N_{IntvT} \leftarrow N_{IntvT} + N_{t_{min}}$
6:     else
7:         $t_{max} \leftarrow t_{med+1}$, $N_{IntvT} \leftarrow N_{IntvT} + N_{t_{max}}$
8:     end if
9: end while
10: $IntvT \leftarrow [t_{min}, t_{max}]$
```

We propose a *DQTI* mechanism to discover temporal constraints and the pseudocode of this procedure is presented at Algorithm 1. The initial values for lower and upper boundaries of temporal interval t_{min} and t_{max} are set to the median t_{med} firstly, where the number of items in temporal interval $IntvT$ is set to $N_{t_{med}}$ accordingly (line 1). The threshold N'_{IntvT} based on chi-square test is calculated by equation (1) (line 2). Boundary values of the temporal interval

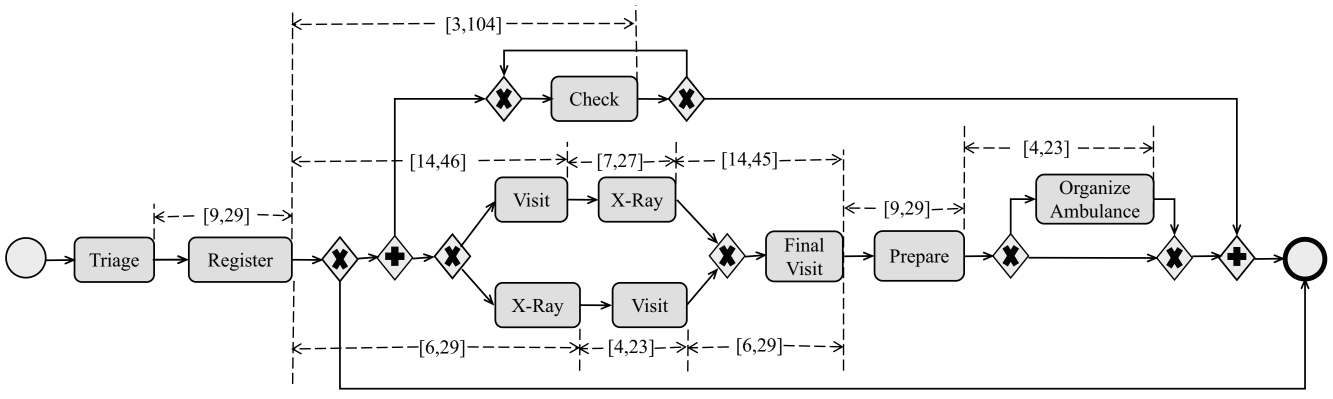


Figure 2. A simplified process in *TSPM* notation mined from the event log available at [5].

are searching through diffusing to both sides with comparing the number of temporal lags N_{IntvT} and the minimum theoretical threshold N'_{IntvT} (lines 3-9). When $N_{IntvT} > N'_{IntvT}$, a qualified temporal interval is discovered, and the minimum and maximum of discrete temporal lags are designated as boundary values at two sides of the temporal interval $IntvT$ (line 10). The established temporal interval is regarded as corresponding temporal constraint, which is attached between corresponding services in *TSPM*.

B. Temporal Distance Graph Construction

Based on *TSPM* formed in Section III-A, services and related temporal constraints are extracted to construct a temporal constraint network [7] which is adopted to solve temporal challenges. Generally, services in *TSPM* correspond to nodes, and transitive relations are transformed into connections without any binding. A pair of certain services labeled by [$t^{i,j}_{min}$, $t^{i,j}_{max}$] represents the temporal relationship between j and i, where the minimum and maximum of temporal interval are $t^{i,j}_{min}$ and $t^{i,j}_{max}$, respectively.

Formally, we associate the temporal constraint network with an edge-weighted graph represented as a *TDG* as shown in Figure 3. A path from service i to j including i = i_0, i_1, $\ldots$, i_k = j corresponds to a length of shortest temporal distance $d_{i,j}$, where the classical Floyd-Warshall's shortest path algorithm can be adopted.

C. Temporal Rigidity and Transitive Relation

1) Temporal Rigidity of Services: The temporal distance of two services represents the execution lags of functionalities to which a business process instance is committed, and it reflects the compactness between these two services. To quantify temporal lags between services, *TDG* is converted into a distance matrix, where $D(i,\, j)$ is the temporal distance between services i and j. Temporal constraints between any pair of adjacent services are discrepant, and each value in the matrix represents the shortest temporal distance between them. We adopt the relative temporal relationship to evaluate inter-service emergency, where the relative rigidity between services is calculated based on distance matrices as follows:

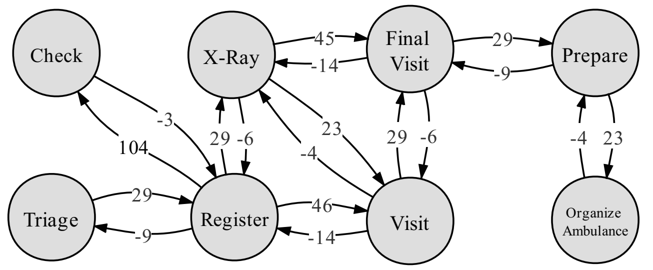


Figure 3. *TDG* corresponds to Figure 2, where nodes are services and edges indicate temporal weight in terms of temporal constraints.

$$Rig(i,j) = \frac{1}{1 + D(i,j) + D(j,i)} \tag{2}$$

where $D(i,\, j)$ and $D(j,\, i)$ are the shortest distances in the temporal matrix from service i to service j and the reverse, respectively. Therefore, $Rig(i,\, j)$ is a relative value in [0, 1]. If service i and service j are two completely rigid services, $Rig(i,\, j)$ = 1, at the opposite extreme, whereas $Rig(i,\, j)$ = 0 when they do not have any constraint.

2) Transitive Relations between Services: Transitive relations between separate services are mined by analysing business process instances in large-scale event logs. We adopt the proportion of transitive relations between consecutive services to reveal transmission interaction as follows:

$$Tsr(i,j) = \frac{N(i,j)}{N_{total}} \tag{3}$$

where $N(i,\, j)$ is the occurrence times of two consecutive services, and N_{total} is the number of all connections in the event log. $Tsr(i,\, j)$ is a value between 0 and 1, such that the larger the $Tsr(i,\, j)$ is, the more frequent the transmission between these two services is.

IV. Service Configuration Strategy

After mining the correlation between services, this section proposes to configure services on the edge-cloud collaboration framework, where the factors including service delay and energy consumption are considered in this paper.

A. Edge-Cloud Collaboration Framework System

In fact, a service request can be divided into a series of single-structured services with transitive relations denoted as $\mathcal{M} = \{1, \ldots, M\}$. These services should be configured on edge nodes or the cloud, where the edge-cloud collaboration network includes a cloud server and N edge nodes (denoted $\mathcal{N} = \{1, \ldots, N\}$). Suppose that an edge node can support up to C types of services, where the rest of services are supported by the cloud. There are $\mathcal{K}$ types of different services indexing by $\mathcal{K} = \{1, \ldots, K\}$. For ease of illustration, we denote $h_n = \{h_{n,k}\}_{k=1}^{K}$ as service hosting decisions of a certain edge node n, where $h_n \in \{0,\ 1\}$ is a binomial item to clarify that whether type-k service is supported by edge node n or not. Due to capacity limitation of edge nodes, it is not possible that all services are allocated to edge nodes. Therefore, the service configuration strategy has great impact on the cost of achieving complex service requests owing to transitive relations of adjacent services between physical facilities. l_m^{m+1} is adopted to record service configuration locations of adjacent services m and $m+1$, where $1 \leq m \leq M-1$. There are three possible conditions of l_m^{m+1}: (i) c_1: two services are hosted on the same edge node n_i; (ii) c_2: two services are hosted on separate edge nodes n_i and n_j; and (iii) c_3: one service is hosted by an edge node n_i and another service is configured on the cloud. The composition of separate services for service requests is essentially a collaboration and cooperation between different facilities, where these three situations lead to different models that have significant influence on service delay and energy consumption.

B. Service Delay Model

Service delay consists of two ingredients as computation and communication delays. The computation delay for a certain service is discrepant depending on the processing velocity of different types of nodes presented as follows:

$$D_{comp}(m) = \begin{cases} \mu_k(m)/f_n & \text{if} \quad h_n = 1 \\ \mu_k(m)/f_0 & \text{if} \quad h_n = 0 \end{cases} \tag{4}$$

where $\mu_k(m)$ refers to the required number of CPU cycles for a certain service, f_n and f_0 are the CPU frequency of edge nodes and cloud, respectively. Note that in general, $f_0 > f_n$ due to the fact that computing capacity and processing rate of the cloud are greater than edge nodes.

Note that transmission delay of two adjacent services on the same physical node can be neglected. The transmission delay between different nodes is computed as follows:

$$D_{comm}(m, m+1) = \begin{cases} 0, & \text{if} \quad l_m^{m+1} = c_1 \\ \frac{\tau_k(m)}{w_n} & \text{if} \quad l_m^{m+1} = c_2 \\ \frac{\tau_k(m)}{w_0} + t_0 & \text{if} \quad l_m^{m+1} = c_3 \end{cases} \tag{5}$$

where $\tau_k(m)$ refers to the data size of the certain service, and w_n (or w_0) represents the transmission bandwidth between edge nodes (or cloud). Since the magnitude of the transmission distance from edge nodes to the cloud exceeds that from an edge node to another, inherent transmission delay of two edge nodes can be ignored without any influence, and t_0 is the round-trip time from edge nodes to the cloud.

C. Energy Consumption Model

The energy consumption for service requests includes two parts: (i) energy consumption for computing each service, and (ii) energy required for communication between different nodes. Generally, energy consumption for computing a service on an edge node is calculated as follows:

$$E_{comp}(m) = E_0 + e_n \times \mu_k(m) \tag{6}$$

where E_0 is the energy consumption for maintaining the liveness of the node. $e_n = cf_n^2$ is the unit energy consumption consumed by the operation of a CPU cycle, which is influenced by the CPU architecture parameter c and CPU frequency of edge nodes f_n [3].

Communication energy consists of transmitting and receiving parts, where energy consumption of transmission and reception of a packet with $\tau_k(m)$ are calculated as follows:

$$\begin{aligned} &E_{comm}(m, m+1) = E_{Tx}(\tau_k(m), d) + E_{Rx}(\tau_k(m)) \\ &= \begin{cases} 0 & \text{if} \quad l_m^{m+1} = c_1 \\ 2 \times E_{elec} \times \tau_k(m) & \text{if} \quad l_m^{m+1} = c_2 \\ E_{elec} \times \tau_k(m) + \epsilon_{amp} \times \tau_k(m) \times d^p & \text{if} \quad l_m^{m+1} = c_3 \end{cases} \end{aligned} \tag{7}$$

where E_{elec} is the energy consumption constant for the transmission and reception of data packets. ϵ_{amp} is the constant of transmitting amplifier. d is transmission distance and p is the attenuation index of transmission. The distance between edge nodes is not considered because of their geographical proximity.

D. Problem Modelling

The objective is to optimize service configurations through planning and allocating related services upon edge nodes and/or the cloud, where temporal constraints and transitive relations are considered. This problem can be transformed into a constrained multi-objective optimization with two subproblems (**SP1** and **SP2**) as follows:

- *Objective Functions*:

1) **SP1**: $\min_{\forall m \in M} \sum_{l_m^{m+1}=c_i} Rig(m,\, m+1) \times p_i$
2) **SP2**: $\min_{\forall m \in M} \sum_{l_m^{m+1}=c_i} Tsr(m,\, m+1) \times \tau_k(m) \times p_i$

- *Constraints*:

1) $\sum_k h_{n,k} \leq C$
2) $|D_{comm}(m) + D_{comp}(m,\, m+1)| \in [t_{min}^{m,m+1},\, t_{max}^{m,m+1}]$
3) $E_{comm}(m) + E_{comp}(m,\, m+1) \leq E_{rsd}^{n}$

where i = 1, 2, 3 in objective functions, p_i is the parameter for link relations that represent the effect on the transmission process. The proportion of transmission bandwidths between nodes is used as normalization variable for p_i. Generally, the first constraint is for the hosting capacity of each edge node, which can not exceed its maximum ability. The second constraint means that service delay for any two adjacent services should be within the interval of corresponding temporal constraints. The energy consumption constraint of edge nodes is presented at the third constraint. Only when the residual energy of edge nodes is sufficient with respect to energy requirement of the hosted service provisioning, the edge node can unceasingly support subsequent service computation and facility transmission.

E. Optimization Algorithm for Service Configuration

An Improved Non-domained Sorting Genetic Algorithm *INSGA-II* is proposed to solve the aforementioned constrained multi-objective optimization problem, where a chromosome corresponds to a service configuration strategy, and the aggregation of chromosomes comprises a population. Specifically, *INSGA-II* are presented by Algorithm 2. Some parameters are initialized firstly (line 1). A population $P_{initial}$ is initialized through configuring services on edge nodes and the cloud randomly. $P_{initial}$ is regarded as the first parent generation P_{prt} (line 2). Each individual in the current population is ranked by non-dominated sorting mechanism as presented at Algorithm 3 (line 3). A new offspring population is produced by crossover and mutation operations (lines 4-5). Individuals in P_{prt} and P_{ofs} are combined into a new population P_{mrg}, and P_{ofs} is transferred to P_{prt} (line 7). Algorithm 3 is invocated on P_{mrg} to obtain the rank of each individual $R(x_i)$, the set of individuals sorted by obtaining rank levels P_{rank} and the maximum rank level MAX_{rank} (line 8). Each individual is assigned to the corresponding rank. A new P_{ofs} is selected in terms of the priority of ranks. If the number of all individuals with the same priority is less than the existing residual capacity, they can be put into P_{ofs}. Otherwise, the values of two sub-objective equations are calculated for all the individuals with corresponding priority. These objective values are sorted through a simple sorting operation to obtain two sequences $P1$ and $P2$. The crowding distance of each individual $d(x_i)$ is computed, which aims to effectively maintain the diversity of solutions. The selection is made according to $d(x_i)$ until the size of P_{ofs} equals the individual number of population n (lines 10-26). Crossover and mutation operations are applied to produce a new offspring population P_{ofs} (lines 27-28). This process is iteratively performed until the maximum number of iterations is reached and a set of strategies in P_{ofs} is obtained. An approximate optimal solution x_{opt} is selected among the individuals with the highest priority in the population based on the sum of distance between

Algorithm 2 INSGA-II : An Improved Non-dominated Sorting Genetic Algorithm II

```
Require:
    - $P_{initial}$ : initial populations including $n$ individuals.
    - $MAX_{gen}$ : the maximum number of iterations.
    - $n$ : the number of individuals in a population.
Ensure:
    - $x_{opt}$ : the optimal service placement strategy.
1:  $P_{ofs} \leftarrow \varnothing$, $D \leftarrow \varnothing$, $i \leftarrow 0$
2:  $P_{initial} \leftarrow \{x_1, \ldots, x_n\}$, $P_{prt} \leftarrow P_{initial}$
3:  $P_{rank} \leftarrow NdSoring(P_{prt})$
4:  $P_{ofs} \leftarrow P_{rank} \cup CrossoverOperation$
5:  $P_{ofs} \leftarrow P_{ofs} \cup MutationOperation$
6:  while $looptime < MAX_{gen}$ do
7:      $P_{mrg} \leftarrow P_{ofs} \cup P_{prt}$, $P_{prt} \leftarrow P_{ofs}$, $P_{ofs} \leftarrow \varnothing$
8:      $R(x_i), P_{rank}, MAX_{rank} \leftarrow NdSoring(P_{mrg})$
9:      $rank \leftarrow 1$
10:     while $rank < MAX_{rank}$ do
11:         if $|P_{rank}\{R(x_i) = rank\}| \leq n - |P_{ofs}|$ then
12:             $P_{ofs} \leftarrow P_{ofs} \cup P_{rank}\{R(x_i) = rank\}$
13:             $rank \leftarrow rank + 1$
14:         else
15:             for $\forall x_i \in P_{rank}\{R(x_i) = rank\}$ do
16:                 $P1 \leftarrow P1 \cup \mathbf{SP1}(x_i)$, $P2 \leftarrow P2 \cup \mathbf{SP2}(x_i)$
17:             end for
18:             $P1 \leftarrow P1 \cup SortingOperation$
19:             $P2 \leftarrow P2 \cup SortingOperation$
20:             $d(x_i) \leftarrow |P1(x_{i+1}) - P1(x_{i-1})| + |P2(x_{i+1}) - P2(x_{i-1})|$
21:             $D \leftarrow D \cup d(x_i)$, $D \leftarrow D \cup SortingOperation$
22:             while $|P_{ofs}| \leq n$ do
23:                 $P_{ofs} \leftarrow P_{ofs} \cup \{D(i)\}$, $i \leftarrow i + 1$
24:             end while
25:         end if
26:     end while
27:     $P_{ofs} \leftarrow P_{ofs} \cup CrossoverOperation$
28:     $P_{ofs} \leftarrow P_{ofs} \cup MutationOperation$
29: end while
30: $P_{rank} \leftarrow NdSoring(P_{ofs})$
31: for $\forall x_i \in P_{rank}\{R(x_i) = 1\}$ do
32:     $f(x_i) = (\frac{(SP1(x_i) - MIN_{SP1})}{MIN_{SP1}})^2 + (\frac{(SP2(x_i) - MIN_{SP2})}{MIN_{SP2}})^2$
33: end for
34: $x_{opt} \leftarrow MIN_{f(x_i)}$
```

objective values of certain service configuration strategy and absolute optimal objective values (lines 31-34).

Algorithm 3 NdSorting : Non-dominated Sorting

```
Require:
    - $P$ : a populations produced by Algorithm 2.
Ensure:
    - $R(x_i)$ : the rank for each individual $x_i$.
    - $P_{rank}$ : individuals have been obtained rank levels.
    - $MAX_{rank}$ : the maximum of rank levels.
1:  $P_{rank} \leftarrow \varnothing$, $rank \leftarrow 1$
2:  while $P \neq \varnothing$ do
3:      for $\exists x_i \in P$ do
4:          while $x_j \in P, i \neq j$ do
5:              if $\nexists$ (SP1$(x_i) \geqslant$ SP1$(x_j)$ && SP2$(x_i) \geqslant$ SP2$(x_j)$) then
6:                  $R(x_i) \leftarrow$ rank, $P \leftarrow P - \{x_i\}$, $P_{rank} \leftarrow P_{rank} \cup \{x_i\}$
7:              end if
8:          end while
9:      end for
10:     $rank \leftarrow rank + 1$
11: end while
12: $MAX_{rank} \leftarrow rank$
```

V. Implementation and Evaluation

A. Experiment Settings

Table I
Parameter Settings in Our Experiments.

Parameter	Value
$\mu_k(m)$	[50, 200] MB
f_n	4 GHz
f_0	8 GHz
$\tau_k(m)$	[10, 100] MB
w_n	1 Mbps
w_0	1 Mbps
t_0	2 s
E_0	1 J
E_{elec}	5×10^{-4} J/bit
ϵ_{elec}	1×10^{-6} J/(bit $\times m^2$)
p	2

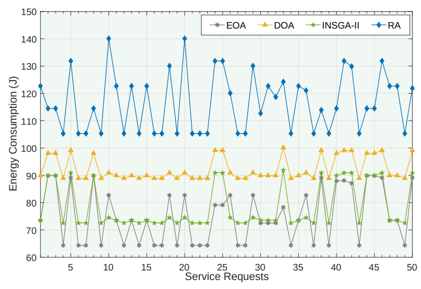
A prototype has been implemented in Java program. The parameter settings for our experiments are presented in Table I. Specifically, we adopt an event log, which is publicly accessible [5], for our experiment purpose. This event log contains 100 thousand traces. 8 to 9 events are included in each trace, where there are a total of *894,708* items. We build service requests to verify our method through leveraging some existing business process instances in the event log.

B. Experimental Evaluation

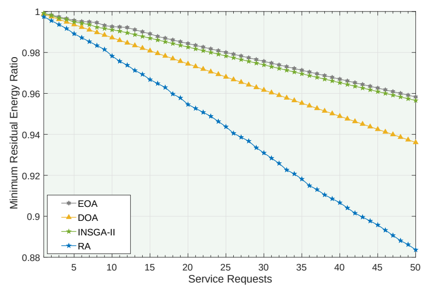
1) Performance Evaluation of Our INSGA-II: Based on the dataset presented in Section V-A, *50* service requests are extracted from the event log for eliminating the impact of randomness. There are three benchmark methods compared with our proposed approach including: (i) Random Allocation (*RA*), (ii) Delay Optimal Approach (*DOA*) that aims to minimize service delay regardless of energy consumption, and (iii) Energy Optimal Approach (*EOA*) that primarily concerns with energy consumption. Energy consumption for *EOA*, *DOA*, *INSGA-II* and *RA* is shown at Figure 4(a). Generally, *INSGA-II* and *EOA* can save more energy than other two approaches *DOA* and *RA*. The minimum residual energy ratio for these four different approaches is presented by Figure 4(b). The minimum residual energy of edge nodes is standardized to form a ratio from 0 to 1, which represents the proportion of the remaining power of edge nodes. This figure shows that *EOA* and *INSGA-II* have a larger minimum residual energy and have a small difference. This means that *EOA* and *INSGA-II* have more significant effect on preserving energy and they optimize energy efficiency through configuring services reasonably compared to other two approaches *DOA* and *RA*. The logarithm of residual energy variance of all edge nodes is presented by Figure 4(c). These curves are presented in logarithm for observing the minor difference of *EOA* and *INSGA-II* obviously. Generally, the variance expresses the unbalance of energy consumption for edge nodes in the whole network. The variances of *DOA* and *RA* are in a large magnitude than other two approaches, which means that they are relatively poor on maintaining load balancing and thus prolonging lifetime of the network.

Figure 4(d) shows the service delay for service requests when adopting *EOA*, *DOA*, *INSGA-II* and *RA*. Service delay is discrepant depending on the dependent-structure of service requests. This figure shows that *DOA* and *INSGA-II* perform better in terms of temporal delay, approximately between 200 s to 300 s. Compared with *DOA* and *INSGA-II*, *EOA* has relatively poor effect on delay, where *RA* is the longest and most unstable approach due to the randomness of its configuration strategy. Generally, our proposed approach *INSGA-II* is similar to *EOA* wherein energy consumption for improving energy preservation ability. Besides, *INSGA-II* is effective in reducing service delay like *DOA*, and thus optimizing the deferred cycles.

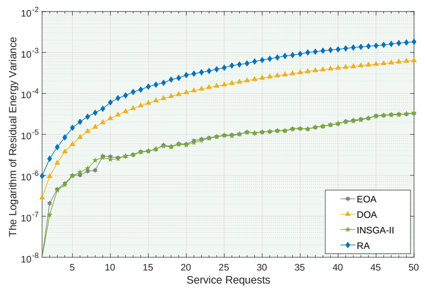
2) Impact of Hosting Capacity: The impact of various hosting capacity for edge nodes to complete 50 service requests on energy consumption and minimum residual energy is shown at Figure 4(e) and Figure 4(f). The hosting capacity of edge nodes is set to a value ranging from 1 to 4 with an increment of 1. The result shows that, when the capacity of edge nodes is 1, the average energy consumption of service requests is about 85 J. With the increasing number of hosting capacity for edge nodes, data transmission between different nodes is gradually replaced. Due to the augment of services in single nodes, energy consumption is reduced. By contrary, the reduction of capacity leads to an increase of the minimum residual energy ratio for edge nodes. Part



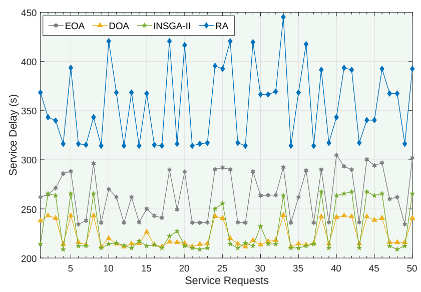
(a)



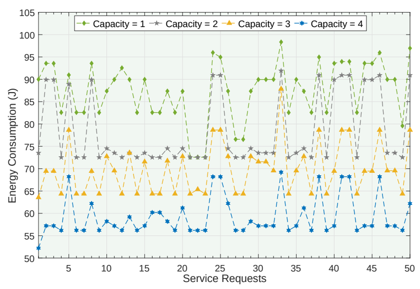
(b)



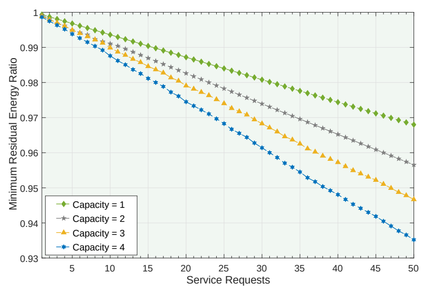
(c)



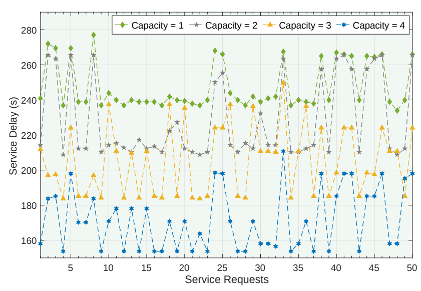
(d)



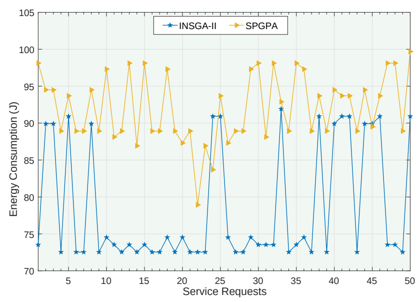
(e)



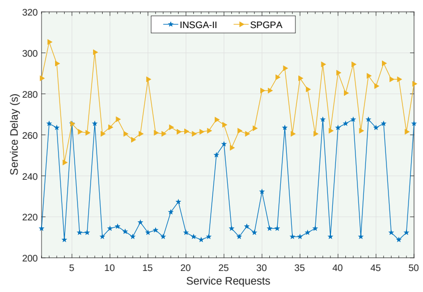
(f)



(g)



(h)



(i)

Figure 4. (a) The energy consumption, (b) The minimum residual energy ratio, (c) The logarithm of residual energy variance, (d) The service delay, for *EOA*, *DOA*, *INSGA-II* and *RA* when algorithms are executed to implement 50 service requests. (e) The energy consumption, (f) The minimum residual energy ratio, (g) The service delay, for the capacity of edge nodes ranging from 1 to 4 adopted to *INSGA-II*. (h) The energy consumption, (i) The service delay, for *INSGA-II* and *SPGPA* when algorithms are executed to implement 50 service requests.

of services are hosted on edge nodes, and the others are configured on the cloud. This strategy can decrease the energy consumption of edge nodes. As shown in Figure 4(g), service delay is floating up and down based on the hosting capacity. Service delay for the capacity of 1, 2, 3 and 4 are about 250 s, 220 s, 200 s and 170 s, respectively. Generally, when more services are configured on edge nodes, the delay of service requests can be significantly reduced.

3) Comparison with SPGPA [8] for the Optimization of Service Placement: This section presents the comparison of our technique *INSGA-II* with a service placement based on graph partition approach (denoted *SPGPA*) [8]. Generally, *SPGPA* introduces a service placement strategy inspired by complex network in the fog environment, where applications are mapped to the community of fog devices. Multiple services of each application are distributed on devices, so as to improving service availability and the satisfaction of quality of services. As many interrelated services as possible are allocated on fog devices that are close to users. In [8], there are two stages. The first is to map applications to an appropriate fog community. However, this step can be omitted in our technique since all service requests are executed in the same region. Each application is divided into multiple dependent services that are configured by prioritizing the placement of interrelated services in fog devices. In our problem model, the correlation between services can be understood as the composition of temporal rigidity and transitive relations. We assume that these two factors are of equal importance and a total transfer coefficient is formed, and different service subsets are configured on edge nodes through matching their capacity. The remaining services are offloaded to the cloud, which forms a service placement policy for processing resource allocation.

The energy consumption and service delay of 50 service requests by adopting *INSGA-II* and *SPGPA* are shown at Figure 4(h) and 4(i). 50 service requests are tested, and the result shows that compared with *SPGPA*, our proposed approach *INSGA-II* is more efficient on energy consumption and service delay. These figures show that *INSGA-II* is more appropriate to be applied. In fact, as presented by Algorithm 3, *INSGA-II* considers two objectives synthetically, instead of integrating multiple objective functions by assigning a weight value to each object. This means that our approach is more appropriate to maintain the diversity of solutions, and thus retaining elite individuals that have greater probability of designing a relatively optimal service configuration.

VI. Related Work and Comparison

To complement cloud computing and optimize the network performance, edge/fog computing has been proposed as an efficient paradigm through allocating services at the network edge. Authors develop a novel online distributed algorithm to adaptively configure services upon fog nodes or the cloud [3]. This approach aims to minimize time-average computation delay while satisfying long-term energy constraints of fog nodes. Li *et al.* propose a task offloading and resource purchasing technique in an edge-cloud collaborative system [9], which optimizes task offloading, while satisfying maximum acceptable delays and minimizing service costs. Generally, these techniques focus mainly on supporting single-structure atomic tasks, and guaranteeing task completion within acceptable latency through reasonably configuring service functionalities. However, the configuration of composite services is not explored extensively.

Due to the complexity of domain applications, structure-complex tasks, which are usually represented in terms of composite services, should be achieved through configuring services upon edge-cloud collaboration networks. In [10], authors present a two-step resource management technique to allocate locality sensitive requests to fog and cloud devices. Requests are distributed to appropriate fog nodes so that the quality of services is guaranteed. Data link cost and delay are main factors that are considered besides the efficiency of fog resource utilization. In [4], a scheduling framework is proposed to minimize the planning cost for a deadline constrained energy-aware workflow. Workflow applications are sequenced based on the emergency of deadlines and achieved through optimizing geo-distributed data transmission. Generally, current techniques consider mainly energy consumption and service latency, and lay emphasis on the minimization of execution time in order to satisfy global deadlines. However, temporal consistency between services are not the focus to be guaranteed. In fact, temporal constraints is a first-class citizen between internal services. In this regard, this paper proposes to optimize service configuration when considering temporal constraints and decreasing the network performance in edge-cloud collaboration networks.

VII. Conclusion

This paper proposes to configure services in complementary edge-cloud networks, where temporal constraints between internal services are guaranteed. Specifically, a timed service-based process model is proposed, where temporal constraints specified upon relations between services are generated through our proposed *DQTI* mechanism. To configure services properly at (i) the network edge through discovering appropriate hosting edge nodes, or (ii) the cloud in terms of task offloading, service configuration is reduced into the constrained multi-objective optimization problem, and an optimal solution is generated through the *INSGA-II* algorithm. Extensive experiments have been conducted, and evaluation results demonstrate that our approach performs better than the state-of-art's techniques.

References

[1] Y. Li, S. Xia, M. Zheng, B. Cao, and Q. Liu, "Lyapunov optimization based trade-off policy for mobile cloud offloading in heterogeneous wireless networks," *IEEE Transactions on Cloud Computing*, vol. PP, pp. 1–1, 2019.

[2] W. Du, T. Lei, Q. He, W. Liu, Q. Lei, H. Zhao, and W. Wang, "Service capacity enhanced task offloading and resource allocation in multi-server edge computing environment," in *IEEE International Conference on Web Services*, 2019.

[3] L. Chen, P. Zhou, L. Gao, and J. Xu, "Adaptive fog configuration for the industrial internet of things," *IEEE Transactions on Industrial Information*, vol. 14, no. 10, pp. 4656–4664, 2018.

[4] X. Li, W. Yu, and R. R. J. Zhu, "Energy-aware cloud workflow applications scheduling with geo-distributed data," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 2020.

[5] F. Mannhardt, M. de Leoni, H. A.Reijers, and W. M. P. van der Aalst, "Data-driven process discovery - revealing conditional infrequent behavior from event logs," *Advanced Information Systems Engineering*, vol. 10253, pp. 545–560, 2017.

[6] S. Ma and J. L. Hellerstein, "Mining partially periodic event patterns with unknown periods," in *IEEE International Conference on Data Engineering*, 2001.

[7] R. Dechter, I. Meiri, and J. Pearl, "Temporall constraint networks," *Artificial Intelligence*, vol. 49, pp. 61–95, 1991.

[8] I. Lera, C. Guerrero, and C. Juiz, "Availability-aware service placement policy in fog computing based on graph partitions," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3641–3651, 2019.

[9] R. Li, Z. Zhou, X. Chen, and Q. Ling, "Resource price-aware offloading for edge-cloud collaboration: A two-timescale online control approach," *IEEE Transactions on Cloud Computing*, vol. PP, pp. 1–1, 2019.

[10] O. Fadahunsi and M. Mahewaran, "Locality sensitive request distribution for fog and cloud servers," *Service Oriented Computing and Applications*, vol. 13, no. 2, pp. 127–140, 2019.