Reinforcement Learning Based Architectures for Dynamic Generation of Smart Home Services

Mingming Qiu, Elie Najm, Rémi Sharrock, Bruno Traverson

▶ **To cite this version:**

Mingming Qiu, Elie Najm, Rémi Sharrock, Bruno Traverson. Reinforcement Learning Based Architectures for Dynamic Generation of Smart Home Services. 2022. hal-03860833

HAL Id: hal-03860833

https://telecom-paris.hal.science/hal-03860833

Submitted on 18 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reinforcement Learning Based Architectures for Dynamic Generation of Smart Home Services

Mingming Qiu
Télécom Paris, EDF R&D
Palaiseau, France
Mingming.Qiu@telecom-paris.fr

Elie Najm
Télécom Paris
Palaiseau, France
Elie.Najm@telecom-paris.fr

Rémi Sharrock
Télécom Paris
Palaiseau, France
Remi.Sharrock@telecom-paris.fr

Bruno Traverson
EDF R&D
Palaiseau, France
Bruno.Traverson@edf.fr

***Abstract*—A smart home system is realized by implementing various services. However, the design and deployment of smart home services are challenging due to their complexity and the large number of connected objects. Existing approaches to the smart home system to create services either require complex input from the inhabitant or can only work if the inhabitant specifies regulation solutions rather than targets. In addition, smart home services may conflict if they access the same actuators. Learning methods to dynamically generate smart home services are promising ways to solve the above problems. In this paper, depending on the ability to consider the composition of services and their mutual influence, we propose several reinforcement learning-based architectures for a smart home system to dynamically generate services. The expected advantages are, first, that the smart home services can propose the states of the actuators by considering the target values of the controllable environment states given by the inhabitant or by interacting with the inhabitant in a simple and natural way; and second, that there is no conflict between these propositions. We compare the performance of the proposed architectures using several simulated smart home environments with different services and select the architectures with the best performance concerning our predefined metrics.**

***Index Terms*—service orientation, reinforcement learning, multi-agent, smart home**

I. Introduction

Implementations of various services contribute to the realization of a smart home system. Depending on the problem and the corresponding proposed solutions, there are different definitions of services. For example, in [1], each device is considered as a service. In [2], a service is implicitly expressed as an action that can be performed if certain conditions are met. In [3], a service is described as a target of functions that are triggered under certain conditions. In our work, we define that each smart home service manages a controllable environment state by instructing actuators to perform appropriate actions with respect to environment state values detected by the corresponding sensors.

Many approaches have been proposed for developing smart home systems by creating various services. Most of these approaches belong to the knowledge-based approaches. However, despite the capability to provide understandable services, the knowledge-based approaches usually require manual inputs from the inhabitant. And the inputs become complicated when the desired services are complex or there are many connected objects. In addition, this kind of approaches cannot create services if the inhabitant only specifies the target value for the controllable environment state, but not the regulation solution. And conflicts may arise when services request the same actuators to perform different actions. Furthermore, it is important to consider the inhabitant's reactions when designing a user-friendly smart home system according to [4]. With the existing work on transforming services created by learning methods into understandable ones [5], learning methods for dynamically generating smart home services are promising solutions to overcome the above problems and create a user-friendly smart home. In this paper, we propose several learning method-based architectures with a collective name SHOMA (Smart HOme-based Multi-services Architecture). Each service in these architectures is modeled based on reinforcement learning (RL) [6]. Since the basic idea of RL is that an artificial agent learns the system's behavior patterns by interacting with the environment, it can be used to dynamically generate smart home services considering the inhabitant's reactions to the proposed actuators' actions.

In the rest of the paper, Section II presents existing work on the development of smart home systems through the creation of various services. Section III introduces how RL is used to dynamically generate a single smart home service. Section IV presents the proposed architectures to dynamically generate multiple smart home services. Section V describes the simulated environments used to evaluate the proposed architectures. Section VI analyzes the results of the experiments and selects the architectures with the best performance. Section VII summarizes the work and provides interesting perspectives.

II. Related work

Many applications try to implement a smart home system by creating multiple services. For example, in [7], the proposed smart home system contains low and high levels. The low level includes the execution of various devices, while the high level includes system modeling, rule transformation, and system reasoning. System modeling uses the semantic web to model the environment and ECA (Event-Condition-Action)-based descriptions to simulate the operation of services. Rule transformation is about converting ECA into executable se-

978-1-5386-5541-2/18/$31.00 ©2018 IEEE

mantic rules, e.g., SWRL (Semantic Web Rule Language). And reasoning is about deriving new facts based on SWRL. These new facts are converted into commands and sent to the devices for execution. However, in this method, the inhabitant has to manually design rules and there is no guarantee that the services are conflict-free. [8] proposes an application called RHDL (Recognition Habit of Daily Living) to generate the habitual smart home service. RHDL uses a clustering algorithm to learn the inhabitant's habitual behavior which is then converted into rules. These rules are used together with the knowledge representation to derive new facts, and the new facts are translated into control commands for devices that change the environment state values. However, RHDL cannot consider the inhabitant's (dis)satisfaction with a proposed smart home service. [9] proposes a platform called Synapse using four parts to create services. First, Synapse collects information from the real world through the sensor event collection part and converts the information into useful contexts. Then, the service control part controls various devices to provide services. Next, the Synapse core part uses the HMM (Hiden Markov Model) to model the relationship between services and events collected by sensors. Finally, the services recommended by the HMM are listed in the last part of a user interface where users can browse and launch the available services and update the recommended services. However, this work also does not consider the inhabitant's reactions to the recommended services.

In this work, we propose RL-based SHOMA architectures for a smart home system to dynamically generate services. This system can reduce the manual intervention of the inhabitant, interact more naturally with the inhabitant, and ensure that there are no conflicts between the generated services.

III. Dynamic Generation of One Smart Home Service

Before presenting the SHOMA architectures for modeling a smart home system to dynamically generate multiple services, we first show how a single smart home service can be dynamically generated based on RL.

According to the proposed service definition, we know that a service is implemented by proposing actuators' actions according to the observable environment states detected by the sensors. Moreover, to allow a service to interact with the inhabitant during its implementation, it should be able to consider the inhabitant's (dis)satisfaction or the target value specified by the inhabitant. Therefore, RL is used to model a single smart home service that can be dynamically generated by the dynamic propositions of the actuators' actions. The corresponding principle is shown in Fig.1.

A smart home service z contains an interpreter, an RL algorithm, and a policy. An interpreter is used, first, to select states S from observable environment states O, where S is used as input to the RL algorithm; second, to generate a reward r using a reward function that considers the input S and the inhabitant's (dis)satisfaction. The RL algorithm proposes action quality values for each possible action of each actuator. In our study, Q-learning [10] is used. The

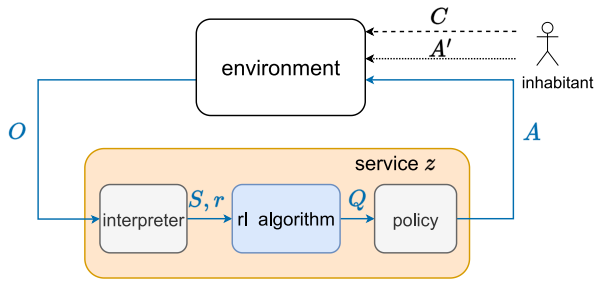

Fig. 1. Leverage RL to dynamically generate a single smart home service

policy is used to select the action that each actuator will perform. The process is as follows: After the sensors detect the observable environment states O, the interpreter selects the RL algorithm's input states S from O. The action quality values Q for each action of each actuator are then proposed by the RL algorithm, and the policy uses a function or rule, e.g., the ϵ-Greedy function, to select the action for each actuator, where A is the set of selected actions for all actuators. The execution of the actuators contributes to the updating of the controllable environment state within O. In this situation, the inhabitant has two options to provide his input: He can have the actuators perform different actions A' to achieve the desired controllable environment state. Changing the actions of the actuators from A to A' contributes to the computation of the reward by the interpreter. Or, he can directly specify the target value C of the controllable environment state, and the interpreter can compute the reward by measuring the difference between the updated controllable environment state and C. Then, based on the input of the inhabitant and the updated O, the interpreter selects the RL algorithm's input states S from O and calculates the reward r. Both S with the original and updated values and r are used to train the RL algorithm. The updated RL algorithm is then used to repeat the above process.

IV. Proposed Architectures for Dynamic Generation of Multiple Smart Home Services

To better illustrate the principle of SHOMA architectures, we first present a smart home system with three services. The process of creating a smart home system with more or less than three services using SHOMA architectures is the same as the process with three services presented in this section. We refer to the three services as z_1, z_2, and z_3. They belong to different types of services and can represent any specific smart home services. The actuators associated with each service are: (1) $\{d_1, d_2, d_3\}$ for z_1; (2) $\{d_2, d_4, d_5\}$ for z_2; (3) $\{d_2, d_3, d_6\}$ for z_3, where d_2 is shared by the three services and d_3 is shared by z_1 and z_3. In addition, the environment states that each service considers are: (1) $O_{z_1} = \{o_{z_1,1}, o_{z_1,2}, \cdots\}$ for z_1; (2) $O_{z_2} = \{o_{z_2,1}, o_{z_2,2}, \cdots\}$ for z_2; (3) $O_{z_3} = \{o_{z_3,1}, o_{z_3,2}, \cdots\}$ for z_3. The environment states converted by the interpreter and used as input to the RL algorithm in each service are expressed as S_{z_1}, S_{z_2}, and S_{z_3}. The corresponding target or preferred values for these inhabitant-specified controllable environment states are

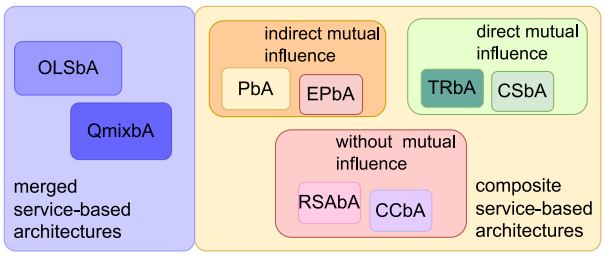

Fig. 2. Classification of SHOMA architectures

denoted as C_{z_1}, C_{z_2}, and C_{z_3}, and $C = C_{z_1} \cup C_{z_2} \cup C_{z_3}$. Assuming that z_1, z_2, and z_3 are individually implemented by respecting the principle in Fig.1, we have the following modules: interpreter RF_{z_1}, rl algorithm RL_{z_1} and policy PO_{z_1} for z_1; interpreter RF_{z_2}, RL algorithm RL_{z_2} and policy PO_{z_2} for z_2. The same is true for z_3.

Using the three services and their associated modules in different ways, we define eight architectures in SHOMA based on the concept of multi-agent RL (MARL) [11] to model a smart home system to dynamically generate services: One Learning System-based Architecture (OLSbA), Qmix-based Architecture (QmixbA), Remove Shared Actuators-based Architecture (RSAbA), Common Controller-based Architecture (CCbA), Priority-based Architecture (PbA), Equal Priority-based Architecture (EPbA), Total Reward-based Architecture (TRbA), and Context Sharing-based Architecture (CSbA). As shown in Fig.2, these architectures can be divided into two types: "merged service-based architectures" and "composite service-based architectures". The difference between them depends on whether only a single service is used to model the entire smart home system. For "composite service-based architectures", the architectures can be further divided into three subtypes: "indirect mutual influence", "direct mutual influence" and "without mutual influence". The difference between them lies in how the shared actuators' actions are determined. For "indirect mutual influence" and "direct mutual influence", each service proposes actions for the shared actuators under the indirect or direct influence of other services. In "without mutual influence", the actions of the shared actuators are determined by one service or controller, thus eliminating the influence between services.

A. Merged service-based architectures

The first architecture category is the merged service-based architecture, where all services of the smart home system are merged as a single service to regulate multiple controllable environment states. Two SHOMA architectures belong to this category: OLSbA and QmixbA.

1) One Learning System based Architecture: The OLSbA shown in Fig.3 models the entire smart home system as a single service z. In the initial time step, the interpreters $\mathsf{RF}_{z_1}, \mathsf{RF}_{z_2}$ and RF_{z_3} transform the observable environment states O_{z_1}, O_{z_2} and O_{z_3} into states S_{z_1}, S_{z_2} and S_{z_3}, which are used as input to the RL algorithm. And the algorithm generates action quality values for all possible actions of all available

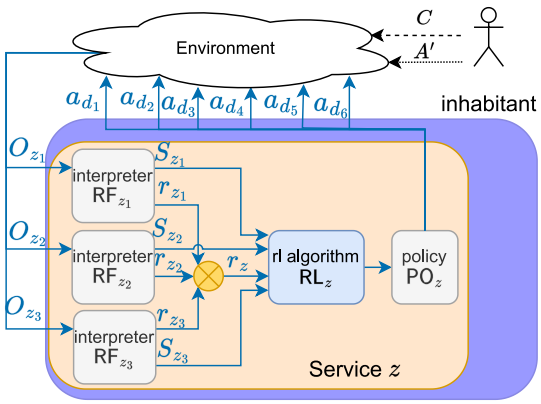

Fig. 3. Structure of OLSbA

actuators. Then the policy PO_z uses a function to decide the action for each actuator, e.g., the greedy policy selects the action with the highest action quality value for the actuator. The controllable environment states update their values as the actuators perform the selected actions. Based on the updated controllable environment states and the inhabitant's target value C for the controllable environment state or actions A', $\mathsf{RF}_{z_1}, \mathsf{RF}_{z_2}$ and RF_{z_3} calculate the reward r_{z_1}, r_{z_2} and r_{z_3}. The total reward $r_z = r_{z_1} + r_{z_2} + r_{z_3}$ is used together with the environment states before and after the update to train the RL algorithm in z. Then, the updated RL algorithm or the updated service z is used to repeat the above process.

2) Qmix based Architecture: We use the principle of [12] to implement the QmixbA shown in Fig.4. To determine the actions of the actuators, QmixbA introduces a learning system called Qmix that takes the action quality value outputs $Q_{z_1}, Q_{z_2}, Q_{z_3}$ from the RL algorithm $\mathsf{RL}_{z_1}, \mathsf{RL}_{z_2}$ and RL_{z_3} as inputs and a hyper neural network [13] to determine the values of its parameters. Moreover, the total reward $r_z = r_{z_1} + r_{z_2} + r_{z_3}$ is used to determine the learning direction of Qmix and the hyper neural network. Finally, Qmix generates the action quality values for the possible actions of each actuator and a policy is used to select the final action that each actuator will perform.

B. Composite service-based architectures

In composite service-based architectures, each controllable environment state is controlled by a service. Since the implementation of a service is realized through the execution of one or more actuators, certain actuators may be shared by multiple services, leading to potential conflicts. Based on the mechanisms used to decide the states of these shared actuators, "composite service-based architectures" can be further divided into three subcategories: "without mutual influence", "indirect mutual influence", and "direct mutual influence".

1) Without mutual influence: "without mutual influence" refers to architectures where shared actuators are controlled by only one service or controller, which involves two architectures: RSAbA, and CCbA.

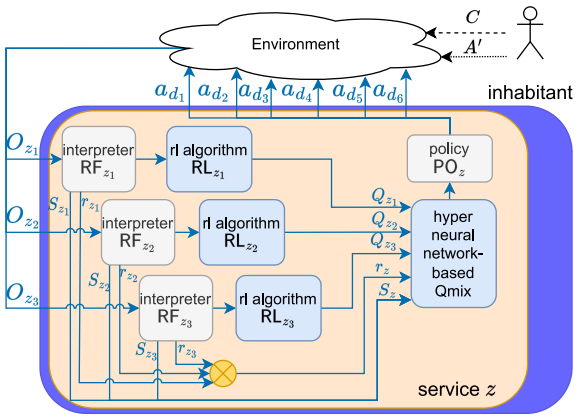

Fig. 4. Structure of QmixbA

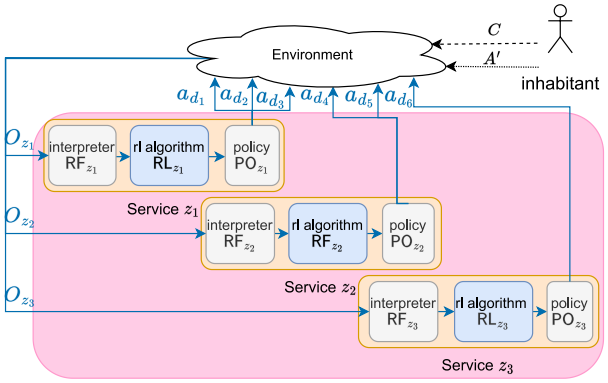

Fig. 5. Structure of RSAbA

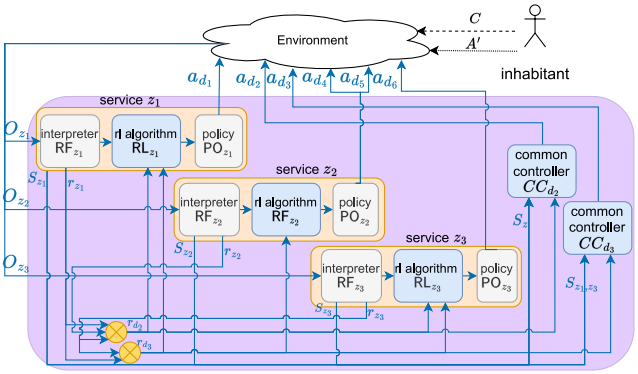

Fig. 6. Structure of CCbA

a) Remove Shared Actuators based Architecture: In RSAbA, only one shared service is maintained to determine the states of the shared actuators. The service that is maintained is the one whose corresponding dynamic characteristic about the controllable environment state is the simplest. In the given example with multiple services, we assume that the order of complexity of each service is: $complexity(z_1) < complexity(z_2) < complexity(z_3)$, therefore, d_2 and d_3 are controlled by z_1. Thus, RSAbA can be expressed in Fig.5.

b) Common Controller based Architecture: CCbA defines a common controller modeled by an RL algorithm for each shared actuator. As shown in Fig.6, a common controller CC_{d_2} is defined for d_2 since d_2 is shared by z_1, z_2, and z_3. CC_{d_2} takes the inputs of the RL algorithms of the shared services $S_z = S_{z_1} \cup S_{z_2} \cup S_{z_3}$ and the total reward $r_{d_2} = r_{z_1} + r_{z_2} + r_{z_3}$ as its input, and proposes the action for the actuator d_2. And for the actuator d_3, a common controller CC_{d_3} is defined since it is shared by z_1 and z_3. It takes the inputs of the RL algorithm of the corresponding shared services $S_{z_1,z_3} = S_{z_1} \cup S_{z_3}$ and the total reward $r_{d_3} = r_{z_1} + r_{z_3}$ as input, and proposes the action for the actuator d_3.

2) Indirect mutual influence: The category "indirect mutual influence" includes architectures where each service proposes states of the shared actuators under the indirect influence of other services. The final states of the shared actuators are determined by considering the propositions of all services. They can influence the states of the shared actuators proposed by each service in the next time step. Therefore, the final states of these shared actuators can be considered as the indirect influence of other services on each service. PbA and EPbA are two architectures of this type.

a) Priority based Architecture: In PbA, a priority calculator modeled by an RL algorithm is defined. Each priority calculator proposes priorities for services that share an actuator. Fig.7 in yellow shows how z_1, z_2 and z_3 can be modeled with PbA: Each service receives its associated observable environment states, which contain the target value of the controllable environment states or action changes of the actuators made by the inhabitant, as input. Then, for non-shared actuators, each service directly proposes the actions that these actuators will perform. For each of the shared actuators, the action quality values of all possible actions proposed by the shared services are multiplied by the priorities of the corresponding shared services to obtain the weighted action quality values. These priorities are proposed by the corresponding priority calculator. The action with the highest quality is then selected for this shared actuator. For example, d_2 is shared by z_1, z_2 and z_3. Each service z uses an RL algorithm to propose action quality values $Q_{d_2}^{z_1}, Q_{d_2}^{z_2}$, and $Q_{d_2}^{z_3}$

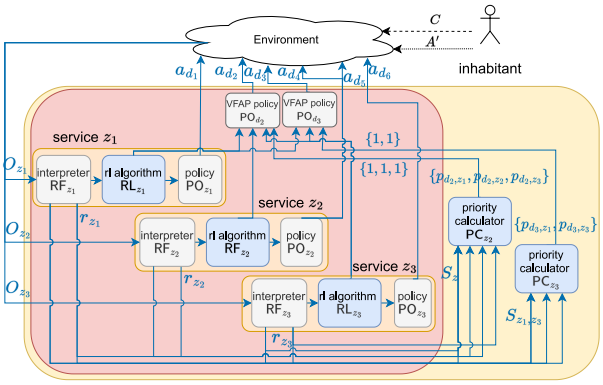

Fig. 7. Structures of PbA (in yellow) and EPbA (in red)

for d_2. A priority calculator PC_{d_2} is defined for d_2. It takes the inputs of the ensemble RL algorithm $S_z = S_{z_1} \cup S_{z_2} \cup S_{z_3}$ as input and proposes three priorities $p_{z_1}^{d_2}$, $p_{z_2}^{d_2}$, $p_{z_3}^{d_2}$ whose sum is one. A policy based on the value function addition principle (abbreviated as VFAP in this paper) introduced in [14] is used to compute the final action quality value Q_{d_2} for d_2:

$$Q_{d_2} = Q_{d_2}^{z_1} \cdot p_{z_1}^{d_2} + Q_{d_2}^{z_2} \cdot p_{z_2}^{d_2} + Q_{d_2}^{z_3} \cdot p_{z_3}^{d_2}, \quad (1)$$

and then selects the action with the highest action quality value in Q_{d_2} as the action that d_2 will perform.

b) Equal Priority based Architecture: The principle of EPbA is shown in red in Fig.7. Instead of computing the priorities of the services sharing the same actuators, EPbA directly summarizes the action quality values of the shared actuators and selects the actions with the highest action quality values by following the VFAP policy. We can also say that all shared services have the same priority with the value of 1 to associate EPbA with PbA. Therefore, the example of using the VFAP policy to calculate the final action quality values for the shared actuator d_2 is:

$$Q_{d_2} = Q_{z_1,d_2} + Q_{z_2,d_2} + Q_{z_3,d_2}. \quad (2)$$

Then, the VFAP policy selects the action with the highest action quality value in Q_{d_2} as the one that d_2 will perform.

3) Direct mutual influence: The direct mutual influence type includes architectures where each service proposes the actions of shared actuators under the direct influence of other services. For example, each service that shares the same actuators considers the total reward which results from adding the rewards of these services. The total reward then directly influences the proposition that each service makes for the state of the shared actuator. Therefore, for each service sharing the same actuators, the total reward can be considered as the direct influence of the other services for each service. TRbA and CSbA are two architectures of this type of SHOMA.

a) Total Reward based Architecture: TRbA, as shown in Fig.8, is a variant of EPbA. Unlike EPbA, TRbA adds the total rewards of services sharing the same actuator to each service's

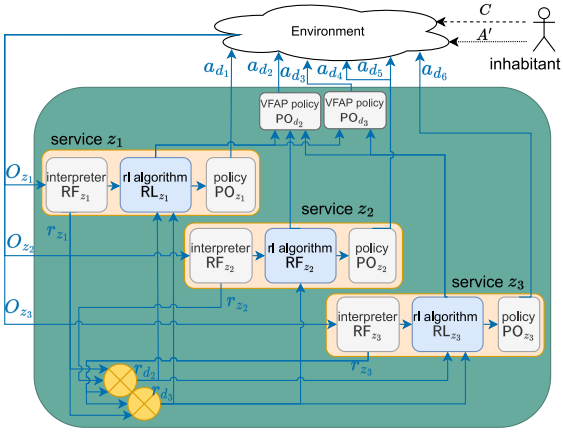

Fig. 8. Structure of TRbA

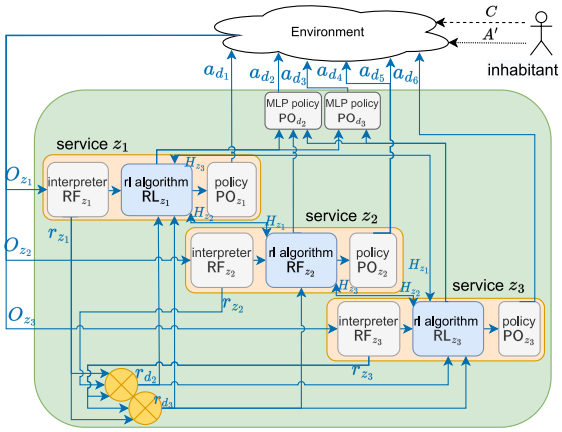

Fig. 9. Structure of CSbA

RL algorithm as an additional reward for determining the states of that shared actuator. For example, d_3 is shared by z_1 and z_3. Therefore, the total reward $r_{d_3} = r_{z_1} + r_{z_3}$ is used as an additional reward and sent to RL_{z_1} and RL_{z_3}. By introducing the total rewards, the states of the shared actuators can be determined by ensuring that all shared services receive high rewards simultaneously.

b) Context Sharing based Architecture: In CSbA, each service takes as input its associated environment states, its hidden states, and the hidden states of other services that use the same actuators. Each hidden state contains past information about the associated service within a short period of time. As shown in Fig.9, service z_1 considers its environment states O_{z_1}, its hidden states H_{z_1}, and the hidden states H_{z_2} and H_{z_3} of z_2 and z_3; while z_2 receives its environment state O_{z_2}, its hidden state H_{z_2}, and the hidden states H_{z_1} and H_{z_3} of z_1 and z_3 as input. The same principle applies to the z_3. In addition, as in TRbA, the total reward is used. For each service, the hidden states and total rewards are used as a direct influence of other shared services that share the same actuator with it. The states of the non shared actuators depend only on the output of the action quality values of the associated services. The states of the shared actuators depend on the action quality values generated by a MultiLayer Perceptron (MLP)-based policy that uses the action quality outputs of the shared services as input.

V. Comparative Experiment

To compare the performance of SHOMA architectures for dynamic generation of smart home services, we conduct several simulated experiments based on three simulated services: a light service, a temperature service, and an air quality service. Before the experiments, we describe how the environments are simulated to determine the values of the controllable environment states that the corresponding services attempt to regulate. When simulating these services, we disregard the low impact actions. For example, we do not consider the heat generated by turning on the lamp.

A. Simulated environment

The involved variables in the environment are as follows: (1) s_{us}: state of the inhabitant. (2) s_{le}: outdoor light intensity. (3) s_{te}: outdoor temperature. (4) s_{ae}: outdoor air quality. (5) s_{lr}: indoor light intensity. (6) s_{tr}: indoor temperature. (7) s_{ar}: indoor air quality. (8) s_{lp}: state of the lamp. (9) $s_{cur}^{light}, s_{cur}^{temp}, s_{cur}^{air}, s_{cur}$: the curtain state proposed by light service, temperature service and air quality service, and the final curtain state. (10) s_{ac}: state of the air conditioner. (11) $s_{win}^{temp}, s_{win}^{air}, s_{win}$: the window state proposed by the temperature service and the air quality service, and the final window state. (12) s_{ap}: state of the air purifier. (13) $s_{wct}^{temp}, s_{wct}^{air}$: working duration for windows and curtains proposed by the temperature service and the air quality service, respectively. (14) s_{act}: air conditioner working duration. (15) s_{apt}: air purifier working duration. And s_{lr}, s_{tr} and s_{ar} are the controllable environment states that light, temperature and air quality services attempt to adjust.

1) Light service: The learning system associated with the light service takes s_{us} and s_{le} as input and s_{lp} and s_{cur}^{light} as output. Among them, $s_{us,t}$ is randomly generated by following the uniform distribution: $s_{us,t} = U_{int}(0, n_{us})$, where n_{us} is the total number of states of the inhabitant and $U_{int}(0, n_{us})$ is a function that randomly generates an integer between 0 inclusive and n_{us} exclusive by respecting the uniform distribution. s_{le} of one day is simulated with a Gaussian distribution [15], [16]: $s_{le,t} = \mathcal{N}(amplititude = 600, mean = 12, stddev = 3) + 5 \cdot U(0, 1)$, where $\mathcal{N}$ denotes the Gaussian distribution. Also, some noise is added to $s_{le,t}$, which is simulated using a uniform distribution with a maximum value of 5. To simplify the experiment, $s_{le,t}$ is generated every 5 minutes every day. $s_{lp,t}$ can have multiple levels represented by integers, with level 0 indicating that the lamp is off and other levels indicating that the lamp is on. The light intensity that $s_{lp,t}$ can provide is $\beta \cdot s_{lp,t}$, where $\beta = 100$ is the light intensity that one level can provide. $s_{cur,t}^{light}$ has three possible values: 0, $1/2$,1 representing that the curtain is closed, half open, and fully open, respectively. Suppose $s_{lp,t}, s_{cur,t}^{light}$ and $s_{le,t}$ at time step t are known, we can obtain $s_{lr,t}$ as follows:

$$s_{lr,t} = \beta \times s_{lp,t} + s_{le,t} \times s_{cur,t}^{light}. \tag{3}$$

2) Temperature service: To adjust s_{tr}, the temperature service learns to propose the states $s_{ac}, s_{win}^{temp}, s_{cur}^{temp}$ and the corresponding working duration s_{act}, s_{wct}^{temp} by considering s_{us} and s_{te}. Among these variables, $s_{us,t}$ is simulated in the same way as in the light service. $s_{te,t}$ is simulated with trigonometric functions [17] and expressed as: $s_{te,t} = A \cdot cos(B \cdot (x - D)) + C$ where $A = -7, B = \pi/12, C = 19$ and $D = 4$ and x is the corresponding time with unit $hour$ at time step t; the relation between x and t is: $x = t \cdot 5/60$ with the time interval between t and $t + 1$ being 5 minutes. The possible values for the temperature service's propositions are as follows: (1) $s_{ac,t} \in \{0, -1, 1\}$ with 0 for off, 1 for heating and -1 for cooling; (2) $s_{cur,t}^{temp}$ has the same range as $s_{cur,t}^{light}$; (3) $s_{win,t}^{temp} \in \{0, 1\}$ with 0 for off, and 1 for open; (4) $s_{act,t}, s_{wct,t}^{temp} \in \{i/10 \text{ for } i \in \{0, 50\}\}$.

To calculate s_{tr} knowing the propositions, we first assume that the watt-hours ($W \cdot h$) of the air conditioner is $\phi_{ac} = 20 \cdot 735$; the specific heat of the air is constant and is $C_p = 1.005$; the air density is constant on average and has the value $\rho = 1.205(kg/m^3)$; and the room under study has the volume $V = 60(m^3)$. Thus, the energy produced after the operation of the air conditioner for the duration of $\Delta s_{act,t}(h)$ is:

$$Q_{ac,t} = \phi_{ac} \cdot \Delta s_{act,t} \tag{4}$$

Assuming that the resulting indoor temperature at time $t + 1$ is $s_{tr,t+1}$, the total energy that should be provided to maintain $s_{tr,t+1}$ is:

$$Q_{heat,t} = C_p \cdot \rho \cdot V \cdot |s_{tr,t+1} - s_{tr,t}| \tag{5}$$

Besides, there is always the air circulation between the indoor and outdoor through the window and curtain, thus, the resulted heat loss is:

$$Q_{loss,t}^{heat} = L_t \cdot s_{wct,t}^{temp} \cdot \rho \cdot C_p \tag{6}$$

where $L_t(m^3/s)$ is the air flow rate for indoor and outdoor air circulation and can be calculated as:

$$L_t = d_h \cdot d_l \cdot \sqrt{\frac{2 \cdot g \cdot (\rho_t^e - \rho_t^r) \cdot h}{\lambda \cdot d_w \cdot \rho_t^r / d_l + \sum \varsigma \cdot \rho_t^r}} \tag{7}$$

where $d_h = 2(m)$, $d_l = 2(m)$, and $d_w = 0.2(m)$ are the height, length, and width of the window, respectively; $g = 9.81(m/s^2)$ is the acceleration rate due to the gravity; $\lambda = 0.019$ is the Darcy-Weisbach friction coefficient; $\sum \varsigma$ is the summarized minor loss coefficient; and ρ_t^e and ρ_t^r are the indoor and outdoor air densities as a function of the corresponding air temperature:

$$\rho_t^{e,\,r} = 1.293(kg/m^3) \cdot 273(K)/(273(K) + s_t^{te,\,tr}\,(^\circ C)) \tag{8}$$

As a result, we have the relation:

$$Q_{heat,t} = Q_{ac,t} + Q_{loss,t}^{heat} \tag{9}$$

According to equations of (4), (5) and (6), we can obtain the resulted indoor temperature:

$$s_{tr,t+1} = \frac{Q_{ac,t} + Q_{loss,t}^{heat}}{C_p \cdot \rho \cdot V} \pm s_{tr,t} \tag{10}$$

where "+" represents the air conditioner is heating, and "-" denotes it is cooling.

3) Air quality service: The air quality service controls indoor air quality by adjusting the s_{ap}, s_{win}, s_{cur} of the actuators and their working duration s_{apt} and s_{wct}^{air} with the consideration of s_{us} and s_{ae}. The detailed descriptions for simulating the above states are as follows: $s_{us,t}$ is simulated in the same way as in the light and temperature service. To simulate $s_{ae,t}$, we first impute the atmospheric carbon dioxide dataset from quasi-continuous measurements on American Samoa [18]. Imputation allows us to replace the anomalous data with surrogate data. In this study, the surrogate data that we use is the average of the corresponding

data. Then, an interpolation is performed to obtain a data set sampled every 5 minutes instead of hourly about $s_{ae,t}$. The states proposed for the actuators are adjusted by the DQN. The possible values for the actuators involved are: (1) $s_{ap,t} \in \{0, 60, 170, 280, 390, 500\}$ with 0 being turning off, and other numbers representing the values of the cubic meter air flow rate (m^3/h or CMH) of the air purifier; (2) $s^{air}_{cur,t}$ has the same range as $s^{light}_{cur,t}$ and $s^{temp}_{cur,t}$; (3) $s^{air}_{win,t}$ has the same range as $s^{temp}_{win,t}$; (4) $s_{apt,t}, s^{air}_{wct,t} \in \{i/10 \; for \; i \in \{0, 50\}\}$.

The controllable indoor air quality is influenced by the outdoor air quality and the air purifier and can be calculated as follows: [19]:

$$s_{ar,t+1} = s_{ar,t} \cdot (1 - \frac{s_{ap,t} \cdot s_{apt,t}}{V} - \frac{L_t \cdot s_{wct,t}}{V} - \frac{n_{us,t} b_{us,t}}{V} \cdot \frac{\Delta x_{us}}{V}) + s_{ae,t} \cdot \frac{L_t \cdot s_{wct,t}}{V} + s_{exha,t} \cdot \frac{n_{us,t} \cdot b_{us,t} \cdot \Delta x_{us}}{V} \tag{11}$$

where V and L_t represent the same value as in the temperature service environment setting; $s_{exha,t} = 38000(ppm)$ is a constant representing the CO_2 content in the exhaled air; $n_{us,t}$ is the number of inhabitants in the room. In this study, $n_{us,t}$ is constant and has the value 1; $b_{us,t}$ is the CO_2 breathing rate of the inhabitant, whose value depends on the inhabitant's activity and can be found in Table 3 and Table 4 in [20]. In this paper, we assume that the inhabitant is between 21 and 30 years old, so the physical activity level M corresponding to s_{us} is $M = \{0, 1.4, 4, 1\}$. The CO_2 breathing rate $b_{us,t}$ associated with s_{us} is $b_{us,t} \in \{0, 11.004(mg/s), 31.44(mg/s), 7.6635(mg/s)\}$. Δx_{us} is the inhabitant's breathing time with a constant value 5 minutes between two time steps.

VI. Experiment Results

In this section, to evaluate the SHOMA architectures and select the architectures with the best performance, we first compare all the architectures with the temperature and air quality services, with and without energy saving requirements in the simulated inhabitant's habitual behaviors. From this comparison, we select the architectures with better performance. Then, we evaluate the selected architectures with three services with and without energy saving requirements to determine the best performing architectures.

Following metrics are defined to evaluate the performance of the architectures: (1) Accuracy of each service to propose the actions of the actuators correctly. (2) Accuracy of all services to propose the actions of the actuators simultaneously correctly. (3) Average of all accuracy, which denotes the general performance of an architecture. The above accuracy indicates the number of samples in which each service or all services correctly propose actuators' actions so that the updated controllable environment states match the inhabitant's target values, as a percentage of the total samples.

A. Evaluation results with two services

Fig.10 shows the result of the predefined metrics for the experiment without energy saving requirement. It can be seen that EPbA generally performs better on all four metrics: the accuracy of the temperature service, the air quality service, the two services that correctly suggest the state of the actuators simultaneously, and the general performance. RSAbA and CCbA perform best after EPbA. PbA is fourth, followed by TRbA. OLSbA, QmixbA, and CSbA perform worst.

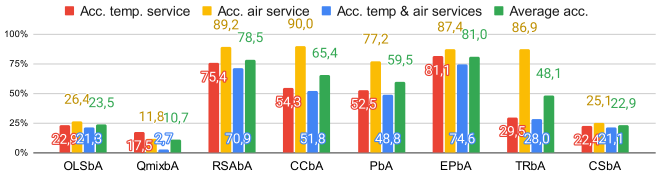

Fig. 10. Architecture evaluations without constraint and with two services

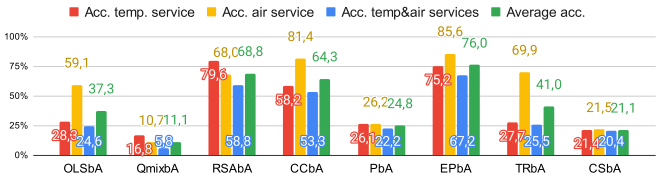

Fig. 11. Architecture evaluations with constraint and two services

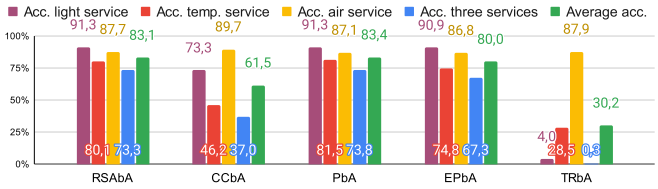

Fig. 12. Architecture evaluations without constraint and with three services

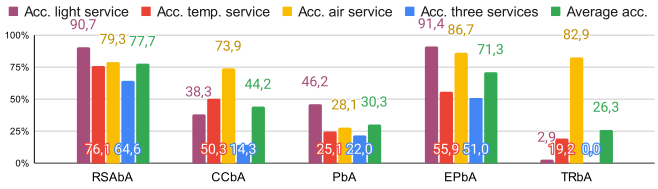

Fig. 13. Architecture evaluations with constraint and three services

We then apply these architectures to the same test dataset but with the saving energy constraint. The result is shown in Fig.11. The result is almost the same as in Fig.10, except that the performance of PbA drops sharply. With the exception of OLSbA and QmixbA, the other architectures also show slight performance degradation, but not as severe as PbA. To remove the doubt that adding the constraint in the reward function is not a good way to realize energy saving, we select architectures: RSAbA, CCbA, PbA, EPbA, and TRbA, with general performance greater than or close to 50%, based on the result in Fig.10 for three service-based experiments.

B. Evaluation results with three services

Fig.12 shows the results without energy saving requirement. Compared to Fig.10, the performance of RSAbA and PbA improves, which means that introducing the light service helps

to improve the learning performance of the temperature and air quality services. For RSAbA, this can be because introducing the light service reduces the number of actuators controlled by the temperature and air quality services. For PbA, introducing the light service may provide more information to decide which service should have higher priority. For EPbA, the performance in the two figures is almost the same, so introducing the light service has no significant impact. The performance of CCbA and TRbA is still not as good as the other three architectures: The performance of CCbA has slightly decreased. The performance of TRbA has dropped significantly, so introducing the light service makes services with TRbA architecture harder to learn the system patterns.

From the result in Fig.13, EPbA and RSAbA always have the best performance that is hardly affected. This shows that introducing energy saving constraint and expressing it as the inhabitant's habitual behaviors work well for them. However, the performance of PbA decreased significantly compared with Fig.12, which is the same phenomenon as that in Fig.11. Therefore, expressing the energy saving constraint in the inhabitant's habitual behavior is not a good method for PbA, and perhaps other methods should be used to describe the energy saving. The performance of CCbA and TRbA has decreased, which may be because introducing more services along with the constraint makes the environments more complex. As a result, learning the characteristics of the environment using the two architectures becomes more complicated.

The above four experiments show that OLSbA and QmixbA perform worse than most composite service-based architectures, which proves the advantage of composite service-based architectures. Moreover, the comparison of the composite service-based architectures shows that EPbA and RSAbA have better performance than the others, and thus are selected for our future research on the smart home system. Even though only experiments with two and three services are discussed in this paper, these experiments considered all possible problems that may occur when more than three services work together. Therefore, our results are most likely applicable for systems with more than three services, even though these systems may need more time to be well trained.

VII. Conclusion

Developing a smart home system to dynamically generate services can be very complex. Existing solutions for building such a smart home system either require complex manual input from the inhabitant, require the inhabitant to specify regulation solutions, or do not consider the potential conflicts when generated services use the same actuators.

In this paper, we propose RL-based SHOMA architectures for a smart home system to dynamically generate services considering the inhabitant's (dis)satisfaction. We conduct several simulated experiments with and without the energy saving requirements specified in the inhabitant's habitual behaviors. The results show the advantage of composite service-based architectures by defining multiple RL-based services and contribute to selecting the architectures with the best performance.

In perspective, it is important to evaluate the selected architectures in the real environment. Moreover, it is promising to integrate the knowledge-based approaches into these architectures to make the generated services understandable.

Acknowledgment

This version of the contribution has been accepted for publication by ICMLA (IEEE International Conference on Machine Learning and Applications), after peer review but is not the version of record and does not reflect post-acceptance improvements, or any corrections.

References

[1] D. Chaki and A. Bouguettaya, "Fine-grained conflict detection of iot services," in *2020 IEEE International Conference on Services Computing (SCC)*. IEEE, 2020, pp. 321–328.

[2] Y. Sun, X. Wang, H. Luo, and X. Li, "Conflict detection scheme based on formal rule model for smart building systems," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 2, pp. 215–227, 2014.

[3] M. Ma, S. M. Preum, W. Tarneberg, M. Ahmed, M. Ruiters, and J. Stankovic, "Detection of runtime conflicts among services in smart cities," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–10.

[4] M. Chan, D. Estève, C. Escriba, and E. Campo, "A review of smart homes—present state and future challenges," *Computer methods and programs in biomedicine*, vol. 91, no. 1, pp. 55–81, 2008.

[5] M. Qiu, E. Najm, R. Sharrock, and B. Traverson, "Pbre: A rule extraction method from trained neural networks designed for smart home services," in *International Conference on Database and Expert Systems Applications*. Springer, 2022, pp. 158–173.

[6] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *arXiv preprint arXiv:1811.12560*, 2018.

[7] L. Mainetti, V. Mighali, L. Patrono, and P. Rametta, "A novel rule-based semantic architecture for iot building automation systems," in *2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2015, pp. 124–131.

[8] P. Wang, H. Luo, and Y. Sun, "A habit-based swrl generation and reasoning approach in smart home," in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015, pp. 770–775.

[9] H. Si, Y. Kawahara, H. Morikawa, and T. Aoyama, "A stochastic approach for creating context-aware services based on context histories in smart home," *COGNITIVE SCIENCE RESEARCH PAPER-UNIVERSITY OF SUSSEX CSRP*, vol. 577, p. 37, 2005.

[10] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/BF00992698

[11] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[12] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4295–4304.

[13] D. Ha, A. Dai, and Q. V. Le, "Hypernetworks," *arXiv preprint arXiv:1609.09106*, 2016.

[14] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[15] S. Ilyas *et al.*, "The impact of revegetation on microclimate in coal mining areas in east kalimantan." *Journal of Environment and Earth Science*, vol. 2, no. 11, pp. 90–97, 2012.

[16] A. S. Juraimi, M. Saiful, M. Begum, A. Anuar, and M. Azmi, "Influence of flooding intensity and duration on rice growth and yield," *Pertanika J. Trop. Agric. Sci*, vol. 32, no. 2, pp. 195–208, 2009.

[17] J. Š. Benth and F. E. Benth, "A critical view on temperature modelling for application in weather derivatives markets," *Energy Economics*, vol. 34, no. 2, pp. 592–602, 2012.

[18] J. M. K.W. Thoning, A.M. Crotwell. (2021) Atmospheric carbon dioxide dry air mole fractions from continuous measurements at mauna loa, hawaii, barrow, alaska, american samoa and south pole. =https://doi.org/10.15138/yaf1-bk21.

[19] G. Ansanay-Alex, "Estimating occupancy using indoor carbon dioxide concentrations only in an office building: a method and qualitative assessment," in *REHVA World Congress on Energy efficient, smart and healthy buildings (CLIMA)*, 2013, pp. 1–8.

[20] A. Persily and L. de Jonge, "Carbon dioxide generation rates for building occupants," *Indoor air*, vol. 27, no. 5, pp. 868–879, 2017.