Human-Centric Automation and Optimization for Smart Homes

Noel Nuo Wi Tay, János Botzheim, *Member, IEEE* and Naoyuki Kubota

Abstract—**A smart home needs to be human-centric, where it tries to fulfill human needs given the devices it has. Various works are developed to provide homes with reasoning and planning capability to fulfill goals, but most do not support complex sequence of plans or require significant manual effort in devising subplans. This is further aggravated by the need to optimize conflicting personal goals. A solution is to solve the planning problem represented as constraint satisfaction problem (CSP). But CSP uses hard constraints and, thus, cannot handle optimization and partial goal fulfillment efficiently. This paper aims to extend this approach to weighted CSP. Knowledge representation to help in generating planning rules is also proposed, as well as methods to improve performances. Case studies show that the system can provide intelligent and complex plans from activities generated from semantic annotations of the devices, as well as optimization to maximize personal constraints' fulfillment.**

Note to Practitioners—**Smart home should maximize the fulfillment of personal goals that are often conflicting. For example, it should try to fulfill as much as possible the requests made by both the mother and daughter who wants to watch TV but both having different channel preferences. That said, every person has a set of goals or constraints that they hope the smart home can fulfill. Therefore, human-centric system that automates the loosely coupled devices of the smart home to optimize the goals or constraints of individuals in the home is developed. Automated planning is done using converted services extracted from devices, where conversion is done using existing tools and concepts from Web technologies. Weighted constraint satisfaction that provides the declarative approach to cover large problem domain to realize the automated planner with optimization capability is proposed. Details to speed up planning through search space reduction are also given. Real-time case studies are run in a prototype smart home to demonstrate its applicability and intelligence, where every planning is performed under a maximum of 10 s. The vision of this paper is to be able to implement such system in a community, where devices everywhere can cooperate to ensure the well-being of the community.**

Index Terms—**Automated planning, human-centric, smart home, weighted constraint satisfaction problem (CSP).**

I. Introduction

As MARK WEISER, the pioneer in ubiquitous computing, put it, technologies should fit the human environment, instead of humans having to adapt to technologies [1]. Leitner [2] has argued that smart homes are developed mostly from the viewpoint of technical capabilities, which is considered "ego-centric." It proposes to shift the smart home paradigm from technological capabilities to that which centered around human's need to enhance their living experience. Thus, the concept of design is "personalized service provision from loosely coupled smart home devices with optimization and semantic reasoning capability," and hence, human-centric system reasoning and optimization. For clarity, the term "reasoning" refers to ontological reasoning in this paper, whereas "optimization" refers to maximizing the number of fulfilled goals while minimizing the cost of actions to achieve it. Human-centric computing is a recent technology paradigm where services are provided to humans anywhere and anytime based on their situation, given the available devices and service providers around [3]. With this system, human centricity applies to automatic and proper binding of devices to fulfill personal constraints of every user within the vicinity of smart control.

In the context of human centricity, we denote the quality of life (QOL) as a set of personalized constraints that need to be fulfilled, such as that they prefer dim lights at night when they just woke up, the duration between meal times must not exceed a certain amount of time, etc. These constraints are described via logic or mathematical formulation. Devices abiding the computing paradigm of human centricity are connected to the central controller, which reasons and guides them according to the functionalities and semantics of the devices.

With the human-centric system being the central controller, it needs to bind (getting the device to communicate with each other) and coordinate (controlling device through automated planning) devices to fulfill the QOL. Kaldeli *et al.* [4] use constraint processing to handle planning where variables of a wider domain are required in the planning process, which is extended to constraints with weights [5]. Therefore, this paper aims to use weighted constraint satisfaction problem (CSP) to perform automated planning for human-centric smart home such that personal constraint optimization through planning can take place, which also deals with over-constraint situations. It also proposes ontologies such that it can automatically generate planning operators derived from human and individual devices that are connected to the smart home. According to our knowledge, no current methods have been shown to optimize plans that cover such wide range of problem domain to maximize fulfillment of human-imposed personal constraints for the smart home, which takes device operation costs into account as well,

Manuscript received June 2, 2017; accepted December 20, 2017. Date of publication February 1, 2018; date of current version October 4, 2018. This paper was recommended for publication by Associate Editor M. K. Oishi and Editor D. Tilbury upon evaluation of the reviewers' comments. *(Corresponding author: Noel Nuo Wi Tay.)*

N. N. W. Tay is with the Graduate School of System Design, Tokyo Metropolitan University, Tokyo 192-0364, Japan (e-mail: taynoel@ieee.org).

J. Botzheim is with the Department of Automation, Széchenyi István University, 9026 Győr, Hungary.

N. Kubota is with the Graduate School of System Design, Tokyo Metropolitan University, Tokyo 192-0364, Japan.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2018.2789658

under one declarative approach. This paper also demonstrates real-time implementation in a prototype smart home, where detailed configurations and settings are given.

That said, in terms of method, the main emphasis is on the realization of weighted CSP for planning in for the human-centric smart home. The weighted CSP of our previous work [5] is converted to solving planning problem of the human-centric smart home using wrapped-up device services. Branch-and-bound depth first search is used, where the problem is relaxed by eliminating preconditions and converting them to subgoals with costs, upon which stochastic search is applied. Search space reduction method is also introduced to make the planner scalable.

To make the system complete, an automatic way of converting device service into planner usable operators and the support of ontological reasoning are established. This paper concentrates on two ontologies: 1) human ontology, which represents human privileges, location, and their state info and 2) device ontology, which represents the device type, location, provided services, and their associated variables. We use methods endorsed by W3C, such as limiting the ontology schema to that supported by OWL, and association rules are made such that they can be implemented via Semantic Web Rule Language (SWRL) or SPARQL Protocol and RDF Query Language (SPARQL) query.

This paper is on automatic device binding and coordination of device operations to optimize QOL fulfillment, but it assumes that the goals, service rules, and costs are manually set, although some natural ways of extending them to adaptive manipulation and cost setting are suggested.

The outline of this paper is as follows. Section II discusses the related works on home automation. The architecture is presented in Section III, where the first part talks about building ontology (related to the secondary problem) in Section III-A, and the second part explains the weighted CSP planner in Section III-B (related to the primary problem). Case studies and discussions are provided in Section IV-C to demonstrate the capabilities of the system in real time. Finally, the work is concluded in Section V.

II. Related Works

Concepts from Web services and AI planning play important roles in various real-life applications, such as in handling hydrogeological disaster [6], fire emergencies with context awareness [7], home surveillance through Web-connected sensors [8], in the field of engineering and industrial automation [9], [10], and smart grid application [11].

Various home automation systems have been developed over time that employs the service-oriented architecture (SOA) [12] complemented by Web technologies. Home automation for inferring environment state is developed [13]. It does not include service composition; nevertheless, it provides insight in the generation of contextual information for service composition to work on. Enhancements are made to the KNX standard in order to support knowledge-based and context-aware functionalities in home automation [14]. The system also supports partial or disjoint matches, and service composition is performed through concept abduction algorithm [15]. But such service composition will not be applicable if service sequence is crucial. Building automation adopting SOA and device profile for Web service (DPWS) is proposed that supports service composition and reasoning [16]. Similar to the enhanced KNX home automation [14], service composition does not support construction of complex sequence of services. Kaldeli *et al.* [4], [17] developed a home automation service composition based on SOA that is capable of complex composition of services using constraint processing that supports parallelism. The system does not require manual construction of subplans. It handles context awareness and is able to deal with uncertain situation by dynamic replanning, but does not consider ontological reasoning.

Smart home requires composition of plans to be functional, where detailed review is given [18]. In [19], OWLS-Xplan uses the semantics of atomic Web services defined in OWL-S for planning purposes, but the semantics are not utilized and semantic awareness is not achieved. Exact matching for service inputs and outputs is required from the planning module, which is the disadvantage. There are frameworks that convert Web service composition tasks into planning problems expressed in Planning Domain Definition Language for further processing [20], [21]. Contrary to OWLS-Xplan [19], when no matches can be found, semantic information is utilized to obtain composite services that best approximate the goal. To handle complex tasks and to reduce planning complexity, hierarchical task network [22] is introduced. It uses method definitions that specify how the complex tasks can be broken down into more manageable tasks [23], [24]. The planner will then solve the planning problem by applying the breaking down of tasks to every task in the list to reduce them to their atomic planning operator constituents. The main disadvantage of this approach is that the planning process requires decomposition rules to be specified by an expert. Answer set programming has also been used for planning [25], [26]. At the moment, for complex sequence of services, these methods require heavy computational load.

The methods discussed thus far either require exact match-ups, assume certain ontologies to handle heterogeneities, or require specifications of user anticipation and procedural templates. Domain modeling through CSP is developed to create a language that allows users to express goals without having to know about the details and interdependencies between services [4], [17], [27], which is of similar concept with the multivalued planning task encoding [28]. Besides, another advantage is that it is able to handle variables with large domain efficiently. Although the CSP planner might be slower than some state-of-the-art methods [24], [29], [30], it supports variable of larger domain more efficiently, which allows natural processing of constraints involving values such as integers and large finite domain data types that can be described under one umbrella of declarative language. Furthermore, CSP planner is extended to support weighted constraints [5].

III. Architecture

Fig. 1 shows the discrete methodologies used to achieve human-centric smart home. Identification module identifies

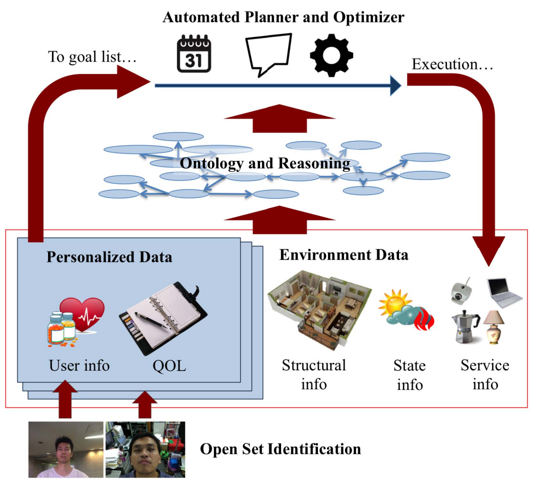


Fig. 1. Gist of methodologies for human-centric smart home.

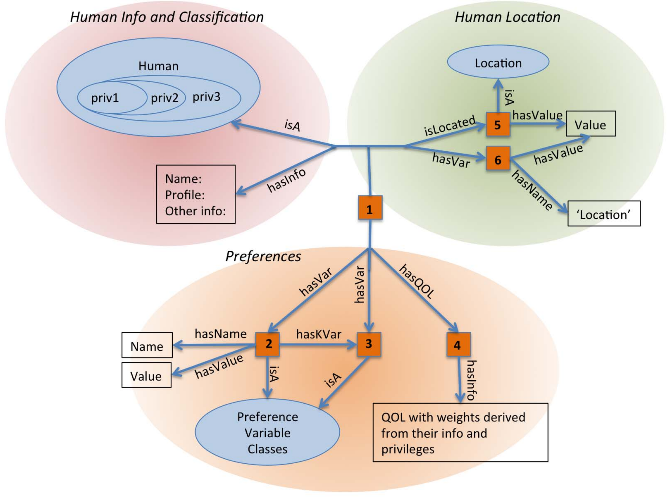


Fig. 2. Ontology for human. Square indicates an instance, ellipse indicates a class, arrow indicates a property, and square indicates values or node that leads to information. These indications apply to subsequent figures on ontology.

and tracks people who are within the vicinity of control such that their personalized data can be extracted, like their personal info and QOL. With the devices connected to the smart home and the house layout, ontology and reasoning module will wrap up their services to become planning operators (or activities) fit for planning. Automated planner and optimizer (which is built up from our previous work to accommodate human-centric smart home [5]) will then coordinate the devices to maximize fulfillment of QOL. For clarity, we will use "operator" to refer to service operators for planning, unlike "service," which is of a much general use. Service composition through service mash-ups that are generated by semantic reasoners is not discussed in this architecture. We assume the right services are already composed that can be used for planning.

Section III-A will explain the ontology and reasoning module of Fig. 1, which is used as knowledge representation for automated planning in Section III-B.

A. Building Ontology

The building ontology, represented via OWL DL, defines the relationships between entities and their concepts that are within the building environment. Although the main emphasis of this paper is not on device binding and reconfiguration, and middlewares that ensure compatibility, this section shows the knowledge representation that assists for automatic plan operator generation. The motivation is to show that existing approaches, such as from [31], can be applied to realize a human-centric system. Two ontologies are defined for the building ontology, which are ontologies for human and for devices. This section gives the gist of the ontology, where specific structures will be illustrated for case studies in Section IV-C.

1) Human Ontology: As human-centric system focuses on fulfilling the needs of human, it needs to have a representation of the human state where whatever decisions and manipulations can be made from and on it. That said, every human that is detected will be assigned a node, where the graph associated with the node is shown in Fig. 2. The node is indicated by 1 and it falls under class *Human*, which consists of classes of different privileges *priv*, where larger privileges subsume lower privileges (for example, $priv2 \subseteq priv3$). Privileges with lower numbers indicate a higher privilege compared with those with higher numbers. For example, the host's *Human* node falls under *priv1*, and the friends fall under *priv2*. The *Human* node is connected to its information via property *hasInfo*, which directs to further or raw information of the person.

Preference variables (shown as nodes 2 and 3) are related to *Human* node through property *hasVar*. Examples of preference variables are alarm clock setting, meal preferences, and checkups per month. Although some are not considered preferences but a requirement set by others (such as checkups per month) or might be inferred through sensor networks such as inferred activities or goal recognition, but for simplicity, we will term them as preferences. Preference variables are associated with a Boolean knowledge variable, which is related to it via property *hasKVar*, where they are set to 1 if the preference variables they are associated to are considered known. Preference variables have names and values, which are associated through property *hasName* and *hasValue*, respectively.

Human location is constantly being updated based on the room he/she is in. Human location can be obtained via property *hasLocation*. There is a special variable with the name of "Location" (shown as node 6) that records the value of whatever being assigned to the human's location.

Human node is connected to QOL (shown as node 4) through property *hasQOL*. These are personal constraints that the automated planner needs to optimize. QOLs are considered manually assigned by the host (*priv1*) for his/her friends (*priv2*) and strangers (*priv3*). All constraints in QOL have individual costs denoted by $w(q, h)$, where q is the constraint index in the QOL and h is the human identifier. The cost is the penalty if the constraint from the QOL is not met. At the current stage of work, costs are manually assigned. The total cost of each constraint is obtained via $w(q, h) + priv(h)$, where *priv(h)* is the privilege cost of person h (the cost

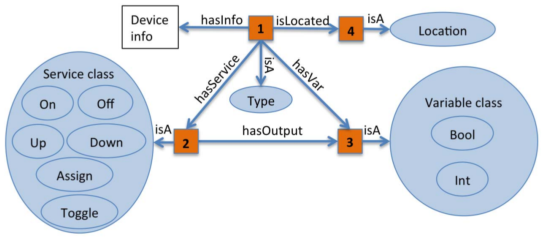


Fig. 3. Ontology for device.

is 50, 30, and 10 for *priv1*, *priv2*, and *priv3*, respectively, which means that penalty is higher if constraints are not met for people with higher privileges). Besides, as stated, currently, we consider QOLs to be assigned manually by the host, who needs to have knowledge on the devices in his/her home or have knowledge to set implicit constraints.

2) Device Ontology: Device ontology is built for every controllable device to enable device and service discovery, interoperation, composition, and configuration, which is a heavily researched field [31], [32]. The associations made are to connect services to compatible and available devices or services, where services are dispensed based on policies. Semantic mark-ups and ontologies provide the means for service composition through the use of semantic reasoners. Here, we introduce the ontology extension to accommodate planning operators.

Besides the device type and services provided by the device, planning operators also require information on the variables. For example, for a dim lamp, apart from assigning a type to the light as dim light, information on what service should be executed to turn ON/OFF the light is needed, as well as an indicator (or variable) to indicate that the light is ON/OFF. Device graph is shown in Fig. 3. As in [31], devices (node 1) are associated with a type class and location. The type classes used in our case studies are shown in Fig. 5. This can support service composition, but in order to extend to planning, services (node 2) and their corresponding variables (node 3) are included as well, both having their own type classes, respectively. Variables can be either *Bool* or *Int*. The service referred here is called the atomic service, which has not been wrapped up with planning rules such as preconditions. Service operators for planning (which is equipped with preconditions and effects) require wrapping up atomic services via rules, which, at the moment, are user-specified. The rules associate the nodes to the preconditions, effects, and other parameters of the planning operators. The rules used for the case studies are shown in Table V in Section IV-B3, where detailed explanations are given.

Currently, atomic service can be assigned to six types of classes: Class On/Off is for services that can assign 1/0 to variables under *Bool*, which is connected via property *hasOutput*. Class *Up/Down* is for services that can increase/decrease values in variables under *Int*. Class *Assign* is for services that assign values to variables under *Int* or *Bool* that is connected via property *hasOutput* from a constant or from other variables that are connected via property *hasInput* (not shown in Fig. 3). Class *Toggle* performs negation on variable under *Bool*.

B. Planner Module

This section explains the automated planner that is realized through weighted CSP.

1) Planning via Weighted Constraint Satisfaction: CSP Domain description for device operation planning is based on the work of [17] and [27], which uses hard constraints. It cannot handle soft constraints to perform optimization efficiently, and thus, an extension to weighted CSP planner that supports dynamic replanning is proposed [5]. Explanation in terms of how the planner is applied to the human-centric system is given.

Various approaches have been used to solve general WCSP, such as clustering and variable elimination as explained in [33], where good heuristics are crucial [34]. Branch-and-bound depth first search is used due to polynomial space complexity, fixed tree layers, and the ability to handle constraints of high arity and wide domain, compared with methods like variable elimination [35] (though variable elimination can be combined with search to obtain much better result [33]).

Given an initial state, an optimized plan means a plan that can fulfill the maximum number of goals with the least operator cost. Costs associated with every goal are imposed as penalties if the goal is not fulfilled in the final state. Constraints imposed by activities are the preconditions and effects. Constraints from preconditions and effects are treated as hard constraints unlike goal-imposed constraints. Unlike costs for goals, operator costs are imposed when an operator is implemented.

Given a set of constraints and costs, the system will try to generate a sequence of activities as plans that is optimum through branch-and-bound search. In order to achieve that, it needs to minimize the following cost function:

$$A^{\mathrm{opt}} = \underset{A_0.A_1\ldots A_{K-1}}{\operatorname{argmin}} \left(\left[\sum_{0\le t<K} C(A_t) + Pc(A_t, S_t) \right) \right] + Gc(\hat{A}\circ S_0)) \quad (1)$$

where $C(A_t)$ is the cost for implementing operator A_t, and $Pc(A_t, S_t)$ gives the cost given whether or not precondition for A_t is fulfilled by state S at sequence t. Due to precondition being a hard constraint, Pc will return infinity if precondition is not met.

$\hat{A}\circ S_0 = A_{K-1}\circ A_{K-2}\ldots A_1\circ A_0\circ S_0$, which gives the final state S_K when operator A_0 is implemented on state S_0, followed by A_1, and so on until finally A_{K-1}. $Gc(S) = \sum_{f\in\mathrm{Goal}} \mathrm{Weight}(f, S)$, which gives the total cost imposed by nonfulfilled goals.

To obtain a larger feasible region by removing restrictions, 1 is relaxed by converting the $Pc(\bullet)$ component to goals, which gives

$$A^{\mathrm{opt}} = \underset{A_0.A_1\ldots A_{K-1}}{\operatorname{argmin}} \left(\sum_{0\le t<K} C(A_t) + \bar{G}c(\hat{A}\circ S_0) \right) \quad (2)$$

$\bar{G}c(\bullet)$ varies from $Gc(\bullet)$ in the sense that it not only considers cost from fulfillment of goals but also extra subgoals introduced by the conversion of $Pc(\bullet)$ [5].

Planner search tree is limited to a finite depth where search will never be stuck in an infinite loop. Therefore, completeness is ensured. But due to the use of BMA stochastic search, search result may not be optimal, although search is enhanced from conditions provided by the subgoals as shown in Section IV-A1. Despite the possibility of nonoptimal plan, the *UB_Refine* function with the search algorithm will filter out such plans [5]. Given a sequence of activities, *UB_Refine* will loop through each operator and calculate whether their omission will produce lower cost. This ensures that no redundant operation will occur, where the worst possible output is a plan with no operation at all. Besides, the fact that planning and execution are occurring continuously in a continual effort to fulfill QOLs as much as possible, complemented by nonredundant plans, nonoptimality's impact is reduced, because subsequent plans can take care of what the previous nonoptimal plan has not achieved. This is demonstrated in Section IV-C.

2) Cost Allocation: Operator *Nop* has cost $wNop = 1$. Every other activities have costs greater than that of *Nop*. This means that operator *Nop* is preferred over others after the goal is achieved. Subgoals (not goals) have cost $wSG = 1$. Every goal (not subgoals) has cost $= K \times wNop + nSG \times wSG$. This puts goals as having higher priority than activities and subgoal fulfillments.

3) Reduction of Operator Search Space: There are a lot of times where goals apply only to certain cases, such as certain time, or when certain events are triggered. An example is the goal to turn ON appropriate lights when the host is in the living room. This goal does not need to be considered when the host is elsewhere. Therefore, in the design of goals, every goal comes with implications $ImpGoal(Kv) \rightarrow Goal$, where if *ImpGoal(Kv)* is met, then *Goal* will be assigned as a goal for planning. *ImpGoal* depends on current knowledge variable *Kv* it is using, but not on the effect variables. This is because knowledge variables cannot be manipulated by activities directly. To determine the truth value, satisfiability is performed on *ImpGoal*, where knowledge variables of *ImpGoal* are fixed to the current value, and effect variables are set as free variables. Intuitively, it means, given the current knowledge variables, is there any assignment of effect variables that will make *ImpGoal* true.

Only activities that are relevant to the given goals are selected. Irrelevant activities to the goal are pruned out to reduce the search space. For every operator a_i, a list of other activities that have at least one effect that has the potential to satisfy the one of the preconditions of a_i are found. Backward action chain is then be computed as used in [36]. This chain of actions can be used to select the activities that are relevant to the imposed goal. An interesting alternative is to use dependency graph as proposed in [37], or to divide into effects' ontology to shortlist relevant activities [38].

IV. Case Studies and Discussion

Case studies shown in this section demonstrate the human-centric home automation system that supports plan generation and optimization to maximize personalized constraints.

Reasoner FaCT++ [39] is used for building ontology reasoning, and Z3 [40] is applied for satisfiability test. A separate module is also built to support necessary SPARQL querying capability (to emulate SWRL) on building ontology supported by FaCT++. For the weighted constraint satisfaction planner, we set the maximum time threshold to 10 s for planning. This means if the planner has not finished searching the optimum solution within 10 s, the current best solution is chosen instead. The whole system is run on a 2.5-GHz Intel Core i5 computer with 4-GB RAM.

Parameters for BMA are the number of generations N_{gen}, the number of bacteria N_{ind}, the number of clones N_{clones}, the mutation segment length l_{bm}, the number of infections N_{inf}, and the infection segment length l_{gt}. For speed and simplicity, l_{bm}, N_{inf}, and l_{gt} are all set to 1. The other parameters are set from preliminary tests on result consistency and time consumption, which are, $N_{\text{gen}} = 10$, $N_{\text{ind}} = 3$, and $N_{\text{clones}} = 3$.

A. Planning Evaluation

1) Problem Relaxation Comparison Test: Comparison is performed to evaluate the improvement contributed by the conversion of precondition to subgoals in problem relaxation for branch and bound, instead of relaxation through eliminating preconditions only (termed normal relaxation), in the smart home environment. The test is performed to compare the time they take to obtain the optimal solution. The experiment is conducted with maximum $K = 9$, 11, and 13 horizon steps of operations. Goals are designed, and are randomly assigned during experiment, such that they need at least K steps to reach the optimal solution. The number of activities is not reduced. Table I is the comparison result. Search will still proceed even when the optimal solution is reached (as it does not know that it is the best solution until full search is completed). Result for time consumption to complete full search (for $K = 9$, as higher K takes more than 3 min) is shown in Table II.

It can be observed that, with subgoal conversion, optimal solution can be obtained very quickly relative to without conversion. On the other hand, search time is only slightly improved. We need the algorithm to obtain the optimal solution fast, and place less emphasis on the total search time as there is a time threshold of 10 s. Therefore, relaxations through subgoal conversion are used due to their superior performance.

2) Evaluation on Time Consumption on Constraint and Operator Numbers: Test is performed to evaluate time consumption to reach 50% improvement from nonaction with a different number of constraints [each constraint consists of grounded terms (for finite domain data type) and /or integers, and Boolean and/or Algebraic relations], where $K = 9$, and the number of activities is 50. Nonaction means the total cost of not doing anything. The reason that the optimum solution is not required is because the planner can replan to reach optimal or near-optimal solution, which is demonstrated in Case 1 in Section IV-C. No operator search space reduction and goal reduction are performed. Result is shown in Table III. It can be observed that time consumption rises quite linearly

TABLE I
TIME CONSUMPTION (SECONDS) TO REACH OPTIMAL SOLUTION FOR DIFFERENT RELAXATION APPROACHES

Description	K	Median	1st Quartile	3rd Quartile	Min	Max
Normal Relaxation	9	30.0	17.1	57.1	8.7	93.2
	11	55.17	19.6	74.4	11.2	110.6
	13	85.6	31.2	151.0	37.0	291.4
With Sub-goal Conversion	9	8.4	7.7	30.7	7.3	55.9
	11	16.6	11.3	31.2	10.4	97.4
	13	34.1	30.7	66.4	23.6	133.0

TABLE II
TIME CONSUMPTION (SECONDS) FOR FULL SEARCH ($K = 9$) FOR DIFFERENT RELAXATION APPROACHES

Description	Median	1st Quartile	3rd Quartile	Mean	Std	Min	Max
Normal relaxation	88.0	59.0	103.7	83.1	28.9	37.9	123.2
With Sub-goal Conversion	74.7	54.2	103.4	75.1	38.2	13.0	122.6

TABLE III
TIME CONSUMPTION WITH DIFFERENT CONSTRAINT NUMBERS

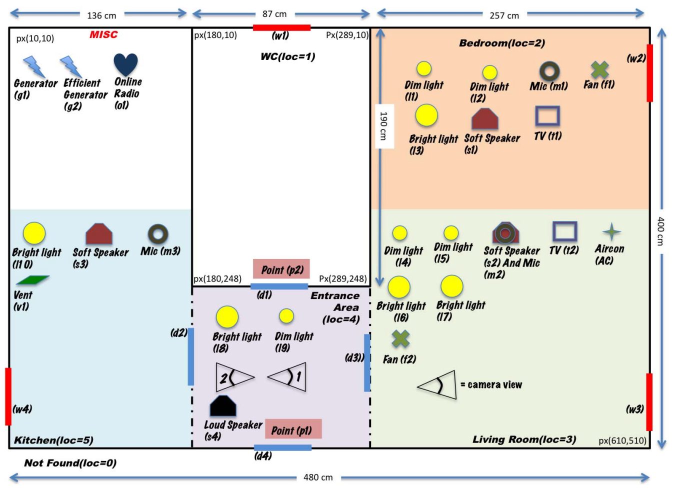
No. of Constraints	20	200	2,000	20,000
Median	8.6	16.3	142.0	1729.3
1st Quartile	8.0	15.4	138.1	1645.3
2nd Quartile	11.4	18.8	147.0	2105.7

TABLE IV
TIME CONSUMPTION WITH DIFFERENT OPERATOR NUMBERS

No. of Activities	50	100	150	200
Median	10.6	31.1	52.7	75.2
1st Quartile	10.2	30.8	50.1	72.8
2nd Quartile	10.8	31.6	58.5	126.8

with the number of constraints, which means it scales well with rising constraints.

Evaluation is also performed on different number of activities (without reduction), with 20–50 constraints. Other test conditions are similar to previous settings. Results are shown in Table IV. It can be observed that time consumption rises linearly with the number of activities as well, though performance consistency drops. Therefore, operator search reduction proposed in Section III-B3 is crucial in ensuring scalability in terms of operator numbers.



(a)

(b) (c)

Fig. 4. Home layout and devices. (a) Home layout and devices' diagram. (b) Camera view 1. (c) Camera view 2.

B. Home and Device Setup

1) Home Setup: An experimental home is set up as shown in Fig. 4, where various snapshots from different camera views of the actual setup are taken. Fig. 4(a) shows the information of the home layout and the devices for each room, and also the direction of camera view for the snapshots.

There are also a clock and temperature sensor that updates variable *time* and *GeneralTemperature*, respectively. Besides that, a fire event sensor is installed in the kitchen, which sets the Boolean variable *KitchenFireEvent* to 1 is there is a fire and 0 otherwise. There is also a Boolean variable *PlanTimer*, which is set to 1 after certain duration. After one planning cycle is finished, it will be set back to 0. It usage as implication for constraints is explained in Section IV-B4.

There are three pressure sensors located in the bed, the toilet, and the floor next to the stove in the kitchen, where their activation indicate, sleeping, in the toilet, and cooking, respectively. Currently, their activation manipulates the variable *HumanState* (more details on the variable in Section IV-B4).

Two Kinect sensors are used to identify and track the humans. Information on human localization will then be used to determine which room the person is in, thus, assigning the location node to the *Human*.

2) Device Setup: Fig. 5 shows the device class type and the assignment of devices to the classes, where the device node of each device will be assigned as shown in Fig. 3. Microphone 2 and speaker 2 are shown as a single object because this is realized by a communication robot we developed called iPhonoid. More information on the robot can be found in [41]. Speakers 1–3 are soft speakers, meaning they can be heard only when the person is in the same room as them. Speaker 4 is a loud speaker that can be heard throughout the house.

Two generators, $g1$ and $g2$, are shown as thunder-shaped object. They do not exist in a particular room, and thus,

TABLE V

RULES FOR ATOMIC SERVICE WRAP-UP

No	Association Rule	Precondition and Effects
1	$Device(?D) \wedge hasService(?D,?S) \wedge On(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$
2	$Device(?D) \wedge hasService(?D,?S) \wedge Off(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} = false$
3	$Device(?D) \wedge hasService(?D,?S) \wedge Up(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} =?V_t + 1$
4	$Device(?D) \wedge hasService(?D,?S) \wedge Down(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D) \rightarrow: assign(?D,?S,?V)$	Precondition: none Effect: $?V_{t+1} =?V_t - 1$
5	$Human(?H) \wedge Speaker(?P) \wedge Mic(?M) \wedge isLocated(?H,?L) \wedge hasVar(?H,?LC) \wedge hasName(?LC, Location) \wedge isLocated(?P,?L) \wedge isLocated(?M,?L) \wedge Priv2(?H) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?H,?P,?M,?L) \rightarrow: assign(?H,?P,?M,?L,?LC,?PC,?KPC)$	Precondition: $?LC_t =?L_t$ Effect: $?PC_{t+1} = Variable_{Res}, ?KPC_{t+1} = true$
6	$Human(?H) \wedge Loud\_Speaker(?P) \wedge Mic(?M) \wedge isLocated(?P,?L) \wedge isLocated(?M,?L) \wedge Priv2(?H) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?H,?P,?M) \rightarrow: assign(?H,?P,?M,?PC,?KPC)$	Precondition: none Effect: $?PC_{t+1} = Variable_{Res}, ?KPC_{t+1} = true$
7	$Human(?H) \wedge Priv2(?H) \wedge TV(?D) \wedge hasService(?D,?S1) \wedge hasService(?D,?S2) \wedge Assign(?S1) \wedge On(?S2) \wedge hasOutput(?S1,?O) \wedge hasOutput(?S2,?T) \wedge hasInput(?S1,?I) \wedge hasVar(?H,?PC) \wedge hasName(?PC, Channel\_Preference) \wedge hasKVar(?PC,?KPC) \wedge nonManaged(?D,?H) \rightarrow: assign(?D,?H,?O,?I,?T,?PC,?KPC)$	Precondition: $(?T_t = true) \wedge (?KPC_t = true)$ Effect: $?I_{t+1} = PC_t$
8	$Loud\_Speaker(?D1) \wedge Radio(?D2) \wedge hasService(?D2,?S) \wedge On(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D1,?D2) \rightarrow: assign(?D1,?D2,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$
9	$Loud\_Speaker(?D1) \wedge Radio(?D2) \wedge hasService(?D2,?S) \wedge Off(?S) \wedge hasOutput(?S,?V) \wedge nonManaged(?D1,?D2) \rightarrow: assign(?D1,?D2,?S,?V)$	Precondition: none Effect: $?V_{t+1} = true$

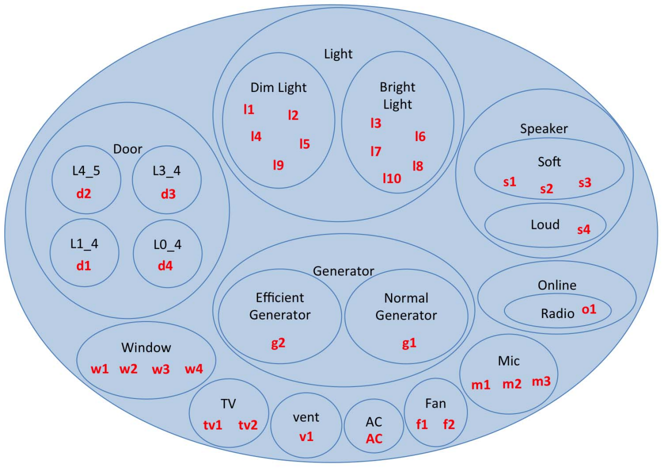


Fig. 5. Device class type—IDs from Fig. 4(a).

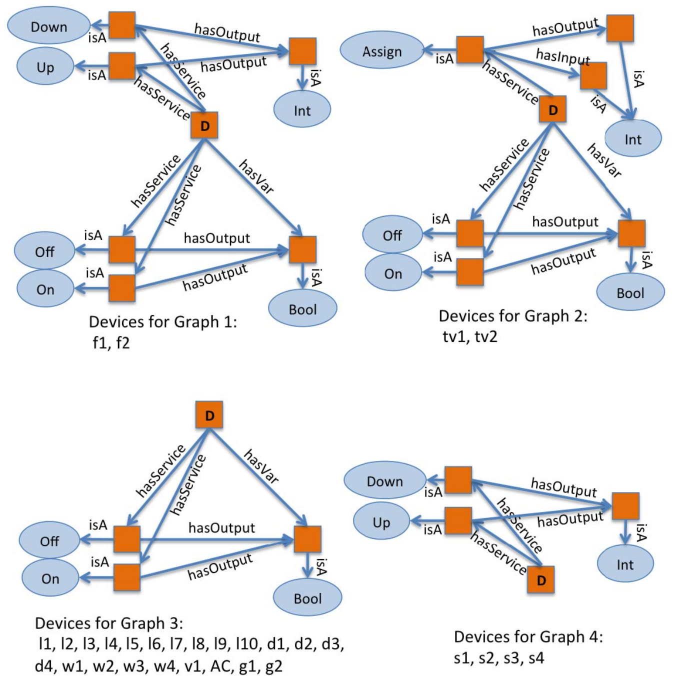

Down
Up
isA
hasOutput
hasService
D
Int
hasVar
Off
On
Bool
Devices for Graph 1: f1, f2
Assign
hasInput
Devices for Graph 2: tv1, tv2
Devices for Graph 3: l1, l2, l3, l4, l5, l6, l7, l8, l9, l10, d1, d2, d3, d4, w1, w2, w3, w4, v1, AC, g1, g2
Devices for Graph 4: s1, s2, s3, s4

Fig. 6. Semantic graphs for the devices. *D* indicates the device node. Device IDs are listed below their corresponding graph.

drawn in the *MISC* area. $g2$ is a lower powered generator compared with $g1$, but it is considered much more efficient. All devices under light, TV, and fan can run given generators 1 and 2, except the air conditioner, which needs $g1$. This requirement is set as extended constraints and will be explained in Section IV-B4. Another component that does not exist in any particular room is the online radio $o1$ (heart-shaped).

The devices described above are all realized through an emulator, except the speaker and the microphones. We believe there will be no loss of generality in using emulated devices as we assume these devices do not affect events. For example, we do not take into account the fact that the turning on of light will cause certain light sensors to pick that up and induce further goals. Such cases are subject of future work. That said, speaker and microphone should be implemented in hardware as they concern interactions with humans.

Semantic graphs (which follows the description in Section III-A) for the devices are shown in Fig. 6. Fans $f1$ and $f2$ apply Graph 1 because they can be turned ON/OFF and be tuned up/down. TV $tv1$ and $tv2$ uses Graph 2 as they can be turned ON/OFF and change the channels by transferring information from the input variable. Speaker $s1$–$s4$ is always ON, but their volume can be controlled, and thus, applies Graph 4. The rest of the devices (except the microphones) can only be turned ON/OFF, which applies Graph 3.

3) Generating Planning Activities From Atomic Services: As explained in Section III-A, devices provide only atomic services. To enable the planner to use them, they need to be

TABLE VI
GENERATED ACTIVITIES

No	Precondition	Effects	Cost
1	none	$g1_{t+1} = true$	3
2	none	$g1_{t+1} = false$	2
3	none	$g2_{t+1} = true$	2
4	none	$g2_{t+1} = false$	2
5-14	none	$l‘n’_{t+1} = true$	2
15-24	none	$l‘n’_{t+1} = false$	2
25-26	none	$f‘n’_{t+1} = true$	2
27-28	none	$f‘n’_{t+1} = false$	2
29	none	$AC_{t+1} = true$	4
30	none	$AC_{t+1} = false$	2
31-32	none	$t‘n’_{t+1} = true$	2
33-34	none	$t‘n’_{t+1} = false$	2
35-38	none	$d‘n’_{t+1} = true$	2
39-42	none	$d‘n’_{t+1} = false$	2
43-46	none	$w‘n’_{t+1} = true$	2
47-50	none	$w‘n’_{t+1} = false$	2
51	$Host\_Loc_t = 2$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}, KHost\_PrefChan_{t+1} = true$	2
52	$Host\_Loc_t = 3$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}, KHost\_PrefChan_{t+1} = true$	2
53	$Host\_Loc_t = 5$	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}, KHost\_PrefChan_{t+1} = true$	2
54	none	$Host\_PrefChan_{t+1} = Host\_PrefChan_{res}, KHost\_PrefChan_{t+1} = true$	4
55	$Frnd\_Loc_t = 2$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}, KFrnd\_PrefChan_{t+1} = true$	2
56	$Frnd\_Loc_t = 3$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}, KFrnd\_PrefChan_{t+1} = true$	2
57	$Frnd\_Loc_t = 5$	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}, KFrnd\_PrefChan_{t+1} = true$	2
58	none	$Frnd\_PrefChan_{t+1} = Frnd\_PrefChan_{res}, KFrnd\_PrefChan_{t+1} = true$	4
59	$(t1_t = true) \wedge (KHost\_PrefChan_t = true)$	$t1chan_{t+1} = Host\_PrefChan_t$	2
60	$(t1_t = true) \wedge (KFrnd\_PrefChan_t = true)$	$t1chan_{t+1} = Frnd\_PrefChan_t$	2
61	$(t2_t = true) \wedge (KHost\_PrefChan_t = true)$	$t2chan_{t+1} = Host\_PrefChan_t$	2
62	$(t2_t = true) \wedge (KFrnd\_PrefChan_t = true)$	$t2chan_{t+1} = Frnd\_PrefChan_t$	2
63	none	$v1_{t+1} = true$	2
64	none	$v1_{t+1} = false$	2
65	none	$o1_{t+1} = true$	2
66	none	$o1_{t+1} = false$	2
67-70	none	$s‘n’vol_{t+1} = s‘n’vol_t + 1$	2
71-74	none	$s‘n’vol_{t+1} = s‘n’vol_t - 1$	2
75-76	none	$f‘n’vol_{t+1} = f‘n’vol_t + 1$	2
77-78	none	$f‘n’vol_{t+1} = f‘n’vol_t - 1$	2

wrapped up by associating them to planning services through rules shown in Table V. *nonManaged* is to check whether the objects of its argument have been dealt with or not. Rules 5 and 6 are special in that they tie the precondition and effect to a script that asks the user of his/her preferred channel through the associated speaker, record and interpret user response via the associated microphone (in this case, recognize the number uttered by the user), and pass the number to the associated variable.

The generated activities are shown in Table VI. For easier reading, in Table VI, the variables are given relevant names. In actuality, all variables are just nodes that have their own IDs. To shorten Table VI, we replace a list of activities by n. For example, l‘n’ means $l1, l2, l3 \ldots l10$, which are the IDs for the lights. Variables with subscript *res* are the response variable. Variable *PrefChan* is preceded by *Host* and *Frnd*, which indicates that the variable comes from the Host and Friend *Human* node, respectively. The same applies to Human location variable shown as *Loc*. *t1chan* and *t2chan* are variables under *assign* type for *tv1* and *tv2*, respectively.

From Table VI, activities 1–50, 63, and 64 are generated from Association 1 and 2 of Table V, operator 67–78 by Association 3 and 4, operator 51–53 and 55–57 by Association 5, operator 54 and 58 by Association 6, operator 59–62 by Association 7, and 65 and 66 by Association 8 and 9.

Given the generated activities, the costs for the activities are manually assigned based on preferences. For clarity, we explain the motivation behind the cost that is set: default cost is the minimum at two. Activities concerning generator $g1$ are set to a higher value of three as the use of $g2$ is preferable compared with $g1$. Activities concerning AC are set high at four as it is not desirable to turn it ON. Activities concerning speaker $s4$ are set at four because its loudness is not desirable.

4) Variables, QOL, and Goals: Preference variables (shown in Fig. 2) store states and information pertaining to the human that is crucial for planning. The relevant preference variables used in the case studies are *WatchTV* (inferred state that the human wants to watch TV), *PrefChan* (preferred TV channel), *PrefTemperature* (preferred thermostat temperature), and *HumanState* (state of human such as sick, well, sleeping, etc., each with their own IDs). The following are the relevant values for *HumanState*: 0 is healthy and sleeping, 1 is healthy and awake, 2 is sick and awake, 3 is sick and sleeping, 4 is cooking, and 5 is in the toilet.

Environment variables store states to describe its state. The relevant environment variables are *time*, *GeneralTemperature*, and *KitchenFireEvent* as laid out in Section IV-B1. Description for the state of *time* is as follows: 0 is early morning, 1 is morning, 2 is noon, 3 is afternoon, 4 is evening, 5 is night, and 6 is midnight.

Constraints in QOL are specified by individuals. Given that there are two persons (a host and his friend) that the smart home knows, thus, there are two sets of QOL from the human in the case studies. Relevant constraints of the QOLs and their costs are shown in Table VII. All formulas are grounded before planning proceeds. The costs shown include costs from privileges, where Host is *priv1* and Friend is *priv2*, thus, 50 and 30, respectively. Constraints 1–14 are the constraints from Host's QOL, and constraints 15–26 are the constraints from Friend's QOL. The remaining constraints are set by the host for the home, and are not tied particularly to any person.

For simplicity, when referring to a variable x that belongs to a device of certain device type and its value, instead of writing $Light(D) \wedge hasVar(D, x) \wedge hasValue(x, 1)$, in Table VII, it will be written as $Light(x) \wedge (x = 1)$ [or $Light(x) \wedge x$ if x is Boolean]. Constraints 27–36 apply to every lights of the house, and constraint 37 applies to every vents.

Predicate $connected(a, b)$ returns whether locations a and b are connected or not. Two rooms are connected if one can walk to the room without obstructions in between such as closed doors. The truth value of the predicate can be easily obtained given the state of doors using simple SAT solvers or from multistage description logic [13].

Constraints 11 and 12 can be generated from constraints 9 and 10, and constraints 23–26 can be generated from constraints 21 and 22. This is because the fulfillment of the original constraints implies the fulfillment of the generated constraints, which enables better optimization by generating more goals. Since such goals can be checked in linear time, it imposes negligible effect.

Constraints 27–38 depend on *PlanTimer*, which is periodically set to one. Therefore, these constraints will be periodically considered. The purpose of these constraints is to maintain default state such as turning the lights OFF if there is nobody there.

Constraints $E1$ and $E2$ are extended constraints. Unlike other constraints that need to be fulfilled at the end of the plan, extended constraints are required to be met throughout the whole planning sequence. $E1$ states that if lights, TV, or fan is turned ON, at least one device under generator class needs to be ON, regardless of whether it is an efficient or normal generator. $E2$ states that if any air conditioner is ON, the normal generator needs to be turned ON.

C. Case Studies

This section will present the case studies to demonstrate the system's capability. Case 1 is on continuous planning and optimization, where planning is done in multistages to reach the optimum solution due to time threshold and plans that are too complex to generate and execute at one go. Case 2 is on reasoning and dealing with conflicting constraints. Case 3 demonstrates the simplicity to extend the planner to support plug-in functions. Case 4 demonstrates the use of extended goals that maintain certain condition throughout the plan, which is crucial if sequence ordering is important.

Case 1 (Continuous Planning and Optimization): Case 1 demonstrates the ability of the system to continually perform planning and optimization with uncertain information. In this case, the time is set at noon, where the temperature is 24 °C. Fans are OFF and their initial speed is one (slowest). The host and the friend are in the living room, where both want to watch TV. Their *PrefChan* variables are not known.

The case starts off by the host and the friend requesting the robot (s2 and m2) that they want to watch TV. The robot is manually programmed to communicate with human and set the proper trigger. Communication with the robot starts off by stating the identity, followed by a keyword that the robot understands. In this case, the host and the friend both request for watching TV, which the robot will set variable *WatchTV* to 1 for both of them as a trigger. The fact that they are in the living room and have their *WatchTV* set means constraints 2 and 14 in Table VII are considered as goals for the planner since they have their implications met (recall that this helps reduce operator search space to increase speed as explained in Section III-B3).

Given the current state, the planner will generate the following sequence of activities (the number in the parentheses is the operator ID from Table VI):

First plan:
Turn ON $g2$(3) ⇒ Turn ON $f2$(26) ⇒ Turn ON *tv2*(32) ⇒ Ask Host about channel(52) ⇒ Increase $f2$ speed by 1(76) ⇒ Ask Friend about channel(56) ⇒ Change *tv2* channel(61) ⇒ End

Second plan:
Turn ON $tv1$(31) ⇒ Change $tv1$ channel(60) ⇒ Increase $f2$ speed by 1(76) ⇒ Increase $f2$ speed by 1(76) ⇒ End.

The plan starts off by turning ON generator $g2$. Since $g2$ has a lower cost to turn ON compared with $g1$, it is selected by the planner. Fan $f2$ is then turned ON, followed by *tv2* in the living room. The robot then proceeds to ask the host about his preferred channel (the host prefers channel 1). The plan also attempts to increase the speed of the fan from the initial speed of one. This is due to constraints 22, 24, and 26. After that, it proceeds to asking the friend of his preferred channel (the friend prefers channel 3). Finally, the *tv2* channel is switched (to host's preference). The device activation flow can be observed in Fig. 7 from sequence 1 to 4.

The host and the friend have different preferences for channel. *tv2* has been used to fulfill the host's constraints due to it having a higher cost.

But all is not lost for the friend because of the constraint 17 in Table VII, which states that any television will do fine, although at a slightly lower cost. At the same time, the fan speed is at two, which does not fulfill constraints 22 and 24.

The second planning therefore proceeds to continue the optimization. Since it already knows about the friend's preference, it turned ON $tv1$ in the bedroom and switched to the preferred channel. The plan is completed by increasing the fan speed to four.

Case 2 (Making Intelligent Choices Under Conflicting Constraints): Case 2 demonstrates the capability to make intelligent choices of devices under conflicting constraints given the dynamic situation in the home. The case is set at night with temperature at 22 °C. Initially, the friend is in the bedroom and the host is in the living room.

The initial state dictates that constraints 5 and 18 of Table VII are considered, where the host wants at least one dim light to be ON and all bright lights to be OFF if he is in the living room, and the friend wants at least one bright light to be ON if he is in the bedroom, respectively. Planning will generate the following plans:

TABLE VII
CONSTRAINTS FROM QOLs

No	Constraints	Cost
1	$(Host\_WatchTV \wedge (Host\_Loc = 2)) \rightarrow (KHost\_PrefChan \wedge t1chan = Host\_PrefChan)$	100
2	$(Host\_WatchTV \wedge (Host\_Loc = 3)) \rightarrow (KHost\_PrefChan \wedge t2chan = Host\_PrefChan)$	100
3	$(Host\_WatchTV) \rightarrow (KHost\_PrefChan \wedge ((t1chan = Host\_PrefChan) \vee (t2chan = Host\_PrefChan)))$	90
4	$((Host\_Loc = 2) \wedge (Host\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Dim\_Light(x) \wedge isLocated(x, 2) \wedge (x = 1)) \wedge \forall y(Bright\_Light(y) \wedge isLocated(y, 2) \wedge (y = 0)))$	100
5	$((Host\_Loc = 3) \wedge (Host\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Dim\_Light(x) \wedge isLocated(x, 3) \wedge (x = 1)) \wedge \forall y(Bright\_Light(y) \wedge isLocated(y, 3) \wedge (y = 0)))$	100
6	$((time = 1) \wedge (Host\_HumanState = 1) \wedge \neg(Host\_Loc = 0)) \rightarrow music$	60
7	$((time = 1) \wedge (Host\_HumanState = 1) \wedge \neg(Host\_Loc = 0)) \rightarrow (s4vol > 5)$	60
8	$(((Host\_HumanState = 0) \vee (Host\_HumanState = 2) \vee (Host\_HumanState = 3)) \wedge \neg(Host\_Loc = 0)) \rightarrow \neg music$	100
9	$((Host\_Loc = 2) \wedge (GeneralTemperature > 25)) \rightarrow (f1 \wedge (f1vol > 2))$	100
10	$((Host\_Loc = 3) \wedge (GeneralTemperature > 25)) \rightarrow (f2 \wedge (f2vol > 2))$	100
11	$((Host\_Loc = 2) \wedge (GeneralTemperature > 25)) \rightarrow (f1 \wedge (f1vol > 1))$	100
12	$((Host\_Loc = 3) \wedge (GeneralTemperature > 25)) \rightarrow (f2 \wedge (f2vol > 1))$	100
13	$(\neg(Host\_Loc = 0) \wedge (GeneralTemperature > 32)) \rightarrow AC$	100
14	$(\neg(Host\_Loc = 0) \wedge (GeneralTemperature > 25) \wedge ((Host\_Loc = 2) \vee (Host\_Loc = 3))) \rightarrow (\neg(f1 = AC) \wedge \neg(f2 = AC))$	100
15	$(Frnd\_WatchTV \wedge (Frnd\_Loc = 2)) \rightarrow (KFrnd\_PrefChan \wedge t1chan = Frnd\_PrefChan)$	80
16	$(Frnd\_WatchTV \wedge (Frnd\_Loc = 3)) \rightarrow (KFrnd\_PrefChan \wedge t2chan = Frnd\_PrefChan)$	80
17	$(Frnd\_WatchTV) \rightarrow (KFrnd\_PrefChan \wedge ((t1chan = Frnd\_PrefChan) \vee (t2chan = Frnd\_PrefChan)))$	70
18	$((Frnd\_Loc = 2) \wedge (Frnd\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow (\exists x(Bright\_Light(x) \wedge isLocated(x, 2) \wedge (x = 1)))$	80
19	$((Frnd\_Loc = 3) \wedge (Frnd\_HumanState = 1) \wedge ((time = 5) \vee (time = 6))) \rightarrow ((l4 + l5 + l6 + l7) > 1)$	80
20	$(((Frnd\_HumanState = 0) \vee (Frnd\_HumanState = 2) \vee (Frnd\_HumanState = 3)) \wedge \neg(Frnd\_Loc = 0)) \rightarrow \neg music$	80
21	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 3))$	80
22	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 3))$	80
23	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 2))$	80
24	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 2))$	80
25	$((Frnd\_Loc = 2) \wedge (GeneralTemperature > 23)) \rightarrow (f1 \wedge (f1vol > 1))$	80
26	$((Frnd\_Loc = 3) \wedge (GeneralTemperature > 23)) \rightarrow (f2 \wedge (f2vol > 1))$	80
27-36	$PlanTimer \rightarrow (\forall x(Light(x) \wedge x = 0))$	10
37	$PlanTimer \rightarrow (\forall x(Vent(x) \wedge x = 0))$	10
38	$PlanTimer \rightarrow (\exists x(Generator(x) \wedge x = 1) \rightarrow (g1 + g2 = 1))$	10
39	$KitchenFireEvent \rightarrow (\forall x((x = Host\_Loc) \rightarrow \neg connected(5, x)))$	50
40	$KitchenFireEvent \rightarrow (\forall x((x = Frnd\_Loc) \rightarrow \neg connected(5, x)))$	50
41	$KitchenFireEvent \rightarrow v1$	50
E1	$\exists x((Light(x) \vee TV(x) \vee Fan(x)) \wedge x = 1) \rightarrow (\exists y(Generator(y) \wedge y = 1))$	50
E2	$\exists x(AC(x) \wedge x = 1) \rightarrow (\exists y(Normal\_Generator(y) \wedge y = 1))$	50

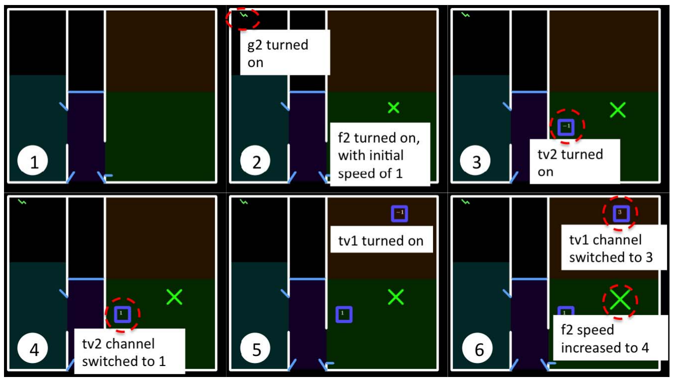

1
2
g2 turned on
f2 turned on, with initial speed of 1
3
tv2 turned on
4
tv2 channel switched to 1
5
tv1 turned on
6
tv1 channel switched to 3
f2 speed increased to 4

Fig. 7. Device visualization for Case 1. Blue square indicates the TV that is turned ON, where the number within represents the channel. Green thunder indicates the efficient generator. Green cross indicates the fan, where its size changes based on its speed.

Turn ON $g2(3) \Rightarrow$ Turn ON Light $l3(7) \Rightarrow$ Turn ON Light $l5(9) \Rightarrow$ End.

Light $l3$ (bright light in the bedroom) and light $l5$ (dim light in the bedroom) are switched ON, where the sequence is shown in Fig. 8 from 1 to 4.

After a short while, the host starts to walk to the bedroom. Constraint 5 is no longer considered, and constraint 4 will be considered, which is conflicting with constraint 18 (from the friend). But since the host possesses higher cost, home automation will try to fulfill his wishes by a newly generated plan as follows:

Turn ON Light $l2(6) \Rightarrow$ Turn OFF Light $l3(17) \Rightarrow$ End.

Bright light $l3$ that is previously turned ON for the friend is turned OFF, and dim light $l2$ is turned ON, shown in sequence 5 of Fig. 8.

Constraints 27–36 will be considered every once in a while due to periodic setting of *PlanTimer*. When these constraints are considered, it requires all light to be turned OFF. But it will not turn OFF $l2$, as the cost of fulfilling constraint 4 is higher. But $l5$ in the living room is still turned ON as it would not turn OFF by itself, since no constraints are violated even when the host moves to the bedroom. The act of turning it OFF will introduce a cost of two. But with constraints 27–36 imposing costs on lights that are turned ON, the planner will proceed by turning OFF $l5$ as shown in sequence 6 of Fig. 8.

Though not directly shown, for case 1 in Section IV-C, the smart home will also exhibit the switch from *tv2* to $tv1$ if the host walks to the bedroom, due to constraint 1 in Table VII. Additionally, *tv2* will not be switched OFF unless additional constraints like those of 27–36 are specified for the TV.

Case 3 (Plug-in Function Demonstration): Case 3 demonstrates the use of plug-in function, where the result is used

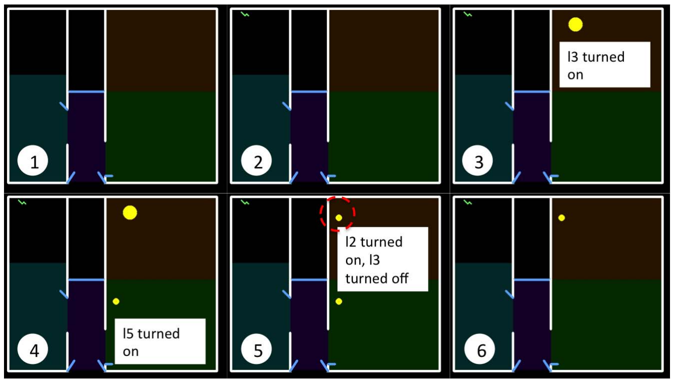


Fig. 8. Device visualization for Case 2. Small yellow circle is a dim light, and large yellow circle is a bright light.

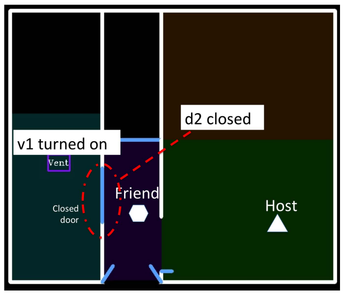


Fig. 9. Device visualization for Case 3. Blue bars indicate doors. Here, door $d2$ is closed.

to determine intelligent plans, as CSP does not concern itself with the data structure and function used, as long as truth values can be obtained.

This case is used on smoke occurrence in the kitchen when both the host and friend are in the kitchen. Although smoke occurrence is considered a safety issue and that it should be given careful consideration from planning point of view in real life, here, the case is just to demonstrate the use of plug-in function. The function is *connected*, as shown in constraints 39 and 40, where it determines whether two rooms are directly connected or not. It makes decision based on the door states and layout of the house. Where there is smoke, the vent should be turned ON to clear the room of smoke. It also tries to disconnect people from the smoke.

When smoke occurs, *KitchenFireEvent* is triggered, causing constraints 39–41 to be considered. Constraint 41 causes the vent to be turned ON. The friend goes to the entrance area and the host goes to the living room. Given this situation, both person can be disconnected from the smoke by closing door $d2$ to fulfill constraints 39 and 40. Thus, the planner will proceed with this plan. The final device state is shown in Fig. 9. The doors will not be closed if they are in the kitchen as *connected* cannot be true. Therefore, the best plan is to do nothing, where no additional operator costs are imposed. In this case, the smart home can be thought of as waiting for them to get out of the kitchen before it proceeds to closing the door.

One can add numerous plug-in functions as goal evaluation can be done extremely fast, which is limited by the complexity of the functions themselves.

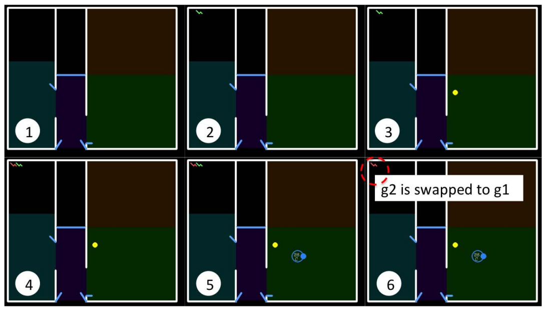


Fig. 10. Device visualization for Case 4. Blue circle is the air conditioner, and the red thunder at the top left indicates the normal generator.

Case 4 (Demonstration of Extended Goals): We demonstrate the use of extended constraints in this case. There are two extended constraints as shown in Table VII, which require at least one generator to be ON if light, fan, or TV is to be turned ON, and that normal generator needs to be turned ON if the air conditioner is to be turned ON. One can consider extended constraints as additional preconditions for every operator.

The case starts off with the host and the friend in the living room at night. Sequence 4 of Fig. 10 shows that dim light $l5$ and efficient generator $g2$ are turned ON to maximize constraint fulfillment. The planner turns ON $g2$ before switching $l5$ to ON as per required by extended constraint $E1$.

The host then requests for the air conditioner to be turned ON. As AC requires normal generator to be ON as per required by $E2$, $g1$ is switched ON as shown in sequence 2, after which AC is switched ON in sequence 3.

Periodic constraint 38 states that at most only one generator can be switched ON. This will cause $g2$ to be switched OFF as shown in sequence 4. Previous constraints are not violated as lights, fan, and TV can also run under $g1$.

D. Issues to be Addressed

Case studies demonstrate that the system can perform rule association for planning. It also shows the ability of using complex rules derived from reasoning (via knowledge representation) as goals. Despite that, various issues require addressing are listed as follows.

Services for devices and QOL constraints and costs have to be manually defined by the user or manufacturers. Since planning and QOL definition operates separately, external systems can be easily plugged in for adaptive QOL specification.

There may be cases where additional rules can be included into the services to aid planning and deliver more complex service sequence. An example is to be able to know a light sensor will be activated if a nearby light is turned ON. Learning is possible to reduce the manual effort required as shown in [42].

Explosion of number of generated activities is an issue given badly designed service parameters; as currently, services are duplicated if multiple variable associations of variables are found. Besides that, increase of devices will introduce more activities, especially for a commercial building or community environment. One way to alleviate this is to introduce metrics (such as the person's distance from the devices) and their privileges to determine whether a device's operation should be added to his/her repertoire of planning operators or not. A more efficient solution is to apply plan chunking (although fast automatic chunking is still considered an unsolved AI problem given scarce data as in the case of personalized control) or automatic operator refinement (which is also at its infancy [43]).

Currently, cost for activities and cost for not fulfilling constraints are manually set by user. Improvements should be made such that costs can be automatically set or at least provide a simpler alternative for users to choose from. That said, this paper divides the task of planning from QOL specification, and thus, whichever adaptive method of introducing goals and cost setting can be directly used, given that it does not need to interfere with the process of planning. For adaptive goal introduction into the QOL, one can use *ad hoc* systems to add, modify, or delete goals from the QOL list. A much more adaptive way that supports learning is to use database for continual data collection and relational machine learning like the Markov logic network [44] to extract and learn rules, where goals can be controlled via inference. On the other hand, for automatic cost setting, one may also use *ad hoc* or adaptive control such as using satisfiability modulo theories (SMT) solver or linear programming to allocate costs. (For example, the user just specifies that the total cost for everyone other than the host should not exceed the cost of each of the host's QOL cost. The SMT solver will then select the appropriate costs on the go.) More work has to be done for more intelligent and automatic assignment of costs.

Another issue related to cost is the danger of being overwritten. If there are conflicting goals between people in higher privileges (like the host) and people of lower privileges (like the friend or strangers), the planner will try to maximize the constraints that manifest itself as trying hard to fulfill constraints for the higher privileged people. Danger occurs if a group of strangers have the same constraints that are conflicting with the host's. The total cost contributed by the group of strangers will cause optimization to occur in their favor.

V. Conclusion

Intelligent plan composition and optimization is implemented on the smart home through using building ontology and solving weighted CSP. Building ontology is used to provide a schema of knowledge representation, including knowledge of building layout and devices connected to the smart home. Through rule associations for service wrap-up, variables and services provided by individual devices can be related to each other. Via representing the problem as weighted CSP given weighted goals, the solution obtained is the sequence of activities that try to fulfill as much personalized constraints as possible. Case study shows the system is capable of composing and executing optimized plans for conflicting constraints. Besides, it shows potential in composing services for complex rules derived from the building ontology. As explain in Section IV-D, this paper is not without limitations and has a lot of room to grow.

The vision of this paper is to be able to implement human-centric system to a community or city. Devices installed by anyone who participated in the human-centric system will be shared in the sense that all these devices will try to cooperate to maximize the QOL of the community.

References

[1] M. Weiser, "The computer for the 21st century," *Sci. Amer.*, vol. 265, no. 3, pp. 94–104, 1991.

[2] G. Leitner, "The WISE future of home technology," in *The Future Home is Wise, Not Smart* (Computer Supported Cooperative Work), R. Harper, Ed. Cham, Switzerland: Springer, 2015.

[3] I. Iida and T. Morita, "Overview of human-centric computing," *Fujitsu Sci. Tech. J.*, vol. 48, no. 2, pp. 124–128, 2012.

[4] E. Kaldeli, E. Warriach, A. Lazovik, and M. Aiello, "Coordinating the Web of services for a smart home," *ACM Trans. Web*, vol. 7, no. 2, pp. 10:1–10:40, 2013.

[5] N. N. W. Tay, J. Botzheim, and N. Kubota, "Weighted constraint satisfaction for smart home automation and optimization," *Adv. Artif. Intell.*, vol. 2016, Oct. 2016, Art. no. 2959508.

[6] L. Bevilacqua, A. Furno, V. S. di Carlo, and E. Zimeo, "A tool for automatic generation of WS-BPEL compositions from OWL-S described services," in *Proc. 5th Int. Conf. Softw., Knowl. Inf., Ind. Manage. Appl.*, Sep. 2011, pp. 1–8.

[7] Z. Shamszaman, S. Ara, I. Chong, and Y. Jeong, "Web-of-objects (WoO)-based context aware emergency fire management systems for the Internet of Things," *Sensors*, vol. 14, no. 2, pp. 2944–2966, 2014.

[8] S. Van Hoecke, R. Verborgh, D. Van Deursen, and R. Van de Walle, "SAMuS: Service-oriented architecture for multisensor surveillance in smart homes," *Sci. World J.*, vol. 2014, Mar. 2014, Art. no. 150696.

[9] L. Durkop, H. Trsek, J. Otto, and J. Jasperneite, "A field level architecture for reconfigurable real-time automation systems," in *Proc. 10th IEEE Workshop Factory Commun. Syst.*, May 2014, pp. 1–10.

[10] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Semantics-based composition of factory automation processes encapsulated by Web services," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2349–2359, Nov. 2013.

[11] I. Georgievski, "Planning for coordination of devices in energy-smart environments," in *Proc. 23rd Int. Conf. Autom. Planning Scheduling*, 2013, pp. 1–5.

[12] M. Papazoglou and W. van den Heuvel, "Service oriented architectures: Approaches, technologies and research issues," *VLDB J.*, vol. 16, no. 3, pp. 389–415, 2007.

[13] D. Bonino and F. Corno, "Rule-based intelligence for domotic environments," *Autom. Construction*, vol. 19, no. 2, pp. 183–196, 2010.

[14] M. Ruta, F. Scioscia, E. Di Sciascio, and G. Loseto, "Semantic-based enhancement of ISO/IEC 14543-3 EIB/KNX standard for building automation," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 731–739, Nov. 2011.

[15] T. Di Noia, E. Di Sciascio, and F. Donini, "Semantic matchmaking as non-monotonic reasoning: A description logic approach," *J. Artif. Intell. Res.*, vol. 29, pp. 269–307, May 2007.

[16] S. N. Han, G. M. Lee, and N. Crespi, "Semantic context-aware service composition for building automation system," *IEEE Trans. Ind. Informat.*, vol. 10, no. 1, pp. 752–761, Feb. 2014.

[17] E. Kaldeli, E. Warriach, J. Bresser, A. Lazovik, and M. Aiello, "Interoperation, composition and simulation of services at home," in *Service-Oriented Computing* (Lecture Notes in Computer Science), vol. 6470, P. P. Maglio, M. Weske, J. Yang, and M. Fantinato, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 167–181.

[18] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu, "Web services composition: A decade's overview," *Inf. Sci.*, vol. 280, pp. 218–238, Oct. 2014.

[19] M. Klusch, A. Gerber, and M. Schmidt, "Semantic Web service composition planning with OWLS-XPLAN," in *Proc. AAAI Fall Symp. Semantic Web Agents*, 2005, pp. 55–62.

[20] O. Hatzi, D. Vrakas, N. Bassiliades, D. Anagnostopoulos, and I. Vlahavas, "Semantic awareness in automated Web service composition through planning," in *Artificial Intelligence: Theories, Models and Applications* (Lecture Notes in Computer Science), vol. 6040, S. Konstantopoulos, S. Perantonis, V. Karkaletsis, C. D. Spyropoulos, and G. Vouros, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 123–132.

[21] O. Hatzi, D. Vrakas, M. Nikolaidou, N. Bassiliades, D. Anagnostopoulos, and L. Vlahavas, "An integrated approach to automated semantic Web service composition through planning," *IEEE Trans. Serv. Comput.*, vol. 5, no. 3, pp. 319–332, Jul. 2012.

[22] I. Georgievski and M. Aiello, "HTN planning: Overview, comparison, and beyond," *Artif. Intell.*, vol. 222, pp. 124–156, May 2015.

[23] E. Sirin, B. Parsia, D. Wu, J. Hendler, and D. Nau, "HTN planning for Web service composition using SHOP2," *Web Semantics, Sci., Services Agents World Wide Web*, vol. 1, no. 4, pp. 377–396, 2004.

[24] T.-C. Au, U. Kuter, and D. Nau, "Web service composition with volatile information," in *The Semantic Web* (Lecture Notes in Computer Science), vol. 3729, Y. Gil, E. Motta, V. Benjamins, and M. Musen, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 52–66.

[25] F. Yang, P. Khandelwal, M. Leonetti, and P. Stone, "Planning in answer set programming while learning action costs for mobile robots," in *Proc. AAAI Spring Symp. Knowl. Represent. Reason. Robot. (AAAI-SSS)*, 2014, pp. 71–78.

[26] S. Zhang, M. Sridharan, and J. Wyatt, "Mixed logical inference and probabilistic planning for robots in unreliable worlds," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 699–713, Jun. 2015.

[27] E. Kaldeli, A. Lazovik, and M. Aiello, "Domain-independent planning for services in uncertain and dynamic environments," *Artif. Intell.*, vol. 236, pp. 30–64, Jul. 2016.

[28] M. Helmert, "Concise finite-domain representations for PDDL planning tasks," *Artif. Intell.*, vol. 173, nos. 5–6, pp. 503–535, 2009.

[29] S. Richter and M. Westphal, "The LAMA planner: Guiding cost-based anytime planning with landmarks," *J. Artif. Intell. Res.*, vol. 39, no. 1, pp. 127–177, 2010.

[30] J. Hoffmann, I. Weber, and F. Kraft, "Sap speaks PDDL," in *Proc. 24th Nat. Conf. Amer. Assoc. Artif. Intell.*, 2010, pp. 1096–1101.

[31] M. B. Alaya, Y. Banouar, T. Monteil, C. Chassot, and K. Drira, "OM2M: Extensible ETSI-compliant M2M service platform with self-configuration capability," *Procedia Comput. Sci.*, vol. 32, pp. 1079–1086, Jan. 2014.

[32] S. Mayer, R. Verborgh, M. Kovatsch, and F. Mattern, "Smart configuration of smart environments," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 3, pp. 1247–1255, Jul. 2016.

[33] J. Larrosa, E. Morancho, and D. Niso, "On the practical use of variable elimination in constraint optimization problems: 'Still-life' as a case study," *J. Artif. Intell. Res.*, vol. 23, pp. 421–440, Jan. 2005.

[34] J. H. Moreno-Scott, J. C. Ortiz-Bayliss, H. Terashima-Marín, and S. E. Conant-Pablos, "Experimental matching of instances to heuristics for constraint satisfaction problems," *Comput. Intell. Neurosci.*, vol. 2016, Jan. 2016, Art. no. 7349070.

[35] R. Dechter, "Bucket elimination: A unifying framework for reasoning," *Artif. Intell.*, vol. 113, no. 1, pp. 41–85, 1999.

[36] E. Kaldeli, A. Lazovik, and M. Aiello, "Extended goals for composing services," in *Proc. Int. Conf. Autom. Planning Scheduling*, 2009, pp. 1217–1218.

[37] V. Degeler and A. Lazovik, "Dynamic constraint reasoning in smart environments," in *Proc. IEEE 25th Int. Conf. Tools Artif. Intell.*, Nov. 2013, pp. 167–174.

[38] F. Corno and F. Razzak, "Real-time monitoring of high-level states in smart environments," *J. Ambient Intell. Smart Environ.*, vol. 7, no. 2, pp. 133–153, 2015.

[39] D. Tsarkov and I. Horrocks, "FaCT++ description logic reasoner: System description," in *Automated Reasoning* (Lecture Notes in Computer Science), vol. 4130, U. Furbach and N. Shankar, Eds. Berlin, Germany: Springer, 2006, pp. 292–297.

[40] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Germany: Springer-Verlag, 2008, pp. 337–340.

[41] J. Botzheim, J. Woo, N. T. N. Wi, N. Kubota, and T. Yamaguchi, "Gestural and facial communication with smart phone based robot partner using emotional model," in *Proc. World Autom. Congr. (WAC)*, Aug. 2014, pp. 644–649.

[42] K. Rasch, "Smart assistants for smart homes," Ph.D. dissertation, School Inf. Commun. Technol., KTH, Stockholm, Sweden, 2013.

[43] D. S. Nau, M. Ghallab, and P. Traverso, "Blended planning and acting: Preliminary approach, research challenges," in *Proc. AAAI*, 2015, pp. 4047–4051.

[44] M. Richardson and P. Domingos, "Markov logic networks," *Mach. Learn.*, vol. 62, no. 1, pp. 107–136, 2006.

Noel Nuo Wi Tay received the bachelor's degree in electronics engineering from the Faculty of Engineering and Technology, Multimedia University (MMU), Cyberjaya, Malaysia, in 2008, and the master's degree in science from the Faculty of Information Science and Technology, MMU, in 2013, where he is working on biologically inspired visual system. He received the Ph.D. degree from Tokyo Metropolitan University, Tokyo, Japan, in 2017, under the supervision of Prof. N. Kubota.

His research interest includes face recognition, human activity monitoring, smart home, and ambient intelligence.

János Botzheim (M'11) received the M.Sc. and Ph.D. degrees in computer science from the Budapest University of Technology and Economics, Budapest, Hungary, in 2001 and 2008, respectively.

He is an Associate Professor with the Department of Automation, Széchenyi University, Gyor, Hungary. His research interests include computational intelligence and cognitive robotics.

Dr. Botzheim is a member of several scientific societies, such as John von Neumann Computer Science Society and Hungarian Fuzzy Association.

Naoyuki Kubota received the B.Sc. degree from Osaka Kyoiku University, Kashiwara, Japan, in 1992, the M.Eng. degree from Hokkaido University, Hokkaido, Japan, in 1994, and the D.E. degree from Nagoya University, Nagoya, Japan, in 1997.

He joined the Osaka Institute of Technology, Osaka, Japan, in 1997. In 2000, he joined the Department of Human and Artificial Intelligence Systems, Fukui University, Fukui, Japan, as an Associate Professor. He joined the Department of Mechanical Engineering, Tokyo Metropolitan University, Tokyo, Japan, in 2004, where he is a Professor with the Department of System Design.