Hierarchical Task Recognition and Planning in Smart Homes with Partial Observability

Dan Wang(✉) and Jesse Hoey

David R. Cheriton School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada
d97wang@uwaterloo.ca, jhoey@cs.uwaterloo.ca

Abstract. This paper proposes a goal recognition and planning algorithm, HTN-GRP-PO, to enable intelligent assistant agents to recognize older adults' goals and reason about desired further steps. It will be used in a larger system aimed to help older adults with cognitive impairments to accomplish activities of daily living independently. The algorithm addresses issues including partial observability due to unreliable or missing sensors, concurrent goals, and incorrectly executed steps. The algorithm has a Hierarchical Task Network basis, which enables it to deal with partially ordered subtasks and alternative plans. We test on simulated cases of different difficulties. The algorithm works very well on simple cases, with accuracy close to 100%. Even for the hardest cases, the performance is acceptable when sensor reliabilities are above 0.95.

Keywords: Hierarchical task network · Goal recognition · Partial observability · Cognitive impairments

1 Introduction

Nowadays, more and more older adults suffer from cognitive impairments, which cause difficulties in activities of daily living (ADLs) [3]. Developing intelligent assistant agents (IAAs) in smart homes to guide them on ADLs becomes urgent. IAAs are intelligent real-time reminders, prompting the older adult whenever he/she is confused in ADLs. IAAs should at least gather sensor signals, be aware of situations [2], recognize ongoing goals, and present effective assistances [9].

Due to limitations of sensors and privacy concerns, not all attributes of physical objects can be measured. Thus IAAs should cope with partial observability due to missing or unreliable sensors. Older adults with cognitive impairments commonly execute ADLs with irrational, repeated and disordered steps. IAAs are required to identify these improper behaviors and present help. Geib et al. [5] discussed several critical considerations of goal recognition for older adults, including multiple concurrent goals, actions used for multiple effects, and failure to observe. According to Hoey et al. [8], smart home assistance should be as passive as possible, so as to maintain feelings of independence.

An IAA helping older adults with cognitive impairments on their ADLs should address the following aspects: (1) Tolerate partial observability caused

S.F. Ochoa et al. (Eds.): UCAmI 2017, LNCS 10586, pp. 439–452, 2017.
DOI: 10.1007/978-3-319-67585-5_45

by missing and unreliable sensors; (2) Recognize concurrent goals; (3) Detect improper steps and rectify the older adult from mistakes; and (4) Present hints or prompts of various detail levels, such as desired next steps or higher level tasks. The proposed **HTN-GRP-PO** algorithm[1] addresses these issues. It adopts the hierarchical paradigm as defined in Hierarchical Task Network (HTN) planning [4]. With HTN, the goal recognition process to recognize ongoing goals and the planning process to generate feasible next steps (or tasks) are combined together. Partially ordered subtasks, alternative ways to achieve a goal, and preconditions of tasks and steps are considered thanks to the expressive power of HTN. When the algorithm is running, it tracks the ongoing goals, updates beliefs based on new observations, reports wrong steps, and presents prompts with different details. Issues like unreliable or missing sensors, concurrent goals, and wrong steps make the problem more difficult, and decrease the performance of the algorithm.

The paper is arranged as follows. Section 2 discusses related works. Section 3 defines the problem, Sect. 4 details the HTN-GRP-PO algorithm, Sect. 5 reports on simulations *in silico*, and Sect. 6 concludes the work.

2 Related Works

Kautz et al. defined goal recognition as finding possible top level tasks (goals) to explain a set of observed steps [10]. The definition of goal recognition indicates its hierarchical nature. In [1], goal recognition is classified as either single layer or hierarchical. For single layer approaches, the reasoning process matches the raw data observations directly to goals. The inference from data to goals, without considering intermediate level tasks, makes status tracking not feasible. So a non-hierarchical method is inappropriate for the IAA in this work.

Hierarchical approaches recognize the highest level goals and inner level subgoals. A milestone in goal recognition is the conceptual framework published by Kautz and Allen [10]. They proposed the hierarchy format to represent top level tasks and low level steps, and the concept of decomposition. Advantages of hierarchical approaches are summarized in [1], including the suitability for recognizing high level tasks with complex structures, interactions with humans and incorporating prior knowledge into the representation. HTN-based and ontology-based hierarchical approaches are two main streams relating to this work.

HTN, a terminology in planning, is firstly proposed by Sacerdoti and Earl in 1975 [15]. The seminal work by Goldman et al. [7] proposed HTN for goal recognition. Their framework is called PHATT (Probabilistic Hostile Agent Task Tracker), which can deal with partially ordered subtasks, overloaded steps, contextual influence on choices of steps and goals, and observation failures. An additional module is added to PHATT to identify abandoned goals in [5]. Follow-up work [6] summarized previous works and integrated PHATT with a constraint reasoning module for parametrized actions and temporal constraints.

[1] "Hierarchical Task Network based Goal Recognition and Planning with Partial Observability".

The PHATT framework is very powerful, but assumes full observability of steps, which is impossible in reality.

Ontology-based approaches highlight the modeling of activities and behaviors with rich semantics. They characterized activities into atomic, simple, and composite ones [13]. Composite activities are formulated using both ontological and temporal modeling formalisms. In [14], two types of composite activities (concurrent and interleaved) and sequential activities are handled. They use ontological reasoning for simple activity recognition and rule-based temporal inference to recognize composite activities. To reason about temporal constraints among subtasks of a composite task, Okeyo et al. [12] proposed a hybrid ontological and temporal approach to model composite activities. Their work focused on temporal constraints, not addressing partial observability.

Our algorithm is a HTN-based approach, where knowledge base is expressed in methods and operators using similar formats described in SHOP2 [11]. It adopts the plan execution concept in [7]. However, our algorithm explicitly considers partial observation of steps and the feasibility of pending steps (or tasks). The algorithm integrates goal recognition in planning by utilizing HTN. It can not only recognize what the older adults is trying to do, but also what are the proper next steps and tasks in order to achieve the recognized goals.

3 Problem Description

Helping older adults with cognitive impairments to implement ADLs needs to recognize ongoing goals and to present prompts for next steps, which is a combination of goal recognition and planning. The definition of the goal recognition and planning problem in this work is given in Definition 1.

Definition 1 (Goal Recognition and Planning Problem). *It is a tuple*

$$P^{rp} = (bs, obs, G, prior, D, PROB, PS),$$

where bs *is belief state,* obs *is sensor readings,* G *is a set of goals,* $prior$ *is the prior probabilities of goals in* G. $PROB$ *is a distribution showing the goal recognition result,* PS *is the planning result with multiple levels, showing the next tasks and steps in order to achieve* $PROB$ *together with probabilities. Its step level is* PS_{step}. $D = (O, M)$ *is the knowledge base (methods and operators).*

Table 1 (left) shows method *prepare-hot-water*. $mName$ is the task name that m can be applied to. A method having multiple branches with each has *precondition* and *subtasks* indicates multiple ways to accomplish a task. *parent* specifies methods whose *subtasks* contains $mName$. *startStep* are the beginning steps of a goal. It is present only when m stands for a goal. Table 1 (right) is an operator which can be applied to step *turn-on-faucet-1*. $effect$ contains fluents which become true after executing the step. It has similar format to that of *precondition*. *parent* specifies all methods whose *subtasks* contains $oName$.

The problem in this work is classified into eight categories (Table 2) based on three properties: the number of goals that the executed steps account for,

Table 1. Method (left) and Operator (right) example

$nName$	prepare-hot-water
$precondition$	{[(kettle-1, has-water, yes), (kettle-1, switch, off), (kettle-1, water-hot, not)]}
$subtasks$	{[kettle-1-heat-water: {pre: [], dec:[]}]}
$parent$	[make-coffee, make-tea]
$startStep$	NA (not a goal)

$oName$	turn-on-faucet-1
$precondition$	(faucet-1, state, off)
$effect$	[(faucet-1, state, on)]
$parent$	[wash-hand, kettle-1-add-water]

Table 2. Problem categories

Sensor config.	Single goal		Multiple foals	
	Correct step	Wrong step	Correct step	Wrong step
Reliability	**p1**	**p2**	**p3**	**p4**
Missing sensor	**p5**	**p6**	**p7**	**p8**

present wrong steps or not, with unreliable sensor or missing sensor. In the "Sensor Config." column, "Reliability" means that every sensor has a reliability. "Missing Sensor" means that some sensors are missing and the agent knows about which ones are missing. As one can imagine, **p1** is the easiest problem category, while **p8** is the hardest one.

4 Algorithm

4.1 Terminologies

Changes of sensor measurements at a time point will trigger an algorithm **iteration**. An iteration reasons about the new bs and the new $PROB$ and PS result by adding the observations from the just happened step. Simultaneous steps are not considered in this work. An example iteration, shown in Table 3, changes P_0^{rp} to P_1^{rp}. Note G, $prior$, and D are neglected because of no change. PS_1 has several levels to provide help in different details. Only the step level (level 0) and a task level (level 1) are shown to save space. obs is not the outcome but the trigger of an iteration. The iteration in Table 3 is triggered by obs_1. Similarly, obs_2 will trigger the next iteration.

The proposed algorithm lies on explanations. Typically, a goal recognition result should explain observations so far. Multiple explanations exist when considering partial observability. The recognition result would be a distribution over possible explanations. To obtain next steps or tasks hint, ongoing statuses of goals should be tracked. Based on those considerations, Definition 2 shows the structure used to explain observations so far in this work.

Table 3. The outcome of an algorithm iteration

P_0^{rp}		P_1^{rp}	
Variable	Value	Variable	Value
bs_0	(*faucet-1*, *state*, {*off*: 0.999, *on*: 0.001})	bs_1	(*faucet-1*, *state*, {***off***: 0.0001, ***on***: 0.9999})
obs_1	[(*faucet-1*, {*state*, *on*})]	obs_2	[(*hand-1*, {*soapy*: *yes*, *dry*: *no*})]
$PROB_0$	*wash-hand*: 0.333, *make-coffee*: 0.333, *make-tea*: 0.333	$PROB_1$	*wash-hand*: 0.3574, *make-coffee*: 0.3213, *make-tea*: 0.3213
PS_0	level-0: *turn-on-faucet-1*: 0.666, *switch-on-kettle-1*: 0.333	PS_1	level-0: *use-soap*: 0.357, *add-water-kettle-1*: 0.643 level-1: clean-hand:0.357, prepare-hot-water:0.643

Table 4. Explanations after the Iteration shown in Table 3

Variable	$expla_1$	$expla_2$	$expla_3$
$prob$	0.3574	0.3213	0.3213
$forest$	[$goalN_1$], in Table 5	[$goalN_2$]	[$goalN_3$]
$pendingStep$	[*use-soap*]	[*add-water-kettle-1*]	[*add-water-kettle-1*]
$startGoal$	*wash-hand*: ***True***, *make-tea*: *False*, *make-coffee*: *False*	*wash-hand*: *False*, *make-tea*: ***True***, *make-coffee*: *False*	*wash-hand*: *False*, *make-tea*: *False*, *make-coffee*: ***True***

Definition 2 (Explanation). *An explanation, $expla \in ExplaSet$, is a tuple*

$$expla = (prob, forest\ \ [\,], pendingStep\ \ [\,], startGoal\{\}),$$

where prob tells to which degree we can rely on this explanation. $forest$ is a list, with each element recording the ***ongoing status*** *of a goal. pendingStep is the next steps suggested by the explanation to proceed towards ongoing goals. startGoal records goals that are ongoing in this explanation.*

Table 5. $goalN_1$ for $expla_1$ in Table 4

Variable	Value
$goalName$	*wash-hand*
$tree$	$tree_1$, in Fig. 1
$expandProb$	1.0
$pendingGoalNet$	[$decompGN_1$] in Fig. 1
$completeness$	*False*
$executeSequence$	{*turn-on-faucet-1*, (*faucet-1*, *state*, *on*)}

Each iteration computes $PROB$ and PS based on explanations (see Definition 2), which are stored in $ExplaSet$. Multiple explanations might exist to explain a given observation series. The iteration in Table 3 gets $ExplaSet$ containing $expla_1$, $expla_2$ and $expla_3$, with each a complete explanation for $obs = \{obs_1\}$. They are shown

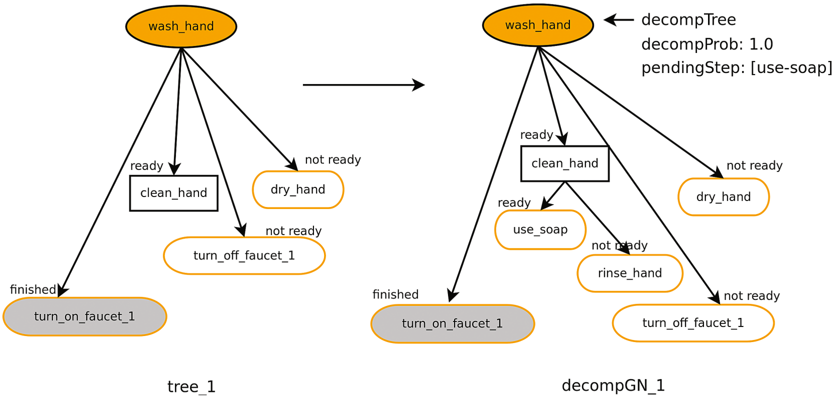

Fig. 1. $tree_1$ and $decompGN_1$ in $goalN_1$

in Table 4. $expla_1$ believes that *wash-hand* is ongoing and the supposed next step is *use-soap*. $goalN_1$ records the ongoing status of *wash-hand*. Table 5 and Fig. 1 explains the **goal network**. *tree* is a hierarchical task network reflecting the ongoing status of the goal $goalName$. For example, $tree_1$ in Fig. 1 shows that *turn-on-faucet-1* for *wash-hand* has been finished. *expandProb* tells the probability of this way being chosen. *pendingGoalNet* is the results of decomposing *tree*, which is a list. Each element in *pendingGoalNet* is a feasible way to proceed towards $goalName$ from the status in *tree*. Figure 1 tells *use-soap* proceeds from $tree_1$ towards *wash-hand*. Only $decompGN_1$ is derived means that there is only one way to decompose *clean-hand* which explains $decompProb = 1.0$.

In summary, an iteration reasons with **Explanation**s, which are stored in **Explanation Set**. A **goal network** in an explanation's $forest$ explains the ongoing status of a goal. More than one goal networks in $forest$ indicate concurrent goals. A **decomposed goal network** in a goal network's *pendingGoalNet* stands for a specific way to proceed towards the corresponding goal.

4.2 The HTN-GRP-PO Algorithm

Figure 2 is the algorithm flow chart, with the iteration in Table 3 as the example. Inputs and outputs of each module are included. **Compute** PS_{step} **Posterior** is the step recognition process adopting Bayesian inference as shown in Eqs. 1 and 2. Its output is $(PS_{step})_{posterior}$. Equation 1 is applied to every step in $(PS_0)_{step}$. Note that 0.999 is used in Eq. 2 because when the precondition of st_t is not satisfied, it is usually impossible to happen. $p(st_t)$ in Eq. 1 takes the corresponding probability in $(PS_0)_{step}$. Equation 3 explains wrong steps detection. Comprehensive experiment results show that if *otherHappenProb* is bigger than **0.75**, a wrong step happens.

Update *bs* also adopts Bayesian inference. Because the "wrong step" branch is dropped, $(PS_{step})_{posterior}$ is normalized to get $(PS_{step})'_{prior}$ which become

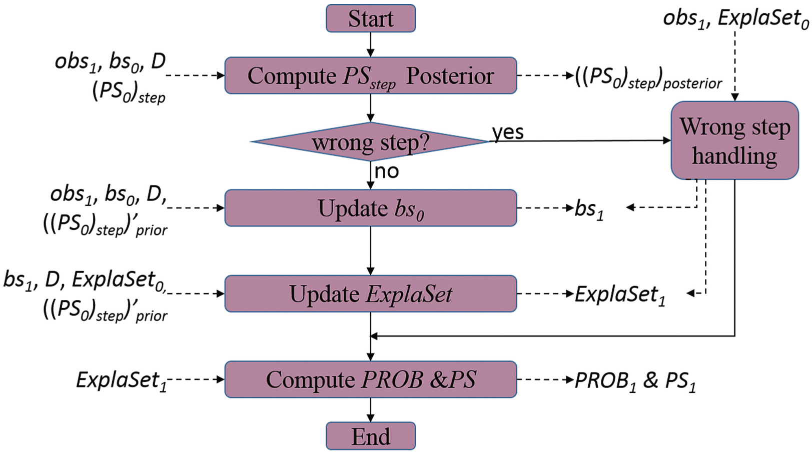

Fig. 2. An algorithm iteration

the new priors of steps. The algorithm applies Eqs. 4 and 2 to every attribute related to the current iteration. Given an attribute, the sum over s_{t-1} in Eq. 4 enumerates all possible values of the attribute.

$$\begin{aligned} p(st_t|obs_t) &= \frac{p(st_t, obs_t)}{p(obs_t)} \propto p(st_t, obs_t) = \sum_{s_t}\sum_{s_{t-1}} p(st_t, s_t, s_{t-1}, obs_t) \\ &= \sum_{s_t}\sum_{s_{t-1}} p(s_t|s_{t-1}, st_t) \times p(obs_t|s_t) \times p(s_{t-1}) \times p(st_t) \end{aligned} \tag{1}$$

$$p(s_t|s_{t-1}, st_t) = \begin{cases} 0.999, \text{ if } st_t(precondition) \subset s_{t-1} \; and \;\; \theta(st_t, s_{t-1}) \subset s_t \\ 0.001, \text{ otherwise} \end{cases} \tag{2}$$

$$otherHappenProb = 1 - \sum_{st \in PS_{step}} (PS_{step})_{posterior}(st) \tag{3}$$

$$p(s_t|obs_t) = \sum_{s_{t-1}} \sum_{st'_t \in (PS_{step})'_{prior}} p(s_t|s_{t-1}, st'_t) \times p(obs_t|s'_t) \times p(s_{t-1}) \times p(st'_t) \tag{4}$$

$$new\_expla(prob) = st_{prob} \times goalNet(expandProb) \times expla(prob) \tag{5}$$

Update *ExplaSet*. Given a step $st \in (PS_{step})'_{prior}$, each explanation $expla \in ExplaSet_0$ will be updated to several new ones, which are stored in $ExplaSet_1$. It includes two procedures: recognition and decomposition. The **recognition** procedure adopts a new $goalNet$ to represent the new ongoing status of the corresponding goal and computes the new explanation probability using Eq. 5. The creation of the new $goalNet$s has two cases.

Case 1, st starts a new goal. Thus there is no $goalNet_{base}$ for creating the new one. A bottom up procedure is used to create a new $goalNet$ from scratch.

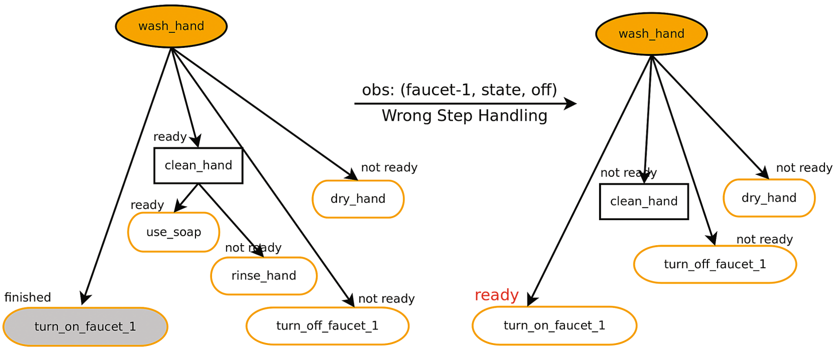

Fig. 3. A wrong step handling example

For example, with st = *turn-on-faucet-1*, when creating $goalN_1$ for $expla_1$ in Table 4, the bottom up procedure creates $tree_1$ as shown in Table 1. Note that case 1 enables the algorithm handle concurrent goals. Case 2, st continues an ongoing goal. In this case, a proper decomposed goal network is chosen from the given $goalNet(pendingGoalNet)$ as the new $goalNet$. For example, given $expla_1$ in Table 4 and st = *use-soap*, $decompGN_1$ (Fig. 1, right) will replace $goalN_1$, becoming the new $goalNet$ in the new explanation.

The **decomposition** procedure creates $pendingGoalNet$ for a $goalNet$. In Table 5, $goalN_1(pendingGoalNet)$ is obtained through the decomposition procedure. The decomposition result is shown in the right part of Fig. 1. When applying methods for decomposition, the probability that a precondition is satisfied is computed and accumulated to derive *decompProb*, which indicates to which degree the corresponding decomposition path is feasible in bs. The decomposition process that ends every leaf in $tree$ is either a node standing for a step or a node standing for a task satisfying $node(data)(readiness) == False$.

Wrong Step Handling. This module rectifies existing explanations so as to restore from the wrong step. Figure 3 is an visualization example. Assume that *expla* contains ongoing status of *wash-hand* as shown in the left tree of Fig. 3. So the desired next step is *use-soap*. However, a wrong step is reported during the computation of $(PS_{step})_{posterior}$. The observation indicates that the effect of step *turn-on-faucet-1* has been destroyed by the wrong step. The wrong step handling module rectifies the ongoing status of *wash-hand* to the point as shown in the right tree of Fig. 3. Consequently, the algorithm will remind the the older adult to do *turn-on-faucet-1* again.

Compute *PROB* and *PS*. This module purely depends on the latest *ExplaSet*. The probability of goal g in $PROB$ is the sum of probabilities of explanations whose *startGoal* contains g. The probability of a task t(or step st) in PS is the sum of probabilities of explanations whose $forest$ contains a node standing for t (or st) with *completeness* being false while *readiness* being true.

5 Experimental Simulations

5.1 Knowledge Base, Sensors, and Simulator

Scenario. Helen is an older adult with mild Alzheimer's disease. She has problems doing three daily tasks in the kitchen: washing hands, making a cup of tea, and making a cup of coffee. Her caregiver reports her common mistakes. When washing hands, she might forget to use soap or turn the faucet off, or repeatedly rinse her hands. Similar issues happen when making a cup of tea or coffee. The caregiver hopes an IAA can help her complete those tasks independently.

The **Knowledge Base** has three goals : *wash-hand*, *make-tea* and *make-coffee*. Although M and O are individual pieces, they implicitly indicate a hierarchical plan graph, as shown in Fig. 4. Root nodes stand for goals G. Leaf nodes are the lowest level steps. Other internal nodes are inner level tasks. Each goal or task node corresponds to a method in M. Each step node corresponds to a step in O. To save space, details of M and O are not given.

According to the knowledge base, 18 virtual binary **sensors** (Table 6) are set up for the sake of simulation. Sensor reliability has four values, [0.99, 0.95, 0.9, 0.8]. We use ID to refer an sensor. *obj* and *att* determines which attribute the sensor is monitor. The **simulator** simulates real environment state changes and step executions. No real human are involved in the experiment, however, our study is applicable to cases in reality. Given an input step, the simulator firstly updates real state according to the effects of the step, and then changes sensor measurements based on the simulated real state and sensor reliability.

5.2 Test Cases and Evaluation Criteria

Each test case is a list of steps in the order of execution. It accounts for one single goal or multiple goals. Noisy wrong steps can exist in the list. The algorithm reasons about $PROB$ and PS for each step. The ground truth of each step's

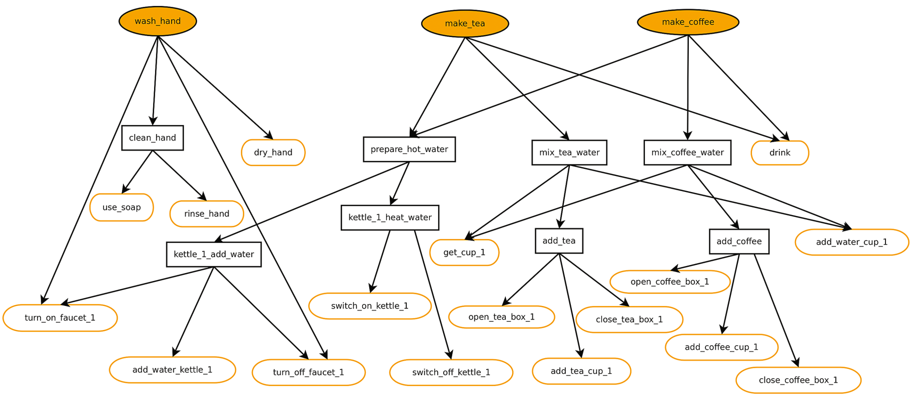

Fig. 4. The hierarchical task network for experiment

Table 6. Sensors used in the experiment (initial values are in **boldface**)

ID	Obj	Att	Value	ID	Obj	Att	Value
1	hand-1	soapy	**no**, yes	10	kettle-1	water-hot	**no**, yes
2	hand-1	dirty	**yes**, no	11	cup-1	location	**cabinet**, table
3	hand-1	dry	**yes**, no	12	cup-1	has-water	**no**, yes
4	faucet-1	state	**on**, off	13	cup-1	has-tea	**no**, yes
5	faucet-1	location	**kitchen**, washroom	14	cup-1	has-coffee	**no**, yes
6	person-1	location	**kitchen**, washroom	15	tea-box-1	location	**table**, cabinet
7	person-1	ability	**0.6**, [0–1]	16	tea-box-1	open	**no**, yes
8	kettle-1	has-water	**no**, yes	17	coffee-box-1	location	**table**, cabinet
9	kettle-1	switch	**off**, on	18	coffee-box-1	open	**no**, yes

Table 7. Test cases for problem categories

Sensor config.	Single goal		Multiple goals	
	Correct step	Wrong step	Correct step	Wrong step
Reliability	Case 1–3	Case 6–9	Case 4–5	Case 10–11
Missing sensor	Case 1–3	Case 6–9	Case 4–5	Case 10–11

$PROB$ and PS in a test case can be obtained from the knowledge base shown in Fig. 4. Table 7 presents test cases for each problem category. All test cases are based on the knowledge base in Sect. 5.1. Table 8 selects one case for each category to show.

An iteration computes $PROB$ and PS after each step. Given a step, $PROB$ is correct if the ongoing goal has the highest probability. PS can be partially correct since it involves different levels. To simplify evaluation, we measure PS in a strict way. PS is correct only when its lowest step level is correct, which guarantees a complete correct PS. Note that recognizing the older adult's intent and providing proper hints are both important for an IAA. Thus $PROB$ and PS are considered with equal weights. Assume that the number of steps in a test case is N, the number of iterations with correct $PROB$ is $PROB_C$, and the number of iterations with correct PS is PS_C. The performance is computed using Eq. 6. Thanks to the strict criterion on PS, the real performance of the algorithm is better than the computed performance. Each test case is run 20 times and the average performance is computed. The algorithm removes explanations with probability smaller than **0.001** to avoid too much calculation.

$$Performance = \frac{0.5 \times PROB_C + 0.5 \times PS_C}{N} \times 100\% \tag{6}$$

5.3 Results and Discussion

The average accuracies of all the test cases with changing reliabilities is presented in Table 9, for which we conclude: (1) The performances positively correlate with

Table 8. Example test case for each category (wrong steps have <u>underlines</u>; steps for *wash-hand* in case 5&10 are **boldface**)

Case 1 *wash-hand*	Case 5 *wash-hand, make-coffee*	Case 10 *wash-hand, make-coffee*
turn-on-faucet-1	turn-on-faucet-1	**turn-on-faucet-1**
use-soap	add-water-kettle-1	**use-soap**
rinse-hand	turn-off-faucet-1	**rinse-hand**
turn-off-faucet-1	switch-on-kettle-1	<u>**rinse-hand**</u>
dry-hand	**turn-on-faucet-1**	**turn-off-faucet-1**
Case 8 *wash-hand*	**use-soap**	turn-on-faucet-1
	rinse-hand	**dry-hand**
turn-on-faucet-1	**turn-off-faucet-1**	add-water-kettle-1
use-soap	**dry-hand**	turn-off-faucet-1
<u>use-soap</u>	switch-off-kettle-1	switch-on-kettle-1
<u>turn-off-faucet-1</u>	get-cup-1	switch-off-kettle-1
turn-on-faucet-1	open-coffee-box-1	get-cup-1
use-soap	add-coffee-cup-1	open-coffee-box-1
rinse-hand	close-coffee-box-1	add-water-cup-1
<u>rinse-hand</u>	add-water-cup-1	<u>close-coffee-box-1</u>
dry-hand	drink	open-coffee-box-1
turn-off-faucet-1		add-coffee-cup-1
		close-coffee-box-1
		drink

sensor reliabilities. When sensor reliabilities reduce, the average accuracies deteriorate as well. (2)The easiest problem category **p1**, which targets problems with single goal and correct steps, has the best performance. The average accuracies are very high even when sensor reliabilities are only 0.8. (3) The hardest problem category **p4**, which targets problems with multiple goals and wrong steps, has the worst performance. The accuracies are acceptable only when sensor reliabilities are above 0.95. This result is reasonable since the algorithm has to deal with noisy sensors, multiple goals and wrong steps. (4) The other two categories, **p2** and **p3**, have similar performances, which are acceptable when sensor reliabilities are above 0.9. The results in Table 9 demonstrate the proposed algorithm's capacity to solve the goal recognition and planning problem described in Definition 1. Our algorithm can efficiently handle issues including partial observability, wrong steps, unordered steps, and simultaneous goals.

The influence of sensor reliabilities on $PROB$**.** Figure 5 shows the $PROB$ of case 10. Wrong steps are marked with *. The convergence of $PROB$ is correlated with sensor reliability. The probabilities of ongoing goals outweigh those of non-happening goals after the second or third steps. The probabilities of goal *make-tea* and *make-coffee* align with each other until step *get-cup-1* because they have the same step sequence before *get-cup-1* (refer Fig. 4). A goal's probability

Table 9. Average performances on test cases (sensor reliabilities, in bold)

Case num.	0.99	0.95	0.90	0.80
Case 1	100%	97%	95%	93%
Case 2	100%	99%	99%	97%
Case 3	100%	100%	98%	98%
Case 4	99%	99%	90%	79%
Case 5	100%	99%	93%	86%
Case 6	100%	98%	93%	**44%**
Case 7	100%	99%	98%	96%
Case 8	100%	96%	94%	**59%**
Case 9	100%	92%	83%	**62%**
Case 10	100%	90%	70%	66%
Case 11	100%	94%	79%	69%

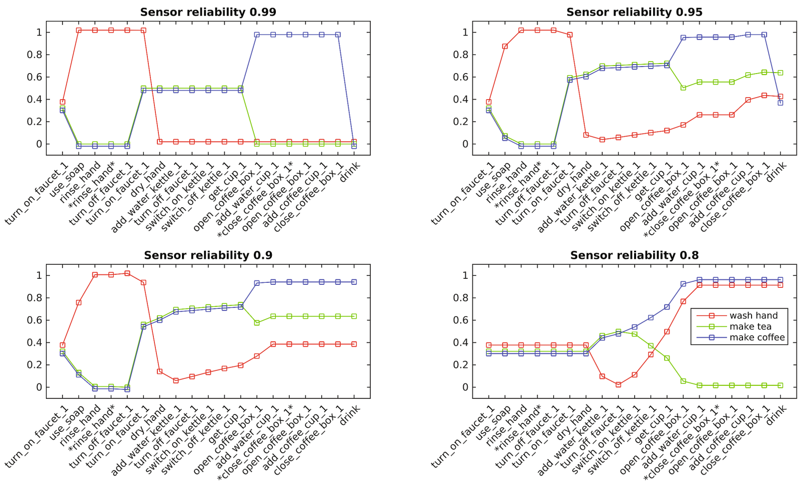

Fig. 5. The *PROB* output for Case 10 (*wash-hand*, *make-coffee*)

drops to 0.0 when it is finished. The horizontally straight lines in plots with sensor reliability 0.9 and 0.8 means the algorithm gets lost and does not update the explanations any more. The algorithm usually get lost when wrong sensor measurements, improper priors, and wrong steps happen together.

A missing sensor is the same as a sensor having reliability 0.5. The algorithm deals with known missing sensors by regarding their reliabilities as 0.5. Experiments with missing sensors suggest how to set up sensors. (1) Sensors related to start steps of goals should not be missing. (2) If a step related to multiple sensors, one of the sensors is missing can be tolerated. (3) A sensor relates to many steps should not be missing. (4) If the older adult repeatedly makes mistakes on some steps, the related sensors should not be missing.

6 Conclusion

This proposed HTN-GRP-PO algorithm is a HTN framework based goal recognition and planning process. The recognition and planning procedures are highly coupled. The HTN framework reduces the search space for goal recognition. The planning procedure generates the desired next steps to proceed towards ongoing goals. It addresses issues including partial observability due to unreliable or missing sensors, concurrent goals, and incorrectly executed steps. The algorithm is tested on cases with different difficulties. An interesting future direction is extending the algorithm to handle step duration and shared steps.

Acknowledgments. We thank the support of AGE-WELL NCE Inc., the Canadian Consortium on Neurodegeneration and Aging, and of the Alzheimers Association (grant number ETAC-14-321494).

References

1. Aggarwal, J.K., Ryoo, M.S.: Human activity analysis: a review. ACM Comput. Surv. (CSUR) **43**(3), 16 (2011)
2. Cook, D.J., Hagras, H., Callaghan, V., Helal, A.: Making our environments intelligent. Pervasive Mob. Comput. **5**(5), 556–557 (2009)
3. Desrichard, O., Köpetz, C.: A threat in the elder: the impact of task-instructions, self-efficacy and performance expectations on memory performance in the elderly. Eur. J. Soc. Psychol. **35**(4), 537–552 (2005)
4. Erol, K., Hendler, J., Nau, D.S.: HTN planning: complexity and expressivity. In: AAAI, vol. 94, pp. 1123–1128 (1994)
5. Geib, C.W.: Problems with intent recognition for elder care. In: Proceedings of the AAAI 2002 Workshop Automation as Caregiver, pp. 13–17 (2002)
6. Geib, C.W., Goldman, R.P.: A probabilistic plan recognition algorithm based on plan tree grammars. Artif. Intell. **173**(11), 1101–1132 (2009)
7. Goldman, R.P., Geib, C.W., Miller, C.A.: A new model of plan recognition. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, pp. 245–254. Morgan Kaufmann Publishers Inc. (1999)
8. Hoey, J., Poupart, P., von Bertoldi, A., Craig, T., Boutilier, C., Mihailidis, A.: Automated handwashing assistance for persons with dementia using video and a partially observable Markov decision process. Comput. Vis. Image Underst. **114**(5), 503–519 (2010)
9. Hwang, A., Hoey, J.: DIY smart home: narrowing the gap between users and technology. In: Proceedings of the Interactive Machine Learning Workshop, 2013 International Conference on Intelligent User Interfaces (2013)
10. Kautz, H.A., Allen, J.F.: Generalized plan recognition. In: AAAI, vol. 86, p. 5 (1986)
11. Nau, D.S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D., Yaman, F.: SHOP2: an HTN planning system. J. Artif. Intell. Res. (JAIR) **20**, 379–404 (2003)

12. Okeyo, G., Chen, L., Wang, H.: Combining ontological and temporal formalisms for composite activity modelling and recognition in smart homes. Future Gener. Comput. Syst. **39**, 29–43 (2014)
13. Okeyo, G., Chen, L., Wang, H., Sterritt, R.: A hybrid ontological and temporal approach for composite activity modelling. In: 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 1763–1770. IEEE (2012)
14. Okeyo, G., Chen, L., Wang, H., Sterritt, R.: A knowledge-driven approach to composite activity recognition in smart environments. In: Bravo, J., López-de-Ipiña, D., Moya, F. (eds.) UCAmI 2012. LNCS, vol. 7656, pp. 322–329. Springer, Heidelberg (2012). doi:10.1007/978-3-642-35377-2_44
15. Sacerdoti, E.D.: A structure for plans and behavior. Technical report, DTIC Document (1975)