Towards Metareasoning for Human-Robot Interaction

Xiaoping Chen[1], Zhiqiang Sui[1], and Jianmin Ji[2]

[1] University of Science and Technology of China
Hefei, 230026, P.R. China
[2] The Hong Kong University of Science and Technology
Hong Kong
xpchen@ustc.edu.cn, {zqsui, jizheng}@mail.ustc.edu.cn

Abstract. This paper proposes a model of metareasoning for Human-Robot Interaction (HRI). Robots' basic abilities for HRI—planning, learning and dialogue—are characterized as three loops in the model, with each spanning ground, object and meta-level. The model provides a conceptualization of HRI and a framework for incremental development of large HRI systems such as service robots by building meta-level functions on top of existing ground/object level components. A case-study focusing on meta-level control shows that the approach is effective and efficient for some application domains. In particular, meta-level control suggests a new opportunity to speed up planning while preserving completeness without any change to object level planners. The experiments also show that, for some basic HRI tasks, there are simple meta-level strategies with performances better than the common strategy in previous work.

Keywords: HRI, metareasoning, modeling, meta-level scheduling

1 Introduction

This paper concerns service robots that work together with humans (hereafter, service robots for short). It follows that human-robot interaction (HRI) is essential to these robots [15, 9]. They should be able to understand users' requests and provide services for users accordingly by taking actions. The symbiosis of service robots and humans suggests new opportunities and challenges to Robotics and AI research. It has been observed that humans' and robots' abilities are complementary in many aspects and thus they should help each other in order to fulfill better services for humans [14, 6, 8]. One means to this end, which has drawn increasing interest recently, is to make robots capable of asking humans for help through human-robot dialogue [5, 13, 11].

Generally, service robots should possess three characteristics: autonomy, adaptability and sociality. Accordingly, these robots must be equipped with three basic abilities: planning, learning and dialogue (not necessarily through speech). For instance, a robot is not autonomous if it cannot by itself generate (and execute) a plan of actions for a task. A further and crucial observation is that the coordination of all the three abilities is needed for robots with these characteristics.

For example, to show its adaptability to a new task, a robot may need to plan its dialogue with humans, acquire knowledge/information through the dialogue, and achieve the task with the acquired knowledge/information. This process has to be realized by coordinating all the three abilities.

There have been a lot of research achievements regarding these abilities themselves. But their coordination has been less studied. This paper proposes a model as a conceptualization for analyzing the research issues and a framework of developing robots with the characteristics. The model consists of three levels —ground, object, and meta-level—as proposed in Metareasoning literature [7]. Each component of the robots is cast as a function at some level and each of the basic abilities, planning, learning and dialogue, is cast as a "loop" spanning the three levels. The main idea is to model the coordination of the three basic abilities as the interaction among the three loops at meta-level. This way, we introduce metareasoning into HRI and put forth an alternative approach against the HRI challenges. The planning loop extends the traditional perception-decision making-action loop with meta-level control and monitoring. The other two loops are similar extensions. Therefore, our model allows one to re-use various ground/object level components well developed in previous work, while strengthen and coordinate the basic abilities with new functions introduced at meta-level.

2 Issues of Metareasoning for HRI

2.1 Metareasoning

As proposed in [7] and well-accepted, metareasoning is captured by a three-level model as shown in Figure 1. The ground and object level in the model constitute the perception-decision making-action loop well known in AI. In this loop, an intelligent agent perceives some stimuli from the environment and behaves to achieve its goals according to the decisions it makes through reasoning. The result of these actions at the ground level is perceived and fed back to the object level, and the cycle continues. Metareasoning is defined as the process of reasoning about this reasoning cycle and thus constitutes the top level of the enlarged loop. In other words, reasoning controls actions at the ground level, whereas metareasoning controls the reasoning at the object level.

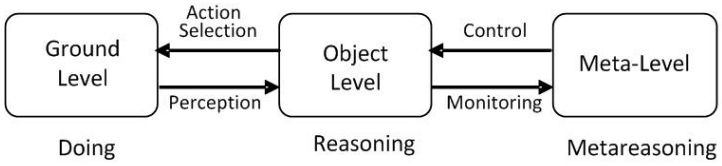

Fig. 1. General Metareasoning Model

Generally, metareasoning consists of the meta-level control and the introspective monitoring of reasoning. For a classical metareasoner, the goal of meta-level control is to improve the quality of its decisions by making some computational effort to decide what and how much reasoning to do as opposed to what actions to do. For example with an anytime algorithm that incrementally refines plans, an agent must choose between executing the current plan or further improving it in order to get a better one. Given that the passage of time itself has a cost, the metareasoner must judge which choice will lead to greater expected benefit and make the choice accordingly. On the other hand, the goal of introspective monitoring is to gather sufficient information with which to make effective meta-level control decisions. For instance, [4] maintains statistical profiles of past metareasoning choices and the associated performance, using them to mediate the subsequent control and dynamic composition of reasoning processes.

2.2 The Model of Metareasoning for HRI

The model of metareasoning for HRI (MM-HRI) proposed here is an extension of the General Metareasoning Model (GMM). The ground-level functions of MM-HRI include three types of actions: perception (like vision), physical actions (such as navigation and manipulation) and exchanging messages with users. Note that there is at least one human user in an HRI setting and thus human-robot communication is a necessary component of the robot. But the message function in the ground-level does not cover the full communication function. For example, ground level is not responsible for deciding what and when to say in human-robot dialogue.

The object level of MM-HRI includes three functions: planning, learning, and dialogue. A service robot that performs complex tasks must be able to plan its behaviors in advance by a planner. The generated plans, courses of ground level primitive actions (or, *atomic actions* for short), will be executed at the object level. In accordance with this object level function, there is a planning loop spanning three levels of MM-HRI, as shown in Figure 2 [7]. Both the plan generation and execution are controlled by metareasoning in order to meet following requirements: 1) computational efficiency, so as to generate (suboptimal) plans timely; 2) dynamic environment, which may cause replanning due to changes of the environment and/or users' intention; 3) incomplete knowledge, which brings about various object-level operations, eg, postponing the decision of what to do until the robot gains the information required for planning; 4) uncertainties, which may also result in replanning when the robot perceives that its actions did not reach the expected effects. Although some of these problems have been tackled at object level (eg, [3]), we argue that our three-level model provides a better framework for development of large-scale intelligent systems, particularly intelligent service robots described in Section 1.

The learner is responsible for acquiring knowledge from outside, which can be used by the robot later for problem-solving. Generally, the required knowledge includes: 1) information about the state of the environment and the robot that is necessary for planning and fulfilling the tasks at hand; 2) user models, e.g.,

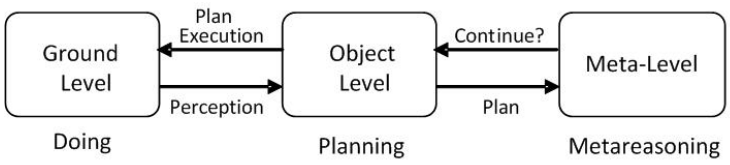

Fig. 2. Planning loop in MM-HRI

a model of the availability and accuracy of human observation providers with which the robot can get help more efficiently [14]; 3) domain knowledge required for the robot's missions but missing due to the unpredictability of real-world applications. Note that this kind of knowledge is usually expressed in natural languages. So we employ the term "material" to cover all the forms received at ground level. At object level, the robot extracts knowledge (learning results) or needs for further learning from the material. The latter is processed by metareasoner to produce new learning goals for the next circle of learning process [7]. At object level, the learning goals create some form of "learning evocator", such as questions to the user, and the cycle continues. The learning loop is shown in Figure 3.

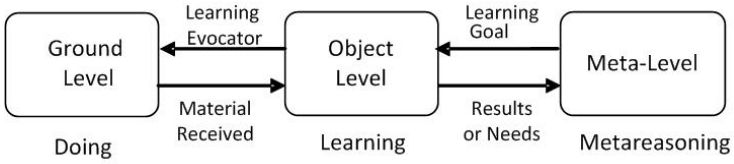

Fig. 3. Learning loop in MM-HRI

Generally, a service robot has a dialogue manager that is responsible for running the dialogue process of receiving users' service requests. This component is abstracted as a function in the object level of our model. In this paper, however, by "dialogue" we mean any information exchange between humans and robots, not limited to those through speech. The dialogue loop in our model is expected to produce human-robot dialogues not only for receiving service requests, but also for various kinds of "help" from users and other external sources. Whatever received from the human-robot communication at ground level is transformed into some internal representation called "expression" in figure 4. Then a component at object level tries to understand it and produce some "content" from it. The real meaning of the content depends on not only the dialogue itself but also what are going on in other MM-HRI loops. Anyway, the metareasoner will generate some "theme" for the next round of dialogue, with which some messages are generated at object level and sent to ground level.

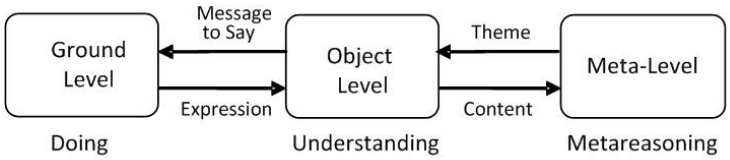

Fig. 4. Dialogue loop in MM-HRI

These three loops must coordinate and interleave their running in order to fulfill the overall performance of the robot. The interactions among the MM-HRI loops mainly occur at meta-level, which is the basic role of the robot's metareasoner. Figure 5 shows the sketch of the interaction and the entire MM-HRI. Note that Figure 2, 3, and 4 only show the internal nodes of each loop, respectively. Actually, either planning or learning loop includes both users and the environment as its external nodes, while dialogue loop includes users as the only external node of it.

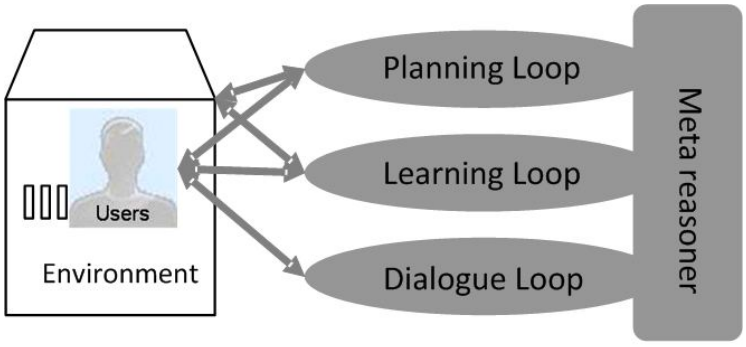

Fig. 5. Sketch of MM-HRI

2.3 Meta-level Interactions

Here we briefly describe meta-level interactions among MM-HRI loops, without specifying the interaction mechanisms formally.

(1) *Planning via Learning.* Lacking knowledge or information causes planning intractable or even infeasible. The difficulty can be reduced or even overcome when robots gain more information and/or knowledge. For example, a robot can observe its environment with its sensors or acquire relevant information/knowledge through dialogues with its users. Therefore, what is really needed is some mechanisms at meta-level with which the robot can coordinate

its planning and learning loop. For instance, the metareasoner can identify missing information that is necessary for the robot's planner at object level and try to get the information before planning. When the robot is required to response to a task-stream, i.e., a stream of tasks, it can plan for some of the tasks with complete information and execute the plan first. Meanwhile during the execution, it can observe the environment to collect information needed for other tasks. In this process, interleaving of planning and learning loop is launched and controlled by the metareasoner.

(2) *Learning via Dialogue.* There are various types of learning. In this paper, we focus on knowledge acquisition through human-robot dialogue. [13] demonstrates this kind of learning where the robot asks humans for help. In one of our previous case-studies, robot KeJia successfully solved a problem, which it had failed before, with knowledge it acquired through human-robot dialogue in a limited segment of English [5]. In these cases, the Understanding component also plays an essential role in learning. The basic topic in this type of learning is about how to ask users' help—what, when and where to ask [14], which requires the coordination of all the three loops. We will present some meta-level strategies of this type in Section 3.

(3) *Dialogue via Planning.* We have described some interactions between dialogue and planning loop above. Besides, there are other ways of interaction between them. Topics discussed a lot in the literature include dialogue generation by planning, understanding by virtue of planning, etc. These are also covered by our model. Moreover, our model covers some topics that are not so well studied yet. A key issue of human-robot collaboration is about how to collaborate on a common task jointly by humans and robots. The human and robot partners should form joint intention and then carry out joint actions for the task [2]. Therefore, a robot partner must coordinate its behavior involving dialogue and planning loop in the collaboration.

3 Case-Study: Meta-level Scheduling

The domain we chose, called eGPSR, is an enhanced variant of a test, called General-Purpose Service Robot, of RoboCup@Home [12]. In this test, a robot is given a set of tasks chosen randomly just before the test begins. These tasks involve a set of behaviors (e.g., following a person, finding a person, grasping and delivering objects), a set of portable objects (e.g., cans, cups and bottles) and a set of locations in a house. Moreover, concepts (e.g., "drink"), which represent classes of objects, are allowed to describe tasks. For simplicity, we identify any concept c with the class of objects it represents by abuse of notation. Any object in class c is called a c-object. When a concept c appears in the description of a task, it refers to any c-object. Therefore, tasks may be partially specified. For instance, the task *give me a drink* is understood as "give me anything to drink", without specifying which particular drink (an individual object) or its position in the environment. Moreover, tasks from one task set may be related to one another in a sense we will explain below.

We implemented a real robot KeJia [5, 6]. In the case-study, three meta-level scheduling strategies were realized in one and the same high-layer subsystem of KeJia, called sub-KeJia. It only contains object-level and meta-level functions needed for the experiments, such as an object-level planner and a missing-information detector that detects what information is missing from the description of a task. The planner can generate a *plan* (a course of atomic actions) that achieves a task, only if the task is fully specified and there exists such a plan. For simplicity, the experiments were conducted on a software test-bed that simulates the domain, eGPSR, and KeJia's ground-level functions needed.

The strategies are evaluated with following criteria, which in turn reflect some prevalent requirements for HRI:

1. Cost of execution. A plan is of smaller cost of execution than another one for the same set of tasks, if the former contains the less number of atomic actions.
2. Efficiency of planning. Planning is time-consuming and service robots are required to provide real-time responses to its users, which depends on the efficiency of planning to a large extent.
3. Performance of asking. Previous experiments indicate that users would not like to be asked too many questions [14]. Asking more also makes a robot less autonomous. So *ceteris paribus*, asking less is better.
4. Completeness of resource-bounded deliberation. A resource-bounded metareasoner is *complete*, if it can solve all problems that can be solved under no resource limitation.

The meta-level scheduling strategies tested in this case-study are described informally below:

Eager-to-Ask strategy: For all input tasks, detect what information is missing; generate questions about the missing information; ask these questions of the users and complete the description of all the partially specified tasks with the information from the answers. Fulfill the completed tasks one by one in the input order by generating a plan for each completed task and executing the plan.

Lazy-to-Ask strategy: Fulfill the input tasks one by one in the input order. For each task, detect what information is missing; generate questions about the missing information if any; ask these questions of the users and complete the description of current task with the information from the answers; generate a plan for the current task; execute the plan.

Relevance-based-Ask strategy: Divide the set of input tasks into subsets, so that they are not related one another. Fulfill the subsets one by one using Lazy-to-Ask strategy, with each subset of tasks being taken as a (complex) task.

Each of these strategies contains operations at three levels. For instance, "generate a plan" is at object level and "execute a plan" ground level. Formal specification of such strategies demands a formal language with level tags of operators. Meanwhile, each strategy involves three loops. Asking is in dialogue loop, and extracting information to complete original tasks belongs to learning loop. So a meta-level scheduling strategy coordinates the three loops to achieve tasks. We illustrate how sub-KeJia works under Lazy-To-Ask Strategy with a

small task set $T = \{$ *give Allen some drink; give Bill a bottle of coke*$\}$. Assume there are only one bottle of coke and one cup of coffee in the environment and the robot knows the facts but does not know the location of each drink in the beginning. Suppose after the first task is analyzed by the missing-information detector, KeJia allocates the cup of coffee to the concept "drink" (perhaps the robot should confirm this with question "how about coffee?" in real services. But we omit this here). Then it generates a question about the location of the coffee. After KeJia gets the missing information from user, it generates a plan of four atomic actions for the completed task with its object-level planner. Then KeJia executes the plan in the ground-level: *move to the location of coffee*, *grasp coffee*, *move to Allen and give the cup of coffee to Allen*. Assume the robot get the missing information of the second task, the location of coke, through its sensors during it executes above actions. Thus the knowledge of the second task is completed and the object-level planner generates a plan of it.

The main deference between the first two strategies is the time point when the robot asks questions of users. With Eager-to-Ask strategy, the robot asks questions before it plans for any task. It starts to plan only after it obtains all missing information in original tasks. This is the common strategy used in most of previous work [1]. One problem with it is that it fails to make use of concurrency of a robot's hardware and software components, as shown in above example. Lazy-to-Ask strategy improves in this aspect and thus got better performance of asking in the experiments (Section 4).

The first two strategies do little metareasoning except scheduling over lower-level operations. In particular, they do not consider the relevance among tasks and just try to fulfill them one by one in the input order. Consequently, the completeness is violated—some tasks cannot be achieved just due to the over-simplified meta-level strategies. Relevance-based-Ask strategy solves this problem with more metareasoning, mainly on the relevance among tasks.

The notion of "relevance" is technically complicated and defined as follows. Without loss of generality, we assume that planning problems are specified in PDDL [10] and each of them contains an initial state, a goal description (tasks), and a domain description which includes specifications of effects and preconditions of actions and other background knowledge.

Definition 1. *Given a domain description, an initial state, and two tasks t_1 and t_2. We say that t_2 is related to t_1 if there exists a plan P_1 that achieves t_1 and there does not exist a plan P_2 such that P_1 appended with P_2 could achieve both t_1 and t_2.*

If a task t_2 is related to another one t_1, then the tasks in the set $\{t_1, t_2\}$ should not be planned and executed separately in the order $\langle t_1, t_2 \rangle$. As the robot may choose a plan P_1 to achieve t_1, but after the execution of P_1, t_2 can no longer be achieved without ruining t_1. For example, suppose a robot as a bar tender is given two tasks,

t_1: give Jim some drink; t_2: give Bob a can of Coke.

Assume there is only one can of Coke in the environment. Then t_2 is related to t_1, as after giving Jim the Coke, t_2 can no longer be achieved. On the other hand, if t_2 is not related to t_1, then a robot can solve $\{t_1, t_2\}$ separately in the order $\langle t_1, t_2 \rangle$. The notion of relevance between two tasks can be generalized to between two sets of tasks.

Definition 2. *Given a domain description, an initial state, and two sets of tasks T_1 and T_2. We say that T_2 is related to T_1 if there exists a plan P_1 that achieves all tasks in T_1 and there does not exist a plan P_2 such that P_1 appended with P_2 achieve all tasks in both T_1 and T_2.*

Proposition 1. *Given a domain description, an initial state, and a set of tasks $T = \{t_1, \ldots, t_n\}$. If for each $1 < i \leq n$, $\{t_i\}$ is not related to $\{t_1, \ldots, t_{i-1}\}$, then T can be achieved if and only if each task in the sequence $\langle t_1, \ldots, t_n \rangle$ can be achieved one after another.*

Proposition 2. *Given a domain description, an initial state, and a set of tasks $T = T_1 \cup \cdots \cup T_n$. If for each $1 < i \leq n$, T_i is not related to $T_1 \cup \cdots \cup T_{i-1}$, then T can be achieved if and only if each set of tasks in the sequence $\langle T_1, \ldots, T_n \rangle$ can be achieved one after another.*

Deciding whether a set of tasks is related to another is generally harder than deciding whether a set of tasks can be achieved by a plan. However, the problem can be greatly simplified in many domains, e.g., eGPSR. The relevance between tasks in this domain only results from "the limitation of resources".

Let $c \subset c'$ denote that c is a proper subset of c'. The set of all concepts appeared in eGPSR is denoted by C and the set of all portable objects in C by O. Then C under $\subset$ forms a hierarchy $(C, \subset)$. We assume that for any c and $c' \subset$ C, if c and c' do not disjoint, then it holds that $c \subset c'$ or $c' \subset c$. We also assume that the size of every concept $c \in C$, denoted by $|c|$, is fixed and known by the robot.

Given a task set T, we use $C(T)$ to denote the set of concepts appeared in T, $n(c, T)$ the number of occurrences of c in T (we do not consider any other requirements on resource in this paper). $C(T)$ can be created from T in linear time. In order to achieve all tasks in T, the planner must allocate to each concept $c \in C(T)$ a c-object; in other words, each occurrence of c in T demands an "occupation" of a c-object. A least efficient way of resource allocation for T_1 is to "run on" objects required by T_2. That is, when there are concepts $c \in C(T_2)$ and $c' \in C(T_1)$ such that $c \subset c'$, the robot allocates only c-objects to every c' in T_1, such that the c-objects may be used up unnecessarily for tasks in T_1, causing T_2 unsolvable. This is possible since a robot cannot consider the requirements by T_2 when it plans for T_1 before for T_2.

The basic idea of our heuristic algorithm (**Algorithm 1**) is to *test* if this least efficient way would cause T_2 unsolvable. In the algorithm, $o(c)$ records the number of c-objects occupied so far in the estimate process. The algorithm terminates in time $O(n^2)$, where n is the length of $C(T_1 \cup T_2)$.

Algorithm 1

For each $c \in C(T_1 \cup T_2)$, $o(c) := 0$;
For each $c \in C(T_1)$ //occupation by T_1//
    $o(c) := o(c) + n(c, T_1)$;
    For each $c' \in C(T_1)$
        If $c \subset c'$ **then** $o(c') := o(c') + n(c, T_1)$;
    If $o(c) > |c|$ **then** return *false*; //T_1 is unsolvable//
For each $c \in C(T_2)$ //occupation by $T_1 \cup T_2$//
    $o(c) := o(c) + n(c, T_2)$;
    For each $c' \in C(T_1 \cup T_2)$
        If $c \subset c'$ **then** $o(c') := o(c') + n(c, T_2)$;
For each $c \in C(T_2)$ //run on objects by T_1//
    For each $c' \in C(T_1)$
        If $c \subset c'$ and $o(c) + n(c', T_1) > |c|$ **then** return *true*;
Return *false*.

Formally, an allocation of objects for a task set T is an assignment of objects to concepts in T, i.e., a mapping δ: $C(T) \to O$. Let $|c|_\delta$ denote the number of c-objects occupied by δ, for every $c \in C$. An assignment δ is called *feasible*, if $|c|_\delta \leq |c|$ for all $c \in C$. A domain is called *resource-determined* if for any task set T, there is a feasible assignment for T implies there is a plan for T. For example, eGPSR is a resource-determined domain. We have the following proposition.

Proposition 3. *Given any task sets T_1 and T_2 in a resource-determined domain. T_2 is not related to T_1 if Algorithm 1 returns 'false'.*

4 Experimental Results

The size of the environment is set as $10m \times 10m$ and there are about 20 objects including portable ones and furniture. Since a real robot's perception ability is limited, we simulate this feature approximately with "observation radius" (OR) in our test-bed. We assume the robot can perceive the information of all objects within the OR and no information of any object outside. The greater the OR is, the more (missing) information may be perceived.

We conducted two tests with OR taken as $1m$ and $2m$, respectively. Each test consists of three groups and each group 20 task sets. All task sets in one group contain the same number of tasks: there are 6 (8, 10) tasks in every task set in group 1 (2, 3, respectively). The tasks in any task set are chosen randomly under an additional restriction described later.

Table 1 shows the experimental results of test 1, where the observation radius is set as $1m$. Relevance-based-Ask strategy achieved all the tasks in each task set, while the other two do not since they cannot deal with relevance between tasks in the same task set. In particular, when two or more tasks in the same task set are related, both strategies may fail to generate plans for and thus fail to achieve some of these tasks, although they are not unsolvable in themselves.

Table 1. The results of test 1

OR=1m	Planning Time(s)	#Ask	#Atomic Actions	#Task Achieved
Group 1: 6 tasks in each task set				
Eager	1.01	8.70	22.55	5.55
Lazy	0.88	5.65	21.40	5.15
Relevance	1.21	5.70	23.50	6.00
Group 2: 8 tasks in each task set				
Eager	1.16	11.00	29.95	7.40
Lazy	1.01	5.90	29.15	7.10
Relevance	1.38	6.35	31.15	8.00
Group 3: 10 tasks in each task set				
Eager	1.68	13.45	37.40	9.25
Lazy	1.40	6.90	37.05	9.10
Relevance	1.82	7.45	38.55	10.00

The efficiency of planning among three strategies is of little difference in this test. The first two strategies produced on average 2 fewer atomic actions than the third strategy, because they achieved fewer tasks in some task sets. An obvious deference is that Eager-to-ask strategy always asks notably more number of questions than the other strategies, just because it obtains missing information only through asking questions, while the other two obtain some through the robot's sensors. Relevance-based-ask strategy asks a little more than lazy-to-ask strategy, as a tiny price for its completeness.

As the observation radius increases to $2m$, we can see from Table 2 that the ask times decreases about 40% with the second and the third strategy. The reason is that the robot can perceive more missing information without any additional effort or cost. This indicates that meta-level control affects the performance of asking remarkably when robots possess powerful perception.

Table 2. The results of test 2

OR=2m	Planning Time(s)	#Ask	#Atomic Actions	#Task Achieved
Group 1: 6 tasks in each task set				
Eager	0.96	8.70	22.85	5.65
Lazy	0.88	2.80	22.30	5.45
Relevance	1.38	3.90	23.75	6.00
Group 2: 8 tasks in each task set				
Eager	1.16	11.00	30.10	7.45
Lazy	1.06	3.05	29.30	7.15
Relevance	1.55	3.90	31.25	8.00
Group 3: 10 tasks in each task set				
Eager	1.58	13.45	37.25	9.20
Lazy	1.41	4.05	37.15	9.15
Relevance	1.94	4.90	38.65	10.00

In above tests, a task set may be divided into several sub-sets under the relevance-based strategy, where all tasks in each of these sub-sets are planned together. The additional restriction for selection of tasks is that any subset contains no more than two (related) tasks. It is well-known that the computation time of (object-level) planning will increase exponentially in the size of tasks (here the subsets of tasks) if all the tasks are planned together, no matter whether or not they are related. We did an additional test on planning for larger task sets with the same sub-KeJia. It showed that planning for a 6-task set frequently needs more than 1 hour. This provides strong evidence that it is necessary to reduce the size of subsets of tasks using some meta-level control techniques, such as relevance-based scheduling we used in the case-study.

5 Discussion and Conclusion

We draw following observations from this work, especially the case-study.

(1) Metareasoning provides an effective approach to HRI, particularly, the coordination of basic abilities of a service robot. This coordination is of most importance to HRI and reflected in our model as the coordination of planning, learning, and dialogue loop. This is the reason why we focus on the coordination in this paper while previous work on metareasoning focuses on internal process of individual loops.

(2) The model MM-HRI supplies a framework with which one can advance the performances of HRI by building meta-level functions on top of existing, well-developed ground/object level components. We implemented meta-level scheduling directly on previously developed low-level components and got new functions and better performances. Particularly, meta-level control suggests a new opportunity to speed up planning while preserving completeness with reasonable price for some application domains, without any change to the object level planner.

(3) The case-study shows that meta-level control affects the performances of HRI remarkably in at least three aspects: planning time, performance of asking, and completeness of resource-bounded deliberation. Compared to the common strategy in previous work, the two strategies we proposed in this paper got higher evaluation in the experiments. One important reason is that meta-level control can make better use of concurrency of a robot's hardware/software components.

One can draw more observations, e.g., asking is more desirable to acquire knowledge or information that cannot be perceived easily through robots' sensors. These observations suggest a lot of future work along this line of research.

Acknowledgments. This work is supported by the National Hi-Tech Project of China under grant 2008AA01Z150 and the Natural Science Foundation of China under grant 60745002 and 61175057. We thank Guoqiang Jin, Feng Wang, Jiongkun Xie, Hao Sun, Min Cheng, Xiang Ke and Kai Chen for their contributions to KeJia Project. The authors are also grateful to Shlomo Zilberstein for his help to this work.

References

1. Asoh, H., Motomura, Y., Asano, F., Hara, I., Hayamizu, S., Itou, K., Kurita, T., Matsui, T., Vlassis, N., Bunschoten, R., et al.: Jijo-2: An office robot that communicates and learns. IEEE Intelligent Systems 16(5), 46–55 (2001)
2. Bauer, A., Wollherr, D., Buss, M.: Human-robot collaboration: A survey. International Journal of Humanoid Robotics 5(1), 47–66 (2008)
3. Brenner, M., Nebel, B.: Continual planning and acting in dynamic multiagent environments. Autonomous Agents and Multi-Agent Systems 19(3), 297–331 (2009)
4. Carlin, A., Zilberstein, S.: Decentralized monitoring of distributed anytime algorithms. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-11). pp. 157–164 (2011)
5. Chen, X., Ji, J., Jiang, J., Jin, G., Wang, F., Xie, J.: Developing high-level cognitive functions for service robots. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10). pp. 989–996 (2010)
6. Chen, X., Jiang, J., Ji, J., Jin, G., Wang, F.: Integrating nlp with reasoning about actions for autonomous agents communicating with humans. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (IAT-09). pp. 137–140 (2009)
7. Cox, M., Raja, A.: Metareasoning: A manifesto. Technical Memo 2028, BBN (2008)
8. Fong, T., Thorpe, C., Baur, C.: Robot, asker of questions. Robotics and Autonomous systems 42(3-4), 235–243 (2003)
9. Fong, T.W., Thorpe, C., Baur, C.: Collaboration, dialogue, and human-robot interaction. In: Proceedings of the 10th International Symposium of Robotics Research. pp. 255–266. Springer-Verlag (November 2001)
10. Ghallab, M., Howe, A., Christianson, D., McDermott, D., Ram, A., Veloso, M., Weld, D., Wilkins, D.: Pddl—the planning domain definition language. AIPS98 planning committee 78(4), 1–27 (1998)
11. Kaupp, T., Makarenko, A., Durrant-Whyte, H.: Human-robot communication for collaborative decision making–a probabilistic approach. Robotics and Autonomous Systems 58(5), 444–456 (2010)
12. Nardi, D., Dessimoz, J., Dominey, P., Iocchi, L., Rybski, P., Savage, J., Schiffer, S., Wisspeintner, T., van der Zant, T., Yazdani, A.: Robocup@home: Rules and regulations (2008)
13. Rosenthal, S., Biswas, J., Veloso, M.: An effective personal mobile robot agent through symbiotic human-robot interaction. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10). pp. 915–922 (2010)
14. Rosenthal, S., Veloso, M.M., Dey, A.K.: Learning accuracy and availability of humans who help mobile robots. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11). pp. 1501–1506 (2011)
15. Thrun, S.: Toward a framework for human-robot interaction. Human–Computer Interaction 19(1-2), 9–24 (2004)