Online Adjusting Task Models for Ubiquitous Robotic Systems

Wenshan Wang, Qixin Cao, Qiang Qiu, Gilbert Cheruiyot

Research Institute of Robotics, Shanghai Jiao Tong University, Shanghai, China
Email: {amigo, qxcao}@sjtu.edu.cn, qiu6401@163.com,
gcherrie@gmail.com

Abstract. Task modeling and task planning are very important in robotic systems especially for large-scale nondeterministic problems. Two widely studied models (the classical planning model and the Markov Decision Process (MDP) model) are inapplicable to such problems due to either inherently assumed determinism or dimensional explosion. An amalgamation of these two results in a new model which is proposed in this study under the name "Reduced Markov Decision Process" (RMDP) model. This new model simplifies the conventional MDP model by reducing the branching factor of state transitions. Further, based on the RMDP model, a modified Dynamic Programming (DP) algorithm is proposed. The RMDP model also facilitates online learning that adapts the model to environmental changes. A "forgetting" model is employed for this online adjustment. In the experiment, a ubiquitous robotic system is implemented for robotic bar-tending task. The results demonstrate that the model conveniently facilitates online-updating to better match the real environment.

Keywords: task modeling, task planning, online learning, ubiquitous robotics

1 Introduction

In recent years ubiquitous robotics has gained the attention of many researchers [1]. This technology is characterized by robotic devices dispersed in the environment such that they become a ubiquitous part of our daily lives. These distributed sensing and acting components provide services by communicating and collaborating over a network.

Such a system should have the capacity to translate user-commands to low-level actions, which can be directly executed by the distributed components. This calls for a task planning method that is capable of planning at the symbolic level for different tasks in a complex and dynamic environment.

Most of the earlier ubiquitous robotic systems are task-specific. For example: urban cleaning [2]; self-localization and mapping [3]; or public guidance [4]; etc. In these task-specific situations, the task planner is left out, because the designers tend to prefer simpler methods such as finite state machine for the action selection. However, when faced with a dynamic environment and the need to provide general-purpose

services, it may be impossible to pre-consider all the situations, in which case a good task planner becomes critical.

The most commonly employed techniques are based on Artificial Intelligence (AI). Young-Guk Ha et al. used SHOP2 planner to decompose services based on semantic knowledge [5]. Robert Lundh et al. implemented a configuration approach for their network robot system also based on SHOP planner [6]. These classical planners such as SHOP2 [7] and Fast-downward [8] are efficient. However, they cannot deal with dynamic situations with uncertainties as is the case in the real world. In response to this, some researchers have used probabilistic models in task planning problems. For example, Marco Barbosa et al. used Partially Observable Markov Decision Processes (POMDP) to model the tasks with uncertainty [9]. Marcello Cirillo et al. implemented RTLplan for probabilistic domains [10]. However, planning methods based on probabilistic models suffer dimension explosion, which limits the size of the state space to impractical applications. Furthermore, the real environment is not only nondeterministic but also dynamic. The stationary model would soon be inaccurate for the nonstationary environment, and need a revision.

In view of the foregoing, we propose in this paper a Reduced MDP (RMDP) model, which is arrived at by introducing further assumptions to the MDP model. It is discussed in detail how this model can accelerate the planning process without compromising its expressivity on nondeterministic problems for real environments. Furthermore, using the RMDP model, we propose an online learning algorithm, which allows the planning model to adapt to the system changes. Consequently, the planning results could be improved as well. A ubiquitous robotic system for bar-tending scenario is implemented as a demonstration platform for these algorithms.

This paper starts with a brief introduction to our ubiquitous robotic system and its component-based structure in section 2. Subsequently, in section 3, the RMDP model is proposed. The online learning capability that makes this method adaptive to dynamic situations is presented in section 4 after which the results of the experiments are discussed in section 5. Finally, conclusions are drawn in section 6.

2 System overview

In contrast to the monolithic robot, the ubiquitous robotic system offers the advantage of distributed robotic sensing and acting devices in the environment to complete different tasks through collaboration [11]. This paper focuses on how to cooperate different robotic devices to complete complex tasks.

The symbolic task planning module consists of a device manager, a task planner and an online learning module as Fig.1 shows. The device manager is responsible for transferring high-level tasks and monitoring low-level status. The planner turns users' abstract commands into sub-task sequences, which can be directly carried out by corresponding robotic components. The online learning module adjusts the domain knowledge used by the task planner in real time, so that the planner could adapt to the environmental changes.

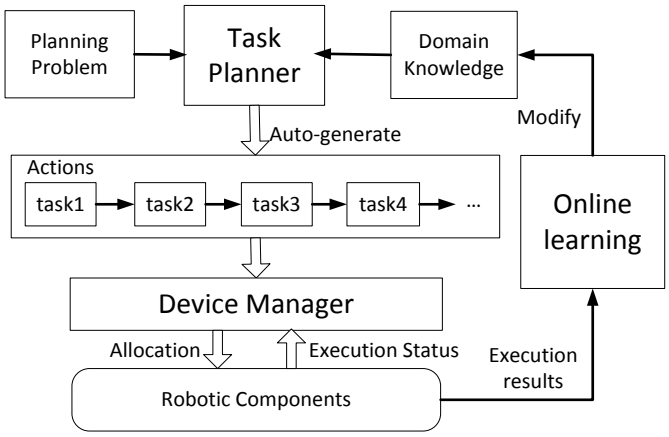

Fig. 1. System diagram for task planner and online learning

The robotic devices in the system are developed based on middleware technology [12-13]. This component-based structure is the foundation of the system. It facilitates the execution of the planning results to fulfill realistic tasks in physical environment. Robotic devices are highly heterogeneous with respect to platforms such as operating system, programming language and communication media. Middleware is thus employed to generalize the components into a uniform abstraction which enables dynamic communication and coordination between any two of the modules. Fig. 2 explains how the middleware "transparentizes" the hardware and software platforms, and offers standardized access to the robotic devices.

We implement our system based on (Robotic Technology) RT middleware [14-15]. In RT component, the ports are categorized into data ports and service ports. The data port is responsible for the continuous exchange of data. Each component can have any number of data in-ports and out-ports. A data out-port sends the data to a corresponding in-port which receives the data. The service port provides the command based communication. The component with a service port, offering a set of services, listens for requests for those services via a connector.

We further define three kinds of service ports, namely FuncGet, FuncSet and ExeStatusGet as Fig. 2 shows. The service port is responsible for the interaction with the task planner and the learning module. FuncGet port reports to the upper layer about the components' state. For example, the environmental camera provides the index of current robots within the field of vision; the mobile platform feeds back its coordinates, etc. FuncSet port provides the functionality invoking, such as setting the target position for the mobile table, setting the localizing target for the environmental camera, etc. ExeStatusGet port returns the execution status, for example whether or not the mobile table has reached its destination, or whether the environmental camera is locking onto its target.

Each component may have any number of data ports for continuous data exchange between components. For instance, the localization information is transferred from the data out-port of environmental camera to the data in-port of the path planning

component. Once two data ports are connected, those two components are able to perform real-time communication to accomplish the task collaboratively.

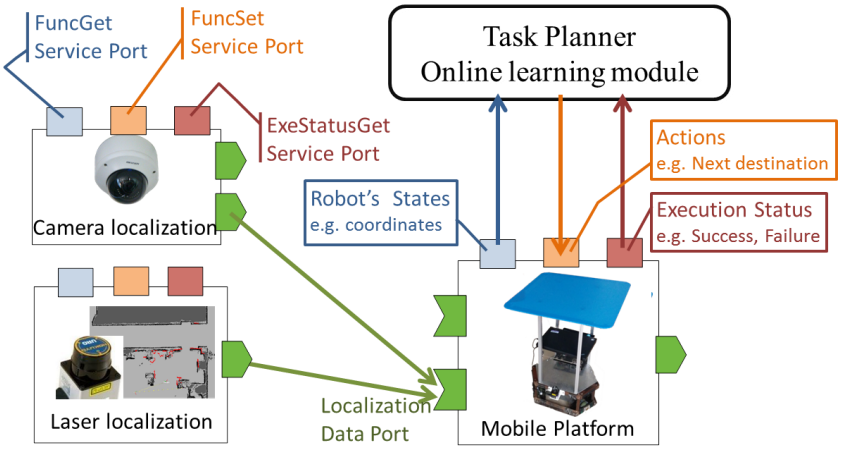

Fig. 2. The distributed components are in the uniformed structure using middleware technology

3 Task modeling and task planning

As mentioned above, the different robotic functionalities are abstracted, and ready to be called through the service ports and data ports. As a result, the tasks in the service layer, can be modeled, analyzed and solved at symbolic level. This paper follows the techniques of automated planning derived from the AI field. The Task planning problem is modeled as a state transition system. Depending on different assumptions, various models are proposed. The two most commonly used models are the classical planning model [16-18] and the MDP model [19].

The classical planning model is not particularly well-suited to practical problems due to its strict assumptions. The MDP model has good expressivity but is less suitable for this study due to the large state space of ubiquitous robotic problems. To achieve better efficiency without sacrificing the expressivity on realistic problems, we propose the RMDP model by reducing the branching factor of the state transitions. The reason for the difficulty of solving MDP model comes from its large branching factor. It is assumed in RMDP model that after actions are executed by robotic components, the outcome could be a few predictable states, which are the successful state and a few failed states. Take grasping action for instance, if the manipulator fails to pick up one object, the object will either remain where it is or drop on the table or floor. This assumption dramatically decreases the branching factor of the state space. As shown in Fig. 3, in MDP model, the branching factor of state transition is as large as the state number $|S|$. While in RMDP model, the branching factor is much smaller. Further, the RMDP employs multi-valued state variables instead of binary variables commonly applied in the existing automated planning domain languages [20-21].

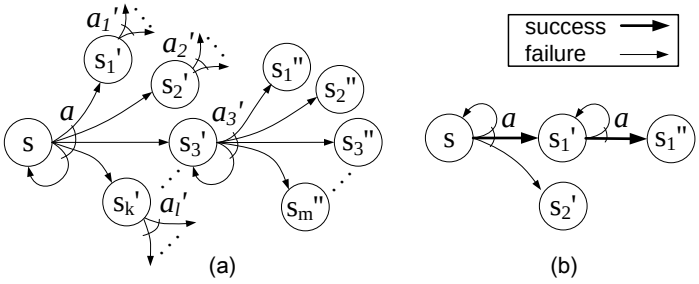

Fig. 3. (a) MDP model for state transition, (b) RMDP model for state transition

Definition 1: RMDP model is defined as a five-tuple $\Pi = (V, D, A, I, G)$:

- $V = \{v_1, v_2, \cdots\}$ is a finite set of state variables;
- $D = \{d_1, d_2, \cdots, d_n\}$ is a finite set of variable domains, each $v_i \in V$ with a finite domain $d_i \in D$. *V* and *D* define the planning space *S*, where state $s \in S$ is represented as a vector $[x_1, x_2, ..., x_n]$, where $x_i \in d_i$ is the value of variable v_i;
- $A = \{a_1, a_2, \cdots, a_m\}$ is a finite set of actions, each $a_i \in A$ is a triple (Pc, Ef, c) referred to the action's preconditions, effects and cost respectively.
- $I \in S$ denotes the initial state;
- $G \subseteq S$ denotes the set of goal states.

Definition 2: A partial state ξ is a value assignment of a subset of state variables $V_\xi \subseteq V$. Assuming $V_\xi = \{v_{\xi 1}, v_{\xi 2}, ..., v_{\xi k}\}$, ξ is represented with a set of variable-value pair $\{(v_{\xi 1}, x_{\xi 1}), (v_{\xi 2}, x_{\xi 2}), ...(v_{\xi k}, x_{\xi k})\}$.

Definition 3: A partial assignment is a function $\Gamma : S \times \xi \mapsto S$ represents the state transition on a partial set of state variables $V_\xi = \{v_{\xi 1}, v_{\xi 2}, ..., v_{\xi k}\}$. Assuming $\Gamma(s, \xi) = s'$ and $(v_{\xi i}, x_{\xi i}) \in \xi$, s' is obtained by setting each variable $v_{\xi i}$ to $x_{\xi i}$ in state *s*.

The precondition *Pc*(*a*) of action *a* is defined with a partial state $\xi^{Pc(a)}$, where $(v_{\xi i}, x_{\xi i}) \in \xi^{Pc(a)}$ denotes the value of variable $v_{\xi i}$ should be $x_{\xi i}$ to satisfy the precondition. The effects *Ef*(*a*) of action *a* is defined with an effect list $[ef_1, ef_2, ..., ef_k]$, where $ef_i = (p_i, \xi_i^{Ef(a)})$ denotes the outcome state will be the parial assignment $\Gamma(s, \xi_i^{Ef(a)})$ with probability p_i, after the action's execution. Each action has a cost *c*, which acts like a reward function in MDP model.

The planning result of RMDP is also a policy $\pi : S \mapsto A$. The expected value $VF^\pi(s)$ of a policy π at a given state s satisfies:

$$
\begin{aligned}
VF^{\pi}(s) &= E\{\sum_{t=0}^{\infty}\gamma^{t}R(s_t,a_t)\,|\,s_0=s\} \\
&= c(\pi(s_0))+\gamma c(\pi(s_1))+\gamma^2 c(\pi(s_2))+... \\
&= c(\pi(s))+\gamma\sum_{(p_i,\xi_i)\in Ef(\pi(s))} p_i\cdot VF^{\pi}(\Gamma(s,\xi_i^{Ef(\pi(s))}))
\end{aligned}
\tag{1}
$$

, where $c(a)$ represents the cost of action a; γ is the discount factor; $Ef(a)$ represents the effects of action a. Similar to the MDP model, the optimal value function satisfies:

$$
VF^{*}(s)=\min_{a\in A}[c(a)+\gamma\sum_{(p_i,\xi_i)\in Ef(a)} p_i\cdot VF^{*}(\Gamma(s,\xi_i^{Ef(a)}))] \tag{2}
$$

The optimal policy π^{*} is the one that minimizes the value function:

$$
\pi^{*}(s)=\arg\min_{a\in A}[c(a)+\gamma\sum_{(p_i,\xi_i)\in Ef(a)} p_i\cdot VF^{*}(\Gamma(s,\xi_i^{Ef(a)}))] \tag{3}
$$

With RMDP model, we derive the following modified Dynamic Programming (DP) algorithm according to equation (2) (3):

Algorithm 1. Modified DP

1. initialize value function $VF(s)=0$ for all $s\in S$
2. repeat for each episode:
3. initialize *s* as the initial state
4. repeat for each step of episode:
5. for each a that is applicable to *s*:
6. $(s_i,\xi_i)\leftarrow Ef(s,a)$
7. $a\leftarrow\arg\min_{a\in A}[c(a)+\gamma\sum p_i\cdot VF(\Gamma(s,\xi_i))]$
8. $VF(s)\leftarrow(1-\alpha)V(s)+\alpha[c(a)+\gamma\sum p_i\cdot VF(\Gamma(s,\xi_i))]$
9. $s\leftarrow\text{execute}(s,a)$
10. until *s* is the goal state
11. until VF converges
12. $\pi(s)\leftarrow\arg\min_{a\in A}[c(a)+\gamma\sum_{(p_i,\xi_i)\in Ef(a)} p_i\cdot VF(\Gamma(s,\xi_i))]$

This algorithm modifies the value function iteratively as depicted in line 7 and line 8, where $\alpha\in(0,1]$ is the step size, which controls the iteration speed. The value function is guaranteed to converge to VF^{*} given a sufficiently large number of iterations.

4 Online model learning

The automated planning methods in AI only consider the offline problems. However for practical problems, the model should adapt to dynamic and nonstationary envi-

ronments. For example, changing the position of furniture would alter the moving ability of the mobile robots. Consequently, as new information gained, the model should be improved online to make it more accurately match the real environment. This means the planner will gradually compute a new way of behaving to match the new model.

The RMDP model describes the environmental uncertainty with the probabilistic effects of actions. This naturally allows a method for online learning of the transition probabilities. Recall that the probabilistic effects $Ef(a)$ is defined with an effect list $[e_1, e_2, ..., e_k]$, where $e_i = (p_i, \xi_i)$ denotes the state will be changed by partial assignment $s_i' = \Gamma(s, \xi_i)$ with probability p_i, after the executing action *a*. Intuitively, the transition probability p_i can be estimated by the number of times that it takes action *a* in state *s* to get to the state *s'*, divided by the total number of times that take action *a* in state *s*:

$$p_i = \frac{\#[(s,a) \mapsto s_i']}{\#[(s,a)]} \tag{4}$$

Following each execution instance of action *a*, the execution result is recorded by the online learning module. This module calculates the transition probabilitis as more execution results are recorded according to equation (4). However, since this method factors in each instance with equal weight, there could be great latency if the environment suddenly changes. A solution is to make the system more "forgetful", and give higher credit to recent experiences.

Suppose the action *a* has been executed *n* times since the system was started. Further supposing these executions happened on time steps t_1^a, t_2^a, …, t_n^a and assuming the execution results are r_1^a, r_2^a, …, r_n^a, where each $r_i^a \in \{1, 2, ..., |Ef(a)|\}$. Define a binary function as follows:

$$\iota^a(k,i) = \begin{cases} 0 \text{ if } r_i^a = k \\ 1 \text{ elsewise} \end{cases} \tag{5}$$

An exponential forgetting model is defined as:

$$p_k^a(n) = \frac{\sum_{i=1}^{n} \iota^a(k,i) e^{-\alpha(t_n^a - t_i^a)}}{\sum_{i=1}^{n} e^{-\alpha(t_n^a - t_i^a)}} \tag{6}$$

, where $\alpha \in \mathbb{R}^+$ is the forgetting factor. The larger the value of α, the more the system is biased towards the recent experiences. α is set to 0.1 in the experiment described in the following section.

An iterative version which is easier to implement is as follows:

$$p_k^a(n) = p_k^a(n-1) + \frac{1}{\varphi(n)}(t^a(k,i) - p_k^a(n-1)) \tag{7}$$

$$\text{where } \varphi(n) = e^{-\alpha(t_n^a - t_{n-1}^a)}\varphi(n-1) + 1 \tag{8}$$

5 Experiment results

5.1 The robot bar-tending experiment

A bar-tending scenario was implemented as ubiquitous robotic system based on the middleware technology discussed in section 2. The task was to serve drinks to customers. Different robotic devices were developed into components as shown in Fig. 4. There were two mobile tables with different color and size, one dual-arm robot for grasping the drinks and snacks on the bar counter, three environmental cameras to help the localization, two laser sensors mounted on the mobile tables for localization and obstacles avoiding, and other software components such as path planning, object recognition, etc.

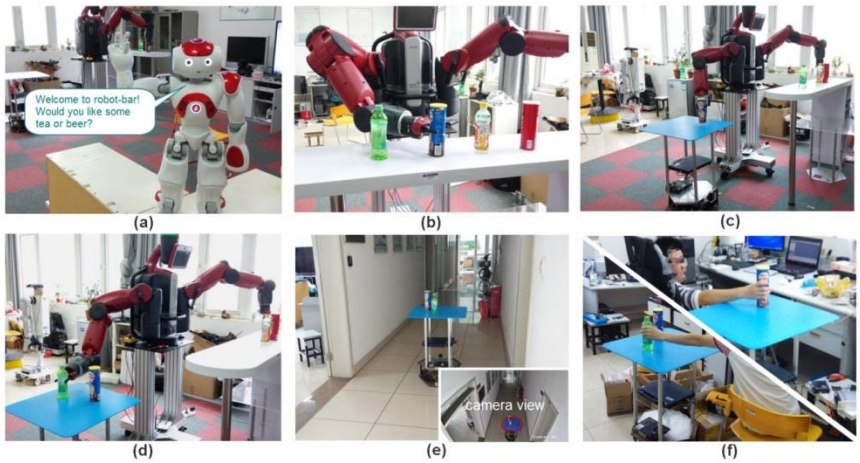

Fig. 4. Execution process of one bar-tending task

This task environment comprised two rooms connected by a corridor. One humanoid robot standing by the entrance took the orders from patrons. The tasks were generated according to the customers' orders.

The states and actions are predefined as domain knowledge. The initial state is reported by each component through the FuncGet service port. The goal state is translated from the customer's order. When all these information is ready, the planner calculates the policy using modified DP algorithm as detailed in section 3.

The actions were then sent to each component for execution. Fig. 5 illustrates the execution process of one such task. First, the mobile table moved to the bar counter

through laser localization. Next the dual-arm robot picked up the drink and snack ordered by customers, and placed them on the mobile table as Fig. 5 (b-d) show. The mobile table then moved to the other room where two customers picked their food and drink while at their tables. As the mobile table traveled through the corridor, the laser failed to provide the localization information, so the environmental camera mounted on the ceiling started to localize the robot as Fig. 5 (e). The execution status of each action was monitored by querying on the ExeStatusGet service port. Once any error or failure occurred, a new action was executed according to the policy without re-planning the whole task.

5.2 Online adapting to environment changes

The online learning method was tested by the following scenario: The objective is to have the drinks delivered to the next room. Initially the doorway was fully open, hence the mobile tables had no difficulty passing through it. Sometime later half of the doorway was closed, so that the larger mobile robot often failed to pass through.

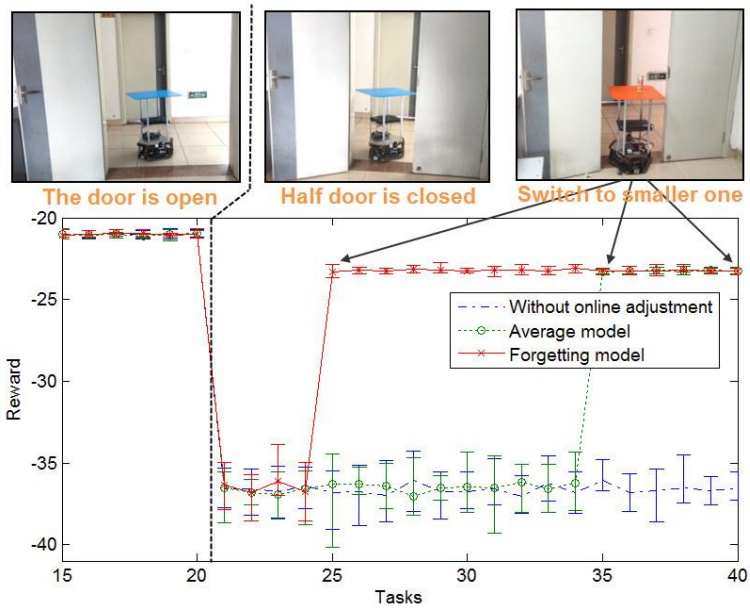

Fig. 5. Online adjusting the model when the environment changes

The same task was repeated 40 times. In the first 20 times, the door was open. The bigger robot - which is initially located closer to the bar counter - was sent to deliver the drinks. From the 21st episode onwards, the doorway was half closed. As Fig. 6 shows, a sharp drop in the reward was registered as a result of execution failure. Three methods were compared. Namely: using the average model; forgetting model; and the one without online adjustment. The two methods with online learning altered the success rate of the corresponding actions. Consequently, they switched to another

policy that employed the smaller sized mobile table to do the task. Conversely, the method without online learning still stuck to the poor policy. It may be readily noted that the one with forgetting model had a quicker response to the environmental changes when compared to the one based on the average model. In fact this effect is more pronounced when more trials are made before the environment changes.

6 Conclusion

Given the increasing popularity of the ubiquitous robotic system as a solution offering better environmental intelligence, the RMDP model proposed in this work has been proved to be suitable for the symbolic planning problems that are especially large-scale and with uncertainties.

This result has been achieved by retaining the probabilistic features of the MDP model while at the same time decreasing the branching factor of the state transitions. A robotic bar-tending task was implemented based on RMDP model and was solved using a modified DP algorithm. The experiments demonstrated that the RMDP model facilitates online learning, which adapts the model to environmental changes. A "forgetting" model is employed for this online learning process. The results showed that the improved model more accurately matched the real environment and the planning results also improved accordingly.

Acknowledgement

The authors gratefully acknowledge YASKAWA Electric Corporation for supporting the collaborative research funds under the project of "Research and Development of Key Technologies for Smart Factory".

Reference

1. A. Chibani, Y. Amirat, S. Mohammed, E. Matson, N. Hagita, M. Barreto, Ubiquitous robotics: Recent challenges and future trends, Robotics and Autonomous Systems, 61 (2013) 1162-1172.
2. M. Reggente, A. Mondini, G. Ferri, B. Mazzolai, A. Manzi, M. Gabelletti, P. Dario, A.J. Lilienthal, The dustbot system: Using mobile robots to monitor pollution in pedestrian area, Proc. of NOSE, 23 (2010) 273-278.
3. B. K. Kim, N. Tomokuni, K. Ohara, T. Tanikawa, K. Ohba, S. Hirai, Ubiquitous localization and mapping for robots with ambient intelligence, IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, p. 4809-4814.
4. Y. Koide, T. Kanda, Y. Sumi, K. Kogure, H. Ishiguro, An approach to integrating an interactive guide robot with ubiquitous sensors, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2004), IEEE, pp. 2500-2505.
5. Y.G. Ha, J.C. Sohn, Y.J. Cho, H. Yoon, A robotic service framework supporting automated integration of ubiquitous sensors and devices, Information Sciences, 177 (2007) 657-679.

6. R. Lundh, L. Karlsson, A. Saffiotti, Autonomous functional configuration of a network robot system, Robotics and Autonomous Systems, 56 (2008) 819-830.
7. D.S. Nau, T.C. Au, O. Ilghami, U. Kuter, J.W. Murdock, D. Wu, F. Yaman, SHOP2: An HTN planning system, J. Artif. Intell. Res.(JAIR), 20 (2003) 379-404.
8. M. Helmert, The Fast Downward Planning System, J. Artif. Intell. Res.(JAIR), 26 (2006) 191-246.
9. M. Barbosa, A. Bernardino, D. Figueira, J. Gaspar, N. Gonçalves, P.U. Lima, P. Moreno, A. Pahliani, J. Santos-Victor, M.T.J. Spaan, ISRobotNet: A testbed for sensor and robot network systems, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 2827-2833.
10. M. Cirillo, L. Karlsson, A. Saffiotti, A Human-Aware Robot Task Planner, in: Proceedings of the 19th International Conference on Automated Planning and Scheduling, AAAI press, 2009, pp. 58-65.
11. W. Wang, Q. Cao, X. Zhu, S. Liang, A Framework for Intelligent Service Environments Based on Middleware and General Purpose Task Planner. 2015 International Conference on Intelligent Environments (IE), IEEE, pp. 184-187.
12. M. Zhai, G. B. McKenna, Surface energy of a polyurethane as a function of film thicknessProceedings of the annual technical conference, Society of Plastics Engineers, ANTEC, Las Vegas. 2014.
13. W. Wang, Q. Cao, X. Zhu, M. Adachi. An automatic switching approach of robotic components for improving robot localization reliability in complicated environment. Industrial Robot: An International Journal, 2014, 41(2), pp. 135-144.
14. M. Zhai, G. B. McKenna, Elastic modulus and surface tension of a polyurethane rubber in nanometer thick films, Polymer, 2014, 55(11): 2725-2733.
15. N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, W.-K. Yoon, RT-middleware: distributed component middleware for RT (robot technology), in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), IEEE, pp. 3933-3938.
16. A.L. Blum, M.L. Furst, Fast planning through planning graph analysis, Artificial Intelligence, 90 (1997) 281-300.
17. H.A. Kautz, B. Selman, Planning as Satisfiability, in: Proceedings of the 10th European conference on Artificial intelligence, John Wiley & Sons, Inc, 1992, pp. 359-363.
18. B. Bonet, H. Geffner, Planning as heuristic search, Artificial Intelligence, 129 (2001) 5-33.
19. R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, MIT press Cambridge, 1998.
20. S. Sanner, Relational dynamic influence diagram language (rddl): Language description, Unpublished ms. Australian National University, 2010.
21. M. Fox, D. Long, PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains, J. Artif. Intell. Res.(JAIR), 2003, 20, 61-124.