Knowledge Granularity and Action Selection

Yiming Ye[1] and John K. Tsotsos[2]

[1] IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 1A4

Abstract. In this paper we introduce the concept of knowledge granularity and study its influence on an agent's action selection process. Action selection is critical to an agent performing a task in a dynamic, unpredictable environment. Knowledge representation is central to the agent's action selection process. It is important to study what kind of knowledge the agent should represent and the preferred methods of representation. One interesting research issue in this area is the knowledge granularity problem: to what detail should an agent represent a certain kind of knowledge. In other words, how much memory should an agent allocate to represent a certain kind of knowledge. Here, we first study knowledge granularity and its influence on action selection in the context of an object search agent - a robot that searches for a target within an environment. Then we propose a guideline for selecting reasonable knowledge granularity for an agent in general.

1 Introduction

An agent is a computational system that inhabits dynamic, unpredictable environments. It has sensors to gather data about the environment and can interpret this data to reflect events in the environment. Furthermore, it can execute motor commands that produce effects in the environment. One important property of the agent is its awareness — it has knowledge about itself and the world. This knowledge can be used to guide its action selection process when exhibiting goal-directed behaviors. Here we address the following question: "How much detail should the agent include in its knowledge representation so that it can efficiently achieve its goal?"

There are two extremes regarding granularity of knowledge representation. At one end of the spectrum is the purely reactive scheme [3] which requires little or even no knowledge representation. At the other end of the spectrum is the purely planning scheme which requires the agent to maintain as much detailed knowledge as possible. Experience suggests that neither purely reactive nor purely planning systems are capable of producing the range of behaviors required by intelligent agents in a dynamic, unpredictable environment. For example, Tyrrell [10] has noted the difficulty of applying , without modification, the model of Brooks [3] to the problem of modeling action selection in animates whose behavior is supposed to mirror that of real animals. On the other hand, although it is theoretically possible to compute the optimal action selection policy for an agent that has a fixed set of goals and that lives in a deterministic or probabilistic environment [10], it is impossible to do so in most practical situations for the following reasons: (A) resource limitations (time limit, computation complexity [13], memory limit); (B) incomplete and incorrect information (knowledge difference [14], sensor noise, etc); (C) dynamic, non-deterministic environment. Thus, many researchers argue to use hybrid architectures [11] [5] [8] [7], a combination of classical and alternative approaches, to build agent systems. One example is the layered architecture [5][8]. In such an architecture, an agent's control subsystems are arranged into a hierarchy, with higher layers dealing with information at increasing levels of abstraction. Thus, the very lowest layer might map raw sensor data directly onto effector outputs, while the uppermost layer deals with long-term goals.

This paper offers an alternative point of view of the spectrum of knowledge abstraction based on the granularity of knowledge representation. The goal is to find the proper balance in representing an agent's knowledge such that the representation is detailed enough for the agent to select reasonable actions, and at the same time it is coarse enough that it does not exhaust the agent's resources when selecting those reasonable actions. Thus, the following important issues arise. The first is how to define the granularity of the agent's representation of a certain kind of knowledge. The second is how this granularity of knowledge representation influences the agent's action selection process. The third is how to find the granularity of representation such that the action selection process achieves a satisfactory performance.

In this paper, we try to answer the above questions in the context of an autonomous object search agent. Object search is the task of searching for a given object in a given environment by a robotic agent equipped with a pan, tilt, and zoom camera. It is clear that exhaustive, brute-force blind search will suffice for its solution; however, the goal of the agent is to design efficient strategies for search, because exhaustive search is computationally and mechanically prohibitive for non-trivial situations. The action selection task for the agent refers to the task of selecting the sensing parameters (the camera's position, viewing direction and viewing angle size) so as to bring the target into the field of view of the sensor and to make the target in the image easily detectable by the given recognition algorithm. Sensor planning for object search is very important if a robot is to interact intelligently and effectively with its environment. In [12] [13] Ye and Tsotsos systematically study the task of object search and give an explicit algorithm to control the state parameters of the camera by considering both the search agent's knowledge about the target distribution and the ability of the recognition algorithm.

In this paper, We first briefly describe the object search agent and its action selection strategy. Then we study the issue of knowledge granularity with respect to object search agent and present experimental results. Finally, we provide a guideline for an agent in general to adapt its knowledge granularity according to environmental and task-specific demands.

2 The Object Search Agent

2.1 Formulation

We need to formulate the agent's sensor planning task in a way that incorporates the available knowledge of the agent and the detection ability of the recognition algorithm.

The search region Ω can be in any form, such as a room with many tables, etc. In practice, Ω is tessellated into a series of elements c_i, $\Omega = \bigcup_{i=1}^{n} c_i$ and $c_i \bigcap c_j = 0$ for $i \neq j$. In the rest of the paper, it is assumed that the search region is an office-like environment and it is tessellated into little cubes of the same size.

An operation $\mathbf{f} = \mathbf{f}(x_c, y_c, z_c, p, t, w, h, a)$ is an action of the search agent within the region Ω. Here (x_c, y_c, z_c) is the position of the camera center (the origin of the camera viewing axis); (p, t) is the direction of the camera viewing axis (p is the amount of pan $0 \leq p < 2\pi$, t is the amount of tilt $0 \leq t < \pi$); (w, h) are the width and height of the solid viewing angle of the camera; and a is the recognition algorithm used to detect the target.

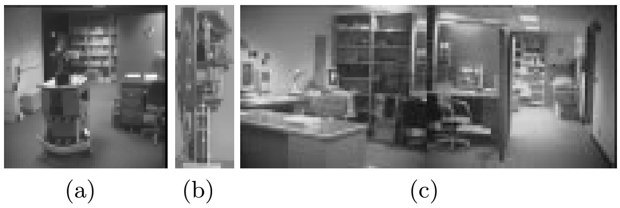

Fig. 1. *An example hardware of a search agent and a search environment. (a) The search agent - a mobile platform equipped with a camera; (b) The pan, tilt, and zoom camera on the platform; (c) An example search region.*

The agent's knowledge about the possible target position can be specified by a probability distribution function $\mathbf{p}$, so that $\mathbf{p}(c_i, \tau_{\mathbf{f}})$ gives the agent's knowledge about the probability that the center of the target is within cube c_i before an action $\mathbf{f}$ (where $\tau_{\mathbf{f}}$ is the time just before $\mathbf{f}$ is applied). Note, we use $\mathbf{p}(c_o, \tau_{\mathbf{f}})$ to represent the probability that the target is outside the search region at time $\tau_{\mathbf{f}}$.

The detection function on Ω is a function $\mathbf{b}$, such that $\mathbf{b}(c_i, \mathbf{f})$ gives the conditional probability of detecting the target given that the center of the target is located within c_i and the operation is $\mathbf{f}$. For any operation, if the projection of the center of the cube c_i is outside the image, we assume $\mathbf{b}(c_i, \mathbf{f}) = 0$. If the cube is occluded or it is too far from the camera or too near to the camera, we also have $\mathbf{b}(c_i, \mathbf{f}) = 0$. It is obvious that the probability of detecting the target by applying action $\mathbf{f}$ is given by

$$P(\mathbf{f}) = \sum_{i=1}^{n} \mathbf{p}(c_i, \tau_{\mathbf{f}}) \mathbf{b}(c_i, \mathbf{f}) \ . \tag{1}$$

The reason that the term $\tau_{\mathbf{f}}$ is introduced in the calculation of $P(\mathbf{f})$ is that the probability distribution needs to be updated whenever an action fails. Here we use Bayes' formula. Let α_i be the event that the center of the target is in cube c_i, and α_o be the event that the center of the target is outside the search region. Let β be the event that after applying a recognition action, the recognizer successfully detects the target. Then $P(\neg\beta \mid \alpha_i) = 1 - \mathbf{b}(c_i, \mathbf{f})$. It is obvious that the updated probability distribution value after an action $\mathbf{f}$ failed should be $P(\alpha_i \mid \neg\beta)$, thus we have $\mathbf{p}(c_i, \tau_{\mathbf{f}+}) = P(\alpha_i \mid \neg\beta)$. Where $\tau_{\mathbf{f}+}$ is the time after $\mathbf{f}$ is applied. Since the above events $\alpha_1, \ldots, \alpha_n, \alpha_o$ are mutually complementary and exclusive, from Bayes formula we get the following probability updating rule:

$$\mathbf{p}(c_i, \tau_{\mathbf{f}+}) \longleftarrow \frac{\mathbf{p}(c_i, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_i, \mathbf{f}))}{\sum_{j=1}^{n,o} \mathbf{p}(c_j, \tau_{\mathbf{f}})(1 - \mathbf{b}(c_j, \mathbf{f}))} \ . \tag{2}$$

where $i = 1, \ldots, n, o$.

The cost $\mathbf{t}(\mathbf{f})$ gives the total time needed to perform the operation $\mathbf{f}$.

Let $\mathbf{O}_{\mathbf{\Omega}}$ be the set of all the possible operations that can be applied. The effort allocation $\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_k\}$ gives the ordered set of operations applied in the search, where $\mathbf{f}_i \in \mathbf{O}_{\mathbf{\Omega}}$. It is clear that the probability of detecting the target by this allocation is:

$$P[\mathbf{F}] = P(\mathbf{f}_1) + [1 - P(\mathbf{f}_1)]P(\mathbf{f}_2) + \cdots + \{ \prod_{i=1}^{k-1} [1 - P(\mathbf{f}_i)]\} P(\mathbf{f}_k) \ . \tag{3}$$

The total cost for applying this allocation is:

$$T[\mathbf{F}] = \sum_{i=1}^{k} \mathbf{t}(\mathbf{f}_i) \ . \tag{4}$$

Suppose K is the total time that can be allowed in applying selected actions during the search process, then the task of sensor planning for object search can be defined as finding an allocation $\mathbf{F} \subset \mathbf{O}_{\mathbf{\Omega}}$, which satisfies $T[\mathbf{F}] \leq K$ and maximizes $P[\mathbf{F}]$.

Since this task is NP-Complete [13], we consider a simpler problem: decide only which is the very next action to execute. Our objective then is to select as the next action the one that maximizes the term

$$E(\mathbf{f}) = \frac{P(\mathbf{f})}{\mathbf{t}(\mathbf{f})} \ . \tag{5}$$

We have proved that in some situations, the one step look ahead strategy may lead to an optimal answer.

2.2 Selecting Camera Parameters

The agent needs to select the camera's viewing angle size and viewing direction for the next action $\mathbf{f}$ such that $E(\mathbf{f})$ is maximized. Normally, the space of available candidate actions is huge, and it is impossible to take this huge space of candidate actions into consideration. According to the image formation process and geometric relations, we have developed a method that can tessellate this huge space of candidate actions into a small number of actions that must be tried.

A brief description of the sensor planning strategy is as follows (please refer to [12] for detail). For a given recognition algorithm, there are many possible viewing angle sizes. However, the whole search region can be examined with high probability of detection using only a small number of them. For a given angle size, the probability of successfully recognizing the target is high only when the target is within a certain range of distance. This range is called the effective range for the given angle size. Our purpose here is to select those angles whose effective ranges will cover the entire depth D of the search region, and at the same time there will be no overlap of their effective ranges. Suppose that the biggest viewing angle for the camera is $w_0 \times h_0$, and its effective range is $[N_0, F_0]$. Then the necessary angle sizes $\langle w_i, h_i \rangle$ (where $1 \leq i \leq n_0$) and the corresponding effective ranges $[N_i, F_i]$ (where $1 \leq i \leq n_0$) are:

$$\begin{array}{l} w_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{w_0}{2})]; \\ h_i = 2arctan[(\frac{N_0}{F_0})^i tan(\frac{h_0}{2})]; \\ N_i = F_0(\frac{F_0}{N_0})^{i-1}; \quad F_i = F_0(\frac{F_0}{N_0})^i; \quad n_0 = \lfloor \frac{ln(\frac{D}{F_0})}{ln(\frac{F_0}{N_0})} - 1 \rfloor \end{array} . \tag{6}$$

For each angle size derived above, there are an infinite number of viewing directions that can be considered. We have designed an algorithm that can generate only directions such that their union can cover the whole viewing sphere with minimum overlap [12].

Only the actions with the viewing angle sizes and the corresponding directions obtained by the above method are taken as the candidate actions. So, the huge space of possible sensing actions is decomposed into a finite set of actions that must be tried. Finally, $E(\mathbf{f})$ can be used to select among them for the best viewing angle size and direction. After the selected action is applied, if the target is not detected, the probability distribution will be updated and a new action will be selected again. If the current position does not seem to find the target, the agent will select a new position and begin to search for the target at the new position.

3 Knowledge Granularity for Search Agent

3.1 General Discussion

As we have illustrated above, the object search agent uses its knowledge about the target position to guide its action selection process. This knowledge is encoded as a discrete probability density that is updated whenever a sensing action occurs. To do this, the search environment is tessellated into a number of small cubes, and each cube c is associated with a probability $p(c)$. To perfectly encode the agent's knowledge, the size of the cube should be infinitely small - resulting in a continuous encoding of the knowledge. But this will not work in general because an infinite amount of memory is needed. In order to make the system work, we are forced to represent the knowledge discretely - to use cubes with finite size. This gives rise to an interesting question: how we should determine the granularity of the representation (the size of the cube) such that the best effects or reasonable effects can be generated.

To make the discussion easier, we denote an object search agent $\mathbf{a}$ as $\mathbf{a} = \langle \mathbf{s}, \mathbf{k}_E, \mathbf{k}_p, \mathbf{G}, \mathbf{I}, \mathbf{t}_{select}, \mathbf{t}_{apply}, M, T, U \rangle$. Where $\mathbf{s}$ is the state parameters of the agent. $\mathbf{k}_E$ is the agent's knowledge about the geometric configuration of the environment. $\mathbf{k}_p$ is the agent's knowledge about the target position and is encoded as probabilities associated with tessellated cubes. $\mathbf{G}$ is the granularity function, which gives a measurement of the granularity of a certain knowledge representation scheme. $\mathbf{I}$ is the inference engine, which selects actions and updates agent's knowledge. By applying $\mathbf{I}$ to $\mathbf{k}_E$ and $\mathbf{k}_p$, an action is generated. The term $\mathbf{t}_{apply}$ is the cost function for applying actions: $\mathbf{t}_{apply}(\mathbf{f})$ gives the time needed to apply an action $\mathbf{f}$ and is determined by the time needed to take a picture and run the recognition algorithms. The term $\mathbf{t}_{select}$ is the cost function for selecting actions. M is the agent's memory limit. The memory used to store all the knowledge and inference algorithms should not exceed this limit. T is the time limit. The total time spent by the agent in selecting actions and executing actions should be within T. U is the utility function, which measures how well the agent performs during its search process within T.

The granularity function $\mathbf{G}$ can be defined as the total memory used by the agent to represent a certain kind of knowledge divided by the memory used by the agent to represent a basic element of the corresponding knowledge. For example, $\mathbf{G}(\mathbf{k}_p)$ gives the granularity measurement of the knowledge representation scheme $\mathbf{k}_p$. Suppose the length of the search environment is L units (the side length of a cube is one unit), the width of the search environment is W units, and the height of the search environment is H unit. Then the total environment contains LWH cubes. The probability $p(c)$ associated with each cube c is a basic element in the representation scheme $\mathbf{k}_p$. Suppose $m[p(c)]$ gives the memory of the agent used to represent $p(c)$. Then the total memory for the agent to represent $\mathbf{k}_p$ is $LWHm[p(c)]$. Thus, $\mathbf{G}(\mathbf{k}_p) = \frac{LWHm[p(c)]}{m[p(c)]} = LWH$.

Here we study the influence of $\mathbf{G}(\mathbf{k}_p)$ on the performance of the search agent. This performance can be measured by the utility and time limit pair $\langle U, T \rangle$. Where $U = P[\mathbf{F}]$ is calculated by Formula (3). The actions in $\mathbf{F}$ are selected according to Section 2. For a finer granularity $\mathbf{G}(\mathbf{k}_p)$, more time will be spent on action selection, leaving less time for action execution. The selected actions are generally with better quality because the calculation of $E(\mathbf{F})$ is more accurate in most situations. For a coarser granularity $\mathbf{G}(\mathbf{k}_p)$, less time will be spent on action selection, leaving more time for action execution. The selected actions are generally of lower quality because calculation of $E(\mathbf{F})$ is less accurate in most situations. In the following sections, we will present experiments to illustrate the influence of knowledge granularity on the agent's performance.

3.2 Experiments

A $2D$ simulation object search system is implemented to test the influence of the knowledge granularity on the performance of the action selection process. The system is implemented in C on IBM RISC System/6000.

The search environment is a $2D$ square as shown in Figure 2(a). If we tessellate the $2D$ square into 1000×1000 small square cells, then the relevant data for the system is as follows. The $2D$ camera has two effective angle sizes. The width of the first angle size is 40^o. Its effective range is $[50, 150]$. Its detection function is: $b(c, \mathbf{f}) = D(l)(1 - \frac{1}{6}\frac{\alpha}{41})$, where $\alpha < 20.5^o$ is the angle between the agent's viewing direction and the line connecting the agent center and the cell center, $D(l)$ is as shown in Figure 2(c), and l is the distance from the cell center to the agent center. According to formulas in Section 2, the width of the second effective angle size is 14^o, and its effective range is $[150, 450]$. The initial target distribution is as follows. The outside probability is 0.05. For any cell c within region A (bounded by $30 \leq x \leq 75$ and $30 \leq y \leq 75$), $p(c) = 0.000004$. For any cell c within region C (bounded by $600 \leq x \leq 900$ and $600 \leq y \leq 900$), $p(c) = 0.000005$. For any other cell c, $p(c) = 0.000001$. The agent is at position $[10, 10]$ in the beginning. We assume that there is only one recognition algorithm, thus the time needed to execute any actions are same.

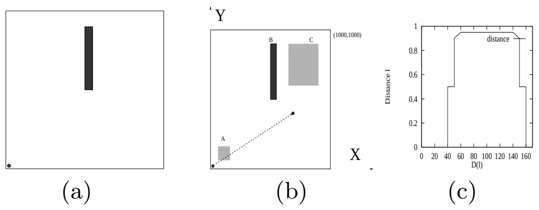

Fig. 2. *(a) The 2D environment. The agent is at the lower left corner of the region. An obstacle is present within the region. (b) The 2D environment when it is tessellated into a square of size* 1000×1000*. (c) The value of* $D(l)$*.*

In the first group of experiments, the agent only selects actions at position $[10, 10]$. In the second group of experiments, the agent first select 7 actions at position $[10, 10]$, then it moves to position $[700, 400]$ to begin the new search. The following sections list the experimental results.

Knowledge Granularity and Action Selection Time To select the next action $\mathbf{f}$, the agent need to calculate $P(\mathbf{f})$ (Equation (1)) for any candidate actions (Section 2). It is obvious that the knowledge granularity $\mathbf{G}(\mathbf{k}_p)$ has a great influence on the action selection time $\mathbf{t}_{select}(\mathbf{f})$. The higher the value of the knowledge granularity, the longer the time needed to select an action. We have performed a series of experiments to test the influence. The results are listed in the following table.

$\mathbf{G}(\mathbf{k}_p)$	[30 × 30]	[40 × 40]	50 × 50]	[60 × 60]	[70 × 70]	[80 × 80]	[90 × 90]	[100 × 100]	[200 × 200]	[300 × 300]	[400 × 400]	[500 × 500]
$\mathbf{t}_{select}$	15	30	41	91	121	157	217	289	1083	2443	4380	7467

Table 1

Note that $\mathbf{t}_{select}(\mathbf{f})$ (measured in seconds) is obtained by taking the difference in times obtained from the command "system("date")" executed before the system enters the action selection module and after the system finishes the action select module. The average value for different actions with the same granularity is taken as the value of $\mathbf{t}_{select}$ for the corresponding granularity. The accuracy is within one second.

The Error Associated with Knowledge Granularity Clearly the approximations involved in discretization will cause errors in calculating various values. In general, the higher the value of the knowledge granularity, the less the error caused by discretization. The error associated with knowledge granularity may influence the quality of the selected actions, and thus influence the performance of the agent.

Figures 3(d)(e)(f)(g) show how the granularity influences the error in calculating $P(\mathbf{f})$. We notice that in general the higher the knowledge granularity, the less the error of the calculated $P(\mathbf{f})$. For example, for $\mathbf{G}(\mathbf{k}_p) = 40 \times 40$, the error for the first action is 0.037115, while for $\mathbf{G}(\mathbf{k}_p) = 500 \times 500$, the error for the first action is 0.002356. Figures (h)(i)(j)(k) show the real probability of detecting the target $P[\mathbf{F}]$ with effort allocation $\mathbf{F}$ for different degrees of knowledge granularity. We can see that the higher the value of $\mathbf{G}(\mathbf{k}_p)$, the faster the system reaches its detecting limits.

Knowledge Granularity and Agent Performance In this section, we analyze the influence of knowledge granularity on the overall performance of the agent. Figure 3(h)(i)(j)(k) show that the higher the knowledge granularity, the better the quality of the selected actions. However, to achieve the expected benefits, we need to execute the action in addition to select them. Thus, both the action selection time and the action execution time are important.

For a higher knowledge granularity, although the selected actions might have good quality, the time needed to get these actions is also longer. If the time needed to execute an action is very long, then it is worth spending more time to select good actions. However, if the time needed to execute an action is very short, it may not be beneficial to spend a lot of time in action selections, because this amount of time can be used to execute all the possible actions. Thus, purely

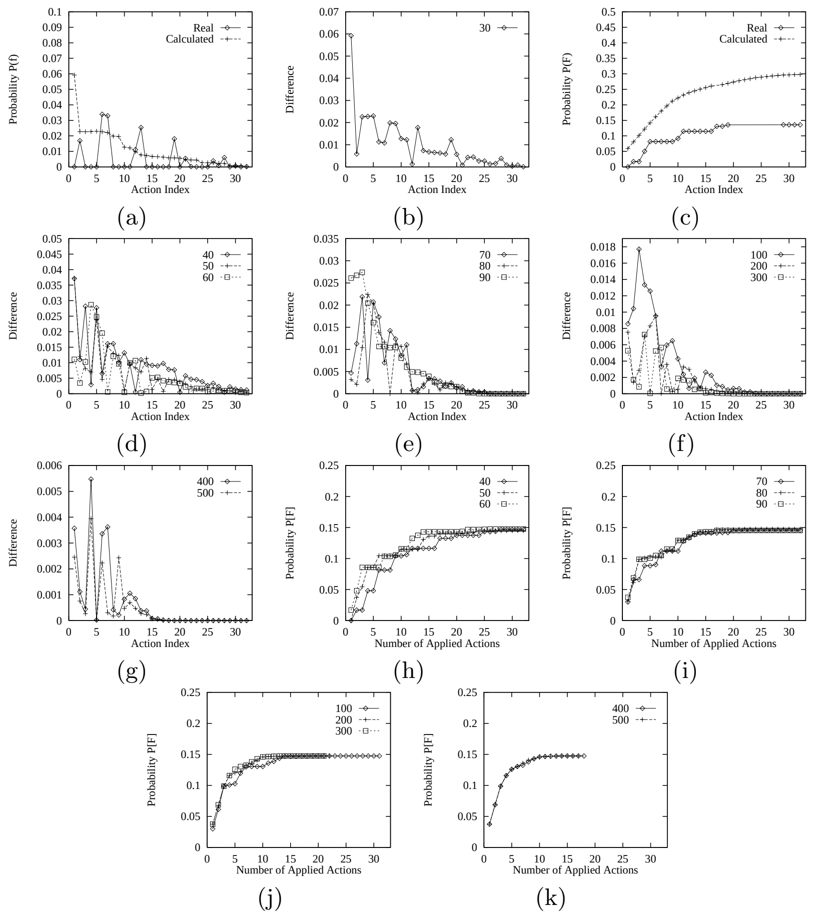

Fig. 3. *(a) The calculated probability $P(\mathbf{f})$ associated with knowledge granularity $\mathbf{G}(\mathbf{k}_p) = 30 \times 30$ for the selected action $\mathbf{f}$, and the real detection probability for $\mathbf{f}$. (b) The difference between the real and calculated probabilities for $\mathbf{G}(\mathbf{k}_p) = 30 \times 30$. (c) The calculated and the real probability of detecting the target for the selected effort allocation $\mathbf{F}$ for $\mathbf{G}(\mathbf{k}_p) = 30 \times 30$. (d), (e), (f), (g) The difference between the real and calculated probability of detecting the target for different knowledge granularity. (h), (i), (j), (k) The real probability of detecting the target for the given effort allocation at position $[10, 10]$.*

reactive strategy (no planning) only wins when the action execution time is short. When the action execution time is very long, we are forced to spend more time (use a higher knowledge granularity) in order to select good quality actions. Figure 4 illustrates how the performance of the agent is affected when assuming $\mathbf{t}_{apply}$ equals 1 second, 100 seconds, 1000 seconds, 10000 seconds, and 100000 seconds, respectively. The performance is represented by the probability of detecting the target for the selected effort allocation $\mathbf{F}$ verses the cost in

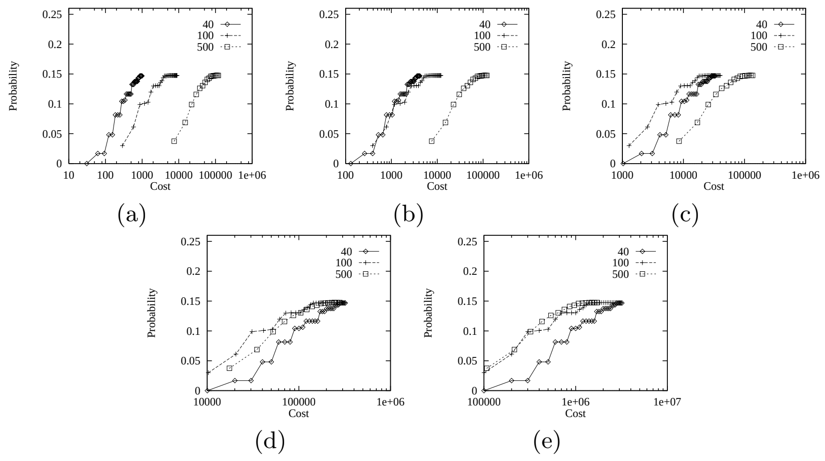

Fig. 4. *The influence of* $\mathbf{t}_{apply}$ *and* $\mathbf{G}(\mathbf{k}_p)$ *on the performance of the agent: (a)* $\mathbf{t}_{apply} = 1$ *seconds; (b)* $\mathbf{t}_{apply} = 100$ *seconds; (c)* $\mathbf{t}_{apply} = 1000$ *seconds; (d)* $\mathbf{t}_{apply} = 10000$ *seconds; (e)* $\mathbf{t}_{apply} = 100000$ *seconds.*

selecting and executing the effort allocation $\mathbf{F}$. We can see from Figure 4(a) that for $\mathbf{t}_{apply} = 1$, the performance of $\mathbf{G}(\mathbf{k}_p) = 40 \times 40$ is better than the performance of $\mathbf{G}(\mathbf{k}_p) = 100 \times 100$ and $\mathbf{G}(\mathbf{k}_p) = 500 \times 500$. As the cost in $\mathbf{t}_{apply}$ increases, the situation changes gradually. When $\mathbf{t}_{apply} = 10000$, $\mathbf{G}(\mathbf{k}_p) = 100 \times 100$ becomes the best knowledge granularity. When $\mathbf{t}_{apply} = 100000$, $\mathbf{G}(\mathbf{k}_p) = 500 \times 500$ becomes the best granularity.

When the Agent is Allowed to Move We also performed experiments for different inference engines $\mathbf{I}$ and similar results are obtained. Figure 5 lists the experimental results when the agent is allowed to move. The agent first selects 7 actions at position $[10, 10]$, then it moves to $[700, 400]$ to continue the search process. ¿From Figure 5, we can observe the same phenomena as we observed in the previous sections.

4 Selecting Knowledge Granularity

The experiments in the above section show that the level of knowledge granularity has a big impact on the quality and speed of the agent's behavior. It is thus important for an agent to adapt its knowledge granularity based on environmental and task-specific demands.

In general, if we represent an agent as $\mathbf{a} = \langle \mathbf{s}, \mathbf{k}_1, \cdots, \mathbf{k}_m, \mathbf{G}, \mathbf{I}, \mathbf{t}_{select}, \mathbf{t}_{apply}, M, T, U \rangle$, where $\mathbf{k}_1$, $\cdots$, and $\mathbf{k}_m$ are the representation schemes for the different kinds of knowledge maintained by the agent, and the other symbols are similar to those in Section 3.1, then we can define the knowledge granularity $\mathbf{G}(\mathbf{k}_i)$ for

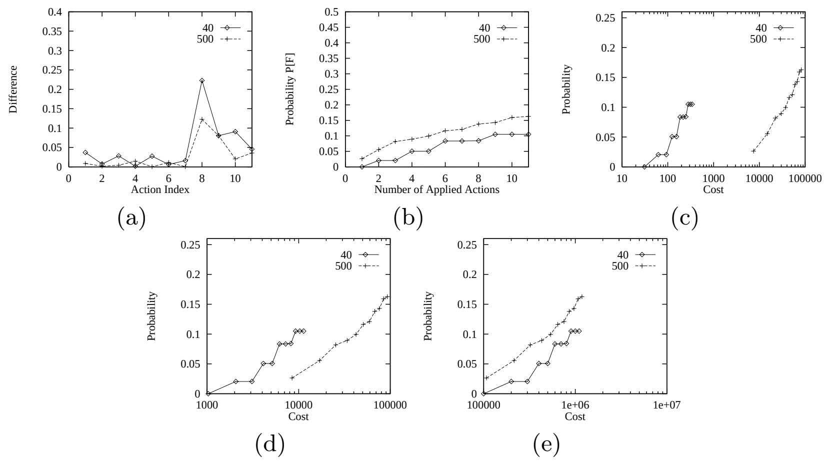

Fig. 5. *Experiments performed for another inference engine for* $\mathbf{G}(\mathbf{k}_p) = 40 \times 40$ *and* $\mathbf{G}(\mathbf{k}_p) = 500 \times 500$*. (a) Error in calculating* $P(\mathbf{f})$*; (b) Different effects with respect to* $P[\mathbf{F}]$*; (c) Performance when* $\mathbf{t}_{apply} = 1$*; (d) Performance when* $\mathbf{t}_{apply} = 1000$*; (e) Performance when* $\mathbf{t}_{apply} = 100000$*.*

$\mathbf{k}_i$ as the total amount of memory needed to represent the corresponding knowledge by scheme $\mathbf{k}_i$ divided by the memory needed to represent a basic element of the corresponding knowledge. In this section, we address the following interesting question: how can we select the knowledge granularity $\mathbf{G}(\mathbf{k})$ for a given representation scheme $\mathbf{k}$ such that the best agent performance or a relatively good agent performance can be achieved?

4.1 Best Granularity

In some situations, we are able to select the best knowledge granularity in the sense that it maximizes the performance of the agent. Here is an example. Suppose we have an agent whose task is to collect food from a region of length L within a time limit T. The agent can use different representation lengths $\Lambda = \{l_1, \ldots, l_q\}$ to represent the region. (suppose $\frac{L}{l_i}$ is integer, where $1 \le i \le q$). If the agent selects $l \in \{l_1, \ldots, l_q\}$ as its representation scheme $\mathbf{k}$ for the corresponding knowledge, then the corresponding knowledge granularity for this scheme will be $\mathbf{G}(\mathbf{k}) = \frac{L}{l}$. The total region is thus divided into $\frac{L}{l}$ units. The process of food collection is as follows. Before the collecting process, all the units of the region will be in the status of "not ready". When the collecting process begin, one of the units becomes "ready". The agent will then search for this unit. The time, $t_s(l)$, used by the agent to locate the unit is the time for the agent to select an action under the current representation scheme. Suppose $t_s(l) = \frac{1}{l}$. After the unit is located, the agent will collect food from this unit. The total time needed for the agent to collect food is the time needed for the agent to execute the selected action. Suppose it is $t_e(l) = Cl$ (where C is a constant).

The total amount of food that is collected is $B(l) = \frac{1}{l}$. When the agent finishes its food collection process at the selected unit, the status of another unit will become "ready". The agent will search for this new unit and collect food again from this new unit. This process will continue until the total time T is used up. If the total time T is exhausted when the agent is locating a unit or when the agent is collecting food within a unit, then the amount of collected food from the corresponding unit will be zero. It is obvious that the number of units that can be processed by the agent within T is $\frac{T}{t_s(l)+t_e(l)}$, and the number of units available is $\frac{L}{l}$.

The performance $\mathbf{P}$ of the agent is measured by the total amount of food collected by the agent and is given by the following formula:

$$\mathbf{P} = \begin{cases} \frac{L}{l}B(l) & \text{if } \frac{T}{t_s(l)+t_e(l)} \geq \frac{L}{l} \\ \lfloor \frac{T}{t_s(l)+t_e(l)} \rfloor B(l) & \text{if } \frac{T}{t_s(l)+t_e(l)} < \frac{L}{l} \end{cases} . \tag{7}$$

This is actually

$$\mathbf{P} = \begin{cases} \frac{L}{l^2} & \text{if } l \leq \sqrt{\frac{L}{T-CL}} \\ \lfloor \frac{Tl}{Cl^2+1} \rfloor \frac{1}{l} & \text{if } l > \sqrt{\frac{L}{T-CL}} \end{cases} . \tag{8}$$

The problem is to find a l in $\Lambda = \{l_1, \ldots, l_q\}$ such that $\mathbf{P}$ is maximized. The set Λ can be divided into two parts $\Lambda_A = \{l_1, \ldots, l_j\}$ and $\Lambda_B = \{l_{j+1}, \ldots, l_q\}$, such that all the elements in Λ_A are less than $\sqrt{\frac{L}{T-CL}}$, and all the elements in Λ_B are greater than or equal to $\sqrt{\frac{L}{T-CL}}$. It is obvious that for elements $l \in \Lambda_A$, the smallest one has the best performance because $\frac{L}{l^2}$ is a decreasing function. For elements $l \in \Lambda_B$, we can calculate the value of $\lfloor \frac{Tl}{Cl^2+1} \rfloor \frac{1}{l}$ to identify the best element. Then we compare the smallest element in Λ_A and the best element in Λ_B to identify the one that maximizes the performance of the system.

The above example shows that in some situations, an agent is able to identify an optimum knowledge granularity based on the task requirement (here T) and the environmental characteristics (here L). The basic method is to try to represent the performance of the agent as a function of the agent's knowledge granularity, and then to find the granularity that maximizes the performance.

In general, it is very difficult or even impossible to find a best knowledge granularity for an agent, because the performance of the agent might be influenced by many other factors in addition to the knowledge granularity. For example, there does not exist a best knowledge granularity for the object search agent, because its performance is also influenced by the initial target distribution. A granularity that is best for one distribution might not be the best for another distribution. Thus, in general, we need to relax our requirements. Instead of finding the best granularity, we search for a reasonable one such that a relatively good performance can be achieved. Because of the variations of different agent systems, it is impossible to provide a detailed procedure to select the acceptable granularity that can be applied to all the agent systems. However, we can provide a general guideline for the selection of the knowledge granularity.

4.2 Selecting Reasonable Granularity in Complex Agent Environment

In an agent environment where the relationships among the task constraints, the environments, and the knowledge granularity are very complex, the "demand-environment-granularity" (DEG) Hash Table can be used to select a reasonable granularity. The DEG Hash Table is a Hash Table such that the "key" is the combination of different factors and the "value" is the granularity that is appropriate for the corresponding factors. When an agent is informed of task requirements, it first transforms the task requirements and the environmental factors into a key. Then it retrieves the granularity from the DEG Hash Table based on the key. This granularity will be used by the agent to represent the corresponding knowledge.

For a complex agent environment, it might have more than one task constraints T_1, ..., T_{n_T}. Each $\mathbf{T_i}$ forms one component in the "key" of the DEG Hash Table. It can be divided into several groups $T_{i,1}$, ..., $T_{i,k}$ based on certain criteria. For example, the task constraint for an object search agent is the total time available for the search. This time constraint can be divided into groups like "from 1 second to 30 seconds", "from 30 seconds to 100 seconds", etc..

In addition to the task constraints, we should also consider the influences of the environmental factors when selecting the granularity. Suppose E_1, ..., E_{n_E} are the environment factors that need to be considered. Like above, each $\mathbf{E_i}$ can be divided into several groups $T_{i,1}$, ..., $T_{i,k}$ based on a certain criteria.

The DEG Hash Table is then looks like following:

$\mathbf{T_1}$	...	$\mathbf{T_{n_T}}$	$\mathbf{E_1}$	...	$\mathbf{E_{n_E}}$	$\mathbf{G}$
t_1	...	t_{n_T}	e_1	...	e_{n_E}	g
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 2

Where each row in the table, except the first one, gives a "key" $(t_1, \ldots, t_{n_T}, e_1, \ldots, e_{n_E})$ and the corresponding granularity value g. Here, t_i is a category (group) for the task constraint factor $\mathbf{T_i}$ and e_i is a category (group) for the environmental factor $\mathbf{E_i}$. Term g is the knowledge granularity value corresponding to the "key" and should be obtained by conducting various simulation experiments or theoretical analysis before the agent performs any task. When an agent is informed of a task, it first determines the key based on the current situations, and then uses this "key" to locate the knowledge granularity.

5 Conclusion

This paper offers an alternative point of view of the spectrum of knowledge abstraction based on the granularity of knowledge representation. It addresses the issue of *knowledge granularity* and its influence on the action selection process of an agent. The concept of *knowledge granularity* is defined and its influence on the performance of an object search agent is tested. The message derived from the experimental results is that knowledge granularity has a big impact on the performance of an agent. Thus, an appropriate knowledge granularity should be selected by an agent in order to guarantee a satisfactory result. Finally, we provide a general guideline on how an agent should adapt its knowledge granularity according to environmental and task-specific demands.

References

1. R. Bajcsy. Active perception vs. passive perception. In *Third IEEE Workshop on Vision*, pages 55–59, Bellaire, 1985.
2. R. Bajcsy. Perception with feedback. In *Image Understanding Workshop*, pages 279–288, 1988.
3. R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, (47):139–160, 1991.
4. Connel. *An Artificial Creature.* PhD thesis, AI Lab, MIT, 1989.
5. I. Ferguson. *Touring Machine: An Architecture for Dynamic, Rational, Mobile Agents.* PhD thesis, University of Cambridge, UK, 1992.
6. T. D. Garvey. Perceptual strategies for purposive vision. Technical Report Technical Note 117, SRI International, 1976.
7. M. Georgeff and A. Lansky. Reactive reasoning and planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682, Seattle, WA, 1987.
8. J. Muller and M. Pischel. Modelling interacting agents in dynamic environments. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, pages 709–713, Amsterdam, The Netherland, 1994.
9. S. Sen, S. Roychowdhury, and N. Arora. Effects of local information on group behavior. In *Proceedings of Second International Conference on Multi-Agent Systems*, pages 315–321, Kyoto, Japan, 1996.
10. T. Tyrrell. *Computational Mechanisms for Action Selection.* PhD thesis, Center for Cognitive Science, University of Edinburgh, England, 1993.
11. M. Wooldridge and N. Jenning. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
12. Y. Ye and J. K. Tsotsos. Where to look next in 3d object search. In *IEEE International Symposium for Computer Vision*, Florida, U.S.A., November 1995.
13. Y. Ye and J. K. Tsotsos. Sensor planning in 3d object search: its formulation and complexity. In *The 4th International Symposium on Artificial Intelligenceand Mathematics*, Florida, U.S.A., January 3-5 1996.
14. Y. Ye and J. K. Tsotsos. Knowledge difference and its influence on a search agent. In *First International Conference on AUTONOMOUS AGENTS*, Marina del Rey, CA, January 1997.