KNOWLEDGE GRANULARITY SPECTRUM, ACTION PYRAMID, AND THE SCALING PROBLEM

YIMING YE

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
E-mail: yiming@watson.ibm.com

JOHN K. TSOTSOS

Department of Computer Science, York University, Toronto, Canada M3J 1P3
E-mail: tsotsos@cs.yorku.ca

In this paper we introduce the concept of knowledge granularity and study the relationship between different knowledge representation schemes and the scaling problem. By scale to a task, we mean that an agent's planning system and knowledge representation scheme are able to generate the range of behaviors required by the task in a timely fashion. 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 how an agent should adapt its methods of representation such that its performance can scale to different task requirements. Here we study the following issues. One is the knowledge granularity problem: to what detail should an agent represent a certain kind of knowledge if a single granularity of representation is to be used. Another is the representation scheme problem: to scale to a given task, should an agent represent its knowledge using a single granularity or a set of hierarchical granularities.

Keywords: Agent; knowledge granularity; planning; performance; behaviors.

1. Introduction

This paper studies the scaling problem with respect to an agent — a computational system that inhabits dynamic, unpredictable environments. An agent 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. Usually, it has certain knowledge about itself and the world. This knowledge can be used to guide its action selection process when exhibiting goal-directed behaviors.^{1,13} It is important for an agent to choose a reasonable representation scheme in order to scale to the task at hand. There are two extremes regarding granularity of knowledge representation. At one end of the spectrum is the scheme that the selection of actions requires little or even no knowledge representation.³ At the other end of the spectrum is the purely planning scheme which requires the agent to maintain and use as much detailed knowledge

as possible. Experience suggests that neither of the above two extreme schemes are capable of producing the *range of behaviors* required by intelligent agents in a dynamic, unpredictable environment. For example, Tyrrell¹⁸ has noted the difficulty of applying, without modification, the model of Brooks³ 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,¹⁸ it is impossible to do so in most practical situations for the following reasons: (a) resource limitations (time limit, computation complexity,²⁰ memory limit); (b) incomplete and incorrect information (knowledge difference,²¹ sensor noise, etc.); (c) dynamic, nondeterministic environment. Thus, many researchers argue to use hybrid architectures,^{9,11,15,19} a combination of classical and alternative approaches, to build agent systems. One example is the layered architecture.^{9,15} 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. Or, the upper abstract space might be used to solve a problem and then the solution might be refined at successive levels of detail by inserting operators to achieve the conditions that were ignored in the more abstract spaces.^{12,14}

Much of the previous work on scaling emphasizes the absolute complexities (efficiency) of planning systems. We, however, believe that the problem of scaling is a relative term and is closely related to the task requirements of an agent in uncertain, dynamic or real-time environments. We will say that an agent *scales* to a given task, if the agent's planning system and knowledge representation scheme are able to generate the range of behaviors required by the task. We consider knowledge abstraction over a spectrum based on the granularity of knowledge representation. Our approach is different from previous approaches^{9,15} in that there is no logical relationship between elements of any two adjacent layers. We study the scaling problem related to different representation schemes, be it a single granularity scheme or a hybrid granularity scheme. Many factors, such as the planning engine, the way knowledge is represented, and the dynamic environment can influence whether an agent scales to a given task. Here, we concentrate on the influences of knowledge granularity. It is obvious that knowledge granularity can influence the efficiency of a given inference engine, since granularity influences the amount of data to be processed by the engine. It has been suggested that one may increase the computational efficiency by limiting the form of the statements in the knowledge base.^{16,7} In this paper, we study the relationship between different representation schemes and the performance of an agent's planning system. The goal is to find the proper scheme for representing an agent's knowledge such that the representation allows the agent to scale to a given task. We address the following issues. The first is how to define the granularity of an agent's representation of a certain

kind of knowledge. The second is how this granularity influences the agent's action selection performance. The third is how the hierarchical granularity representation influences the agent's action selection performance. The study of these issues can help an agent in finding a reasonable granularity or scheme of representation such that its behavior can scale to a given task.

2. A Case Study: The Object Search Agent

To start, we use object search as an example to study the influence of knowledge granularity on the performance of an 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 (Fig. 1). 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 nontrivial 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 Refs. 20 and 23 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 section, we first briefly describe the two-dimensional object search agent and its action selection strategy (please refer to Ref. 23 for corresponding three-dimensional descriptions). Then we study the issue of knowledge granularity with respect to object search agent and present experimental results.

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.

2.1. Task 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 two-dimensional form, such as a two-dimensional room with many two-dimensional tables, etc. In practice, Ω is tessellated into a series of elements c_i , ($\Omega = \cup_{i=1}^n c_i$) and ($c_i \cap c_j = \emptyset$) for $i \neq j$. In the rest of the paper, it is assumed that the search region is a two-dimensional office-like environment and it is tessellated into little square cells of the same size.

An operation ($\mathbf{f} = \mathbf{f}(x_c, y_c, \vartheta, w, a)$) is an action of the search agent within the region Ω . Here (x_c, y_c) is the position of the two-dimensional camera center (the origin of the camera viewing axis); ϑ is the direction of the camera viewing axis, $0 \leq \vartheta < 2\pi$; w is the width of the viewing angle of the camera; and a is the recognition algorithm used to detect the target.

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 square 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 square c_i is outside the image, we assume $\mathbf{b}(c_i, \mathbf{f}) = 0$. If the square 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

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

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 square 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 $(\alpha_1, \dots, \alpha_n, \alpha_o)$ are mutually complementary and exclusive, from Bayes formula we get the following probability updating rule:

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

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

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

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

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

The total cost for applying this allocation is:

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

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 ($\mathbf{F} \subset \mathbf{O}_\Omega$), which satisfies ($T[\mathbf{F}] \leq K$) and maximizes $P[\mathbf{F}]$.

Since this task is NP-Complete,²⁰ 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

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

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

2.2. The sensor planning strategy

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 Ref. 23 for details). 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 ϖ 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 , and its effective range is $[N_0, F_0]$. Then the necessary angle sizes

$\langle w_i \rangle$ (where $1 \leq i \leq n_0$) and the corresponding effective ranges $[N_i, F_i]$ (where $1 \leq i \leq n_0$) are:

$$w_i = 2 \arctan \left[\left(\frac{N_0}{F_0} \right)^i \tan \left(\frac{w_0}{2} \right) \right];$$

$$N_i = F_0 \left(\frac{F_0}{N_0} \right)^{i-1}; \quad F_i = F_0 \left(\frac{F_0}{N_0} \right)^i; \quad n_0 = \left\lfloor \frac{\ln \left(\frac{\varpi}{F_0} \right)}{\ln \left(\frac{F_0}{N_0} \right)} - 1 \right\rfloor. \quad (6)$$

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.²³

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.

2.3. Knowledge granularity for search agent

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 squares, and each square c is associated with a probability $p(c)$. To perfectly encode the agent's knowledge, the size of the square 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 squares with finite size. This gives rise to an interesting question: how we should determine the granularity of the representation (the size of the square) 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 $\mathbf{a} = \langle \mathbf{s}, \mathbf{k}_E, \mathbf{k}_p, \mathbf{G}, \mathbf{I}, \mathbf{t}_{\text{select}}, \mathbf{t}_{\text{apply}}, M, T, U \rangle$, 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 $\mathbf{t}_{\text{apply}}$

is the cost function for applying actions: $t_{\text{apply}}(\mathbf{f})$ 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 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 square is one unit), the width of the search environment is W units. Then the total environment contains LW squares. The probability $p(c)$ associated with each square 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 $LWm[p(c)]$. Thus, $\mathbf{G}(\mathbf{k}_p) = \frac{LWm[p(c)]}{m[p(c)]} = LW$.

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 Eq. (3). The actions in $\mathbf{F}$ are selected according to Sec. 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.

2.4. Experiments

A two-dimensional 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 two-dimensional square as shown in Fig. 2(a). If we tessellate the two-dimensional square into 1000×1000 small square cells, then the relevant data for the system is as follows. The two-dimensional camera has two effective angle sizes. The width of the first angle size is 40° . Its effective range is $[50, 150]$. Its detection function is: $b(c, \mathbf{f}) = D(l)(1 - \frac{1}{6} \frac{\alpha}{41})$, where $\alpha < 20.5^\circ$ 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 Fig. 2(c), and l is the distance from the cell center to the agent center. According to formulas in Sec. 2, the width

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

of the second effective angle size is 14° , 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 is the same.

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

2.4.1. Knowledge granularity and action selection time

To select the next action $\mathbf{f}$, the agent needs to calculate $P(\mathbf{f})$ [Eq. (1)] for any candidate actions (Sec. 2). It is obvious that the knowledge granularity $\mathbf{G}(\mathbf{k}_p)$ has a great influence on the action selection time $t_{\text{select}}(\mathbf{f})$. The higher the value of the knowledge granularity, the longer the time needed to select an action. We have

Table 1.

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

performed a series of experiments to test the influence. The results are listed in the following table.

Note that $t_{\text{select}}(\mathbf{f})$ (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 t_{select} for the corresponding granularity. The accuracy is within one second.

2.4.2. 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)–3(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 3(h)–3(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.

2.4.3. Knowledge granularity and agent performance

In this section, we analyze the influence of knowledge granularity on the overall performance of the agent. Figures 3(h)–3(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 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 t_{apply} equals 1 s, 100, 1000 s, 10 000 s, and 100 000 s, respectively. The performance is represented by the probability of detecting the target for the selected effort allocation $\mathbf{F}$ verses the cost

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

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

in selecting and executing the effort allocation $\mathbf{F}$. We can see from Fig. 4(a) that for $t_{\text{apply}} = 1$, 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 t_{apply} increases, the situation changes gradually. When $t_{\text{apply}} = 10000$, $\mathbf{G}(\mathbf{k}_p) = 100 \times 100$ becomes the best knowledge granularity. When $t_{\text{apply}} = 100000$, $\mathbf{G}(\mathbf{k}_p) = 500 \times 500$ becomes the best granularity.

2.4.4. 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 seven actions at position $[10, 10]$, then it moves to $[700, 400]$ to continue the search process. From Fig. 5, we can observe the same phenomena as we observed in the previous sections.

3. Knowledge Granularity in General

In general, if we represent an agent as $\mathbf{a} = \langle \mathbf{s}, \mathbf{k}_1, \dots, \mathbf{k}_m, \mathbf{G}, \mathbf{I}, t_{\text{select}}, t_{\text{apply}}, M, T, U \rangle$, where $\mathbf{k}_1, \dots$, 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 Sec. 2.3, then we can define the knowledge granularity $\mathbf{G}(\mathbf{k}_i)$ for $\mathbf{k}_i$ as the total amount of memory needed to represent the corresponding knowledge by scheme $\mathbf{k}_i$

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 $t_{\text{apply}} = 1$; (d) Performance when $t_{\text{apply}} = 1000$; (e) Performance when $t_{\text{apply}} = 100000$.

divided by the memory needed to represent a basic element of the corresponding knowledge.

In this section, we study in general the influences of knowledge granularity on an agent's action selection performance. For a task oriented agent, a finer granularity usually results in a better selected action. However, the action selection time for a finer granularity is usually longer. Thus, a finer granularity requires more time for selecting actions, and has less time in executing actions. On the other hand, a coarser granularity requires less time for action selection, thus has more time for action execution. In other words, with respect to a fixed time constraint, an agent can usually execute more low quality actions for a coarser granularity, and less high quality actions for a finer granularity. It is thus very interesting to study how the performance of a task oriented agent is influenced by the degree of knowledge granularity, and how the agent should choose a reasonable granularity from the spectrum of knowledge abstraction.

Different agents use different kinds of knowledge and different kinds of action selection procedures. Because of the complexity and diversity of the world of agents, it is impossible to provide a general conclusion or solution with regard to knowledge granularity. What we can do is to group agents into different categories and study the behavior with respect to each category. It is obvious that the performance of an agent is influenced by the action execution time, t_e , the action selection time, t_s , the

Table 2.

Fig.	(aa)	(ab)	(ac)	(ad)	(ae)	(ba)	(bb)	(bc)	(bd)	(be)
$t_s(g)$	γ_1	γ_1	γ_1	γ_1	γ_1	γ_2	γ_2	γ_2	γ_2	γ_2
$Q(g)$	γ_1	γ_2	γ_3	γ_4	γ_5	γ_1	γ_2	γ_3	γ_4	γ_5
Fig.	(ca)	(cb)	(cc)	(cd)	(ce)	(da)	(db)	(dc)	(dd)	(de)
$t_s(g)$	γ_3	γ_3	γ_3	γ_3	γ_3	γ_4	γ_4	γ_4	γ_4	γ_4
$Q(g)$	γ_1	γ_2	γ_3	γ_4	γ_5	γ_1	γ_2	γ_3	γ_4	γ_5

total time constraint for the given task, T , and the quality Q of the selected and executed actions. t_s and Q are influenced by the knowledge granularity adopted by the agent. Suppose for a granularity g , the *average* time needed in selecting an action is $t_s(g)$, the *average* contributions of a selected action to the task is $Q(g)$. Assuming that the total contributions U made by an agent within the time constraint T can be represented by the sum of the average contributions of all the actions that is executed within T . Then, U can be represented as follows.

$$U(g) = \lfloor \frac{T}{t_e + t_s(g)} \rfloor Q(g).$$

In the following, we study how $U(g)$ is influenced by different $t_s(g)$ and $Q(g)$. This is done by assigning $t_s(g)$ and $Q(g)$ different functions of g and study how the value of $U(g)$ will be influenced. We assume that the total time constraint for the agent to perform a certain task is 100, $T = 100$; the granularity range used by the agent is $[1, 150]$. The following functions are used in our empirical study: $\gamma_1(g) = 5$; $\gamma_2(g) = \ln(g)$; $\gamma_3(g) = g + 1$; $\gamma_4(g) = g * g * g + 1$; $\gamma_5(g) = \exp(g) + 1$. These functions represent different relations between the knowledge granularity g and the entities to be discussed. Of course, there are all kinds of relations, the above functions are only a very small fraction of the possible relations. Function $\gamma_1(g)$ means that the entity is a constant, and thus is not influenced by granularity. For example, if $t_s(g) = \gamma_1(g)$, then the action selection time is not influenced by the granularity. If $Q(g) = \gamma_1(g)$, then the average contributions of a selected action are not influenced by granularity. Functions $\gamma_2(g)$, $\gamma_3(g)$, $\gamma_4(g)$, $\gamma_5(g)$ refer to different degrees of the influence of granularity on the entity. Figure 6 shows how the granularity influences the performance of the agent under different situations. Table 2 lists the indexes in Fig. 6 with respect to their corresponding situations for $t_s(g)$ and $Q(g)$. For example, Fig. 6(aa) corresponds to the situations that $t_s(g) = \gamma_1(g)$ and $Q(g) = \gamma_1(g)$.

From Fig. 6, we can notice that the action execution time t_e is a very important factor that influences the selection of the knowledge granularity. Figures 6(aa)–6(ae) show the situation when action selection time is not influenced by the granularity. In this special case, the finer the granularity, the better the performance, except for the first one (aa). Figures 6(ca)–6(ce) show situations when $t_s = g + 1$, while

Fig. 6. The influence of knowledge granularity on the performance of an agent.

$Q(g)$ equals to $\gamma_1, \dots, \gamma_5$ respectively. We notice that for a large execution time (≥ 50), the granularity should be low in order to guarantee that at least one action can be executed. Figure 6(ca) shows the situation that the benefit of each action is not influenced by the granularity, thus a smaller granularity is preferred no matter what the action execution time is. Figure 6(cc) shows the situation that the action's benefit is influenced in a linear way by the granularity. We notice that the result becomes complex. It depends on how the quality of the selected actions is influenced by the granularity. For example, for a small execution time, there are several granularities that can generate satisfactory results. These reasonable granularities are different for different action execution times. Similar analysis can be applied to other figures.

As a conclusion, in complex situations, the selection of granularity depends on many factors, such as task constraint (total time allowed to perform a task), action execution time, the influence of granularity on action selection time, and the qualities of actions selected with a given granularity, etc. In general, we can construct graphs like Fig. 6 to analyze the effects of granularity on the performance of the agent under different factors and constraints, and then select a favorable granularity.

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 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 $\Lambda = \{l_1, \dots, l_q\}$ to represent the region. (Suppose $\frac{L}{l_i}$ is integer, where $1 \leq i \leq q$.) If the agent selects $l \in \{l_1, \dots, l_q\}$ 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 begins, 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 Ψ of the agent is measured by the total amount of food collected by the agent and is given by the following formula:

$$\Psi = \begin{cases} \frac{L}{l} B(l) & \text{if } \frac{T}{t_s(l) + t_e(l)} \geq \frac{L}{l} \\ \left\lfloor \frac{T}{t_s(l) + t_e(l)} \right\rfloor B(l) & \text{if } \frac{T}{t_s(l) + t_e(l)} < \frac{L}{l} \end{cases} \quad (7)$$

This is actually

$$\Psi = \begin{cases} \frac{L}{l^2} & \text{if } l \leq \sqrt{\frac{L}{T - CL}} \\ \left\lfloor \frac{Tl}{Cl^2 + 1} \right\rfloor \frac{1}{l} & \text{if } l > \sqrt{\frac{L}{T - CL}} \end{cases} \quad (8)$$

The problem is to find a l in $\Lambda = \{l_1, \dots, l_q\}$ such that Ψ is maximized. The set Λ can be divided into two parts $\Lambda_A = \{l_1, \dots, l_j\}$ and $\Lambda_B = \{l_{j+1}, \dots, l_q\}$, such that all the elements in Λ_A are less than $\sqrt{L/(T - CL)}$, and all the elements in Λ_B are greater than or equal to $\sqrt{L/(T - CL)}$. 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 $\left\lfloor Tl/(Cl^2 + 1) \right\rfloor \frac{1}{l}$ 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,

Table 3.

$\mathbf{T}_1$	...	T_{n_T}	E_1	...	E_{n_E}	$\mathbf{G}$
t_1	...	t_{n_T}	e_1	...	e_{n_E}	g
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

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 $T_1, \dots, T_{n_T}$. Each $\mathbf{T}_i$ forms one component in the “key” of the DEG Hash Table. It can be divided into several groups $T_{i,1}, \dots, T_{i,k}$ 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 s to 30 s”, “from 30 s to 100 s”, etc.

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

The DEG Hash Table then looks like the following:

Each row in the table, except the first one, gives a “key” $(t_1, \dots, t_{n_T}, e_1, \dots, e_{n_E})$ 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. Knowledge Granularity Spectrum and Action Pyramid

From the above discussion we know that knowledge granularity has a big influence on the quality of selected actions. Usually, with respect to the knowledge granularity spectrum, the higher the value of the knowledge granularity, the better the quality of the selected actions. However, it is not always beneficial to use high granularity, because the cost usually increases as well. In order to benefit from both the short action selection time of low granularity and the high quality of the selected actions of high granularity, a hierarchy of granularity layers can be used to select actions. For example, we can choose several granularities for the purpose of action selection, as follows. First, the coarsest granularity is used to select a set of actions from the pool of all the actions. Then the second coarsest granularity is used to select an even smaller set of actions from the set of actions chosen before. This procedure continues until the actual action to be applied is selected according to the finest granularity. The sets of actions selected by different granularities form an action pyramid. In this section, we compare the performances of a single layer granularity scheme and the multilayer granularity scheme. We restrict our discussion to two layers because of limited space. Similar results can be obtained for more than two layers.

5.1. *Adding a coarse new layer for filtering out non-interesting actions*

Suppose after some analysis we find that granularity g_o is a favorable choice. We might increase the performance by adding a coarser layer. The idea is to use the coarser granularity to select a small set of actions that are suitable, and then use g_o to select an action to execute from this small set of actions. Suppose originally there are totally N actions to be selected from. Now consider adding another layer of granularity g_c . Suppose that for g_c , we need to collect the first n_{g_c} actions in order to guarantee that enough good actions are within this set of actions. In other words, the quality of the actions selected by g_o from the total N actions is almost the same as the quality of the actions selected by g_o from the n_{g_c} actions.

Suppose that for this planning system the action selection time is governed by $t(g, n)$, where g is the granularity and n is the number of actions to be considered. Now we compare the time needed to select one action for the two strategies. The time T_o^s needed to select an action from the single layer is: $T_o^s = t(g_o, N)$. The time needed to select an action T_n^s for the new strategy is: $T_n^s = t(g_c, N) + t(g_o, n_{g_c})$. Thus, the difference in selecting an action by the two strategy is given by:

$$T_n^s - T_o^s = t(g_c, N) + t(g_o, n_{g_c}) - t(g_o, N).$$

Fig. 7. The performance comparison when a coarse layer of granularity is added to preselect a small set of actions to consider.

Fig. 8. Continued: the performance comparison when a coarse layer of granularity is added to preselect a small set of actions to consider.

Figures 7 and 8 show the results of experiments that are performed to test the performance difference under different strategies. In our experiments, we assume that $t(g, n) = t_1(g)t_2(n)$. Where the function t_1 gives the sensitivity measurement of the granularity g with respect to the action selection time, and t_2 gives the sensitivity measurement of the number of actions to be considered y with respect to the action selection time. The index (i, j) in Fig. 7 means that $t_1(x) = \gamma_i(x)$, $t_2(y) = \gamma_j(y)$, where function γ is defined in the previous section. For example, Fig. 7(2,3) means that $t_1(x) = \gamma_2(x)$, $t_2(y) = \gamma_3(y)$. In the figure, the z axis is T^s , the difference in action selection time. Other axes are g_c and n_{g_c} . In the test, we set $N = 100$ and $g_o = 100$. Let g_c and n_{g_c} change. The range for n_{g_c} is $1 \leq n_{g_c} \leq 100$, the range for g_c is $15 \leq g_c \leq 100$. Figures 7 and 8 show the difference $T_n^s - T_o^s = t(g', 100) + t(100, n_{g'}) - t(100, 100)$ as a function of g_c and n_{g_c} . In order to make the comparison easier, we also draw the surface of $z = 0$. Figure 7(1,1) shows the situation that the action selection time is not influenced by the granularity and the number of actions to be selected from. In this situation, the two layer strategy is always worse than the single layer strategy by a constant. This constant is the time used to preselect the set of actions by the coarse layer. Figures 7(1,1)–(1,3) show the situation when the action selection time is not influenced by granularity. In this case, adding a new coarse layer does not save time, because the coarse layer itself will spend the same time as the old granularity g_o , and extra time must be spent by g_o to select an action from the preselected action pools by g_c . Figures 7(2,1)–(2,5) show the situation that the influence of granularity on the action selection is governed by γ_2 , while the influence of the number of actions is governed by $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ and γ_5 respectively. We notice that the more sensitive the action selection time is influenced by the number of actions in the action pool, the better the two layer strategy. This is illustrated by the increase of the area of those g_c and n_{g_c} that is below the plane $z = 0$. The reason is that a decrease in granularity for a more sensitive situation tends to have a bigger saving in action selection time. The same analysis can be applied to Figs. 7(3,2)–(3,4), Figs. 8(4,1)–(4,5) and Figs. 8(5,1)–(5,5). From Figs. 7 and 8 we also notice that for a fixed granularity g_c , the smaller the value of n_{g_c} , the better the two layer strategy. The reason is that a smaller n_{g_c} tends to save time for g_o . We can also notice that for a fixed n_{g_c} , the smaller the value of g_c , the better the two layer strategy.

From the above experiment, we know that in some situations adding a coarse layer can increase the performance of an agent. Thus, when a single granularity does not allow the agent to scale to the task at hand, we can consider adding a coarse layer to increase the chances of scaling. To do this, we can first draw the performance figure as above, and then select the granularity that corresponds to the lowest point on the surface as the granularity for the coarse layer.

5.2. Adding a finer layer to obtain better quality actions

Another way to use hierarchical representation to increase performance and chances

of scaling is to add a finer layer. The idea is to use the current granularity g_o to preselect a small set of candidate actions, and then use a finer granularity g_f to choose a better quality action to execute.

The utility for the single layer strategy is:

$$U(g_o) = \left\lfloor \frac{T}{t_e + t_s(g_o, N)} \right\rfloor Q(g_o).$$

For the two layer strategy, suppose n_{g_o} is the number of actions that must be selected by g_o in order to guarantee that the actions selected by g_f will reach a desired quality $Q(g_f)$. The time to select an action for the two layer strategy is: $t_s = t(g_o, N) + t(g_f, n_{g_o})$.

Suppose the total time available for the agent is T . The utility of the new strategy is:

$$U(g_f) = \left\lfloor \frac{T}{t_e + t_s(g_o, N) + t_s(g_f, n_{g_o})} \right\rfloor Q(g_f).$$

Experiments have been performed to show the performance difference of the new strategy and the old strategy, $U_{\text{diff}} = U(g_f) - U(g_o)$. In the experiments, we assume: $T = 100$, $g_o = 100$, $N = 100$. We also assume that the action execution time is $t_e = 6$. In general t_e has a big influence on the analyzing result. Here we take $t_e = 6$ as an example to study the influence of other factors on the agent performance.

As in the previous section, we assume $t_s(g, n) = t_1(g)t_2(n)$. $Q(g)$ is another function, which gives the quality of the action selected with granularity g . In the experiments, we take g_f as one variable, and n_{g_o} as another variable, and we draw the surface formed by U_{diff} when g_f and n_r are changing. The domain range for the two variables are $100 \leq g_f \leq 150$ and $1 \leq n_r \leq 100$. In Fig. 9, the index (i, j, k) means that the figure is drawn by setting $t_s(g, n) = \gamma_i(g)\gamma_j(n/70)$ and $Q(g) = \gamma_k(g)$. Figure 9(1,1,1) shows the situation when granularity and the number of actions to be selected has no influence on the action selection time, and granularity has no influence on the quality of the selected actions. In this situation, the two layer strategy is always worse than the one layer strategy, because it is a waste of effort to preselect a set of actions for the finer layer. Figures 9(2,2,2)(2,2,3) and (2,2,5), Figs. 9(2,3,2)(2,3,3) and (2,3,5), Figs. 9(3,3,2)(3,3,3) and (3,3,5) and Figs. 9(3,5,2) and (3,5,3) show that the more sensitive the quality of selected actions with respect to granularity, the more benefit the two layer strategy. The influence of the sensitivity of n_{g_o} with respect to action select time on the performance can be complex. For example, Figs. 9(2,2,2) and (2,3,2) and Figs. 9(3,3,2) and (3,5,2) show that a sensitive n_{g_o} with respect to active selection time is not good for the two layer strategy; Figs. 9(2,2,3) and (2,3,3) show the opposite; while Figs. 9(3,3,3) and (3,5,3) show sometime it is good and sometimes it is not good depending on different n_r and g_f . In general, many factors can influence the performance of the two strategies and a graph needs to be drawn in order to determine which strategy is better.

Fig. 9. The performance comparison when a finer layer of granularity is added to select good quality actions.

6. Conclusion

In this paper, we introduce the concept of knowledge granularity and study the relationship between different knowledge representation schemes and the scaling problem. We promote the viewpoint that the problem of scaling is closely related to an agent's task requirement. By scale to a task, we mean that an agent's planning system and knowledge representation scheme are able to generate the range of behaviors required by the task in a timely fashion. Here, we study the influence of knowledge granularity and related representation schemes on an agent's scaling problem. From the study of related issues with respect to an object search agent, we know that knowledge granularity can greatly influence the performance and scaling of an agent. We then perform experiments to study the influence of knowledge granularity on an agent's performance under various situations. Experimental results show that many factors can influence the values of reasonable granularities that allow an agent to scale to a given task, such as the task constraint and action execution time, etc. In complex situations, a reasonable granularity can be selected by constructing graphs as in Fig. 3 and analyzing the resulted graphs. Finally, we conduct experiments to compare the performance between a single layer granularity scheme and multiple layer granularity scheme. The reason to use several granularities from the granularity spectrum and to choose actions from the corresponding action pyramid is that the agent might benefit from both the short action selection time of low granularities and the high quality of the selected actions of high granularities. Experimental results show that a hierarchical representation scheme can produce a better performance in some situations, especially when the quality of the selected actions is greatly influenced by the granularity or when the action selection time is greatly influenced by the granularity.

Acknowledgments

The authors would like to thank the reviewers for their valuable comments.

References

1. J. Allen, A. Tate and J. Hendler, *Readings in Planning*, Morgan Kaufmann Publishers, USA, 1990.
2. R. Bajcsy, "Active perception vs. passive perception," *Third IEEE Workshop on Vision*, Bellaire, 1985, pp. 55–59.
3. R. A. Brooks, "Intelligence without representation," *Artif. Intell.* **47** (1991) 139–160.
4. D. Chapman, "Planning for conjunctive goals," *Artif. Intell.* **32**, 3 (1987) 333–377.
5. J. Connel, *An Artificial Creature*, Ph.D. thesis, AI Lab, MIT, 1989.
6. K. Currie and A. Tate, "O-plan: the open planning architecture," *Artif. Intell.* **31**, 1 (1986) 49–86.
7. J. Doyle and R. Patil, "Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services," *Artif. Intell.* **48** (1991) 261–298.
8. O. Etzioni, "Acquiring search-control knowledge via static analysis," *Artif. Intell.* **62** (1990) 255–302.

9. I. Ferguson, *Touring Machine: An Architecture for Dynamic, Rational, Mobile Agent*, Ph.D. thesis, University of Cambridge, UK, 1992.
 10. T. D. Garvey, "Perceptual strategies for purposive vision," Technical Report Technical Note 117, SRI International, 1976.
 11. M. Georgeff and A. Lansky, "Reactive reasoning and planning," *Proc. Sixth Nat. Conf. Artificial Intelligence*, Seattle, WA, 1987, pp. 677–682.
 12. F. Giunchiglia, A. Villaforita and T. Walsh, "Theories of abstraction," *AI Commun.* **10**, December (1997) 167–176.
 13. J. Hendler, A. Tate and M. Drummond, "AI planning: systems and technologies," *AI Magazine*, Summer 1990.
 14. C. Knoblock, "Automatically generating abstractions for planning," *Artif. Intell.* **68** (1994) 243–302.
 15. J. Muller and M. Pischel, "Modelling interacting agents in dynamic environments," *Proc. Eleventh European Conf. Artificial Intelligence*, Amsterdam, The Netherlands, 1994.
 16. B. Selman and H. Kautz, "Knowledge compilation and theory approximation," *J. ACM* **43**, 2 (1996) 193–224.
 17. S. Sen, S. Roychowdhury and N. Area, "Effects of local information on group behavior," *Proc. Second Int. Conf. Multi-Agent Systems*, Kyoto, Japan, 1996, pp. 315–321.
 18. T. Tyrrell, *Computational Mechanisms for Action Selection*, Ph.D. thesis, Center for Cognitive Science, University of Edinburgh, England, 1993.
 19. M. Wooldridge and N. Jennings, "Intelligent agents: theory and practice," *Knowledge Engin. Rev.* **10**, 2 (1995) 115–152.
 20. Y. Ye and J. K. Tsotsos, "Sensor planning in 3d object search: its formulation and complexity," *4th Int. Symp. Artificial Intelligence and Mathematics*, Florida, USA, January 3–5, 1996.
 21. Y. Ye and J. K. Tsotsos, "Knowledge difference and its influence on a search agent," *First Int. Conf. Autonomous Agents*, Marina del Rey, CA, January 1997b.
 22. Y. Ye and J. K. Tsotsos, "Knowledge granularity for task oriented agents," *Proc. Third Annual Conf. Autonomous Agents*, USA, 1999.
 23. Y. Ye and J. K. Tsotsos, "Sensor planning for 3d object search," *Comput. Vis. Imag. Underst.* **73**, 2 (1999) 145–169.
-

Yiming Ye obtained his M.S. degree from Chinese Academy of Sciences and his Ph.D. degree in computer vision from the University of Toronto. He joined IBM T.J. Watson Research Center as a research staff member in

1997.

He is an Associate Editor for *Electronic Commerce Research Journal*, Kluwer Academic Publishers and is on Advisory Board of Conferences and Workshops of Intelligent System Division of National Institute of Standards and Technology. He serves as PC members or chairs of various international conferences or workshops in robotics, AI, and intelligent agents. He was a member of Organizing Committee at IBM Research EXPO 2000, the biggest expo at IBM research within five years. He received IBM Invention Achievement Awards in July 1999 and September 2000 respectively; IBM Research Division Technical Group Award in 2000; and IBM Research Division External Recognition Award in 2001.

His interests are in web intelligence, electronic commerce, pervasive computing applied dynamics of computation and complex systems, multi-agent system, natural dialogue systems and computer vision and image processing.

John K. Tsotsos received his Ph.D. in 1980 from the University of Toronto in computer science. From 1980 to 1999 he was a Professor in that department and in the university's Department of Medicine.

He is currently a Profes-

sor of computer science at York University, also in Toronto, and Director of the York Center for Vision Research. He has served on numerous conference committees and the editorial boards of *Computer Vision and Image Understanding*, *Image and Vision Computing Journal*, *Computational Intelligence and AI & Medicine*. He was the General Chair of the 7th IEEE International Conference on Computer Vision, Corfu, Greece, 1999.

His research focusses on biologically plausible models of visual attention, the development of a visually-guided robot to assist physically disabled children and perceptually-guided robot control mechanisms.