BUILDING PERCEPTION FOR SCHEDULING AND EXECUTING A TASK USING MULTIAGENT SYSTEMS

Jaya Sil
Computer Sc. & Engg. Dept.
Calcutta University
92, A.P.C. Road, Calcutta–700009
India
jayasil@hotmail.com

Amit Konar
Electronics & Tele Communication Engg. Dept.
Jadavpur University
Calcutta–700032
India
babu25@hotmail.com

Abstract

Involvement of the multi-agent systems for scheduling and executing a complex task autonomously, has been addressed in the paper to solve the navigational problems in robotics. Here, a complex task is broken into a number of subtasks and one to one mapping between subtask and agent is determined from the given relationship among various agents. In a dynamic environment the state-space model of the multi-agent system has been designed using the agents' spatial coordinate positions and sensory data to facilitate performing a task through cooperation. The paper proposes an efficient strategy for collision resolution that may occur at an instant of time, while the robots navigate in the common environment. The distributed agent concept has been adopted in the paper to implement the whole scheme in JAVA.

Keywords multi-agent systems, state-space model, sensory information, collision, object oriented approach.

1. Introduction

An agent (robot) is an active entity that can execute a task autonomously to achieve a definite goal. Information obtained through various sources, such as sensor, camera, are used to create the knowledge base of the environment within which an agent acts. Different learning methods [1-4] using neural nets have been successfully implemented in pattern classification, function realization problems which have various applications in automated robotic systems. Hopfield neural net [5] was employed by Elfes in his pioneering work [6] for storing environment of a robot. Rumelhart's [7] well-known back-propagation algorithm is invoked by many researchers [8-12] for refinement of knowledge. Hebbian type [5] unsupervised learning has been considered by Pal et. al. [13] for refinement of knowledge where certainty factor of rules are adjusted autonomously for attaining a steady-state value and thus, tuned to better accuracy. Recently Sil et. al. [14] in her work invoked Dempster-Shafer theory [15] for fusing information from multiple case histories and then refine the knowledge using neural nets, realized with Petri nets [16]. Such type of models could be incorporated in a robotic system to build perception about its environment autonomously.

Genetic algorithm (GA) has been successfully used in intelligent search, machine learning, optimization problems and now GA has proved its merit in the field of navigational planning problems of mobile robot where the problem can be formulated as a constraint optimization task. Machalewicz successfully applied GA in navigational planning of mobile robots, as reported in his recent works [17,18].

Multi-agent coordination system recently receives wide attention among researchers [19-21] for its application in customized service fields [22]. In the multi-agent coordination system, a number of agents participate for executing a complex task in a cooperative manner. Asada et. al. in their work [23] discussed the complexity of the environment in the context of multi-agent learning, implemented with the help of a vision-based reinforcement learning method [5].

In the paper, first we propose the decomposition of a complex task into a number of subtasks and define the subgoal corresponding to each subtask using the problem-reduction technique [24] in AI, while dependency among various agents (robots) are available from the problem domain. One to one mapping

0-7695-1062-0/01 $10.00 © 2001 IEEE

between agent and subtask has been resolved, using the given relationship among various agents. In a multi-robot environment, the representation of the environment is much more complex for accomplishing a given task [23],

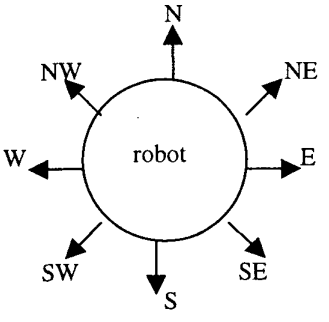

Figure. 1. The schematic view of a robot with 8 ultrasonic sensors

since one robot's behavior can not be predicted by another robot in advance, unless explicit communication is available. The state-space model of the multi-agent system has been designed in the paper using the state-equations of control theory where each state determines the spatial coordinate position of the robot. Next-state is evaluated as a function of the current-state and other system variables, related with the environment and available from the sensory information, at any particular instant of time. Next-state of the robot is computed in 8 different directions, as shown in Fig. 1 and the Eucledian distance between the next-state and the subgoal of a particular subtask (assigned to the robot) is evaluated and stored in a linked list, along with the respective state. The process is repeated for other agents too. For obvious reason, a robot tries to move through some definite states in the environment, so that it can achieve its subgoal travelling minimum distance, compare to other probable states. There may occur collision when more than one agent try to achieve a particular state fulfilling the above criteria. Under such circumstances, at a time, only one agent is allowed to move to that particular state for which the agent will be smallest distant apart from its subgoal state, compare to other agents. For rest of the agents the same procedure is invoked to determine their path in the environment. Thus, individual agent meets its subgoal and as a result the task will be completed autonomously without any prior training imparted to the agents.

The whole scheme is implemented using object oriented programming approach where subtasks are designed as classes and agents are the instances of those classes. Through state-equations one agent can communicate with other agents to perform a complete task that supports the message passing technique between objects.

2. Task Scheduling

Suppose, in a dynamic environment multiple robots are employed to execute a complex task say, T and each robot can navigate in 8 different directions (vide Fig. 1), one at a time. First of all T is broken into T1, T2 .....Tn, i.e. n number of subtasks using the well-known problem reduction technique [24] in AI. Agent to agent dependency equation is supplied from the problem domain, analyzing their knowledge bases. For example, the equation A1 $\rightarrow$ A2, A3 denotes that agent A1 is dependent on agent A2 and A3.

2.1. Terminologies

Before describing the task scheduling algorithm, for the sake of the readers' convenience the following definitions are formalized below.

Def. 2.1: Agent to agent dependency matrix (AADM) P, is an (m x m) dimensional matrix where $p_{ij}=1$, if agent a_j depends on agent a_i; otherwise 0.

Def. 2.2: In the Agent to agent dependency matrix (AADM) P, if $p_{ij}=0$ for all i then agent a_j is called a **free agent of order zero**. A free agent of order zero is employed to perform a subtask, at time t=0.

Def. 2.3: An agent q_j becomes a **free agent of order x-1**, if $q_{ij}=0$ for all i, where $Q=P^x$ (P is named according to Def. 2.2) and x>1.

Def. 2.4: An agent q_i is called a **cyclic agent**, if $q_{ii}=1$, for any value of x>1, where $Q=P^x$ (P is named according to Def. 2.2).

2.2. Task scheduling algorithm

Input: set of cyclic agents, sets of free agents with different order, set of subtasks, number of free agents with different order.

Output: one to one mapping between agent and subtask.

```
Begin
  If there is no free agent of order zero then
report failure.
   Else
     Begin
       k=0 ;
       while (all subtasks are not exhausted) do
        Begin
           For i=1 to (number of free agents of
                       order k)
              Begin
                 (free agent of order k)
                  is assigned to a (particular
                  subtask) ;
                   k=k+1 ;
               End ;
           End ;
      End;
End.
```

3. State-space Model

After mapping an agent to a subtask, the next phase of the work is to design the state-space model of the multi-agent system. Here, the state of an agent is defined by its spatial coordinate position while next-state is determined using state-equations of control theory, vide expression (1) & (2).

$$x_N^A(t+1) = x_N^A(t) + r.\, u^A(t) \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$y_N^A(t+1) = y_N^A(t) + s.\, u^A(t) \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where $x_N^A(t+1)$, $y_N^A(t+1)$ denote (x,y) coordinate of agent **A**, towards north direction **N,** at a time (t+1). $u^A(t)$ represents information obtained by the sensors of agent A, at time t and r,s are the coefficients used to measure the displacement from current-state ($x_N^A(t)$, $y_N^A(t)$) to next-state ($x_N^A(t+1)$, $y_N^A(t+1)$)

Similarly, for every agent, (x,y) coordinate in 8 different directions are computed synchronously.

3.1 Navigational Planning

An agent computes its next-state in different directions, vide expression (1) & (2).

$$d_{min}^1(t) = \min(ed_N^1(t), ed_S^1(t), \ldots.) \quad \ldots\ldots\ldots\ldots (3)$$

where, $ed_N^1(t)=[(x_N^1(t) - x_g^1)^2+(y_N^1(t) - y_g^1)^2]^{1/2}$

$d_{min}^1(t)$ is the minimum distance among $ed_N^1(t)$, $ed_S^1(t)$, ..., that Agent-1 may attain in order to achieve its subgoal and $ed_N^1(t)$, $ed_S^1(t)$ represent Eucledian distance between next-state and subgoal of Agent_1, towards N direction, S direction, respectively.
Actually, for Agent_1 the $ed_N^1(t)$, $ed_S^1(t)$ ..... are stored in an array say, B1 in ascending order. Similarly, Eucledian distance between next-state and subgoal for Agent_2, Agent_3, ... are stored in arrays, B2, B3, ... respectively, for handling collision among the agents.

It is worth mentioning here, that the total number of agents in an environment should not exceed the number of directions a robot can navigate to keep motion of every agent, at an instant of time t.

3.2. Collision Resolution Strategy

Suppose, at a time all agents (Agent_1, Agent_2 and Agent_3, vide Fig.2), in the environment try to attain a particular state (state_1), for achieving their respective subgoals (subgoal_1, subgoal_2, subgoal_3) and hence, collision occurs at that particular state (State_1).

To resolve the crisis only one agent is allowed to move in that particular state using expression (4).

$$D_{min}(t) = \min(d_{min}^1(t), d_{min}^2(t), \ldots d_{min}^8(t)) \ldots\ldots (4)$$

Where $D_{min}(t)$ represents minimum distance among $d_{min}^1(t)$, $d_{min}^2(t)$, ... $d_{min}^8(t)$, i.e. among the first elements of B1, B2 ....B8, respectively. Using expression (4) only a single agent, say Agent_3 has been selected to move state_1 and thereby resolving the crisis.
Other agents try to move towards different directions satisfying condition stated, in expression (3) if no more collision occurs. In case of collision, among rest of the agents (Agent_1 and Agent_2, at state_2), the same

procedure is repeated and another agent (Agent_2), vide expression (4), determines its next-state (state_2) from the linked list, corresponding to the second minimum distance i.e. among the second elements of B2, B3, ... B8, respectively. Thus, all agents responsible for collision can determine their next-state for navigation.

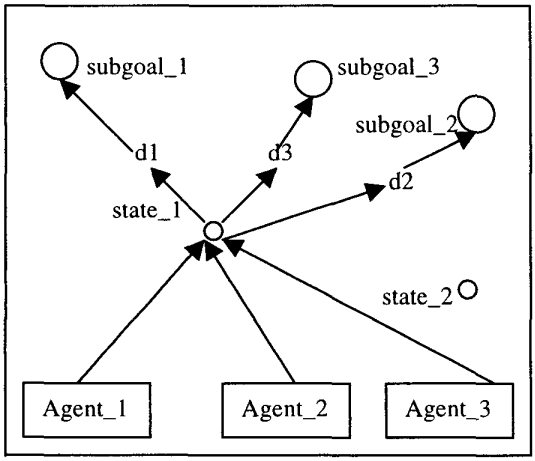

Figure 2. An illustrative example of proposed collision resolution strategy

4. Conclusions

The paper proposes a simple method of mapping an agent to a subtask where execution of each subtask yields completion of the complex task. Next, the proposed state-space model of the multi-robot environment can be represented into a vector-matrix form to facilitate autonomous navigation and execution of tasks while collisions among the robots have been tackled efficiently in the paper.

JAVA has been used to implement the proposed work to coordinate a task by multi-robots in object oriented approach. Here, each subtask has been implemented as a class and agents are instances of classes, called objects. Information obtained through the sensors of individual agent, act as data or properties of the agent and an agent can only have information regarding the next-state of other agents while sensory data are kept hiding supporting one of the main feature of object oriented programming approach. The concept of distributed agent technology is emerged from this work where the researchers have a great scope to concentrate.

References

[1] Meng M., "A neural production system and its application in vision guided mobile robot navigation," Proceedings of IEEE International conference on Neural Networks, San Francisco, Vol. II, 1993, pp. 807-812.

[2] William. R.J., "On the use of back-propagation in associative reinforcement learning", in IEEE Int. Conf. On Neural Networks, NY, Vol.1, 1988, pp. 263-270.

[3] Widrow. B., "Generalization and information storage in networks of ADALINE neurons", in Self-organizing systems, eds. M.C. Yovits, G.T. Jacobi and G.D. Goldstein, 1962, pp. 435-461.

[4] Meng. H. and Picton. P.D., "Neural Network for Local Guidance of Mobile Robots", The Third Int. Conf. On Automation, Robotics and Computer Vision (ICARCV'94), Nov. 1994, Singapore.

[5] Chin-Teng Lin and C.S. George Lee, Neural Fuzzy Systems, Prentice Hall PTR, NJ, 1996, ch. 16, pp. 470-478.

[6] Elfes, A., "Using occupancy grids for mobile robot perception and navigation", IEEE Trans. on Computer, 1989, pp. 46-57.

[7] Rumelhart, D.E., Hilton, G..E. and williams, R..J., "Learning internal representations by error propagation", in D.E. Rumelhart and J.L. McClelland, eds. Parallel data processing,Vol.1, Cambridge, MA: The M.I.T. Press, 1986, pp. 318-362.

[8] J.M. Keller and H. Tahani, " Backpropagation neural networks for fuzzy logic," Inform. Sci. vol. 62, pp 205-221, 1992.

[9] Konar A., Artificial Intelligence and Soft Computing: Behavioral and Cognitive Modeling of the Human Brain, ch. 12, CRC Press, 1999.

[10] S. Datta, M.K. Banerjee and J. Sil, " Recognition of two_dimensional objects using probabilistic predicate transition net," proceedings of IASTED conference on Modeling & Simulation, pp.17, Pittsburgh, USA, May 13-16, 1998.

[11] J.M. Keller and H. Tahani, " Backpropagation neural networks for fuzzy logic," Inform. Sci. vol. 62, pp 205-221, 1992.

[12] Sil. J., Intelligent expert and learning system using Petri net, Ph. D. dissertation , Jadavpur University, 1996

[13] Konar A. and Pal S., " Modeling Cognition with Fuzzy neural nets", in Fuzzy Systems Theory: Techniques and Applications, edited by C.T. Leondes, C.T., NY: Academic Press.

[14] Sil J. and Konar. A., " Knowledge Acquisition and Data Fusion by Neural Petri Net", communicated on August, 2000, to Int. Journal on Expert Systems with Applications.

[15] Shafer G., A mathematical theory of evidence, Princeton University Press, 1976.

[16] Peterson J. L., Petri net theory and modeling of systems, Pretice hall, NJ, ch.2, pp. 7-30, 1981.

[17] Michalewicz Z., Genetic algorithms+Data Structure=Evolution Programs, 3rd. edition, NY: Springer-Verlag, 1986.

[18] Xiao J., Michalewicz Z., Zhang L. and Trojanowski K., "Adaptive evolutionary Planner/ Navigator for Mobile Robots", IEEE Ttrans. On Evolutionary Computation, Vol. 1, No. 1, April 1997.

[19] Liberman H., " Integrating User Interface Agents with Conventional Applications ", http://lieber.www.media.mit.edu/people/lieber/Lieberary/Integating.../Integrating-UI-Agents.htm

[20] Caglayan, A., M. Snorrason, J. Jacoby, J. Mazzu and R. Jones, "Lessons from Open Sesame! A User Interface Learning Agent", Conf. On Practical Applications of Agents and Multi-Agent Systems, [PAAM-96], London, 1996.

[21] Liberman H., and Mondrain: A Teachable Graphical Editor, in [Cypher, ed. 93].

[22] Gaxiola D. and Tatlin: Integrating Commercial Applications into Programming by Demonstration, MIT BS Thesis, 1995.

[23] Asada M., Uchibe Eiji and Hosoda Koh, "Cooperative behavior acquisition for mobile robots in dynamically changing real worlds via vision-based reinforcement learning and development," Artificial Intelligence, pp. 275-292, 1999.

[24] Jackson P., Introduction to Expert systems, Addison-Wesley, 1988.