A Kind of Adaptive Program Design Method

Dong Li

***Abstract*—The current computer program is artificially designed to execute the process，which has the weakness of passivity，rigidity and lack of flexibility. This paper proposes a method of program design based on reinforcement learning mechanism ，and realizes the corresponding algorithm. According to the environment and requirements，the agent can choose executive process independently and arrive at the optimal result by learning，realize the layered calls. Using this method，the executing program is decision-making，has a way to realize the adaption ，and reduces the dependence on designer. The result shows that the method can achieve satisfactory execution efficiency.**

***Index Terms*—Adaptive programming, reinforcement learning, q-learning, agent, optimization algorithm.**

I. Introduction

Computer program has experienced the change from the process oriented to the object oriented. But whether process oriented, or object oriented, and its design process is according to project requirement to realize the code, which has the passivity; once the operation environment changes, the designed program must manually transform, and the program can't adapt the demands of environment, which has the rigidity; If program flow changes, the programmer needs to keep the program corresponding adjustment and which lacks of flexibility. Along with the computer software function becomes more and more powerful and abundant, software maintenance cost is also more and more high, which makes the adaptability of new business worse and worse, and becomes the main obstacles to the popularization of software.

In this case, objectively computer software is required to have certain adaptability, has the ability to learn and correct errors by itself to adapt to the change of new environment. This kind of adaptive program can reduce the load to do a lot of code for programmers, computer program through continuous autonomous learning adapts to the different operating environment, finally obtain the ability of producing correct results. Adaptive software [1] is aiming to the needs of users and different environment to adjust its behavior, to meet different environment and achieve the design goal.

The research work of adaptive software carries out in a short period, Advanced Research Projects Agency(ARPA) of American Department of Defense from 1998 took adaptive software into the study plan, began to aid some research institutions engaging in this kind of work [2]. Then adaptive software also gradually became a research hotspot, the representative is the adaptive software international conference sponsored by artificial intelligence laboratory in MIT, the researchers tried to adopt all kinds of method to solve the problem. This aspect of domestic research just started, is mostly used in component technology. This article is based on the idea of strengthened learning, puts forward an adaptive program design method with agent oriented, according to the requirements of programmer, the program can make an autonomous decision, and produces the behavior to adapt to the requirements of environment, to reduce dependence on the designer.

Manuscript received March 11, 2012; revised April 13, 2012.

D. Li is from college of Computer and Information Engineering, inxiang University, inxiang, China

II. Intensified Learning Mechanism

Intensified learning is an important branch of machine learning; its thought comes from psychology and biology. Different with supervised learning and unsupervised learning, Intensified learning emphasizes on interactive study in environment, through appraisive feedback signals obtained in the learning process as return, in the learning process, its target is maximizing its accumulation return and selecting the best behavior.

Most of the identified learning researches are based on the theory framework of Markov Decision Process (MDP) [3-4], MDP consists of following four parts: status set S, action set A, reward value function $R:S\times\to R$, status transfer function $T:S\times\to\prod(S)$. In each time step, agent gets environment condition $s_i\in S$, and determines the next action $a_i\in A_{s_t}$, the environment according to the action a_i , in next time step feedbacks to the agent a reward value r_{r+1} , agent comes into a new environment condition s' . To each given (s,a) by groups, there is reward value r_s^a , at same time which transfers to the new environment s', transition probability is $p_{ss'}^a$. In order to get the optimal behavior, agent goal is in every discrete state s, seeking $Q^\pi(s,a)$ as the largest, hereinto the definition of $Q^\pi(s,a)$ is as follows:

$$Q^\pi(s,a)=E\{r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\cdots|s_t|=s,a_t=s,\pi\}$$
$$0<\gamma<1 \tag{1}$$

Q value corresponding to state behavior can be obtained:

$$Q^*(s,a)=R(s,a)+\gamma\sum_{s'}\left[p_{ss'}^a\max_{a'}Q^*(s',a')\right] \tag{2}$$

If environment model and transition probability are known, the above problems are reduced to the dynamic planning.

However, in the actual application, environment model and transition probability is often unknown; aiming to this problem Watins [5] put forward Q-learning meanwhile to prove its convergence. It is expressed as:

$$Q(s_{t+1},a_{t+1}) = Q(s_t,a_t) + \alpha\left(r + \gamma \max_a (s',a') - Q(s,a)\right) \quad 0<\gamma<1 \tag{3}$$

After each iteration agent updates $Q(s,a)$ value, after many times of iteration $Q(s,a)$ value convergences.

Via above definition about Q value, it is easy through strategy iteration to get the optimal strategy π^*, which is shown as:

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

III. Adaptive Program Design Method

A. Basic Ideas of Adaptive Program Design

A program is to perform a particular sequence of statements, according to the condition establishment or not to do jumping control; this control is often designed artificially. And adaptive program should have autonomy, can independently choose the execution flow [6]. The method in the paper is based on this idea, adopts the thought of intensified learning, according to the reward value from study to adaptively choose the optimum behavior. In traditional program design method, the environment is very simple, often the programmers can foresee and control, but the adaptive program environmental factors are often unknown, so it is more complex. Traditional program design is based on the function and manually written the code, and in adaptive program, agents replace this and the behavior is more intelligent, their behavior comes into being through continuous training, according to the changes of environment which is better to adapt to the environment with more strong generality. Traditional programming method is based on the logic reasoning, and adaptive program is based on the probability and has better fault tolerance and fuzziness.

In this paper, a programming model with agent is proposed as executive main body. Suppose state set is S, executive statement set is A, each execution of the statements can be regarded as one action of agent. When agent executes one step, environmental state can be felt $s_i \in S$, that is, the condition fulfills the conditions, if in this environment there is an optional statement set $\{statement_1, statement_2, \cdots, statement_n\}$, agent selects and carries out the statement according to executive strategy $statement_i \in A_{s_t}$, after the execution of this statement, environment feedbacks to the agent a reward value r_i, and going into the next state s' until the running program reaches a goal or overtime point. If program executes in the best process, the environment will give a larger reward; conversely, if a program appears error, environment will give a proper penalty. The aim for the agent is in each discrete states s, seeking the maximum of $Q^\pi(s,statement)$.

$$Q^*(s,statement) = \max_a Q^\pi(s,statement) \tag{5}$$

After multiple iterations, Q value reaches convergence, based on expression (4), the optimal strategy can be obtained, and this is the optimal program execution process. Applying Q study method, the algorithm can be illustrated as follows.

B. Implementation of Adaptive Programming Algorithm

Input parameter's values of α, γ *;*

Initialization;

While (do not meet termination conditions)

{Reach to the current state s;

Under current state s the executive set of statements are $\{statement_1, statement_2, \cdots, statement_n\}$ *, to choose statement* $statement_i$ *in some strategy (such as:* $\varepsilon - greedy$ *) according to Q value;*

Execute statement $statement_i$ *to obtain the reward value* r_t *, and come into the next state* s' *;*

Select study rate α *according to the convergence;*

Update Q value

$$Q(s, statement_i) = (1-\alpha)Q(s, statement_i) + \alpha\left[r + \gamma \max_{statement'_i} Q(s', statement'_i)\right];$$

IV. Layered Intensified Study and Modular Adaptive Program

The above method provides an adaptive programming design framework. Traditional program design adopts the method of divide and rule, the big problem is divided into small problems, and realizes the calls between modules, meanwhile modules can also be reused. In this paper the adaptive programming method still adopts the modular design. In modular design, the upper modules use the method of adaptive learning to choose the lower optimum modules to call, to further improve the performance. As shown in figure 2, in modular adaptive program, the modules invoked by module M are $M_0, M_1, \cdots, M_i, \cdots, M_m$, the modules invoked by submodular M_i are $M_{i_0}, M_{i_1}, \cdots, M_{i_j}, \cdots, M_{i_n}$, the behaviour sequences of bottom module M_{i_j} is abstracted as a Option, by intensified study to get the optimal strategy, as the result of submodular's learning outcomes.

All other bottom modules can through the study get the correct program execution process. This thought can be expanded to its upper layer of modules, for example: suppose module M selects the best module M_i from $M_0, M_1, \cdots, M_i, \cdots, M_m$ to execute, also using the intensified learning method introduced in this paper, the behavior of M_i is abstracted as an Option to learn. Father module can get its implementation process like the

submodular through the intensified learning, in constant learning process to get optimization to comply with the solution of the problem. That is, the submodular is learning, father modules also can learn. Of course after the study task is decomposed, each module including the father and son modules can concurrently or parallelly perform other learning process. In addition, using this method makes the optimized modules strategy can be called by more top modules, so as to realize the strategy reuse.

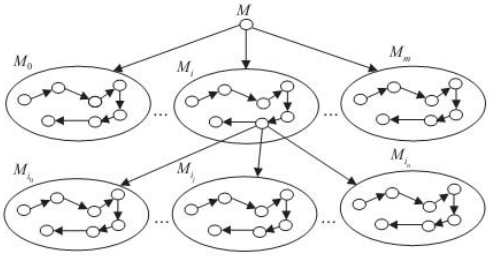

Fig. 1. The modular calls of adaptive programs.

Sutton [7], Dietterich[8] and Parr[9]put forward three different layer of intensified study method for modular adaptive design program to provide theory basis. Layer of intensive study [10] increases the abstract mechanism based on the intensive learning, the task is decomposed several different levels of subtasks, and every subtask can be solved in problem space with small scale, at the same time the strategy got from the sub space can execute software reuse.

Study tasks can be abstracted as several Options [7], every Option is a basic action sequences. Each behavior is a simple basic action, also is another Option, through the call of upper layer Option to the lower Option or the basic action call, layered control structure can form. In the hierarchical intensified learning system these Options as a kind of special "action" join the original action set. For any Option o, Using $E(o,s,t)$ represents events. AT moment t the reward value of $R(s,o)$ and the status transition probability $P(s'|s,o)$ to be defined as follows:

$$R(s,o)=E\left[r_t+\gamma r_{r+1}+\cdots+\gamma^{\tau-1}r_{t+\tau-1}\middle|E(o,s,t)\right] \tag{6}$$

$$P(s'|s,o)=\sum_{\tau=1}^{\infty}p(s',\tau)\gamma^{\tau} \tag{7}$$

The optimized Q value formula can be expressed as:

$$Q_o^*(s,o)=R(s,o)+\sum_{s'}P(s'|s,o)\max_{o'\in O_{s'}}Q_o^*(s',o') \tag{8}$$

If the environment model and the transition probability are unknown, the Q learning update formula based on Option is:

$$Q(s_t,a_t)=Q(s_t,a_t)+\alpha\left[r+\gamma\max_a(s',a')-Q(s,a)\right] \tag{9}$$

Precup [11] proves in standard Q learning convergence conditions, the Q learning algorithm based on Option is from probability 1 convergence to Q_o^* .Utilizing the above theory, the set of statements executed by programs can be divided into every fixed program sequence set to reduce the times of system's decision, making the learning scale greatly reduced, at the same time, the modularization can be realized.

V. Experimental Results and Analysis

The learning efficiency is always the concern in this paper, there is the experiment aiming to the problems in figure 1, under this premise without any prior knowledge, study entities is provided the necessary Input parameters and have no any information about how to execute program, the program executive process depends on the strategy results of learning. In the initial stage of learning, in each round learning step concussion is more severe, after about 120 rounds of learning can achieve convergence, starting from source point S to target point G the step number is 14, search strategies achieve the optimal, adaptive results are correct. The learning process curve is shown in Fig. 2 below.

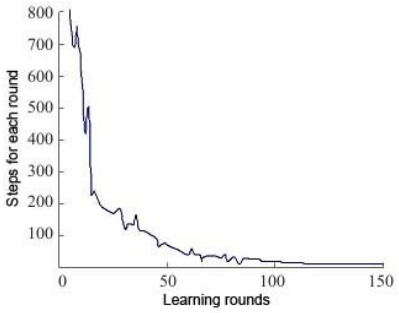

Fig. 2: Learning process curve

VI. Epilogues

This paper puts forward a kind of adaptive program method, without any specific program execution process, which only submits environment and conditions met by the program, and the expected results to the process entities, executive entity will automatically search the optimal process. It is found through experiments that the entity learning process is acceptable. Meanwhile, aiming to complex problems, layered calling method still can be used to reduce the complexity and improve the learning efficiency. This method can use parallel computing or distributed computing method to make learning efficiency more increased. This method proposed in this study is versatile, can be applied to many fields, such as routing problem, resource allocation, searching, web combination of services, data mining, and robots and network topology optimization, all can use the above methods to get good results.

References

[1] P. Nerving, Chon D. *Adaptive Software* [EB/OL]. http：//www.norvig.com/adapaper-pcai.html.

[2] Q. X. Wang, J. R. Shen, and H. Mei. *Adaptive Software Research*. Computer Science, vol. 31, no. 10, pp.168-171, 2004.

[3] L. P. Kaelbling, M. L. Littman, and A. W, "Moore, Reinforcement Learning：a Survey," *Journal of Artelligence Research*, pp. 237-285,1996.

[4] C. Su, Y. Gao, and S. F. Chen, etc. "Research of Options Algorithm Generated Autonomously Based on SMDP Environment," *Pattern Recognition and Artificial Intelligence*, pp. 679-683, 2005.

[5] P. Watins and N. Dayan, Q-Learning. *Machine Learning,* pp. 279-292,1992.

[6] C. Simpkins, S. Bhat, and M. Mateas, "Towards Adaptive Programming Integrating Reinforcement Learning into a Programming Language,"*ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages ,and Applications*, Nashville ,TN,USA, pp. 19-23, 2008.

[7] R. S. Sutton, D. Precup, and S. P. Singh, "Between MDPs and Semi-MDPs ： a Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, pp. 181-211, 1999.

[8] T. G. Dietterich, "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *Journal of Artificial Intelligence Research*, pp. 227-303, 2003.

[9] R. Parr, "Hierachical Control and Learning for Markov Decision Processes. Berkeley," USA：University of California,1998.

[10] J. Shen. *Layered Intensified Learning Theory and Method*. Harbin: Harbin Engineering University Press, 2007.

[11] Precup D. *Temporal Abstraction in Reinforcement Learning*. Amherst, USA：University of Massachusetts,2000.