Basic Research on Speed-Up of Reinforcement Learning Using Parallel Processing for Combination Value Function

Tsuguhisa Touma\textsuperscript{a}, Yuuki Nakama\textsuperscript{b}, Koji Yamada\textsuperscript{a}, Satoshi Endo\textsuperscript{a}

\textsuperscript{a}University of Ryukyus, Okinawa, Japan
\textsuperscript{b}Kumamoto University, Kumamoto, Japan

Abstract

In this paper we use parallel processing to combine value functions in order to speedup reinforcement learning. We propose an asynchronous method of periodically composing Q table of local learning clusters to form global Q table. In this research, two approaches are implemented. First is discontinuance learning. Second is combination of value function by asynchronous communication. The asynchronous combination method is compared with a synchronous combination method in order of learning times. A cluster of 40 PCs were used in the experiments are presented. The convergence time and learning times are evaluated and discussed.

© 2011 Published by Elsevier B.V. Open access under CC BY-NC-ND license.

Keywords: Reinforcement Learning, Parallel Processing, Multi-Agents System;

1. Preface

The reinforcement learning is an algorithm to acquire appropriate behavior through interaction with the environment. In reinforcement learning, a robot learns the control of the robot by trial and error. At the beginning it acts randomly, however gradually it improves behavior from repeated success and failures. One of the shortcomings of reinforcement learning is that the convergence of value function and learning time can be computationally prohibitive for very complex systems, such as robots. In this study we applied reinforcement learning using parallel processing and combined value function\cite{1}\cite{2} to the task of controlling motion of a brittle star type robot.

2. About a brittle star type robot

We modeled the flexible joint of the brittle star fish by a module\cite{3}\cite{4}\cite{5} The leg of the robot is made by connecting the modules. The robot moves by applying torque to the module of the leg. We developed a simulation model of the robot using ODE (Fig.1).
3. Reinforcement learning

The reinforcement learning is an algorithm [6][7][8] to which can be used to teach optimal control to a robot. Learning takes place by means of a value function expressed in state space and action space, which is updated by repeated trials resulting in success and failures. Successful outcomes are rewarded, whereas failures are penalized.

3.1. Problems of the reinforcement learning

There are three problems of the reinforcement learning by the robot control. Firstly, the state space is exponentially large. Second is a long learning trial. Third is a simultaneous learning problems. We approach these problems using parallel processing to improve the efficiency of the learning process.

3.2. Parallel of the reinforcement learning

The purpose of parallel processing is 2-folds. First is to reduce the number of iterations to reach convergence towards learning.[9][10][11] Second is to shorten learning time.[12] We were able to reduce the experiences by combining result that were learned in parallel, but we could not decrease learning time, because the combination of a value function had to be synchronized. Therefore, in this study we shortened learning time by combining value function in asynchronous fashion.

4. Synchronizing combination value function in parallel

In the method for the combination of value function in parallel [13][14], the independent value function of each agent is updated in parallel by PC-cluster. A new value function is made by combining the value function of each learning result synchronously. Furthermore, the combination value function is handed to each agent, and they restart learning. As a result, learning efficiency is improved by this method. The flow of the method is shown below.

1. Each agent begins learning.
2. Each agent learns to specified number of trials.
3. The value function is combined.
4. The combination value function is handed to each agent.
5. The above steps are repeated until termination that specified iterations.

The value function $Q(s,a)_i$ is updated by the independent value function of each agent (number of agents = i ). The number of the update of the value function is stored by $C(s,a)$. Each value function $Q(s,a)_i$ is combined by equation (1).

$$Q'(s,a) = \frac{\sum_{i=0}^{n-1} Q(s,a)_i \times C(s,a)_i}{\sum_{i=0}^{n-1} C(s,a)_i} \quad (1)$$
4.1. Assessment task

The state space and problem space in brittle star type robot are architected is shown in Fig.2. The Action space is expressed by dividing the range of the servomotor into 16 states. The reward is distance moved in the positive x-axis. The learning for movement acquisition is performed by ODE [15].

![Fig.2. State of each module](image)

4.2. Evaluation result

Fig.3 and Fig.4 are learning curves for 10000 iterations by synchronous combination of value function at every 1000 iterations. The horizontal axis is the number of the learning iterations (one learning to repeat four times of five motions). The vertical axis is movement distance (distance is achieved by one learning). The change of the value is shown in the learning curve by combination every 1,000 times. The learning result is shown by Table.1. The learning result in parallel is better than the learning result for a single agent. However, learning time is increases as the number of agents increase.

<table>
<thead>
<tr>
<th>Agents</th>
<th>Distance</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>42.61</td>
<td>105450</td>
</tr>
<tr>
<td>20</td>
<td>80.22</td>
<td>120156</td>
</tr>
<tr>
<td>40</td>
<td>86.03</td>
<td>153757</td>
</tr>
</tbody>
</table>

![Fig 3. 20 Agents (SYNC)](image)  ![Fig 4. 40 Agents (SYNC)](image)
5. Method of combination value function in the asynchronous

In this research, two approaches are implemented for speedup. First approach is to discontinue learning when the estimated learning result is expected to worsen. In fact, one learning to repeat four times of five motions is terminated if direction of movement is towards the negative x-axis. Because, we thought that it is not optimized solution even if moved to minus direction and then moved to plus direction. As a result, learning time of each agent is varied for learning time of some agents decreased. Hence, queuing time is generated when combine value function. Therefore we plan to further speedup by combining value function asynchronously (Fig.5). The value function is combined by equation (1).

\[
\text{Value function} = \text{Value function}_1 + \text{Value function}_2
\]

Fig.5. Asynchronous Combine

6. Evaluation result and consideration

Computer experiments on 40 PC-clusters are presented. The spec of the PC-clusters is shown on Table. 2.

<table>
<thead>
<tr>
<th>Table. 2. Spec of Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
</tr>
<tr>
<td><strong>Compiler</strong></td>
</tr>
</tbody>
</table>

6.1. Learning time

Fig.6 compares the learning time of asynchronous to synchronous. In asynchronous approach, learning time decreases compared to synchronous approach. It is an effective approach to the speedup of learning to combining value function by asynchronously.

Fig. 6. Comparison of time in synchronous and asynchronous
6.2. Learning curve

Fig. 7 and Fig. 8 are learning curves that they are learning to 10000 times by asynchronous combination every 1000 times. Fig. 8 is the same as synchronous combination, and altered value by combination every 1000 times. On the other hand, Fig. 9 does not have change of value, but movement distance increases.

6.3. About of combination value function

In synchronous combination of value function, all agents wait for the end of the learning. Thus, the same combination value function is handed to each agent. However, the combination value function handed to each agent are different when value function is combined asynchronously. This is because, the value function of an agent updates the combination value function without waiting for the learning result of all agents. The last agent to finish learning has the final result of value function.

6.4. Convergence time

Fig. 9 and Fig. 10 are learning curves of the last agent after learning had finished. The distance increases with two curves, but does not converge. This can be attributed to small number of trials as well as discontinuity in learning.
6.5. Movement distance

Movement distance acquired by 10,000 times of trials is shown in Table 3. They are better result than single agent.

Table 3. Distance by asynchronous

<table>
<thead>
<tr>
<th>Agents</th>
<th>20</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance</td>
<td>91.59</td>
<td>51.72</td>
</tr>
</tbody>
</table>

7. Conclusion

In this study, we proposed a method of combining value function asynchronously to speedup learning by parallel processing. We discussed effectiveness of the method through learning of the brittle star type robot to move direct distance. As a result, learning time in async is shorter than learning time in sync. Also the learning result did not converge, but movement distance increases. Therefore we could show effectiveness of reinforcement learning in parallel because parallel learning produces better results compared to a single agent. For future research, we plan on dealing with issues on convergence time as well as discontinuity in learning.

References

7. Michie D. and Chambers R A. "BOXES: An Experiment in Adaptive Control", in Dale E. and Michie E. Machine Intelligence 2, pp.137-152, Oliver and Boyd (1968)