ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2008-7493

INFORMATION TECHNOLOGY AND MANAGEMENT SCIENCE
INFORMĀCIJAS TEHNOLOĢIJA UN VADĪBAS ZINĀTNE

AMBIGUOUS STATES DETERMINATION IN NON-MARKOVIAN ENVIRONMENTS

Jurij Chizhov, Mg.sc.ing., Ph.D. student, Department of Modelling and Simulation, Riga Technical University 1 Kalku Street, Riga LV-1658, Latvia, e-mail: Jurijs.Cizovs@rtu.lv.

Tatyana Zmanovska, Mg.sc.ing., Department of Modelling and Simulation, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: zm@itl.rtu.lv.

Arkady Borisov, Dr.habil.sc.comp., Professor, Department of Modelling and Simulation, Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia, e-mail: Arkadijs.Borisovs@cs.rtu.lv.

Keywords: reinforcement learning, non-Markovian deterministic environments, intelligent agent control

1. Introduction

One of the most topical tasks of artificial intelligence is searching for optimal policy of interaction with an environment by autonomous intelligence software agent. Classical methods of reinforcement learning are performing successfully in the so-called Markovian environments. In this work the idea and implementation approach are stated for non-Markovian environments. The approach offered represents a way of ambiguous states detection. The paper includes definition and detailed scrutiny of ambiguous states forms. The detection is the prerequisite phase for building optimal policy using internal representation of states. In its turn, the internal representation preserves properties and advantages of the classical algorithm of reinforcement learning and its condition of convergence is important.

The central part of the paper is focused on the indication of ambiguous cells and its detection. As shown in the paper, it is rather difficult to recognize different “copies” of the same states. The paper contains a theoretical introduction, ideas and problem description, and, finally, an illustration of results and conclusions.

2. Agent task and environment

The experiments are carried out in a well-known task - searching for a path in labyrinth (kind of agent control), in other words, building an optimal policy (strategy) of actions by exploration of the environment. The reason for our choice is obviousness and simple understanding of the received results. SARSA(λ) learning algorithm [1] is preferred as a base learning algorithm, which will be combined with the investigated algorithms.

Let's consider the static cellular world - a labyrinth. Each cell of the labyrinth is either free for agent pacing, or occupied by some static obstacle. One of the cells contains food (sometimes called target cell or goal cell). In our case, the goal cell is labelled by letter G. An example of simple grid world is depicted in Figure 1.

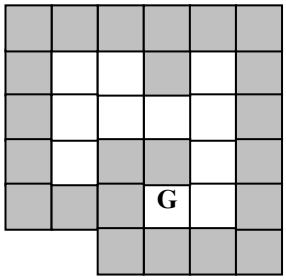


Figure 1. Example of simple grid world

To make the grid world usable like the environment, we must define a set of data which represent the state of the agent. Usually the state is all the information available to an agent that is provided by its sensors. The set of agent sensors (according to available information) depends on task conditions. In our case the agent possesses only four sensors: s_S, s_N, s_W, s_E, each of them returns value 1, if the obstacle is detected in a neighbour cell, in the corresponding direction, otherwise value 0. Thus, agent state might be expressed in binary form, where each bit corresponds to each sensor. For example, if the obstacle is north and east directions, then state in binary form will be equal to $S = 0101_2$. For further calculation convenience, it is better to convert the binary value to decimal form: $S = 1010_2 = 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 5_{10}$. Thus, on each step by getting values of sensors the agent can compute the current state by equation (1):

$$S = 8 \cdot s_{south} + 4 \cdot s_{east} + 2 \cdot s_{west} + s_{north} \quad (1)$$

The value obtained is the state of the agent. Only 16 different states are available in our task (including zero state interpreted as the absence of obstacle around). Figure 2 depicts the grid world with calculated values for each cell (state).

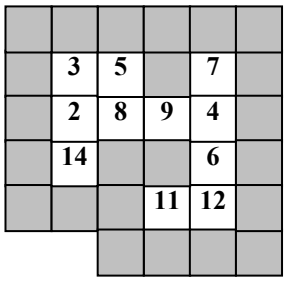


Figure 2. Grid world with evaluated states for each cell

The agent possesses four actions: to step only one cell in each of four directions. It is important to point out that each action of the agent is surely executed. In other words, if in a previous state *s* action *a* was applied, the probability of obtaining next state *s'* is equal to one: P(s, a, s') = 1. If we wish to use a probabilistic model for defining probabilities of transitions, it requires including additional 3-dimensional table.

Having a set of state, actions and probability of transition, the grid-world-like environment now might be represented in transition graph form which is often used for representing the Markovian processes [2]:

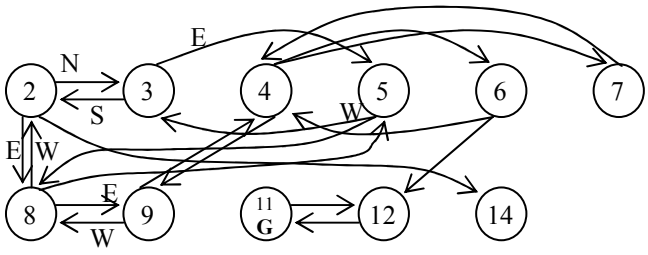


Figure 3. Agent environment in the graph form (states and transitions form)

In the depicted graph, arrows denote possible transitions through implementing corresponding actions (moving north, east, south and west). It is important to point out that the grid world form is the form which is closer to the real world form and keeps more properties than a graph one. The graph form is a model which only keeps information sufficient enough to the agent. For example, looking at the grid world we can see why a transition from one cell to another is available. In graph mode we have only the fact of available transitions. The reason might be interpreted as additional information available for operation. In our case the essential difference between two forms is the properties that belong to the nature of cell. Additional existence and the number of non-Markovian states depend on the precision of external world reproduction. Moreover, the number of non-Markovian states might be reduced by involving eight sensors instead of the existing four. So, the graph representation is less attractive than other forms; nevertheless it is worth considering in detail because of its representation of state-action model, which is used in reinforcement learning algorithms. In Table 1 the main differences between grid-world form and graph form are given.

Table 1

Differences between two forms of environment representation

Grid world form	**Graph form**
• The coordinate system in the labyrinth is caused by the conception of cell	• There is no conception of a cell, thus the absence coordinates takes place
• Transitions move the agent through cells	• Transitions move the agent from one state to another
• The obstacles and cells arrangement describe the actions execution ability	• There are no reasons that could describe the ability of executing actions
• Equally valued state cells are different copies of the same state	• There is no conception of the same state copies, instead of that the non-Markovian conception or inconstancy of transition takes place.

Two last distinctions cause the greatest interest. These distinctions raise three fundamental problems in the task of agent control in non-Markovian environments:

1) detection of ambiguous states;
2) detection of states with inconstant transition;
3) agent learning and its ability to distinct different copies of the same states.

The paper considers the ambiguous states detection problems for situated agent (according to state-action concept [3]).

3. The sensor signals interpreting problem

As was mentioned above, in task of building optimal policy through exploring non-Markovian environment three problem cases might be revealed.

Case 1. The case supposes that from some state at different moments of time it is necessary to execute different actions. In other words, the current state does not fully define the next action [2, 5]. This case is called non-Markovian state. The *Woods101* environment is a classical example of such case [4] (see Figure 4).

The cells denoted by *a* and *b* correspond to the same state because of their equal values evaluated upon agent sensors signals. Having appeared in that state the agent sometimes is compelled to move east, but sometimes west. Thus, having only current state's information, the agent is not capable of making a decision and defining the optimal action.

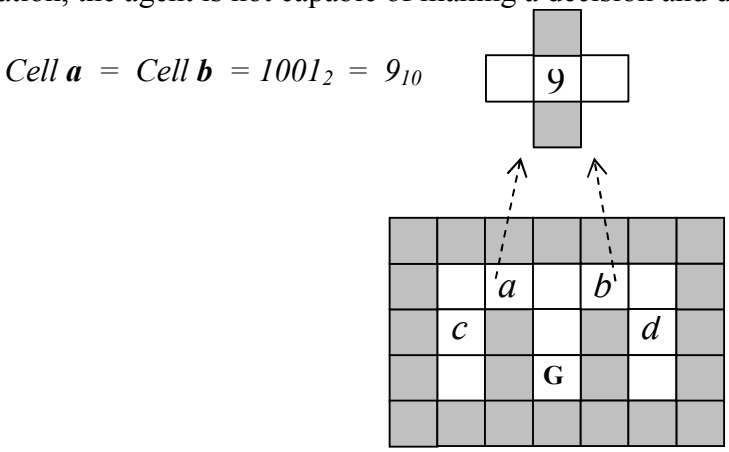


Figure 4. Non-Markovian environment Woods101 (bottom) and example of ambiguous state

Case 2 is the evolution of the previous case and consists in taking the same action from the same state at different times, and the different reaction of environment is observed. Let's call this case the transition inconstancy .Let's see cells *c* and *d,* for example. Again, each of them represents the same state. An attempt to move north will lead to different future states (see Figure 5).

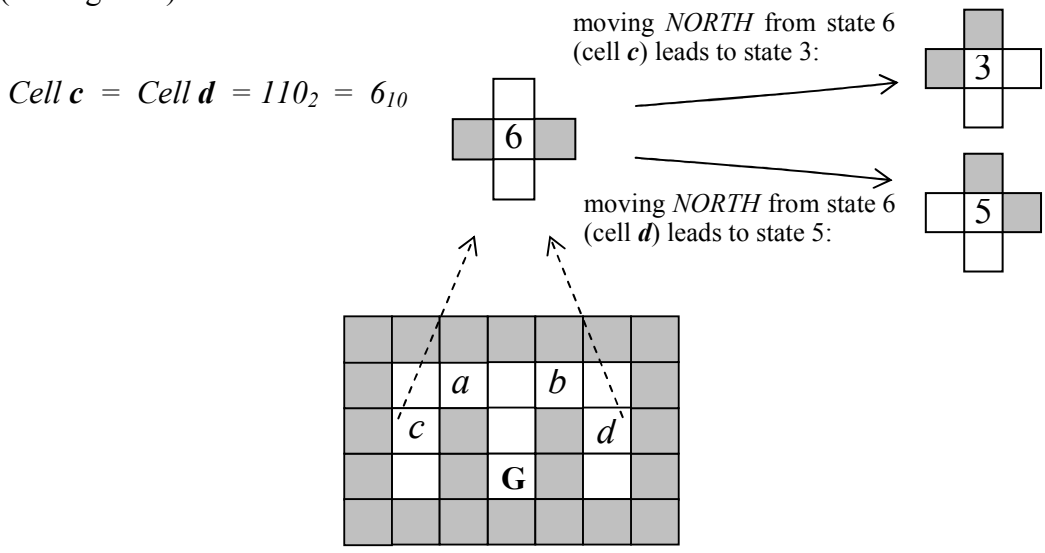


Figure 5. Example of inconstant transition from state 6

Case 3 relates to the problem of interpreting the return of the same state (equal to source state). Let's consider state «9» depicted in Figure 6 (left). While the agent is trying to move north, it meets the obstacle, so, the environment returns the agent to the source (previous) state, which equals «9». Thus, two ways of interpreting the situation are appropriate: 1) the agent was returned to the source state (see Figure 6, on the right), or 2) the agent was moved to another copy of the same state (like moving west or east, see Figure 6, on the left). Different interpretations of states are possible: either we do not take sensors nature into account or instead of sensors methodology the special channel for already evaluated state transmission is used. If the agent "understands" the sensors meaning, it might be used as additional signs. These signs might be sufficient to distinguish different copies of the state. The model and agent-environment conditions interaction are defined by the task.

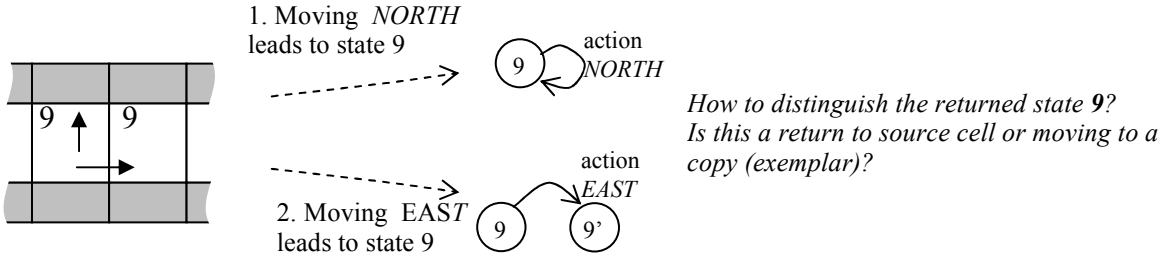


Figure 6. Two ways of interpreting the state «9»

The problem of distinguishing two equal states is connected to that of representing same states in a graph form or Markov chains. Should we duplicate state «9» or leave it unique but having multiple connections to neighbours? In its turn, there should be «return links» which are responsible for agent returning in source state in case of bumping to obstacle.

3.1. The indication of ambiguous state and algorithm for its detection

Non-Markovian states and states with inconstant transition due to their common nature have common simple indication: if observations of transition from state *s* by action *a* moves agent to different target states then source state *s* is ambiguous. A simple example is shown in Figure 7.

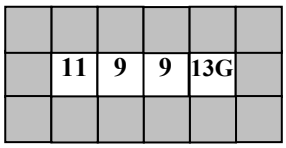


Figure 7. A simple example with ambiguous state «9»

During environment exploration, the agent finds out that applying of action «step east» being in state «11» always moves him to state «9». In its turn, applying action «step east» being in state «9» sometimes moves it to state «9» and sometimes to state «13». Such an uncertainty makes building the Q-table difficult. The formal indication of ambiguous state might be expressed as follows (2):

$$(s_i, a_j)^t : s^{t+1} \neq (s_i, a_j)^{t'} : s^{t'+1} \quad (2)$$

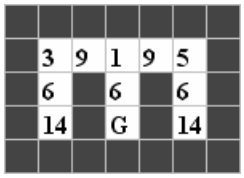
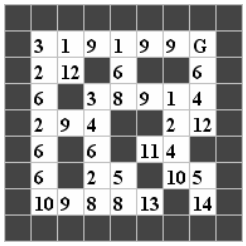
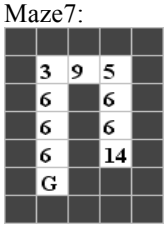
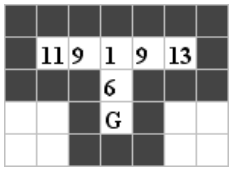
where $(s_i, a_j)^t$ and $(s_i, a_j)^{t'}$ are same corteges observed at different moments of time, S^{t+1} and $S^{t'+1}$ - states returned by the environment (through sensors) at next time tick.

Ambiguous states detection is executed within the framework of Q-learning cycles and requires only additional table size SxA for keeping observable transitions. Each cell [s, a] keeps target state *s'*. As soon as next observation brings different target state, the source state *s* will be marked as ambiguous.

It is important to point out, that the indication does not require knowing of the goal state. Ambiguous states detection occurs while Q-learning builds its policy and does not require special exploration steps of the agent. For experiments, the algorithm was executed on a set of MacCallum's mazes and other environments. A short analyzing log for each environment is given in Table 2. For the environment depicted in Figure 2 no ambiguous states were detected. This result is true.

Table 2

Found ambiguous states

Maze	Found ambiguous states (state:action)
Woods101:  	(6 :↑) (9: →) (9: ←)
Maze5:  	(1:→); (1:↓); (1:←); (2:→); (2:↓); (2:↑); (3:→); (3:↓); (4:←); (4:↓); (4: ↑); (5:←); (5:↓); (6: ↑); (6:↓); (8:→); (8:↑); (8:←); (9:→); (9:←); (10:→); (10:↑); (12:↑);
Maze7:  	(6:↑); (6:↓);
MazeT:  	(9:→); (9:←);

```
Initialize Q(s,a) arbitrarily and
   for all s,a: e(s,a) = 0, L[s,a] = -1
Repeat (for each  episode):
   Initialize s, a
   Repeat (for each step of episode):
      Take action a, observe r, s´
      CheckTransition(s,a,s')
      Choose a´ from s´ using e-greedy
      δ ← r + γQ(s´,a´) – Q(s,a)
      e(s,a) ← e(s,a) + 1
      for all s,a :
         Q(s,a) ← Q(s,a) + α δ e(s,a)
         e(s,a) ← γ λ e(s,a)
         s ← s'; a ← a´
   until s is terminal
```

```
function CheckTransition(s, a, s')

  if L(s,a) = -2 then // skip if nonM
    exit
  if L(s,a) = -1 then
    L(s,a) ← s'
    exit
  if L(s,a) <> next_s then
    L(s,a) ← -2

EndOfFunction
```

Figure 8. SARSA(λ) upgraded up to ambiguous states detection

In the listed algorithm (see Figure 8) only detection of ambiguous states occurs. The bolded fragments are modifications of original SARSA(λ) algorithm.

In task of ambiguous state identification, the *L* array keeps state-action history of deep one. Initially each action *a* in state *s* is marked by -1. In the course of exploration, each next-state *s'* is stored in *L*-table for appropriate state *s* and action *a*. If a new value of next-state *s'* differs from the existing one, *L(s,a)* is marked by value -2. Since *L*-table is of the same size and structure like *Q*, it might be integrated into *Q*-table's structure. Implementing task of internal state representation the *L*-table might be upgraded by additional signs and features, so it could be declared separately.

The modifications do not affect the convergence, but take more memory size (like additional Q-table). After executing, the gained *L*-table keeps value -2 for all ambiguous state-action pairs. All three forms are included.

4. Conclusions

The proposed algorithm is the first look at non-Markovian environment in terms of forms of ambiguous states. The proposed algorithm is integrated into a host one and is able to recognize ambiguous states while a host algorithm is running. It easily copes even with difficult (i.e. several number of copies of the same states series are exist) mazes like Maze5.

Three forms and indication of ambiguous states were formulated. The algorithm for ambiguous states detection is provided and described. The formulated ideas were successfully implemented in Borland Delphi environment. The results of developed software are given in Table 2.

The proposed algorithm still does not cope with the problem of dynamically changed goals. This problem is one of the significant disadvantages of reinforcement learning [6].

The future research will be focused on using the gained result for optimal policy building in non-Markovian environment by internal representation of states. To solve this problem, the *L*-table must be provided with additional features and supplied with corresponding algorithms.

References

1. Sutton, R., Barto, R., 1998. Reinforcement learning. An Introduction // Cambridge, MA: MIT Press.
2. Russell, S., Norvig, R. 2003, Artificial Intelligence: A Modern Approach // Prentice Hall. New Jersey, 2nd edition.
3. Padgham, L., Winikoff, M., 2004. Developing Intelligent Agent Systems. A practical guide // John Wiley & Sons, Ltd.
4. Kwee, I., Hutter, M., Schmidhuber J. Market-Based Reinforcement Learning in Partially Observable Worlds. 2001.
5. Lin, L-J., 1993, PhD thesis: Reinforcement Learning for Robots Using Neural Networks, Carnegie Mellon University, Pittsburgh, CMU-CS-93-103.
6. Jiming Liu, 2001. Autonomous agents and multi-agent systems. Hong Kong Baptist University.

Čižovs Jurijs, Zmanovska Tatjana un Borisovs Arkādijs. Daudznozīmīgo stāvokļu noteikšana nemarkova vidēs

Klasiskās apmācības ar pastiprināšanu metodes veiksmīgi funkcionē tā saucamajās nemarkova vidēs. Šajā darbā tiek izvirzīta ideja un algoritma izpildīšana nemarkova videi. Ierosinātā pieeja piedāvā daudznozīmīgo stāvokļu apmeklēšanas paņēmienu. Raksts definē un sīki apskata daudznozīmīgu stāvokļu formas nemarkova vidēs. Apmeklēšana ir svarīga fāze pirms politikas uzbūvei nemarkova vidēs izmantojot stāvokļu iekšējo reprezentāciju. Savukārt, stāvokļu iekšējai reprezentācijai ir jāsaglābj konverģences noteikumi, apmācības ar pastiprinājumu īpašības un priekšrocības. Raksta būtiska daļa ir veltīta daudznozīmīgo stāvokļu izrādīšanas pazīmēm un atrāšanas algoritmiem. Kā ir rādīts rakstā, pats grūtākais uzdevums ir atšķirt viena un tā paša stāvokļa divus eksemplārus. Raksts iekļauj sevī teorētisko ievadu, problēmu aprakstu un tās izpausmi šūnu pasaulē. Tika formulēti daudznozīmīgu stāvokļu trīs formas un izvirzīta tās atklāšanās ideja. Lai iegūtu praktiskus rezultātus, tika izstrādāta programmatūra, kas dotai pasaulei veido visu daudznozīmīgu stāvokļu sarakstu. Pamatojoties uz iegūtiem rezultātiem, izvirzīti secinājumi. Šīs darbs ir nepieciešamā iepriekšēja fāze aģenta nemarkova vidē funkcionēšanas uzdevuma izpētē.

Jurij Chizhov, Tatyana Zmanovska and Arkady Borisov. Ambiguous states determination in non-Markovian environments

Classical methods of reinforcement learning are performing successfully in the so-called Markovian environments. In this work the idea and implementation approach are stated for non-Markovian environments. The approach offered represents a way of ambiguous states detection. The paper includes definition and detailed scrutiny of ambiguous states forms. The detection is the prerequisite phase for building optimal policy using internal representation of states. In its turn, the internal representation preserves properties and advantages of the classical algorithm of reinforcement learning; its condition of convergence is important. The central part of the paper is focused on the indication of ambiguous cells and its detection. As shown in the paper, it proves quite difficult to recognise different "copies" of the same states. The paper includes a theoretical introduction, problem description and its display in grid worlds (mazes). Three forms of ambiguous states and the idea of its detection are posed. Testing software is developed to obtain the practical results. The work is a preliminary step in research of agent functioning task in non-Markovian environments.

Чижов Юрий, Змановская Татьяна и Борисов Аркадий. Обнаружение неоднозначных состояний в немарковских средах

Классические методы обучения с подкреплением успешно функционируют в так называемых немарковских средах. В данной работе выдвигается идея алгоритма и его исполнение для немарковских сред. Рассматриваемый подход предлагает способ обнаружения так называемых неоднозначных состояний. Статья включает определение и детальное рассмотрение трех форм неоднозначных состояний в немарковских средах. Фаза обнаружения является предварительным этапом в процессе построения политики в немарковских средах через внутреннее представление состояний. В свою очередь, введение внутреннего представления состояний должно сохранить условия сходимости,

свойства и преимущества классических алгоритмов обучения с подкреплением. В данной статье внимание уделено признакам обнаружения неоднозначных состояний и, собственно, их обнаружению. Как показано в статье, основную сложность составляет задача различения отдельных экземпляров одного и того же состояния. В статье приведено теоретическое введение, описаны проблемы и их проявление в клеточном мире, сформулированы три формы неоднозначных состояний. Выдвинута идея обнаружения неоднозначного состояния. Для получения практических результатов разработана программа, которая для каждой рассматриваемой среды формирует ряд найденных неоднозначных состояний. Данная работа является предварительным исследованием в задаче функционирования агента в немарковской среде.