International Core Journal of Engineering Volume 7 Issue 6, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

Application of Markov Decision Process in Delivery Robot Path Planning System

Mengmeng Wang[1], Fujie Sun[2], Sipei Gu[3,*]

[1]Lyle School of Engineering, Southern Methodist University, Dallas, Texas 75206, America;

[2]South China Agriculture University, Guangzhou, Guangdong510640, China;

[3]Shanghai Normal University, Shanghai 201418, China.

*Corresponding author. Email: gusipei@126.com

Abstract

In brand-new AI era, instead of manual service, library robots are going to transport books for readers. Under such condition, it is not only convenient for readers to access books, but also makes all the books placed in their proper position. This paper employs MDP as the model of robot's path planning. Markov decision process is well applied in many engineering works. It expresses the results of an action with some probability and an overall policy can be determined by combining these probabilities based on which goal specification we choose and how reality problem is converted into mathematic model. In our research, SSP is explored as our specification. Based on Markov decision process, value iteration and LRTDP are implemented. In addition, this paper analyzed the difference between value iteration and LRTDP algorithm via time and space complexity. Through the simulation experiment, it is concluded that LRTDP has fast convergence and is sub optimal, which is our preferred algorithm compared to VI.

Keywords

Delivery Robot; MDP; SSP; LRTDP; VI; RTDP; Comparation.

1. Introduction

Library robots are increasingly replacing human services[1]. Due to bigger and bigger library, it is more complicated for readers to achieve the books they want. In the digital epoch, people came up with the strategies of coding the book. Then readers were able to discover the books according to its specific subregion and serial number. Nonetheless it still takes readers numerous time to reach the exact positions of the books. In today's AI era, based on our paper, people can command the robot to access the book via inputting the code of the book they want into the library robot. This method not only avoids the phenomenon of readers piling up books disorderly, but also lets readers quickly obtain the books. The process of library robot accessing books is related to Dynamic Path Planning system[2].

In recent years, plenty of researchers have studied Uncertainty Intelligence Planning Problems[3]. These planning problems are typically formulated as an exemplification of a Markov Decision Process or MDP[4]. The MDP model has the following characteristics:1)states; 2)actions; 3)transition probability; 4)action costs; 5)goal states. In order to conclude an optimal path, the agent must consider the probability as well as cost of an action. In fact, MDP with basic note is insufficient to tackle the DP problems. More specific restrictions should be added to MDP. In particular, SSP condition would be applied to MDP model, which is also what we prefer to make use of in our paper.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

There are two fundamental algorithms to solve the SSP MDP problems--Policy Iteration and Value Iteration[5]. Both algorithms employ Bellman Equations as the backup to reach the Q* and then repeat this procedure until the V-value of each state converges. The optimal route is the convergence order of the states.

In addition, there are heuristic search solutions[6]. One of the conventional DP algorithms is called LRTA*[Korf, 1990] which is a real-time heuristic search algorithm widely used. After evaluating the heuristic value of each state from the goal state, this algorithm starts to update each states' value via serial trails[7]. In particular, Carlos.H and Pedro.M then presented LRTA*(k)[8], a new LRTA*-based algorithm that is capable of updating the heuristic estimates of up to k states and cuts down on some unnecessary steps. After that, RTDP[9] which is the generalization of LRTA* attached with probability came out. Real-Time Dynamic Programming(RTDP) allows the map to be quickly explored and execute a reasonable action in the present state. After every update, RTDP will consider the random issue, in other words, chances are that the agent may not reach the better state from the current state. In that case, RTDP needs to perform Bellman Backups again. However, this algorithm possesses a shortcoming--lacking the mechanism of convergence detection. Hence, Bonet.B and Geffner.H proposed Labeled RTDP[10], avoiding the repeated convergence when seeking for the optimal route. LRTDP has the recognition of approaching the V* because ϵ will be set to monitor the residual between the present V-value and the last V-value. For instance, if ϵ is less than 0.0001, states will be reckoned as converged and labeled. In that case, the agent can pass through the labeled states directly and explore the rest space. Time has been greatly saved and an optimal path also achieved owing to LRTDP.

In this project, LRTDP and VI would be applied to deal with DP problems based on SSP MDP.

2. Preliminaries

The problem to solve is how to plan an optimal route for the library robot. It is a stochastic intelligence planning problem. To unravel this issue, we establish library 2D map based on MDP model and set it in a SSP situation. On this basis, VI, RTDP and LRTDP algorithm are applied to address this problem. All of this knowledge will be specifically elaborated as follows:

2.1 MDP

The MDP has the following properties.

1) State: the state of the robot includes the location of the robot and value of the states[11] which can be utilized to evaluate the distance between each state and the target. The value of the state is 0, when the state belongs to goal state. When the state is not in the target, in the VI, we assign 100 to the states. In the RTDP, the value of states is assigned according to the shortest distance between the state and the target state, which is also called heuristic value. Besides that, states also embrace the conditions of doors. For example, in state1(location(n8), door1(0), door2(0), door3(1)), 'door3(1)' indicates that the door3 is open.

2) Precondition of action & Probability: the process from one state to another is called action. The occurrence of action has its corresponding preconditions. For instance, only when the robot is in position n10, can the action "from n10 to n11" be taken. This probability is determined by the specific background of the problem, such as the status of doors or the appearance of temporary obstacles.

3) Cost: each action also incorporates its corresponding cost. The definition of cost could be time, distance, or the consumption of work. The size of cost is the key to discover the local optimal value V*.

$$V^*(\bar{s}) = E^{min}_{M,\bar{s}}\left(\sum_{i=0}^{nG} c(s_i, a_i)\right) = inf_{\pi} E^{\pi}_{M,\bar{s}}\left(\sum_{i=0}^{nG} c(s_i, a_i)\right)$$

SSP

Most MDP solution methods with SSP[12] bind demand some extra assumptions about the model.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

1) Goal can be reached with probability 1

2) There are no 0-cost cycles

2.2 Bellman Equations

Bellman equations is the core of the different kinds of algorithms (VI, RTDP, LRTDP), which is like a compass to lead the robot to reach the goal state.

$$Q^*(s,a) = \sum T(s,a,s')[C(s,a,s') + V^*(s')]$$

There are several cardinal variables to explain:

1) Action cost refers to the time or energy required to execute an action (defined by the specific issues).

2) In VI, value of state can be set to like 100 except the goal state which should be set to 0. In RTDP, value of state should be assigned according to the Dijkstra algorithm.

3) For each action, Q-value[13] is the most significant variable. It is achieved by Bellman equation. the Q-values of different directions are compared so as to decide the local optimal direction Q*. The value of each state will be updated after each trail.

4) Transition probability represents the probability of each action occurred.

2.3 Value Iteration

VI[14] is an algorithm to learn the optimal path based on MDP. To begin with, it is necessary to set the initial value of all the states except goal state as a fixed number like 100. Instead, the value of goal state would be assumed as 0. After successive refinements, if the residual between this value and its last value is less than ϵ, the system will grant this state has converged. Figure 1 shows the pseudo-code of the VI.

```
initialize V_0 arbitrarily for each state
n ← 0
repeat
    n ← n + 1
    foreach s ∈ S do
        compute V_n(s) using Bellman backup at s
        compute residual_n(s) = |V_n(s) − V_{n−1}(s)|
    end
until max_{s∈S} residual_n(s) < ϵ;
return greedy policy: π^{V_n}(s) = argmin_{a∈A} Σ_{s'∈S} T(s,a,s') [C(s,a,s') + V_n(s')]
```

Figure 1. Pseudo-code of the VI

There are numerous advantages and disadvantages of VI.

Advantage: it is exceptionally universal. Once fixing the destination, we can achieve the whole optimal route via Bellman equation iteration. These routes do not require a starting point, since they extend from the end to every possible starting point.

Disadvantage: due to abandon the starting point, it calculates every possible starting point, which requires a colossal sum of calculation.

2.4 RTDP

Just like VI, RTDP is also an algorithm based on Bellman Equation. However, it does not need to repeatedly calculate the path from each point to the destination. Contrariwise, before planning path, values of each state are needed to be calculated with the Dijkstra algorithm[15]. The solution is: given graph A and source vertex G, find the shortest path from G to all vertices in the graph. After setting a start point and an end point, RTDP is able to trace an optimal route according to comprehensive calculation of heuristic values, cost and probability of the correlative states. The pseudo-code of RTDP is shown in Figure 2.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

```
RTDP(s_0)
begin
    V_l ← h
    while there is time left do
        TRIAL(s_0)
    end
    return π*_{s_0}
end

TRIAL(s_0)
begin
    s ← s_0
    while s ∉ G do
        a_best ← argmin_{a∈A} Q^{V_l}(s, a)
        V_l(s) ← Q^{V_l}(s, a_best)
        s ← execute action a_best in s
    end
end
```

Figure 2. Pseudo-code of RTDP

In the Figure 2, $argmin_{a\in A}Q^{V_l}(s,a)$ determines the next direction to be taken, after applying bellman equation.

Next, according to the given probability, random sampling is carried out to determine whether the agent can walk through this door and then update the state of each door. As a shorthand, this process is defined as check function. If the result of the sampling is that door is open, the state will be transferred to the next state, otherwise the original state will be retained.

Finally, these steps would be repeated until the agent reach the goal state. In addition, a limited time needs to be placed. The cycle will continue until the time is exhausted.

```
LRTDP(s_0, ε)
begin
    V_l ← h
    while s_0 is not labeled solved do
        LRTDP-TRIAL(s_0, ε)
    end
    return π*_{s_0}
end

LRTDP-TRIAL(s_0, ε)
begin
    visited ← empty stack
    s ← s_0
    while s is not labeled solved do
        push s onto visited
        if s ∈ G then
            break
        end
        a_best ← argmin_{a∈A} Q^{V_l}(s, a)
        V_l(s) ← Q^{V_l}(s, a_best)
        s ← execute action a_best in s
    end
    while visited ≠ empty stack do
        s ← pop the top of visited
        if ¬CHECK-SOLVED(s, ε) then
            break
        end
    end
end
```

Figure 3. Pseudo-code of LRTDP

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

2.5 LRTDP

Without loss of generality, many simple problems are not computationally intensive, accordingly they run thousands of iterations in just a few seconds. Because RTDP will always cycle before running out of time, it is pointless to continue to run RTDP since the terminal has been concluded, which is a serious waste of computing resources. We call it "over-convergence". However, LRTDP can address this problem, depending on its feature of labelling the state as solved when the state has converged. The process of checking states' situation is to monitor the residual (which we will regard as ϵ) between the one state's current V-value and last V-value. If ϵ is less than the fixed number, this state will be reckoned as solved and then should be labelled. In general, convergence is an up-down process.

In this way, the trail will terminate when it achieves the optimal path instead of running until the limited time. The pseudo-code of LRTDP is shown in Figure 3.

3. Experiment

In order to verify the feasibility and superiority of these algorithms in this problem, we write several codes in Python to compare their convergence speed and optimality.

3.1 Experiment Result of Value Iteration

{'n1': 100, 'n2': 100, 'n3': 100, 'n4': 100, 'n5': 100, 'n6': 100, 'n7': 100, 'n8': 100, 'n9': 100, 'n10': 100, 'n11': 0, 'n12': 100, 'n13': 100}
{'n1': 103.0, 'n2': 103.0, 'n3': 103.0, 'n4': 103.0, 'n5': 101.0, 'n6': 101.0, 'n7': 101.0, 'n8': 83.898, 'n9': 101.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 102.0}
{'n1': 104.1, 'n2': 87.85309999999998, 'n3': 104.1, 'n4': 104.1, 'n5': 102.0, 'n6': 102.0, 'n7': 84.898, 'n8': 71.000298, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 12.004999999999999, 'n2': 74.8429381, 'n3': 88.8581, 'n4': 105.105, 'n5': 85.898, 'n6': 85.898, 'n7': 72.000298, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.40025, 'n2': 11.492146905, 'n3': 75.8431881, 'n4': 89.85834999999999, 'n5': 73.000298, 'n6': 73.000298, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1700124999999995, 'n2': 8.32460734525, 'n3': 12.492159404999999, 'n4': 76.84320059999999, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.158500624999999, 'n2': 8.1662303672625, 'n3': 9.32460797025, 'n4': 13.492160029999999, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.15792503125, 'n2': 8.158311518363124, 'n3': 9.166230398512498, 'n4': 10.3246080015, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1578962515625, 'n2': 8.157915575918157, 'n3': 9.158311519925624, 'n4': 10.166230400075, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1578948125781245, 'n2': 8.157895778795908, 'n3': 9.157915575996281, 'n4': 10.15831152000375, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1578947406289055, 'n2': 8.157894788939796, 'n3': 9.157895778799812, 'n4': 10.157915576000187, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1578947370314445, 'n2': 8.15789473944699, 'n3': 9.15789478893999, 'n4': 10.15789577880001, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}
{'n1': 7.1578947368515715, 'n2': 8.157894736972349, 'n3': 9.157894739446998, 'n4': 10.15789478894, 'n5': 7.0, 'n6': 7.0, 'n7': 6.0, 'n8': 5.0, 'n9': 4.0, 'n10': 2.0, 'n11': 0, 'n12': 2.0, 'n13': 4.0}

Figure 4. The results of python code (n11 was set as goal state)

As shown in Figure 4, the red marked represents the convergence of nodes. The result has 13 iterations. According to the order of the nodes' convergence, the optimal path can be concluded.

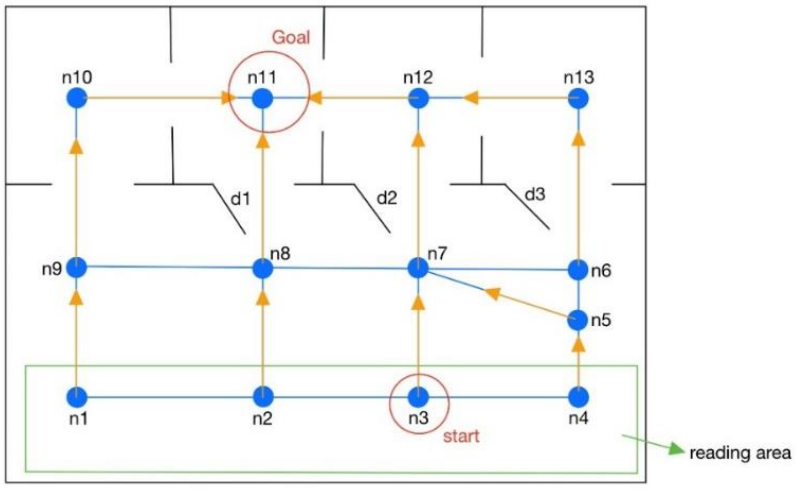

Figure 5. The simple map of the library

For example, in Figure 5, n3 is taken as start state. The optimal path from n3 to n11 is 'n3_n7, n7_12, n12_n11'.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

3.2 Experiment Result of LRTDP

The result of LRTDP is shown in Figure 6. A restriction was added to avoid RTDP from repeating for so many unnecessary trails.

```
Heuristic: {'n1': 6.0, 'n2': 5.0, 'n3': 6.0, 'n4': 7.0, 'n5': 4.0, 'n6': 4.0, 'n7': 3.0, 'n8': 2.0, 'n9': 3.0, 'n10': 2.0, 'n11': 0.0, 'n12': 2.0, 'n13': 4.0}
first visited: ['n3', 'n7', 'n8', 'n11']
after check solved: ['n3', 'n7']
solved list: {'n1': 0, 'n2': 0, 'n3': 0, 'n4': 0, 'n5': 0, 'n6': 0, 'n7': 0, 'n8': 0, 'n9': 0, 'n10': 0, 'n11': 1, 'n12': 0, 'n13': 0}
first visited: ['n3', 'n7', 'n8']
after check solved: ['n3', 'n7']
solved list: {'n1': 0, 'n2': 0, 'n3': 0, 'n4': 0, 'n5': 0, 'n6': 0, 'n7': 0, 'n8': 0, 'n9': 0, 'n10': 0, 'n11': 1, 'n12': 0, 'n13': 0}
first visited: ['n3', 'n7', 'n8']
after check solved: []
solved list: {'n1': 0, 'n2': 0, 'n3': 0, 'n4': 0, 'n5': 0, 'n6': 0, 'n7': 1, 'n8': 1, 'n9': 0, 'n10': 0, 'n11': 1, 'n12': 0, 'n13': 0}
first visited: ['n3']
after check solved: []
solved list: {'n1': 0, 'n2': 0, 'n3': 1, 'n4': 0, 'n5': 0, 'n6': 0, 'n7': 1, 'n8': 1, 'n9': 0, 'n10': 0, 'n11': 1, 'n12': 0, 'n13': 0}
Please wait for time runs out...
..
Path ['n3', 'n7', 'n8', 'n11']
```

Figure 6. The result of LRTDP

4. Conclusion

Logically, LRTDP should take less time than Value iteration owing to less pointless search. However, it is astonishing that running LRTDP code takes a lot more time than VI. As shown in Figure 7.

```
Path ['n3', 'n7', 'n8', 'n11']
Time of running the LRTDP:9.707354

Time of running VI:0.000997
```

Figure 7. Time of running LRTDP and VI

On our reflection, it is the LRTDP's heuristic step (generating initial values) that takes substantial time. In comparison the generation of the initial values in VI is too simple, just making all of them 100. Deeper reason requires further explore.

Generally speaking, VI algorithm spends considerable space on producing the complete and copious routes. It costs LRTDP plentiful space to produce heuristic values. However, VI expends much more space than LRTDP in general. As shown in Table 1.

Table 1. Comparison of VI and LRTDP

Algorithm	Space Consumed	Speed	Optimal
VI	more	Slow convergence	optimal
LRTDP	less	Fast convergence	Partially optimal

Back to our goal, our algorithms are designed to allow library robots to deliver books and guide readers. After the comparison above, we find that LRTDP has fast convergence and is sub optimal, which is our preferrable algorithm.

Acknowledgments

Meng Meng Wan, Fu Jie Sun and Si Pei Gu contribute equally to this work.

References

[1] Weld, D.S.(1999) Recent advances in AI planning. Artificial Intelligence Magazine. 20(2), pp. 93–122.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202106_7(6).0058

[2] Ronald A. Howard.(1960) Dynamic Programming and Markov Processes. MIT Press, New York.

[3] Bagnell, J.A. et al.(2001) Solving uncertainty markov decision processes. Technical Report CMU-RI-TR-01-25, Robotics Insitute, Carnegie Mellon University.

[4] Martin L. Puterman.(1994) Markov Decision Processes. Bulletin of Mathematical Biology.

[5] Richard Bellman.(1957) Dynamic Programming. Prentice Hall. Upper Saddle River.

[6] Bonet, B., Geffner, H. (1999) Planning as Heuristic Search: New Results. European Conference on Planning: Recent Advances in Ai Planning. Springer-Verlag.

[7] Shimbo, M., Ishida, T.(2003) Controlling the learning process of real-time heuristic search. Artificial Intelligence, 146(1):1-41.

[8] Hernandez, C., Meseguer, P.(2005) LRTA*(k). In Proc. IJCAI.

[9] Steven Bradtke et al.(1993) Learning to Act Using Real-Time Dynamic Programming. Article in Artificial Intelligence.

[10] Bonet, B., Geffner, H.(2003) Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS). Trento. Italy.

[11] Munos, R.(2004) Efficient resourses allocation for markov decision processes. In: Proc. of NIPS.

[12] Tsitsiklis, B.J.N.(1991) An analysis of stochastic shortest path problems[J]. Mathematics of Operations Research. 16(3):580-595.

[13] Bonet, B., Geffner, H.(2006) Learning Depth-First Search: A Unified Approach to Heuristic Search in Deterministic and Non-Deterministic Settings, and its application to MDPs. Proc. of 16th Int. Conf. on Automated Planning and Scheduling (ICAPS). Cumbria, UK. AAAI Press. Pages 142-151.

[14] Jiang, Z. (2004) Artificial intelligence -- a modern method. The Posts and Telecommunications Press. Second Edition. Beijing.

[15] Adelson-Velsky, G. M., Levner, E.(2002) Project scheduling in and--or graphs: a generalization of dijkstra's algorithm.