The comparison of using Markov and non-Markov decision processes for inverting double pendulum

Yu-Che Huang

***Abstract*— Reinforcement learning has been widely used for solving sequential decision tasks. The difference in control performance by using Markov and non-Markov in double pendulums problem are investigated. It is demonstrated that non-Markov provide a higher control accuracy than Markov solution. However, it still has some drawback with spending longer settling time in the step response and composed by a complex solution structure.**

***Keywords*— artificial intelligence, genetic programming non-Markov decision, reinforcement learning cart double pendulum**

I. Introduction

REINFORCEMENT LEARNING (RL) is a subfield of machine learning which offers the ability to enhance the performance of the dynamic system in the future by using previous experience via interacting with the environment. Due to some difficulties in describing the hidden states or actions of existing control models, designing a control system might be a conundrum. Therefore, RL plays a crucial role in alleviating the issue by performance evaluation by using different approaches while operating in the same circumstances to find the best control strategy [1].

In addition, Markov Decision Process (MDP) is employed as a concept to emulate RL algorithm by modelling the interaction between the agent and its environment as a finite state machine which is illustrated in fig. 1. The principle is shown as follows: the agent makes the corresponding action depends on the current state contributed by the environment. Simultaneously, the environment responds to the new status affected by the current action determined by the agent and generates the reward or punishment value as an evaluation of decision quality [2].

However, due to the limitation of sensors set, they might not always offer adequate information to represent the current state, these hidden information may result in the improper algorithm control which may lead to an inaccuracy in the decision-making process [3]. Therefore, this research analyses the performance which adopts the non-Markov decision process that offers the algorithm generating the action by considering not only the current states but also the previous states and action as explicit inputs [4]. Moreover, the research evaluates whether implementing the non-Markov decision process may contribute a better control performance. The results are then compared with MDP to solve the inverted pendulum problem. In addition, to simulate the dynamic problem, the work employees the double pendulum as the environment for algorithm training and analyses the control performance by considering the different number of previous explicit inputs.

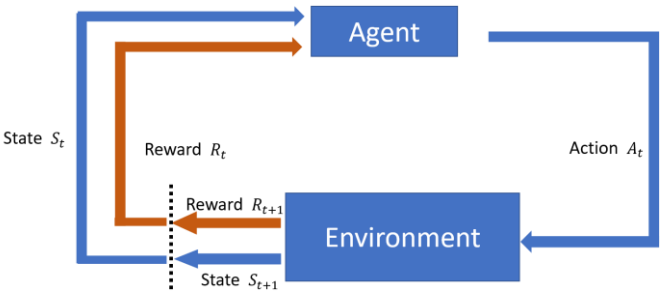


Fig. 1. Markov decision processes model

A. Reinforcement learning

To analyse the behaviour of the dynamic system, the procedure from initial status to the term status is composed of several step states and actions that described as an n-dimensional vector. The process of dynamic system movement can be represented into the following equation (1), where S and A are indicated the state and the action in the current timestep.

$$S_0 \xrightarrow{a_0} S_1 \xrightarrow{a_1} S_2 \xrightarrow{a_2} S_3 \xrightarrow{a_3} S_4 \ldots \quad (1)$$

In addition, the fitness function is employed to offer the ability to evaluate candidate performance. In each time step, the current fitness value is added by the reward given by the current state and the previous fitness value with attenuation.
The concept is demonstrated as following formula (2) where ϒ, R(S) and N is a discount factor, transform function and episode length [1].

$$fitness_i = \sum_{t=0}^{N} \gamma^t \, \mathrm{R}[state(t)] \quad (2)$$

This MSc Individual Project was performed at the Department of Electronic & Electrical Engineering at the University of Sheffield, UK It was supervised by Peter Rockett (e-mail: p.rockett@sheffield.ac.uk).

Where $R(S_t)$ is the transform function used for generating the gradient reward value by passing the current status of the dynamic system in each time-step. In addition, the discount factor Υ ($0 \leq \Upsilon < 1$) is employed to provide a corresponding weight while considering different previous performance behaviour via accumulating reward value through the whole simulation. The weight variation by using different discount value is illustrated in the following figure 2.

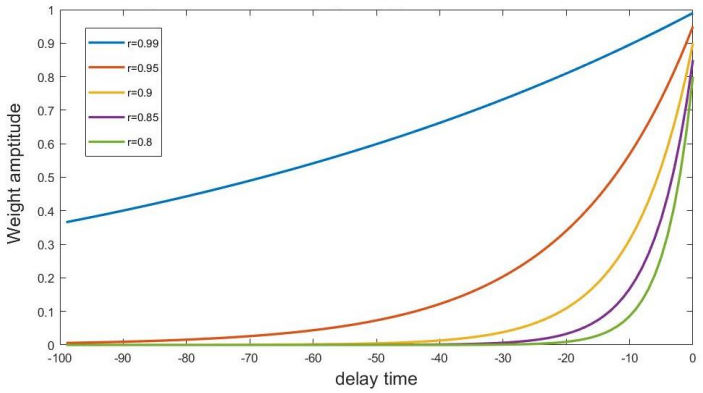
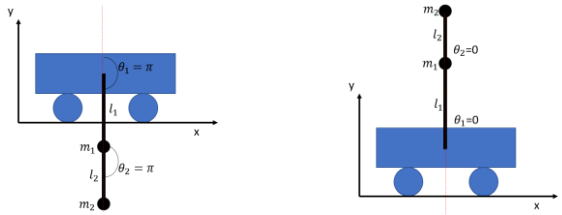


Fig. 2. Weight Attenuation by using a different discount factor

While using a lower discount factor, it provides a fast-fading weight variation when considering the previous behaviour performance. Therefore, the result of the fitness value concentrates on the performance behaviour that manipulated recently. In contrast, if adopting Υ =1 as the discount value, it indicates that the weight for considering the previous behaviour is the same, and the result is concerned whole the performance behaviour during the whole experiment [5].

B. Double pendulum

The double pendulum is a classic mechanical system with highly non-linear behaviour which consists of two links that a pendulum connected by a cart and ended by a second pendulum on the other side [6]. The cart is given by two directions of external force generated by an algorithm to actuate links as a passive double pendulum. To analysis the movement of pendulums, the status can be represented as vector x=[θ_1, θ_2, $\dot{\theta}_1$, $\dot{\theta}_2$, $\ddot{\theta}_1$, $\ddot{\theta}_2$], where θ, $\dot{\theta}$ and $\ddot{\theta}$ are the angle, angular velocity and angular accelerator of each pendulum [7]. In addition, each angle of the pendulum is defined as 0 radius when pendulums are standing straight up of the cart. In the swing-up task, the pendulum is given the initial state x=[π, π,0,0,0,0] to the target status with x=[0,0,0,0,0,0] and stays remain in the balance position as shown as figure 3.



(a) Initial status (c) Balance status

Fig. 3. Cart-pole Swing -up task

In order to simulate the pendulum movement, the work adopts the equation of the motion by using the direct methodology according to Euler-LaGrange formula to identify the angular acceleration of each pendulum [8]. The damping effects and external force are embedded in this formula. The equation for solving the value of each element in state vector is demonstrated as the following formula (3), where $\gamma_1$and γ_2 are extra terms due to considering the damping factor k and external force F (formula 4). Furthermore, the work assumes the movement of cart is an independent system that it is not affected by the motion of the double pendulum.

$$\dot{\theta}_1 = \frac{d\theta_1}{dt}$$

$$\dot{\theta}_2 = \frac{d\theta_2}{dt}$$

$$\ddot{\theta}_1 = \frac{m_2 l_1 \dot{\theta_1}^2 \sin\left(2(\theta_1 - \theta_2)\right) + 2m_2 l_2 \dot{\theta_2}^2 \sin(\theta_1 - \theta_2)}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]}$$

$$+\frac{2m_2 l_2 \dot{\theta_2}^2 \sin(\theta_1 - \theta_2) + 2gm_2 \cos(\theta_2)\sin(\theta_1 - \theta_2)}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]}$$

$$+\frac{2gm_1 \sin(\theta_1) + \gamma_1}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]}$$

$$\ddot{\theta}_2 = \frac{m_2 l_2 \dot{\theta_2}^2 \sin\left(2(\theta_1 - \theta_2)\right)}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]}$$

$$+\frac{2(m_1 + m_2) l_1 \dot{\theta_1}^2 \sin(\theta_1 - \theta_2)}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]}$$

$$+\frac{2g(m_1 + m_2)\cos(\theta_1)\sin(\theta_1 - \theta_2) + \gamma_2}{-2l_1[m_1 + m_2\, sin^2(\theta_1 - \theta_2)]} \quad (3)$$

$$\gamma_1 = 2\left(k\dot{\theta}_1 - F\right) - 2(k\dot{\theta}_2 - F)\cos(\theta_1 - \theta_2)$$

$$\gamma_2 = 2\left(k\dot{\theta}_1 - F\right)\cos(\theta_1 - \theta_2) - \frac{2(m_1+m_2)}{m_2}(k\dot{\theta}_2 - F) \quad (4)$$

C. Genetic Programming

Genetic programming(GP) has been used frequently due to it provides a powerful ability to adapt any environment by using the evolution theory purposed by Darwin [9]. The algorithm adopts a considerable candidate sequential control functions to enhance as an approximate solution by evaluating via the fitness function to simulate the phenomenon of evolution in the algorithm. The concept of evolutionary computation can be divided into three sections including: initialize, creating generation and evaluation.

The initialisation is the first step of the evolutionary algorithm that generating the candidate tree according by given

critical parameter including maximum tree depth and the probability of each genetic operator [10]. Initial trees are restricted by the given size and consisted of two primary sets that including possible of terminals T and internal nodes I where T represents input value such as θ and $\dot{\theta}$ or other constant value and I implies the operator of the control function. The example of the tree-based GP is illustrated as figure 4. The structure of tree-based GP can be assembled by two common methodologies: grow method and full method, and the algorithm of these methods are identified in the previous research [11].

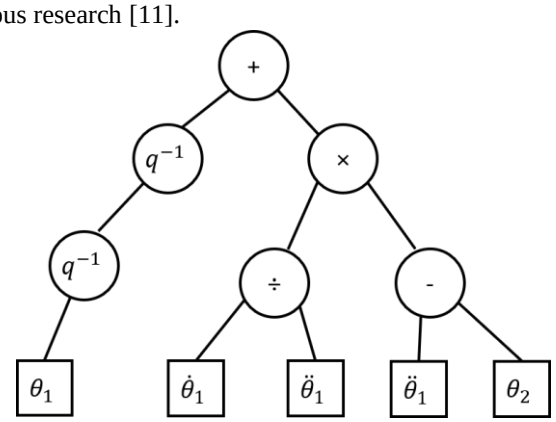


Fig. 4. Example of genetic programming candidate solver

In order to diversify the candidate solutions, crossover and mutation are adopted by the algorithm. The crossover is implemented by exchanging genetic material, which is the subsection of the parent structure tree. In contrast, the mutation operator offers a function that only substituting a single individual genetic material by the subtree with a randomly created tree. By using these two genetic operators, it implements the simulation that each solution has a slightly different ability to adapt to the environment from others according to the evolution theory. In addition, these operators allow the algorithm approaches to the term solution gradually in each generation step to find the optimize solution.

II. Economic, Legal, Social, Ethical and Environmental Context

Due to the function flexibility offered by machine learning, the algorithm can accomplish almost any given task [12]. Based on the characteristic, there is a potential risk while adopting collecting the training data. For instance, while developing the algorithm for the social network such as friend suggestion, a considerable private information may be adopted as the training data to accomplish the given aim. It implies that it is unethical for developer accessing the private information without any owner's permission.

III. Methodology

In order to provide an accurate simulation of pendulum movement, the formula of dynamic system (3) is solved by using fourth-order Runge-Kutta which provides adaptive optimise step size control [13]. The parameters of the pendulum system used are demonstrated in Table I. Moreover, the simulation offers 12.5 m of the cyclic track to avoid the limitation of decision that determined by controller. In the swing-up task, the external force is allowed a continuous rage of $F \in [-100\ 100]$ and applied to the dynamic system every 0.01s.

TABLE I
Dynamic System Parameter

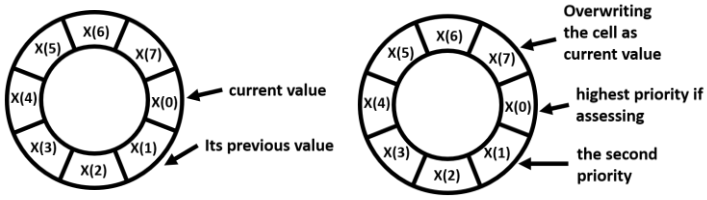
Parameter	Symbol	Value
Mass of cart	m_{cart}	1 kg
Mass of pendulum 1	m_1	1 kg
Mass of pendulum 2	m_2	1 kg
Length of link 1	l_1	1 m
Length of link 2	l_2	1 m
Damping factor	k	0.1
Gravitational constant	g	9.8
Length of Track	N/A	12.5 m
Time step	N/A	0.01 sec

TABLE II
Terminal Set

Parameter	Description
θ_1	Angle one
θ_2	Angle two
$\dot{\theta}_1$	Angular velocity at angle one
$\dot{\theta}_2$	Angular velocity at angle two
$\ddot{\theta}_1$	Angular accelerator at angle one
$\ddot{\theta}_2$	Angular accelerator at angle two
R	Random constant $\in [-10, 10]$

The algorithm consisted of a given range of random constants and the terminal nodes as shown (Table II) is adopted to optimise the control strategy in both swing-up and balance task. To achieve the project aim, the work separates the task into two experiments: swing-up task and balance control by using both Markov and non-Markov decision processes model and evaluating the performance by analysing step respond and the stable pendulum position. In addition, the time delay operators (q^{-1}) are employed in non-Markov experiment which allows algorithm considers the different number of previous explicit inputs. Moreover, to avoid the issue caused by conventional division operator when denominator is zero, analytic quotient operator (AQ) is employed to alleviate the issue with a lower mean squared errors over a range of regression tasks [14]. Each decision process model are using different number of genetic operator set: $Markov = \{+, -, \times, AQ, \}$ and $nonMarkov = \{+, -, \times, AQ, q^{-1}, q^{-2}, q^{-3}\}$.

To implement the non-Markov decision processes, the ring buffer is employed in the project to store the past information observed from the environment. The structure of the ring buffer is illustrated in figure 5. By shifting the current flag, the oldest element x[7] is overwritten as the current value while its previous neighbour index x[0] becomes the highest priority location while assessing. By using the technique, it can alleviate the computation load significantly comparing with traditional shift register.



(a) current status flag when t(n) (b) current status flag when t(n+1)

Fig. 5. Structure of the ring buffer

A. *Swing up*

In this task, the algorithm is configured to evolve the sequence control strategy to swing pendulums up in the minimum time. The condition to accomplish the aim is not only to swing both pendulums to the target range of position but must be ensured the $\dot{\theta}_1$and $\dot{\theta}_2$ is small enough for balance task to solve it. The fitness function is determined to ensure given a proportional fitness of candidate solution according to its performance. Each candidate solution is given the same episode length as a range of time for individual solving the dynamic system problem. The function shown in formula 5 is designed for giving a higher reward value when using shorter time to accomplish the task to observe the highest efficiency solution for the task.

$$fitness(i) = Episode\ Length - t_{up}(i) \tag{5}$$

Where $t_{up}(i)$ is the time that action is taken to swing pendulums up when the status vector satisfies the target position condition that demonstrated as follow:

$$(|\theta_1| < 10^{\circ})\ and\ (|\theta_2| < 10^{\circ})\ and\ \left(\left|\dot{\theta}_1\right| < 1.5\right)\ and\ \left(\left|\dot{\theta}_2\right| < 1.5\right)\ and\ (|x_{cart}| < 0.2) \tag{6}$$

It must be ensured that the algorithm has an adequate generation range to evolve the optimise solution. Both Markov and non-Markov process employee 200,000 times for evaluation. In addition, given appropriate time for each simulation round for candidate to accomplish the task is important. Thus, each round is allocated 20 second which is around 2 times the best result with 8.8 seconds in the previous research[15].

B. *Balance*

In order to alleviate the defect status of pendulums given by the swing-up task, the solution must offer the ability to stabilise the pendulums, with the slight angular shift of the inverted position. Therefore, the candidate solutions are given by $10^{\circ}$of the pendulums as initial status which is the worst condition that swing up task might end with. Furthermore, the work expects the solution provides the ability of not only balancing as longer as it can offer but minimising the time it took to stabilise the pendulum. Moreover, Gaussian distribution function is exploited to offer gradient reward value based on the offer gradient reward value according to diverse state given by the dynamic system. The gradient function must provide a clear difference affected by even slight changes in either θ , $\dot{\theta}$ or x_{cart} in the state vector.

The equation is demonstrated in formula (7) where R_t is the reward value given by the current state and ω is the weight of the different status element. The distribution function is illustrated as the following figure by using the weight parameter in Table III.

$$\mathrm{R}_t = e^{\left[\omega_\theta\left({\theta_1}^2+{\theta_2}^2\right)+\omega_\theta\left({\dot{\theta}_1}^2+{\dot{\theta}_2}^2\right)+\omega_d\left({x_{cart}}^2\right)\right]} \tag{7}$$

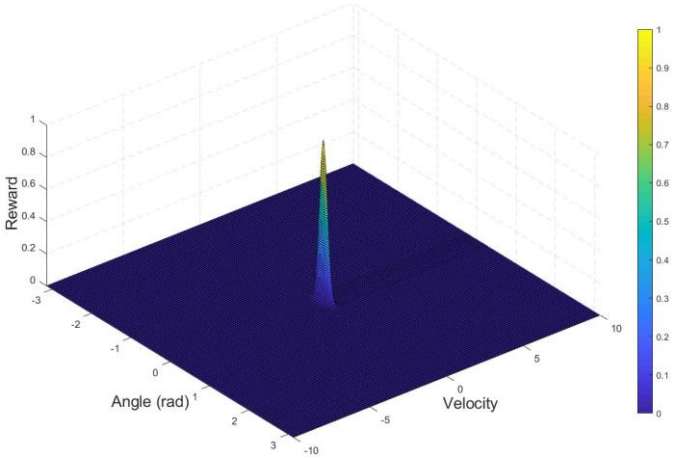


Fig. 6. Reward distribution

TABLE III
WEIGHT VALUE FOR GAUSSIAN DISTRIBUTION FUNCTION

Parameter	VALUE
Discount Factor	1
Simulation length	6,000
Theta weight	60
Velocity weight	30
Displacement weight	1

In this task, the discount factor is assigned as 1 that offers the same weight for considering the previous behaviour and provides the algorithm to observe a solution that stable the pendulum with a shorter time to reach the peak position to maximize the reward value. Furthermore, the complexity of the solution structure and the number of actions used for accomplishing the task, are considered in this project to evaluate the performance of the task by adopting different decision models. Moreover, to evaluate the efficiency in balancing task, the concept of step respond is employed that adopt 0.005 as the stable threshold in this work [16].

IV. RESULTS & EVALUATION

The result of achieving swing up and balancing that adopting both Markov and non-Markov decision model is demonstrated separately into two main sections with the learning efficiency first followed by the control performance and solution complexity. Both sections are illustrated the result of using Markov model then non-Markov one.

A. *Learning efficiency*

The fitness variation of swing-up task by adopting Markov and non-Markov process during the whole experiment is illustrated in figure 7. In the Markov process, the first solution for swing up task is observed until approximately 33,660 evolutions and grows slightly to the best solution at the last 20,000 evolutions with taking only 65 actions to accomplish the task. Figure 8 shows the variation of fitness over 200,000 times of evolution in balancing task. The fitness value increases significantly at the beginning of the algorithm, which means the algorithm identifies the acceptable solution by using within 20,000 evolution. The fitness value raises considerably at the initial 15,000 evolutions. The first solution that achieves stabling the pendulum during whole the experiment is observed at 12137 evolution and maximise the reward to supply the ability to stabilise the pendulums within the shortest time in this experiment.

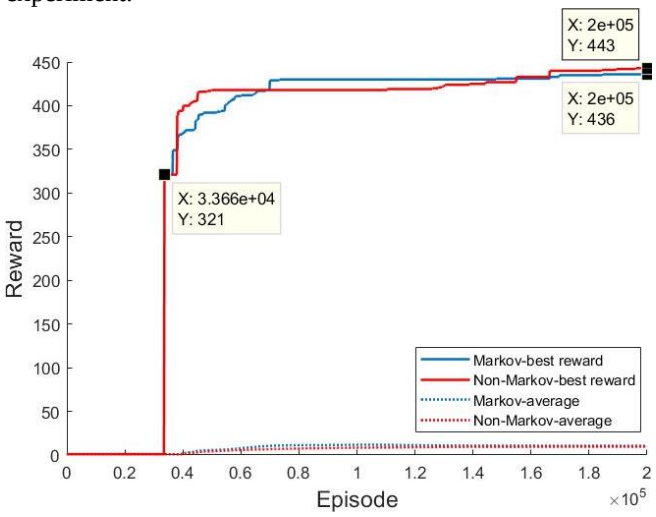


Fig. 7. The variation of best performance and average observed by GP in the swing-up task by using Markov and Non-Markov processes

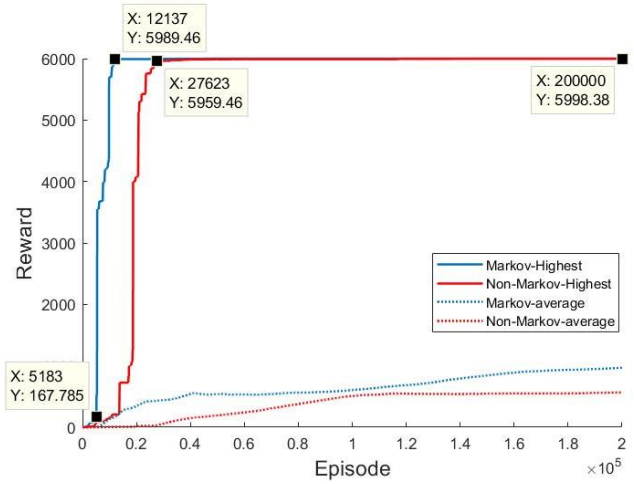


Fig. 8. The variation of best performance and average observed by GP in balance task by using Markov and Non-Markov processes

In the non-Markov model, the first solution in the swing-up task is found after approximately 33,660 evolution and increases significantly to above 400 of reward vale at around 40,000 generations and optimised slightly to the best solution with only using 57 actions to accomplish the task. In the balance task, the fitness value increases slowly until 15,000 but rising dramatically to fine the solution that can stable the pendulums through the whole simulation. The candidate solution is enhanced gradually for reducing the time using while stabling the pendulums to maximise the fitness value.

B. *Control performance and solution complexity*

In this section, both the control performance and solution complexity evaluation are demonstrated separately with the first swing-up task followed by balancing task. The complexity of the best control solution in each task by using different decision model is shown as Table IV.

For the swing-up task, the complexity of the optimised strategy by adopting the Markov process consists of 87 nodes. The best control solution of the experiment offers the ability to reach the target position by taking 87 actions. The state of the pendulum is ended by x=[-7.1297°, -6.51273°, 1.06969, -0.126827, 0.052558], where each element represents $\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2$ and x_{cart}. The control performance of each state element is illustrated in the following figure.

TABLE IV
THE COMPLEXITY OF THE SOLUTION STRUCTURE IN EACH TASK

Process model	Task	TREE DEPTH	TREE NODES USED
Markov	Swing up	13	87
	Balance	19	213
Non-Markov	Swing up	11	65
	Balance	24	469

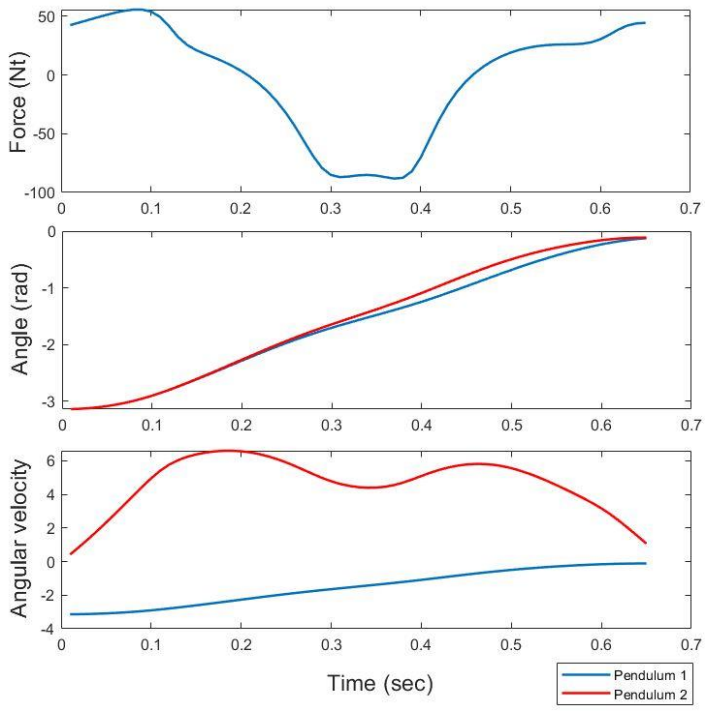


Fig. 9. Best GP swing-up controller by using Markov processes. The figure demonstrates the variation of angle and angular velocity in each pendulum affected by given different input force.

In contrast, the optimised solution by exploiting the non-Markov process in the swing-up task is assembled by 65 nodes which is more compactable comparing with the Markov result. The optimised control strategy takes only 57 actions to achieve the aim position via considering maximum 5 steps of previous information as the explicit inputs comparing with taking 65 decisions in Markov method. The status of the system is ended by x=[-1.4637°, -4.3235°, 1.46538, -0.205297, 0.072282] that offers a higher accuracy position contrast with Markov one. The value of each pendulum status performed in each time step is illustrated in Figure 11.

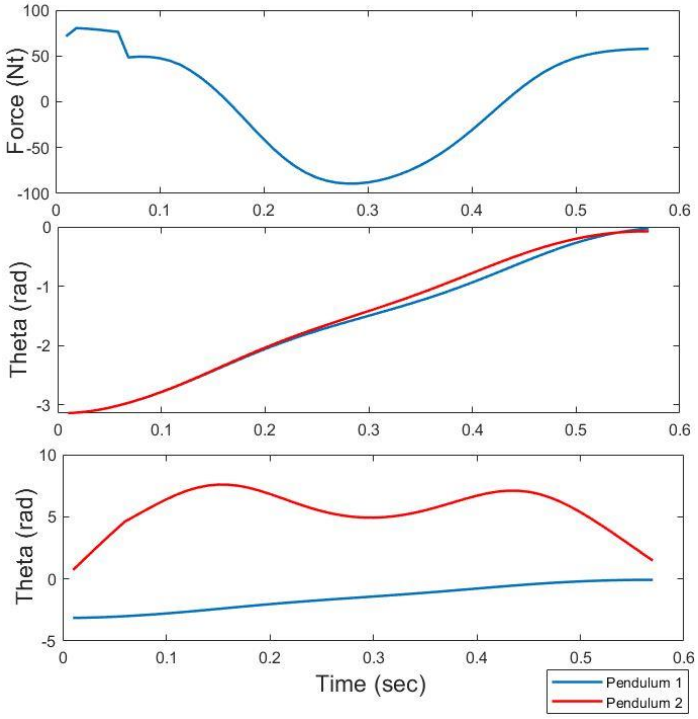


Fig. 10. Best GP swing-up controller by using non-Markov processes. The figure demonstrates the variation of angle and angular velocity in each pendulum affected by given different input force.

For balancing task, the complexity of the Markov result is assembled by 213 nodes and the function only taking 0.67 sec to stabilise the pendulums form the given position. The final position of the state vector is x=[1.70332e-05, - 1.08316e-04, -0.0095, -0.01]. The variation of each element during the simulation is illustrated in Figure 11. The solution reduces the given angle to a smaller value by using only 10 steps but both pendulums are shaking until 0.78 sec. In the last 1 section of the simulation, both angles still have tiny tremor variation due to the external force.

In addition, the non-Markov solution is consisted of 469 nodes and finishing the stabilisation by using 2.2 sec with an increasing value at around 2 sec (Figure 13). The state vector is end by x=[1.32298e-07, -2.01263e-05, -0.0188, -0.0141] by considering maximum 9 steps of previous information. The state during the simulation is shown in Figure 12. Worth to mention that the force calculated by non-Markov result is within 1000 comparing with using nearly 2,000 in the Markov one. In this task, non-Markov result adopts a complex solution and taking longer time than Markov one to stabilise the dynamic system.

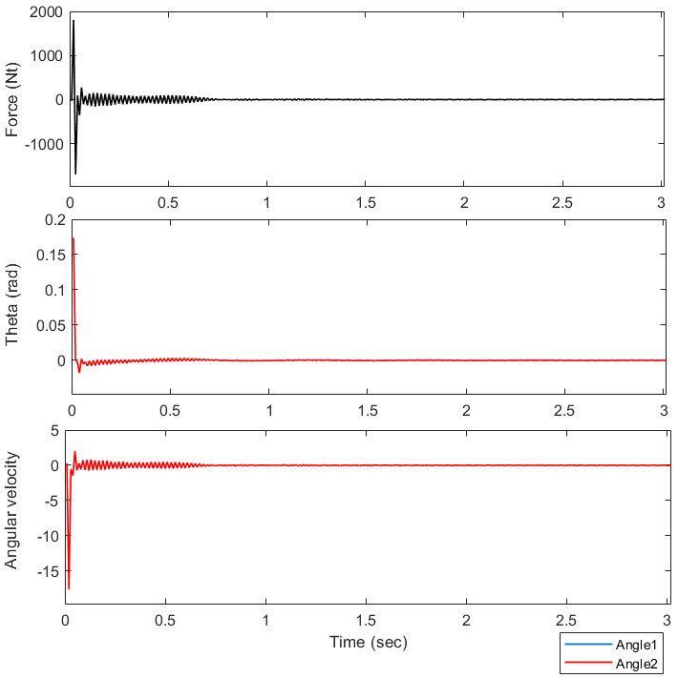


Fig. 11. Best GP Balance controller by using Markov processes. The figure demonstrates the variation of angle and angular velocity in each pendulum affected by given different input force.

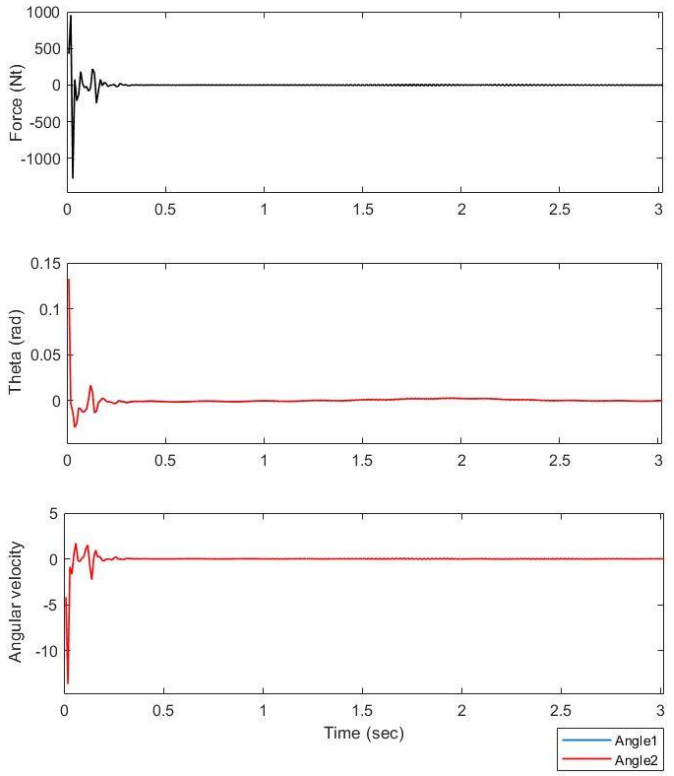


Fig. 12. Best GP Balance controller by using non-Markov processes. The figure demonstrates the variation of angle and angular velocity in each pendulum affected by given different input force.

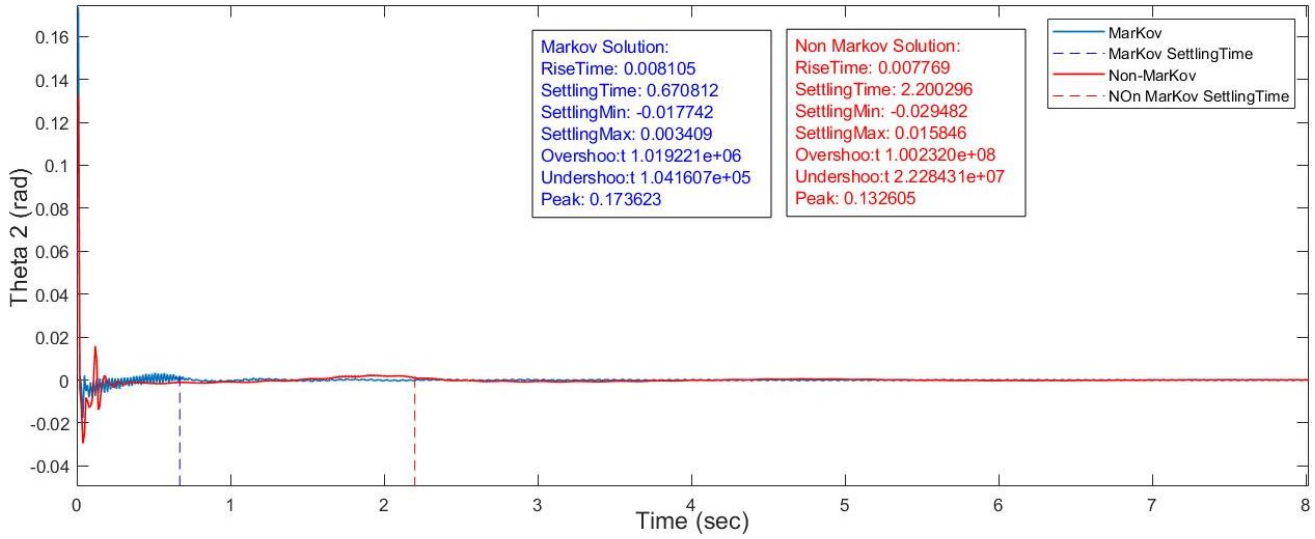


Fig. 13. The comparison of Markov and non-Markov solution in balance task.

V. Discussion and Conclusions

Due to the current status of a dynamic system is highly sensitive to the previous position [17], the solution based on Markov process model may not provide an appropriate decision [3] because of only considering the current information while determining the action. Thus, the non-Markov process is adopted to alleviate the issue that offers a function for concerning previous output form the environment as the explicit inputs. This work offers the evidence to prove that non-Markov solution does provide an accurate performance with a smaller status tolerance and a shorter time used comparing with Markov one in swing-up task. However, the strategy still generates a non-zero force to the cart even when the current status of pendulums is at the target position, due to it considers several previous statuses as the explicit inputs. Based on the characteristic, the solution occupies a longer time for stabilising both pendulums to a stable position. The result of the comparison is shown in figure 13. Even though, non-Markov result still supplies a better performance with the pendulums literally operating at the straight up of the cart without any shaking after 16 sec of the simulation. In contrast, Markov solution still provides some advantage that offers a compact tree structure and higher learning efficiency in the balancing task. In addition, it provides a fast strategy for stabilising the pendulums form a given initial position.

To conclude, the non-Markov solution can provide a higher accurate control performance than Markov one. However, the prior one does not always offer a compact structure of the solution and take longer respond time to achieve the aim of the given task.

VI. Future Work

As a continuation of the research, we are interesting in the difference of the control performance by adopting the non-Markov process in other reinforcement learning structure such as neural network and Q-learning

References

[1] B. Recht, “A Tour of Reinforcement Learning: The View from Continuous Control,” pp. 1–28, 2018.

[2] R. Ortner, “Adaptive aggregation for reinforcement learning in average reward Markov decision processes,” *Ann. Oper. Res.*, vol. 208, no. 1, pp. 321–336, 2013.

[3] P. Smyth, “Hidden Markov models and neural networks for fault detection in dynamic systems,” *Neural Networks Signal Process. III - Proc. 1993 IEEE Work. NNSP 1993*, vol. 27, no. 1, pp. 582–591, 1993.

[4] S. D. Whitehead and L. J. Lin, “Reinforcement learning of non-Markov decision processes,” *Artif. Intell.*, vol. 73, no. 1–2, pp. 271–306, 1995.

[5] N. Yoshida, E. Uchibe, and K. Doya, “Reinforcement learning with state-dependent discount factor,” *2013 IEEE 3rd Jt. Int. Conf. Dev. Learn. Epigenetic Robot. ICDL 2013 - Electron. Conf. Proc.*, pp. 1–6, 2013.

[6] E. Bayo, J. Garcia De Jalon, and M. A. Serna, “A modified lagrangian formulation for the dynamic analysis of constrained mechanical systems,” *Comput. Methods Appl. Mech. Eng.*, vol. 71, no. 2, pp. 183–195, 1988.

[7] D. C. Dracopoulos and B. D. Nichols, “Genetic programming for the minimum time swing up and balance control acrobot problem,” *Expert Syst.*, vol. 34, no. 5, pp. 1–10, 2017.

[8] J. Chen and J. Chen, “Chaos from simplicity : an introduction to the double pendulum.”

[9] M. Pagel, “Darwin ’ s evolution Marseillaise,” *October*, 1999.

[10] Y.-K. Kim, O.-S. Kwon, Y.-W. Cho, and K.-S. Seo, “Genetic Programming based Illumination Robust and Non-parametric Multi-colors Detection Model,” *J. Korean Inst. Intell. Syst.*, vol. 20, no. 6, pp. 780–785, 2011.

[11] S. Whiteson, M. E. Taylor, and P. Stone, “Empirical Studies in Action Selection with Reinforcement Learning,” vol. 15, pp. 33–50, 2007.

[12] B. Abdulhai and L. Kattan, “Reinforcement learning: Introduction to theory and potential for transport applications,” *Can. J. Civ. Eng.*, vol. 30, no. 6, pp. 981–991, 2003.
[13] GNU Team, “GNU Scientific Library,” *Int. Urol. Nephrol.*, vol. 40, no. 1, pp. 249–253, 2008.
[14] T. Dou, Y. K. Lopes, and P. Rockett, “Model Predictive Control of Buildings Using Genetic Programming Dynamic Models,” no. April, 2019.
[15] X. Z. Lai, J. H. She, S. X. Yang, and M. Wu, “Comprehensive unified control strategy for underactuated two-link manipulators,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 2, pp. 389–398, 2009.
[16] C. Wongsathan and C. Sirima, “Application of GA to design LQR controller for an inverted pendulum system,” *2008 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2008*, no. 2, pp. 951–954, 2008.
[17] F. L. Lewis and D. Vrabie, “Adaptive dynamic programming for feedback control,” *Proc. 2009 7th Asian Control Conf. ASCC 2009*, pp. 1402–1409, 2009.

APPENDICES

A. Markov Swing-up controller

((AQ(AQ(1.4,x[2]),-(-(3.5))) + (4.8 + (x[3] + 3.4))) * ((AQ(1.4,-((7.4 * x[5]))) + (3.7 - x[5])) - ((((7.1 + 0.5) * x[5]) * AQ(3.2,(-(-((((x[4] - 1.1) + (x[3] - x[3])) - ((x[4] - x[3]) - (6.7 - 9.6))))) + AQ(AQ((x[2] + 0.7),-(-((-(3.9) - (0.7 - 5.0))))),AQ(AQ(-(AQ((0.7 * x[3]),-(5.1))),(x[1] * 8.5)),((x[2] - 1.3) - (2.7 * 1.0))))))) * AQ(3.2,3.7))))

B. Markov balance controller

(5.1 * ((5.1 * ((-(AQ((AQ(-((3.5 + x[1])),(x[5] * x[3])) + ((-(AQ(((x[2] * x[3]) - (AQ(3.8,2.4) + -(-(((4.4 * x[1]) * AQ(8.8,5.3)))))),((x[2] * 8.7) * -(7.8)))) + (((((x[3] - 4.8) + (7.8 + x[5])) * AQ((x[4] * x[1]),(((-(5.8) - (0.8 - x[2])) * (-(x[2]) - (4.3 + 0.9))) - x[3]))) - x[2]) * (9.2 + x[2]))) * ((((5.3 + x[1]) - -(x[4])) - 1.3) + x[4])),(x[4] * 8.4))) + (8.0 * x[4])) + (-(((1.3 - AQ(x[1],1.0)) * (9.2 + x[2]))) * ((x[2] * 8.7) * -(7.8))))) + ((6.1 * ((x[5] + x[2]) * ((-(-((9.4 - x[1]))) * -(0.3)) - ((x[4] - 9.0) - AQ((4.2 * ((1.3 - AQ(x[1],1.0)) * (9.2 + x[2]))),((x[2] * (((-(-((7.4 + 3.7))) - x[3]) * (8.1 + x[5])) - -((x[2] - -((x[4] * x[1])))))) - -((((x[5] - 6.9) - (x[4] - 3.3)) * ((AQ(((x[3] - x[5]) * (3.9 - 4.5)),((5.6 + 8.0) + -(3.6))) * x[1]) * AQ(x[4],9.0)))))))))) + -(AQ(((x[1] * 0.1) + (x[3] + x[2])),AQ(AQ(AQ(5.4,x[2]),9.0),(x[1] * -(0.3))))))))

C. Non-Markov balance controller

((q2(q3(((AQ(9.3,AQ((AQ(x[1],0.6) * (4.5 * x[5])),((x[3] - 1.9) + (8.7 - x[2])))) + 0.5) + -((x[1] * x[4])))))) + ((1.9 + (((6.9 - x[5]) * (x[3] * x[5])) - -((3.5 - x[1])))) + (q3(9.3) + x[5]))) * ((q2(0.8) + q1(q1(AQ(q3((6.9 - x[5])),q3((x[2] - 3.1)))))) - (7.4 * x[5])))

D. Non-Markov balance controller

AQ(((9.8 + q3(AQ(q2(1.1),AQ(((q2(AQ((x[3] + x[5]),((((x[1] + (((0.8 + x[2]) + q2(3.7)) + -((x[5] + 5.5)))) + (x[2] + 8.4)) * ((5.6 + 1.8) - AQ(7.6,x[2]))) - x[2])))) + (0.9 * x[2])) + (-(4.8) - (0.2 - 2.4))),-(q3(q3(x[4])))))) * ((AQ((x[3] * 5.5),(1.9 - ((((AQ(x[2],5.2) * -(x[5])) * -((x[1] - (x[1] + x[4])))) * -(-6.5)) * -(x[5])))) + ((x[4] + 8.9) + (x[1] * (((-(x[1]) - (2.5 * 5.9)) * (-(AQ(AQ((6.9 + x[1]),((((q1(0.7) * AQ(4.0,(AQ((AQ((6.7 * 2.1),AQ(x[4],x[3])) * x[5]),4.3) + AQ(AQ(x[1],5.9),(7.1 - x[3]))))) - (2.7 - x[3])) - AQ((x[2] * (x[3] * x[2])),q2(q1(q1(x[3])))))) - (AQ((q2(AQ(x[4],x[2])) + (q1(x[3]) + AQ(x[5],x[2]))),(-(((AQ(5.9,x[3]) + (x[4] + 5.2)) * ((0.4 + 0.4) - (x[2] * 0.8)))) - (x[4] + x[5])))) - (q2((0.4 - x[4])) * (0.3 + x[4]))))),(0.2 - AQ(AQ(AQ(6.9,AQ(AQ(AQ(4.5,x[1]),AQ(8.5,5.1)),((3.5 - 4.0) + (x[5] - 8.2)))),(x[5] - 1.3)),-(((x[3] * x[4]) * AQ(x[1],x[2])))))) + AQ(x[4],5.2))) * 7.5)))) * 9.4)),AQ((x[4] + 3.5),(AQ((8.0 * 9.4),(AQ(x[4],(((x[2] * 0.3) * -(9.0)) + AQ(((3.0 + x[2]) + AQ(2.7,x[3])),(q2(0.3) + AQ(3.0,AQ(x[4],5.2)))))) + AQ(((AQ((6.7 * 2.1),AQ(q2((2.5 * 5.9)),x[3])) * x[5]) * ((((x[4] - 9.0) + -(AQ(x[5],7.6))) - (AQ(AQ(((AQ((x[3] * 5.5),(1.9 - ((q1(0.7) * ((1.6 + x[1]) - AQ(x[2],1.5))) * -(x[5])))) + ((x[4] + 8.9) + (x[1] * (((-(x[1]) - (2.5 * 5.9)) * (-(AQ(AQ((6.9 + x[1]),x[3]),q3((x[4] + 3.5)))) + AQ(x[4],5.2))) * 7.5)))) - (7.1 * 9.7)),((4.4 - x[5]) + (((((x[4] - 9.0) + AQ(x[5],4.7)) - (AQ(0.7,6.7) - (8.7 + x[5]))) * 4.7) * 6.5))),-(AQ(x[1],q1(1.1)))) - (8.7 + (((x[5] * x[5]) * (1.9 - 1.4)) - (AQ(AQ(((6.2 + 4.5) + -(q3(q3(x[4])))),AQ((AQ(x[1],3.0) + AQ(q2((AQ(6.3,x[4]) - (x[4] * x[2]))),x[2])),((x[3] - 4.8) - (2.7 * x[3])))),x[4]) * (x[2] * x[3])))))) * 4.7)),(4.7 + AQ(((1.6 + x[1]) - AQ(x[2],AQ(AQ((2.7 * x[4]),AQ(x[1],q1(1.1))),((x[3] * x[5]) * q2(x[2])))))),AQ(((((x[3] - x[4]) * (3.7 * x[5])) + (AQ(3.8,9.7) * q2(8.4))) * 3.6),(1.4 + 9.5)))))) + (q1(((2.5 - 0.7) - AQ(x[1],7.7))) - -(6.5)))))