Probability Redistribution using Time Hopping for Reinforcement Learning

Petar S. Kormushev, Fangyan Dong, and Kaoru Hirota
Department of Computational Intelligence and Systems Science
Tokyo Institute of Technology
Yokohama, Japan
Email: {petar, tou, hirota}@hrt.dis.titech.ac.jp

***Abstract*—A method for using the Time Hopping technique as a tool for probability redistribution is proposed. Applied to reinforcement learning in a simulation, it is able to re-shape the state probability distribution of the underlying Markov decision process as desired. This is achieved by modifying the target selection strategy of Time Hopping appropriately. Experiments with a robot maze reinforcement learning problem show that the method improves the exploration efficiency by re-shaping the state probability distribution to an almost uniform distribution.**

***Index Terms*—Reinforcement learning, Markov decision process, probability redistribution, simulation, discrete time systems.**

I. Introduction

Reinforcement learning (RL) algorithms [21] address the problem of learning to select optimal actions when limited feedback (usually in the form of a scalar reinforcement function) from the environment is available. Many different action selection methods exist for Reinforcement Learning [8], and a variety of successful practical applications have been reported [9]. One main reason for the popularity of RL is that it is very similar to the natural way of learning to perceive and act by trial and error [25].

General RL algorithms like Q-learning [24], SARSA and TD(λ) [20] have been proved to converge to the globally optimal solution (under certain assumptions) [6][24]. They are very flexible, in the sense that they do not require a model of the environment and have been shown to be effective in solving a variety of RL tasks. This flexibility, however, comes at a certain cost: these RL algorithms require extremely long training to cope with large state space problems [23]. Even for a relatively simple control task such as the cart-pole balancing problem on a limited-length track, tens of thousands of steps are necessary [7][4].

Many different approaches have been proposed for speeding up the RL process. One possible technique is to use function approximation [19], in order to reduce the effect of the "curse of dimensionality". Unfortunately, using function approximation creates instability problems when used with off-policy learning [3].

Significant speed-up can be achieved when a demonstration of the goal task is available [15][5] , as in Apprenticeship Learning [14]. Although there is a risk of running dangerous exploration policies in the real world [2], successful implementation of apprenticeship learning for aerobatic helicopter flight exists [1].

A state-of-the-art RL algorithm for efficient state space exploration is E3 [10]. It uses active exploration policy to visit states whose transition dynamics are still inaccurately modeled. Because of this, running E3 directly in the real world might lead to a dangerous exploration behavior.

Instead of using value-iteration-based RL algorithms, some researchers have focused on completely different algorithms, the so-called policy search RL algorithms [16]. Examples include the Natural Actor-Critic architecture [17], as well as the Policy Gradient RL algorithm [18], which has been applied successfully to robot control [11]. An alternative way to represent states and actions also exists, known as Relational Reinforcement Learning [22], which generalizes RL by relationally representing states and actions.

Instead of executing RL algorithms in the real world, simulations are commonly used. This approach has two main advantages: speed and safety. Depending on its complexity, a simulation can run many times faster than a real-world experiment. Also, the time needed to set up and maintain a simulation experiment is often less compared to a real-world experiment. The second advantage, safety, is also very important, especially if the RL agent is a very expensive equipment (e.g. a fragile robot), or a dangerous one (e.g. a chemical plant).

Whether the full potential of computer simulations has been utilized for RL, however, is an open question. A new trend in RL suggests that this might not be the case. For example, two techniques have been proposed recently to better utilize the potential of computer simulations for RL: Time Manipulation [12] and Time Hopping [13]. They share the concept of using the simulation time as a tool for speeding up the learning process.

The first technique, called Time Manipulation, suggests that doing backward time manipulations inside a simulation can significantly speed up the learning process and improve the state space exploration. Applied to failure-avoidance RL problems, such as the cart-pole balancing problem, Time Manipulation has been shown to increase the speed of convergence by 260% [12].

This work was supported in part by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT).

The second technique, called Time Hopping, can be applied successfully to continuous optimization problems. Unlike the Time Manipulation technique, which can only perform backward time manipulations, the Time Hopping technique can make arbitrary "hops" between states and traverse rapidly throughout the entire state space. It has been shown to accelerate the learning process more than 7 times on some problems [13]. Time Hopping possesses mechanisms to trigger time manipulation events, to make prediction about possible future rewards, and to select promising time hopping targets.

This paper focuses on the second technique: Time Hopping. The original concept of Time Hopping is to use the simulation time as a tool for speeding up the learning process. This paper proposes a new concept: of using Time Hopping as a tool for probability redistribution, i.e. re-shaping the state probability distribution as desired. It can be achieved by changing the Time Hopping target selection strategy appropriately, as explained in Section II.

In order to evaluate the proposed approach, experiments with a robot maze RL problem are conducted. The task is for the robot to find the shortest path from the start to the goal inside the maze without hitting the walls. The experiments show that probability redistribution allows for more efficient learning by doing more purposeful exploration during the training.

The following Section II makes a brief overview of the Time Hopping technique and outlines the way it can be used for probability redistribution. Section III proposes a concrete method to implement probability redistribution by modifying a component of the Time Hopping technique. Section IV presents the results from experimental evaluation of the proposed probability redistribution method on a robot maze RL problem.

II. Probability Redistribution using Time Hopping

A. Overview of Time Hopping

Time Hopping is an algorithmic technique which allows maintaining higher learning rate in a simulation environment by hopping to appropriately selected states [13]. For example, let us consider a formal definition of a RL problem, given by the Markov Decision Process (MDP) on Fig. 1. Each state transition has a probability associated with it. State 1 represents situations of the environment that are very common and learned quickly. The frequency with which state 1 is being visited is the highest of all. As the state number increases, the probability of being in the corresponding state becomes lower. State 4 represents the rarest situations and therefore the most unlikely to be well explored and learned.

When applied to such a MDP, Time Hopping creates "shortcuts in time" by making hops (direct state transitions) between very distant states inside the MDP. Hopping to low-probability states makes them easier to be learned, while at the same time it helps to avoid unnecessary repetition of already well-explored states [13]. The process is completely transparent for the underlying RL algorithm.

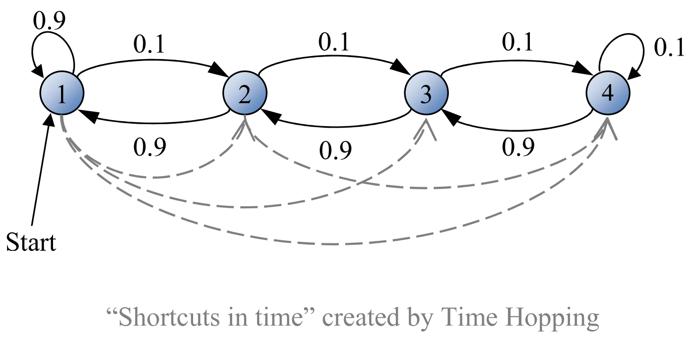


Fig. 1. An example of a MDP with uneven state probability distribution. Time Hopping can create "shortcuts in time" (shown with dashed lines) between otherwise distant states, i.e. states connected by a very low-probability path. This allows even the lowest probability state 4 to be learned easily.

B. Probability Redistribution using Time Hopping

The proposed method for using the Time Hoping technique for probability redistribution is to provide "shortcuts in time" to such low-probability states, making them easier to learn, while at the same time avoiding unnecessary repetition of already well-explored states. This is done by externally manipulating the computer simulation in a way which is completely transparent for the RL algorithm, as demonstrated in Section III.

The "shortcuts in time" are created by direct hops between distant states of the MDP. Depending on how it is used, Time Hopping can change the state probability distribution to, for example, an almost uniform distribution. In this way all the states can be visited (and therefore, learned) almost equally well. Fig. 2 shows what would be the effect of Time Hopping when applied to the same MDP from Fig. 1.

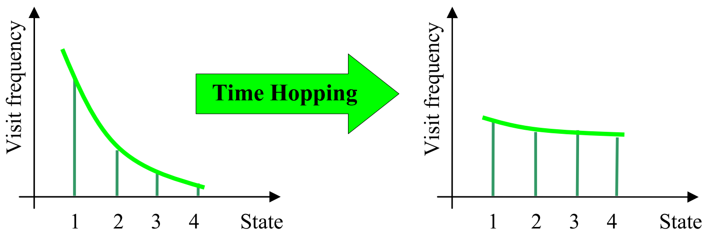


Fig. 2. Time Hopping can change the state probability distribution to an almost uniform distribution. The 4 states shown correspond to the same 4 states from Fig. 1.

The remaining question is how to efficiently implement probability redistribution using Time Hopping for RL problems. Section III offers a possible answer to this question.

III. Implementation of probability redistribution

This section proposes a concrete method to implement probability redistribution by modifying a component of the Time Hopping technique.

A. Components of Time Hopping

When applied to a conventional RL algorithm, the Time Hopping technique consists of 3 components:

1) Hopping trigger – decides when the hopping starts;
2) Target selection – decides where does it hop to;
3) Hopping – performs the actual hopping.

The flowchart on Fig. 3 shows how these 3 components of Time Hopping are connected and how they interact with the RL algorithm.

When the Time Hopping trigger is activated, a target state and time are selected by the target selection component. After that, hopping can be performed. It includes setting the RL agent and the simulation environment to the proper state, while at the same time preserving all the acquired knowledge by the agent.

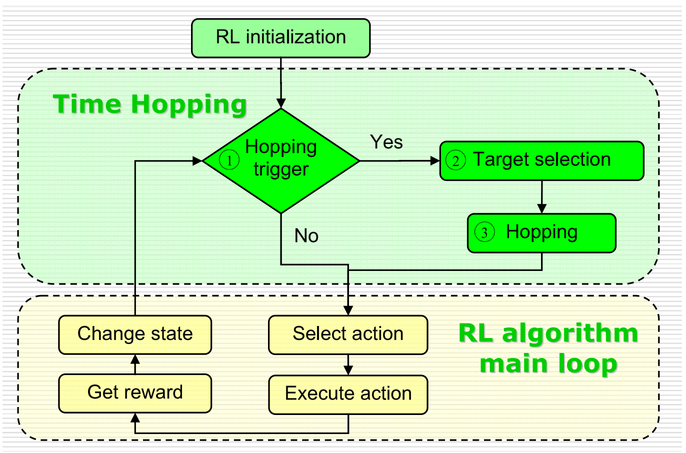


Fig. 3. Time Hopping technique applied to a conventional RL algorithm. The lower group (marked with a dashed line) contains the conventional RL algorithm main loop, into which the Time Hopping components (the upper group) are integrated.

The proposed implementation of probability redistribution is based on modifying the target selection component of Time Hopping. Many relevant properties of the already explored states can be considered, such as probability, visit frequency, level of exploration, connectivity to other states (number of state transitions), etc. In this particular implementation, we use the visit frequency of states to implement a "Least explored first" target selection policy. By maintaining the number of times each explored state was visited, it is easy to keep all states sorted by it and select hopping targets among states with low visit frequency. This way the exploration can be purposefully redirected to less explored areas of the state space which, in effect, equalizes the state probability distribution. Table I lists the proposed implementation for each component. The "Gamma pruning" and the "Basic hopping" components are implemented as described in [13].

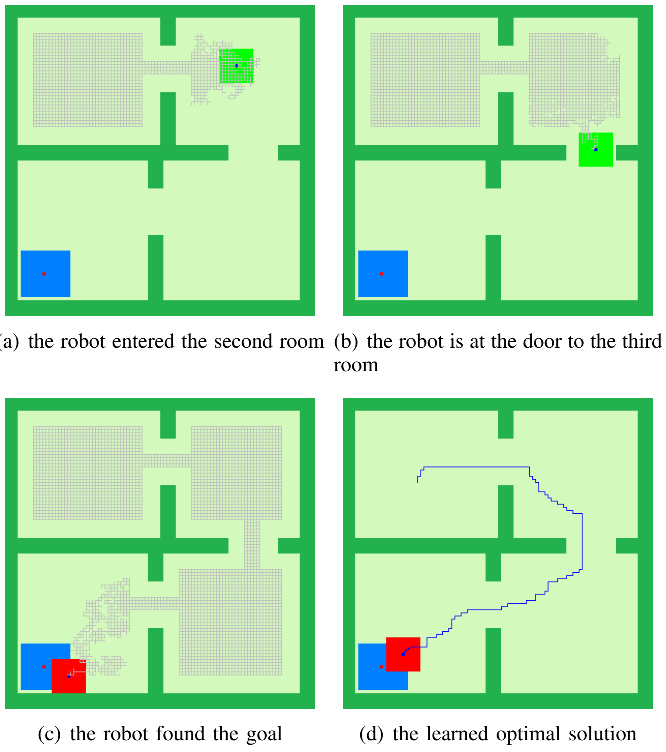
In the proposed implementation, Q-learning [24] is used as the underlying off-policy RL algorithm, in order to guarantee the convergence (as explained in [13]).

TABLE I
PROPOSED IMPLEMENTATION OF EACH TIME HOPPING COMPONENT FOR PROBABILITY REDISTRIBUTION.

	Component name	**Proposed implementation**
1	Hopping trigger	"Gamma pruning"
2	Target selection	"Least explored first"
3	Hopping	"Basic hopping"

IV. Evaluation of Probability Redistribution

This section presents the results from experimental evaluation of the proposed probability redistribution method on a robot maze RL problem. The goal of the experiment is to use probability redistribution to achieve more efficient learning by doing more purposeful exploration during the training.


(a) the robot entered the second room (b) the robot is at the door to the third room

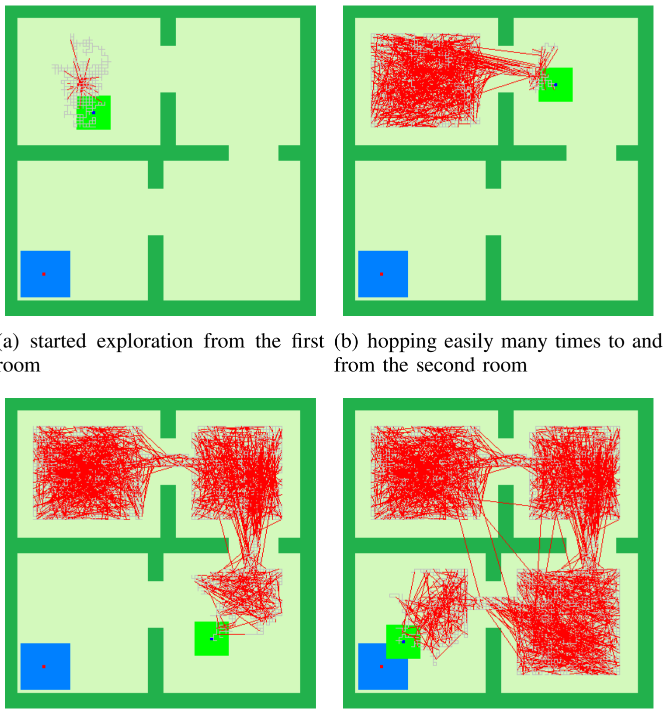
(c) the robot found the goal (d) the learned optimal solution

Fig. 4. The robot maze RL problem. The robot (smaller square) starts from the upper-left corner and is trying to reach the goal in the lower-left corner (bigger square). There are four "rooms" in the maze, with narrow "doors" between them, which reduces the probability of moving from one room to another.

A. The robot maze RL problem

The task is for the robot to find the shortest path from the start to the goal inside the maze without hitting the walls, as shown on Fig. 4. There are four "rooms" in the maze, with narrow "doors" between them, which reduces the probability of moving from one room to another. The four rooms resemble the 4-state MDP from Fig. 1, except that this time the probability of moving from one state to another is very small on both directions. The state space is partitioned in 10000 discrete states. The robot has 4 different actions -

moving up, down, left, and right. The conventional Q-learning algorithm needs around 80000 steps of training to find the goal. Selected moments of one such training is shown on Fig. 4. Using exactly the same settings, when probability redistribution is enabled using the Time Hopping technique, only around 17000 steps are enough to find the goal. This means that probability redistribution improves the efficiency of learning almost 5 times. The mentioned results are averaged results of 10 trials. Fig. 5 shows selected moments of the training when probability redistribution is enabled.


(a) started exploration from the first room

(b) hopping easily many times to and from the second room

(c) entered the third room before completely exploring the second one

(d) found the goal

Fig. 5. The robot maze RL problem with probability redistribution enabled. The straight lines show the hopping transitions which were done by the Time Hopping technique. These direct hopping transitions between distant states do not have to obey the physics laws of the simulation, which explains why some of them cross the walls of the maze.

In order to compare the state probability distribution with and without Time Hopping, we use the state visit frequency (number of times each state was visited per trial). Fig. 6 shows the comparison of the two corresponding experiments from Fig. 4 and Fig. 5 based on the state visit frequency. The experimental results confirm the expected effect of Time Hopping, as they follow closely the theoretically predicted effect from Fig. 2. Using probability redistribution, we achieved almost uniform state probability distribution. Also, the total number of visits when probability redistribution is used is significantly smaller than without probability redistribution. This is due to the improved exploration efficiency from reducing the redundant exploration, which also leads to reduced execution time.

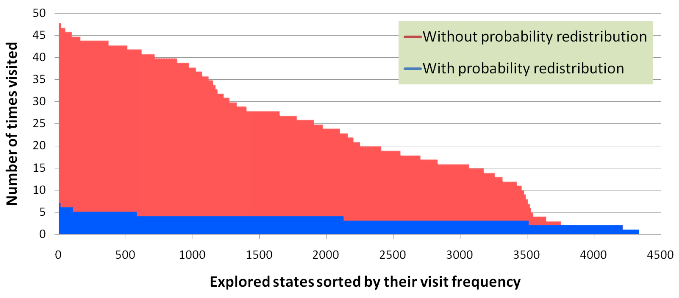


Fig. 6. A comparison of the state visit frequency with and without probability redistribution. All the explored states are sorted in decreasing visit frequency. This figure reflects the state at the time of reaching the goal by each algorithm.

V. Conclusion

A method for using the Time Hopping technique as a tool for probability redistribution is proposed. Applied to Reinforcement Learning in a simulation, it is able to re-shape the state probability distribution of the underlying MDP as desired. This is achieved by modifying the target selection strategy of Time Hopping appropriately.

The conducted experiments with a robot maze RL problem show that probability redistribution allows for more efficient learning by doing more purposeful exploration during the training. This is a very important advantage of the proposed approach, especially when the simulation involved is computationally expensive. In this case, probability redistribution using Time Hopping can save computational time by reducing the number of simulation steps in favor of Time Hopping steps and redirecting the exploration where it is most needed.

It is important to note that the proposed method does not directly change the transitions probabilities of the underlying MDP. Instead, the Time Hopping technique creates virtual “shortcuts in time”, thus making direct hops between distant states of the MDP. These hops are not part of the original MDP and they do not follow the physics laws of the simulation involved. These hopping transitions are the key to achieving state probability redistribution. They redirect the exploration to parts of the state space where further exploration is most desired.

The fact that the original MDP transitions probabilities are preserved intact means that the proposed method is transparent for the MDP and the simulation implementations. Therefore, it could be integrated seamlessly into other simulation-based RL problem solvers which also use MDP to model the state transitions.

References

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *In Advances in Neural Information Processing Systems 19*, volume 19. MIT Press, 2007.

[2] P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *Proceedings 21st International Conference on Machine Learning*, pages 1–8. ICML, 2005.

[3] S. T. Anton and A. Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*. Erlbaum, 1993.

[4] R. S. Bapi, B. D'Cruz, and G. Bugmann. Neuro-resistive grid approach to trainable controllers: A pole balancing example. *Neural Computing and Applications*, 5:33–44, 1997.

[5] A. Coates, P. Abbeel, and A. Y. Ng. Learning for control from multiple demonstrations. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 144–151, New York, NY, USA, 2008. ACM.

[6] P. Dayan and T. J. Sejnowski. Td() converges with probability 1. *Machine Learning*, 14, No. 3:295–301, 1994.

[7] S. Geva and J. Sitte. The cart-pole experiment as a benchmark for trainable controllers. Technical report, Australia, 1993.

[8] M. Humphrys. *Action Selection methods using Reinforcement Learning*. PhD thesis, University of Cambridge, June 1997.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[10] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. In *Machine Learning*, pages 260–268. Morgan Kaufmann, 1998.

[11] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2619–2624, 2004.

[12] P. Kormushev, K. Nomoto, F. Dong, and K. Hirota. Time manipulation technique for speeding up reinforcement learning in simulations. *International Journal of Cybernetics and Information Technologies*, 8, No. 1:12–24, 2008.

[13] P. Kormushev, K. Nomoto, F. Dong, and K. Hirota. Time hopping technique for faster reinforcement learning in simulations. *IEEE Transactions on Systems, Man and Cybernetics part B*, submitted in January, 2009.

[14] A. Ng. Reinforcement learning and apprenticeship learning for robotic control. *Lecture Notes in Computer Science*, 4264:29–31, 2006.

[15] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. learning and generalization of motor skills by learning from demonstration. In *international conference on robotics and automation (icra2009)*, 2009.

[16] L. Peshkin. *Reinforcement Learning by Policy Search*. PhD thesis, MIT, November 2001.

[17] J. Peters and S. Schaal. Natural actor-critic. *Neurocomput.*, 71(7-9):1180–1190, 2008.

[18] J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement Learning for Humanoid Robotics. In *Humanoids2003, 3rd IEEE-RAS International Conference on Humanoid Robots, Karlsruhe*, 2003.

[19] D. Precup, R. S. Sutton, and S. Dasgupta. Off-policy temporal-difference learning with function approximation. In *Proceedings 18th International Conf. on Machine Learning*, pages 417–424. Morgan Kaufmann, San Francisco, CA, 2001.

[20] R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[22] P. Tadepalli, R. Givan, and K. Driessens. Relational reinforcement learning: An overview. In *Proceedings of the ICML-2004 Workshop on Relational Reinforcement Learning*, pages 1–9, 2004.

[23] S. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Computer Science Department, Pittsburgh, PA, 1992.

[24] C. J. C. H. Watkins and P. Dayan. Technical note: q-learning. *Mach. Learn.*, 8(3-4):279–292, 1992.

[25] S. D. Whitehead and D. H. Ballard. Learning to perceive and act by trial and error. *Mach. Learn.*, 7(1):45–83, 1991.