خوارزمية المحاولة والخطأ بالاعتماد على اختيار الشجرة (Tree-select) للتأقلم مع الأعطال في الروبوتات ذات الفائضية

علي ديب[1]، عهد البدين[2]، د. شادي بيطار[3]

[1]المعهد العالي للعلوم التطبيقية والتكنولوجيا، نظم إلكترونية. إيميل: ali.abddeeb@hiast.edu.sy

[2] المعهد العالي للعلوم التطبيقية والتكنولوجيا، ماجستير تحكم وربوتيك.
إيميل: ahed.albadin@hiast.edu.sy

[3] دكتور في المعهد العالي للعلوم التطبيقية والتكنولوجيا، رؤية وربوتيك.
إيميل: shadi.albitar@hiast.edu.sy

تاريخ الإيداع:2023/8/29
تاريخ القبول:2023/10/24

الملخص:

نعرض في هذا البحث خوارزمية جديدة نسميها (Tree-select Trial and Error" TTE) والتي يمكن أن تساعد الذراع الآلية التي تتضمن فائضية على التأقلم مع الأعطال التي قد تحدث أثناء عملها. تقترح هذه الخوارزمية استراتيجية بحث جديدة تسمح للروبوت بتوليد سلوكيات جديدة، بدلاً من التقيد بالسلوكيات التي تم تعلمها سابقاً من أجل تعويض آثار الضرر. وبالتالي، أصبح تعلم السلوكيات المناسبة عملية مستمرة تعزز معارف الروبوت. أثبتت المحاكاة والنتائج التجريبية على ذراع مستوٍ بست محركات فعالية خوارزميتنا في مساعدة الروبوت على الوصول إلى مساحة كبيرة من مساحة عمله في عدد قليل من المحاولات على الرغم من سيناريوهات الأعطال المختلفة.

الكلمات المفتاحية: التأقلم مع الأعطال، المحاولة والخطأ، الروبوتيك، الأذرع الآلية.

Tree-select Trial and Error Algorithm for Adaptation to Failures of Redundant Manipulators

Ali DEEB[1], Ahed ALBADIN[2*], Dr. Chadi ALBITAR[3]

[1] Higher Institute for Applied Sciences and Technology, Engineer, Electronic Systems, Email: ali.abddeeb@hiast.edu.sy
[2] Higher Institute for Applied Sciences and Technology, Engineer, Control and Robotics, Email: ahed.albadin@hiast.edu.sy
[3] Higher Institute for Applied Sciences and Technology, Assistant Professor vision and robotics, Email: shadi.albitar@hiast.edu.sy

Received: 29/8/2023
Accepted: 24/10/2023

Abstract

In this paper, we introduce a novel algorithm we call "Tree-select Trial and Error" (TTE) that can help a redundant robotic arm to adapt to failures that might occur during its functioning. This algorithm proposes a new search strategy allowing the robot to generate new behaviours, rather than being restricted to the previously learned ones, in order to compensate damage effects. Thus, learning suitable behaviours become an online process that promotes the robot's knowledge. The simulation and the experimental results on a planar manipulator with six actuators proved the effectiveness of our algorithm in helping the robot to reach a large area of its workspace in a few number of trials despite the different damage scenarios.

Keywords: Adaptation to Failure, Trial and error, Robotics, Manipulators

1. Introduction

Robots are physical systems equipped with actuators permitting sufficient degrees of freedom to perform desirable tasks within certain environments. Like any physical system, damages can occur badly causing serious challenges with common fragile robots (Sanderson, 2010), (Carlson & Murphy, 2005), especially when the tasks are performed outside the laboratories, or when human interference is not a valid option.

Traditional damage recovery strategies involved two main phases; fault diagnosis followed by selecting the most promising pre-specified solution (Blanke, *et al.*, 2006). Such self-diagnosing robots were expensive because of the high price sensors and design complexity. However, it is difficult to foresee all possible damage scenarios.

Recently, these challenges were tackled by adaptation to failure, skipping the phase of fault diagnosis through trial and error (Cully, *et al.*, 2015), (Koos, *et al.*, 2013), and (Ren, *et al.*, 2015). As a result, damage recovery has become a Reinforcement Learning (RL) problem in which the robot needs to maximize a certain reward function consulting the best action in each state despite the damage (Kober, *et al.*, 2013). Traditional RL methods learn to choose the best action from each state (Sutton, *et al.*, 1998), (Mnih, *et al.*, 2015).

Thus, RL algorithms rely mostly on policies, rather than value functions, to estimate the parameters that lead to better actions. These methods were used for dynamic movement primitives (Ijspeert, *et al.*, 2002) and general-purpose neural networks in robotics field (Levine & Koltun, 2013). Direct policy search adopts algorithms that learn through optimization problems using gradient-based or black-box optimization algorithms (Stulp & Sigaud, 2013). However, these algorithms encountered difficulties due to the high-dimensionality of search spaces (Deisenroth, *et al.*, 2013). Model-based policy search algorithms tend to differentiate between learning a model of the robot and learning a proper policy concerning the learned model (Deisenroth, *et al.*, 2013) (Chatzilygeroudis, *et al.*, 2017). Another adaptation to damage algorithm (DA-PPO) was proposed combining damage diagnosis with deep reinforcement learning (Verma, *et al.*, 2020). It used Long Short Term Memory based supervised learning network to diagnose the damage, and predict the best action through a single shot.

The former algorithms were proved to be very data-sufficient, since they require fewer trials to learn policies. Adaptation in dynamic real-world environments was achieved in recent researches through meta-reinforcement learning (Nagabandi, *et al.*, 2019). However, the complexity of dynamics scales exponentially with the number of components moving as large number of observations were required from the real system to learn a useful model. Mitigating this limitation was achieved by FAMLE algorithm (Kaushik, *et al.*, 2020) that meta-trained several initial starting points allowing the robot to select the most promising starting point to adapt to the current situation.

Intelligent Trial and Error (IT&E) algorithm was proposed in (Cully, *et al.*, 2015), and permitted a robotic manipulator, as well as a hexapod robot, to perform well in cases of some damages. It is consisted of two main phases; building a behaviour-performance map, followed by searching only within the previously stored behaviours in cases of damage. The last phase made the adaptation to failure restricted to what the robot has previously learned and stored in the map, unable to learn new behaviours. In (Kume, *et al.*, 2017), MMPRL algorithm allowed storing behaviours more efficiently. However, the adaptation process were strictly held by the stored behaviours solely. Besides, after each trial, the robot needed to be reset to the initial state. This problem was solved by RTE algorithm (Chatzilygeroudis, *et al.*, 2018), which did not need any reset between episodes and scaled well with respect to the dimensionality of state-action spaces, which made it suitable for complex robots, as it takes into account the environment changes. RTE proved to make a break through the adaptation to failure algorithms for robots and it was tested on a six-legged walking robot and made it able to recover most of its locomotion abilities in an environment with obstacles without any human intervention. However, RTE is applicable when the model of the robot is defined under the form of states and transitions.

Furthermore, APROL algorithm was proposed recently allowing redundant robots to enlarge their behaviour repertoire by allowing the simulation model to generate repertoires for many different situations concerning damaged cases (Kaushik, *et al.*, 2020). So, the robot extends its learning phase and starts learning to face some damage scenarios before they happen. This modification resulted in faster adaptation for a damaged hexapod robot.

Recently, Albadin, *et al.*, ensured the effeciency of the methods that does not rely on diagnosing the failure (Albadin, *et al.*, 2022) by developping the (IT&E) algorithm adding a coefficient related to the distance between the robot's position and the optimal trajectory. The simulation on a mobile hexapod robot proved that the quality of the chosen behaviours outperformed the previous one by four time. Also, the chosen behaviour did not relied on the damaged leg.

In this paper, we introduce a novel algorithm we call "Tree-select Trial and Error" (TTE). The novelty of our algorithm is in the adaptation phase as the robot generates new behaviours to compensate the effects of the damages in order to reach goals within its workspace after a few numbers of trials regardless the damage scenarios. Thus, learning suitable behaviours is an online process during which the robot extends its knowledge. The main contributions of this paper are as follows:

- A novel adaptation to failure strategy during which the robot applies an online search within the whole behavioural space to compensate the damage effects.
- Extensive simulation experiments of a planar manipulator with six actuators for workspace points considering many damage scenarios to achieve statistical comparison between the proposed TTE, IT&E and Traditional Bayesian Search.
- Experimental validation on a real planar manipulator with six actuators to reach a diverse set of plan points for different damage scenarios.

2. Proposed Algorithm

The introduced algorithm (TTE) suggests searching within the high-dimensional behavioural space. However, this search is subjected to "Tree-select" strategy that perform a votes method on the actuators allowing a series of them to conduct a move. In addition, predicting new behaviours is guided through a strategy we call "Guided Bayesian Search" (GBS), that helps focusing the search, during each step, on the neighbourhood of the best solution resulted from previous steps.

To perform the adaptation, the search tree is structured as the following:

- The tree depth is equal to the number of actuators (n).
- The node at the $i^{(th)}$ level describes a behaviour of the $i^{(th)}$ motor.
- Search strategy is divided into n steps.
- At the $i^{(th)}$step: a decision concerning actuators from i to n is to be made.

The search strategy is illustrated in (Figure 1). This strategy helps reducing the mean number of search space dimensions, so that the robot does not need to search for a controller of length n in every single step.

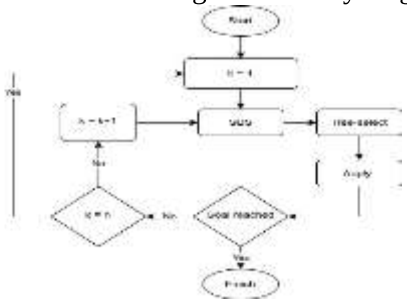


Figure (1) Proposed search strategy.

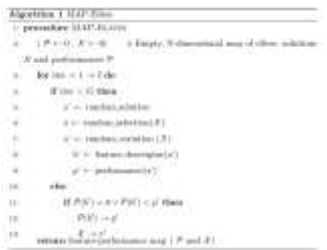
2.1 Learning the action repertoire

Solving the equations of the inverse kinematic model for redundant robots is challenging in most cases. Circumventing this probelm was adressed by using MAP-Elites algorithm (Algorithm 1) that builds a multi-dimensional array mapping between the two spaces of performances and behaviours (Mouret & Clune, 2015).

This process starts by searching randomly in the high-dimensional behavioural space and storing valid behaviours, then searching within the pre-stored behaviours to derive new ones. A behaviour is chosen to be stored once it meets certain criteria.

2.2 Learning with a Gaussian Process

A Gaussian Process is a stochastic process describing a collection of random variables indexed by time or space, such that every finite collection of those random variables has a multivariate normal distribution specified by mean m and standard deviation k (Rasmussen & Nickisch, 2010).



```
Algorithm 1 MAP-Elites
1: procedure MAP-ELITES
2:     (P ← ∅, X ← ∅)        ▷ Empty, N-dimensional map of elites: solutions X and performances P
3:     for iter = 1 → I do
4:         if iter < G then
5:             x' ← random_solution()
6:         else
7:             x ← random_selection(X)
8:             x' ← random_variation(x)
9:         b' ← feature_descriptor(x')
10:        p' ← performance(x')
11:        if P(b') = ∅ ∨ P(b') < p' then
12:            P(b') ← p'
13:            X(b') ← x'
    return feature-performance map (P and X)
```

$$f(x)\sim GP(m(x),k(x,x')) \qquad (1)$$

The prediction process depends on Bayes's theorem (Smets, 1993) which states that the posterior probability of a model M given evidence E is proportional to the likelihood of E given M multiplied by the prior probability of M (Brochu, et al., 2010):

$$P(E) \propto P(M)P(M) \qquad (2)$$

A Gaussian Process is used to predict the best behaviour that can lead to every desired goal.

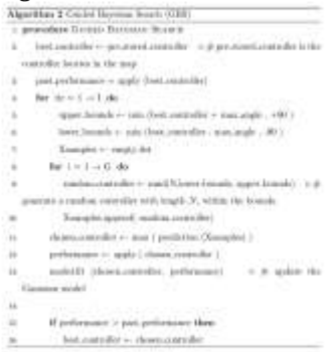
2.3 Guided Bayesian Search

After the $i^{(th)}$ trial, pre-tested behaviours in trials $1, 2, \dots, i-1$ are sorted to select the one with the best performance, let's call it "best_behaviour". During the $(i+1)^{(th)}$trial, the robot selects the most promising behaviour within the "angular neighbourhood" of "best_behaviour". This methodology allows the robot to benefit most from the knowledge it gains through trial and error. However, this strategy does not restrict the search space to one neighbourhood since a vast transitions can be done, but it requires more than a single step. The strategy of GBS is illustrated in (Algorithm 2).

To make a fine comparison, GBS was tested on the planar manipulator as a trial and error algorithm. That allows illustrating the value added by both GBS and our proposed TTE in minimizing the number of trials needed by the robot to adapt to damage scenarios.

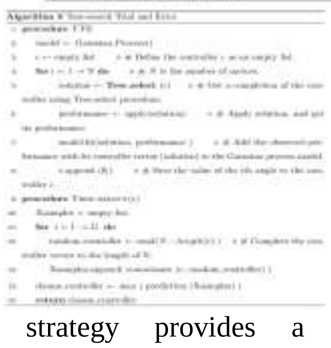
2.4 Probabilistic Optimal Planning using tree search

At the beginning of each episode, the controller vector should be chosen using a Gaussian Process that holds all the information gained from previous trials. This choice is made by searching within a neighbourhood of the previously chosen controller with the best performance based on the prediction of lower error (Euclidean distance) between the actual state and the goal.



Algorithm 2 Guided Bayesian Search (GBS)

```
1:  procedure GUIDED BAYESIAN SEARCH
2:      best_controller ← pre_stored_controller    ▷ # pre_stored_controller is the controller located in the map
3:      past_performance ← apply (best_controller)
4:      for itr ← 1 → I do
5:          upper_bounds ← min (best_controller + max_angle , +90 )
6:          lower_bounds ← max (best_controller - max_angle , -90 )
7:          Xsamples ← empty list
8:          for i ← 1 → G do
9:              random_controller ← rand(N,lower_bounds, upper_bounds)    ▷ # generate a random controller with length N, within the bounds
10:             Xsamples.append( random_controller)
11:         chosen_controller ← max ( prediction (Xsamples) )
12:         performance ← apply ( chosen_controller )
13:         model.fit (chosen_controller, performance)    ▷ # update the Gaussian model
14:
15:         if performance > past_performance then
16:             best_controller ← chosen_controller
```



Algorithm 3 Tree-select Trial and Error

```
1:  procedure TTE
2:      model ← (Gaussian Process)
3:      c ← empty list    ▷ # Define the controller c as an empty list
4:      for i ← 1 → N do    ▷ # N is the number of motors
5:          solution ← Tree-select (c)    ▷ # Get a completion of the controller using Tree-select procedure
6:          performance ← apply(solution)    ▷ # Apply solution and get the performance
7:          model.fit(solution, performance )    ▷ # Add the observed performance with its controller vector (solution) to the Gaussian process model
8:          c.append ($θ_i$)    ▷ # Store the value of the ith angle to the controller c
9:  procedure TREE-SELECT(c)
10:     Xsamples = empty list
11:     for i ← 1 → L do
12:         random_controller ← rand(N - length(c))    ▷ # Complete the controller vector to the length of N
13:         Xsamples.append( concatenate (c, random_controller) )
14:     chosen_controller ← max ( prediction (Xsamples) )
15:     return chosen_controller
```

This strategy provides a remarkable performance where the endpoint managed to reach a diverse set of plan points within few trials exceeding the performance of previous algorithms with a fewer number of trials and a larger reachable area. The strategy of TTE is clarified in (Algorithm 3).

3. Results

3.1 Simulation results

To highlight the performance of our proposed algorithm TTE, we considered the model of a planar manipulator with six actuators. Each of the four algorithms: Traditional Bayesian Search (TBS), Intelligent Trial and Error (IT&E), Guided Bayesian Search (GBS) and the novel TTE is tested to reach each point of the workspace for 12 different faults illustrated in (Figure 2). Considering these fault scenarios, Figure 3 and Figure 4 show the number of trials needed by each algorithm to reach workspace points represented by degradation of the blue color, while the yellow color represents the workspace points that the robot could not reach. We can notice that GBS was able to produce a performance close to IT&E. However, TTE surpassed both and proved to be able to reach the largest area among the four algorithms within a fewer number of trials. To make the comparison clearer, Figure 5 and Figure 6 shows box-plots illustrating the number of trials needed by each of the four algorithms. To conclude, our proposed algorithm TTE did not only helped the robot to overcome the damages in a fewer trials but also enlarged the reachable workspace despite the damages.

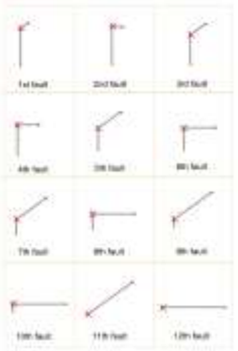


Figure (2) Set of faults to test algorithms. Each fault includes one motor stuck in 45^o or 90^o angles clockwise.

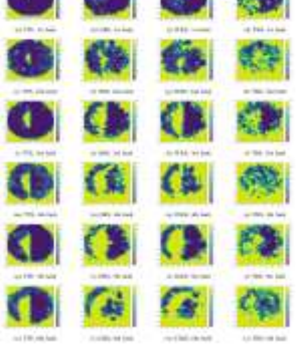


Figure (3) Number of trials for the first six faults (from the left to the right: TTE, GBS, IT&E, TBS)

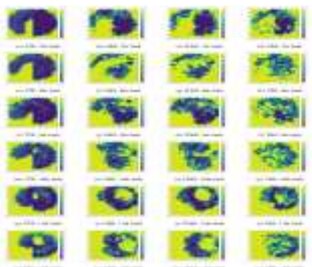


Figure (4) Number of trials for the second six faults (from the left to the right: TTE, GBS, IT&E, TBS)

3.2 Experimental results

We tested our proposed algorithm on a real redundant planar manipulator. The manipulator is composed of six links and six joints actuated by AX-12 servo motors (Figure 7).

The robot is controlled directly by a PC and we used a camera to detect the position of its end-effector. TTE was implemented in Python language to be applied with the robot for different damage scenarios. Experimental results proved the robot's ability to overcome these damages and to reach the desired goals in its workspace in a few number of trials. Figure 8 shows that despite the damages in the fifth and the sixth motors, the robot reached a desired goal in only five trials.

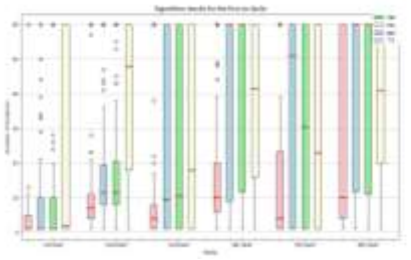


Figure (5) The number of trials needed for each algorithms for the first six faults.

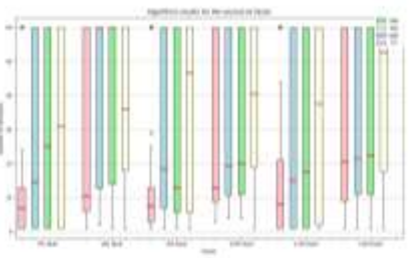


Figure (6) The number of trials needed for each algorithms for the second six faults.

Figure (7) Physical planar manipulator.

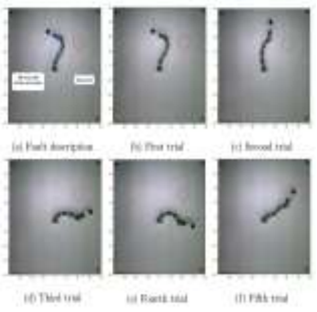


Figure (8) Evaluation on a real manipulator.

4. Conclusion

Like any physical system, robots may face damages or faults. If human interference is not a valid option, the robot will need to overcome these damages and faults in order to perform its tasks. To avoid fault diagnosis, adaptation to failure algorithms were proposed to help the robot to recover through trial and error strategy.

In this paper, we proposed a new adaptation to failure algorithm, we call "TTE" (Tree-search Trial and Error), that allows a redundant manipulator to reach a diverse area of its workspace despite many failure scenarios. TTE is based on a strategy allowing the robot to compensate damage effects by generating new behaviours rather than being restricted to the previously learned ones. However, the robot benefits from its stored knowledge to select a starting point from which it generates new behaviours towards the goal. The simulation results showed that TTE performed better than TBS, IT&E and the proposed GBS as it required a lower number of trials in most damage scenarios. Besides, TTE enlarged the reachable workspace point despite the damages. TTE were also evaluated on a real redundant planar manipulator and its performance surpassed the competing algorithms.

5. References

1. Albadin, A., Albitar, C. & Jafar, A., 2022. Modified IT and E Algorithm for Mobile Robot Self-Adaptation to Failure. *Journal of Global Research in Computer Sciences.*
2. Blanke, M. et al., 2006. *Diagnosis and fault-tolerant control.* s.l.:Springer.

3. Brochu, E., Cora, V. M. & De Freitas, N., 2010. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599.*
4. Carlson, J. & Murphy, R. R., 2005. How UGVs physically fail in the field. *IEEE Transactions on robotics,* Volume 21, pp. 423-437.
5. Chatzilygeroudis, K. et al., 2017. *Black-box data-efficient policy search for robotics.* s.l., IEEE, pp. 51-58.
6. Chatzilygeroudis, K., Vassiliades, V. & Mouret, J.-B., 2018. Reset-free trial-and-error learning for robot damage recovery. *Robotics and Autonomous Systems,* Volume 100, pp. 236-250.
7. Cully, A., Clune, J., Tarapore, D. & Mouret, J.-B., 2015. Robots that can adapt like animals. *Nature 521,* Volume 7553, pp. 503-507.
8. Deisenroth, M. P., Fox, D. & Rasmussen, C. E., 2013. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence,* Volume 37, pp. 408-423.
9. Deisenroth, M. P., Neumann, G. & Peters, J. a. o., 2013. A survey on policy search for robotics. *Foundations and trends in Robotics,* Volume 2, pp. 388-403.
10. Ijspeert, A., Nakanishi, J. & Schaal, S., 2002. Learning attractor landscapes for learning motor primitives. *Advances in neural information processing systems,* Volume 15.
11. Kaushik, R., Anne, T. & Mouret, J.-B., 2020. *Fast Online Adaptation in Robotics through Meta-Learning Embeddings of Simulated Priors.* Las Vegas, NV, USA, IEEE Press, p. 5269–5276.
12. Kaushik, R., Desreumaux, P. & Mouret, J.-B., 2020. Adaptive Prior Selection for Repertoire-based Online Adaptation in Robotics. *Frontiers in Robotics and AI,* Volume 6.
13. Kober, J., Bagnell, J. A. & Peters, J., 2013. Reinforcement learning in robotics: a survey. *The International Journal of Robotics Research,* pp. 1238-1274.
14. Koos, S., Cully, A. & Mouret, J.-B., 2013. Fast damage recovery in robotics with the T-resilience algorithm. *The International Journal of Robotics Research,* Volume 14, pp. 1700-1723.
15. Kume, A. et al., 2017. Map-based multi-policy reinforcement learning: enhancing adaptability of robots by deep reinforcement learning. *arXiv preprint arXiv:1710.06117.*
16. Levine, S. & Koltun, V., 2013. *Guided policy search.* s.l., PMLR, pp. 1-9.
17. Mnih, V. et al., 2015. Human-level control through deep reinforcement learning. *Nature,* Volume 518, pp. 529-533.
18. Mouret, J.-B. & Clune, J., 2015. Illuminating search spaces by mapping elites. *arXiv:1504.04909 [cs.AI].*
19. Nagabandi, A. et al., 2019. Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning. *arXiv: Learning.*
20. Rasmussen, C. E. & Nickisch, H., 2010. Gaussian processes for machine learning (GPML) toolbox. *The Journal of Machine Learning Research,* Volume 11, pp. 3011-3015.
21. Ren, G. et al., 2015. Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation. *Information Sciences,* Volume 294, pp. 666-682.
22. Sanderson, K., 2010. *Mars rover spirit (2003--10).* s.l.:Nature Publishing Group.
23. Smets, P., 1993. Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem. *International Journal of approximate reasoning,* Volume 9.
24. Stulp, F. & Sigaud, O., 2013. Robot skill learning: From reinforcement learning to evolution strategies. *aladyn, Journal of Behavioral Robotics,* Volume 4, pp. 49-61.
25. Sutton, R. S., Barto, A. G. & others, 1998. Introduction to reinforcement learning. *MIT press Cambridge.*
26. Verma, S. et al., 2020. *Deep Reinforcement Learning for Single-Shot Diagnosis and Adaptation in Damaged Robots.* s.l., ACM, pp. 82-89.