International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 01 | Jan - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

Optimising Resource Allocation for Increased Network Reliability using Reinforcement Learning

Anup Kumar Yadav[1], Kalpna Sachan[2], Harimohan Soni[3]

[1]Mechanical Engineering Department, Bansal Institute of Science and Technology, Bhopal (M.P.), India.
[2]Mechanical Engineering Department, Bansal Institute of Science and Technology, Bhopal (M.P.), India.
[3]Mechanical Engineering Department, Bansal Institute of Science and Technology, Bhopal (M.P.), India.

Abstract - Networks deliver essential services to society (e.g., electricity grid, telecommunications, water supply, transportation) yet are susceptible to interruption. Motivated by this, we examine a sequential choice problem whereby an initial network is enhanced over time (e.g., through the addition or augmentation of edge dependability), and incentives are accrued over time based on the network's all-terminal reliability. Actions within each time frame are constrained by the availability of resources, including time, financial capital, and labour. We employed a Deep Reinforcement Learning (DRL) methodology executed in OpenAI-Gym utilising Stable Baselines to address this issue. A Proximal Policy Optimisation (PPO) algorithm was employed to determine the edge requiring enhancement or a new edge to be included, contingent upon the network's present condition and the allocated budget. A reliability polynomial was utilised to compute the all-terminal dependability of the network. To comprehend the model's behaviour throughout diverse settings, we examined many network configurations with varying beginning link reliability, additional link reliability, node quantities, and budget frameworks. We finish with a review of the insights acquired from our series of constructed experiments.

Key Words: Deep reinforcement learning, proximal policy optimisation.

1.INTRODUCTION

We use networks in every part of our lives. Making sure networks work right is important so that everyone can go about their daily lives. For example, people use a network of roads to drive, a network of phones to talk to each other, and a network of power outlets to power items in their homes. These are just a few of the big networks that everyone relies on. An awful lot of people will be hurt if they can't be trusted.

In the past, there have been many infrastructure breakdowns that caused problems for many people. The 2003 blackout in the Northeast is a well-known case. As per History.com [13], this blackout was caused by a problem with the software and impacted fifty million people in the US and Canada. The disaster at Three Mile Island in Pennsylvania and the failure of the levees in Louisiana during Hurricane Katrina are two other examples of infrastructure networks going wrong. There was a problem with one part of the plant that led to the Three Mile Island accident. This problem shut down the whole system. The US Nuclear Regulatory Commission [44] says that this failed part led to a partial nuclear meltdown that affected tens of thousands of people in the area around the plant. Pruitt [34] says that Louisiana's levees were not ready for the amount of water from Hurricane Katrina. As a result, they broke because of the pressure, flooding a lot of New Orleans. These are only a few examples of why the United States' infrastructure often breaks down. People who are touched by infrastructure failures are in a lot of pain because they can sometimes lead to major injuries or death. These examples make it clear how important it is to keep key networks safe.

In its most basic state, a network is just a group of nodes that are linked together by edges. A lot of the things we use every day are linked together in a network. Table 1 shows the different types of networks that Newman [32] talks about.

Network Type	Description	Examples
Technological	The physical infrastructure networks that form the backbone of our society	Internet (physical connections), infrastructure networks (i.e. transportation, telephones, power grid)
Information	Networks of data linked together	World Wide Web (web pages are nodes and links are edges)
Social	A network of people connected through modes such as friendship	Social Media Platforms
Biological	Networks occurring naturally in biology	Food webs and neural networks

Table 1: Types of Networks

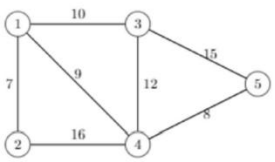

Figure 1: An undirected network

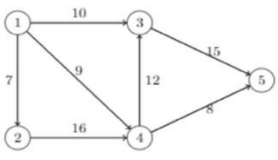

Figure 2: A directed network

A significant challenge in assessing network dependability is the extensive scale of the networks. As network size increases, efficient analysis becomes increasingly challenging.
Current methodologies lack the capacity to assess network reliability efficiently. Prior studies have also investigated the challenges of constructing dependable networks.
This issue is more intricate as it need continual assessment of network dependability.

2. Network Reliability

There are several ways to categorise networks, which helps us determine which ones to investigate further. The many degrees of complexity in network reliability models are covered by Ball [4]. The degree of intricacy is highly dependent on how well the network is linked. A network dependability problem can be either 2-terminal, k-terminal, or all-terminal. An exception to the k-terminal issue exists in the two-terminal and all-terminal scenarios. For the k-terminal reliability issue, there is one root node (s) and k terminal nodes. The likelihood that each of the k-terminal system's nodes are linked to the root node is, in general, the system's dependability. Reliability in a 2-terminal reliability model is defined as the likelihood that the two nodes in the system are linked. In this model, there are only two nodes. A network that is all-terminal simply has all of its nodes linked to all of its other nodes. Reliability is the likelihood of a completely linked system. The computational difficulty of calculating the system's dependability increases as the number of nodes and links in the network increases, with all-terminal reliability problems being the most difficult.

The computational difficulty of calculating the dependability of an all-terminal network increases exponentially with the number of nodes and links connected, as discovered by Provan & Ball [33] to be a #P-complete issue. A plethora of approaches for determining network reliability have been employed to aid in the analysis of this issue. You can get a precise number or a rough idea of the dependability using these techniques. Artificial neural networks are one example of such a technique, computation of boundaries, optimisation, time exponential and polynomial techniques, and enumeration of states; and Monte Carlo simulation. This section delves into the inner workings of these methodologies and examines their prior studies in order to define their strengths and weaknesses.

Ball et al. [5] summarises exact methods for calculating network reliability, including algorithms that run on polynomial time for restricted classes of networks and algorithms that run on exponential time for general networks. They also cover other methods, such as bounds on network reliability and Monte Carlo simulation. In addition to describing and discussing the limits of various network dependability approaches, Gaur et al. [15] provided extensive descriptions of several methods, such as neural networks, state enumeration, and minimal cut enumeration.

A minimal cut set is defined as a collection of system components that, when failing, cause the system to fail (Su et al., [42]). To avoid system failure, minimum cut sets do not include any additional subsets of cuts. To determine the network's reliability, minimal cut enumeration techniques first count the reliability of each minimum cut set and then use these totals to get the network's dependability. One precise way to determine a network's dependability is by cut enumeration. While it works well for smaller networks, its limits become apparent fairly fast. For issues with dependability between two terminals, cut enumeration is the method of choice, as stated by Gaur et al. [15]. There is an exponential increase in the number of cut sets as the network expands. Because of this, it is tedious to calculate all potential combinations for two-, k-, and all-terminal dependability. In order to determine the network's dependability, which is defined as the percentage of states in which the network is operating correctly, Monte Carlo simulation (MCS) methods select a random sample of states to investigate.

To mimic edge failures, Karger [25] used MCS to determine if Due to the randomly chosen edge, the network failed. One of the problems with the MCS method, in his opinion, is how sluggish it is when the likelihood of failure is minimal. For their study on structural dependability, Cardoso et al. [8] used neural networks in combination with Monte Carlo simulation. It can take a long time to compute dependability using MCS since it only permits one network structure to be calculated at a time. In order to address this, they integrated neural networks with MCS, which reduced calculation time and yielded more accurate dependability assessments.

Similar to how the human brain uses neural networks, artificial neural networks (ANNs) mimic these structures. In order to learn from experiences, the parts of ANNs collaborate in series and parallel, much like the brain. In order for this learning to take place, a training set containing inputs and known, desired outputs is utilised. The primary applications of artificial neural networks (ANNs) are control, pattern recognition, optimisation, associative memory, and prediction (Jain & Mao, 2021). In order to assess the dependability of the network, Srivaree-ratana et al. [41] employed an artificial neural network. Their research involved training the ANN using a variety of topologies and connection reliabilities. The best network topology was then determined by using the ANN to predict the network's reliability in relation to the topology and the link reliabilities. In the end, they determined the precise dependability for each topology by using it. Through comparisons to an accurate technique and an upper bound determined from a polynomial time algorithm, they show that their estimation works well empirically while being computationally costly.

Setting limits on the network's dependability provides an alternate method to both exact and approximate reliability estimations. The limit-finding approach has the benefit of being less computationally costly than the other methods; nevertheless, it lacks accuracy as it simply gives bounds and not a specific dependability. An algorithm was developed to identify by Sebastio et al. [37]. limits on the dependability of a two-terminal network. The algorithm allows the user to choose the time it should take to execute. Their method takes mincuts and minpaths into account. A minimum route is a set of interconnected nodes in a network that would remain linked even if a single connection were to be severed.

In order to decrease the dependability between the upper and lower bounds, their method searches the network for the most critical minimal paths and cuts. Bounds have another use, which Satitsatian and Kapur [36] addressed. In order to calculate the precise reliability and reliability boundaries, they discovered a lower limit for the dependability of the network. In order to get the lower reliability, bound with minimal computing work, they developed a method to identify a subset of lower boundary points. When it comes to all-terminal network reliability allocation issues (RAP), Ramirez-Marquez & Rocco [35] introduced a novel approach. Their objective in solving this challenge was to find the optimal network cost while keeping dependability in mind.

The three-stage process they developed consists of the following: first, producing network configurations; second, determining each network's dependability using MCS; third, penalising networks that fail to satisfy the reliability

requirement; and last, ranking the networks from best to worst. Their algorithm uncovered solutions were, at most, 21% less expensive and, at worst, 7% cheaper than those identified in the literature before. Another group that suggested a way to use Monte Carlo Simulation for resource allocation was Yeh et al. [46]. Movable cluster swarm optimisation (MCS-PSO) was their suggested strategy. Meeting the dependability limits while keeping component costs low was their goal. Their approach outperformed MCS on its own when it came to efficiency and reliability approximation. Other people have offered other ways to put together trustworthy networks.

For generic systems, Mettas [30] investigated the component-level reliability allocation problem. Based on his research, we may infer the minimal system dependability requirements for various components reliability. Methods to examine network topology were employed by both Jan et al. [22] and AboElFotoh & Al-Sumait [1]. Finding the best topological architecture of links that minimised cost while meeting the minimal need for network stability was their goal. A branch-and-bound decomposition approach was used by Jan et al. [22]. Their approach breaks the network down into smaller issues that their branch and bound algorithm can handle, according to the number of links. Using an ANN, AboElFotoh & Al-Sumait [1] were able to resolve the identical issue.

2.1 Training with Positive Reward

The many methods of machine learning are categorised according to the algorithm's learning process. Reinforcement learning is one such method. In order to train a computer to respond appropriately to new situations, reinforcement learning relies on a training data set that contains both positive and negative reinforcement, as described by Zhang [47]. As a result of this input, the machine is able to do the job better next time. The value of an action is determined by the reinforcement learning algorithm, which randomly picks an action. An immediate reward and the value of reaching a new state both contribute to the value of the acts. The goal of reinforcement learning is to maximise total reward by learning the value of optimum state/action combinations through repeated application of this procedure.

A wide variety of fields can benefit from reinforcement learning (RL). The gaming industry was an early adopter of RL. The player often takes control of a character and decides what to do in a video game. The choices they make are seen by RL as the actions. Researchers were able to train the system to choose the optimal action in every given condition by using RL to test out a wide variety of scenarios. Flappy Bird and Breakout were the two video games that Lin et al. [27] tackled. Together, a neural network and reinforcement Q-learning were used to train both games. neural network and one that does not. Using a neural network significantly reduced the time it took to train the model compared to not using one.

The use of networks is not the only one where reinforcement learning is useful. In order to investigate methods for distributing computer network resources, Yang et al. [45] developed a deep reinforcement learning model. Ensuring the system's dependability from start to finish was their primary objective. In order to keep the system's channels from falling short of their quality criteria, they used a Q-learning algorithm to assist the system in allocating resources. After around 100 attempts of training, the Q-learning method was determined to be successful in their study. Research on RL and other forms of AI in healthcare systems was conducted by Gottesman et al. [17]. The health of patients depends on decisions made about when to do specific jobs at a healthcare facility. Following training, RL can assist healthcare providers in determining the best course of therapy for a patient based on their baseline condition by studying the outcomes of previous decisions. The application of RL in healthcare has allowed for the optimisation of patients' treatment sequences.

2.2. Progress in Reliability

The goal of reliability growth is to increase a system's dependability all through its lifecycle, from design to development to operation. The fundamental idea is to put a system through its paces, find its weak spots, and then fix its design such that those weak spots are less likely to recur, making the system more reliable overall. One way to make design modifications that will have a positive impact on a system's dependability is to employ reliability growth modelling. Among the first to examine the development of dependability was Duane [12]. When comparing mechanical and electromechanical systems, he discovered that both improved dependability at about the same pace during development. His research centred on determining how long it takes for systems to learn to reliably anticipate future events. The link between the logarithm of cumulative failure rate and cumulative operating hours was nearly linear, according to his findings. Reliability in relation to system age was further investigated by Crow [10].

The AMSSAA model, which stands for Army Material Systems Analysis Activity, was one of his suggestions. He suggested a nonhomogeneous Poisson process model with a Weibull intensity function to investigate age-dependent reliability, as stated in "The AMSAA Reliability Growth Guide," which "summarised the benefits of reliability growth management in finding unforeseen deficiencies, designing improvements, reducing risk, and increasing the probability of meeting objectives" (Kurtz et al., [26]).

There are three applications of reliability growth models for a generic system, as stated by Cahoon et al. [7]. Planned improvements to the system's reliability, monitored progress towards those goals, and maintaining project momentum are all part of this. The testing of systems by the Department of Defence (DoD) is another real-world use of reliability growth modelling. There are two varieties of reliability growth models utilised by the Department of Defence. One kind is system-level, which employs nonhomogeneous Poisson processes (NHPPs). The other is competitive risk, which examines several failure modes in series. The number of failures and the time between failures may be monitored with the use of NHPP models. All of the competing risk models look at the system from the perspective of its individual parts working in tandem. So, for the system to function, every part has to be operational.

3. General Model Formulation

Our challenge involves an initial network including n nodes and a specified set of n(n-1) edges. We examine a sequential

decision problem including m time periods, where in each period t=1,2,…,m, we can execute restricted investments to include possible edges from the collection E={{i,j}:i=1,2,…,n−1;j=i+1,2,…,n} into the network and/or enhance the dependability of current edges. An edge $\{i, j\} \in E$ that has been included into the network and enhanced $z$ij times is presumed to possess dependability ki+$z$ij$l$ij, where kij and $l$ij are parameters.

Our objective is to optimise the total discounted reward accrued throughout the time intervals t=0, 1…, m−1, where the reward at time interval t is contingent upon the network's all-terminal dependability immediately after that interval. A predetermined budget $BBtt$ is allocated at the commencement of each time period t=0, 1…, m−1 and may be utilised for immediate activities or deferred for future usage. Parameters $cciiii$ and $ppiiii$ delineate the expense associated with adding an edge $\{i, j\} \in E$ and enhancing an existing edge $\{i,j\} \in E$, respectively.

The network state s before to any time period is characterised by the tuple

$$s=(t,R,\beta),$$

where $t \in \{0, 1, m-1\}$ denotes the number of completed time periods, R is a $|E|$-vector indicating the reliability of each edge in the network, and β represents the remaining budget. Denote the components of R as $r$ij, $\{i, j\} \in E$, where $r$ij=0 if the edge $\{i, j\}$ has not been included into the network.

In state s= (t, R, β), an action is described as a= (X, Y) where X and Y are $|E|$-dimensional vectors. The vector X comprises of elements xij, where $\{i, j\} \in E$, with xij = 1 if the edge $\{i, j\}$ is included into the network; 0 otherwise. The vector YY comprises elements $y$ij, $\{i,j\} \in E$, where $y$ij=1 indicates that the edge $\{i,j\}$ has been enhanced; 0 otherwise. The operation with $x$ij=$y$ij=0 for every $\{i, j\} \in E$ signifies the decision to progress to the subsequent time period without augmenting or enhancing any supplementary edges. The viable actions in state s= (t, R, β) are delineated by the equations:

$$\sum_{\{i,j\} \in E}(x_{ij} + y_{ij}) \leq 1 \quad (1)$$

$$\sum_{\{i,j\} \in E}(c_{ij}x_{ij} + p_{ij}y_{ij}) \leq \beta \quad (2)$$

$$x_{ij} = 0, \forall \{i,j\} \in E: r_{ij} > 0 \quad (3)$$

$$y_{ij} = 0, \forall \{i,j\} \in E: r_{ij} = 0 \quad (4)$$

$$r_{ij} + k_{ij}x_{ij} + l_{ij}y_{ij} \leq 1, \ \forall \{i,j\} \in E \quad (5)$$

Equation (1) says that we can only do one or zero actions during a given time period. Equation (2) says that the actions we do must be less than or equal to the amount we have left for the period. Keep in mind that any spending that wasn't used in time period bb can be carried over and used in later time periods. For this reason, it might be best to move on to the next period even if there are enough resources to do one of the other tasks. For action $x$ij=1 to be possible, equation (3) says that $r$ij=0. This means that an edge $\{i, j\} \in E$ can't be added if it's already in the network. For action $y$ij=1 to be possible, equation (4) says that $r$ij > 0. This means that an edge $\{i, j\}$ in the network can only be made better if it was already there.

One of the three actions that are possible is to improve an edge (i.e., $y$ij=1 for some 19 $\{i, j\} \in E$), add an edge (i.e., $x$ij=1 for some $\{i, j\} \in E$), or choose to move on to the next time (i.e., $x$ij=0 $\forall$ $\{ii, jf\}$. If you do something with $x$ij=1 or $y$ij=1 for some $\{i, j\} \in E$, it doesn't mean we'll be in a new time period; it just means that the state variables R and β have changed. The fifth equation makes sure that an edge can't be given a reliability number higher than 1.

The state transition function is now defined as (t', R′, β') =g (s, a) for an action a= (X, Y) done in state s=(t,R,β). The new state is set by if $x$ij=$y$ij=0 $\forall$ $\{i, j\} \in E$

$$t' = t + 1, \quad (6)$$

$$R' = R, \quad (7)$$

$$\beta' = \beta + B_{t+1}. \quad (8)$$

Equation (6) indicates that the time period, t', following a state transition is one period subsequent to the preceding time period, t. Equation (7) asserts that the network's dependability remains constant throughout a state shift. Equation (8) indicates that the budget in the new state post-transition is determined by the residual budget from the prior state in conjunction with the fixed budget allocated for the new period.

4. First Model Experiments and Analysis

The RL problem from [6] was used to train our model. It was written in Python using the OpenAI stable baselines and followed a standard approach. There are a number of reliable versions of reinforcement learning methods in the OpenAI package. There is a pre-trained RL agent in each application. This agent learns from what it sees, does, and is rewarded.

We used Python version 3.8.15 for our study. We used stable OpenMPI baselines to make sure that all methods could work. For stable baselines, we had to say how many training episodes our models should have. We used 5000 episodes for all of them.

A Maskable Proximal Policy Optimisation (M-PPO) method [18] was used for our tests. Based on the problem constraints, this method narrows down the action space to only actions that are possible. This means that for our problem, the only things that can be done are to add edges that aren't already in the network, make edges that are already in the network better, and make sure that all of these actions don't go over budget. A mask, which is a vector that keeps track of acceptable acts, is also used by the algorithm. This will restrict the activities that can be done with the XX and YY vectors.

We used an iterative method to build and study our network. To start, we made a simple model that only let one choice be made at a time. At first, this model only let you do one thing, which was to improve the reliability of current network edges by sending resources to them. This first model was made to make sure that the model and RL code were working right.

Then, we made the model more complicated so that we could make more than one choice in each time and take more than one type of action (like adding new lines or improving existing ones).

The first model was mostly about making one of the network's edges better. This model looks at the benefits that are randomly given to each edge at the start and figures out the next best change to make. These benefits are shown as higher dependability that comes from upgrading to the chosen edge. With this update, the benefits for this edge will be worth less in the future.

5. Results and discussion:

We also analyzed the different reward ratios to see if they had an effect on the decisions the model made. During our analysis, we found that when the ratio was more favorable for immediate rewards, the model chose to improve more links, but when the ratio was more favorable for long-term rewards, the model chose to add more links. Figure 6 shows the trend in regard to the rewards ratio. The model also found many times when it could find better answers for networks with smaller budgets or shorter time frames. These results might have been because of how the model was trained. It's possible that the model could have done a better job with the bigger networks if it had more training sessions. In the five-node network with initial and new link reliabilities of 0.9/0.9, the problem instance with five periods, a 1:2 rewards ratio, and a 3000 per period budget had a final all-terminal reliability of 1.0. On the other hand, the problem instance with seven periods, a 1:2 rewards ratio, and a 3000 per period budget had a final all-terminal reliability of 0.9995.

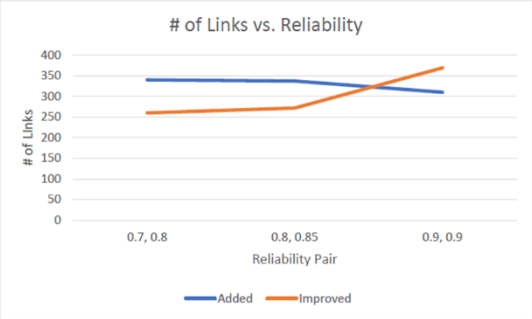

Figure 5: # of Links vs. Reliability Graph

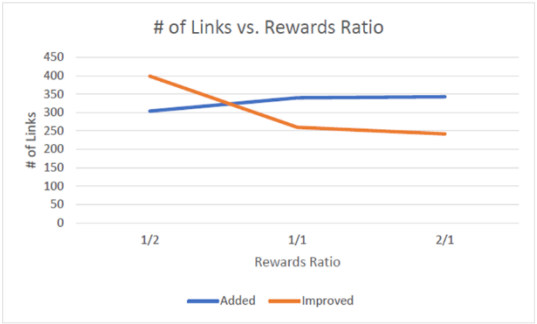

Figure 6: # of Links vs. Rewards Ratio Graph

The optimal 7-node network has an all-terminal reliability of 0.9999. The problem instance that led to this dependability had initial and new link reliability of 0.9/0.9, seven periods, a 1:2 rewards ratio, and a per period budget of 7000. The model opted to incorporate nine linkages and enhance six.

6. Conclusion

Our research concentrated on constructing 5-, 7-, and 10-node networks utilising same prices for connection enhancement and addition. To further investigate the model's conclusions about the addition or enhancement of linkages, various combinations of addition and improvement costs should be investigated. Our research concentrated solely on three network sizes, which became increasingly computationally intensive as the networks expanded; hence, further investigation is necessary to develop a model capable of making judgements more rapidly. While other sources of complexity may have influenced extended run-times, one potential enhancement is to substitute the existing dependability-polynomial method for assessing network reliability with a more scalable option for bigger instances.
We also made a lot of assumptions about the original link reliability, the new link reliability, the amount to improve, the cost to add and improve, the benefits, and so on. The results of our experiments were restricted by these assumptions, so more models with less rigid assumptions should be made in the future.

REFERENCES

1. AboElFotoh, H. M., & Al-Sumait, L. S. (2001, December). A Neural Approach to Topological Optimization of Communication Networks, with Reliability Constraints. IEEE Transactions on Reliability, 50(4), 397-408.
2. Acevedo, P. E., Jackson, D. S., & Kotlowitz, R. W. (2006). Reliability Growth and Forecasting for Critical Hardware through Accelerated Life Testing. Bell Labs Technical Journal, 11(3), 121-135.
3. Ahmad, N., Bokhari, M., Quadri, S., & Khan, M. (2008). The exponentiated Weibull software reliability growth model with various testing-efforts and optimal release policy: A performance analysis. The International Journal of Quality & Reliability Management, 25(2), 211-235.
4. Ball, M. O. (1986, August). Computational Complexity of Network Reliability Analysis: An Overview. IEEE Transactions on Reliability, 35(3), 230-239.
5. Ball, M. O., Colbourn, C. J., & Provan, J. S. (1995). Network Reliability. In Handbooks in Operations Research and Management Science (Vol. 7, pp. 673-762).
6. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540.
7. Cahoon, J., Sanborn, K., & Wilson, A. (2021, November). Practical reliability growth modeling. Quality and Reliability Engineering International Special Issue: Advancing Statistical Methods for Testing and Evaluating Defense Systems, 37(7), 3108-3124.
8. Cardoso, J. B., de Almeida, J. R., Dias, J. M., & Coelho, P. G. (2008, June). Structural reliability analysis using Monte Carlo simulation and neural networks. Advances in Engineering Software, 39(6), 505-513.
9. Cinque, M., Cotroneo, D., Pecchia, A., Pietrantuono, R., & Russo, S. (2017). Debugging-workflow-aware software reliability growth analysis. Software: Testing Verification, and Reliability, 27(7).
10. Crow, L. H. (1975). Reliability Analysis for Complex, Repairable Systems. Aberdeen Proving Ground, Maryland: Army Material Systems Analysis Activity.

11. Dempsey, P. J., & Sheng, S. (2013, May). Investigation of data fusion applied to health monitoring of wind turbine drivetrain components. Wind Energy, 16(4), 479-489.
12. Duane, J. T. (1964, April). Learning Curve Approach to Reliability Monitoring. IEEE Transactions on Aerospace, 2(2), 563-566.
13. Editors, H. (2009, November 24). Blackout hits Northeast United States. Retrieved from History: https://www.history.com/this-day-in-history/blackout-hits-northeast-united-states
14. Fries, A. (1993, June). Discrete Reliability-Growth Models Based on a Learning-Curve Property. IEEE Transactions on Reliability, 42(2), 303-306.
15. Gaur, V., Yadav, O. P., Soni, G., & Rathore, A. P. (2021). A literature review on network reliability analysis and its engineering applications. Journal of Risk and Reliability, 235(2), 167-181.
16. Goldbeck, N., Angeloudis, P., & Ochieng, W. Y. (2019, August). Resilience assessment for interdependent urban infrastructure systems using dynamic network flow models. Reliability Engineering & System Safety, 188, 62-79.
17. Gottesman, O., Johansson, F., Komorowski, M., Faisal, A., & Sontag, D. (2019, Jan). Guidelines for reinforcement learning in healthcare. Nature Medicine, 25(1), 16-18.
18. Huang, S., & Ontañón, S. (2020). A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. arXiv preprint arXiv:2006.14171.
19. Hui, K.-P. (2005). Network Reliability Estimation. Doctoral Dissertation.
20. Hussin, M., Hamid, N. A., & Kasmiran, K. A. (2015, January). Improving reliability in resource management through adaptive reinforcement learning for distributed systems. Journal of Parallel and Distributed Computing, 75, 93-100.
21. Jain, A. K., & Mao, J. (1996). Artificial Neural Networks: A Tutorial. Computer, 31-44.
22. Jan, R.-H., Hwang, F.-J., & Cheng, S.-T. (1993, March). Topological Optimization of a Communication Network Subject to a. IEEE Transactions on Reliability, 42(1), 63-70.
23. Kanoun, K., Kaaniche, M., Beounes, C., Laprie, J.-C., & Arlat, J. (1993, June). Reliability Growth of Fault-Tolerant Software. IEEE Transactions on Reliability, 42(2), 205-219.
24. Kapur, P., & Garg, R. (1992, July). A Software-Reliability Growth-Model for an Error-Removal Phenomenon. Software Engineering Jornal, 7(4), 291-294.
25. Karger, D. R. (1999). A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM Journal on Computing, 29(2), 492-514.
26. Kurtz, J., Sprik, S., & Bradley, T. H. (2019). Review of transportation hydrogen infrastructure performance and reliability. International Journal of Hydrogen Energy, 12010-12023.
27. Lin, C.-J., Jhang, J.-Y., Lin, H.-Y., Lee, C.-L., & Young, K.-Y. (2019). Using a Reinforcement Q-Learning-Based Deep Neural Network for Playing Video Games. Electronics.
28. Lloyd, D. K. (1986). Forecasting Reliability Growth. Quality and Reliability Engineering International, 2, 19-23.
29. Mahmoodzadeh, Z., Wu, K.-Y., Droguett, E. L., & Mosleh, A. (2020). Condition-Based Maintenance with Reinforcement Learning for Dry Gas Pipeline Subject to Internal Corrosion. Sensors, 20(19).
30. Mettas, A. (2000). Reliability Allocation and Optimization for Complex Systems. Annual Reliability and Maintainability Symposium. 2000 Proceedings. International Symposium on Product Quality and Integrity, 216-221.
31. Milanovic, J. V., & Zhu, W. (2018, September). Modeling of Interconnected Critical Infrastructure Systems Using Complex Network Theory. IEEE Transactions on Smart Grid, 9(5), 4637-4648.
32. Newman, M. (2010). Networks: An introduction. Oxford, United Kingdom: Oxford University Press.
33. Provan, J. S., & Ball, M. O. (1983, November). The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected. SIAM Journal on Computing, 12(4), 777-788.
34. Pruitt, S. (2020, August 27). How Levee Failures Made Hurricane Katrina a Bigger Disaster. Retrieved from History: https://www.history.com/news/hurricane-katrina-levee-failures
35. Ramirez-Marquez, J. E., & Rocco, C. M. (2008, November). All-terminal network reliability optimization via probabilistic solution discovery. Reliability Engineering & System Safety, 93(11), 1689-1697.
36. Satitsatian, S., & Kapur, K. C. (2006, June). An Algorithm for Lower Reliability Bounds of Multistate Two-Terminal Networks. IEEE Transactions on Reliability, 55(2), 199-206.
37. Sebastio, S., Trivedi, K. S., Wang, D., & Yin, X. (2014, November 1). Fast computation of bounds for two-terminal network reliability. European Journal of Operational Research, 238(3), 810-823.
38. Sen, A., & Fries, A. (1998). Model Misspecification Effects Within A Family of Alternative Discrete Reliability-Growth Models. Lifetime Data Analysis, 65-81.
39. Serrano, W. (2019). Deep Reinforcement Learning Algorithms in Intelligent Infrastructure. Infrastructures, 4(3).
40. Shao, X., & Fang, X. (2016). Estimation in discrete reliability growth, a growth model with coefficient condition. Journal of Systems Science and Complexity, 1112-1122.
41. Srivaree-ratana, C., Konak, A., & Smith, A. E. (2002, June). Estimation of all-terminal network reliability using an artificial neural network. Computers & Operations Research, 29(7), 849-868.
42. Su, Y.-C., Mays, L. W., Duan, N., & Lansey, K. E. (1987, December). Reliability-Based Optimization Model for Water Distribution Systems. Journal of Hydraulic Engineering, 113(12), 1539-1556.
43. Talafuse, T. P., & Pohl, E. A. (2017). Small sample reliability growth modeling using a grey systems model. Quality Engineering, 29(3), 455-467.
44. U.S. NRC. (2018, June 21). Backgrounder on the Three Mile Island Accident. Retrieved from United States Nuclear Regulatory Commission: https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/3mile-isle.html
45. Yang, T., Hu, Y., Gursoy, M. C., Schmeink, A., & Mathar, R. (2018). Deep Reinforcement Learning based Resource Allocation in Low Latency Edge Computing Networks. 2018 15th International Symposium on Wireless Communication Systems (ISWCS), 1-5.
46. Yeh, W.-C., Lin, Y.-C., Chung, Y. Y., & Chih, M. (2010, March). A Particle Swarm Optimization Approach Based on Monte Carlo Simulation for Solving the Complex Network Reliability Problem. IEEE Transactions on Reliability, 59(1), 212-221.
47. Zhang, X.-D. (2020). Machine Learning. In A Matrix Algebra Approach to Artificial Intelligence (pp. 223-225). Singapore: Springer.