sensors

MDPI

Article

An Improved DDPG and Its Application in Spacecraft Fault Knowledge Graph

Xiaoyu Xing [1,2,*], Shuyi Wang [1,2] and Wenjing Liu [1,2]

[1] Beijing Institute of Control Engineering, Beijing 100190, China
[2] Science and Technology on Space Intelligent Control Laboratory, Beijing 100190, China
* Correspondence: xingxiaoyu0904@163.com; Tel.: +86-178-6311-6833

Abstract: We construct a spacecraft performance-fault relationship graph of the control system, which can help space robots locate and repair spacecraft faults quickly. In order to improve the performance-fault relationship graph, we improve the Deep Deterministic Policy Gradient (DDPG) algorithm, and propose a relationship prediction method that combines representation learning reasoning with deep reinforcement learning reasoning. We take the spacecraft performance-fault relationship graph as the agent learning environment and adopt reinforcement learning to realize the optimal interaction between the agent and the environment. Meanwhile, our model uses a deep neural network to construct a complex value function and strategy function, which makes the agent have excellent perceptual decision-making ability and accurate value judgment ability. We evaluate our model on a performance-fault relationship graph of the control system. The experimental results show that our model has high prediction speed and accuracy, which can completely infer the optimal relationship path between entities to complete the spacecraft performance-fault relationship graph.

Keywords: knowledge graph; relational reasoning; representation learning; deep reinforcement learning

check for updates

Citation: Xing, X.; Wang, S.; Liu, W. An Improved DDPG and Its Application in Spacecraft Fault Knowledge Graph. *Sensors* **2023**, *23*, 1223. https://doi.org/10.3390/s23031223

Academic Editors: Anastasios Doulamis and Paolo Gastaldo

Received: 25 November 2022
Revised: 28 December 2022
Accepted: 18 January 2023
Published: 20 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, space robots used for spacecraft fault maintenance have been widely considered. They need accurate and fast fault location ability, which is one of the difficulties in application. At present, the mapping relationship between the performance and fault of a spacecraft control system is generally expressed in the form of failure mode and effects analysis (FMEA) or a fault tree. When the spacecraft control system is abnormal, the ground experts can locate the fault source by manual inquiry, so it is difficult to make real-time diagnosis, accurately locate complex faults, and visualize fault diagnosis. Knowledge graph [1–9], as an effective tool to describe massive knowledge, knowledge attributes, and knowledge relationships, provides a new means for fault diagnosis. We can manually or automatically construct the knowledge graph of the corresponding relationship between the performance and fault of the spacecraft control system by combining various model knowledge, expert knowledge, and data, which we call the spacecraft performance-fault relationship graph. However, there is no systematic and comprehensive spacecraft fault knowledge base in the aerospace field, so it is difficult to construct a large-scale spacecraft performance-fault relationship graph for spacecraft fault diagnosis. Taking the spacecraft control system as an example, its working environment is harsh. Because of its closed-loop characteristics, there are few fault samples accumulated during long-term in-orbit operation, and the fault mechanism cannot be traversed. Meanwhile, its structure is complex, and the components are closely related. Therefore, it is difficult to establish an accurate and complete performance-fault relationship graph of a spacecraft control system, which affects the accuracy of fault diagnosis results. It is necessary to use relational reasoning to predict the relationship and complete the relationship.

The commonly used relationship reasoning methods at present include representation learning reasoning, path reasoning, reinforcement learning reasoning, and graph convolution network reasoning. However, the spacecraft performance-fault relationship graph has complex relationships, various entities, and rich physical meanings. These methods are difficult to directly apply to a spacecraft performance-fault relationship graph.

The TransE [10] model is a classic model for learning reasoning. It maps entities and relationships into a continuous vector space, and calculates the entity vectors to predict the relationships. However, this method is only suitable for dealing with the one-to-one correspondence of entities and is not suitable for spacecraft performance-fault relationship graphs with complex and diverse relationships. A typical path reasoning model is the Path Ranking Algorithm (PRA) [11], which extracts path feature values from the knowledge graph, and makes relationship judgment according to path features. COR-PRA (Constant and Reversed Path Ranking Algorithm) [12] adds a constant path search mechanism on the basis of PRA, which makes the relationship reasoning more comprehensive. However, path reasoning pays too much attention to the graph structure information, ignoring the semantic attributes of entities and relationships, and the reasoning accuracy is not high. Reference [13] first applied reinforcement learning to relational reasoning and put forward a reinforcement learning framework DeepPath for learning multi-hop relational paths. Based on this, reference [14] added long short-term networks and a graphic attention mechanism as memory components. Reference [15] presented agent MINERVA, which has the built-in flexibility to take paths of variable length and learns to perform query answering by walking on a knowledge graph conditioned on an input query. References [16–19] improved reward functions and strategies for different datasets based on reinforcement learning reasoning. However, reinforcement learning reasoning requires setting a high reward function and value function. For the spacecraft performance-fault relationship graph with various states and complex physical meanings, it is difficult to set an accurate reward function and value function to achieve a good path finding effect. The graph convolution network [19–29] directly aggregates the adjacent nodes of the graph, which can better reflect the structural relationship of graph data and is more suitable for processing directed graph and large graph data. However, its model is complex and lacks interpretability, so it is not suitable for the complicated spacecraft performance-fault relationship graph.

To solve the above problems, we improve the DDPG algorithm [30] by combining representation learning reasoning with deep reinforcement learning reasoning. We study a Semantic relation and Position Deep Deterministic Policy Gradient reasoning model (SPDDPG) and apply it to the spacecraft performance-fault relationship graph to realize relational reasoning. First, this model extracts the semantic and location features of the embedded vectors of entities and relations. The high-dimensional semantic embedded vectors of entities and relations are obtained by using the TransE model. However, its excessively high dimension and scattered features will affect the reasoning accuracy. To solve this problem, Principal Component Analysis (PCA) [31] is adopted to reduce its dimension to obtain the key semantic feature vectors. Meanwhile, according to the position of the entity in the spacecraft performance-fault relationship graph, the position vector is obtained by Boolean vector conversion. Then, we splice the semantic vector and the position vector as the embedded vector of the entity and input it into DDPG for processing. DDPG takes the spacecraft performance-fault relationship graph as the environment, in which we set different corresponding states and rewards for complex entity types in the spacecraft performance-fault relationship graph. The strategy network of DDPG outputs the optimal action vector according to the embedded vector of the current state, and the value network calculates the action value according to the state vector and the action vector, updating the network parameters in reverse. Both the strategy network and value network are deep neural network models, which can fit complex strategy and value functions. Thus, they are suitable for the spacecraft performance-fault relationship graph with intricate relationships and complex physical meanings.

In short, this model can achieve accurate reasoning of the spacecraft performance-fault relationship graph due to two innovations: 1. It extracts the semantic information and position information from the spacecraft performance-fault relationship graph. 2. It has a reasonable network and state setting to realize fast optimal action prediction.

2. Materials and Methods

2.1. Fault Diagnosis Process Based on Spacecraft Performance-Fault Relationship Graph

The fault diagnosis process based on the spacecraft performance-fault relationship graph is divided into two parts. First, data processing and feature extraction are carried out on spacecraft on-orbit data, and the obtained results are matched with fault symptom entities in the performance-fault relationship graph. The fault symptom entity corresponding to the current on-orbit data is then determined. Second, in the performance-fault relationship graph, reasoning is started from the failure symptom, and the failure cause entity or failure mode entity corresponding to the current failure symptom is obtained through an entity-relationship-entity triple search. There may be many fault causes or modes, so the final fault diagnosis result should be determined by information fusion of fault occurrence probability and expert experience. As a tool of fault diagnosis, the spacecraft performance-fault relationship graph must be complete and accurate. We propose a relational reasoning model SPDDPG for the performance-fault relationship graph and use it to improve the performance-fault relationship graph, which provides reliable and effective support for subsequent fault diagnosis.

2.2. SPDDPG Model and Process

The steps of relationship reasoning with the SPDDPG model are as follows: 1. Build the SPDDPG framework based on the spacecraft performance-fault relationship graph and transform the components in the relationship graph into the basic elements of the SPDDPG framework. 2. Take the entity in the performance-fault relationship graph as the current state and select the optimal action by using the strategy network, which we call the Actor Network, to obtain the entity corresponding to the next state. 3. Use the value network, which we call the Critic Network, to fit the action value function, and update the parameters of the strategy network and the value network. The reasoning process of this model is shown in Figure 1.

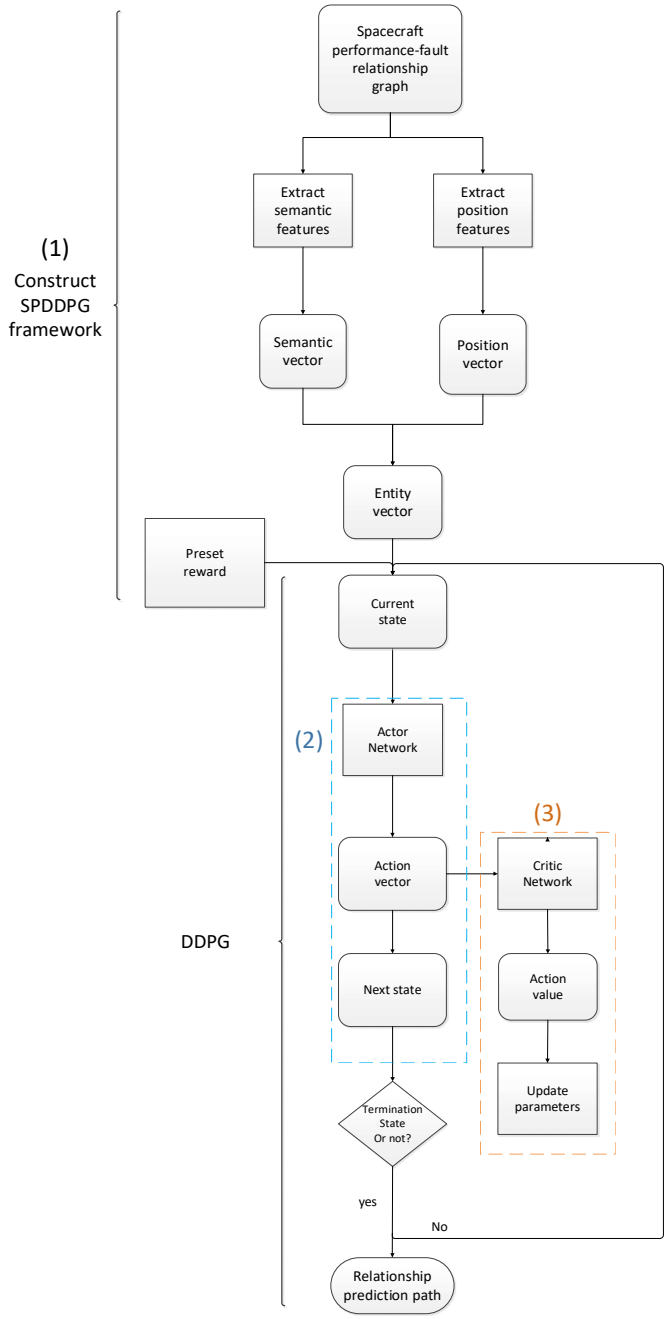

Figure 1. The reasoning process of SPDDPG.

2.3. Construct SPDDPG Framework

We use the deep reinforcement learning algorithm for relational reasoning, so we need to construct the reinforcement learning framework first. We configure the basic elements of the framework, including environment, state, action, and reward, according to the performance-fault relationship graph of the spacecraft control system.

Environment. We use the performance-fault relationship graph of the spacecraft control system as the environment to interact with the agent. Because the performance-fault relationship graph is difficult to directly calculate and process by agents, we transform it into an $n \times n$ (where n is the number of entities) dimensional environment matrix $\boldsymbol{E}$. The u relationships between entities are arranged in sequence. If there is a relationship $q \in (0,u)$ between entity i and entity j, then E_{ij} is set to q; otherwise, it is set to 0. Finally, we obtain environment matrix $\boldsymbol{E}$, which is convenient for a computer to calculate and process the content of the performance-fault relationship graph.

State. The relationship path of performance-fault relationship graph is complex and involves many states, so it is difficult to achieve ideal experimental results only by setting the normal state and the termination state. We divide the states into the normal state, termination state, and imminent termination state. The imminent termination state refers to the first-order adjacent state to the termination state, and when the agent reaches the termination state, it will complete the path finding of relational reasoning. Our innovative setting of the imminent termination state can help the agent to find the path in the direction of the termination state quickly. The entities in the performance-fault relationship graph are taken as the state. However, they need to be converted into a vector form convenient for computer identification and calculation, which is called the state vector in this paper. The state vector includes two parts: semantic features and location features.

The first part is semantic feature extraction. We use the TransE model to process triples in the performance-fault relationship graph, extracting the semantic high-dimensional vectors of entities and relationships. Then, we use PCA to reduce the dimension and extract the principal components of the semantic features of entities and relationships, which we call the semantic vector.

The second part is location feature extraction. We arrange all entities in the order of (e_1, e_2, ... e_n), and adopt one-hot coding to obtain the position vectors of all entities. Specifically, we define an n-dimensional zero vector as the position vector of every entity. If entity i has a relationship with the entity e_j, the jth element of the vector is set to 1. The position vector contains the global position characteristics of the entity in the performance-fault relationship graph.

Action. In the case of entities as states, the relationships between entities act as the action to connect states. The agent outputs the predicted action vector according to the current state to choose the action to reach the next state. The action vector is represented by the relational semantic vector obtained from the TransE model.

Reward. As the basis for judging the current state, a reward should be set artificially according to the path distance and path type between the current state and the termination state. We use the critic network to fit the complex action value function, so the setting of the state reward can follow the simple and effective principle. The normal state occupies most of the path of relational reasoning and should not be set too tendentiously, so it is set to 0. The imminent termination state should guide the agent to the terminal state, so it is set to 1, and the termination state reward is set to 2. The formula is as follows:

$$R_t = \begin{cases} 0,\ S_t \notin N_1(END)\ and\ S_t \neq END \\ 1,\ S_t\ \epsilon\ N_1(END) \\ 2,\ S_t = END \end{cases}$$

where t is the number of steps of agent path finding, S_t is the current state, R_t is the reward corresponding to the current state, END is the end state, and $N_1(END)$ is the first-order neighborhood of the end state. The unique reward of the terminal state can help the agent approach the terminal state quickly from many complicated paths, and enhance the training effect.

2.4. DDPG Model

The DDPG model consists of two kinds of neural networks, the actor network and critic network. The actor network is responsible for predicting actions according to the state, and the critic network is responsible for predicting action values according to the state and actions. The two types of networks have their own current network and target network. The current network calculates the estimated value according to the current state, while the target network calculates the target value according to the next state. The parameters of the neural network are updated according to the estimated value and the target value. Next, we elaborate action selection based on the actor network and parameter update based on the critic network in detail. Figure 2 shows the structure of the DDPG model.

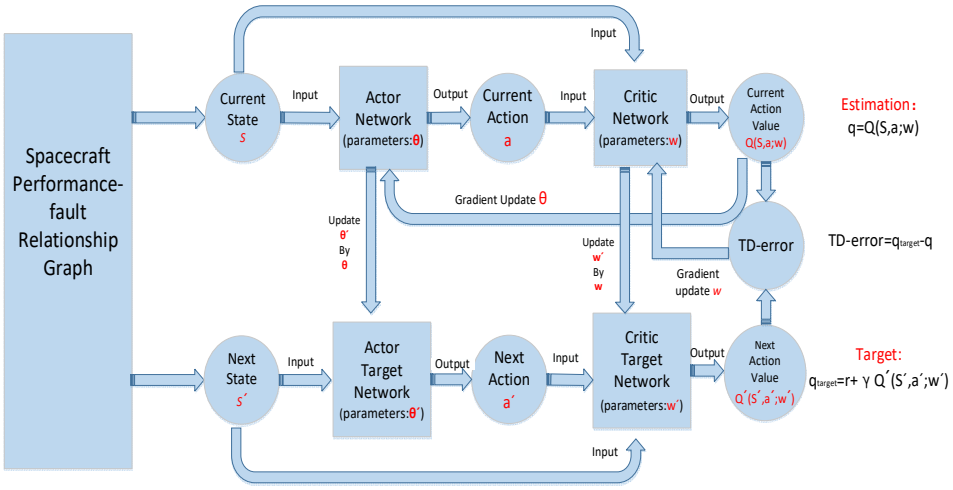

Figure 2. The structure of DDPG model.

2.4.1. Action Selection Based on Actor Network

We use the semantic relation and position multilayer perceptron (SPMLP) model as the actor network. We take the state vector as the input of the SPMLP model and output the action prediction vector. The model structure is shown in Figure 3.

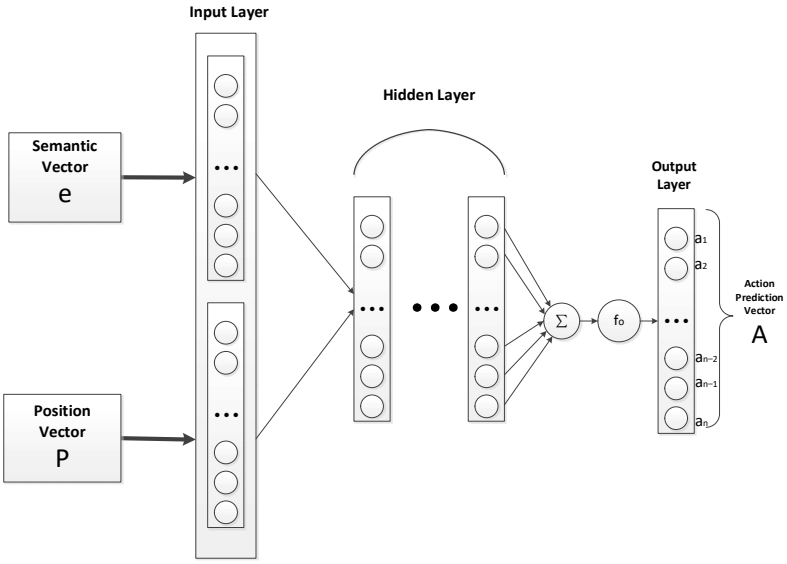

Figure 3. The structure of SPMLP model.

We take the entity vector $\boldsymbol{X}$ and reward R corresponding to the current state S. The entity vector $\boldsymbol{X}$ is used as the input vector of the SPMLP input layer. Each element of the entity vector $\boldsymbol{X}$ is multiplied by the weight θ_1 of the first hidden layer, and summed up. After adding the offset value b_1, the output $\boldsymbol{h_1}$ of this layer is obtained through the *sigmoid* activation function f_1, and the formula is as follows:

$$\boldsymbol{h_1} = f_1(\theta_1 \boldsymbol{X} + b_1) \tag{1}$$

Take the output of the first hidden layer as the input of the next hidden layer, and repeat Formula (1) to obtain the output of the last hidden layer as $\boldsymbol{h_t}$. The weight is θ_y and the offset is b_y. The activation function f_o of the output layer selects the *softmax* function. The final output action prediction vector $\boldsymbol{A}$ is as follows:

$$\boldsymbol{A} = f_o(\theta_y \boldsymbol{h_t} + b_y) \tag{2}$$

The element position of the prediction vector $\boldsymbol{A}$ corresponds to the relationship in the performance-fault relationship graph. The element A_i with the highest probability in $\boldsymbol{A}$ is the selected optimal action a. Find the next state S' corresponding to the optimal action from the environment matrix $\boldsymbol{E}$, take S' as the current state, and repeatedly take the above steps until the end state is reached. Finally, the agent outputs the relationship prediction path.

2.4.2. Parameters Updating Based on Value Network

The multilayer perceptron (MLP) model is adopted as the critic network, which takes the action vector and state vector as input and outputs the action value. The hidden layer structure of the model is the same as that of SPMLP, and the calculation process is similar to the previous section. The difference is that the action value output by the value network is a real number, the activation function of the output layer is the *Relu* function, and the network parameter is w.

We set-up the actor target network and the critic target network, and the structure is consistent with the actor network and the critic network. The actor target network selects the next optimal action a' according to the next state S', and the network parameter θ' is periodically copied from θ. The critic target network calculates the action value Q' (S, a', w') of the next state, and the network parameter w' is copied from w regularly. Put the {S, A, R, S', A'} quintuple of each cycle into the empirical playback set $\boldsymbol{D}$; take p samples {S_j, A_j, R_j, S_j', A_j'} from $\boldsymbol{D}$, where j = 0, 1, 2, ... p; calculate the current target Q value Q_{target}:

$$Q_{target} = R_j + \gamma Q'\left(S_j', a_j'; w'\right) \tag{3}$$

where γ is the discount factor, $\gamma \in (0, 1)$, and it is 0.9 in this paper.

Parameters w of the current value network are updated by gradient back propagation of the neural network, and MSE (Mean Square Error) is used as the loss function of the value network.

$$\mathrm{MSE} = \frac{1}{p}\sum_{j=1}^{p}\left(Q_{target} - Q\left(S_j, a_j; w\right)\right)^2 \tag{4}$$

Parameters θ of the current strategy network are updated by gradient back propagation of the neural network, and the loss function of the strategy network is:

$$J(\theta) = -\frac{1}{p}\sum_{j=1}^{p} Q\left(S_j, a_j; \theta\right) \tag{5}$$

Set the update frequency c and soft-update the parameters of the actor target network and critic target network every c cycle:

$$w' \leftarrow \tau w + (1-\tau)w' \tag{6}$$

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{7}$$

where τ is the renewal coefficient, is generally small, and is 0.1 in this paper.

3. Experimental Results and Discussion

3.1. Performance-Fault Relationship Graph of Spacecraft Control System

We apply the SPDDPG algorithm to the performance-fault relationship graph of the spacecraft control system constructed by us. First, based on the models of each part of the spacecraft control system, the attitude dynamics equation, and the attitude kinematics equation, we construct the performance-fault relationship graph of the spacecraft control system. Taking the system composed of three orthogonal gyroscopes, rolling axis infrared

earth sensors, and three orthogonal momentum wheels as the object, the attitude dynamics equation of the spacecraft is as follows:

$$I_x\dot{\omega}_x + \left(I_z - I_y\right)\omega_y\omega_z = -\dot{h}_x \tag{8}$$

$$I_y\dot{\omega}_y + (I_x - I_z)\omega_x\omega_z = -\dot{h}_y \tag{9}$$

$$I_z\dot{\omega}_z + \left(I_y - I_x\right)\omega_y\omega_x = -\dot{h}_z \tag{10}$$

where I_x, I_y, and I_z are the three-axis moments of inertia; ω_x, ω_y, and ω_z are the components of the spacecraft's space angular velocity along the main inertia axis; h_x, h_y, and h_z are the three-axis angular momenta of the momentum wheel.

The attitude kinematics equation of spacecraft is

$$\begin{cases} \omega_x = \dot{\varphi} - \omega_0\psi \\ \omega_y = \dot{\theta} - \omega_0 \\ \omega_z = \dot{\psi} + \omega_0\varphi \end{cases} \tag{11}$$

where φ, θ, and ψ are Euler angles, and ω_0 is the orbital angular velocity of the spacecraft rotating around the central gravitational body.

Infrared earth sensor model:

$$\varphi_h = \varphi + b_\varphi + N_{\varphi h} + f_\varphi \tag{12}$$

$$\theta_h = \theta + b_\theta + N_{\theta h} + f_\theta \tag{13}$$

where φ_h is the ground sensitivity measurement output of the roll-axis, b_φ is the ground sensitivity constant error of the roll-axis, $N_{\varphi h}$ is the ground sensitivity measurement noise of the roll-axis, f_φ is the ground sensitivity fault of the roll-axis, θ_h is the ground sensitivity measurement output of the pitch-axis, b_θ is the ground sensitivity constant error of the pitch-axis, $N_{\theta h}$ is the ground sensitivity measurement noise of the pitch-axis, and f_θ is the ground-sensitive fault of the pitch axis.

Gyro measurement model:

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} d_1 + b_1 \\ d_2 + b_2 \\ d_3 + b_3 \end{bmatrix} + \begin{bmatrix} f_{g,1} \\ f_{g,2} \\ f_{g,3} \end{bmatrix} \tag{14}$$

$$g_4 = \begin{bmatrix} \sqrt{3}/3 & \sqrt{3}/3 & \sqrt{3}/3 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + f_{g,4} \tag{15}$$

where g_i is the gyro measurement output, d_i is the gyro exponential drift term ($d_i = -\frac{1}{\tau_i}d_i$), b_i is the gyro constant drift term, $f_{g,i}$ is the gyro fault, and i = 1, 2, 3.

Momentum wheel model:

$$\dot{h}_x = J_1\dot{\overline{\omega_1}} = -u_x + f_{\overline{\omega},1} \tag{16}$$

$$\dot{h}_y = J_2\dot{\overline{\omega_2}} = -u_y + f_{\overline{\omega},2} \tag{17}$$

$$\dot{h}_z = J_3\dot{\overline{\omega_3}} = -u_z + f_{\overline{\omega},3} \tag{18}$$

where J_1, J_2, and J_3 are the rotary inertias of the momentum wheel; $\overline{\omega_1}$, $\overline{\omega_2}$, and $\overline{\omega_3}$ are the rotation speeds of the momentum wheel; u_x, u_y, and u_z are the expected output torques of the momentum wheel; $f_{\overline{\omega},1}$, $f_{\overline{\omega},2}$, and $f_{\overline{\omega},3}$ are the momentum wheel failures.

Based on the models of each part of the spacecraft control system, the entities and relationships of the performance-fault relationship diagram are extracted according to the following principles:

1. For each part of the spacecraft control system model, the known quantity and unknown quantity can be regarded as the "known quantity entity" and "unknown quantity entity", respectively.
2. The purpose of this experiment is to infer whether there is a relationship among variables, so all kinds of errors, noises, constant parameters, and faults are not considered when constructing the graph.
3. When there are multiple variables on one side of the equation, an "*AND entity*" needs to be added to represent them.
4. The relationships of variables in the equation are divided into proportional equivalence, derivative equivalence, equivalence, addition, subtraction, and multiplication.

According to the above principles, the extracted entities are shown in Table 1.

Table 1. Triple set of spacecraft control system model.

Known quantity entity	φ_h, θ_h, g_1, g_2, g_3, u_x, u_y, u_z, $\overline{\omega_1}$, $\overline{\omega_2}$, $\overline{\omega_3}$, h_x, h_y, h_z, AND
Unknown quantity entity	ω_x, ω_y, ω_z, g_4, φ, θ, ψ
Relation	equivalence, proportional equivalence, equivalence, derivative equivalence, addition, subtraction, multiplication
Triplet	(g_1 , proportional equivalence, ω_x); (ω_x , addition, AND); (ω_y , addition, AND); (ω_z , addition, AND); (g_4 , equivalence, AND)

According to the extracted entities and relationships, we construct the performance-fault relationship graph of the spacecraft control system, containing 24 entities, 5 relationships, and 28 triplets. As shown in Figure 4, the colors of connecting lines corresponding to different relationships are at the top.

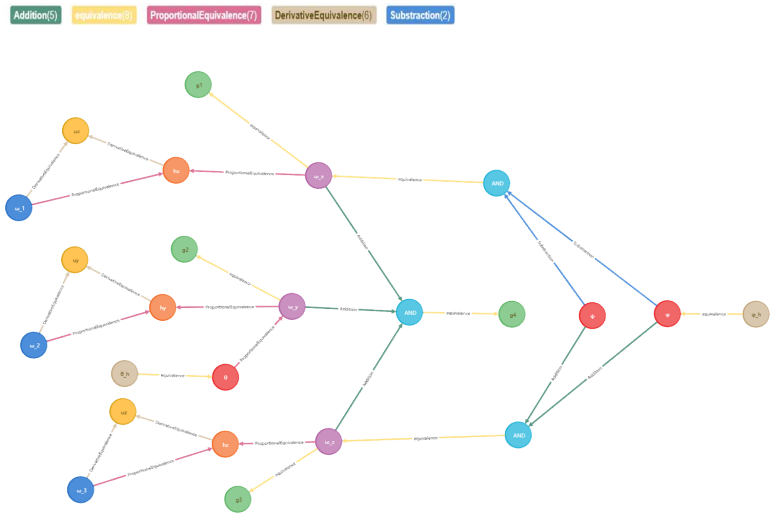

Figure 4. Performance-fault relationship graph of spacecraft control system.

3.2. Experimental Results of SPDDPG

We apply SPDDG to the performance-fault relationship graph of the spacecraft control system for relational reasoning and relational completion, and the parameters are set as follows. The output vector dimension of the TransE model is 100, and the dimension of the semantic vector after PCA dimension reduction is set to 20. The input of the actor network is a 42-dimensional state vector, and the output is a 7-dimensional predicted action vector. The number of neurons in the three hidden layers is set to (80, 80, 80), and the learning rate lr = 0.0005. The input of the critic network is a 40-dimensional vector, the output is a one-dimensional action value, the single hidden layer contains 60 neurons, and the learning rate lr = 0.0005. The training times are set to 60,000. Each training will randomly perform 100 pathfinding times. The longest relation path of the control system performance-fault relation graph is 7, so the training requirements can be met when the training times are reached or the average path-finding steps per hundred trainings are within seven steps. Actually, the training reaches the termination condition in 54,200 times, and the average path-finding step curve of the agent is as Figure 5:

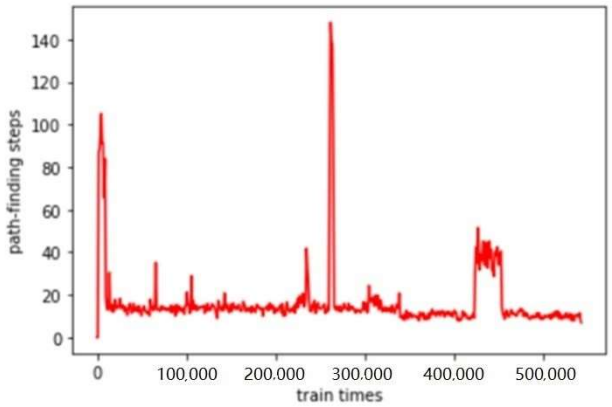

Figure 5. Average path-finding step learning curve.

We take the momentum wheel speed ω_1 as an example. It is taken as the initial state, and the infrared sensor output measurement φ_h is taken as the final state. The path-finding step of the agent is 5, which costs 0.08 s, and the visual path-finding process is shown in Figures 6 and 7:

```
Initial state: {entity:ω_1}
relation:proportional equivalence -->nextstate:{entity:hx}
relation:proportional equivalence -->nextstate:{entity:ω_x}
relation:equivalence-->nextstate:{entity:AND}
relation:subtraction-->nextstate:{entity:Φ}
relation:equivalence-->nextstate:{entity:Φ_h}
Terminal state:{entity:Φ_h}
Path-finding steps:5
```

Figure 6. Visual output result of program.

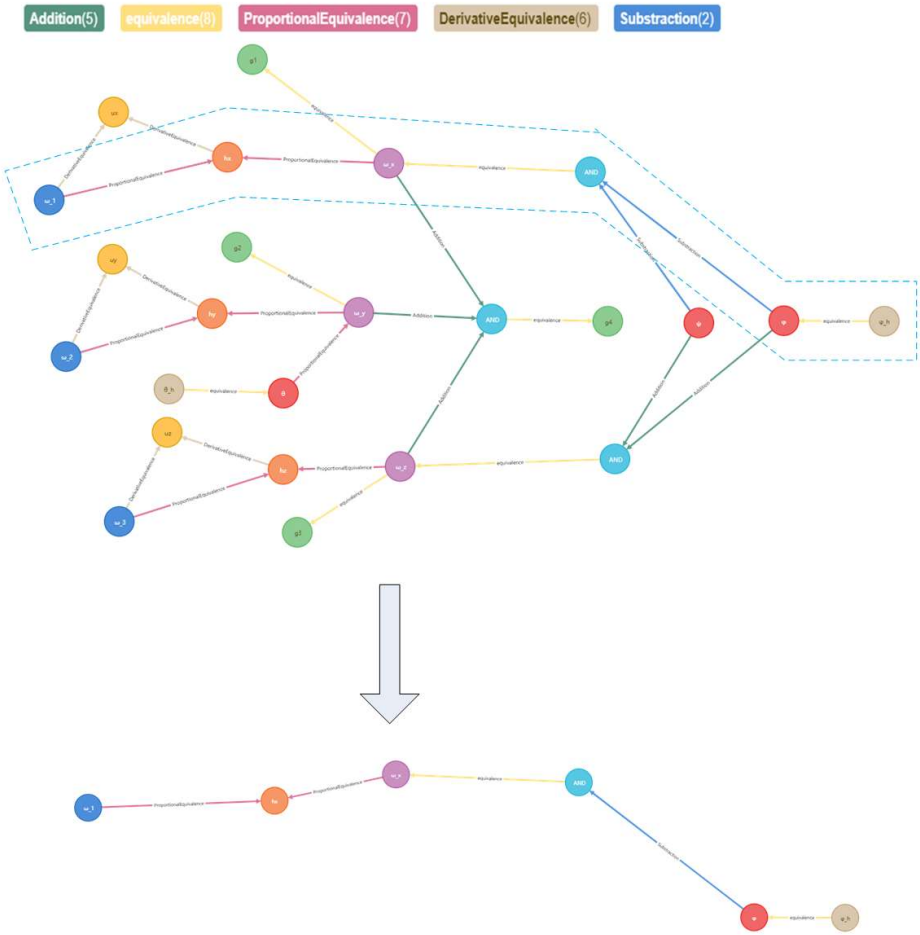

Figure 7. Visualization path finding process of Neo4j.

The computer directly predicts the existence of the relationship between entities, and manually determines the path of entities that are difficult for the computer to directly determine. We use SPPDPG to discover 51 new relationships, and the completed spacecraft performance-fault relationship graph is shown in Figure 8.

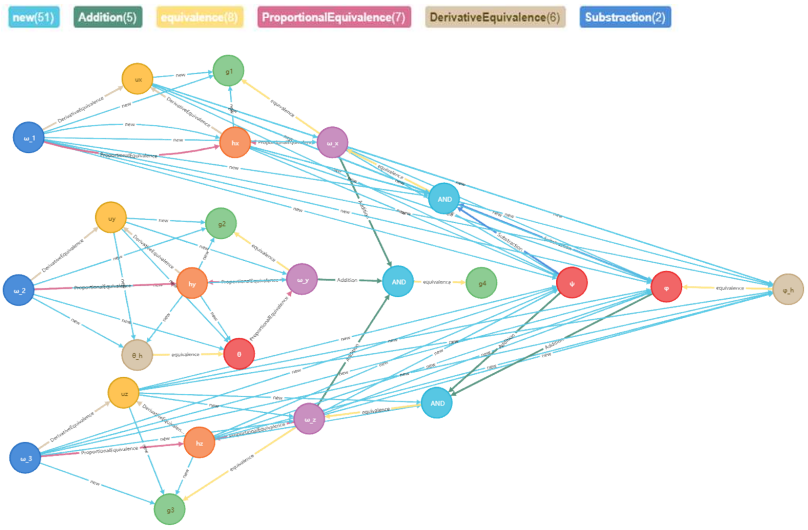

Figure 8. Completed performance-fault relationship graph of spacecraft control system.

3.3. Experimental Results of Various Models

To further verify the advantages of SPDDPG, we make a comparison between the performance of SPDDPG and other published models. Due to the practical application requirements, the relationship reasoning model applied to the spacecraft performance-fault relationship graph not only needs to predict the existence of the relationship between entities, but can also output the visual relationship path between entities for manual judgment. Therefore, we focus on selecting the reinforcement learning methods that meet the requirements, DeepPath [13] and MINERVA [15], and compare the experimental results from the path-finding accuracy and average path-finding steps. In order to verify the improvement of the DDPG algorithm, we add two DDPG models, DDPG(transE) and DDPG(State). DDPG(transE) only obtains the embedded vector of the entity and relationship through TransE, and DDPG(State) only sets the normal state and the termination state. The other parts of the two models are the same as SPDDPG. The experimental results are shown in Table 2, and the data are the best results obtained after many experiments.

Table 2. The comparison of experimental results.

Model	Path-Finding Accuracy	Average Path-Finding Steps
DeepPath	65.91%	19.43
MINERVA	72.86%	14.62
DDPG (transE)	77.64%	12.77
DDPG(State)	83.42%	14.29
SPDDPG	100%	6.91

The experimental results show that the SPDDPG algorithm has a good inference effect on the performance-fault relationship graph of the spacecraft control system. After the target entity is given, the accuracy of the agent finding the relationship path between entities reaches 100%, and the multi-hop relationship path between entities can be successfully obtained. In the performance-fault relationship graph, the longest relationship path is 7 steps, and the minimum average path-finding step number of agents is 6.91, which can control the path-finding step number to a few times and has high reasoning efficiency. In addition, compared with the experimental results of two kinds of DDPG models that have not been improved, two conclusions can be drawn: 1. The embedded vector extracted by semantic information and location information can contain more obvious entity features and improve the accuracy of relational reasoning. 2. Adding the imminent termination state to the normal state and the termination state can make the purpose of the agent's routing clearer, reduce the path-finding steps, and improve the efficiency of relational reasoning. Therefore, this algorithm provides an interpretable reasoning means for completing the relationship of the spacecraft performance-fault relationship graph and has application significance.

4. Conclusions

The spacecraft performance-fault relationship graph has complex physical meanings, numerous entities, and coupling relationships; we propose a relationship reasoning model SPDDPG to solve the above problems, which combines the representation learning model and deep reinforcement learning model. Its advantages are as follows:

(1) The representation learning model is used to extract the semantic features of the entity and relationships, and the global location features of entities are obtained through Boolean information conversion. PCA is used to reduce the dimensions of entity vectors, to retain the high-order features of entities and avoid overfitting. It helps overcome the difficulty of distinguishing numerous entities, which is beneficial to the training of neural networks.

(2) The actor network is used to replace the traditional action selection strategy, and the critic network is used to fit the complex and uncertain value function. The deep neural

network can distinguish complex physical meanings of the spacecraft performance-fault relationship graph and improve the efficiency of relationship reasoning.

We apply SPDDPG to the performance-fault relationship graph of the spacecraft control system, and the experimental results show that the algorithm has a good relationship reasoning effect on the knowledge graph of the spacecraft system level. In the future work, we are going to build a model of relationship prediction based on SPDDPG's relationship reasoning path, so that it has both predictive ability and interpretability. We will apply this model to a larger spacecraft performance-fault relationship graph, and improve the reasoning efficiency.

Author Contributions: Conceptualization, X.X. and S.W.; methodology, X.X.; software, X.X.; validation, W.L., X.X. and S.W.; formal analysis, X.X.; investigation, W.L.; resources, S.W.; data curation, S.W.; writing—original draft preparation, X.X.; writing—review and editing, W.L.; visualization, X.X.; supervision, S.W.; project administration, S.W.; funding acquisition, S.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62022013 and National Key R&D Program of China grant number 2021YFB1715000.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hogan, A.; Blomqvist, E.; Cochez, M. Knowledge graphs. *J. ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–37.
2. Liu, Q.; Li, Y.; Duan, H.; Liu, Y.; Qin, Z. Knowledge graph construction techniques. *J. Comput. Res. Dev.* **2016**, *53*, 582–600.
3. Zhou, Y.C.; Wang, W.J.; Qiao, Z.Y.; Xiao, M.; Du, Y. A survey on the construction methods and applications of scitech big data knowledge graph. *Sci. Sin. Inf.* **2020**, *50*, 957–987. [CrossRef]
4. Dong, X.; Gabrilovich, E.; Heitz, G.; Horn, W.; Lao, N.; Murphy, K.; Strohmann, T.; Sun, S.; Zhang, W. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 601–610.
5. Nickel, M.; Murphy, K.; Tresp, V.; Gabrilovich, E. A review of relational machine learning for knowledge graphs. *Proc. IEEE* **2015**, *104*, 11–33. [CrossRef]
6. Wang, Q.; Mao, Z.; Wang, B.; Guo, L. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Trans. Knowl. Data Eng.* **2017**, *29*, 2724–2743. [CrossRef]
7. Lin, Y.; Han, X.; Xie, R.; Liu, Z.; Sun, M. Knowledge representation learning: A quantitative review. *arXiv* **2018**, arXiv:1812.10901.
8. Paulheim, H. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semant. Web* **2016**, *8*, 489–508. [CrossRef]
9. Wu, T.; Qi, G.; Li, C.; Wang, M. A Survey of Techniques for Constructing Chinese Knowledge Graphs and Their Applications. *Sustainability* **2018**, *10*, 3245. [CrossRef]
10. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, r5–r10.
11. Lao, N.; Mitchell, T.; Cohen, W. Random walk inference and learning in a large scale knowledge base. In Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27–31 July 2011; pp. 529–539.
12. Lao, N.; Minkov, E.; Cohen, W. Learning relational features with backward random walks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, 26–31 July 2015; pp. 666–675.
13. Xiong, W.; Hoang, T.; Wang, W.Y. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv* **2017**, arXiv:1707.06690.
14. Wang, H.; Li, S.; Pan, R.; Mao, M. Incorporating graph attention mechanism into knowledge graph reasoning based on deep reinforcement learning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 2623–2631.
15. Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv* **2017**, arXiv:1711.05851.
16. Shen, Y.; Chen, J.; Huang, P.S.; Guo, Y.; Gao, J. M-walk: Learning to walk over graphs using monte carlo tree search. *arXiv* **2018**, arXiv:1802.04394.
17. Zeng, X.; He, S.; Liu, K.; Zhao, J. Large scaled relation extraction with reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.

18. Lin, X.V.; Socher, R.; Xiong, C. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 3243–3253.
19. Fu, C.; Chen, T.; Qu, M.; Jin, W.; Ren, X. Collaborative Policy Learning for Open Knowledge Graph Reasoning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 2672–2681.
20. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv* **2013**, arXiv:1312.6203.
21. Schlichtkrull, M.; Kipf, T.N.; Bloem, P.; Van Den Berg, R.; Titov, I.; Welling, M. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*; Springer: Cham, Switzerland, 2018; pp. 593–607.
22. Vashishth, S.; Sanyal, S.; Nitin, V.; Talukdar, P. Composition-based multi-relational graph convolutional networks. *arXiv* **2019**, arXiv:1911.03082.
23. Shang, C.; Tang, Y.; Huang, J.; Bi, J.; He, X.; Zhou, B. End-to-end structure-aware convolutional networks for knowledge base completion. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3060–3067.
24. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
25. Nathani, D.; Chauhan, J.; Sharma, C.; Kaul, M. Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4710–4723.
26. Xie, Y.; Zhang, Y.; Gong, M.; Tang, Z.; Han, C. Mgat: Multi-view graph attention networks. *Neural Netw.* **2020**, *132*, 180–189. [CrossRef] [PubMed]
27. Chami, I.; Wolf, A.; Juan, D.C.; Sala, F.; Ravi, S.; Ré, C. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 6901–6914.
28. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
29. Wang, S.; Wei, X.; Nogueira dos Santos, C.N.; Wang, Z.; Nallapati, R.; Arnold, A.; Xiang, B.; Yu, P.S.; Cruz, I.F. Mixed-curvature multi-relational graph neural network for knowledge graph completion. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 1761–1771.
30. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
31. Abdi, H.; Williams, L.J. Principal component analysis. *Wiley Interdiscip. Rev. Comput. Stat.* **2010**, *2*, 433–459. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.