J Intell Robot Syst (2008) 51:187–201
DOI 10.1007/s10846-007-9186-1

Tri-tier Immune System in Anti-virus and Software Fault Diagnosis of Mobile Immune Robot Based on Normal Model

Tao Gong · Zixing Cai

Received: 13 October 2006 / Accepted: 30 September 2007 /
Published online: 31 October 2007

Abstract In this paper, anti-virus problem and software fault diagnosis of mobile robot, an immune robot, is discussed with proposal of a novel tri-tier immune system (TTIS). TTIS is a novel artificial immune system, which is comprised of three computing tiers and based on the normal model. The three tiers include inherent immune tier, adaptive immune tier and parallel immune tier. The tri-tier immune model is built on some theories of human immune system and has many good features, such as adaptability, immunity, memory, learning, and robustness. At the same time, for such immune robot, a novel normal model for the robot software is also proposed. The normal model is built on the space–time properties of each component in the robot software and can uniquely identify the normal state of the robot software. Such tri-tier immune system based on the normal model is suitable for anti-virus and fault diagnosis, which enable the immune robot to detect all viruses and faults in the robot software, recognize many viruses and faults, eliminate the viruses and faults, and repair the damaged robot software to its normal state. Meanwhile, simulation results show that the tri-tier immune system has the properties of immunity, security and robustness.

Keywords Artificial immune system · Normal model · Immune robot · Anti-virus · Fault diagnosis

1 Introduction

Robot is a biomimetic technique inspired from some animals and human beings in nature, and biomimetic robots/systems are also inspired from the natural mechanisms of some creatures. So the robot technique is developed on the basis of the bioinformatics in nature.

T. Gong (✉)
College of Information Science and Technology, Donghua University, Shanghai 201620,
People's Republic of China
e-mail: taogongchina@gmail.com

T. Gong · Z. Cai
School of Information Science and Engineering, Central South University, Changsha, Hunan 410083,
People's Republic of China

For example, based on the artificial immune system (AIS) the immune robot was proposed and investigated, which was suitable for dangerous environments [1]. The biomimetic robot navigation was investigated on the behavior descriptions [2]. The basic motion control of a free swimming biomimetic robot fish was designed [3]. In this paper, a novel tri-tier immune model is used to improve anti-virus and fault diagnosis of the robot software, which is a mobile robot simulator.

Immune computation is a biomimetic mechanism inspired from the natural immune system of human beings and other creatures. The artificial immune system is very complex, adaptive and intelligent. Many diseases, such as smallpox and measles, have been recognized and eliminated by human immune system [4, 5]. On the other hand, the artificial immune system has been used to detect, recognize and eliminate the computer viruses, especially in the network such as Internet [6–8]. And the technique has been also used to detect faults in the hardware and software of the computer system [9–10]. In this paper, the immune model is comprised of three tiers. The inherent immune tier and the adaptive immune tier are inspired from the natural immune system. And the parallel immune tier is based on the parallel computer technique.

2 Tri-tier Immune Model Based on Normal Model

The tri-tier immune model of the robot software is based on the normal model of the software system, and the normal model is built on the space–time properties of each component in the software. The space property of each file is represented with the absolute pathname of the file, and the time property of each file is represented with the last change time of the file. The absolute pathname is a term in the operation system, and the last change time refers to the last time that the file is changed. The tri-tier immune model is inspired from some natural immune theories, and includes inherent immune tier, adaptive immune tier and parallel immune tier, shown in Fig. 1. The parallel immune tier is built on the parallel computing theorems in [11], and used to increase efficiency. When the antigen enters the AIS, the random detectors begin to detect whether the antigen is self or non-self, by matching the features of the antigen with the self information in the self database. The self is the part of the AIS, such as the system file of the AIS. And the non-self is not any part of the AIS or compatible with the AIS. The self information is used to define the features of the self and represent the normal state of the AIS. And all self information is stored in the self database.

In Fig. 1, the immunization process is maximizing the percent of the selfs and minimizing the amount of the non-selfs in the artificial immune system. When an antigen is determined as a non-self, pattern recognition of the non-self is started in two ways. One is the way of matching features, and the other is the way of matching rules. The former is done through querying records in the database and matching the feature information of the detected non-self with the record information in the non-self database, where the entire known non-selfs store. The latter is done in a random way through searching some antibodies in the rule base. If the non-self is known by the AIS (i.e. its features match some non-self information in the non-self database), then the destroyer is called to eliminate the non-self. For computer viruses and software faults, the deletion command in the operating system is a kind of destroyer. Otherwise, the rule matching is used to recognize the non-self by the antibody and the rule-base on the adaptive immune tier. The rule includes two parts: the first one is the condition of the non-self feature, and the second one is the conclusion of the rule-based reasoning, which shows the type of the non-self and the elimination

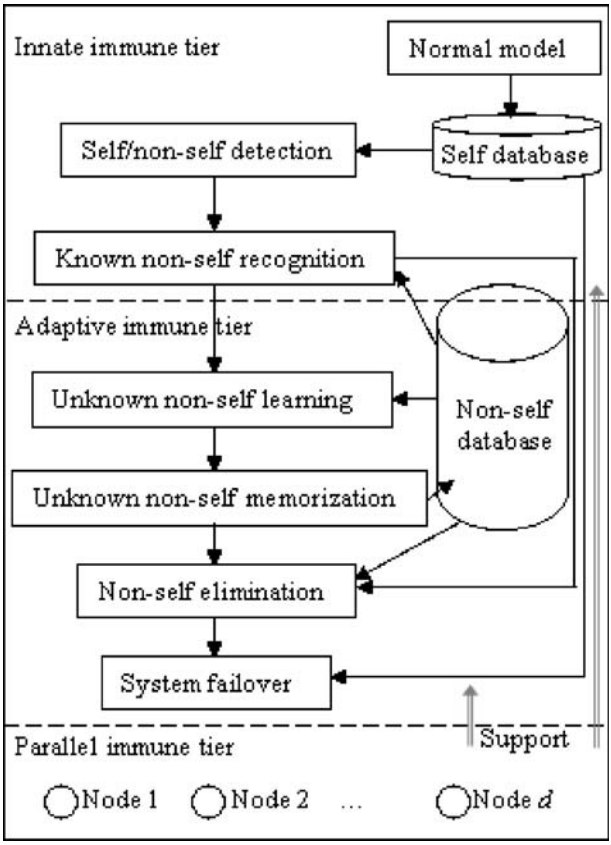

Fig. 1 Tri-tier immune model based on normal model

approach of the non-self. The rule matching is similar to the combination of DNA genes. And the immune algorithm is built on the random search of the rules. If the random search is done through evolutionary algorithm, then the immune algorithm is built on the evolutionary algorithm [12–14]. Cooperative co-evolutionary adaptive genetic algorithm (CCAGA) is suitable for parallel computation, which is convenient to solve complicated problems [15].

In the immune model, the immune computation has the threshold alike in the immune response of the natural immune system. And the threshold is caused by the long-time cost of the random search. Moreover, the antibody search and rule matching are large-scaled in the chaos state.

3 AIS for Anti-virus and Fault Diagnosis of Mobile Immune Robot

In unknown environment such as the Mars and the moon, mobile robot often encounters with dangerous obstacles such as fire, trap, earthquake and enemy. In Internet/wireless environment, the web control based mobile robot has the same bottleneck of anti-virus security as the web applications on computers. Both viruses and dangerous environments can cause faults in the robot software, and the reasons for the faults include wrong change and missing component(s) in the software.

Suppose a robot software is represented as S, a component of the software S is represented as c_i, the absolute pathname of the component c_i is represented as p_i, the last change time of the component c_i is represented as t_i. The normal state of the component c_i is uniquely identified by its space–time properties p_i and t_i [16]. Moreover, the normal state of the robot software is uniquely identified by the normal states of all the components in the robot software. Therefore, the normal state of the robot software can be uniquely represented with the space–time properties of all the components in the software.

$$f : \{s(S)|N(s(S)) = 1\} \rightarrow \{< p_i, t_i > |N(s(c_i)) = 1, i = 1, 2, \cdots, n\} \quad (1)$$

Here, f denotes a mapping function from the normal model to the self database, $s(x)$ denotes the state function of an object x, n denotes the sum of the components in the robot software, and $N(\cdot)$ denotes the normal function that is defined as such.

$$N(y) = \begin{cases} 1 & , y \text{ is a normal state} \\ 0 & , y \text{ is an abnormal state} \end{cases} \quad (2)$$

The mobile robot is a good test-bed for the immune model [1], and the mobile robot simulator is also a good test-bed of the immune model, as shown in Fig. 2.

3.1 Encoding of Normal Model

The Java-based robot simulator in Fig. 2 is comprised of 230 files and 9 directories, and without any file in the simulator software the system will not work normally. Each file or directory has its unique absolute pathname according to the managing rules of the operation system, and the absolute pathname can be read with the Java method *getAbsolutePath*(). At any time, the last change time of each file or directory is also unique, and the last change time can be read with the Java method *lastModified*(). For a file c_i of the robot simulator S,

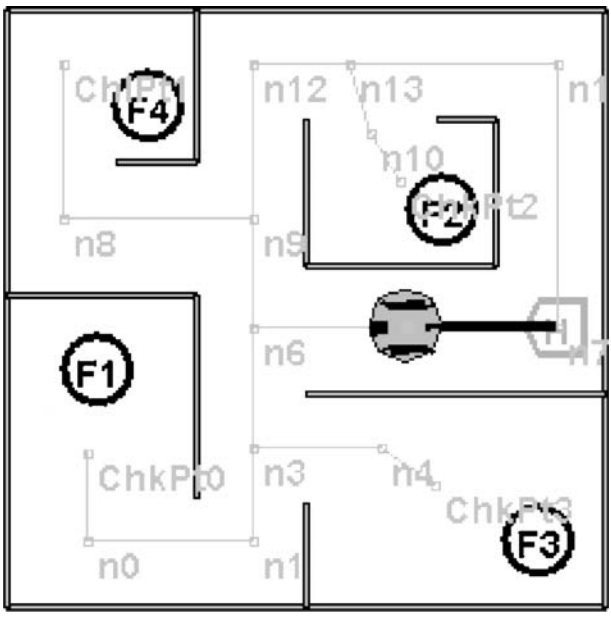

Fig. 2 Simulator of a fire-fighting robot

the absolute pathname of the file c_i can read and denoted as p_i, and the last change time of the file c_i can also be read and denoted as t_i. The values of both the space property p_i and the time property t_i are encoded as a record in the self database *SDB*.

$$\langle p_i, t_i \rangle \rightarrow \text{SDB} \tag{3}$$

Theorem 1 When all the files of the robot software S are normal, the space–time properties$\{\langle p_i, t_i \rangle | N(s(c_i)) = 1, i = 1, 2, \cdots, n\}$ of all the files uniquely identify the normal states(S), $N(s(S)) = 1$ of the robot software S [16].

Through mapping from the space–time properties$\{\langle p_i, t_i \rangle\}$ of the files in the robot simulator to the self database *SDB*, the normal model of the robot simulator is built. In a visual 3D graph, the normal model can be visualized as some cells, as shown in Fig. 3.

In Fig. 3, the visualization of the mobile robot simulator is modeled in three dimensions, which include the hierarchy, the directory and the file. The hierarchy means the hierarchy of the directory where the file lies. The directory means the number of the directory under some hierarchy. And the file means the number of the file in some directory.

3.2 Virus/Fault Detection

Non-AIS abnormity detection is based on recognizing the features of the abnormity such as virus or fault, and traditional AIS-based detection is also based on this idea, which is called non-self detection [13, 17, 18]. After the normal model of the robot software is established, the selfs of the software are uniquely identified, and thus the non-self detection can be carried out through detecting self(s). For an object in the robot software, if the object is not detected as a self, then the object is determined as a non-self.

$$d(o) = \begin{cases} 1 & , \langle p_o, t_o \rangle \in \text{SDB} \\ 0 & , \langle p_o, t_o \rangle \notin \text{SDB} \end{cases} \tag{4}$$

Here, $d(\cdot)$ denotes the detector function of the artificial immune system, o denotes an object in the robot software, p_o denotes the absolute pathname of the object o, t_0 denotes the

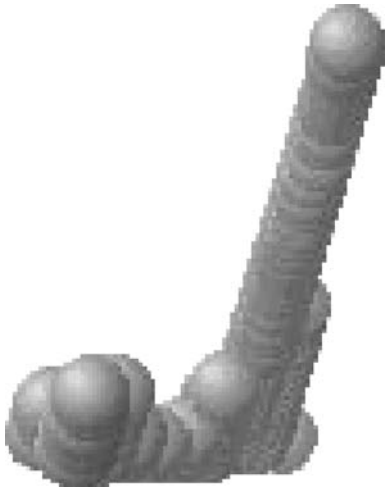

Fig. 3 Visualization of the mobile robot simulator

last change time of the object o, 1 represents that the object o is determined as a self, and 0 represents that the object o is determined as a non-self.

The non-self may be a kind of virus or fault, and the type of the non-self can be recognized in the next step. Pattern recognition of the non-self is carried out by matching the features of the non-self in the non-self database.

Theorem 2 Based on the normal model of the artificial immune system S, the detection rate of the self is 100% and the detection rate of the non-self is also 100% [16].

3.3 Virus Recognition and Fault Recognition

In fact, the non-self is strictly a kind of fault, but not always a kind of virus. If the non-self is not a virus, then the non-self is just a fault. Pattern recognition of viruses can be categorized into two types; one is direct recognition of known viruses in the innate immune tier, the other is adaptive recognition of unknown viruses in the adaptive immune tier. The direct recognition is based on matching the features of known viruses through querying in the virus database, and the adaptive recognition is based on learning the types of unknown viruses from all known viruses and memorizing the learnt ones.

The faults of the robot software include two types; one is caused by changing some components or adding foreign objects in the software, the other is caused by missing some components of the software.

$$u(o) = \begin{cases} 1 & , \langle p_o, t_o \rangle \notin \text{SDB}, o \in S \\ 0 & , \langle p_o, t_o \rangle \in \text{SDB}, o \notin S \end{cases} \tag{5}$$

Here, $u(\cdot)$ denotes the function of fault diagnosis, o denotes an object, p_0 denotes the absolute pathname of the object o, t_0 denotes the last change time of the object o, 1 represents that the fault is caused by the non-self o, and 0 represents that the fault is caused by missing the component o.

The viruses in the robot software can be known or unknown; known viruses are all recorded in the virus database *VDB*, and unknown viruses are not recorded in the virus database *VDB* at first, but after the unknown viruses are learnt by the AIS their features are memorized into the virus database *VDB* in the end. Therefore, the unknown viruses can be transformed into known ones by the AIS.

$$v(o) = \begin{cases} 1 & , o \in \text{VDB} \\ 0 & , o \notin \text{VDB} \end{cases} \tag{6}$$

Here, $v(\cdot)$ denotes the anti-virus function, o denotes a possible virus, 1 represents that the virus o is known, and 0 represents that the virus o is unknown.

The unknown viruses can be learnt with some learning mechanisms such as an improved BP neural network (IBPNN), RBF neural network, evolutionary learning, and example-based learning [16].

3.4 IBPNN-based Learning of Unknown Viruses and Learning Example

In natural immune system, adaptive learning of immune cells against unknown viruses is a kind of very complex process, which is even known little by doctors and immunologists. Alike, the adaptive learning of the artificial immune system against unknown computer

viruses is also very complex. To explore the secret, the improved BP neural network is built and used for the adaptive learning, and then the robot simulator S is immunized on its normal model and the improved BP neural network.

The BP neural network consists of three tiers, which include input tier, hidden tier and output tier, and one of its examples is shown in Fig. 4 [16, 19]. In the input tier of the example, 5 features of the known viruses are represented as $\{x_i | i = 1, 2, \cdots, 5\}$, and these features include coding language, proliferation manner, engine, feature string and damage [16]. In the hidden tier, 3 types of the known viruses are represented as$\{x_i | i = 6, 7, 8\}$. In the output tier, the most similar type of the known virus for the unknown non-self or its new type is represented as x_9. The IBPNN structure in Fig. 4 is the initial state, and any new type of unknown viruses that have been learnt can be added into the hidden tier as a new node. The new function of updating the hidden tier is improved in this neural network, which is different from other BP neural networks. Of course, new feature for recognizing unknown viruses can also be added into the input tier, but that case will be more complex.

In Fig. 4, x_{new} represents a new type of unknown virus that has been learnt, and the new type can be added into the BP neural network as a new node of the hidden tier. The BP neural network satisfies the following formulas.

$$\begin{aligned} P &= \sum_y \left(\sum_z \left(d_{yz} - O_{yz} \right)^2 \right) \\ \Delta W_{i \to j} &= r O_j (1 - O_j) \beta_j \\ \beta_j &= \sum_k w_{j \to k} O_k (1 - O_k) \beta_k \\ \varepsilon_z &= d_z - O_z \end{aligned} \tag{7}$$

Here, P represents the performance of the BP neural network; y represents the training input; z represents the output node; d_{yz} represents the anticipant output of the node z through the input y, and d_j represents the anticipant output of the jth node; O_{yz} represents the actual output of the node z through the input y, and O_j represents the actual output of the jth node; $w_{i \to j}$ represents the weight value between the nodes of the ith tier and those of the

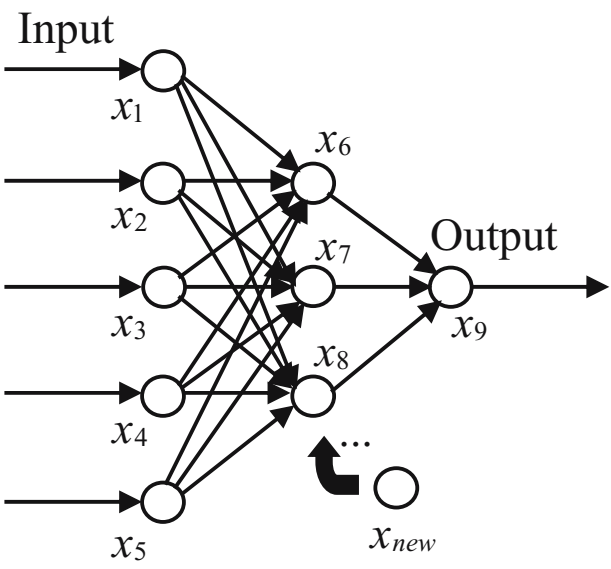

Fig. 4 Structure of the improved BP neural network

jth tier, and $\Delta W_{i\to j}$ represents the weight value between the nodes of the ith tier and those of the jth tier, and $\Delta W_{i\to j}$ represents its change; r represents the learning rate; β_j represents the value of the jth node, ε_z represents the error of the output node. Let the threshold for classifying the known viruses be σ, if virus $v \notin \rho(x_j, \sigma), j = 6, 7, 8$, then the virus v is unknown and can be given a new type x_{new}. Here, $\rho(x_j, \sigma)$ represents the neighbor space from the node x_j with the neighbor radius σ.

Example 1 Suppose the artificial immune system S has learnt to recognize and eliminate only 3 types of viruses, which are the loveletter worm, the happytime worm and the Jessica worm. So for the immune system S, other worms and the viruses that are not worms are unknown. In the example, a new worm CrazyVBS.vbs is unknown for the immune system S, and the feature string of the new worm is different from that of any type of 3 known worms. Besides, the coding language, the proliferation manner, engine and damage of the new worm are same as those of the known loveletter worm.

At first, the BP neural network is trained with all examples of 3 known types of worms, and the curve for training the BP neural network is shown in Fig. 5. Here, the amount for training the neural network is represented as m, and its value is about 28,000.

In Fig. 5, when the BP neural network has been trained for 22,800 times with the data of all the known worms, the error of the output node in the neural network becomes enough small.

Second, the important features of the new worm CrazyVBS.vbs are attained and represented as a feature vector (1,1,1,4,1), and the values of the five dimensions for the feature vector denotes the properties of coding language, proliferation manner, engine, feature string and damage.

Third, the feature vector of the new worm CrazyVBS.vbs is input into the trained BP neural network, and the output of the neural network is the result for learning the new worm. The output of the neural network is 1.0000132865184477, which means that the first known worm is most similar to the unknown worm.

At last, the type of the new worm CrazyVBS.vbs is added as a new node into the hidden tier of the improved BP neural network shown in Fig. 6, and the features of the new worm are memorized in the artificial immune system S.

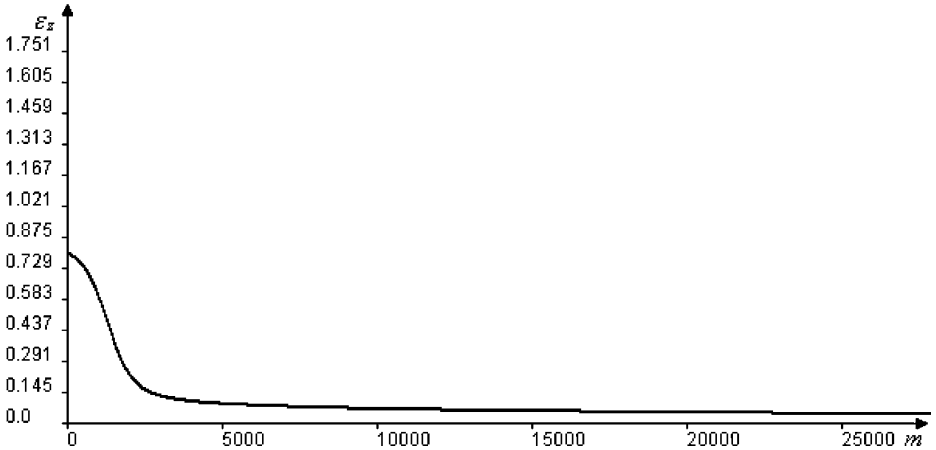

Fig. 5 Curve for training the BP neural network

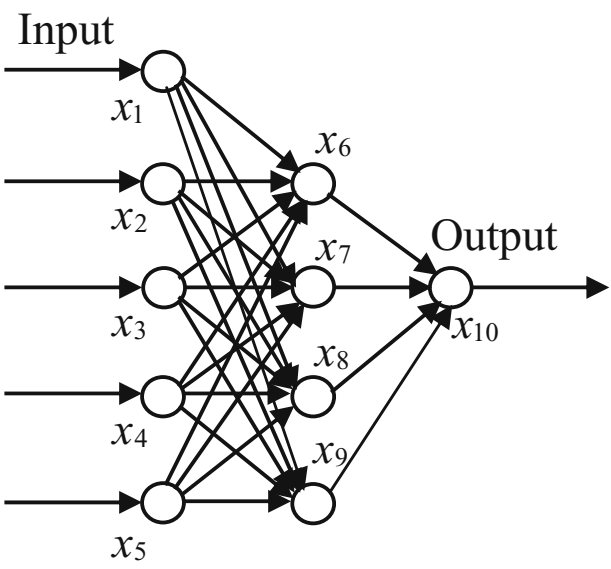

Fig. 6 Structure of the improved BP neural network after learning the new worm

In Fig. 6, the type of the new worm CrazyVBS.vbs is represented as the new node x_9 of the hidden tier in the improved BP neural network, and the old output node x_9 is named with the new name x_{10}. Therefore, the structure of the improved BP neural network can be expanded by learning new viruses, and the improved BP neural network is dynamic during learning.

Example 2 Suppose the artificial immune system S has learnt to recognize and eliminate only 3 types of viruses, which are the loveletter worm, the happytime worm and the Jessica worm. In the example, 7,168 viruses are not worms and are all unknown for the immune system S. The features of the non-worm viruses are different from those of any type of 3 known worms, and they are classified into several types. The new types of the non-worm viruses are added into the hidden tier of the improved BP neural network in Fig. 4 after learning the non-worm viruses with the trained BP neural network in Fig. 5, and the new structure of the BP neural network is shown in Fig. 7.

In Fig. 7, the types of the non-worm viruses are represented as the new node from x_9 to x_{8+l} of the hidden tier in the improved BP neural network, and l represents the amount of the types for the non-worm viruses.

3.5 Viruses Elimination and Self Repair

After all the non-selfs are detected and recognized, the non-selfs can be eliminated with some tools such as *delete* command of the operation system. If a virus file is opened at that time, the virus file will have to be closed before it is deleted.

$$\{o|\langle p_o, t_o\rangle \notin \text{SDB}, o \in S\} \rightarrow 0$$

After the step of eliminating the non-selfs, all the changed files of the robot simulator have been deleted. At that time, other files in the software are all normal, but some files are

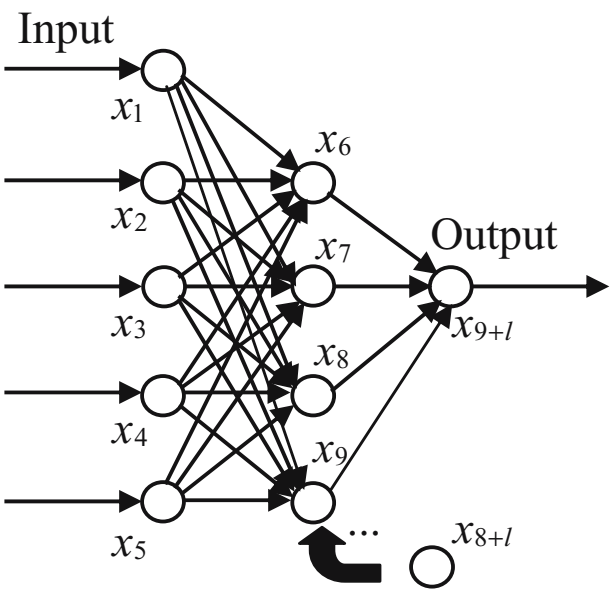

Fig. 7 Structure of the improved BP neural network after learning the non-worm viruses

missing. To repair the damaged robot simulator, the step of self repair is activated by repairing the missing files with their backup files.

$$\{o|\langle p_o, t_o\rangle \in \text{SDB}, o \notin S\} \rightarrow \{o|\langle p_o, t_o\rangle \in \text{SDB}, o \in S\}$$

In the example of the mobile robot simulator, the computer viruses are three worms. One is the love worm, and the second is the happy-time worm. The former copies itself to the computer disk, and spreads through e-mails. And the latter spreads all through e-mails. Both the two worms destroy the operation system by overwriting the system files with the worm files and deleting the crucial files. The last worm is a variant of the love worm.

4 Simulations

The immune model is simulated for the mobile robot simulator and the three worms. An immune program is developed with JDeveloper tool and used to detect, recognize and eliminate the worms in the mobile robot simulator. And the simulation results are shown with the visualization technique based on the Java Applet.

After the three worms enter the mobile robot simulator and infect some files of the simulator, the immune program detects every file by matching the features of the file with the information in the self database. The detection results of the immune program show in Fig. 8.

In Fig. 8, 20 non-self files are detected, including the files infected by the love worm, the happy-time worm and the variant of the love worm. After these files are detected, the files are recognized through feature matching in the non-self database and learning with the improved BP neural network. The variant of the love worm is unknown for the artificial immune system *S*, and the results for recognizing the non-self files and learning the variant show in Fig. 9.

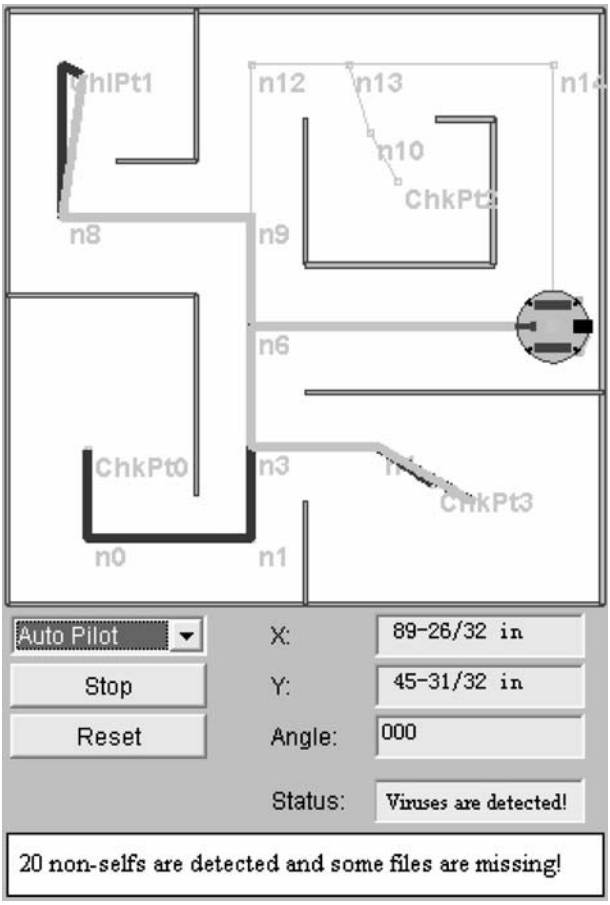

Fig. 8 Detection results of the immune program

In Fig. 9, the non-self files are recognized as the infected files of the happy-time worm through the string feature of 'happy time' etc. And the non-self files are recognized as the infected files of the love worm through the string feature of 'loveletter' and '.copy' etc.

After the visualization of the results, the recognition results show in Fig. 10.

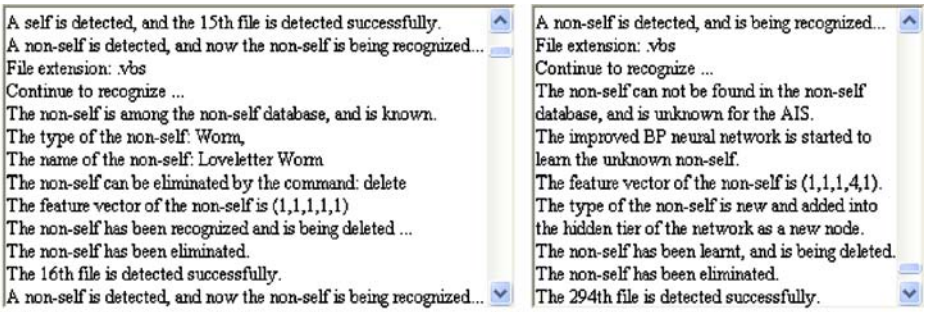

Fig. 9 Results for recognizing the non-self files and learning the variant

Fig. 10 Visualization of the recognition results

In Fig. 10, the visualization graph of the recognition results is represented with three colours and molecules. The black molecules represent the self files. The white molecules represent the non-self files infected by the love worm. And the grey molecules represent the non-self files infected by the happy-time worm.

After the non-self recognition, the infected files are eliminated immediately through the deletion command of the operation system. When the infected file is useful system file of the mobile robot simulator, such as the file test.htm, the file is marked in the useful-file database before it is deleted. The results of the non-self elimination show in Fig. 11.

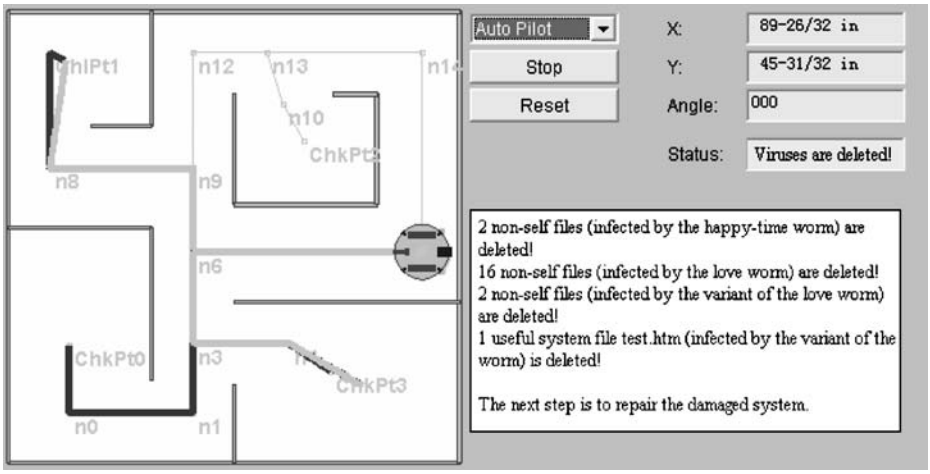

Fig. 11 Elimination results of the non-self files

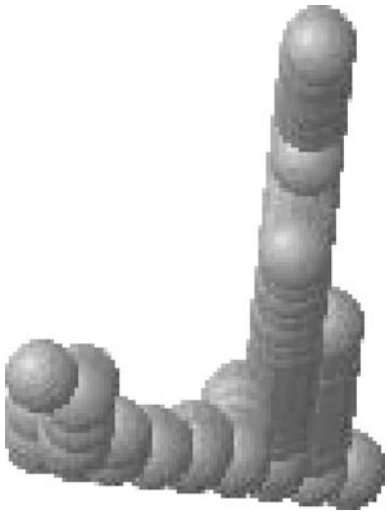

Fig. 12 Visualization of the non-self elimination

In Fig. 11, 2 non-self files, which are infected by the happy-time worm, are deleted. And 16 non-self files, which are infected by the love worm, are deleted. And 2 non-selfs, which are infected by the variant of the love worm, are deleted. One of the two non-self files is the system file test.htm, and it is marked in the useful-file database before it is deleted.

And the visualization graph of the elimination results shows in Fig. 12.

In Fig. 12, some non-self files in Fig. 6 are deleted. After the system file test.htm is infected, marked and deleted, the file is repaired with its backup file in the backup disk. And the failover results of the system file in the mobile robot simulator show in Fig. 13.

In Fig. 13, the black molecules represent the self files, and the yellow molecule represents the repaired file test.htm. After the virus elimination and the failover of the useful system file test.htm, the mobile robot simulator restores normal and can be started with the Web explorer again. The system file test.htm shows normal, though it has been infected by the variant of the love worm, deleted, and then repaired by the immune program.

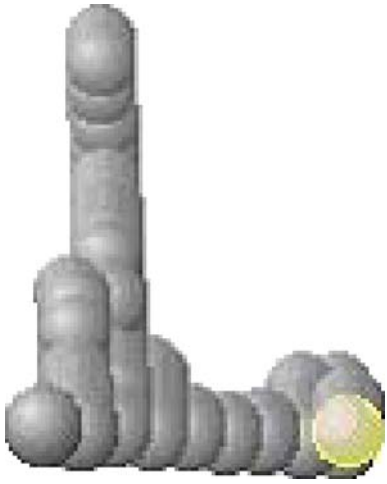

Fig. 13 Visualization of the failover results

The above example of the mobile robot simulator and its immune simulation show that, the non-self detection, the non-self recognition, the non-self elimination and the failover of the useful system files are all effective in the theories and experiments. So the tri-tier immune model is correct and useful in the applications of the mobile robot simulator. Moreover, further work will be emphasized on the complex theories of the more complex artificial immune system and the complex applications in the real world.

Above all, anyway, the immune approach is another biomimetic technique inspired from the natural immune system, like the robot technique inspired from some animals and human beings in nature.

5 Conclusions

Like the robot, the immune model is also biomimetic. Besides, the robot is a good test-bed of the burgeoning immune technique. And the immune robot will be a new useful robot in the kingdom of robots.

In this paper, the tri-tier immune model has been proposed and analyzed for the immune robot. And the simulation of the immune application to the mobile robot simulator shows that, the immune technique is effective in the non-self detection, the non-self recognition, the non-self elimination and the failover of the useful system files. More investigations and applications will be done on the more complex theories and real-world problems of the AIS technique and the robot technique.

Acknowledgments We sincerely thank editors and reviewers for their good advice, and greatly thank support from the National Natural Science Foundation of China under Grant 60234030 & 60404021 and the Foundation of Donghua University under Grant 104-10-0044017.

References

1. Gong, T. Cai, Z. X.: Mobile immune-robot model. In: Proceedings of IEEE International Conference on Robotics, Intelligent Systems and Signal Processing , 1091–1096 (2003).
2. Matthias, O.F., Hanspeter, A.M.: Biomimetic robot navigation. Robot. Auton. Syst. **30**, (1), 133–153 (2000)
3. Yu, J.Z., Wang, S., Tan, M.: Basic motion control of a free-swimming biomimetic robot fish. Proc. IEEE Conf. Decis. Control. **2**, 1268–1273 (2003)
4. Jerne, N.K.: Towards a network theory of the immune system. Ann. Immunol. (Paris). **125C**, 373(1974)
5. Deem, M.W., Lee, H.Y.: Sequence space localization in the immune system response to vaccination and disease. Phys. Rev. Lett. **91**, (6), 068101/1–4 (2003)
6. Balthrop, J., Forrest, S., Newman, M.E.J., et al.: Technological networks and the spread of computer viruses. Science, **304**, (5670), 527–529 (2004)
7. Dasgupta, D., González, F.: An immunity-based technique to characterize intrusions in computer networks. IEEE Trans. Evol. Comput. **6**, (3), 281–291 (2002)
8. Harmer, P.K., Williams, P.D., Gunsch, G.H., et al.: An artificial immune system architecture for computer security applications. IEEE Trans. Evol. Comput. **6**, (3), 252–280 (2002)
9. Branco, C.P.J., Mendes, V.R., Dente, J.A.: Using immunology principles for fault detection. IEEE Trans. Ind. Electron. **50**, (2), 362–373 (2003)
10. Luh, G.C., Cheng, W.C.: Identification of immune models for fault detection. Proc. Inst. Mech. Eng., Part I, J. Syst. Control Eng. **218**, (5), 353–367 (2004)
11. Gong, T., Cai, Z.X.: Parallel evolutionary computing and 3-tier load balance of remote mining robot. Trans. Nonferrous. Met. Soc. China. **13**, (4), 948–952 (2003)
12. de Castro, L.N., Timmis, J.: Artificial immune systems as a novel soft computing paradigm. Soft Comput. **7**, (8), 526–544 (2003)

13. de Castro, L.N., Timmis, J.: Artificial Immune Systems: A New Computational Intelligence Approach. Springer, London (2002)
14. Watkins, A., Timmis, J., Boggess, L.: Artificial immune recognition system (AIRS): an immune-inspired supervised learning algorithm. Genetic Programming and Evolvable Machines **5**, (3), 291–317 (2004)
15. Cai, Z.X., Peng, Z.H.: Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems. J. Intell. Robot. Syst. **33**, 61–71 (2002)
16. Gong, T., Cai, Z.X.: Anti-worm immunization of web system based on normal model and BP neural network. In: Wang, J., et al. (Eds.) ISNN 2006, LNCS **3973**, 267–272 (2006)
17. Madhusudan, B., Lockwood, J.W.: A hardware-accelerated system for real-time worm detection. IEEE Micro. **25**, (1), 60–69 (2005)
18. Verma, V., Gordon, G., Simmons, R., et al.: Real-time fault diagnosis robot fault diagnosis. IEEE Robot. Autom. Mag. **11**, (2), 56–66 (2004)
19. Huang, G.B., Saratchandran, P., Sundararajan, N.: A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans. Neural Netw. **16**, (1), 57–67 (2005)