ISBN 978-981-11-7861-0
Proceedings of 2018 the 8th International Workshop on Computer Science and Engineering (WCSE 2018)
Bangkok, 28-30 June, 2018, pp. 7-13
doi: 10.18178/wcse.2018.06.002

CPN Modeling and Dangerous State Identification of Aircraft Engine Control Software

Shunyao Kang, and Chenxin Wang [+]

College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029 China

Abstract. Aircraft engine control software is the core control software of the aircraft, and its safety is very important. How to construct the model of the software and effectively identify the dangerous state is also a difficult point to study the safety of the aircraft engine control software. This paper uses the colored Petri net (CPN) to model the aircraft engine control software, and then use the improved genetic algorithm to identify the dangerous state of the software. Finally, the method of this paper is applied to a real aircraft engine control software to implement CPN modeling and dangerous state identification. The results show that the method proposed in this paper can identify the dangerous state in the software. And through the improvement of the genetic algorithm, the execution efficiency of the recognition algorithm is improved.

Keywords: aircraft engine control software, colored petri nets, genetic algorithm.

1. Introduction

The aircraft engine full authority digital electronic control software (referred to as engine control software, FADEC) belongs to a typical aviation software. Once the engine control software fails, it is likely to cause catastrophic consequences, such as engine surge, parking in the sky, and flameout. Therefore, the engine control software has significant quality characteristics such as high safety index requirements and serious consequences of failure. [1]

For complex software such as aircraft engine control software, it is an effective way to form software model to describe various dynamic characteristics of software formally, and then realize the analysis and research of software. As a mathematic model with strict mathematics definition and good graphical presentation, the colored Petri net has the intuitiveness of its graphical expression and the technical characteristics of easy programming. It has become one of the main tools for the modeling of complex systems. [2]

In this paper, colored Petri net is used to model the engine control software of an aircraft. The dangerous state recognition is performed on the model, and the software state and corresponding parameter values that make it run to a dangerous state are given. At the same time, this work will identify the dangerous state. The search problem is transformed into model identification and parameters. In the implementation process, an improved genetic algorithm is used to guide the search and an effective identification of the dangerous state of the control software of an aircraft engine is achieved.

2. Related Work

2.1. Colored petri nets (CPN)

[+] Corresponding author. Tel.: 18811308122; fax: + 053284831057
E-mail address:744634128@qq.com.

Colored Petri nets (CPN) is an extension of classical Petri nets,which can effectively enhance the Petri net's ability to describe software and make it more suitable for modeling and analysis of large-scale complex software. In recent years, colored Petri nets have been more widely used.[3]

The formal definition of a colored Petri net is given below:

Definition 1 Colored Petri net [4]. Colored Petri net is a nine-tuple $CPN = (P, T, F, \sum, N, C, G, E, I)$

P is a finite set of place; T is a finite set of transition; F is a finite set of arcs and satisfies: $P \cap T = P \cap F = T \cap F = \phi$; $\sum$ represents a finite, non-null set of colors; N is a node function, $N: F \to (P \times T) \cup (T \times P)$; N is a color function, the color function associates each place with a color; G is a guard function, the guard function determines the trigger conditions for each transition; E is a arc expression function; I is a initialization function and satisfies: $\forall p \in P:[Type(I(p))=C(p)_{MS}]$.

The CPN model provides a more highly abstract Petri net model, which greatly improves the ability of the classical Petri net system to describe large complex systesms and also reduces the risk of state space explosion due to high system complexity. Therefore, the CPN model is more and more popularized in practical applications. [5, 6]

2.2. Genetic algorithm

In the 1960s, influenced by the evolutionary theory of "natural selection and survival of the fittest", Holland proposed the concept of genetic algorithm for the first time. The core idea is to obtain the optimal solution or approximate optimal solution of the problem to be solved through the evolution of individuals in the population. Genetic algorithm has been widely used in various fields because of its good usability and versatility [7].

As a relatively common search algorithm, genetic algorithm can be used in solving various problems. In order to solve different optimization problems, it is necessary to design a specific individual coding method, fitness function, genetic operation and evolutionary strategy according to the characteristics of the problem, in order to find a solution that satisfies the optimization problem. [8]

3. Research on CPN Modeling of Aircraft Engine Control Software

3.1. Analysis of the logic relationship between aircraft engine control software functions

An aircraft engine control software contains 11 function modules, and the start of function modules has a certain execution sequence. For example, the "Local Throttle" module can only be started after the "Ground Start" module is executed. In addition, the execution of the function module not only has the requirements of the execution order, but also the requirements of some preconditions. Only when the corresponding preconditions are satisfied can the execution of the subsequent modules be started. For example, the control software is initialized, then start the "Cold Run" function, it must also satisfy: "Cold running signal" is valid, "landing gear signal" is invalid, "start button" is valid, and the “PLA≤3°”.

The engine control software of a certain type of aircraft has many functions, and the logic relationship among various functional modules is more complicated. There may be cases of selection, sequence, and concurrent execution among functional modules.

(1) Selection execution of function modules

Function modules have multiple follow-up modules to choose from. For example, for the "Initialization" module, select "Cold Run", "False Drive", "Ground Start", "Communication Function" or "State Monitoring" module as its subsequent start up module, according to the precondition to determine which function to follow.

(2) Sequence execution of function modules

The functional module has only one follow-up module, and execution of one functional module is followed by execution of the next module. in a certain aircraft engine control software, the " Ground Start " function has only a " Stopping " function for subsequent functions. When the software is in the "false driving" function, if the precondition is satisfied, the software starts the "Stopping" function module.

(3) Concurrent execution of function modules

The function module has multiple follow-up modules to choose from, and their trigger conditions may be satisfied at the same time. These subsequent modules can be executed concurrently. For example, for the “Initialize” module, there are five functional modules “Cold Run”, “False Drive”, “Ground Start”, “Communication” and “Status Monitoring”. The precondition for the start of the “Cold Run” module is T1. Communication "The precondition for the start of the functional module is T4; T1 and T4 can be true at the same time, and the " Cold Run " function and the " Communication " function are started at the same time. At this time, the system executes the " Cold Run " function and the " Communication " function concurrently.

3.2. The construction of an aircraft engine control software CPN model

The CPN model of an aircraft engine control software is mainly divided into the following three steps:

(1) Analysis and Identification of the Corresponding Elements of the CPN Model of an Aircraft Engine Control Software.

According to the description of the software function in the outline design of a certain aircraft engine control software, it can identify a total of 11 functions, and can determine the logical relationship between the various functional modules. Analyzing the precondition of its function execution can identify its trigger conditions and the types of parameters involved. Analyze the external parameters needed for its function execution, determine the source of the corresponding external parameters, and establish external parameter libraries.

(2) Establishing the Mapping of Aircraft Engine Control Software and CPN Model.

According to the corresponding relationship between the established aircraft engine control software components and the CPN model, map the function modules of an aircraft engine control software, the execution sequence among the functions, the preconditions of the execution of the function modules, and the types of parameters to the CPN model.

(3) Create a CPN model

Using CPN Tools to build the CPN model of a certain aircraft engine control software，the established CPN model is shown in figure 1.

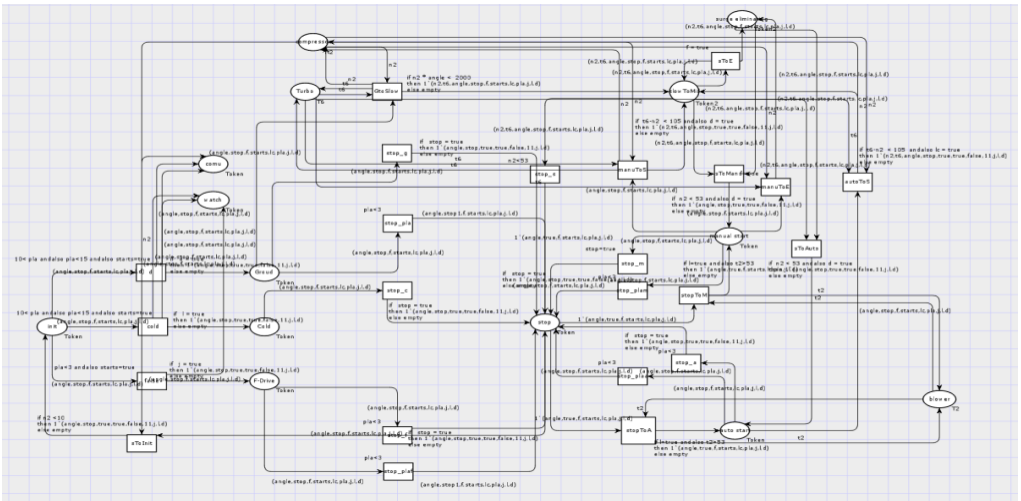

Fig. 1: The CPN model of an aircraft engine control software.

The CPN model of an aircraft engine control software contains 11 function place and 3 external parameter place. Transition represent the logical relationship between functions. A directed arc conditional expression represents a precondition for execution of a functional module. The weights on the directed arc represent some restrictions on the value of the module parameter, or the number of tokens flowing to the target library and the change of the parameter value. The token flow on the CPN model reflects the dynamic performance of the aircraft engine control software .

4. Dangerous State Identification of Aircraft Engine Control Software CPN Model Based on Genetic Algorithm

The purpose of the dangerous state identification is to verify whether there is a dangerous state in the software, that is, for a potentially dangerous state, whether a software state and a set of parameter combinations can be found to trigger this potentially dangerous state, resulting in a danger. Therefore, based on the genetic algorithm, this paper designs a recognition algorithm for the dangerous state of a certain aircraft engine software CPN.

4.1. Aircraft engine control software's potential dangerous state

There are two potential dangerous states in the Aircraft Engine Control Software, Their consequences and their representation in the CPN model are shown in the following Table I

TABLE I: POTENTIAL DANGEROUS STATE AND ITS CPN MAKING

No	Potential Dangerous State	making
T1	When the "Stopping" function and "eliminate" function of an aircraft engine control software are executed at the same time, the engine may be overheated or over-rotated, causing the engine control software of a certain type of aircraft to fail to operate normally.	{0,1,1,0,0,0,2,4,2,1,0,0,1,0}
T2	When an aircraft's engine control software's "auto air start" function and "elimination" function are executed at the same time, it may cause inconsistent fuel supply, resulting in failure of the aircraft's engine air start or engine in a surge state, causing the engine control software of a certain type of aircraft to fail to operate normally.	{0,1,1,0,0,0,4,3,2,1,0,0,0,1}

4.2. A genetic algorithm-based dangerous state recognition algorithm framework for aircraft engine control software CPN model

The algorithm framework is shown in Figure 2.

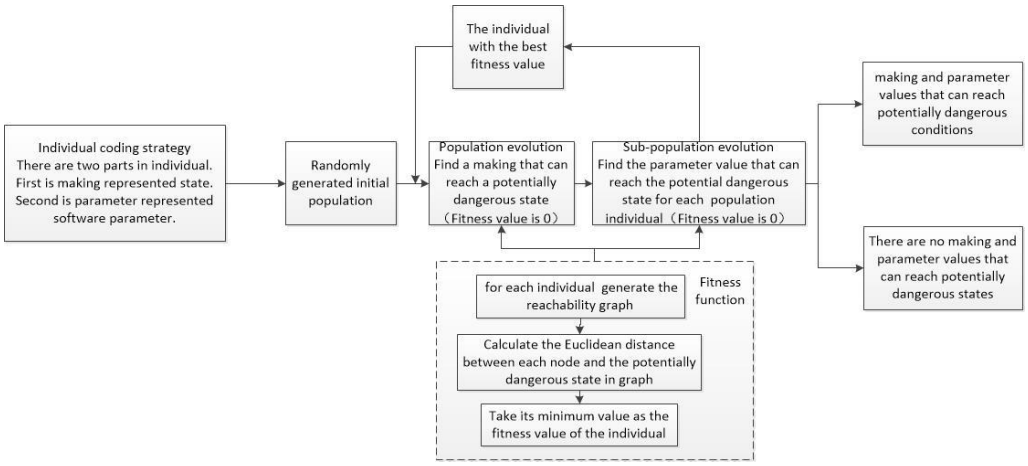

Fig. 2: Algorithm framework diagram.

4.3. Design of danger state recognition algorithm for CPN model of aircraft engine control software

1. Individual coding strategy

Assuming that the CPN model has n places, the individual's main population is an n-dimensional vector $M = \{m_1,m_2,....m_i....,m_n\}$, where m_i denotes the number of tokens that the corresponding place has.

Assuming that the total number of parameters on the token in the CPN software is s, then the individual sub-population part is a s-dimensional vector $V = \{v_1,v_2,....v_j....,v_s\}$ Where v_j represents the value of the corresponding parameter.

2. Population construction

For the individual coding strategy of the aircraft engine control software CPN model constructed in this paper, there are two types of information in the individual: token distribution information and software parameter information, and their influence degree on identifying dangerous states is different.So in this paper, the concept of main population and sub-population is designed. The construction of the population is aimed at the main population part and the sub-population corresponding part of each individual in the population,

that is, the distribution of the tokens in the CPN and the values of the carried parameters, which are generated according to the software requirements of the constructed model.

3. Fitness function

To calculate the distance between two state vectors,the following fitness function is adopted:

$$Fitness(ind, M) = \min\left(\text{euclidean\_ metric}\left(\text{R(ind)[j], M}\right)\right)\left(j = 1,2,...,30\right)$$

Assume that the potentially dangerous state in the software is M. The method which the individual Ind generates a state diagram that may trigger the state diagram is R( ). Since the subsequent state diagram may trigger a large generation of the state diagram, this paper only takes the first 30 states to perform the calculation. Euclidean metric() represents the method of calculating the Euclidean distance of two vectors. Min() represents the minimum method for taking these 30.

4. Population evolution termination conditions

If the traditional genetic algorithm is used to perform the iterative evolution inside the sub-population, if the state is not reachable, the sub-population will continue to perform internal iterations until it reaches the set maximum number of iterations before it can obtain a valid solution. This will result in a huge cost of algorithm execution.In order to improve the efficiency of the genetic algorithm, two optimization strategies are designed in this paper.

1) The traditional genetic algorithm uses the maximum number of iterations to terminate the iterative evolution, and increases the strategy of premature termination of the evolution when the individual fitness value does not increase continuously.

2) When the evolution of the sub-population is terminated, the optimal individuals generated during the individual evolution of the sub-population are used as the parameter part of the corresponding population for the selection process of the next-generation population.

5. Experimental Design and Analysis

5.1. Experimental purpose and experimental design

The experimental purpose of this chapter is only two: First, verify the effectiveness of this method, that is, whether it can identify the real dangerous state. The second is to analyze the efficiency of the recognition algorithm, that is, to analyze the influence of the improvement of the genetic algorithm on the efficiency.

(1) Verify the validity of the method. In this experiment, for the given two potentially dangerous states, the genetic algorithm was used 10 times to identify the dangerous state of the CPN model of an aircraft engine control software. Record the value of the generated token distribution and parameters, and verify whether the software status corresponding to these token distributions and parameter values is true by comparing the aircraft engine control software requirements documentation with the safety analyst. To verify if these two potentially dangerous conditions are indeed dangerous.

(2) Analyze the efficiency of the recognition algorithm. When the genetic algorithm is used to identify the dangerous state on the CPN model of an aircraft engine control software, two improved strategies are implemented for the genetic algorithm. This experiment analyzes the effect of algorithm improvement strategy on execution efficiency.

5.2. Validation of the identification of dangerous states

For the potentially dangerous states,the makings that use the dangerous state recognition algorithm to generate are shown in Table II:

TABLE II: CPN MAKING AND ITS MEANING IN AIRCRAFT ENGINE CONTROL SOFTWARE

No	CPN making	meaning in Aircraft Engine Control Software
t1	{1,0,0,0,0,0,4,3,2,0,0,0,0,0}	The software is in “initialization” function execution
t2	{0,1,1,0,0,0,4,4,2,0,0,1,0,0}	The software is in “manual air start” function execution

t1 is the software state that can reach T1,t2 is the software state that can reach T2,they are both real existence software state,so T1 and T2 are dangerous states.

5.3. The efficiency analysis of dangerous state recognition

In this paper, the basic genetic algorithm is improved by breaking out the sub-population advance strategy and retaining the optimal individual strategy of the sub-population. In order to analyze and improve the reduction of algorithm overhead, this paper designs the experiment based on the following two research problems, and analyzes these research problems based on the experimental results.

RQ1:Does the use of sub-populations breaking out strategy influence the efficiency of the algorithm?

RQ2:When the sub-population jumps ahead, does the strategy of retaining the optimal individual influence the execution efficiency of the algorithm?

First of all, aiming at research question 1, in order to answer the efficiency improvement of dangerous state recognition by genetic algorithm using sub-population break out early strategy, this paper conducted 20 experiments respectively for two potential dangerous states. The proportion of the sub-population breaking out early is shown in Figure 3.

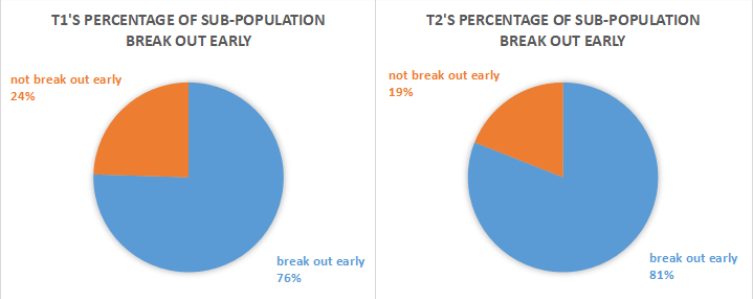

Fig. 3: The proportion of the sub-population breaking out early.

As can be seen from Figure 3, in the iterative process of sub-populations, there is a large part of the situation that is an iterative process that break out of the sub-population early. Therefore, if we consider that the sub-population will break out early, the time cost of the algorithm will be shortened.

The genetic algorithm and the genetic algorithm with sub-population breaking out early strategy were executed 20 times respectively. The results are shown in Table III.

TABLE III: AVERAGE NUMBER OF ITERATIONS AND AVERAGE OF RUNNING TIME

Potentially dangerous state	Average number of iterations		Average of running time (s)	
	GA	GA with sub-population breaking out early	GA	GA with sub-population breaking out early
T1	223.81	242.51	3331	1484
T2	192.38	211.23	2856	1125

The strategy of setting the sub-population breaking out early will increase the iteration number of the population slightly. The reason may be that the evolutionary process of the sub-population caused by the sub-population breaking out early is not enough. However, the running time of the entire algorithm has been greatly shortened. This is because many sub-populations will break out of the iteration of the sub-population once the fitness value does not change continuously and do not need to iterate until the set sub-population maximum iteration number.

For the research question 2, based on the sub-population breaking out early strategy , the strategy of keeping the current optimal individual of the sub-population when breaking out sub-population early was added. Take the current best individual as the representative of the sub-population and enter the selection process of the next generation of population. Similarly, the experiment was repeated 20 times. The number of iterations and time of the population is shown in Table IV.

When setting the sub-population breaking out early strategy and keeping the best individual when the sub-population breaks out early, it can be found that both the population iteration times and the time overhead are reduced. This is because the optimal individual that retains from the sub-population enters the

selection process of the next generation of population, and it can guarantee the support of the optimal population of sub-populations for population evolution.

TABLE IV: AVERAGE NUMBER OF ITERATIONS AND AVERAGE OF RUNNING TIME

Potentially dangerous state	Average number of iterations		Average of running time (s)	
	GA with sub-population breaking out early	GA with sub-population breaking out early and keeping optimal	GA with sub-population breaking out early	GA with sub-population breaking out early and keeping optimal
T1	242.51	101.34	1484	766
T2	211.23	102.51	1125	652

6. Conclusion

This paper uses a certain aircraft engine control software as an object to study how to use the colored Petri net (CPN) to model it. Based on this, using the improved genetic algorithm, the dangerous state recognition based on the established CPN model is performed. Through experiments, the effectiveness of the dangerous state recognition algorithm presented in this paper is verified and the efficiency of the algorithm is improved by the improvement of the genetic algorithm.

7. References

[1] Q. Wang, S.J.Wu, M.S.Li. ASoftware defect prediction technology[J]. Journal of Software, 2008(07):1565-1580.

[2] Sun Y, Zhang H. A software safety analysis method based on S-invariant of Petri Net[C]// International Conference on Reliability, Maintainability and Safety. IEEE, 2011:487 - 492.

[3] Jensen K, Kristensen L M. Colored Petri nets:a graphical language for formal modeling and validation of concurrent systems[J]. Communications of the Acm, 2015, 58(6):61-70.

[4] Jensen K. Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use. Volume 3: Practical Use[C]// IEEE Colloquium on Discrete Event Systems: A New Challenge for Intelligent Control Systems. Springer-Verlag New York, Inc. 1997:493-493.

[5] Maci àH, Valero V, D áz G, et al. Complex Event Processing Modeling by Prioritized Colored Petri Nets[J]. IEEE Access, 2016, 4(99):7425-7439.

[6] Song H, Liu J, Schnieder E. Validation, verification and evaluation of a Train to Train Distance Measurement System by means of Colored Petri Nets[J]. Reliability Engineering & System Safety, 2017, 164:10-23.

[7] Saitou K, Malpathak S, Qvam H. Robust design of flexible manufacturing systems using, colored Petri net and genetic algorithm[J]. Journal of Intelligent Manufacturing, 2002, 13(5):339-351.

[8] Zhao R, Harman M, Li Z. Empirical Study on the Efficiency of Search Based Test Generation for EFSM Models[C]// Third International Conference on Software Testing, Verification, and Validation Workshops. IEEE, 2010:222-231.