A Novel Method of Model-based Diagnosis by Propagating Failure Value

Dantong Ouyang[1,2], **Liming Zhang**[1,2], **Xiangfu Zhao**[1,2]

[1] *School of Computer Science and Technology, Jilin University, Changchun 130012, China*
[2] *Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China*

Summary
Model-based diagnosis is a new intelligent diagnostic technique which can overcome the shortcomings of traditional diagnostic methods. In this paper, a novel method of propagating failure value in model-based diagnosis is proposed, and the computing procedure is characterized by combining revised SE-tree (set enumeration tree) with closed nodes to produce all the diagnoses. It can directly compute all minimal diagnoses (MDs), without computing all the conflict sets and therefore the hitting sets of the collection of the corresponding conflict sets like the classical methods, and then the combinatorial explosion caused by calling ATMS, known as an NP-complete problem, can be avoided as well. As the closed nodes are added into the SE-tree, the non-minimal diagnoses can never be produced, and the true resolutions can not be missed by pruning either. The program is easy to be realized, and the diagnosis efficiency is highly improved by this method to satisfy real-time requirement, even for a complex system.

Key words:
model-based diagnosis; minimal diagnosis; set enumeration tree (SE-tree); failure value

1. Introduction

Model-based diagnosis is a new type of intelligent reasoning technique proposed for overcoming the serious drawbacks of traditional diagnostic methods [1]. And it is one of the active branches of Artificial Intelligence, playing both the role of test bed of several approaches to KR and reasoning and the role of generator of new problems and approaches. Conflict recognition, aiming at generating all minimal conflict sets, and candidate generation, aiming at generating all minimal hit sets, are of the two important steps towards to final diagnosis results. We have proposed an algorithm of combining SE-tree to generate all minimal conflict sets [2] and an algorithm of deriving all minimal conflict sets based on ATMS[3]. As it is an NP-hard problem [4], conflict set recognition must be avoided as possible.

How to derive all MDs? Chittaro[5] et al. proposed a hierarchical model which can represent multiple behavioral modes of one component in its various states. Baroni[6] et al. proposed a dynamic system model based on the finite-state automata. Console [7] et al. described the diagnostic problem based on the process algebra. Fattah and Dechter[8], Stumptner and Wotawa[9] respectively proposed a polynomial-time algorithm for diagnosing the tree-structured systems. Friedrich [10] also presented a polynomial-time approach for finding a minimal diagnosis. Based on the information of system structure, Luan and Dai [11] proposed a method of diagnosing a physical system without computing the conflict sets. Console [12] and Milde [13] proposed a method of diagnosing an actual system by generating a decision tree based on the system model. In order to deploy the real-world applications of model-based diagnosis, several relevant contributions were proposed in the literature (Refs [14-16]).

The main shortcomings of the above methods include: (1) Computing by above methods is quite complex or NP-complete problem. (2) Algorithm is polynomial time only in special conditions[8.9.11.12.13]. (3) The completeness of the results cannot be guaranteed [14.15.16].

In order to overcome the shortcomings mentioned above, we propose a novel method for computing all MDs. Furthermore, the computing procedure is formalized by combining revised SE-tree with closed nodes to produce all the resolutions gradually. The advantages of this method include: (1) It can directly compute all MDs, without computing all the conflict sets and therefore the hitting sets of the collection of the corresponding conflict sets. (2) As the closed nodes are added into the SE-tree, the non-minimal diagnoses can never be produced, and the true resolutions can not be missed by pruning either. (3) Though it is described by tree structure, it is not necessary to be implemented by using tree or graph structure, but only using simple link structure, so it is easily implemented. (4) All MDs obtained are minimal, not need to be reduced finally, and all MDs are bound to be obtained.

This paper is organized as follows. The failure value propagation diagnostic method is informally described in Section 2. An example is shown in Section 3. System

Manuscript received November 5, 2007
Manuscript revised November 20, 2007

implement is outlined in Section 4. Conclusions are drawn in Section 5.

2. FaultComps-tree method

2.1 FaultComps-tree method

The main idea of this method is as follows. A FaultComps-tree is a tree, where the sets of nodes generated are from shorter to longer, and it can enumerate all subsets of a given set. The pruned FaultComps-tree is generated depending upon whether the set is diagnosis results or not. Furthermore, all MDs are derived finally.

The set-enumeration tree (SE-tree) was proposed by Rymon[17]. A complete SE-tree can systemically enumerate all the elements of the power-set by a pre-imposed order, such as the alpha-beta order, numerical order, etc. For example, a full SE-tree of an ordered set S = {*A, B, C, D*} is shown in Fig.1.

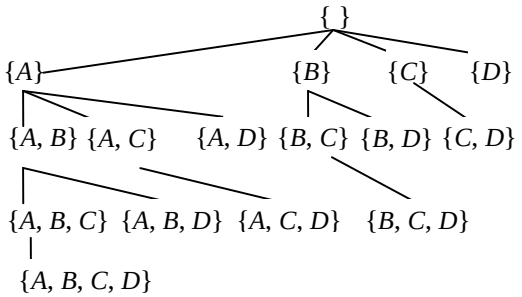


Fig.1. The SE-tree of the set {*A, B, C, D*}

Procedure 1. To derive all MDs of a system, we generate a pruned FaultComps -tree *T* in width-first by the following procedure.

1.for (every set *faus* from right to left at the same level.)
2. for (every set *tems* generated which is marked with " √ " at *faus*'s right hand.)
3. if (*faus* ⊇ *tems*)
4. mark set *faus* with "×", not to be extended it latter;
5. break;
6. end
7. end
8. if (*faus* is diagnosis result)
9. mark the set *faus* with " √ ", not to be extended it latter.;
10. end
11.end

Finally, all MDs of the system are the sets which are marked with " √ " in the pruned FaultComps-tree *T*. Let's show the soundness and the completeness of this algorithm.

(1) Soundness: We mainly show the soundness of the two pruning rules as follows. On one hand, if the label set of a node is a diagnosis set (DS), then all of its children nodes must be proper supersets of it. Therefore, it is marked with " √ ", not to be extended latter. On the other hand, if the label set *A* is not a DS, then for each DS node *B* generated which is marked with " √ " at *A*'s right hand, if $A \supseteq B$, then *A* is a proper superset of *B*, and *A* need not be further extended. Therefore, it avoids the generation of the proper superset of MDs. Otherwise, we will judge whether the set *A* is a DS or not.

(2) Completeness: As an SE-tree can enumerate all the elements of the power-set by a pre-imposed order, so all the possible sets, each of which is not a proper superset of some MDs can be generated. All the MDs will be generated in the end.

(3) The complexity of the algorithm is $O(2^k)$ in the worst situation, where *k* is the number of the system components. However, many nodes have been pruned by pruning rules, and hence the complexity is much less than $O(2^k)$ in general.

A method of using failure behavior to identify whether *faus* is DS or not is given in the next subsection.

2.2 Propagating failure value method

If the component set is a DS, the output value of the component in this set will be a failure value. So we will use a variable to express the output value of the component. Then we compute all the output value of components which have relation to the corresponding variable (based on ATMS in special directions). Therefore, we have some equations about all the variables.

We use the linked list of *obsval* to store the relevant information of parameters of the observation, the linked list of *sysval* to store the relevant information of parameters of non-fault systems. Let the linked list of *tempobsval* be a copy of *sysval*, to be used for modifying the relevant information of the component, which is one element of *faus* set before being computed. The *faus* set is a DS when the equations have a root (it needs to check whether the equations have a root or not.) and *tempobsval* is consistent with *obsval* (Note: *tempobsval* is consistent with *obsval* when every node's value of *obsval* is equal to the corresponding node's value of *tempobsval*).

To check whether the set *faus* is a DS or not, we have the following theorems clearly.

Theorem 1: If *faus* is empty, then *faus* is a DS when *obsval* is consistent with *tempobsval*.

Theorem 2: If *faus* is empty, then *faus* is not a DS when *obsval* is inconsistent with *tempobsval*.

Theorem 3: If *faus* is not empty, then *faus* is not a DS when the equations (derived from the above method) have a root and *obsval* is inconsistent with *tempobsval*.

Theorem 4: If *faus* is not empty, then *faus* is a DS when the equations (derived from the previous method) have a root and *obsval* is consistent with *tempobsval*.

Theorem 5: If *faus* is not empty, then *faus* is not a DS

when the equations (derived form the above method) do not have a root.

3. An Example

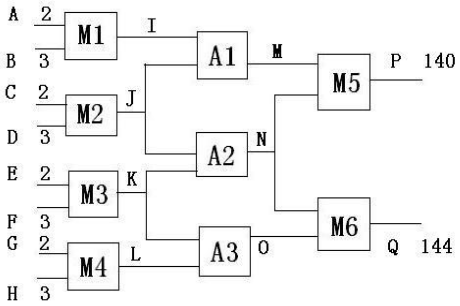


Fig.2 poly-box system

One of the poly-box systems, depicted in Fig.2, contains 9 components, where M1, M2, M3, M4, M5 and M6 are multipliers, A1, A2 and A3 are adders, respectively. The input and output of the system are depicted in Fig.2. The pruned FaultComps-tree *T* is shown in Fig.3.

The procedure of generating a pruned FaultComps -tree *T* is as follows.

Set {M6}: Let the output variable of M6 be *x*, then *x* = 144. Though the equations have a root, *tempobsval* is inconsistent with *obsval*. Therefore, {M6} is not a DS.

Set {M5}: Let the output variable of M5 be *x*, then *x* = 140. The equations have a root and *tempobsval* is consistent with *obsval*. Therefore, {M5} is a DS.

Set {M3}: Let the output variable of M3 be *x*, then *x**12 = 144. Though the equations have a root, *tempobsval* is inconsistent with *obsval*. Therefore, {M3} is not a DS.

Set {A2}: Let the output variable of A2 be *x*, then *x**12 = 144, *x**12 = 140. The equations have not a root, {A2} is not a DS.

Set {A1}: Let the output variable of A1 be *x*, then *x**12 = 144. The equations have a root and *tempobsval* is inconsistent with *obsval*. Therefore, {A1} is a DS.

Set {M4}: Let the output variable of M4 be *x*, then (*x*+6)*12 = 144. Though the equations have a root, *tempobsval* is inconsistent with *obsval*. Therefore, {M4} is not a DS.

Set {M3}: Let the output variable of M6 be *x*, then (*x*+6)*(*x*+6) = 144, (*x*+6)*12 = 140. The equations have not a root, {M3} is not a DS.

Set {M2}: Let the output variable of M6 be *x*, then (*x*+6)*(*x*+6) = 140, (*x*+6)*12 = 144. The equations have not a root, {M2} is not a DS.

Set {M1}: Let the output variable of M1 be *x*, then *x* = 144. The equations have a root and *tempobsval* is inconsistent with *obsval*. Therefore, {M1} is a DS.

Set {M2, M6}: Let the output variable of M2 be *x* and the output variable of M6 be *y*, then *y* = 144, (*x*+6)*(*x*+6) = 140. The equations have a root and *tempobsval* is consistent with *obsval*. Therefore, {M2, M6} is a DS

And so on.

Finally, a FaultComps-tree T containing 25 nodes is derived, in which nodes marked with “ √ ” or “ × ” need not to be extended. Then all the MDs are obtained from the nodes marked with “ √ ”

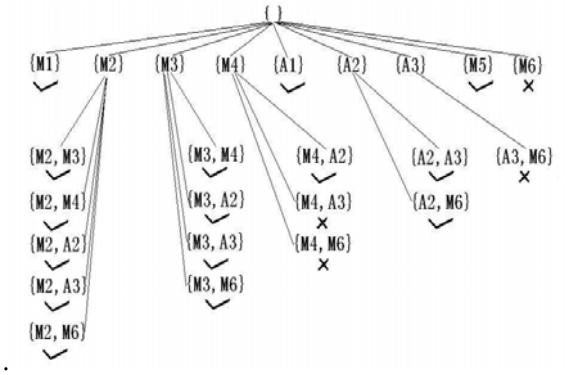


Fig.3 FaultComps-tree *T*

4. System implement

We have implemented a program in Visual C++6.0 (AMD Athlon(tm) 64 X2 Dual Core Processor 3600+ , 1.90 GHz, 1GRAM, Windows XP), and increased the scale of poly-box system. The GUI of the program is depicted in Fig. 4, and the result is shown in table 1.

From table 1, we can see clearly that: (1) The time of computing all MDs is very low. (2) As increasing the scale of the poly-box system, the computing time has increased smoothly. While in Reiter’s method, the result cannot be derived about 20 components in a poly-box system. (3) The number of the nodes in FaultComps-tree *T* is not exponentially increasing as the increasing of the scale of poly-box system, with very few nodes irrelevant to DS being generated.

Table 1 the all MDs of poly-box and running time.

Components of system	Nodes of FaultComps-tree	Numbers of MDs	Running time(ms)
5	8	4	0.222
9	25	15	1.271
14	63	43	5.476
20	135	100	20.635

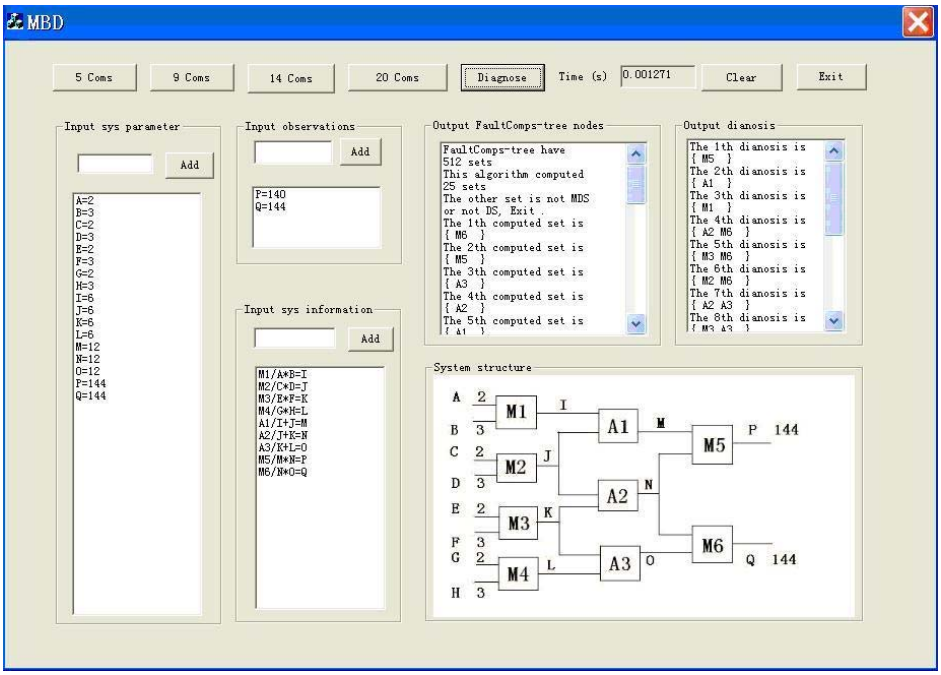


Fig.4 The GUI of 9 components poly-box

5. Conclusions

Reiter proposed a method of finding diagnosis by computing the conflict sets, whose complexity is exponential time cost. Davis [18], de Kleer and Williams [19] proposed constrain propagation method to compute the conflict sets. Unlike their methods, our approach can find all MDs without computing the conflict sets. The method is easy to understand and implement. Although the computing procedure is formalized by a tree, it is implemented with the linked list structure. Our approach allows attention to be focused on the variable propagation in a special direction of the components in the system, avoiding the combinatorial explosion caused by calling ATMS, known as an NP-complete problem. The number of the nodes in FaultComps-tree *T* which is irrelevant with a DS is very few. As the closed nodes are added into the SE-tree, the non-minimal diagnoses can never be produced, and the true resolutions can not be missed by pruning, either.

Acknowledgement

This paper is supported by NSFC Major Research Program 60496321, Basic Theory and Core Techniques of Non Canonical Knowledge; Program for New Century Excellent Talents in University; the Chinese National High Technology Research and Development Plan (Grant No. 2003AA118020); and Jilin Province Science and Technology Development Plan (Grant No. 20060532).

References

[1] Reiter R. A theory of diagnosis from first principles [J]. Artificial Intelligence, 32 (1) (1987): 57-96.

[2] Zhao X.F., Ouyang D.T. New methods for deriving all minimal conflict sets in model-based diagnosis [J]. Journal of Jilin University (Engineering and Technology Edition), 37(2) (2007): 413-418.

[3] Zhang L.M., Ouyang D.T., Zhao X.F. A new method for deriving all minimal conflict sets based ATMS [J]. Computer Engineering and Science (accepted).

[4] Garey M. R., Johnson D. S. Computers and intractability: a guide to the theory of NP-completeness [M]. Murray Hill, 1979, NJ: Bell Labs.

[5] Chittaro L., Ranon R. Hierarchical model-based diagnosis based on structural abstraction [J]. Artificial Intelligence, 155 (1-2) (2004): 147-182.

[6] Baroni P., Lamperti G., Pogliano P., Zanella M. Diagnosis of large active systems [J]. Artificial Intelligence, 110 (1999): 135-183.

[7] Console L., Picardi C, Ribaudo M. Process algebras for systems diagnosis [J]. Artificial Intelligence, 142 (2002): 19–51.

[8] Fattah Y. E., Dechter R, Diagnosis tree-decomposable circuits [C]. Proceedings of International Joint Conference on Artificial Intelligence, Montreal, Canada, 1995, 572-578.

[9] Stumptner M., Wotawa F. Diagnosis tree-structured systems [J]. Artificial Intelligence, 127 (1) (2001): 1-29.

[10] Friedrich G., Gottlob G., Nejdl, W. Physical impossibility instead of fault models [C]. AAAI, (1990): 331-336.

[11] Luan S.M., Dai G.Z. An approach to diagnosing a system with structure information [J]. Chinese Journal of Computers, 28(5) (2005): 801-808.

[12] Console L. Temporal decision trees: model-based diagnosis of dynamic systems on- board [J]. Journal of Artificial Intelligence Research, 19 (2003): 469-512.

[13] Milde H. Integrating model-based diagnosis techniques into current work processes- three case studies from the INDIA project [J]. AI Communications, 13 (2000): 99-123.

[14] Mozetic I. A polynomial-time algorithm for model-based diagnosis [C]. Proceeding of the 10th European Conference on Artificial Intelligence, Vienna, (1992): 729-733.

[15] Childress R. L., Valtorta M. Polynomial-time model-based diagnosis with the critical set algorithm [C]. Proceeding of the 4th International Workshop on Principles of Diagnosis, Aberystwyth, Wales, (1993): 166-177.

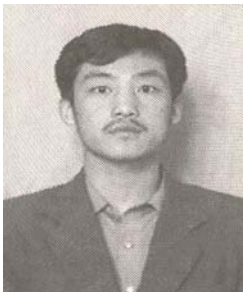
[16] Haenni R. Aquery-driven anytime algorithm for argumentative and abductive reasoning [C]. Proceedings of the 1st International Conference on Software. Belfast, (2002): 114-127.

[17] R. Rymon. Search through systematic set enumeration [C]. In: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, 1992, 539–550.

[18] Davis R. Diagnostic reasoning based on structure and behavior [J]. Artificial Intelligence, 24 (1984) : 347-410.

[19] de Kleer J., Williams B. C. Diagnosing multiple faults[J]. Artificial Intelligence, 32 (1987) : 97-130.

Dantong Ouyang received the M.S. degree in Computer Science from Jilin University in 1993, and received the Ph.D. degree in Computer Science from Jilin University in 1998. Since 2001, she has been a professor in the school of computer science & technology of Jilin University, and now she is also a Ph.D. supervisor. Her research interests include artificial intelligence, model-based diagnosis, and automated reasoning.



Liming Zhang is an M.S. student in the school of computer science & technology of Jilin University. His research interest is model-based diagnosis.

Xiangfu Zhao received the M.S.degree in Computer Science from Jilin University in 2006, and now is a Ph.D. student in the school of computer science & technology of Jilin University. His research interest is model-based diagnosis.