# A Multi-Agent Model for Information Processing in Computational Problem Solving

M. A. Teh Noranis and N. Shahrin Azuan

*Abstract*—**Problem solving is vital in the computer programming course. It is the earliest topic that is emphasized and more time is allocated to teach the topic. Problem solving requires the problem understanding knowledge that novice students usually lacks. In order to assist novice students in computational problem solving, a multi-agent model is designed. The proposed model is different from existing model in terms of the unique architecture that utilizes agents for information processing, specifically to extract, transform and generate information. Five agents are designed for this purpose namely the GUI, PAC, IPO, Flowchart and Algorithm agents. The model is tested with three different kinds of problem statement and produced correct results.**

*Index Terms*—**Computational problem solving, information processing, multi-agent, problem understanding.**

## I. Introduction

Problem solving is essential not only in the computer science field, but also in other fields such as medical [1], engineering [2] and mathematics [3]. Problem understanding is very important before execution of any procedure.

The problem solving topic is taught in most of the courses that are offered in the universities. In the fundamental of computer programming course, problem solving is a topic that is given emphasis and more time allocation before proceeding with the technical topic. However, students especially novices face difficulties in problem understanding [4]-[6]. Therefore, the implementation of a procedure can go wrong due to lack of knowledge in problem solving, for example, the misunderstanding of novices in computational problem solving, results in unexpected output due to wrong concept application.

There should be a guide for students to assist them in problem solving. It can be an intelligent system that can sense the environment, process the input, give recommendation to the students on how to solve the problem and perform a sort of communication to send related information. These are the features of a multi-agent system that exhibits the autonomous, reactive, proactive and social ability features. Agents have been widely applied in various applications such as interactive tutoring [7], medical diagnosis [8] and image analysis [9].

The motivation of this research work is to support novice difficulties in computational problem solving by utilizing the multi-agent technology.

This paper is organized into seven sections. Section two explains about the works that are related to our research. Section three presents our proposed model and section four describes our proposed algorithm to extract, transform and generate information. The experimental results are explained in section five where our proposed model is tested with three different problem statements. Section six discusses about the results findings and finally the last section which is section seven is the conclusion and future works.

Manuscript received June 11, 2013; revised August 23, 2013.

This research is supported by the Kementerian Pengajian Tinggi (KPT) Malaysia under the grant of Institut Pengajian Tinggi Awam (IPTA), Fundamental Research Grant Scheme (FRGS), project code 02-12-10-1000FR.

M. A. Teh Noranis is with the Department of Computer Science, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor, MALAYSIA (e-mail: nuranis@fsktm.upm.edu.my).

N. Shahrin Azuan is with the Telekom Malaysia Research and Development, Lingkaran Teknokrat Timur, 63000 Cyberjaya, Selangor, MALAYSIA (e-mail: shahrin@tmrnd.com.my).

## II. Related Research

### A. Agent-Based Computational Problem Solving Models

Many existing multi-agent computational problem solving models are based on visualization models. These visualization models visualize pseudo-code language [10], expression in C programming [11] and object parameters [12]. Social agents are used to support the understanding of program visualization through discussion with the user and explanation from the agent [10], [11] adapt the level of details in the visualizations based on the learner's knowledge and focus on topic that are not understood by the user. An approach to change the visualization parameters for example shape, colors and style is used to help novices in programming [12]. Jeliot 3 [13] integrates multi-agents namely Student Agent, Record Agent, Modeling Agent, Learning Object Agent and Evaluation Agent to provide dynamic and adaptive learning materials to individual users.

### B. Information Extraction and Retrieval

Several research works on multi-agent technique that perform information extraction have been implemented. Authors have proposed Wisconsin Adaptive Web Assistant (WAWA) that used the reinforcement learning technique and created a home-page finder agent and a seminar-announcement extractor agent for retrieving and extracting information [14], [15] proposed a keyword-based patent map using the text mining technique for information extraction, visualization and analysis in the web. An information agent shell consisting of four main components namely the OntoCrawler, OntoExtractor, OntoClassifier and OntoRecommender is designed for information searching, information extracting, information classifying and information representing/ranking applying Protégé and relevant Application Programming Interface (API) to

construct the domain ontology and its related ontology services to support all of the ubiquitous services of the shell [16], [17] suggested using a text mining techniques for biomedical text on cancer to extract novel knowledge from scientific text. Information retrieval and recommender system techniques are applied to access music information services that support music navigation discovery, sharing and formation of user communities [18].

### C. Discussion

Conventional computational problem solving techniques consists of Problem Analysis Chart (PAC), Input Process Output (IPO) chart, flowchart, algorithm/pseudocode, structure chart, Nassi-Shneiderman chart and Warnier-Orr diagram. Existing multi-agent computational problem solving visualization models applied these techniques in their models as mentioned. However, existing models are different from what we proposed because our model provide a step-by-step way from the very beginning to understand the problem statement, converts the problem statement to PAC, PAC to IPO, IPO to flowchart and flowchart to algorithm where multi-agents are used as knowledge representation. This way provides extraction from one problem solving technique and transformation to another problem solving technique. In addition, module number is generated to show the sequence of processing. Multi-agents communicate by passing information, giving a clear understanding of the problem to be solved.

As mentioned, information extraction and retrieval have been applied in web-based system, ubiquitous services, biomedical system and music application. Information extraction related to our multi-agent computational problem solving model is novel. Moreover, we combine the transformation and generate module number implemented by multi-agents. The next section will explain in detail about our proposed model.

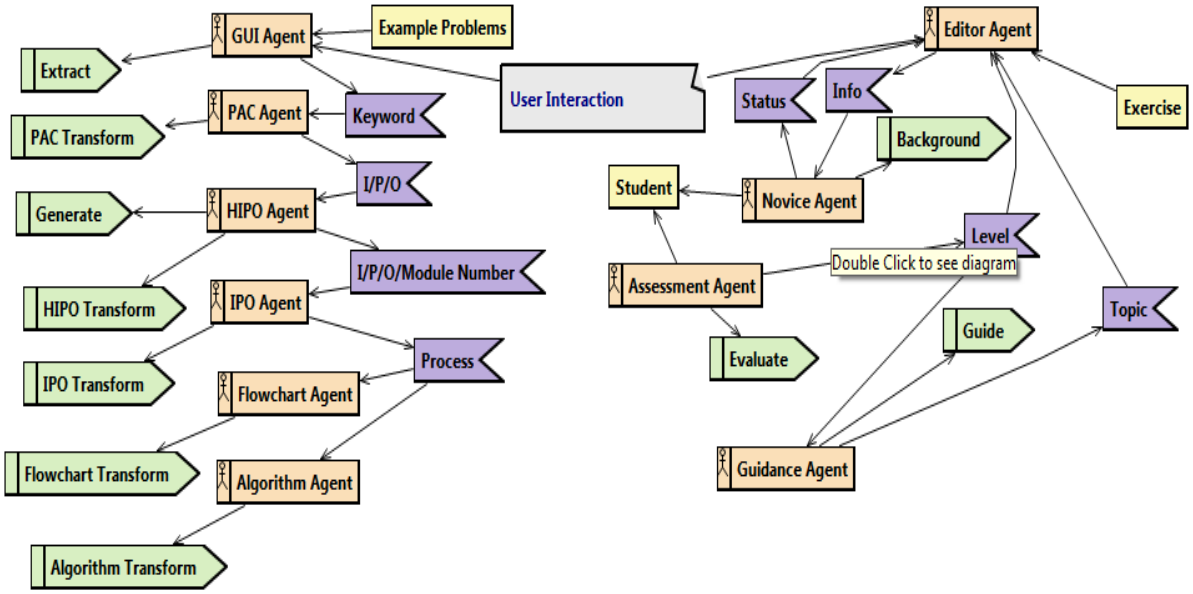


Fig. 1. Proposed multi-agent computational problem solving model.

## III. Proposed Model

Our multi-agent computational problem solving model is very different from other proposed research works in terms of the unique architecture and algorithm. In our work, we designed the Problem Solving Comprehension Module and the Assignment Module as shown in Fig. 1. The left part of the dotted lines is the Problem Solving Comprehension Module and the right side of the dotted lines is the Assignment Module. These agents are modeled using the Prometheus notation [19], [20].

The Problem Solving Comprehension Module consist of five agents namely GUI, PAC, IPO, Flowchart and Algorithm agents which are responsible to extract and transform information. Additionally, to produce the IPO, the IPO agent role is to generate module number that shows sequential processing for the problem statement. Each of the agents represent the techniques of computational problem solving that is related to each other. For example, to produce an IPO chart, information needs to be extracted from the PAC and module number needs to be generated so that information can be transformed to the IPO chart.

The Assignment Module consists of four agents namely Editor, Novice, Assessment and Guidance agents which are responsible to interact with the user. This paper focus on the Problem Solving Comprehension Module and the Assignment Module will be designed in detail for future works.

The agent class diagram is presented in Fig. 2. The Problem Solving Agent GUI extends the AgArch class and communicates with the gui Agent, pac Agent, ipo Agent, flow chart Agent and algorithm Agent. The gui Agent reads the problems given in text form and acts as an interface with the user. This paper focuses on the design of subagent, packaging and ipoAgent. The flowchart Agent and algorithm Agent design is still under construction. The

ProblemSolvingAgentGUI constructor displays the interface containing a button that is used by the novice to start the problem understanding process. The act method executes an action by the agent and the stop method is called by the infrastructure tier when the agent is about to be killed. The agents are programmed using Jason Agent Speak interpreter. Our proposed algorithm is integrated in the process method which will be explained in the next section.

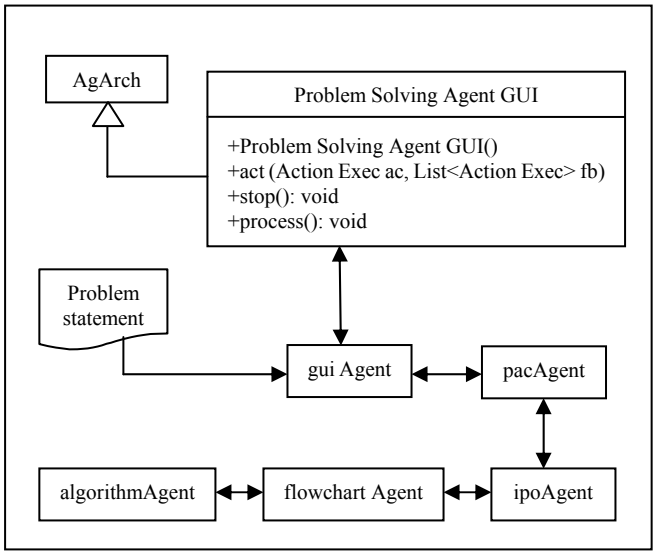


Fig. 2. Multi-agent class diagram.

## IV. The Process Method

Our proposed algorithm is presented in Fig. 3. The algorithm is extended from our previous work [21] which is integrated inside the process method in the ProblemSolvingAgentGUI class as in Fig. 2.

The document contains problems in text form. Currently, we cater for simple sequential problems. We intend to extend the problems using looping or iterations in our future works. The problems in text form needs to be read by tokens and stored in array a, that controls the loop. "KeyA" – "KeyF" in Fig. 3 are important keyword representing the process, input and output in the problem. Words that are extracted from the problems based on "KeyA" – "KeyF" by the pacAgent to be transformed to the PAC. Information that is stored in arrays b and c are used later by the ipoAgent that involves the extraction, transformation to IPO and generate module number processes.

The Problem Solving Comprehension Module consist of five agents namely GUI, PAC, IPO, Flowchart and Algorithm agents which are responsible to extract and transform information. Additionally, to produce the IPO, the IPO agent role is to generate module number that shows sequential processing for the problem statement. Each of the agents represent the techniques of computational problem solving that is related to each other. For example, to produce an IPO chart, information needs to be extracted from the PAC and module number needs to be generated so that information can be transformed to the IPO chart.

process( ) algorithm
1. read the problem token by token and store in array a
2. initialize j to 0
3. extract PAC information:
  3.1 loop (j < array length a)
    3.1.1 if (a[j] = = "KeyA" to "KeyF")
      3.1.1.1 extract WordA to WordF and transform to PAC
    3.1.1.2 store wordA to wordF in array b to be used later by IPO
      3.1.1.3 capture related word in PAC in array c to be used by IPO
    3.1.2 j++
  3.2 end loop
4. extract IPO information:
  4.1 initialize variable j, k, l to be used in loop
  4.2 loop (k < array length b && k < array length c)
    4.2.1 if (b[j] = = null) go out of the loop
    4.2.2 extract data, generate module number and transform to IPO
  4.3 j++, k++, l++
  4.4 end loop

Fig. 3. Multi-agent class diagram.

## V. Experimental Results

The proposed model is tested with three different problem statements which are in text form.

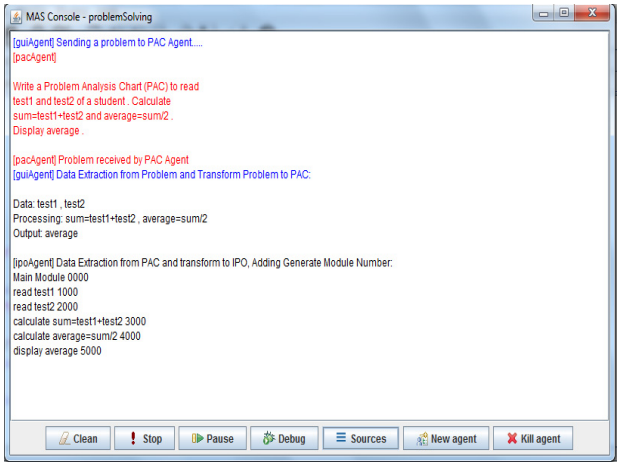


Fig. 4. Problem statement one results.

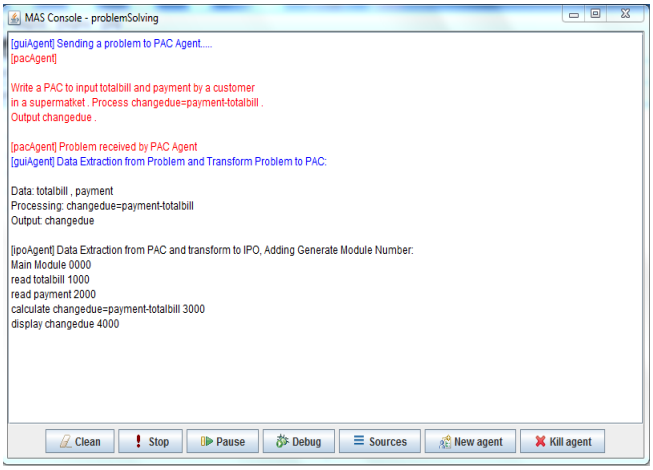


Fig. 5. Problem statement two results.

### A. Problem Statement One

The first problem is as follows:

Write a Problem Analysis Chart (PAC) to read test1 and test2 of a student. Calculate sum=test1+test2 and average=sum/2. Display average.

The results are shown below in Fig. 4.

### B. Problem Statement Two

The second problem is as follows:

Write a Problem Analysis Chart (PAC) to get length and width of a rectangle. Compute area=length*width. Print area.

The results are shown below in Fig. 5.

### C. Problem Statement Three

The third problem is as follows:

Write a Problem Analysis Chart (PAC) to input totalbill and payment by customers in a supermarket. Process changedue=payment-totalbill. Output changedue.

The results are shown below in Fig. 6.

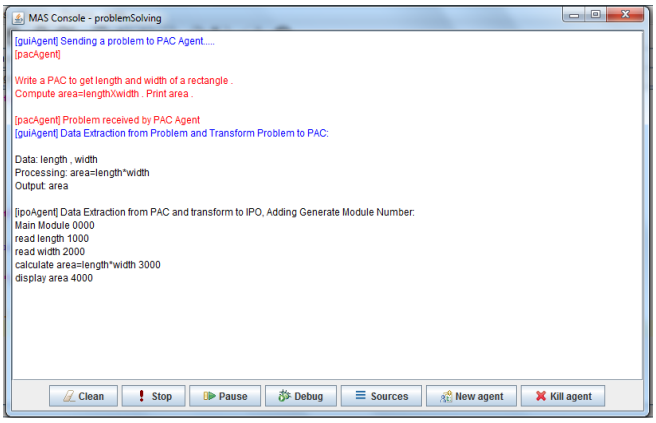


Fig. 6. Problem statement three results.

### D. User Interface

Fig. 7 shows the user interface controlled by the novice to start understanding the problem statements.

## VI. Result Findings

Our algorithm that has been improved from our previous work [21] used samples of sequential problem and succeeded to produce accurate results as expected. The input, process and output are extracted and transformed correctly by the PAC agent following the correct sequence by the IPO agent and the module number generated accordingly. The results also illustrate communication between guiAgent, pacAgent and ipoAgent that demonstrate the visualization of knowledge representation by these agents.

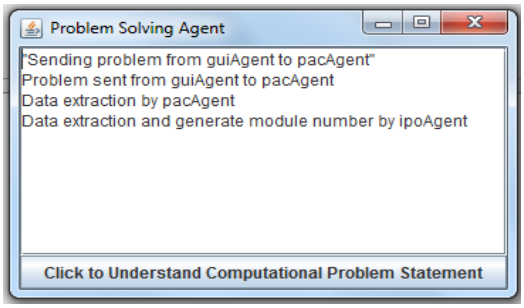


Fig. 7. Problem understanding user interface.

In our research work, we are using the existing conventional problem solving techniques consisting of PAC, IPO, flowchart and algorithm techniques. Our contribution is mainly on the new model that is different from other visualization problem solving model which starts from the very beginning of the problem statement that shows an exact flow of information from one agent to another which gives a clear explanation of the problem statement.

We are currently in the process of extending and improving the design of the Problem Solving Comprehension Module for the flowchart and algorithm agents and coding of the algorithm using Jason AgentSpeak [22], an agent programming language which is built on top of the Java programming language.

Our model represent agents role as problem solvers. It is expected that our agent knowledge representation model can assist novices in computational problem solving, as problem solving is very important not only in the computer science field but also other fields such as medical, engineering and business. Logically, our algorithm can be extended to be applied in other fields besides the computer science field. Moreover, it is expected that this model can enhance the students' computational problem solving skills.

## VII. Conclusion and Future Works

A model based on agents has been constructed for computational problem solving. The Problem Solving Comprehension Module function is for extraction, transformation and module number generation. The Assignment Module is for interaction with the user. The agent in the Problem Solving Comprehension Module consists of GUI, PAC, IPO, flowchart and algorithm agents. The agent in the Assignment Module consists of the Editor, Novice, Assessment and Guidance agents. The proposed model is evaluated using samples of sequential problem statement and produced correct extraction and transformation results. In addition, the proposed model illustrates visualization, communication and knowledge representation features by software agents.

For future works, the detailed design and coding of the Problem Solving Comprehension Module and the Assignment Module will be implemented. In addition, the proposed algorithm will be extended to provide more keywords to cover wide problem statements consisting of looping problems and more advanced object-oriented problems like inheritance, polymorphism and encapsulation. Furthermore, we plan to integrate our agent-based algorithm in a computer programming development environment tool.

## Acknowledgment

I would like to thank Kementerian Pengajian Tinggi (KPT) Malaysia for sponsoring this research work under the grant of Institut Pengajian Tinggi Awam (IPTA), Fundamental Research Grant Scheme (FRGS), and project code: 02-12-10-1000FR.

## References

[1] H. C. Jung and C. Y. Ta, "Supporting the development of collaborative problem-based learning environments with an intelligent diagnosis tool," *Expert Systems with Applications*, vol. 35, pp. 622-631, 2008.

[2] A. Nouy, "Generalized spectral decomposition for solving stochastic finite element equations: Invariant subspace problem and dedicated algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 197, pp. 4718-4736, October 2008.

[3] T. Muir, K. Beswick, and J. Willamson, "I'm not very good at solving problems: An exploration of students' problem solving behaviours," *The Journal of Mathematical Behavior*, vol. 27, pp. 228-241, 2008.
[4] P. Guo and H. Qi, "Study on Interactive System for Teaching Programming (ISTP) based on intelligent agent," in *Proc. 2010 6th International Conference on Wireless Communications WiCOM 2010*, Shenzhen, China, 2010, pp. 23-25.
[5] K. E. Boyer, W. Lahti, R. Phillips, M. D. Wallis, M. A. Vouk, and J. C. Lester, "Principles of asking efective questions during student problem solving," in *Proc. SIGCSE'10 - the 41st ACM Technical Symposium on adaptive Computer Science Education*, 2010, pp. 460-464.
[6] W. Pedrycz and P. Rai, "A multifaceted perspective at data analysis: A study in collaborative intelligent agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 38, pp. 1062-1072, 2008.
[7] S. Yaskawa and A. Sakata, "The application of intelligent agent technology to simulation," *Mathematical and Computer Modeling*, vol. 37, pp. 1083-1092, 2003.
[8] B. L. Iantovics, "Agent-based medical diagnosis systems," *Computing and Informatics*, vol. 27, pp. 593-625, 2008.
[9] D. A. Bell, A. Beck, P. Miller, Q. X. Wu and A. Herrera, "Video mining –learning patterns of behaviour via an intelligent image analysis system," in *Proc. Rio de Janeiro, Brazil,* 2007, pp. 460-464.
[10] R. Miraftabi, "Intelligent agents in program visualizations: A case study with seal," in *Proc. of the First International Program* Visualization *Workshop*, 2001, pp.53-58.
[11] P. Brusilovsky and H. D. Su, "Adaptive visualization component of a distributed web-based adaptive educational system," in *Proc. 6th* International *Conference on Intelligent Tutoring Systems (ITS'2002* , London, vol. 2363, 2002, pp. 229-238.
[12] M. Lattu, J. Tarhio, and V. Meisalo, "How a visualization tool can be used – Evaluating a tool in a research & development project," in *Proc. the 12th Psychology of Programming Interest Group (PPIG) Workshop*, 2000.
[13] M. B. Ari, R. Bednarik, R. B. B. Levy, G. Ebel, A. Moreno, N. Myller, and E. Sutinen, "A decade of research and development on program animation: The Jeliot experience," *Journal of Visual Languages and Computing*, vol. 22, pp. 375-384, 2011
[14] T. Eliassi-Rad and J. Shavlik,"A system for building intelligent agents that learn to retrieve and extract information," *User Modeling and User-Adapted Interaction*, vol. 13, pp. 35-88, 2003.
[15] S. Leea, B. Yoonb, and Y. Parkc, "An approach to discovering new technology opportunities: Keyword-based patent map approach," *Technovation*, vol. 29, pp. 81–497, 2009.
[16] Y. S. Yuan, "OntoIAS: An ontology-supported information agent shell for ubiquitous services," *Expert Systems with Applications*, vol. 38, pp. 7803–7816, 2011.
[17] F. Zhu, P. Patumcharoenpol, C. Zhang, Y. Yang, J. Chan, A. Meechai, W. Vongsangnak, and B. Shen, "Biomedical text mining and its applications in cancer research," *Journal of Biomedical Informatics*, vol. 46, pp. 200–211, 2003.
[18] M. Kaminskas and F. Ricci, "Contextual music information retrieval and recommendation: State of the art and challenges," *Computer Science Review*, vol. 6, pp. 89-119, 2012.
[19] L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*, Chichester, U.K.: Wiley, 2004.
[20] M. Woorldridge, *An Introduction to MultiAgent Systems*, U.K.: Wiley, 2009.
[21] T. N. M. Aris, "Object-oriented programming semantics representation utilizing agents," *Journal of Theoretical and Applied Information Technology*, vol. 31, pp. 10-20, 2011.
[22] M. Gustave. (1865). Jason A Java-based Interpreter for an Extended Version of Agent Speak. [Online]. Available: http://www. jason.sourceforge.net/Jason/Jason.html

**M. A. Teh Noranis** received her bachelor of information technology degree in 1994 from University Utara Malaysia, M.S. degree in Artificial Intelligence in 2001 from University Putra Malaysia and Ph.D degree in Computer Science in 2008 from University Kebangsaan Malaysia. She is currently a Senior Lecturer at University Putra Malaysia. Her research interests are Artificial Intelligence, parallel processing, programming science, software agents and bioinformatics.

**N. Shahrin Azuan** received his B.S degree in ccomputer science in 1990 from Clarkson University, NY, U.S.A., M.S. degree in information systems engineering from UMIST, Manchester, U.K. in 1997 and Ph.D. degree in electrical engineering from University Teknologi Malaysia, Skudai, Malaysia, in 2008. His Ph.D. subject was on face recognition problems. He has been with Telekom Malaysia (TM) since 1990, and is currently holding a research position at the TM research & development. His research interests include computational intelligence, biometric technology, information security and networking.