Dynamic Fuzzy Logic Control of Genetic Algorithm Probabilities

Huijuan Guo[a], Yi Feng[b], Fei Hao[c], Shengtong Zhong[d], Shuai Li[e]
[a] Department of Computer Science
Taiyuan Normal University, Taiyuan, China
Email: tysyjsj2008@163.com
[b] Dalarna University, Borlange, Sweden
[c] Department of Computing
University of Bradford, Bradford, UK
Email: feehao@gmail.com
[d] Department of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
Email: shket@idi.ntnu.no
[e] Department of Computer Science
Ubiquitous Healthcare Research Center
Inje University, Gimhae, Busan, Korea
Email: ShuaiLi.sli@gmail.com

***Abstract*— Genetic Algorithms are traditionally used to solve combinatorial optimization problems. The implementation of Genetic Algorithms involves of using genetic operators (crossover, mutation, selection, etc.). Meanwhile, parameters (such as population size, probabilities of crossover and mutation) of Genetic Algorithm need to be chosen or tuned. In this paper, we propose a hybrid Fuzzy-Genetic Algorithm (FLGA) approach to solve the multiprocessor scheduling problem. Based on traditional Genetic Algorithms, a fuzzy logic controller is added to tune parameters dynamically which potentially can improve the overall performance. In detail, the probabilities of crossover and mutation is tuned by a fuzzy logic controller based on fuzzy rules. Compared to the Standard Genetic Algorithm (SGA), the results of experiments clearly show that the FLGA method performs significantly better.**

***Index Terms*— Genetic Algorithms, FLGA, SGA, Multiprocessor Scheduling, Fuzzy Logic Controller**

I. Introduction

GENETIC Algorithms (GA) have been widely used for classification, model selection, and other optimization tasks [1], [6], [9]. However, the performance of Genetic Algorithms is largely affected by choosing the values of their parameters, such as the population size, probabilities of crossover and mutation. A poor parameter settings usually lead to several problems such as the premature convergence, local optimum etc.

In this paper, we target to solve the multiprocessor scheduling problem. The multiprocessor scheduling problem consists of finding a task schedule that minimizes the execution time of the parallel program and the number of required processors. As Genetic Algorithms are naturally fit for the combinatorial optimization problem, it is an ideal choice for the multiprocessor scheduling problem. To prevent premature and local optimum, the selection of parameter setting are critical for the performance of Genetic Algorithms. In this paper, we propose to use a Fuzzy Logic Controller (FLC) to dynamically adapt the crossover and mutation probabilities in order to get a better performance. The fuzzy rules [2] of FLC are designed according to the consideration of improving fitness.

The structure of this paper is organized as follows: the multiprocessor scheduling problem is briefly introduced in Section 2. In Section 3, the detail of standard Genetic Algorithm are discussed. And then we introduce the basic concepts of fuzzy logic and our hybrid Fuzzy-Genetic Algorithm (FLGA) by adding a fuzzy logic controller in the standard Genetic Algorithm in Section 4. The results of experiments and discussions are provided in Section 5. Finally, Section 6 concludes the paper.

II. The Multiprocessor Scheduling Problem

The Multiprocessor Scheduling Problem is to minimize the execution time of the parallel program in order to optimize the problem. A multiprocessor system is a set of identical processors which has its own memory, and each pair of processors communicate exclusively by message passing through an interconnection network [3], [8], [10]. Each task is associated with a cost, representing its execution time. In order to be executed, each task of a given parallel program must be scheduled to some processor of a given multiprocessor system. In general, different

This work was supported by the National Natural Science Foundation of China (GrantNo.61273294).

doi:10.4304/jcp.9.1.22-27

schedules may need different numbers of processors; these communications can slow down the execution of the parallel program. So, different schedules of each task satisfying the priority constraints lead to different execution time of the parallel program.

The multiprocessor scheduling problem is a kind of typical optimization problems, it has been extensively studied by a large number of researchers [4], [14], [19]. Because an exhaustive search is often unrealistic, most of the work has been done on heuristic methods to find near-optimal solutions. There are two most studied heuristic methods for multiprocessor scheduling problem: list heuristic and meta-heuristic. The meta-heuristic is known as Genetic Algorithm.

In this paper, the Genetic Algorithm is incorporated to solve the multiprocessor scheduling problem [5]. A Genetic Algorithm is a guided random search method where elements (called individuals) in a given set of solutions (called the population) are randomly combined and modified (we call these combinations crossover and mutation respectively) until some termination condition is achieved. The population evolves iteratively (in the Genetic Algorithm terminology, through generations) in order to improve the fitness of its individuals. The fitness of an individual is said to be better than the fitness of another individual if the solution corresponding to is closer to an optimal solution. In each iteration, the crossover generates a new population in which the individuals are supposed to keep the good characteristics of the individuals of previous generation.

In the context of multiprocessor scheduling problem, Hou et al [7] proposed a kind of standard Genetic Algorithm. As standard Genetic Algorithms has some known drawbacks such as premature and local optimum, it is demanded to extend the Genetic Algorithms for better performance.

III. Genetic Algorithm

Genetic Algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover [1], [6], [13]. It has been widely used in computing to find exact or approximate solutions to optimization and search problems.

A. Basic Principals of Genetic Algorithm

Genetic Algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or the genotype or the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

A typical Genetic Algorithm requires two things to be defined: 1. A genetic representation of the solution domain. 2. A fitness function to evaluate the solution domain.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly, and then improve it through repetitive application of mutation, crossover, and selection operators.

B. Detail Steps of Genetic Algorithm

1) ***Initialization****:* Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be ”seeded” in areas where optimal solutions are likely to be found.

2) ***Selection****:* During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time-consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection. In our case, we will use roulette wheel selection with a certain selection rate.

3) ***Reproduction****:* The next step is to generate a second-generation population of solutions from those selected through genetic operators: crossover (also called recombination), and mutation.

Crossover is usually applied to selected pairs of ”parents” with a probability equal to a given crossover rate, and generate two new individuals by crossing the parents characteristics. Hence, the ”Children” share many of the characteristics of their ”parents”. Normally, there are several ways to go in crossover procedure like one-point crossover, two-point crossover and order crossover. Mutation is used to modify an individual with a certain

mutation rate. After crossover and mutation, we desire that some ”good” individuals will survive and reproduce, while some ”bad” individuals will be eliminated. Generally, the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above.

4) ***Termination****:* Repeat the steps mentioned above, it is expected to get some individuals with better and better fitness from generation to generation. Usually, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

IV. Hybrid Fuzzy-Genetic Algorithm

A. The Basic Concepts of Fuzzy Logic Controller

Fuzzy logic [2] is widely used in machine control. The term itself inspires a certain skepticism, sounding equivalent to ”half-baked logic” or ”bogus logic”, but the ”fuzzy” part does not refer to a lack of rigor in the method, rather to the fact that the logic involved can deal with fuzzy concepts - concepts that cannot be expressed as ”true” or ”false” but rather as ”partially true”. Although Genetic Algorithms can perform just as well as fuzzy logic in many cases, fuzzy logic has the advantage that the solution to the problem can be cast in terms that human operators can understand, so that their experience can be used in the design of the controller. This makes it easier to mechanize tasks that are already successfully performed by humans A Fuzzy Logic Controller (FLC) that is composed of the following: Knowledge base that includes the information given by the expert in the form of linguistic control rules; Fuzzification interface, which has the effect of transforming crisp data into fuzzy sets; Decision making unit, make the decision according to the knowledge base by using a reasoning method; Defuzzification interface, which produces a quantifiable result in fuzzy logic [8]. The general structure of an FLC is shown in Fig.1

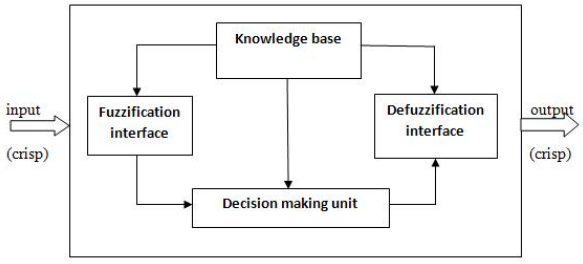

Figure 1. General Structure of a Fuzzy Logic Controller

The advantages of FLC lies in: it does not require precise, noise-free inputs. The output control is a smooth control function despite a wide range of input variations; in the target control system, FLC uses user-defined rules, it can be modified easily to improve system performance; new rules can easily be incorporated into the system; because of the rule-based operation, any reasonable number of inputs can be processed and numerous outputs generated; FL can control nonlinear systems that would be difficult or impossible to model mathematically.

B. Genetic Algorithm with Fuzzy Logic Controller

Standard Genetic Algorithm (SGA) has been known to offer significant approximate solution for a optimization or searching problem. However, despite of the successful application of SGA to these problems, the identification of the correct setting of genetic parameters (probabilities of crossover and mutation etc) for the problem is not an easy task. The performance of the Genetic Algorithm is directly affected by the careful selection of parameters.

In SGA, when a fixed probability of parameter is given at the beginning of algorithm, we can’ t change it in the process of implementation until it stops. In this situation, it is possible to destroy a high fitness individual when a large crossover probability is set. For a low crossover probability, sometimes it is hard to obtain better individuals and gets a slow convergence. High mutation probability will bring too much diversity and takes a longer time to get the optimal solution. Low mutation probability can not prevent the premature convergence and miss some near-optimal solutions.

How to set the parameters correctly? Which value is the most adaptive? Can we change the parameters in the process of implementation? If change, what value of the variation is suitable? This is the motivation to introduce fuzzy logic to control the parameters.

In this paper, the probabilities of crossover and mutation are just the objectives that we will control [16], [20]. So that we design two fuzzy logic controllers, one is for probability of crossover ($\Delta p_c(t)$), the other one is for probability of mutation ($\Delta p_m(t)$). A diagram of Hybrid Fuzzy-Genetic Algorithm is shown in Fig.2.

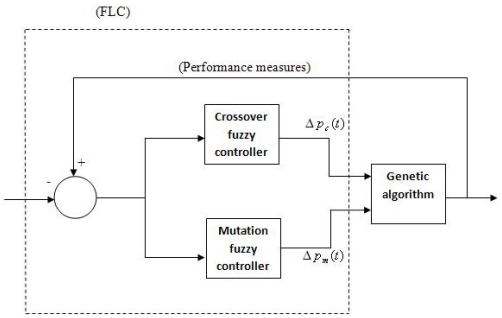

Figure 2. Hybrid Fuzzy-Genetic Algorithm

The dashed area above is just the FLC. The main idea is to use the current performance measures of GA as the inputs of FLC, and the outputs of the FLC are the new GA’s parameters. Current performance measures measured by the fitness are sent to the FLC, and then, FLC computes the new control parameters through the certain fuzzy rules. When GA gets the new parameters from FLC, it will repeat the steps above until the termination

condition is met. So the value of controller parameters may be renewed in every generation in our algorithm (FLGA). Fuzzy rules can be designed using different strategies. The general idea for updating the crossover and mutation probabilities is to consider the changes of the maximum fitness and average fitness in the GA population of two continuous generations.

V. Results of Experiments

A. Experimental Benchmarks

In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials. Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures. The experimental benchmarks in this paper are based on a multiprocessor scheduling problem with DAG (Directed Acyclic Graph). After a series of theoretical analysis, we will test the scheduling problem with benchmarks. The emphasis of this paper is to test the use fuzzy logic controller to dynamically control the parameters of Genetic Algorithm.

The benchmark instances in the experiment are provided by a tool called ALPES [11], [12]. It contains several well-known parallel programs, and each parallel program has two kind of size: large (-l) and medium (-m). Fig.3 shows the different number of tasks.

Parallel Program		Number of Tasks	Parallel Program		Number of Tasks
Bellford	-m	365	FFT	-m	194
	-l	992		-l	1026
Diamond1	-m	258	Gauss	-m	782
	-l	1026		-l	1227
Diamond2	-m	486	Iterative	-m	262
	-l	1227		-l	938
Diamond3	-m	731	MS-Gauss	-m	768
	-l	1002		-l	1482
Diamond4	-m	731	Prolog	-m	214
	-l	1002		-l	1313
Divconq	-m	382	Qcd	-m	326
	-l	766		-l	1026

Figure 3. Characteristics of test graphs

Notice that, different programs have different number of tasks, that means the best and average makespans of solutions are diverse which potentially makes the experiments results more general.

B. Results of Experiments and Discussions

In order to compare the quality of the solutions provided by SGA and FLGA, a traceable procedure is added in our algorithms so that we can record the best and average solutions of each generation. In this paper, we mainly focus on the effect of fuzzy logic controller, compare the results which from the Standard Genetic Algorithm (SGA) without FLC with our hybrid Fuzzy-Genetic Algorithm (FLGA). Actually, many factors can affect the results of Genetic Algorithms. Such as coding method, population size, different restrictions, different rules of crossover and mutation etc [15], [17], [18]. Our algorithm may get a better result if we change these factors, but that is not the main point of the experiments, the aim target here is to design an algorithm which can dynamically control the parameters of GA, so that the Genetic Algorithm can be self-controlled, and get a better performance.

In our experiments, two different strategies FLGA-1 and FLGA-2 are incorparated in order to see the diversity of our proposed method. The best and average solutions obtained by SGA and FLGA are shown in Fig.4.

Graph(DAG)		SGA (p_c=0.3, p_m=0.3)		SGA (p_c=0.8, p_m=0.02)		FLGA-1 The First Strategy			FLGA-2 The Second Strategy			Best Known
Name	Size	Best Schedule	Average Schedule	Best Schedule	Average Schedule	Best Schedule	Average Schedule	Average Improve	Best Schedule	Average Schedule	Average Improve	Best Schedule
Bellford	-m	327	364	341	392	112	137	227	91	105	259	71
	-l	1206	1302	1002	1065	413	465	837	285	299	1003	193
Diamond1	-m	286	301	326	397	268	286	15	186	243	58	131
	-l	1437	1466	1323	1401	761	801	665	330	351	1115	276
Diamond2	-m	520	586	536	604	253	260	326	156	266	320	127
	-l	**1508**	1602	1206	1264	892	915	687	344	402	1200	218
Diamond3	-m	937	966	890	920	520	586	380	432	465	501	176
	-l	1153	1207	957	1103	664	677	530	670	677	530	228
Diamond4	-m	436	502	392	466	226	231	332	195	224	339	132
	-l	735	965	742	968	335	372	593	310	346	619	164
Divconq	-m	326	377	354	397	217	220	157	229	259	118	97
	-l	902	1103	866	925	442	495	608	442	492	611	169
FFT	-m	137	194	115	134	94	109	85	76	86	108	29
	-l	562	669	435	496	216	265	404	193	212	457	102
Gauss	-m	726	913	771	886	386	412	501	290	341	572	226
	-l	**2012**	2231	1322	1435	715	886	1345	551	560	1671	307
Iterative	-m	92	118	116	131	39	71	47	39	65	53	16
	-l	253	320	286	299	168	224	96	175	224	96	47
MS-Gauss	-m	14071	14156	14965	15007	5113	5694	8462	4662	4933	9223	4598
	-l	**15816**	17963	13226	13495	3029	3730	14233	2866	3551	14112	2559
Prolog	-m	204	291	265	272	86	143	148	86	129	162	60
	-l	1524	1576	1433	1505	311	396	1180	324	404	1172	258
Qcd	-m	4591	6011	5022	5705	1426	1503	4508	1330	1366	4645	1173
	-l	16018	18026	15526	17216	4102	4215	13811	3116	3660	14366	2038

Figure 4. Comparison of the best and average solutions obtained by SGA and FLGA

From this table, we can clearly see that the results of FLGA are much better than those of SGA. There are two SGA, one is with a fixed $p_c = 0.3$ and $p_m = 0.3$, and the other one is with a fixed $p_c = 0.8$ and $p_m = 0.2$. The two SGA have a common ground that the probabilities of crossover and mutation will be a constant during the test process. However, in our proposed algorithm, the probabilities of crossover and mutation can be changed at each or certain generation according to a certain fuzzy rule.

The results are obvious, both FLGA-1 and FLGA-2 have much better solutions, which means we can get a much better schedule of multiprocessor scheduling problem. The results obtained by two SGA look like similar, but the results of the first SGA are worse than the second SGA at most situations. Because of the first SGA is equal to $p_m = 0.3$, actually it is a high probability of mutation, although it can jump out from the local-optimal, however, it may break some good individuals of each generation, so it may destroy some near-optimal solutions. Notice that, the bold number indicate the results will become very bad

if p_m is too high. The results may change significantly in the process of convergence.

We also compared the experimental results with the best well-known results. We can found that the solutions obtained by FLGA is closer to the best well-known solutions than SGA. However, both of them cannot be very close to the best well-known results, this is because of the drawback of the coding method used in the standard algorithm [7]. The standard Genetic Algorithm may miss some optimal solution. In spite of the fuzzy logic can make good efforts, it can't get the optimal solution. As mentioned before, the coding method and the problem itself are not the main point of this paper, so the improving of performance is acceptable. Moreover, the most important thing is that the fuzzy controller makes effort, which indicates the Genetic Algorithms can perform better with fuzzy logic controller.

VI. Conclusion

In this paper, we introduced a Hybrid Fuzzy-Genetic Algorithm, by incorporating Fuzzy Logic Controller to the Standard Genetic Algorithms, based on an existing hybrid Genetic Algorithm working on the multiprocessor scheduling problem. The Fuzzy Logic Controller is designated to tune the mutation and crossover probabilities in a dynamic fashion within the Genetic Algorithm evolution. The results of experiments clearly show that the FLGA outperform SGA in the test benchmark of the multiprocessor scheduling problem. Although it will cost extra time to adjust the parameters in each generation, it converges to the global optimum in a smaller number of generations. FLGA is an ideal alternative for the combinatorial optimization problems.

Acknowledgment

The authors are grateful to the anonymous referees for their valuable comments and suggestions for this paper.

References

[1] G. F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, Addison-Wesley, ISBN: 0-805-31196-3, 1998.

[2] L. A. Zadeh, *Fuzzy sets*, Information and Control, vol. 8, pp. 338-353, 1965.

[3] H. Li, D. Wang, Y. Zhang, *Knowledge-based genetic algorithms data fusion and its application in mine mixed-gas detection*, Journal of Software, Vol. 7, No. 2, pp. 303-307, Feb. 2012.

[4] H. Liu, Z. Xu, A. Abraham: *Hybrid Fuzzy-Genetic Algorithm Approach for Crew Grouping*, pp. 332-337, ISDA 2005.

[5] R. C. Correa, A. Ferreira, P. Rebreyend, *Scheduling multiprocessor tasks with genetic algorithms*, IEEE Trans. on Parallel and Distributed Systems, 10(8), pp. 825-837, 1999.

[6] S. Li, F. Hao, M. Li, H.-C. Kim, *Medicine Rating Prediction and Recommendation in Mobile Social Networks*, Proceedings of Springer LNCS, The 8th International Conference on Grid and Pervasive Computing (GPC 2013) and Colocated Workshops, Seoul, Korea, May 9-11, Vol. 7861, pp. 216-223, 2013.

[7] E. Hou, N. Ansari, H. Ren, *A genetic algorithm for multiprocessor scheduling*, IEEE Transactions on Parallel and Distributed Systems, vol. 5, no. 2, pp. 113-120, Feb. 1994.

[8] A. King, R. T. F. Radha, B. Rughooputh, H. C. S, *A fuzzy logic controlled genetic algorithm for optimal electrical distribution network reconfiguration*, ISBN: 0-7803-8193-9, 2004.

[9] F. Hao, S. Zhong, *Variable Precision Concepts and Its Applications for Query Expansion*, International Conference on Intelligent Computing (ICIC 2009), LNAI 5755, pp.154-165, 2009.

[10] L. Lin, M. Gen, *A Self-controlled Genetic Algorithm for Reliable Communication Network Design*, ISBN: 0-7803-9487-9, 2006.

[11] J. Kitajima, C. Tron, B. Plateau, *ALPES: A tool for the performance evaluation of parallel programs", in Environments and Tools for Parallel Scientific Computing*, J. J. Dongarra and B. Tourancheau, Eds., Amsterdam, The Netherlands, North-Holland, Series: Advance in Parallel Computing vol. 6, pp. 213-228, 1993.

[12] J. Kitajima, B. Plateau, *Modeling parallel program behavior in ALPES*, Information and Software Technology, vol. 36, no. 7, pp. 457-464, July 1994.

[13] Y. Kang, H. Lu, J. He, *A PSO-based Genetic Algorithm for Scheduling of Tasks in a Heterogeneous Distributed System*, Journal of Software, Vol. 8, No. 6, pp. 1436-1442, Jun 2013.

[14] D. Bertsekas, J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall International Editions, 1989.

[15] O. Ibarra, S. Sohn, *On mapping systolic algorithms onto the hypercube*, IEEE Transactions on Parallel and Distributed Systems, vol. 1, no. 1, pp. 48-63, Jan.1990.

[16] C. T. King, W. H. Chou, L. Ni, *Pipelined data-parallel algorithms: part ii - design*, IEEE Transactions on Parallel and Distributed Systems, vol .1, no. 4, pp, 486-499, Oct. 1990.

[17] C. Scheiman, P. Cappello, *A processor-time-minimal systolic array for transitive closure*, IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 3, pp. 257-269, May 1992.

[18] X. Yuan, *Multi-objective Optimization of Fuzzy Parallel Machines Scheduling Problem Using Nondominated Genetic Algorithms*, Journal of Software, Vol. 6, No. 10, pp. 2036-2042, Oct. 2011.

[19] G. Brassard, P. Bratley, *Fundamentals of Algorithmics*, Prentice-Hall ISBN: 0-13-335068-1, 1996.

[20] Y. Feng, *Dynamic Fuzzy Logic Control of Genetic Algorithm Probabilities*, Master thesis, Dalarna University, Sweden, 2008.

Huijuan Guo received the B.Sc. degree in Department of Computer Science and Engineering from Taiyuan Normal University, Taiyuan, China, 2002 and the M.Sc. degree from School of Mathematics and Computer Engineering from Xihua University, Chengdu, China, 2006. She has been working as a teacher with 7 years of teaching experience in Department of Computer Science and Engineering at Taiyuan Normal University since 2006. She holds the title of lecturer in 2008. Her interests include Artificial Intelligence, Pattern Recognition and Image Processing.

Yi Feng received the B.Sc. degree in School of Mathematics and Computer Engineering from Xihua University, Chengdu, China, in 2004. In 2008, he got his M.Sc in Artificial Intelligence from Dalarna Unveristy, Borlange, Sweden. Now he is working in Schneider Electric, Beijing, China.

Fei Hao received the B.Sc. and M.Sc. degrees in School of Mathematics and Computer Engineering from Xihua University, Chengdu, China, in 2005 and 2008, respectively. He is currently working in Huazhong University of Science and Technology (HUST) as a distinguished researcher. He is studying toward the PH.D degree in the Department of Computing at the University of Bradford, Bradford, UK. He has published over 20 research papers in International and National Journals as well as conferences. His research interests include intelligent information processing, social computing and time series data mining.

Shengtong Zhong received the B.Sc. degree in School of Mathematics and Computer Engineering from Xihua University, Chengdu, China, in 2005. Later of the same year, he joined Dalarna Unveristy, Borlange, Sweden where he finished his M.Sc in Artificial Intelligence in 2007. He is currently working in Norwegian University of Science and Technology (NTNU) as Ph.D. (Research Fellow). His research interest focus on time series analysis, especially with dynamic Bayesian networks.

Shuai Li was born in Aug 1987 in Zhejiang Province, China. In Sep 2011 he entered Inje University (Located in Gimhae, Busan, Korea) to pursue Master Degree of Computer Science, where he is studying at Department of Computer Science, and Ubiquitous Healthcare Research Center. Prior to this, he worked in SyChip Inc. (A Spin-out from the U.S. Bell Labs) and Huawei Technology. His research interests include Machine Learning, Data Mining, and Ubiquitous Computing.