Theme Article: Special Section on Secure and High-Speed Electronic Systems

Task Scheduling Strategy for Heterogeneous Multicore Systems

Juan Fang, Jiaxing Zhang, Shuaibing Lu, and Di Zhang
Beijing University of Technology

Hui Zhao and Yuwen Cui
University of North Texas

***Abstract*—For heterogeneous computing systems, various types of processor cores cause system performance degradation due to uneven load. In addition, the inability of multitasking to match the appropriate processor core is also an urgent problem. This article proposes a swarm intelligence task scheduling strategy based on the genetic algorithm (GA) for high-performance heterogeneous multicore processors. In order to avoid the falling into local optimal solutions, we employ an adaptive mutation and injection strategy in the algorithm design. This swarm intelligence solution detects the computing capacities of different cores by processing specified tasks beforehand, and then an appropriate solution will be explored by introducing an adaptive mutation GA. Our technique aims to execute various types of tasks on heterogeneous processing cores for optimal performance. Experimental results show that this scheduling strategy can reduce the additional overhead and improve parallel computing efficiency and system performance.**

Computer systems have changed a lot with the advance in manufacturing technologies. The single-core processor structure is restricted by physical design and other factors such as energy consumption, which will cause the focus of Moore's Law to shift from the simply increasing number of transistors to adding more cores that can be integrated on a chip.[1–3] Consequently, the direction of the processor development is changing from a single-core architecture to a multicore architecture. To handle the massive data computation in today's applications, the number of CPUs integrated on the single chip

Digital Object Identifier 10.1109/MCE.2021.3073654
Date of publication 19 April 2021; date of current version 6 December 2021.

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

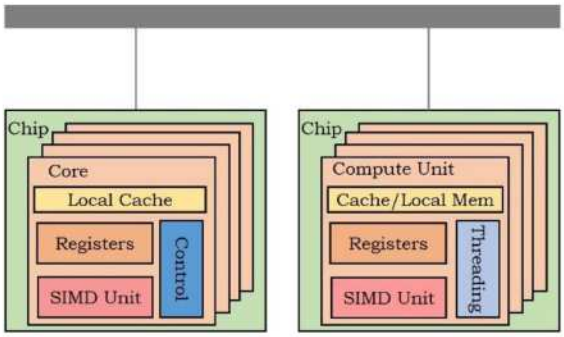

Figure 1. CPU–GPU heterogeneous computing platform.

has reached dozens. It is difficult for a processor to support such a huge amount of calculation.[4,5] Heterogeneous computing systems have so far been developed to incorporate various computing units such as CPUs, DSPs, GPUs, and FPGAs. Such heterogeneous systems combine computing units for various types of instruction sets, allowing different cores to cooperate to deal with multiple computing scenarios and improve system performance effectively. While heterogeneous systems provide a novel platform for emerging computing paradigms, such as big data and cloud computing, they also face many challenges. Figure 1 shows a CPU-GPU heterogeneous computing platform architecture. One of the challenges is the inability to match the appropriate processor cores during the execution of multitasking, which seriously restricts the performance of heterogeneous computing systems. An efficient task scheduling strategy is of great importance to solve this problem to improve system performance.

In this article, we focus on improving the performance of heterogeneous multicore processors and propose a swarm intelligence task scheduling strategy based on a genetic algorithm (GA). First, we propose a swarm intelligence task scheduling strategy for high-performance heterogeneous multicore processors based on a GA to improve the performance. Second, we introduce a mechanism to detect the computing capacities of different cores by characterizing specified tasks beforehand, and we propose to use an adaptive mutation strategy and injection strategy to avoid premature convergence and fall into a local optimal solution. Third, we perform an extensive experimental evaluation of the scheduling algorithm proposed. The evaluation results show that our solution can significantly improve system performance.

BACKGROUND

Reducing the load unbalance on heterogeneous computing systems and fully utilizing the computing capabilities of heterogeneous processing cores have become an urgent challenge to be tackled in the field of high-performance computing.

To solve the task scheduling problem in multitask computing, linear programming-based methods[6] have been explored by Baruah *et al.* There are other prior work focusing on the assignment problem of multitasks.[7–9] A combination of GA and particle swarm optimization (PSO) algorithm[10] was proposed by Li *et al.* An efficient hybrid GA and PSO algorithm for molecular dynamics simulation on heterogeneous supercomputers load balancing was further developed. To improve the efficiency of multitasks execution, a bypass shared cache management method[11] was developed by Juan *et al.* A free and open-source software toolkit based on GUN radio[12] was developed by Perea *et al.* Techniques have also been proposed to utilize the support vector machine classification method to classify computing tasks into CPU type and GPU type.[13,14] Machine learning-based techniques have also been proposed to improve consumer electronics performance.[15,16]

SWARM INTELLIGENCE TASK SCHEDULING STRATEGY

The scheduling strategy designed in this article aims to improve the overall operating efficiency of the system. Binding the execution relationship between various types of tasks and heterogeneous processor cores appropriately, we employ an improved genetic algorithm (IM_GA) to reduce the additional overhead caused by general task scheduling schemes. Aiming at the problem of falling into local optimal solutions and premature convergence in a GA, we first do prescheduling by detecting the computing capacities of different cores by processing specified tasks beforehand, and then we

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

propose to add an adaptive mutation and an injection strategy to alleviate these problems. Adaptive mutation solves the problem that genes of excellent individuals with high adaptabilities are destroyed and enters a random search. If the mutation probability is too low, then it is difficult to introduce new genes, which makes the algorithm's later search stalled.[17] The injection strategy has a powerful supplementary diversity mechanism, which alleviates the earlier convergence in the algorithm and can make the algorithm easier to avoid reaching the local optimal solution.

GA for NP-Hard

The task scheduling problem of heterogeneous multicore processors is an NP-hard problem. How to quickly obtain the task scheduling strategy of the multitask model is a key indicator to evaluate the performance of the algorithm. Efficient algorithms can reduce the time consumed to solve the task scheduling strategy. The scheduling strategy designed in this article improves the GA to build the model. Selection, crossover, and mutation in the GA are the key operations of population iterative evolution. The final evolution and optimization effects of the entire population are related to the fitness function, crossover probability, and mutation probability set in the algorithm. Therefore, in order to solve the problem of low efficiency and local optimal solution in the GA, we add adaptive mutation and injection strategies to improve the GA.

Tasks Prescheduling

To bind the execution relationship between heterogeneous processing cores and various types of tasks, we detect the computing capacities of different cores by processing specified tasks beforehand. We introduce an regulatory factor γ to adjust the proportion of tasks for testing, where $0 < \gamma < 1$. Based on this, we calculate the total completion time and find a minimum one as the initial setting of the intelligence task scheduling algorithm.

Steps of Intelligence Task Scheduling Strategy

We propose a swarm intelligence algorithm that improves the GA using the following steps. The flow chart is shown in Figure 2.

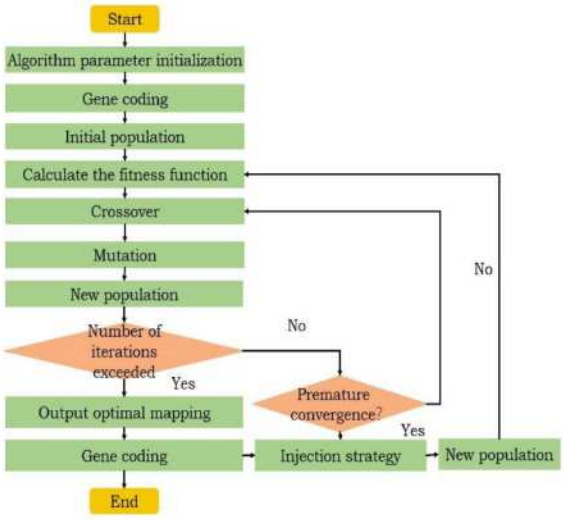

Figure 2. Flow chart of GA modeling and solving.

Step 1: Initialize parameters of the intelligence task scheduling strategy. Each individual in the initial population denotes a scheduling scheme. In this model, we combine the task pre-scheduling and the dependencies between tasks as follows:

(i) We first calculate the $H(T_i)$, where $H(T_i)$ denotes the height value of task T_i based on the DAG diagram.

(ii) The tasks are allocated to GPU and CPU based on the prescheduling strategy.

(iii) According to the $H(T_i)$ obtained in step (i), the tasks randomly assigned on each core are sorted in an increasing order, and the sorting result is the execution order of tasks on the processing unit.

(iv) When the size of the initial population does not meet the requirements, we switch back to (ii); otherwise, we go to step 2.

Step 2: We calculate the fitness function of all individuals in the population, and all individuals in the population are ranked based on the fitness order. The selection of the fitness function affects whether the algorithm can find the optimal solution and the convergence speed directly. We judge the quality of the solution according to the value of the fitness function.

We use S_{ca} to denote the size of the current population, and we use $T_{total}(S)$ to denote the

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

time required to execute the scheduling scheme S. The total completion time of the current population T_{sum} is represented by the following equation:

$$T_{sum} = \sum_{i=0}^{S_{ca}-1} T_{total}(S),\ 0 \le S \le S_{ca} - 1. \quad (1)$$

The value of the fitness function for the scheduling scheme S is defined as follows:

$$Fit\ (S) = \frac{T_{sum} - T_{total}(S)}{T_{sum}}. \quad (2)$$

Step 3: Crossover. The rate of the crossover is 0.5. In the last step, two adjacent chromosomes are crossed and sequenced to produce offspring. We then recalculate the fitness of the parents and offspring and choose a new offspring in descending order based on the fitness. The sizes of the population and the new population are consistent with the size of the parent population.

Step 4: Mutation. P_m is the selection of mutation probability in parameters has a great impact on the performance and the behavior of the GA. When the probability of mutation is too high, the genes of outstanding individuals with high adaptability are easy to destroy, and then we will enter a random search. Therefore, we propose an adaptive mutation strategy in this article. When $Fit_{max} \ge Fit$, the P_m is calculated using (3), otherwise, P_m is set to 0.8. k_m is set to 0.2.

$$P_m = K_m \frac{Fit_{max} - Fit_s}{Fit_{max} - Fit}. \quad (3)$$

Here Fit_{max} denotes the fitness function with the maximum value among all scheduling strategies and Fit_S denotes the fitness of the scheduling strategies in the population.

Step 5: If the number of iterations reaches the maximum number, we will select the task scheduling scheme with the largest fitness function; otherwise, the optimal solution of the continuous multigeneration population is found, and then the Hamming distance among the optimal solutions of the continuous multigeneration population to determine whether potential premature convergence has occurred. If there is no premature in the procedure, we go to step 3; otherwise, we start to use the injection strategy and go back to step 2. After successive generations, the injection strategy is applied if the solution remains the same.

EVALUATIONS

We use Sugon W580-G20 to build our platform and the operating system is Linux. The CPU is Intel Xeon E5-2620 and the GPU is NVIDIA Tesla P100. In our platform, the memory is 128Gb and the hard disk is 2Tb. We use the widely used task generation tool to create a test program and then conduct simulations to test the performance of our GA-based schemes for heterogeneous multicore processors. TGFF was originally a standardized random benchmark developed by R.P. Dick and D.L. was widely used in the research of task scheduling and allocation in the field of software and hardware collaboration.[18] TGFF is suitable for many applications that need to generate pseudo-random graphs. The latest version (3.0) expands the TGFF function and provides a highly configurable random graph generator that can generate multiple types of tasks.

Convergence Speed Test

Convergence speed is a key indicator for evaluating the performance of GAs. It is expressed by the time of obtaining the global approximate optima solution by the swarm intelligence algorithm interaction. We use a first-come-first-serve scheduling algorithm (FCFS), PSO, traditional GA, and the IM_GA proposed to test the convergence speed.

The convergence times of the TGFF-1 scheduling scheme solved by the three swarm intelligence algorithms are shown in Figure 3 as the first group of bars. Experiments show that IM_GA has the strongest convergence ability compared with the PSO and the GA. When the IM_GA is used to model and solve the optimal task scheduling scheme, the global optimal solution is found when the iteration proceeds to the 27th time. The task scheduling scheme obtained by this method is the most efficient among the three.

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

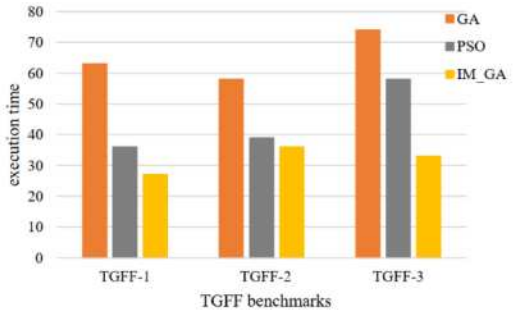

Figure 3. Convergence times of TGFF series solved by three algorithms.

The convergence speed results of GA, PSO, and IM_GA tested by the TGFF-2 experiment are the second group of bars shown in Figure 3. The IM_GA is still the fastest of the baseline algorithms. In the TGFF-2 test program, the iteration times of GA, PSO, and IM_GA reach the convergence state are 58, 39, and 36, respectively. PSO and IM_GA have little difference in the number of iterations and this is because the particle swarm algorithm falls into the local optimal solution at the 39th iteration, and the obtained scheduling strategy is the least efficient of the three algorithms. The convergence of the three algorithms in solving the optimal task scheduling scheme under the TGFF-3 is the third group of bars in Fig.3.

Although the execution time of the task scheduling strategy obtained after the optimal solutions decoding of the PSO and IM_GA are the same, the convergence speed of IM_GA is still the fastest of the three. The optimal solution is found about the 33rd time, while the PSO needs 58 times to find the same optimal solution as IM_GA. In summary, from the convergence of the three sets of test programs (TGFF-1,TGFF-2, TGFF-3) created by TGFF, IM_GA has the strongest convergence capability compared with the other two algorithms.

Performance of CPU–GPU Heterogeneous Multicore Processor

In many heterogeneous multiprocessor systems, due to the very high-cost in terms of GPU energy, CPU–GPU heterogeneity has become the most commonly used architecture in recent heterogeneous systems. Therefore, we first conducted a performance test for the CPU–GPU heterogeneous multicore processor. We run

Table 1. Different configurations of CPU and GPU.

CPU	0	2	4	8	16	32
GPU	1	1	1	1	1	1

eight different test instances on the GPU through the default method and records the execution time, and then gradually increase the number of CPU cores enabled in the experiments. As the number of CPU cores involved in task processing increases, the overall efficiency ratio of GPU to CPU will gradually decrease. Under the load balancing measures of this strategy, the workload of CPU will also gradually increase. To observe the impact of changes in CPU configuration on this strategy, in the five experiments after the first run, the number of CPU cores enabled changes to 2, 4, 8, 16, and 32, respectively. The setting is shown in Table 1.

As shown in the first group in Figure 4(a), the completion time of the system using FCFS, GA, PSO, and IM_GA in TGFF-1 is 4851, 4786, and 4302 ms, respectively, that is, the performance obtained by the four algorithms becomes better gradually (FCFS < GA < PSO < IM_GA). The task

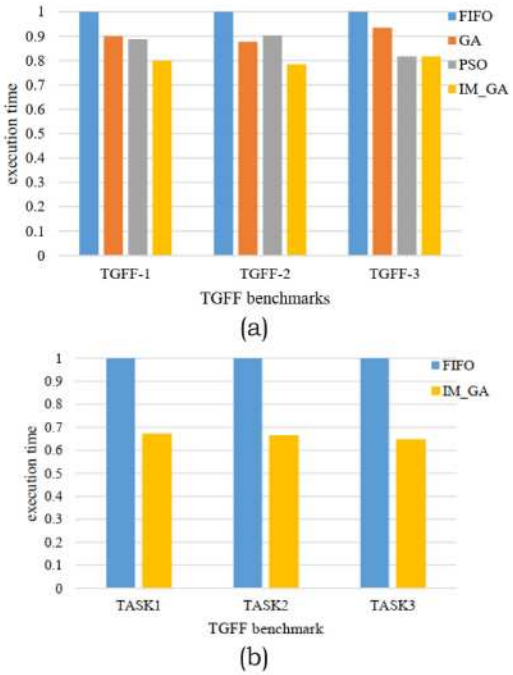

Figure 4. Execution time of GA, PSO and IM GA under the (a) CPU–CPU and (b) heterogenous multi-core processors.

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

scheduling strategy decoded by IM_GA is better than the task scheduling strategy decoded by the GA and PSO, which is increased by 10.19% and 8.97%, respectively. GA and PSO fall into a local optimal solution; the execution time of the scheduling strategy obtained by the three algorithms in TGFF-2 are 6693, 6897, and 5983 ms, respectively. The performance of the GA is better than that of PSO, but these two are once again caught in the local optimal solution, which is shown in the second data group in Figure 4. The performance of the two is 9.30% and 11.98% lower than that of the IM_GA respectively; in the TGFF-3 test program experiment, the performance of the medium PSO strategy and the IM_GA strategy are both 5072 ms, which is 11.84% higher than the 6502 ms of the GA, and neither of them falls into a local optimal solution.

Performance of Heterogeneous Multicore Processor

In the last section, we have proved that the IM_GA has a significant performance improvement in its convergence speed and heterogeneous multicore processor systems. On this basis, we have expanded the types of heterogeneous multicore processors. We add NPU, FPGA computing units to the processor core. As shown in Figure 4(b), in the three sets of experiments we tested, compared with the FCFS scheduling strategy, the performance of the scheduling strategy based on GA increased by 33.02%, 33.73%, and 35.54%, which has an average increase of 34.10%. The specific performance is as follows: when the FCFS scheduling strategy is used, the system calculates the time for the three groups of test programs to be 3528, 2416, and 2760 ms, respectively. When the GA-based scheduling strategy is used, the execution times for the three groups of test programs are respectively 2363 ms, 1601 ms, and 1779 ms, which proves once again that the task scheduling strategy of heterogeneous multicore processors based on IM_GA can significantly improve the performance of the system.

CONCLUSION

This article proposes a GA-based scheduling strategy that allocates different tasks to the most matching processor cores for computation, which reduces the amount of system idle time. In order to solve the problem of premature convergence and falling into local optimal solution in GAs, we propose to add an adaptive mutation and injection strategy to the swarm intelligence algorithm. Both strategies can increase the diversity of the population gene types in the later stage of the GA, which allows the algorithm to jump out of the local optimal solution faster.

ACKNOWLEDGMENTS

This work was supported in part by the Beijing Natural Science Foundation (4192007), in part by the National Natural Science Foundation of China (61202076), along with other government sponsorships, and in part by the National Science Foundation (NSF) Grants (#2046186 and # 2008911).

REFERENCES

1. A. Danowitz, K. Kelley, J. Mao, J. P. Stevenson, and M. Horowitz, "CPU DB: Recording microprocessor history," *Commun. ACM*, vol. 55, no. 4, pp. 55–63, 2012, doi: 10.1145/2181796.2181798.
2. R. Chau, "Process and packaging innovations for Moore's law continuation and beyond," in *Proc. IEEE Int. Electron Devices Meet.*, 2019, pp. 1–1, doi: 10.1109/IEDM19573.2019.8993462.
3. S. P. Mohanty, "Ai for consumer electronics-has come a long way but has a long way to go," *IEEE Consum. Electron. Mag.*, vol. 9, no. 3, pp. 4–5, May 2020, doi: 10.1109/MCE.2020.2968754.
4. X. Liao, and N. Xiao, "Emerging high-performance computing systems and technology," *SINICA Informationis*, vol. 46, no. 9, pp. 1175–1210, 2016, doi: 10.1360/N112016-00147.
5. W. Shen, L. Sun, D. Wei, W. Xu, X. Zhu, and S. Yuan, "Load-prediction scheduling for computer simulation of electrocardiogram on a CPU-GPU PC," in *Proc. IEEE 16th Int. Conf. Comput. Sci. Eng.*, 2013, pp. 213–218, doi: 10.1109/CSE.2013.42.
6. S. K. Baruah, "Task partitioning upon heterogeneous multiprocessor platforms," in *Proc. IEEE Realtime Embedded Technol. Appl. Symp.*, 2004, pp. 536–543, doi: 10.1109/RTTAS.2004.1317301.
7. L. Wan, K. Li, J. Liu, and K. Li, "Efficient cpu-gpu cooperative computing for solving the subset-sum problem," *Concurrency Comput., Pract. Exp.*, vol. 28, no. 2, pp. 492–516, 2016, doi: 10.1002/cpe.3629.

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.

8. J. Kim, Y. Nam, and M. Park, "Energy-aware core switching for mobile devices with a heterogeneous multicore processor," *IEEE Consum. Electron. Mag.*, vol. 8, no. 6, pp. 68–75, Nov. 2019, doi: 10.1109/MCE.2019.2941352.
9. Y. Xu, K. Li, J. Hu, and K. Li, "A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Inf. Sci.*, vol. 270, pp. 255–287, 2014, doi: 10.1016/j.ins.2014.02.122.
10. D. Li, K. Li, J. Liang, and A. Ouyang, "A hybrid particle swarm optimization algorithm for load balancing of MDS on heterogeneous computing systems," *Neurocomputing*, vol. 330, pp. 380–393, 2019, doi: 10.1016/j.neucom.2018.11.034.
11. J. Fang, X. Hao, Q. Fan, K. Li, and H. Zhao, "Efficient data transfer in a heterogeneous multicore based CE systems using cache performance optimization," *IEEE Consum. Electron. Mag.*, vol. 8, no. 5, pp. 46–50, Sep. 2019, doi: 10.1109/MCE.2019.2923928.
12. P. J. L. Perea, E. Jurez, F. Pescador, and C. Sanz, "Efficient open source software radio on heterogeneous multi-core embedded platforms," *IEEE Consum. Electron. Mag.*, vol. 10, no. 2, pp. 27–36, Mar. 2021, doi: 10.1109/MCE.2020.3010179.
13. Y.-H. Wang, J.-Z. Qiao, S.-K. Lin, and T.-I. Zhao, "A task allocation model for CPU-GPU heterogeneous system based on SVMS," *J. Northeastern Univ. (Natural Sci.)*, vol. 37, no. 8, 2016, Art. no. 1089, doi: 10.12068/j.issn.1005-3026.2016.08.006.
14. S. Iturriaga, S. Nesmachnow, F. Luna, and E. Alba, "A parallel local search in CPU/GPU for scheduling independent tasks on large heterogeneous computing systems," *J. Supercomput.*, vol. 71, no. 2, pp. 648–672, Feb. 2014, doi: 10.1007/s11227-014-1315-6.
15. M. A. Sayeed, S. P. Mohanty, E. Kougianos, and H. P. Zaveri, "Neuro-detect: A machine learning based fast and accurate seizure detection system in the IOMT," *IEEE Trans. Consum. Electron.*, vol. 65, no. 3, pp. 359–368, Aug. 2019, doi: 10.1109/TCE.2019.2917895.
16. R. K. Nath, H. Thapliyal, A. Caban-Holt, and S. P. Mohanty, "Machine learning based solutions for real-time stress monitoring," *IEEE Consum. Electron. Mag.*, vol. 9, no. 5, pp. 34–41, Sep. 2020, doi: 10.1109/MCE.2020.2993427.
17. L. Morra, S. P. Mohanty, and F. Lamberti, "Artificial intelligence in consumer electronics," *IEEE Consum. Electron. Mag.*, vol. 9, no. 3, pp. 46–47, May 2020, doi: 10.1109/MCE.2019.2962163.
18. R. P. Dick, D. L. Rhodes, and W. Wolf, "Tgff: Task graphs for free," in *Proc. 6th Int. Workshop Hardw./Softw. Codesign.*, 1998, pp. 97–101, doi: 10.1109/HSC.1998.666245.

Juan Fang is a Professor with the Faculty of Information Technology, Beijing University of Technology, Beijing, China. Contact her at fangjuan@bjut.edu.cn

Jiaxing Zhang is a student with the Faculty of Information Technology, Beijing University of Technology, Beijing, China. Contact her at S201761381@emails.bjut.edu.cn.

Shuaibing Lu is a Lecturer with the Faculty of Information Technology, Beijing University of Technology, Beijing, China. Contact her at lushuaibing@bjut.edu.cn.

Hui Zhao is an Assistant Professor with the Department of Computer Science and Engineering, University of North Texas, Denton, TX, USA. Contact her at hui.zhao@unt.edu.

Di Zhang is a student with the Faculty of Information Technology, Beijing University of Technology, Beijing, China. Contact her at zhangd@emails.bjut.edu.cn.

Yuwen Cui is currently working toward the Ph.D. degree at the University of North Texas, Denton, TX, USA. Contact him at cuiyuwen@my.unt.edu.

Authorized licensed use limited to: University of North Texas. Downloaded on April 28,2022 at 21:21:28 UTC from IEEE Xplore. Restrictions apply.