An Effective Approach for Genetic-Fuzzy Mining Using the Graphics Processing Unit

Chun-Hao Chen[1], Yu-Qi Huang[2] and Tzung-Pei Hong[2, 3]
[1]Department of Information and Finance Management, National Taipei University of Technology, Taipei, Taiwan
[2]Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[3]Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan
Email: chchen@ntut.edu.tw, cream08111230@gmail.com, tphong@nuk.edu.tw

***Abstract*—Association analysis is an important technique for finding relationships among the given transactions. In real applications, since transactions may have quantitative values, the fuzzy-set theory was utilized for mining fuzzy association rules. To extract useful rules, the given membership functions were the critical factor. The genetic-fuzzy mining approaches were thus presented to obtain appropriate membership functions to mine fuzzy association rules. However, the evolution process was time-consuming. In this paper, we then propose an algorithm to reduce the processing time using the graphics processing unit (GPU), namely the GPU-based Genetic-Fuzzy Mining algorithm (GPU-GFM). It first collects the chromosomes from the population and the chromosomes generated by genetic operators. Then, chromosomes are sent to GPU to calculate the fitness values. As a result, a fitness value matrix is returned. At last, when reaching the termination condition, the best chromosome will be outputted for mining fuzzy association rules. Experiments were also conducted on simulation datasets to show the performance of the proposed approach.**

Keywords-Association rule; genetic algorithm; fuzzy set; fuzzy association rule; graphics processing unit.

I. INTRODUCTION

Data mining is commonly used to extract knowledge from the given datasets, and the Apriori algorithm is the well-known technique to be utilized for discovering relationships among the transactions [2]. An association rule is an expression of the relevance between items. For instance, $X \rightarrow Y$ is an association rule, where X and Y are itemsets. It means that when someone buys the items in the X, then the customer has a high probability of buying Y at the same time. For example, a customer who buys *milk* and *jam* will also buy *bread* could be found as an association rule and represented as $\{milk, jam\} \rightarrow \{bread\}$.

The abovementioned rule mining approach can only be used to mine binary association rules [2]. In other words, items in the transaction can only be considered as to buy or not to buy, which limits the content of data analysis. However, in real applications, the purchased quantity exists and should be taken into consideration in the mining process. Therefore, by using fuzzy sets, many algorithms have been proposed for mining fuzzy association rules [8][9][10][14]. The main concept of those mining algorithms is that the quantitative values are first transformed into fuzzy representations using the given membership functions. Then, the fuzzy representations are employed to discover fuzzy association rules. For example, Hong et al. proposed an approach for mining fuzzy association rules from quantitative data [8]. Ouyang et al. proposed an algorithm to mine direct weighted and indirect weighted fuzzy association rules [14].

In those fuzzy association rule mining algorithms, the membership functions are given in advance. Because the predefined membership functions may not be appropriate for all kinds of datasets to mine fuzzy association rules, and because to obtain appropriate membership functions is an optimization problem, the genetic-fuzzy mining algorithms have then been proposed to obtain the membership functions for mining fuzzy association rules using various evolutionary algorithms, chromosome representations, genetic operators as well as evaluation functions [1][4][5][6][7][13][16][17]. However, the main problem of the existing approaches is the evolution process is time-consuming.

With the prevalence of General-Purpose computing on Graphics Processing Units (GPGPU), in this paper, we propose a GPU-based Genetic-Fuzzy Mining algorithm (GPU-GFM) for handling the problem. It first generates the initial population randomly. Then, the population is sent to GPU to execute the Max-Min-Arithmetical (MMA) crossover operator. The offspring and the original chromosomes will return to CPU. After that, the mutation operator is performed. To calculate the fitness values of chromosomes, all chromosomes and transactions are sent to GPU to calculate the fuzzy values. As a result, the fuzzy value matrix is returned to the CPU. At last, the chromosomes and the fuzzy value matrix are again sent to the GPU for calculating the fitness values of chromosomes. A fitness value matrix is then returned to the CPU. When reaching the termination condition, the best chromosome is outputted for mining fuzzy association rules. Experiments were also conducted on simulation datasets with different parameter setting to show the efficiency and effectiveness of the proposed approach.

II. RELATED WORK

In this section, the genetic-fuzzy mining algorithms are stated in Section II.A. The graphics processing unit based optimization approaches are described in Section II.B.

A. *Genetic-Fuzzy Mining Algorithms*

Hong et al. proposed an algorithm that consists of two phases for mining fuzzy association rules [6]. In the first phase, the genetic algorithm has been utilized to obtain the membership functions according to the number of large 1-itemsets and suitability of membership functions in a chromosome. In the second phase, the derived membership functions are used to discover rules. To reduce the time for the evolution process, Hong et al. took the divide-and-conquer strategy into consideration and proposed another algorithm for solving the genetic-fuzzy mining problem [7]. The main concept is that every item has its own genetic process to find membership functions. The obtained membership functions are gathered for mining fuzzy association rules. Because various criteria should be considered for the optimization process, Alhajj et al. proposed a multi-objective genetic algorithm for automated clustering to obtain fuzzy association rules [1]. Considering multiple minimum supports, Chen et al. then proposed an optimization algorithm for finding membership functions for items at a certain level. Then, the obtained membership functions are employed to extract multi-level fuzzy association rules [4]. In addition, the multi-objective genetic-fuzzy mining algorithm has been proposed for discovering multi-level fuzzy association rules [5]. Matthews et al. proposed an evolutionary-based approach for mining temporal fuzzy association rules for web usage data [8]. Palacios et al. proposed an algorithm, namely FARLAT-LQD, for obtaining both suitable membership functions and fuzzy association rule from imprecise transactions [15]. They first use the genetic algorithm to membership function based on 3-tuples linguistic representation model. Then, the frequent-pattern tree-based algorithm is employed to mine fuzzy association rules. Ting et al. proposed an enhanced genetic-fuzzy mining algorithm for membership functions and rule discovery [16]. The main advantage of the algorithm is that it used the structure-based representation, which considered the structures of membership functions for chromosome encoding.

B. *GPU-based Optimization Approaches*

With the popularity of computational intelligence nowadays, we often rely on computers to find the near optimization solution using metaheuristic algorithms. However, it usually needs a lot of time to obtain the result. After the general-purpose computing on the graphic processing unit was launched in 2011 by NVIDIA, the GPU parallel processing was employed to speed up the evolution process. For instance, Yousef et al. designed the genetic algorithm with GPU to solve the university course timetable problem [18]. Benaini et al. proposed an optimization algorithm with GPU to solve the vehicle routing problem because the path should be arranged in a short time. As a result, the proposed approach significantly reduced the time cost of obtaining the routing path [3]. Due to the government policies and the increase in environmental protection awareness in recent years, the energy-saving and efficient dynamic flexible flow shop scheduling has become a dynamic problem worthy of studying. To maintain the original efficiency, the principle of energy saving must be taken into consideration. In addition, scheduling problems will change with the different situations, so the time cost is a major issue. Luo et al. executed the GA method by GPU for parallel calculation to reduce significantly the time cost [8]. In the field of 3D printing, it often hopes that the loss of materials is as small as possible. Therefore, the support material needs to be calculated to find the closest or best solution. Li et al. used the GPU to handle the optimization problem to discover the schedule [12].

III. PROPOSED GPU-BASED GENETIC-FUZZY MINING ALGORITHM

In this section, the framework of the proposed GPU-based Genetic-Fuzzy Mining algorithm (GPU-GFM) is illustrated in Section III.A. The pseudo code of the GPU-GFM is stated in Section III.B. Components of the GPU-GFM are described in Section III.C.

A. *The Framework of the GPU-GFM*

The GPU-GFM framework is shown in Fig. 1.

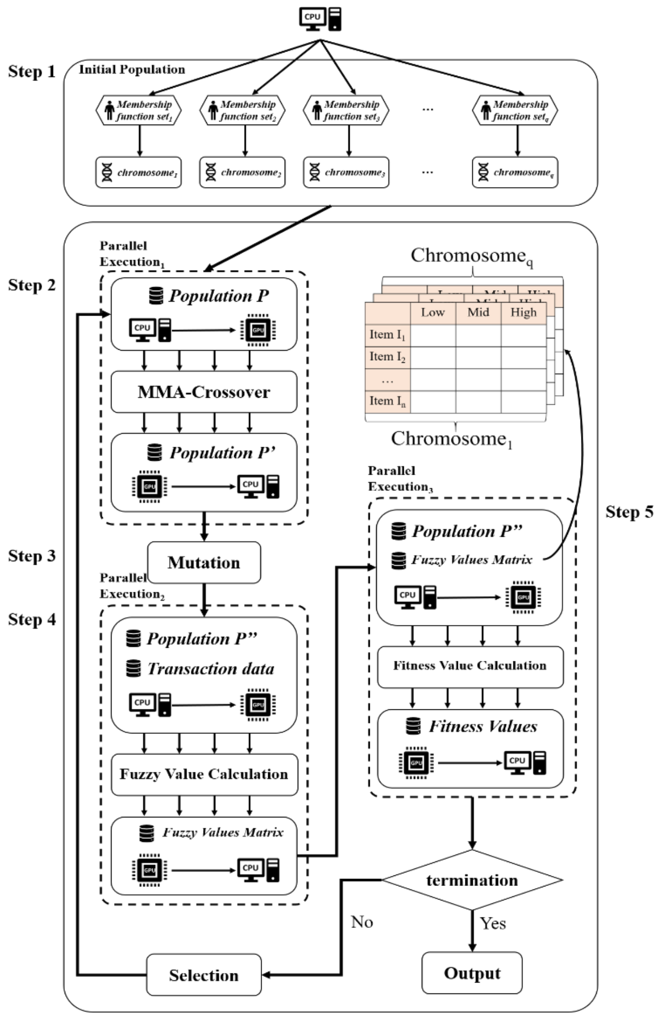

Figure 1. The framework of the GPU-GFM.

In Fig. 1, it shows that the proposed GPU-GFM contains five steps. They are: (1) The initial population *P* is generated randomly according to the predefined population size.; (2) The crossover operator is executed by GPU for speeding up the process to generate offspring which is merged to P to get P'; (3) The mutation operator is executed. After mutation, the population P'' is generated; (4) The GPU is utilized to transform quantitative transactions to fuzzy values for chromosomes; (5) Based on the fuzzy values matrix, the fitness values for chromosomes are calculated by GPU. Steps 1 to 5 will continue until reaching the termination condition.

In the following, we give a simple example to state the GPU-GFM. Assume that the population size is fifty. In Step 1, fifty chromosomes are generated randomly as the initial population P. Each chromosome represents a set of membership functions for all items.

In Step 2, assume that the crossover rate is 0.8. Forty chromosomes will be selected to generate offspring. Let two chromosomes as a pair. Thus, totally twenty pairs will be sent to GPU for offspring generation. The used crossover operator, the MMA crossover, will generate four candidate chromosomes for a given pair. Hence, after crossover, eighty offspring will be generated and sent back to the CPU. Then, the eighty chromosomes are merged to the P to form P'. In other words, P' has 130 chromosomes after the crossover operator.

In Step 3, for mutation operator, assume that the mutation rate is 0.04 and two chromosomes are mutated and added to P' to form P''. After mutation, the P'' has 132 chromosomes.

In Step 4, the quantitative transactions and P'' are sent to GPU for fuzzy value calculation. After calculation, a three-dimension matrix called the fuzzy value matrix will be generated. The index for the matrix including the chromosome number, item number, and fuzzy region number. Take (C_1, I_1, *Low*) is 5 as an example. It means the fuzzy value of the fuzzy region *Low* for item I_1 in chromosome C_1 is 5. The matrix is then sent back to the CPU for the next step.

In Step 5, the P'' and the *fuzzy value matrix* are sent to GPU again for calculating the fitness values of the 132 chromosomes. In the GPU, a thread is used to calculate the fitness value of a chromosome. It first calculates the number of large 1-itemset according to the given *fuzzy value matrix* and the predefined minimum support. Then, the suitability of the chromosome is calculated. After calculation, an array of fitness values is used to store the fitness value of chromosomes and returned to the CPU. At last, if the termination condition is reached, the best chromosome is outputted. Otherwise, it will go for the next generation.

B. Pseudo Code of the GPU-GFM

Based on the GPU-GFM framework, the pseudo code of the proposed algorithm is stated in Table I.

TABLE I. PSEUDO CODE OF GPU-GFM ALGORITHM.

Input:
Transaction data *TD.*
Parameters:
Population size *pSize*, crossover rate p_c, mutation rate p_m, generation *G,*
Population *P*, number of Items *itemNum*, minimum support *ms,*
Fuzzy Value Matrix *FVM.*
Output:
The best chromosome *BC.*
Procedure GPU-GFM:
1. *P* ← InitialPopulation(*pSize, itemNum*)
2. FOR *iteration* = 1 to ***G*** DO
3.   *GPU_P* ← cuda.memcpy_htod(*P*)
4.   *GPU_P'* ← MMA_Crossover(p_c, *GPU_P, GPU_ThreadIdx*)
5.   *P'* ← cuda.memcpy_dtoh(*GPU_P'*)
6.   *P''* ← Mutation(*p'*, p_m)
7.   (*GPU_P'', GPU_TD)* ← cuda.memcpy_htod(*p'', TD*)
8.   *GPU_FVM* ← FuzzyValueCalculation(*GPU_P'', GPU_TD, GPU_ThreadIdx*)
9.   *FVM* ← cuda.memcpy_dtoh(*GPU_FVM*)
10.   (*GPU_P'', GPU_FVM*) ← cuda.memcpy_htod(*P'', FVM*)
11.   *GPU_FitnessValues* ← FitnessValueCalculation(*GPU_P'', GPU_FVM, GPU_ThreadIdx*)
12.   *FitnessValues* ← cuda.memcpy_dtoh(*GPU_FitnessValues*)
13.   *P* ← selection(*P''*, *FitnessValues, pSize*)
14. END *iteration* FOR LOOP
15. BestChromosome ← selectBestChro(*P*, *FitnessValues*)

From Table I, the proposed algorithm first generates the initial population *P* randomly according to the predefined *pSize* (Line 1). Then, it starts the evolution process (Lines 2 to 14). The MMA crossover is then executed on GPU to generate offspring, and the results are stored in *GPU_P'* (Lines 3 to 4). The *GPU_P'* will return to CPU and store in P' (Line 5). The mutation operator is executed to get P'' (Line 6). To calculate fuzzy values of chromosomes, it sends P'' and transactions *TD* to GPU (Line 7). The fuzzy values of chromosomes are calculated (Line 8). The result *GPU_FVM* is returned to the CPU and stored in *FVM* (Line 9). For the fitness evaluation, the P'' and *FVM* are again sent to GPU (Line 10) for calculating fitness values (Line 11). The result *GPU_FitnessValues* is returned to CPU and stored in *FitnessValues* (Line 12). The selection process is executed to generate the next population (Line 13). Finally, if reaching the termination condition, the best chromosome is outputted (Line 15).

C. Components of the GPU-GFM

1) Encoding Scheme

In the proposed approach, a chromosome is used to represent a set of membership functions that are: $MFSet_1$, $MFSet_2$, …, $MFSet_i$, …, $MFSet_n$. The $MFSet_i$ means the membership functions for the *i*-th item. Let *m* linguistic terms are used for an item, then the $MFSet_i$ can be represented as ((c_1, w_1), (c_2, w_2), …, (c_j, w_j), …, (c_m, w_m)), where c_j and w_j are center and width of a membership function.

2) Initial Population and Genetic Operators

In the proposed GPU-GFM, the initial population is generated randomly. As to the genetic operators, the max-min-arithmetical (MMA) crossover operator and one-point mutation are employed to generate offspring. The elitist selection strategy is utilized for reproduction.

3) Fitness Evaluation

The fitness function used to evaluate a chromosome in the proposed approach is the same with the existing work [7]. The formula is stated as follows:

$$f(C_q) = |L_1| \,/\, \text{suitability}(C_q),$$

where $|L_1|$ is the number of large 1-itemsets that can be generated using the membership functions in C_q, and suitability(C_q) is used to avoid bad membership functions that are overlapping or separate too much.

IV. Experimental Results

In this section, experiments were made to show the performance of the proposed approach. The experimental environment is stated as follow: CPU: Intel(R) Core(TM) i5-9300 CPU @ 2.4GZ, GPU: NVIDIA GeForce GTX 1650. The proposed approach is implemented by Python 3.6.12. with the PyCUDA 2020.1 and CUDA v10.2 for deploying the algorithm on the GPU. The experimental datasets are generated by the IBM generator. By using the four parameters that are *T*: average transaction length， *I*: average maximum large itemset length，*N*: number of items, *D*: transaction size, different simulation datasets can be generated.

Experiments were first made to show the convergence of the proposed approach. After 1000 generations, the results are shown in Fig. 2.

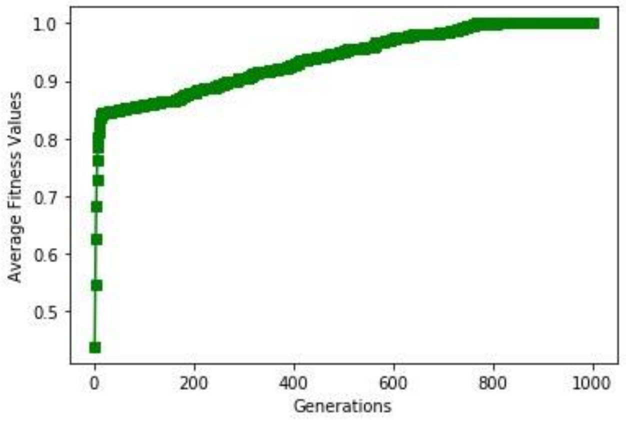

Figure 2. Convergence results of the proposed approach.

From Fig. 2, we can see that the average fitness values grow along with the increase of the generations, and finally converge to a certain value.

Experiments were then made to show the execution time of the proposed approach on the datasets with 170 items but different transaction sizes, including 10K, 30K, 50K, 90K. The results are shown in Fig. 3.

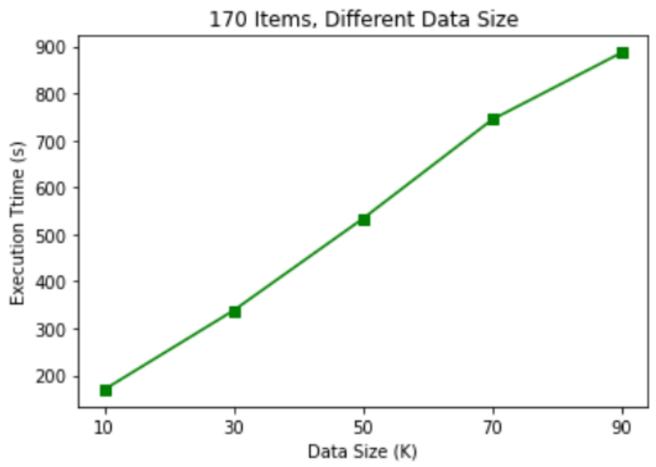

Figure 3. Execution time of the GPU-GFM on different transaction sizes.

From Fig. 3, we can observe that the execution time on different data sizes increase linearly. It indicates that the proposed approach is efficient. Then, the experiments on the datasets with 10K transactions but different numbers of items were made, and the results are shown in Table II.

TABLE II. Execution Time of the Proposed Approach with Different Number of Items .

Dataset	Execution Time (s)	Increasing Ratio
T2I2N0.032D10	85	-
T4I2N0.064D10	153	1.8 (= 153/85)
T6I2N0.096D10	217	1.4 (= 217/153)
T8I2N0.128D10	240	1.1 (= 240/217)
T10I2N0.16D10	298	1.2 (= 298/240)

Table II shows along with the increasing number of items from 32 to 160, the execution time increases from 89 to 298 seconds. From the increasing ratio, when we double the number of items from 32 to 64, the ratio is 1.8. The other three values are between 1.1 to 1.4. It means the execution time still increases linearly.

At last, comparisons of the proposed approach and the previous approach [6] in terms of execution time for a generation on the datasets with different transaction sizes are shown in Table III.

TABLE III. Comparisons of Proposed and Previous Approaches in Terms of Execution Time.

Data Size	Proposed Method	Previous Method	Speed-Up Ratio
10K	0.647 sec.	572.997 sec.	885
30 K	1.697 sec.	1710.271 sec.	1007
50 K	2.764 sec.	3154.801 sec.	1141
70 K	3.818 sec.	4537.069 sec.	1188
90 K	4.674 sec.	5172.524 sec.	1106

From Table III, we can easily observe that the proposed approach is better than the previous approach in terms of the execution time of the evolution process, and the highest speed-up ratio is up to 1188 times. From the experimental results, we can conclude that the proposed GPU-GFM is efficient significantly.

V. Conclusion and Future Work

Association rule mining is always an interesting research topic since it can be utilized to discover useful relationships among items. In real applications, transactions may have quantitative values. Fuzzy association-rule mining algorithms are employed to handle that. To extract more information, the genetic-fuzzy mining algorithms have then been presented to find membership functions automatically for fuzzy association-rule mining. Because the evolution process is time-consuming, in this paper, we thus propose an algorithm, namely the GPU-based Genetic-Fuzzy Mining algorithm (GPU-GFM), to speed up the evolution process. Experimental results show that: (1) the GPU-GFM is efficient no matter the increase of the number of transactions or items; (2) When compared to the previous approach, the highest speed-up ratio is up to 1188 times in terms of execution time. In the future, we will try to enhance the proposed approach to observe more useful rules, e.g., using all large itemsets instead of only large 1-itemsets as an evaluation function.

Acknowledgment

This research was supported by the Ministry of Science and Technology of the Republic of China under grant MOST 109-2622-E-027-032 and MOST 110AO12B.

References

[1] R. Alhajj and M. Kaya, "Multi-objective genetic algorithms based automated clustering for fuzzy association rules mining," *Journal of Intelligent Information Systems*, Vol. 31, No. 3, pp. 243-264, 2007.

[2] R. Agrawal, T. Imielinski and A. Swami, "Database mining: a performance perspective," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 6, pp. 914-925, 1993.

[3] A. Benaini and A. Berrajaa, "Genetic algorithm for large dynamic vehicle routing problem on GPU," *International Conference on Logistics Operations Management*, pp. 1-9, 2018.

[4] C. H. Chen, T. P. Hong and Vincent S. Tseng, "Genetic-fuzzy mining with multiple minimum supports based on fuzzy clustering," *Soft Computing*, Vol. 15, No. 12, pp. 2319-2333, 2011.

[5] C. H. Chen, J. S. He and T. P. Hong, "MOGA-based fuzzy data mining with taxonomy," *Knowledge-Based Systems*, Vol. 54, pp. 53-65, 2013.

[6] T. P. Hong, C. H. Chen, Y. L. Wu and Y. C. Lee, "A GA-based fuzzy mining approach to achieve a trade-off between number of rules and suitability of membership functions," *Soft Computing*, Vol. 10, No. 11, pp. 1091-1101, 2006.

[7] T. P. Hong, C. H. Chen, Y. C. Lee and Y. L. Wu, "Genetic-fuzzy data mining with divide-and-conquer strategy," *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 2, pp. 252-265, 2008.

[8] T. P. Hong, C. S. Kuo and S. C. Chi, "Trade-off between computation time and number of rules for fuzzy mining from quantitative data," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 9, No. 5, pp. 587-604, 2001.

[9] T. P. Hong, C. S. Kuo and S. C. Chi, "Mining association rules from quantitative data," *Intelligent Data Analysis*, Vol. 3, No. 5, pp. 363- 376, 1999.

[10] C. Kuok, A. Fu and M. Wong, “Mining fuzzy association rules in databases,” *SIGMOD Record*, Vol. 27, No. 1, pp. 41-46, 1998.

[11] J. Luo, S. Fujimura, D. E. Baz and B. Plazolles, “GPU based parallel genetic algorithm for solving an energy efficient dynamic flexible flow shop scheduling problem,” *Journal of Parallel and Distributed Computing*, Vol. 133, pp. 244-257, 2019.

[12] Z. Li er al., "A GPU based parallel genetic algorithm for the orientation optimization problem in 3D printing," *International Conference on Robotics and Automation*, pp. 2786-2792, 2019.

[13] S. G. Matthews, M. A. Gongora, A. A. Hopgood and S. Ahmadi, "Web usage mining with evolutionary extraction of temporal fuzzy association rules," *Knowledge-Based Systems*, Vol. 54, pp. 66-72, 2013.

[14] W. Ouyang and Q. Huang, "Mining direct and indirect weighted fuzzy association rules in large transaction databases," *International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 128-132, 2009.

[15] A. M. Palacios, J. L. Palacios, L. Sánchez and J. Alcalá-Fdeza, "Genetic learning of the membership functions for mining fuzzy association rules from low quality data," *Information Sciences*, Vol. 295, No. 20, pp. 358-378, 2015.

[16] C. K. Ting, T. C. Wang, R. T. Liaw and T. P. Hong, "Genetic algorithm with a structure-based representation for genetic-fuzzy data mining," *Soft Computing*, Vol. 21, No. 11, pp. 2871–2882, 2016.

[17] T. C. Wang and R. T. Liaw, "Multifactorial genetic fuzzy data mining for building membership functions," *IEEE Congress on Evolutionary Computation*, pp. 1-8, 2020.

[18] A. H. Yousef et al., "A GPU based genetic algorithm solution for the timetabling problem," *International Conference on Computer Engineering & Systems*, pp. 103-109, 2016.