Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UTILIZING GENETIC ALGORITHM INTEGRATED WITH INTELLIGENT OPERATORS AND SARSA FOR EXTRACTING HIGH UTILITY ITEMSETS

LOGESWARAN K[1] , SAVITHA S[2], SURESH S[3], ANANDAMURUGAN S[4]

[1]Assistant Professor(Sr.G), Department of AI, Kongu Engineering College, Tamilnadu, India

[2]Assistant Professor, Department of CSE, K.S.R. College of Engineering, Tiruchengode, Tamilnadu, India

Associate Professor, Department of Database Systems, School of Computer Science and Engineering, Vellore, India.

Associate Professor, Department of IT, Kongu Engineering College, Tamilnadu, India

E-mail: [1]klogesbech@gmail.com

ABSTRACT

This research article presents a novel approach for mining High Utility Itemsets (HUIs) by integrating Genetic Algorithm (GA) with SARSA algorithm. It begins by providing a comprehensive overview of GA's fundamental principles and operational procedures, followed by an in-depth exploration of SARSA algorithm components, supported by diagrammatic representations. The core contribution of this study is the introduction of the Intelligent Genetic Algorithm with on-policy Reinforcement Learning (IGA_RLON) methodology, which is thoroughly elaborated upon. The effectiveness of IGA_RLON is meticulously evaluated in terms of execution time, convergence speed, and the percentage of successfully mined HUIs, through comparative analysis with established methods such as IGA_RLOFF, HUPEUMU-GRAM, and HUIM-BPSO. This article aims to advance the field of HUI mining by proposing a robust and efficient algorithmic framework.

Keywords: *Genetic Algorithm, SARSA Algorithm, Reinforcement Algorithm, High Utility Itemset Mining, Data Mining, Control Parameters*

1. INTRODUCTION

High utility itemset mining (HUIM) is a significant task in data mining and machine learning. It is concerned with discovering itemsets that are highly valuable, based on a utility function that measures their usefulness. HUIM is used in a variety of domains, such as marketing, healthcare, e-commerce, and finance, where the identification of valuable itemsets can provide valuable insights for decision-making.

1.1. Genetic Algorithm

GA is a computational technique inspired by the process of natural selection and evolution in biology. It is a metaheuristic optimization algorithm that is used to find optimal solutions to complex problems. GA works by simulating the process of evolution, where candidate solutions (individuals) are treated as genes and undergo genetic operations such as crossover and mutation to produce offspring (new solutions) [1]. The fitness of these solutions is evaluated using an objective function, and the process is repeated iteratively until the optimal solution is found. GA has been applied in various domains, including engineering, economics, and machine learning, due to its ability to handle complex and multi-dimensional problems [2].

GA has been successfully applied in HUIM to efficiently discover valuable itemsets from large datasets. In HUIM, the objective is to find itemsets with high utility, which is determined by a utility function that measures the usefulness of an itemset. GA-based approaches for HUIM involve representing itemsets as chromosomes, and using genetic operations such as crossover and mutation to generate new itemsets [3]. The fitness of these itemsets is evaluated using the utility function, and the process is repeated iteratively until the optimal solution is found.

The key objective of HUIM is to find itemsets with high utility, which is determined by a utility function. The utility function measures the usefulness or value of an itemset in a given context. For instance, in the context of retail sales, the utility function may be defined in terms of the profit obtained from selling an itemset [4]. In healthcare, the utility function may measure the effectiveness of a treatment based on the outcomes of a clinical trial. In finance, the utility function may be defined in terms of the return on investment of a portfolio of securities.

The applications of HUIM are numerous and diverse. In marketing, HUIM can help identify product bundles that are likely to be purchased together, which can increase revenue and customer satisfaction [5]. In healthcare, HUIM can aid in identifying effective treatments for specific diseases based on patient outcomes. In e-commerce, HUIM can help identify items that are frequently purchased together, which can be used to provide personalized recommendations to customers. In finance, HUIM can aid in portfolio optimization by identifying securities that are likely to provide a high return on investment.

One of the primary challenges in HUIM is to efficiently discover high utility itemsets from a large dataset. Since the number of itemsets can be exponential, it is necessary to develop efficient algorithms to identify the most valuable itemsets [6]. Several evolutionary based algorithms have been proposed for HUIM, including HUPEUMU-GRAM and HUIM-BPSO. This current research explores about the discovery of HUIs using GA with its operators calibrated using SARSA learning.

1.2. GA Working Procedure

Figure 1 represent the flowchart of workflow process involved in GA. The algorithm begins by initializing a population of candidate solutions, which are evaluated based on their fitness. The fittest individuals are then selected for reproduction, with the aim of producing even better solutions in the next generation [7].

Crossover and mutation are applied to the selected individuals to produce new offspring, which are then evaluated for their fitness. The fittest individuals from the new generation are selected again for further reproduction through crossover and mutation, and the process is repeated until a stopping criteria is satisfied [8].

The stopping criteria may be based on a fixed number of iterations, reaching a certain level of fitness, or other factors. If the stopping criteria is not satisfied, the algorithm loops back to the selection stage to continue the process. Once the stopping criteria is satisfied, the algorithm terminates and returns the best solution found.

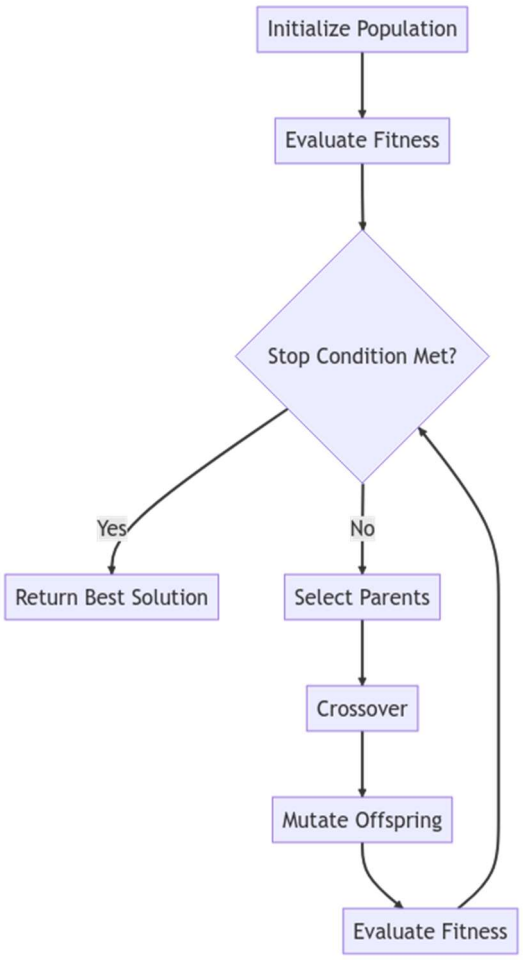

Figure 1 : Genetic Algorithm Workflow

1.3. Crossover Operation

Figure 2 illustrates the process involved in crossover operation of GA. There are two parents that contribute genetic material to the offspring. Before the crossover operation occurs, a random number is generated. If the random number is below the crossover rate (C_R), then a crossover operation occurs at a randomly chosen crossover point. Otherwise, the offspring are simply copies of their respective parents [9]. The fitness of each offspring is then evaluated and the resulting offspring are generated

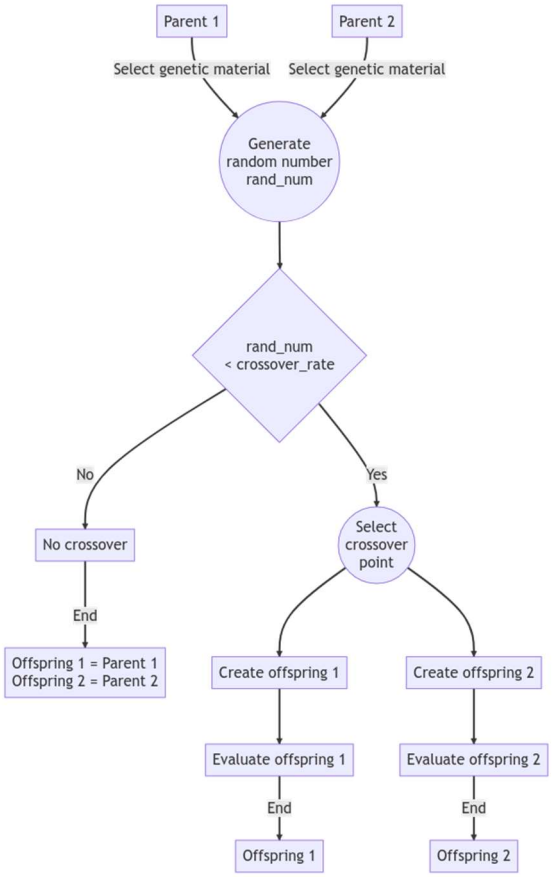

Figure 2 :Crossover Operation

1.4. Mutation Operation

Figure 3 shows the process involved in mutation operation. For Mutation process, the output of crossover operation goes as an input. Mutation process is applied on each individual. It takes single parent and undergoes mutation with a certain probability, determined by a specified mutation rate. Before mutation occurs, a random number is generated [10]. If the random number is below the mutation rate, then a gene is randomly selected for mutation, and a new allele is generated for the selected. The resulting offspring is then evaluate. If the random number is above the mutation rate, the parent is simply copied to become the offspring

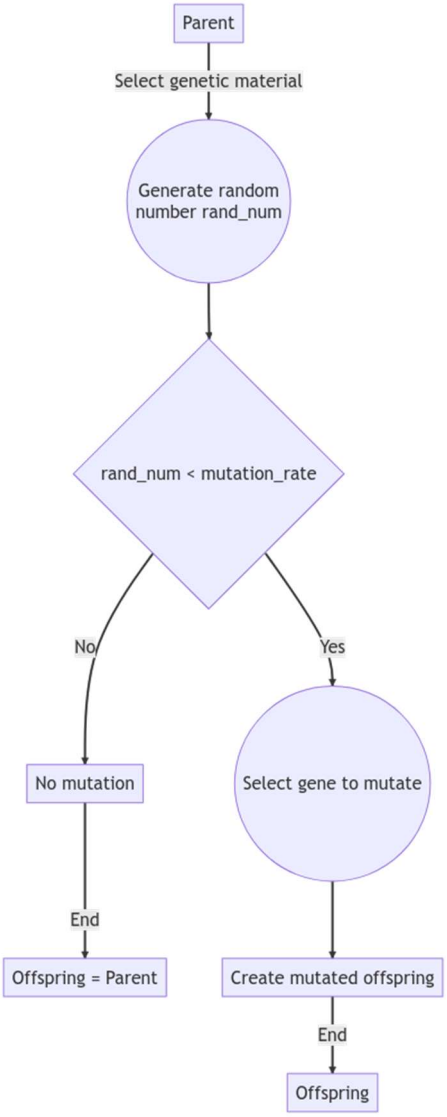

Figure 3 : Mutation Operation

2. LITERATURE SURVEY

The methodology used in [11] involves the development of a Decomposition based on a compact Genetic Algorithm (DcGA) for mining closed high-utility itemsets (CHUIs) in large-scale databases. The process begins with transforming the transaction database into a graph network, followed by the application of community detection to create groups of highly correlated transactions. The compact genetic algorithm is then applied to each community to find local closed high utility patterns, and the results are concatenated to derive global closed high utility patterns. This approach aims to efficiently mine CHUIs in a limited time and obtain a good predictive model for pattern recommendation. The methodology also includes a

comparison of the proposed DcGA with existing pattern mining algorithms in terms of runtime and effectiveness analysis, as well as convergence performance.

A novel genetic algorithm (GA)-based approach for safeguarding sensitive high utility itemsets during utility mining, aiming to minimize information loss while protecting critical data was designed in [12]. It introduces a flexible evaluation function and leverages the downward closure property and pre-large concept to accelerate chromosome evaluation, reducing database rescanning costs. Highlighting the proliferation of electronic data and the necessity for privacy-preserving techniques, it underscores the importance of addressing confidentiality concerns. This GA-based strategy represents the first attempt at privacy-preserving high utility itemset mining, employing transaction insertion for data concealment. It underscores the complexities of data mining, particularly in managing privacy, and underscores the need for efficient algorithms to mitigate these challenges.

The researcher proposed an Ant Colony Optimization (ACO)-based methodology for mining high-utility itemsets in [13]. It involves leveraging the behavior of ant colonies to efficiently explore the search space and identify itemsets with high utility values in large datasets. The methodology likely includes the design of pheromone update rules, heuristic information, and exploration-exploitation strategies tailored for high-utility itemset mining. Additionally, it may incorporate mechanisms for handling constraints and optimizing performance metrics such as runtime and solution quality.

The methodology involves the development of an evolutionary algorithm that optimizes for both frequency and utility simultaneously in [14]. This entails creating specialized fitness functions and employing Pareto-based optimization strategies to efficiently extract itemsets meeting both criteria. Additionally, the methodology likely includes techniques for addressing scalability and efficiency concerns when dealing with large datasets.

An evolutionary approach caled Artificial Bee Colony (ABC) algorithm was proposed in [15] to discover high utility itemsets. It involves initializing artificial bees to explore the solution space iteratively. The bees adjust positions to represent changes to candidate itemsets, guided by the principles of the ABC algorithm. Fitness evaluation assesses itemset utility based on predefined criteria. Selection mechanisms choose promising itemsets for the next iteration. The iterative process continues until convergence criteria are met, with performance evaluated based on effectiveness in discovering high utility itemsets.

3. PROPOSED METHODOLOGY

In the proposed IGA_RL$_{ON}$ approach, the GA control parameters namely Crossover and Mutation operators are calibrated intelligently using the action chosen from the SARAS learning algorithm. Later, IGA_RL$_{ON}$ was used to mine the high utility itemset from the benchmark dataset

3.1. Methodology of SARSA

A typical methodology involved in SARSA learning is illustrated in the Figure 4. Below are the steps involved in SARSA learning.

1. Initialize the Q(s,a) table with arbitrary values for all possible state-action pairs.
2. Select an action (a) using an $\varepsilon - greedy$ policy, which means that there's a chance of selecting a random action with probability ε (e.g., 10%), and selecting the action with the highest Q-value with probability $1-\varepsilon$.
3. Perform the selected action 'a' and observe the reward 'R' and the next state 's''.
4. Select the next action 'a'' for the next state 's'' using the same $\varepsilon - greedy$ policy.
5. Update the $Q(s,a)$ value for the current state-action pair using the SARSA update rule as in the Eq. 1, which is:

$$\underbrace{Q(s,a)}_{\text{Updated Q Value}} = \underbrace{Q(s,a)}_{\text{Current Q Value}} + \alpha[\underbrace{R + \gamma \overbrace{Q(s',a')}^{\text{Target Policy as Behaviour Policy}}}_{\text{Target Q value}} - \underbrace{Q(s,a)}_{\text{Current Q Value}}] \quad (1)$$

where α is the learning rate, γ is the discount factor, and $Q(s',a')$ is the Q-value for the next state-action pair.

6. Set the current state to the next state 's'' and the current action to the next action 'a''.
7. Repeat steps 2-6 until the algorithm converges to the optimal Q-values for all state-action pairs.

3.2. Design of IGA_RL$_{ON}$

Figure-3 illustrate the architecture design of IGA_RL$_{ON}$ algorithm. Initially, the itemsets are represented as binary chromosomes of population. Roulette wheel selection (RWS) strategy is used to

Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

select parent chromosomes from the population. State set S_s is calculated from the parent chromosomes. An appropriate action should be chosen based on $\varepsilon - greedy$ action selection scheme. From the action chosen, the Crossover rate (C_R) and Mutation Rate (M_R) are calibrated intelligently.

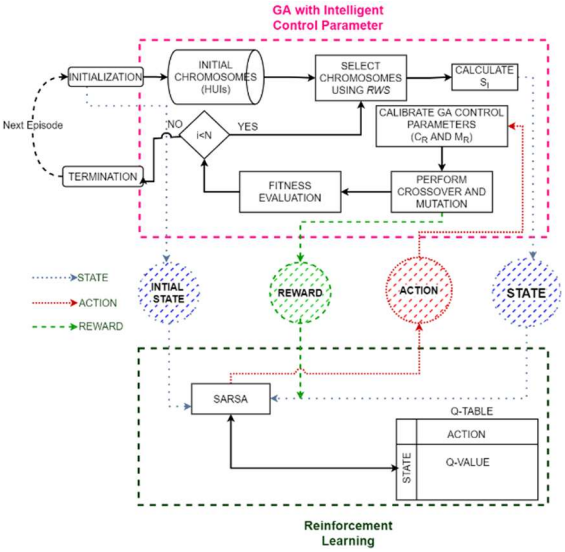

Figure 4 : Architecture Design of IGA_RL$_{ON}$

Now the chromosomes are updated using (C_R) and (M_R) and the fitness of chromosomes are evaluated. Finally the chromosomes having fitness greater than the minimum utility threshold are added to HUI list. Parallelly the Q-value and Q-Table are updated based on SARSA learning.

3.3. Design Methodology

IGA_RL$_{ON}$ starts with initializing the P_S, T_{max} and t and generate representation for chromosomes. Fitness of the population is calculated using the Eq. 2 and calculate the state using the Eq. 3. Choose the action to be taken using the Eq. 5 and calculate the reward for current action using the Eq. 4. Update the Q-value and Q-table using Eq. 1 and calibrate the GA control parameter namely C_R and M_R by choosing appropriate values from the Table 4.4 using the current action a. Perform crossover and mutation operation using the updated C_R and M_R values. Calculate the fitness of the updated offspring individual and if it is greater than min_util then add it into HUI list. Replace the existing parent individual with new offspring individual in the population. Repeat the above process until termination condition is reached. Figure 5 illustrate above process by pictorial representation.

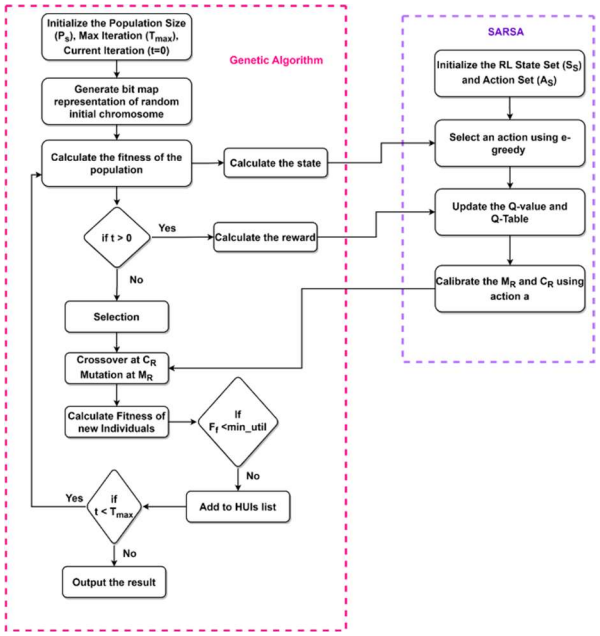

Figure 5 : Design Methodology of IGA_RL$_{ON}$

$$F_f(C_i) = \upsilon(X, L_j) \qquad 2$$

$$f(s_v) = f(f_f) + f(\mu_f) + f(p_d) \qquad 3$$

$$\boldsymbol{R = \omega_c R_c + \omega_m R_m} \qquad 4$$

$$\Upsilon(s_t, a_t) = \begin{cases} Q_{\max}(s_t, a)) & with\ probability\ (1-\varepsilon) \\ random(a) & with\ probability\ (\varepsilon) \end{cases} \qquad 5$$

4. EXPERIMENTAL EVALUATION

Performance metrics namely Execution time, convergence speed and discovered HUIs are used to measure the delivery of IGA_RL$_{ON}$. The performance of proposed IGA_RL$_{ON}$ approach is compared with HUPE$_{umu}$-GRAM, HUIM-BPSO and IGA_RL$_{OFF}$. These algorithms are applied on standard dataset namely chess, accident_10%, mushroom and connect from SPMF repository.

4.1. Execution Time

Figure 6 to 9 represents the execution time taken by the IGA_RL$_{ON}$, HUPE$_{umu}$-GRAM, HUIM-BPSO and IGA_RL$_{OFF}$ to mine the HUIs from the chess, accident_10%, mushroom and connect datasets are measured and a graph is plotted by taking

Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

min_util along the x-axis and execution time in seconds along y-axis.

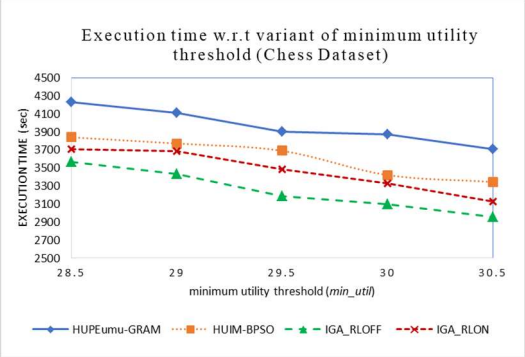

Figure 6 : Execution Time for Chess dataset

The main inference from the Figure 6 is that the proposed IPGA_RL$_{ON}$ reduces the execution time required to mine the HUIs from chess dataset by 12.52% and 4.05% when compared with HUPE$_{umu}$-GRAM and HUIM-BPSO respectively. On the other side, IGA_RL$_{ON}$ takes 6.74% more execution time when compared with the IGA_RL$_{OFF}$ to mine HUIs from chess dataset.

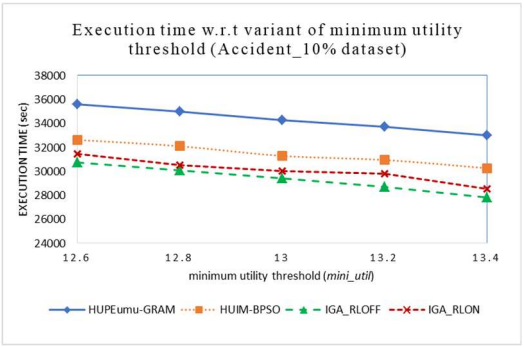

Figure 7 : Execution Time for Accident_10% dataset

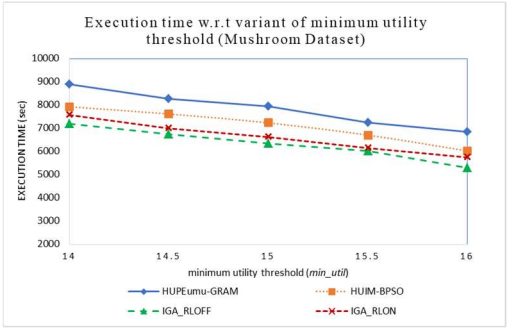

Figure 8 : Execution Time for Mushroom dataset

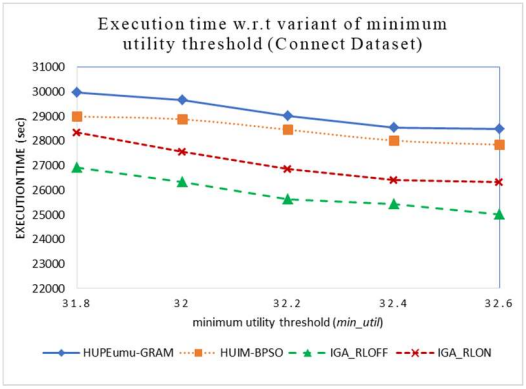

Figure 9 : Execution Time for Connect dataset

Figure 7 shows that the proposed IGA_RL$_{ON}$ reduces the execution time required to mine the HUIs from accident_10% dataset by 12.44% and 4.43% when compared with HUPE$_{umu}$-GRAM and HUIM-BPSO respectively. On the other side, IGA_RL$_{ON}$ takes 2.37% more execution time when compared with IGA_RL$_{OFF}$ to mine the HUIs from accident_10% dataset.

Figure 8 shows that the proposed IGA_RL$_{ON}$ reduces the execution time required to mine the HUIs from mushrom dataset by 15.54% and 6.82% when compared with HUPE$_{umu}$-GRAM and HUIM-BPSO respectively. On the other side, IGA_RL$_{ON}$ takes 4.68% more execution time when compared with IGA_RL$_{OFF}$ to mine the HUIs from mushroom dataset.

Figure 9 shows that the proposed IGA_RL$_{ON}$ reduces the execution time required to mine the HUIs from connect dataset by 6.97% and 4.71% when compared with HUPE$_{umu}$-GRAM and HUIM-BPSO respectively. On the other side, IGA_RL$_{ON}$ takes 4.78% more execution time when compared with IGA_RL$_{OFF}$ to mine the HUIs from mushroom dataset.

4.2. Convergence Speed

The convergence speed of IGA_RL$_{ON}$ is measured using the dataset chess, accident_10%, mushroom and connect dataset from SPMF repository and compared with HUPE$_{umu}$-GRAM, HUIM-BPSO and IGA_RL$_{OFF}$. Graph is plotted by varying *min_util* along x-axis and No. of HUIs discovered along y-axis.

Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

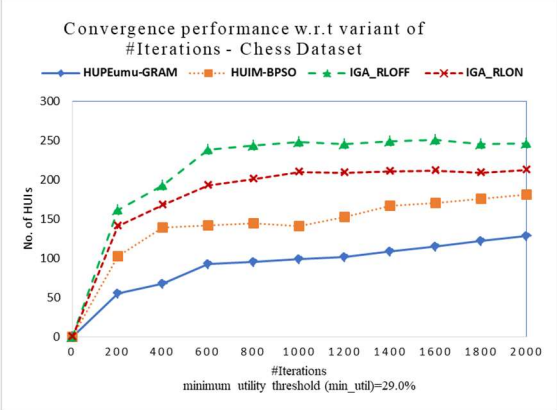

Figure 10 : Convergence speed for Chess dataset

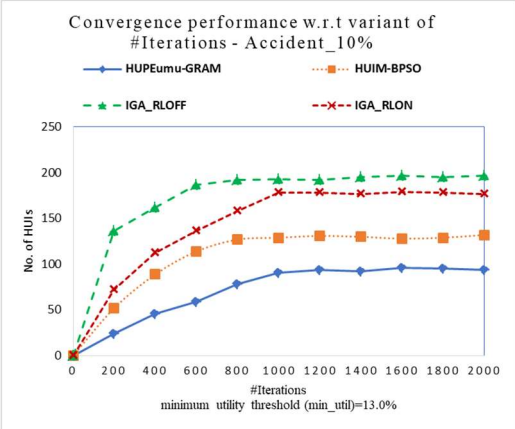

Figure 11 : Convergence speed for Accident_10% dataset

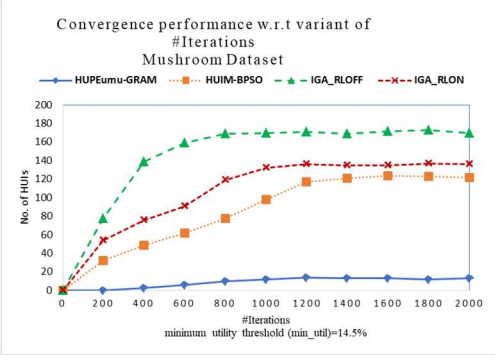

Figure 12 : Convergence speed for Mushroom dataset

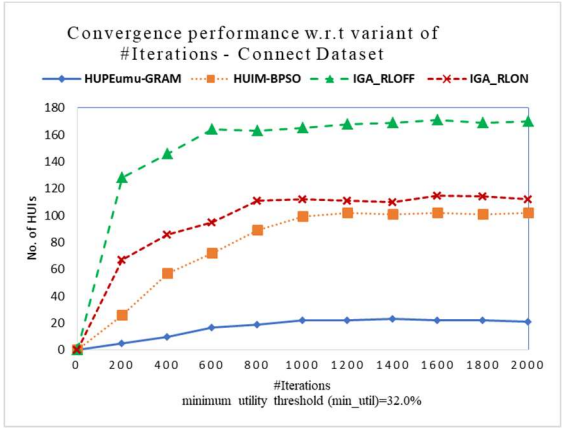

Figure 13 : Convergence speed for Connect dataset

Figure 10 to 13 infers that IGA_RL$_{ON}$ converges at reasonable speed with good number of HUIs when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO. Where as IGA_RL$_{ON}$ converges bit slowly when compared with IGA_RL$_{OFF}$.

4.3. Discovered HUIs

Percent of discovered HUIs using IPSO_RL$_{ON}$ from chess, accident_10%, mushroom and connect dataset is analyzed by comparing it with HUPE$_{umu}$-GRAM, HUIM-BPSO and IPSO_RL$_{ON}$. Graph is plotted using bar chart by varying min_util along x-axis and % of discovered HUIs along y-axis.

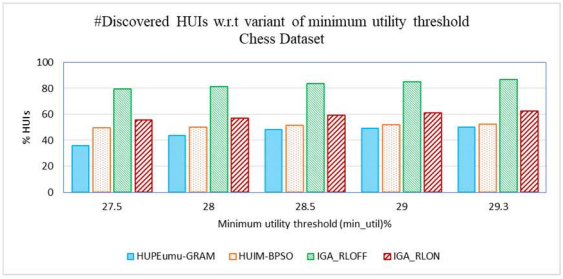

Figure 14 : Percent of discovered HUIs from Chess dataset

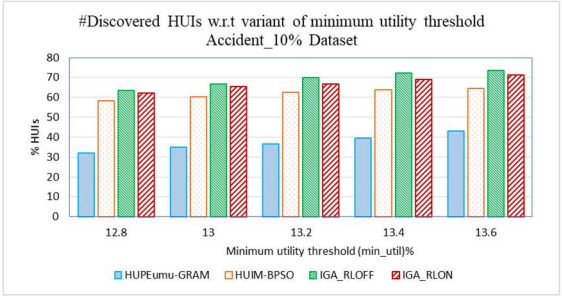

Figure 15: Percent of discovered HUIs from Accident_10% dataset

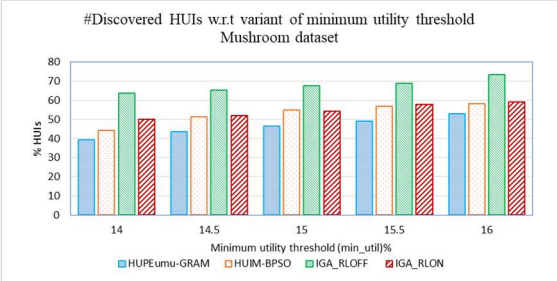

Figure 16 : Percent of discovered HUIs from Mushroom dataset

Inference from the Figure 14 is that percent of discovered HUIs from chess dataset by IGA_RL$_{ON}$ is improved by 29.85% and 15.35% when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO. Also, percent of discovered HUIs by IGA_RL$_{ON}$ is less by 29.10% when compared with IGA_RL$_{OFF}$.

Figure 15 illustrates that, percent of discovered HUIs from accident_10% dataset by IGA_RL$_{ON}$ is improved by 80.14% and 8.25% when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO. Also, percent of discovered HUIs by IGA_RL$_{ON}$ is less by 3.21% when compared with IGA_RL$_{OFF}$.

Observation form the Figure 16 is that, percent of discovered HUIs from mushroom dataset by IGA_RL$_{ON}$ is improved by 18.14% and 3.08% when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO. Also, percent of discovered HUIs by IGA_RL$_{ON}$ is less by 19.12% when compared with IGA_RL$_{OFF}$.

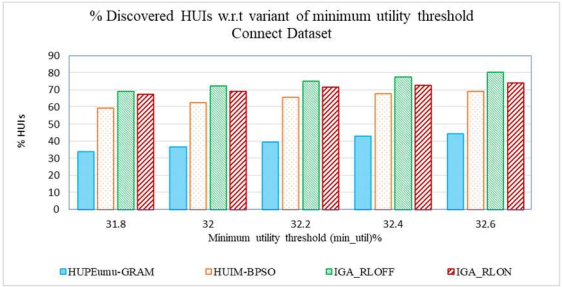

Figure 17 : Percent of discovered HUIs from Connect dataset

Observation form the Figure 17 is that, percent of discovered HUIs from connect dataset by IGA_RL$_{ON}$ is improved by 79.87% and 9.42% when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO. Also, percent of discovered HUIs by IGA_RL$_{ON}$ is less by 5.15% when compared with IGA_RL$_{OFF}$.

5. CONCLUSION

In the current research the fundamental architecture of SARSA, design methodology of the proposed IGA_RL$_{ON}$ approach and its performance was explored. The experimental analysis gives conclusion that IGA_RL$_{ON}$ performs better in terms of execution time, convergence speed and percent of discovered HUIs when compared with HUPE$_{umu}$-GRAM, HUIM-BPSO.

The IGA_RL$_{ON}$ algorithm achieves notable reductions in execution time, with decreases of 12.52% and 4.05% compared to HUPEumu-GRAM and HUIM-BPSO respectively across various datasets. However, it exhibits a slight increase of 6.74% in execution time compared to IGA_RL$_{OFF}$ for mining HUIs.

IGA_RL$_{ON}$ demonstrates competitive convergence speed and generates a significant number of high utility itemsets compared to HUPEumu-GRAM and HUIM-BPSO. Nevertheless, it exhibits a slightly slower convergence rate in comparison to IGA_RL$_{OFF}$.

IGA_RL$_{ON}$ significantly improves the percentage of discovered High Utility Itemsets (HUIs) compared to HUPEumu-GRAM and HUIM-BPSO across datasets, showing enhancements of up to 80.14%. However, it exhibits a lower percentage of discovered HUIs compared to IGA_RL$_{OFF}$, with reductions of up to 29.10%.

The proposed methodology leverages the synergy between GA and SARSA to enhance the efficiency and effectiveness of HUI mining. By integrating reinforcement learning principles into the genetic algorithm framework, IGA_RL$_{ON}$ demonstrates remarkable performance improvements, paving the way for future advancements in this domain.

REFERENCES

[1] M. Han, Z. Gao, A. Li, S. Liu, and D. Mu, "An overview of high utility itemsets mining methods based on intelligent optimization algorithms," *Knowl. Inf. Syst.*, vol. 64, no. 11, pp. 2945–2984, 2022, doi: 10.1007/s10115-022-01741-1.

[2] S. Pramanik and A. Goswami, "Discovery of closed high utility itemsets using a fast nature-inspired ant colony algorithm," *Appl. Intell.*, vol. 52, no. 8, pp. 8839–8855, 2022, doi: 10.1007/s10489-021-02922-1.

[3] S. Guo and H. Wei, "Fast Mining High Utility Itemsets with Multiple Minimum Utility Thresholds Fast Mining High Utility Itemsets with," pp. 0–25, 2022.

[4] K. Logeswaran and P. Suresh, "High utility itemset mining using genetic algorithm assimilated with off policy reinforcement learning to adaptively calibrate crossover operation," *Comput. Intell.*, Nov. 2021, doi: 10.1111/coin.12490.

[5] S. Yacoubi, G. Manita, H. Amdouni, S. Mirjalili, and O. Korbaa, "A modified multi-objective slime mould algorithm with orthogonal learning for numerical association rules mining," *Neural Comput. Appl.*, vol. 0123456789, 2022, doi: 10.1007/s00521-022-07985-w.

[6] S. B. Patel, S. M. Shah, and M. N. Patel, "INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING An Efficient High Utility Itemset Mining Approach using Predicted Utility Co-exist Pruning," vol. 10, no. 4, pp. 224–230, 2022.

[7] R. Kumar and K. Singh, "A survey on soft computing-based high-utility itemsets mining," *Soft Comput. 2021 2613*, vol. 26, no. 13, pp. 6347–6392, Jan. 2022, doi: 10.1007/S00500-021-06613-4.

[8] U. Ahmed, J. C. W. Lin, G. Srivastava, R. Yasin, and Y. Djenouri, "An Evolutionary Model to Mine High Expected Utility Patterns from Uncertain Databases," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 1, pp. 19–28, 2021, doi: 10.1109/TETCI.2020.3000224.

[9] J. M. T. Wu, G. Srivastava, M. Wei, U. Yun, and J. C. W. Lin, "Fuzzy high-utility pattern mining in parallel and distributed Hadoop framework," *Inf. Sci. (Ny).*, vol. 553, pp. 31–48, 2021, doi: 10.1016/j.ins.2020.12.004.

[10] K. Logeswaran, P. Suresh, and S. Anandamurugan, "Particle Swarm Optimization Method Combined with off Policy Reinforcement Learning Algorithm for the Discovery of High Utility Itemset," *Inf. Technol. Control* , vol. 52, no. 1, pp. 25–36, Mar. 2023, doi: 10.5755/J01.ITC.52.1.31949.

[11] J. C. W. Lin, Y. Djenouri, G. Srivastava, U. Yun, and P. Fournier-Viger, "A predictive GA-based model for closed high-utility itemset mining," *Appl. Soft Comput.*, vol. 108, p. 107422, 2021, doi: 10.1016/j.asoc.2021.107422.

[12] C.-W. Lin, T.-P. Hong, J.-W. Wong, G.-C. Lan, and W.-Y. Lin, "A GA-Based Approach to Hide Sensitive High Utility Itemsets," *Sci. World J.*, vol. 2014, no. 1, pp. 1–12, 2014, doi: 10.1155/2014/804629.

[13] J. M. T. Wu, J. Zhan, and J. C. W. Lin, "An ACO-based approach to mine high-utility itemsets," *Knowledge-Based Syst.*, vol. 116, pp. 102–113, 2017, doi: 10.1016/j.knosys.2016.10.027.

[14] L. Zhang, G. Fu, F. Cheng, J. Qiu, and Y. Su, "A multi-objective evolutionary approach for mining frequent and high utility itemsets," *Appl. Soft Comput. J.*, vol. 62, pp. 974–986, 2018, doi: 10.1016/j.asoc.2017.09.033.

[15] W. Song and C. Huang, *Discovering high utility itemsets based on the artificial bee colony algorithm*, vol. 10939 LNAI. Springer International Publishing, 2018. doi: 10.1007/978-3-319-93040-4_1.