Using BTA Algorithm for finding Nash equilibrium problem aiming the extraction of rules in rule learning

M. Boudaghi 1 · F. Mahan 1 · A. Isazadeh 1

Accepted: 12 October 2021 / Published online: 13 November 2021 © The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
It is crystal clear that discovering the rules for finding a specific pattern among given data for extraction of association rules in rule-based learning systems has been defined in previous researches. Making use of game theory for the processes contributing to discovery of rules can be seen in numerous researches. In recent years, modeling based on game theory in rule learning sphere has gained much more attention for computer scientists. When two or more players use different strategies independently, the strategy game modeling could be used. In this view, strategic play is a desirable model for situations with no permanent strategic relationship among interactions. In addition, Nash equilibrium is the most widely used solution concept in game theory. This concept is a state-of-the-art interpretation of a strategy game. Each player has an accurate prediction of other players’ behavior and acts according to such a rational prediction. In the present study, by extracting rules from frequent patterns we have presented a model that can extract learning rules by abstraction based on game theory, which can be used not only for association rules but also for rule-based learning systems. Also, the introduced method can be easily generalized to fuzzy data. To find Nash equilibrium (FNE) in the proposed method, we used meta-heuristic bus transportation algorithm. The results indicated that the method reduces computational complexity in the associate rule discovery process and rule learning, provided that FNE is solved.

Keywords Rule learning · Game theory · Nash equilibrium · Meta-heuristic bus transportation algorithm

1 Introduction
Rule learning methods are popular techniques among data mining and machine learning methods with applications in expert systems, controller systems (especially fuzzy controller systems), describing economic processes and predicting related efficient strategies, distributed systems (especially cloud computing systems), and informatics for prediction and detection of different biological structures. Generally speaking, rule learning methods are efficient in any given systems dealing with rule-based strategies (Theocharopoulos et al. 2017; Iancu and Gabroveau 2010; Millette 2012; Bell 2020).

All rule-based systems use the “if ... then” form to express rules. Therefore, this form is always used for finding the rules of frequent patterns in specific set of data. There is a clear difference between the point of view or the purpose of using rules dealing with specific set of data. The two general purposes for using learning rules in discovering frequent data patterns are to discover learning rules for describing and also predicting given data (Fürnkranz and Kliegr 2015; Novak et al. 2010; Carmona et al. 2018). Discovering the rules in rule learning to describe data means finding a pattern among frequent data. Data prediction rules in rule learning include the extraction of a set of rules that cover frequent data and the entire sample space, making it possible to predict each sample in the sample space.

In discovering data description rules, the focus is on the statistical validity of rules not on predicting the data. In this type of rule learning—in supervised learning mode—the goal is to discover subgroups when the desired characteristics exist. Because the desired relationship between items is explored—in unsupervised rule learning mode—the goal is to find associative rules (Novak et al. 2009; Fürnkranz et al. ...
2012; Triantaphyllou and Felici 2006). When rule learning aims at predicting the data, in practice, the training information is generalized so that predictions for new examples become possible. Therefore, when the entire sample space is being searched, it does not matter what the computational complexity is. In rule learning concerning the description of data, only the frequent part of the training data is being monitored to reduce computational complexity through new methods and algorithms.

So, there exist two fundamental challenges facing discovering rules in rule learning, namely to say the introduction of an algorithm that can significantly reduce the computational complexity and generalizability of the same algorithm to fuzzy data. In the present study, based on the first goal of rule learning, i.e., the discovery of rules to describe data, a method is proposed that uses game theory abstraction to extract rules by reducing computational complexity. This method can be generalized to fuzzy data in addition to factual data. The contribution of the present study includes the following issues:

1. Changing the abstraction of the problem related to the learning rule extraction in rule learning systems into game theory.
2. Proposing an efficient Payoff function contributing to the reduction of the rule extraction computational complexities.
3. Reducing the Nash equilibrium in rule extraction game’s rule learning systems to 0–1 IP (integer programming) and solving the 0–1 IP with BTA (bus transportation algorithm) (Bodaghi and Samieefar 2019).
4. Extracting learning system’s super rules and extracting the same system’s rules according to obtained super rules.
5. Putting the computational complexity of the proposed method in comparison with the previous abstractions.
6. Presenting the empirical results of the proposed method for extracting rules with hypothetical data.

Section 2 elaborates on previous research concerning the improvement of rule learning algorithms. Section 3 explains proposed method in five subsections: the first subsection, i.e., Sect. 3.1, is about data preparation resorting to abstract game theory; the second subsection, i.e., Sect. 3.2, refers to the calculation of player Payoff by abstracting game theory. The subsection introduces a new model for calculating players’ (data generation resources) Payoff; the third subsection, i.e., Sect. 3.3, covers the solution for created game d by finding the game’s equilibrium points; the fourth subsection, i.e., Sect. 3.4, reduces the NEP to the zero–one integer programming (0–1 IP) problem; and the fifth subsection, i.e., Sect. 3.5, describes how to solve the 0–1 IP problem through the BTA. The fourth section, i.e., Sect. 4, discusses implementation details and introduces the results along with hypothetical data.

2 Related works

Recently, there have been various approaches regarding rule learning. Even more recently, parallelization methods have been introduced for optimization of the algorithms related to these approaches (Zhu et al. 2021), improvement of the Apriori algorithm (Huiqi 2020), methods for fuzzy rule learning systems (Cózar et al. 2018), or algorithms for improving rule-based classification performance (Liu and Chen 2019). The origin of all these new methods is the discovery of association rules. The recurrence of association patterns based on frequent patterns is evaluated according to Support and Confidence criteria. Rule detection algorithms such as Apriori and FP-growth and their improved variants use these two criteria to detect frequent patterns. Algorithms that use these two criteria to evaluate the resulting rules may undergo many repetitive and useless calculations. However, in improved versions of rule discovery algorithms significant efforts have been made to reduce computational complexity. In non-improvement mode (classical mode), in the Apriori algorithm the computational complexity is $O(2^n) + O(2^k)$ (Telikani et al. 2020); in FP-growth algorithm, the computational complexity is $O(n \times n)$. Recent research has improved computational complexity for the Apriori algorithm, especially regarding massive data environments (Yu 2020) and the FP-growth (Shabtay et al. 2021) algorithm.

Many algorithms are used to discover and extract the rules in rule learning systems. They have progressed over time, but the traditional methods used for discovering the rules for finding frequent patterns can be considered as subsets of the two Apriori and FP-growth algorithms. The FP-growth algorithm operates in its search without frequent data production. Therefore, it does not have the weakness of the Apriori algorithm, but even the optimized versions of Apriori algorithm ultimately fail to reduce the computational complexity (Lin et al. 2011; Yin et al. 2018; Zeng et al. 2015) significantly. Although scholars have conducted adequate research to generalize the FP-growth algorithm for fuzzy data (Sabita et al. 2010; Hoque et al. 2015; Wang et al. 2017), the development of these algorithms for fuzzy data also has certain complexities. Since the 1990s, much effort has been made to reduce these algorithms’ computational complexity. However, in spite of available big data we still face major complexities in critical cases (Ai et al. 2018; Vasoya and Koli 2016; Ait-Mlouk et al. 2017). The methods being used to improve the Apriori algorithm can be divided into the following categories:
1. Some methods make use of specific data structures (Singh et al. 2015; Soysal et al. 2016; Wang and Zheng 2020). For example, one of the most popular of these methods is the use of hashing function and a file with direct access to store items $k > 1$ in buckets where the amount of support is calculated based on buckets rather the individual data. Hence, access becomes easy to frequent patterns (Rathinasabapathy and Bhaskaran 2009).

2. There are methods based on the elimination of data generation sources that do not produce frequent patterns. These data sources are like transactions that contain items which have not to be repeated in any other transactions. Thus, the discovery of such sources of data generation can reduce the search space of scenarios. Some of these methods are available in Yuan (2017), Cheng et al. (2015), Palshikar et al. (2007), and Vijayalakshmi and Pethalakshmi (2015) research.

3. There are methods that allow parallel processing in big data environments. These methods generally divide the state space into different parts and define a value as a \( \text{minsup} \), i.e., the minimum support value. The minimum support value of each part of the state space is considered a condition (this value must be equal to \( l > 1 \)); other values less than \( l > 1 \) are not considered. From this value, the final rules are extracted. Of course, these methods usually do not pose superiority in computational complexity because they only ensure the results. As such, it is necessary to scan the entire database after extracting the patterns (Saabith et al. 2016; Xiangyang and Ling 2016; Gan et al. 2019).

4. The use of sampling techniques in big databases is another method that has led to increased Apriori efficiency. Examples of these techniques can be seen in Chakaravarthy et al. (2009) and Thakur and Ninoria (2017) research.

5. There are also studies that generalize the Apriori algorithm to fuzzy data which have preserved the algorithm’s traditional abstraction. These methods face increasing computational complexity (Sowan et al. 2013; Mangalampalli and Pudi 2009; Pierrard et al. 2018).

6. Recently, the use of machine learning techniques to discover frequent patterns in the Apriori algorithm has become commonplace, given that this algorithm uses the knowledge of the previous stage in the following one (Bhagat et al. 2010). Combined techniques such as PSO–Apriori, GA–Apriori, and other hybrid algorithms are used to discover frequent patterns in the Apriori algorithm (Djenouri and Comuzzi 2017).

7. In recent years, the game theory has been used in learning techniques related to the Apriori algorithm’s level-to-level strategy. However, the basis of these methods is still the traditional Apriori algorithm. Hence, in some cases, game theory has reduced the search space and increased the Apriori algorithm’s efficiency (Wang 2006; Miyaji and Rahman 2011).

8. Recently, a method of combining fuzzy logic and deep neural networks has been proposed (Asghar et al. 2020), which seems to be effective for extracting rules from repetitive patterns in a rule learning system. Although this method is designed to measure customer satisfaction on the web, the idea can contribute to efficient extraction of rules with high confidence.

Limitations of the previous studies could be summarized as follows:

1. The major drawback of the rule extraction methods introduced so far is their high computational complexity. Although efforts have been made to improve their computational complexity, the limitations of the abstraction chosen by these methods have prevented from significant improvements in recent research in this area.

2. In studies that have made use of game theory, the abstraction of the game theory has been utilized as part of the supposed method. For instance, the main abstraction of the method has been graphs or probabilities, and in part of the method, the game theory has been used to improve the method. The problem with such research is that the game theory’s abstraction capacity is not fully exploited. For example, in these studies, there is no need for an efficient Payoff function, and therefore, we will not see a significant improvement in the old abstraction.

3. Any reduction in the search space of the problem must not lead to ignoring the items of the problem that should have been used to extract the rules of learning. This has often been overlooked in previous research and therefore reduced the confidence of the obtained rules.

The chronological review of the rule extraction methods in rule learning systems is summarized in Table 1.

So far, there are not any given methods that extract the rules based on the discovery of frequent patterns from the abstraction of the game theory from the beginning to the end of the algorithm. In the present study, we show how utilizing this abstraction can reduce rule discovery’s computational complexity.

### 3 Materials and methods

During the past two decades, the game theory abstraction has focused on data mining activities including rule learning (Narahari 2010; Stahl 2000). For example, data mining activity can be modeled as a “non-cooperative game” performed by multiple players. A game defines players, possible acts for each player, and the Payoff of the players’ actions. If no player...
| Researcher and year | Case study | Abstraction and methods | Scope | Contribution |
|---------------------|------------|-------------------------|-------|--------------|
| Piatetsky-Shapiro (1991) | Discovery, analysis, and presentation of strong rules | Discovering efficient rules in databases using probability abstraction to analyze different criteria of attractiveness | High computational complexity in the vast search space, providing the same analysis for different data, not providing a solution to reduce the search space in database items, not being generalized to fuzzy data | Development of the Apriori algorithm with probabilistic abstraction |
| Agrawal and Srikant (1994) | Fast algorithms for mining association rules in large databases | Utilizing probability abstraction to learn association rule learning, which until then was an efficient and convenient way to find attractive and often unseen relationships between variables in a large database | High computational complexity in the vast search space, providing the same analysis for different data, not providing a solution to reduce the search space in database items, not being generalized to fuzzy data | Development of the Apriori algorithm with probabilistic abstraction |
| Durkin and Durkin (1998) | Expert systems: design and development | In expert systems, using the propositional logic and the propositional calculus system, knowledge discovery is done through the rules that the tools of propositional logic provide to this system | High computational complexity in wide search space and NP, lack of mechanism to control computational complexity by reducing the search space in database items, complex mechanism in parallel implementation, lack of generalization capability to fuzzy data | Development and expansion of abstraction based on propositional logic |
| Han et al. (2000) | Mining frequent patterns without candidate generation | Graph-based abstraction, which divides the database into a set of databases, each with a duplicate item, and explores each database separately | High computational complexity in wide search space, lack of mechanism to control computational complexity by reducing the search space in database items, lack of ease in implementation, and complexity in parallel implementation, lack of generalization capability to fuzzy data | Development of FP-growth algorithm with graph based abstraction |
| Wang (2006) | Intelligent manufacturing strategy selection | In this method, strategic game modeling was used to discover frequent patterns. The method assumes that the action of each agent in a game is based on predetermined and defined strategies. The set of actions of each agent is quite clear against the other agent(s), and the action of each agent can affect the actions of other agents | Although it seems that the abstraction of game theory is used, without defining the Payoff function, a practically incomplete abstraction of game theory is used in this method, which does not reduce the computational complexity. Lack of ease in implementation, complexity of parallel implementation, lack of capability in generalization to fuzzy data | Making use of game theory in extraction of rules and its use to improve the two algorithms of Apriori and FP-growth and other algorithms for discovering frequent patterns to extract related rules |
| Researcher and year | Case study | Abstraction and methods | Scope | Contribution |
|---------------------|------------|-------------------------|-------|--------------|
| Miyaji and Rahman (2011) | Privacy-preserving data mining: a game-theoretic approach | Utilizing game theory to improve the two algorithms of Apriori and FP-growth and other algorithms for discovering frequent patterns in order to extract rules | Because the basic abstraction in these methods is the same as probabilistic and graph abstractions, we do not see a significant reduction in computational complexity. Parallel implementation complexity, no generalization to fuzzy data | Making use of game theory in extraction of rules and its use to improve the two algorithms of Apriori and FP-growth and other algorithms for discovering frequent patterns to extract related rules |
| Isazadeh et al. (2014) | ECA rule learning in dynamic environments | Using fuzzy tree-based abstraction and neural network, they presented a method for responding to event-condition-action (ECA) in dynamic environments that require automatic response | High computational complexity in large dynamic contexts, parallel implementation complexity | Extraction of rules in rule learning systems in dynamic environments |
| Cózar et al. (2018) | Learning compact zero-order TSK fuzzy rule-based systems for high-dimensional problems using an Apriori + local search approach | A method based on fuzzy logic | High computational complexity in dynamic contexts, parallel implementation complexity | Extraction of fuzzy rules for Rule Learning systems |
| Zhu et al. (2021) | Parallel multipopulation optimization for belief rule base learning | Making use of a parallel population algorithm to derive the rules of rule based | High computational complexity in large dynamic contexts, and parallel implementation complexity in the same spaces | Extraction of parallel rules for rule based systems |
| Huqi (2020) | Improvement parallelization in Apriori algorithm | Possibility of parallel implementation of Apriori algorithm without changing the abstraction of the algorithm | High computational complexity and implementation in large dynamic contexts and lack of generalization to fuzzy data | Improved Apriori algorithm for parallel implementation |
deviates from their strategy, a Nash equilibrium is achieved. Recently, this kind of view or to be more precise, this type of mathematical modeling based on game theory has become very popular in data mining, so that research on new methods in data clustering, data classification, data pattern extraction, or data prediction models has been extended (Durlauf and Blume 2010). For the first time in his research, Stahl proposed a method based on game theory and Nash learning for data prediction learning rules (Stahl 1997; Wang 2006). His method was based on logical behavior which has a probabilistic basis. This study strongly rejected Nash learning because of the probabilistic basis of logical behaviors. However, it introduced the rule’s extraction based on a game by distributing the probabilities in the game’s actions in a hypothetical game. Pham et al. proposed a method based on game theory for extracting association rules. In this research, each player’s action in a game is assumed according to predetermined and defined strategies. The set of actions of each player against another player is quite clear, and each player’s action can effect other players’ actions. Finally, any given set of association rules of the studied system is extracted from individual player’s practical actions. Because the method presupposes actual behavior instead of logical behavior, it can consider Nash equilibrium as a game answer designed to learn data prediction rules (Wang et al. 2006). As noted before, a definite Nash equilibrium cannot easily be found for rational behavior that has a probabilistic basis, especially for modeling data learning rules. Because a certain space has a wide range of strategies with different probabilistic distributions, and the probability distribution update at each round, the Nash equilibrium in these systems falls into the total find non-deterministic problem (TFNP) computational complexity class (Papadimitriou 2015). Nash equilibrium can also be reduced to the 01IP (Wu et al. 2014). Our main idea is to find rules that cover the most frequent examples. This idea can be modeled based on non-cooperative game. Concerning the Nash equilibrium concept, it is possible to find a method that can be systematically implemented to find rules that cover a larger set of instances. The equilibrium point in the designed game is where the search to find the rules ends. Because finding the Nash equilibrium point falls in the TFNP complexity class, we used the BTA. But before taking action to use the BTA, we reduced the Nash equilibrium to 01IP problem. Figure 1 shows the phases of rule extraction in a system based on rule learning according to the present study’s proposed method.

Each of these phases will be explained in detail in the following sections.

3.1 Preparing data

This section explains how to prepare data for game abstraction. The non-cooperative game is a kind of strategy game. A strategy game is a model of interactive decision making. Each decision maker chooses its movement plan once and for all, and the choice is simultaneous. A strategy game includes:

1. A finite set \( N \) (set of players).
2. An infinite set \( A_i \) for each player \( i \in N \) (set of available movements for player \( i \)).
3. A priority function \( >_i \) on \( A = \times_{j \in N} A_j \) for each player \( i \in N \) (priority relation of player \( i \)). If the set of movements of each player \( i (A_i) \) is finite, we call the game finite.
4. Define set \( C \) of results as a function \( g : A \rightarrow C \) where this function is related movements to its results, and we define it \( >_i^g \) of priority relation on \( C \). The priority relation \( >_i^g \) for each player \( i \) is defined as \( a >_i^g b \) if and only if \( g(a) >_i g(b) \).

Having a player (any entity that produces data) forms the matrix of strategies (players). Given that each of the game agents (players) can adopt different strategies, we will use the
extensive form to represent the present study’s main game in a tree structure. The game’s tree structure is a graphical representation of a stage game that provides information about players, final results, modes, and actions. The game’s tree comprises some vertices indicating the nodes that players can act on it, and these nodes connect by an edge, which denotes the actions that may be performed on a specific node. The first node (or root) indicates the first decision that has been made. Each set of edges is considered as branches of the first node moving across the tree to the final node, representing the game’s end. Each final node is tagged with each player’s final result if the game ends in that node. Using a tree form to represent problem space for games is often appropriate. Root node includes game start point. For each node containing the current status, a decision must be made to select the best next action. A branch of the tree indicates each legible action. By applying an evaluation function, the status of the game is evaluated. Leaf knots represent the game’s final state where one of the values can be a win, a draw, or a loss. In our proposed method, the status of game tree structure is constructed by defining \( n \) default states (by the number of players). The tree’s root in each default state starts with one of the players and expands over time until it reaches the leaf. Each node is an agent (player) that selects a behavior (set of strategies) from among the possible behaviors that come out of the node as a downward edge. Static behavior contributes to lack of change in strategies. We consider the path from the root to each node as that node’s strategy (player). Tabs are nodes that sum up the values of the combination of operating strategies. The critical point is that each player has a combination of predefined strategies at the beginning that moves from the root node to the entire set of the same strategies. This is because there may be very significant connections in datasets from the beginning that, without calculations, it is possible to discover the rules among the same datasets. The data are defined so that the Payoff of the combination of strategies increases along with it, and the final formed rules avoid duplicate calculations. Depending on the choice of roots, a different tree structure is created, which will vary depending on the number of strategic combinations of players. However, they are equal in the number of edges and nodes. If the number of players in the set \( P = \{ P_1, P_2, \ldots, P_n \} \) is equal to \( |P| \) and the strategy Payoff for each player \( i \) is \( \alpha_i \), so that \( \sigma = \{ \alpha_1, \alpha_2, \ldots, \alpha_k \} \) is the set of strategy composition Payoff of all the players in the combination form and each tree \( T_i \) so that \( T = \{ T_1, T_2, \ldots, T_m \} \) is the set of trees in the whole game where \( m = |P|! \) shows the number of game trees depending on the number of players. The time complexity of the game in each tree will be \( O(P^n) \), and the complexity of the entire game space is at least \( O(P!) \). In contrast, if the combination of default strategies based on the basic strategies of each player according to the number of combined strategies of each player reduces the computational complexity of the whole game can be reduced to a maximum of \( O(P) \).

### 3.2 Calculating the amount of Payoff

This section explains how to calculate the amount of Payoff via combining possible strategies for each player. We define game \( \vartheta \) based on the structure of the game tree to discover rules in rule learning as follows:

\[
\vartheta : \langle G, p, \sigma, \Gamma, E \rangle.
\]

\( P \): is a set of players or agents that in the present study’s game has \( n \) players whose behavior, as mentioned in the previous sections, is defined based on the set of data they produce. Therefore, it can be said that:

\[
P = \{ P_1, P_2, \ldots, P_n \}.
\]

Usually, the data vertices of the graph \( V \) or a subset of them are introduced as game agents, i.e., \( P \subseteq \langle V \rangle \). If we select a subset of nodes as game agents, the selection of agents from graph nodes can be introduced as a free parameter in this model. If we assume that \( v_i \) nodes in the game graph are defined as \( G_p \), then the vector \( \theta_p \) represents the membership of nodes within the set of game agents, and then, we have:

\[
\theta_p(v_i) = \begin{cases} 
1 & v_i \in P \\
0 & v_i \notin P
\end{cases}
\]

\( \sigma \): At each stage, players can choose their behavior from a specific set (a set of data generated by them). If we assume that player \( P_i \) has a permissible behavioral range \( \sigma_i \), then we represent and will have a set of these permissible behaviors for all agents by \( \sigma \):

\[
\sigma = \{ \sigma_1, \sigma_2, \ldots, \sigma_k \}.
\]

The decision about what behaviors each player is allowed to adopt can also be considered as a free parameter. If we assume that all acceptable behaviors in the game memory in set \( \sigma_{\text{Allowed}} \), then:

\[
\sigma_{\text{Allowed}} = \{ \sigma_{\text{allowed1}}, \sigma_{\text{allowed2}}, \ldots, \sigma_{\text{allowedk}} \}.
\]

And the matrix \( \theta_\sigma \) with size \( |P| \times k' \) will be equal to:

\[
\theta_\sigma(i, j) = \begin{cases} 
1 & \sigma_{\text{allowed}} \in \sigma_i \\
0 & \sigma_{\text{allowed}} \notin \sigma_i
\end{cases}
\]

\( \Gamma \): As mentioned before, each game node’s strategy is equal to each player’s sequential behavior originating from root to that node. If we assume that the adopted strategy in
node $\chi$ memory in the set $\mathcal{Y}_\chi$, then the Payoff of $P_i$ in node $\chi$ is $\Gamma_i(\mathcal{Y}_\chi)$. For each player, we can set up coefficients as free parameters independent of single variable agents. If in the function $\Gamma_i$ there is $q$ free parameter, then matrix $\theta_{parameter}$ with value of $|P| \times q$ will be defined as:

$$\theta_{parameter} = [\theta_{parameter}(i, j)]_{P \times q}.$$  

(7)

$\theta_{parameter}(i, j)$ is $j$th coefficients in the function $\Gamma_i$ for $i$th player; this parameter is a weight for a player. Thus, the function $\Gamma_i$, in addition to the adopted strategy, also depends on matrix $\theta_{parameter}$, and the profit of the player $P_i$ in the node $\chi$ will be equal to:

$$\Gamma_i(\mathcal{Y}_\chi, \theta_{parameter}(i, :)).$$  

(8)

$\Gamma$ is the vector of total players Payoff, which is obtained from their strategies. So:

$$\Gamma = (\Gamma_1, \Gamma, \ldots, \Gamma_n).$$  

(9)

Like most frequent pattern algorithms, using ($Support, s(X \rightarrow Y) = \frac{\sigma(X \cup Y)}{\sigma(X)}$), the support value of each data set (item) is extracted, and then, the average of all supports is calculated and called $Supp_{ave}$. In player i’s strategy set expressed as $A_i = (a_1, a_2, \ldots, a_k)$, and if the j'th strategy is in the form of $a_j \geq Supp_{ave}$, we call it $\phi_j$, otherwise we call it $\psi_j$. The value of $\Gamma_i$ for each player $i$ is calculated as follows:

$$\Gamma_i = \frac{\left| \sum_{j=1}^{k} \phi_j - \sum_{j=1}^{m} \psi_j \right|}{|P| \times \sum_{k=1}^{P} |A_k|},$$  

(10)

where $n$ is the number of $\phi$ in the set $A_i = (a_1, a_2, \ldots, a_k)$ and $m$ is the number $\psi$ in the same set. Also $|P| \times \sum_{k=1}^{P} |A_k|$ is the total number of data generated by all players. If $\sum_{j=1}^{n} \phi_j = 0$, then $\Gamma_i = 0$, and if $\left( \sum_{j=1}^{n} \phi_j - \sum_{j=1}^{m} \psi_j \right) = 0$ (i.e., $\sum_{j=1}^{n} \phi_j = \sum_{j=1}^{m} \psi_j$), then:

$$\Gamma_i = \frac{\sum_{j=1}^{n} \phi_j}{|P| \times \sum_{k=1}^{P} |A_k|} \quad \text{or} \quad \Gamma_i = \frac{\sum_{j=1}^{m} \psi_j}{|P| \times \sum_{k=1}^{P} |A_k|}.$$  

(11)

The critical point is that this method unlike conventional rules extraction methods has a strong abstraction for fuzzy data and can be easily generalized to this type of data. For fuzzy data, in addition to $\Gamma_i$, it is sufficient to use the membership degree $\mu_{A_i}(a_v)$, which indicates the membership of the element $v$ from the fuzzy set $\tilde{A}_i = (a_1, a_2, \ldots, a_k)$, i.e.,

$$a_v \in \tilde{A}_i = (a_1, a_2, \ldots, a_k).$$

Finally, defuzzification methods can be used for $(\Gamma_i, \mu_{A_i}(a_v))$ sets.

$E$: The condition for stopping the game is set by rule $E$ because, in big games, the number of game stages can vary. When the balance is achieved, agents are reluctant to change their strategy. Nash equilibrium is the most well-known equilibrium in the game theory model, in which changing each player’s strategy reduces that player’s Payoff interest.

### 3.3 Eliminating the combination of redundant strategies

In this section, we explain how to determine the amount of $\minsup$ and eliminate the combination of strategies that are less than this amount. Priority relation $\succ_i$ for player $i$ in a strategy game with a Payoff function of $u_i : A \rightarrow R$ (also called the utility function) is shown when $a \succ_i b$ then $u_i(a) \geq u_i(b)$. The values of such function are called Payoff (or benefits). If the relation between a player’s priorities and the efficiency function it represents is clear, we will use the $(\langle N, (A_i), (\succ_i) \rangle)$ sign instead of $(\langle N, (A_i), (u_i) \rangle)$ to define the game components. If we specify an arbitrary $\minsup$ value for the Payoff function (e.g., $k_{u_i}$), then it is evident that it is always $u_i < k_{u_i}$. Therefore, the game’s components to be included in the rule extraction process in the present study are defined as $\langle N, (A_i), (u_i) < k_{u_i} \rangle$.

### 3.4 Reducing FNE to 0–1 IP

In this section, we explain how to reduce the problem of FNE based on the remaining “combination of strategies” from the previous phase to the 0–1 IP problem. Nash equilibrium is the most widely used solution concept in game theory. The concept is an interpretation of a state-of-the-art space of a strategy game. Each player has a accurate prediction of other players’ behaviors so that acts on this rational prediction. In this sense, there is no attempt to test how this state-space creates. A strategy game equilibrium $\langle N, (A_i), (u_i) \rangle$ is a representation $a^* \in A$ of possible moves that for all actors $i \in N$ has the following property for all $a_i \in A_i$:

$$(a_{i-1}^*, a_i^*) \succ_i (a_{i-1}^*, a_i^*).$$  

(12)

Therefore, for $a^*$ to be a Nash equilibrium, it must not be possible for player $i$ to make a move that results in a better outcome than the selection of $a^*_i$, and each player $j$ must also choose its equilibrium action $a^*_j$. In short, no player can make a lucrative move against the given moves of others. In some open problems, the formulation of this definition 13 is also useful. For each $a_{i-1} \in A_{i-1}$, $B_i(a_{i-1})$ can be defined as
Using BTA Algorithm for finding Nash equilibrium problem aiming the extraction of rules... 447

Each finite N-player game with mixed strategy has at least one Nash equilibrium point. Strategies \((n_1^*, \ldots, n_N^*)\) with \(n_i^* \in M_i\) for \(i \in N\) form an equilibrium solution if:

\[
\begin{align*}
\sum_{j \in N} a_{n_1^*, n_2^*, \ldots, n_N^*} & \geq \sum_{j \in N} a_{n_1, n_2^*, \ldots, n_N^*}, \forall n_1 \in M_1 \\
\sum_{j \in N} a_{n_1^*, n_2, \ldots, n_N^*} & \geq \sum_{j \in N} a_{n_1^*, n_2, \ldots, n_N}, \forall n_2 \in M_2 \\
& \cdots \cdots \\
\sum_{j \in N} a_{n_1^*, n_2^*, \ldots, n_N^*} & \geq \sum_{j \in N} a_{n_1^*, n_2^*, \ldots, n_N}, \forall n_N \in M_N.
\end{align*}
\]

(14)

Two \(N\)-player games with the results functions \(a_{i,n_1,n_2,\ldots,n_N}^i, b_{i,n_1,n_2,\ldots,n_N}^i\) are called equivalent strategies if there are \(\alpha_i > 0\) and the number \(\beta_i\) for \(i = 1, 2, \ldots, n\) such that:

\[
a_{i,n_1,\ldots,n_n} = \alpha_i b_{i,n_1,\ldots,n_n} + \beta_i, \forall i \in N.
\]

(15)

Nash solution with mixed strategy: The mixed strategy of \(y^{x_1}, y^{x_2}, \ldots, y^{x_N}\) with \(y^{x_i} \in \Theta M_i\) for \(i \in N\) forms an equilibrium solution if:

\[
\begin{align*}
\sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N} a_{n_1 n_2 \cdots n_N}^1 & \geq \sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N}^1 a_{n_1 n_2 \cdots n_N}, \forall y^1 \in \Theta M_1 \\
\sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N} a_{n_1 n_2 \cdots n_N}^2 & \geq \sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N}^2 a_{n_1 n_2 \cdots n_N}, \forall y^2 \in \Theta M_2 \\
& \cdots \cdots \\
\sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N} a_{n_1 n_2 \cdots n_N}^N & \geq \sum_{n_1} y_{n_1} y_{n_2} \cdots y_{n_N}^N a_{n_1 n_2 \cdots n_N}, \forall y^N \in \Theta M_N,
\end{align*}
\]

(16)

then 17 is a Nash equilibrium for the finite \(N\)-player game that has a mixed strategy.

\[
\begin{align*}
\psi^1_{x_1}(y^1, y^2) &= \left[ y^1_1 y^2_1 a_{11}^1 + y^1_2 y^2_2 a_{12}^2 + y^2_1 y^2_2 a_{21} + y^2_1 y^2_2 a_{22} \right] \\
& - \left[ y^1_1 a_{11}^1_1 + y^2_1 a_{11}^2_1 \right] \\
\psi^i_{n_i}(y^1, \ldots, y^N) &= \min \left\{ \psi^1_{n_i}(y^1, \ldots, y^N), 0 \right\} \\
y^{i,j}_{n_i} &= \frac{y^i_{n_i} + c_i}{1 + \sum_{j \in N} c_j}, \\
y^{i,j}_{n_i} &= y^i_{n_i}, \forall n_i \in M_i, \text{ and } \forall i = 1, \ldots, N.
\end{align*}
\]

(17)

One of the problems with non-cooperative game theory is that there are usually more equilibria for different outcome vectors. Another problem is that even if there is a single strategic Nash equilibrium, it may not be considered a logical response or a predictable Payoff. These problems occur when rule learning is modeled as a non-cooperative game. Therefore, we need an algorithm that detects all Nash equilibria so that we can logically evaluate the obtained equilibria to conclude that the obtained rules (at the equilibrium point) are valid. Therefore, our idea is to change the defaults of the BTA to meet these two goals.

### 3.5 Solve 0–1 IP with BTA

Now, we solve the 0–1 IP problem of the previous phase with the BTA. The Nash equilibrium problem is an NP-hard problem, but a particular case of finding Nash equilibrium for a two-player game can be reduced to an NP-complete problem, 0–1 IP, and one can solve this problem by BTA proposed in the present study. Regarding the reduction of the Nash equilibrium problem to the 0–1 IP problem, it is sufficient to adopt the following conditions:

Mixing strategy \((x^*, y^*)\) with a double matrix \((A, B)\) is called a Nash equilibrium solution if and only if in problem 18, \(x^*, y^*, p^*, q^*\) are hold 17. This equilibrium under the following conditions will be a 0–1 IP problem, which is a particular problem 17.

\[
\begin{align*}
\max_{x, y, p, q} \left\{ x A y^T + x B y^T - p - q \right\} \\
A y^T & \leq p e_m \\
B^T x^T & \leq q e_n \\
x_i &= 0 \text{ or } 1, \forall i \\
y_j &= 0 \text{ or } 1, \forall j
\end{align*}
\]

(18)

Thus, the Nash equilibrium problem can be quickly reduced to a 01IP problem.

It has already been pointed out that the present study suggested use of the BTA for solving the 01IP problem. However, BTA researchers believe that with modifications we can use BTA for continuous functions. This algorithm has a particular abstraction that includes buses, stations, and passengers. To use this algorithm in any search query, buses, stations, and passengers must be at first identified. The problem-solving method’s main idea is implementing the BTA by loading and unloading passengers, which continues until the optimal answer is achieved. Stations are auxiliary memories that help us simplify and organize the problem at each phase, and as mentioned before, they provide an opportunity to evaluate the answers obtained. Passengers (variables zero or one) move among the stations in question to reach the stations we want (optimized answers); this shifting is based on learning, which
overssees how individuals and groups function. The variables of the problem are intelligent travelers. As intelligent agents they have individual intelligence and try to optimize the problem’s objective function in interaction with other factors and create collective intelligence. Although the number of stations can be unlimited, each station belongs to one of the following groups of stations:

1. Stationary Station Group (SSG): No doubt the variables in this groups possess certain values.
2. Short-Term Station Group (STSG): The variables of this group do not have specific values. The variables entering this station remain in the problem, while the other variables would stay out of the problem.
3. Medium-Term Station Group (MTSG): This group includes those variables that reside somewhere between indefinite and approximately definite states. Variables either return to short-term stations or go to long-term stations through buses.
4. Long-Stage Stations Group (LSSG): In this group, the variable are in a balanced state to some extent, and probably, the accuracy of the variables is high. As such, they might return to the middle stations based on future experiences.

There are at least two stations in each station group:

1. The station that the variable’s value becomes zero and it is loaded on the next bus.
2. The station that the variable’s value becomes one and it is loaded on the next bus.

Each bus is a processing system that contains several variables and transports them to different stations. So increasing the number of buses will mean running a separate bus algorithm that can be implemented with parallel programming. The number of stations and each agent’s memory (variable), and the collective memory are fixed numbers. In the BTA, to improve the problem, a heuristic function will be added to the variables’ intelligence so that the variables tend to a local optimization more quickly. Random selection between two modes of using heuristic function or the best solutions found in the group experience matrix proposes avoiding local optimization.

The proposed heuristic function for each variable $j$ is introduced as follows:

$$ F_j = \alpha \left( \frac{c_j}{\sum_{l=1}^{n} c_l} \right) - \beta \left( \max_j \left( \frac{a_{ij}}{b_{ij}} \right) \right). $$

(19)

$\alpha$ and $\beta$ are intelligent numbers in the interval $[0, 1]$ determined by moving to achieve the local answer. We also convert the following function for converting $F$ to a vector with numbers in the interval $[0, 1]$ because the answer of the above interval lies in the region $[-1, 1]$.

$$ F_j' = \frac{F_j + 1}{2}. $$

(20)

First, to initialize the variables based on the heuristic function, the more extensive variables than $\frac{1}{2}$ must be chosen and set with 1; if 10 percent of the variables do not gain the value 1, then more extensive variables than $\frac{1}{3}$ will have a value of 1. This procedure continues to reach a limit of $\frac{1}{7}$, where $t$ is a constant number. This process helps us solve the problem and reduce repetition by removing variables, i.e., load/unload actions. The more feasible solution is, the more the constraints are satisfied, and so we can increase the significance of the objective function in the heuristic function by increasing. If we fail to find a feasible solution in our efforts, increasing the importance of the restrictions will increase $\beta$ value. The value of $\beta$ must be increased linearly along with a factor of 10% until it reaches the value above 1. If the value of 1 is achieved, we get the last value with a random number weighted average of 1 to 2. To improve the heuristic function, as the increment of one of these variables increases, the second variable decreases correspondingly. For each acceptable answer, a constant neighborhood radius also is considered for changing some other variables’ value. If there are a consistent number of acceptable responses, the experience matrix becomes more potent because the possibility of other feasible solutions in the neighborhood of existing feasible solutions is high. The position improvement algorithm is used to find the neighborhood radius. The algorithm can terminate in two ways:

1. If all the variables go to the last stations, the feasible solution is sufficiently investigated, and the maximum recovery has also occurred with a very high probability.
2. At the algorithm’s input, a certain number $\zeta$ specifies the maximum number of loads/unload. If $\zeta$ exceeds the set value during the execution of the algorithm, the algorithm stops. Increasing the value of $\zeta$ means that the algorithm remains in local optimization and the answer cannot be improved by changing a fixed number of variables. As a result, the best thing to do in these cases is to restart the algorithm. Parallel implementation of the algorithm is another way, because different local optimizations will be generated and global optimization can be found among them.

Flowchart 2 shows how to extract rules using game theory abstraction, and the Nash equilibrium concept when reduced to a 0–1 IP problem input in each phase. The flowchart is the process of converting items into strategies and calculating related Payoff and then calculating the Nash equilibrium after reducing it to the 0–1 IP problem. Finally, the problem is
Fig. 2 Extracting rules with the proposed method

1. Defining a set of players
2. Define the player action set \( \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\} \)
3. Define the value of support
4. Define the average of all supports \( \text{Supp}_{\text{ave}} \)
5. If \( \sigma_i \geq \text{Supp}_{\text{ave}} \) then:
   - Define \( \psi_j \)
   - Define \( \phi_j \)
   - If \( \sum_{i=1}^{n} \phi_i \neq 0 \) and \( \sum_{i=1}^{n} \phi_j \neq \sum_{j=1}^{N} \psi_j \)
     - Finding Nash equilibrium (FNE)
     - Reduce (FNE) to (0-1 IP)
     - Extracting rule
   - If \( \sum_{i=1}^{n} \phi_i = 0 \) and \( \sum_{i=1}^{n} \phi_j = \sum_{j=1}^{N} \psi_j \)
     - \( F_i = \frac{\sum_{i=1}^{n} \phi_j - \sum_{j=1}^{N} \psi_j}{|P| \times \sum_{k=1}^{N} |A_k|} \)
     - Extracting rule
solved resorting to BTA contributing to the extraction of the rules.

Algorithms 1 and 2 are pseudocodes for the rule extracting process by abstracting game theory and solving the FNE with BTA. The output of the algorithm is a vector which its elements are 0 and 1. 1 indicates the existence of a strategy (data) in the extracted rules, and 0 indicates the absence of a strategy in the extracted rules.

Algorithm 1 Extracting rules by abstracting game theory and solving Nash equilibrium with BTA.

1: Step1: Primary abstraction of the problem
2: Defining a set of players \( P = \{P_1, P_2, \ldots, P_n\} \) from a set of data generation sources (such as transactions \( T = \{T_1, T_2, \ldots, T_n\} \));
3: Defining the player action set \( \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \) from the generated dataset (such as items);
4: Defining the value of support with \( \text{Supp}_{\text{ave}} \) and called \( \text{Supp}_{\text{ave}} \);
5: Defining \( \phi_j \) if for \( \sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\} \) we have \( \sigma_i \geq \text{Supp}_{\text{ave}} \) otherwise we call it \( \varphi_j \);
6: Defining a set of player Payoff \( \Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_n\} \) with the following conditions:
7: \( \sum_{j=1}^{n} \phi_j = 0 \) then \( \Gamma_i \neq 0 \) and if \( \sum_{j=1}^{n} \varphi_j \neq 0 \) (i.e., \( \sum_{j=1}^{m} \phi_j \neq \sum_{j=1}^{m} \varphi_j \)) then:
8: \( \frac{\sum_{j=1}^{n} \phi_j}{|P| \times \sum_{k=1}^{m} |A_k|} \) or \( \frac{\sum_{j=1}^{m} \varphi_j}{|P| \times \sum_{k=1}^{n} |A_k|} \)
9: \( \frac{\sum_{j=1}^{n} \phi_j - \sum_{j=1}^{m} \varphi_j}{|P| \times \sum_{k=1}^{n+m} |A_k|} \)
10: otherwise
11: Step2: Reducing Nash equilibrium to 0-1 IP
12: Solve the following problem:
13: \[
\max_{x, y, p, q} \{x A y^T + x B y^T - p - q\}
\]
14: \( A y^T \leq p e_m \)
15: \( B x^T \leq q e_n \)
16: \( x_i = 0 \) or \( 1, \forall i \)
17: \( y_j = 0 \) or \( 1, \forall j \)
18: \( \sum_{i=1}^{m} x_i = 1 \)
19: \( \sum_{j=1}^{n} x_j = 1 \)

Algorithm 2 Continuation of algorithm 1

1: while(1 of terminating conditions is not satisfying)
2: 
3: 
4: loadpassegers(from STSG);
5: result \( \leftarrow \) analyze_candiates();
6: stations find a solution(based on result, IKD, SKD,
7: heuristic_function);
8: update heuristic_function();
9: if(possible(stations))
10: 
11: unload_passegers(to stations);
12: update IKD; //for passerger to decide which station to go
13: update SKD; //find other solutions
14: 
15: else
16: 
17: try again to find a possible solution;
18: 
19: }

3.6 Computational complexity of the proposed method

Researchers showed that the Nash equilibrium complexity falls into the conventional PPAD class (Daskalakis et al. 2006). The PPAD class was first introduced by Papadimitriou (2015). PPAD is a subclass of the TFNP class, which is itself a subclass of the FNP. The TFNP class ensures that there is definitely a solution in the FNP class. Chen and Deng showed that since there is definitely at least one Nash equilibrium in strategic game, the complexity of the Nash equilibrium falls into a class that always has at least one correct answer. The PPAD computational complexity class is expressed as follows:

Definition 1 A binary relation \( P(x, y) \) is a TFNP if and only if there is an algorithm in a deterministic polynomial time that can determine that \( P(x, y) \) has any given \( x \), \( y \) and at least for each \( x \) exists a \( y \). In PPAD subclass, there is the guarantee, so that \( P(x, y) \) is defined as a directional graph.
Given that in the proposed method the FNE problem would be solved by BTA—which is a meta-heuristic algorithm, and the complexity of this algorithm depends on the sensitivity required for the extraction rules—it is impossible to measure the computational complexity for this part of the method, but it is possible or other parts. After calculating the Support level of all items through a linear search at the order of \(O(n)\) the items that are less than the average value, Support can be omitted. If we assume that in the worst case no item is omitted, another search at the order of is \(O(n + n \log n)\) needed to calculate the Payoffs because each item must be calculated in two \(\text{Supp_{avg}} \leq I_{\text{item}}\) and \(\text{Supp_{avg}} > I_{\text{item}}\) modes. Therefore, the computational complexity of the present research’s method, i.e., \(O(n + n \log n)\) would be in addition to the computational complexity of finding Nash equilibrium with BTA.

4 Implementation and results

To simulate, we considered 30 hypothetical transactions of a hypothetical store. Each of the ten items in the hypothetical store randomly contains some items purchased by the hypothetical customer. Each customer can buy any amount of items from 1 to 10. Therefore, out of these 30 transactions, we face with transactions with a variety of items purchased. The goal is to find rules that have 60% confidence. So, a non-cooperative game with 30 players is defined. That is, we have \(P = \{P_1, P_2, \ldots, P_{30}\}\). The extensive form of the game tree will be defined based on Eqs. 3, 4, and 5. Figure 3 shows part of this tree due to space constraints.

4.1 Implementation

Given that in the proposed example, for simulation, \(\sigma = \sigma_{\text{Allowed}}\) we have:

\[
\begin{align*}
\sigma &= \{\sigma_1, \sigma_2, \ldots, \sigma_{30}\} = \sigma_{\text{Allowed}} \\
&= \{\sigma_{\text{allowed}1}, \sigma_{\text{allowed}2}, \ldots, \sigma_{\text{allowed}30}\} \\
|\sigma_{\text{Allowed}}| &= k = 10.
\end{align*}
\]

The matrix \(\theta_\sigma\) with size \(|P| \times 10\) will be equal to:

\[
\theta_\sigma(i, j) = \begin{cases} 
1 & \text{if } \sigma_{\text{allowed}} = 1 \\
0 & \text{otherwise } 1 \leq i \leq 10.
\end{cases}
\]

We said that \(I_i(T_j)\) is the amount of Payoff value \(P_i\) in node \(\chi\). In function \(I_i\), there is the number of \(q\) free parameters, where \(q \leq 10\), and then, the matrix \(\theta_{\text{parameter}}\) with size of \(|P| \times q\) will be equal to \(\theta_{\text{parameter}} = [\theta_{\text{parameter}}(i, j)]_{|P| \times q}\). If equation 16 \(N = 30\) and \(M = 10\) are in the continuous state, the goal is to find the answer to Eq. 17, and in the reduced form to the 0–1 IP, the goal is to find the solutions to Eq. 18. Table 2 shows the hypothetical database of the present study.

Each player’s strategy includes exactly \(A = (a_1, a_2, \ldots, a_k)\), where \(a_j = I_i\) and if an item is not selected by the other player (transaction) then \(a_j = 0\) and otherwise \(a_j = 1\). Therefore, the number of elements in the strategy set is the same for all players, i.e., \(|A_i|\) means the number of elements in the set \(A\) for player \(i\), then \(|A_1| = |A_2| = \ldots = |A_k|\), where \(k\), as previously described, is the maximum number of data generated in the game \(k = 10\). In other words, the data source or the game players (in our hypothetical example, are the transactions) make up the strategies. Equation 10 is used to calculate the game Payoff for each player \(i\) as \(I_i\) in the game Payoff state or \(\Gamma = (I_1, I_2, \ldots, I_n)\). Table 3 shows the support value of each item (data) for the hypothetical example of the present study, and Table 4 shows the set of strategy \(A_i = (a_1, a_2, \ldots, a_k)\) for each player \(i\) plus \(I_i\) based on Eq. 10.

Elimination of players who do not have useful strategies for extracting rules is a policy based on the amount of Payoff. Depending on the type of issue, the policy varies contributing to a significant impact on the computational complexity of the current method. For example, in Table 4, if the policy of selecting rules is based on 0.1 \(\leq I_i\), then the rules are extracted from the set illustrated in Table 5; so, the search space is practically reduced.

This reduction in complexity is more pronounced when players with the same strategies are eliminated (Table 6).

Another policy could be to choosing strategies with higher Payoff values than the average total Payoff. However, the amount of Payoff on strategies indicates the strength of the process’s participation in the rule’s extraction. The higher the Payoff value, the lower the computational complexity and the higher the rate of confidence would be. However, the number of extracted rules may decrease. Nevertheless, the number of extracted rules may decrease because it is impossible for a rule with Payoff value to be less than \(I_i\), a subset of the extracted super rules. Therefore, depending on the problem

\(\theta\) Springer
Table 2  The hypothetical database used in the present study

| Transactions/data generation sources | Selected items (player/transaction generated data) |
|-------------------------------------|-------------------------------------------------|
| T1                                  | I1, I5, I8                                      |
| T2                                  | I2, I5, I8                                      |
| T3                                  | I3, I6, I9                                      |
| T4                                  | I4, I7, I9                                      |
| T5                                  | I1, I5, I8                                      |
| T6                                  | I2, I5, I8                                      |
| T7                                  | I2, I6, I10                                     |
| T8                                  | I1, I7, I8                                      |
| T9                                  | I3, I5, I10                                     |
| T10                                 | I2, I5, I8                                     |
| T11                                 | I4, I5, I9                                      |
| T12                                 | I3, I6, I9                                      |
| T13                                 | I4, I6, I9                                      |
| T14                                 | I2, I5, I8                                      |
| T15                                 | I5, I6, I8                                      |
| T16                                 | I1, I5, I9                                      |
| T17                                 | I2, I5, I8                                      |
| T18                                 | I3, I5, I10                                     |
| T19                                 | I3, I6, I9                                      |
| T20                                 | I2, I6, I9                                      |
| T21                                 | I2, I5, I8                                      |
| T22                                 | I4, I6, I9                                      |
| T23                                 | I1, I5, I10                                     |
| T24                                 | I5, I6, I8                                      |
| T25                                 | I5, I6, I8                                      |
| T26                                 | I2, I5, I8                                      |
| T27                                 | I1, I6, I9                                      |
| T28                                 | I1, I5, I10                                     |
| T29                                 | I3, I6, I9                                      |
| T30                                 | I2, I5, I8                                      |

Table 3  Support percentage of each item (data) and average support

| Items/generated data | Support percentage |
|---------------------|--------------------|
| I1                  | 25%                |
| I2                  | 36.66%             |
| I3                  | 30%                |
| I4                  | 13.33%             |
| I5                  | 53.33%             |
| I6                  | 26.66%             |
| I7                  | 25%                |
| I8                  | 50%                |
| I9                  | 33.33%             |
| I10                 | 16.66%             |
| Supp_{ave}          | 30.997%            |

Table 4  Strategy set for each player and Payoff of each strategy set

| Players | Strategy | Payoff |
|---------|----------|--------|
| P1      | (1, 0, 0, 0, 1, 0, 0, 1, 0, 0) | 0.29   |
| P2      | (0, 1, 0, 0, 1, 0, 1, 0, 0)   | 0.5184 |
| P3      | (0, 0, 1, 0, 0, 1, 0, 0, 1, 0) | 0.08640 |
| P4      | (0, 0, 1, 0, 0, 1, 0, 1, 0, 0) | 0.01851 |
| P5      | (1, 0, 0, 0, 1, 0, 0, 1, 0, 0) | 0.29   |
| P6      | (0, 1, 0, 0, 1, 0, 1, 0, 0, 1) | 0.5184 |
| P7      | (0, 0, 0, 0, 0, 1, 0, 0, 0, 0) | 0.02466 |
| P8      | (1, 0, 0, 0, 0, 0, 1, 1, 0, 0) | 0.1851 |
| P9      | (0, 0, 1, 0, 1, 0, 0, 0, 0, 1) | 0.02470 |
| P10     | (0, 1, 0, 0, 1, 0, 1, 0, 0)   | 0.5184 |
| P11     | (0, 0, 1, 0, 1, 0, 0, 1, 0, 0) | 0.2715 |
| P12     | (0, 1, 0, 0, 1, 0, 1, 0, 0)   | 0.08640 |
| P13     | (0, 0, 0, 0, 1, 0, 1, 0, 0, 1) | 0.0370 |
| P14     | (0, 1, 0, 0, 0, 1, 0, 0, 0, 1) | 0.5184 |
| P15     | (0, 0, 0, 0, 1, 0, 1, 0, 0, 1) | 0.2715 |
| P16     | (1, 0, 0, 0, 0, 1, 0, 1, 0, 0) | 0.06174 |
| P17     | (0, 1, 0, 0, 1, 0, 0, 1, 0, 0) | 0.5184 |
| P18     | (0, 0, 1, 0, 0, 0, 0, 0, 0, 1) | 0.0    |
| P19     | (0, 0, 1, 0, 0, 0, 1, 0, 0)   | 0.08640 |
| P20     | (0, 0, 0, 0, 1, 0, 1, 0, 1, 0) | 0.1604 |
| P21     | (0, 0, 0, 0, 1, 0, 1, 0, 0, 0) | 0.5184 |
| P22     | (0, 0, 0, 1, 0, 0, 0, 0, 1, 0) | 0.2715 |
| P23     | (1, 0, 0, 0, 1, 0, 0, 0, 0, 1) | 0.04322 |
| P24     | (0, 0, 0, 1, 1, 0, 0, 0, 0, 0) | 0.01851 |
| P25     | (0, 0, 0, 0, 1, 1, 0, 0, 0, 0) | 0.5184 |
| P26     | (0, 0, 0, 0, 1, 0, 0, 0, 0, 0) | 0.2283 |
| P27     | (0, 0, 0, 0, 0, 0, 0, 0, 0, 0) | 0.08640 |
| P28     | (1, 0, 0, 0, 1, 0, 0, 1, 0, 0) | 0.04322 |
| P29     | (0, 0, 0, 1, 0, 0, 0, 1, 0, 0) | 0.08640 |
| P30     | (0, 1, 0, 0, 1, 0, 0, 1, 0, 0) | 0.5184 |

of determining the boundary, I_j is determinative, i.e., if only the existence of high confidence is needed, the values of I_j are higher. If more rules are needed player who has strategies with (∑^{m}_{j=1}Φ_j - ∑^{n}_{j=1}Φ_j) > 0 or even ∑^{n}_{j=1}Φ_j > 0 demonstrate a lower value of I_j. This can be proved easily by induction. We can choose a policy of less than Supp_{ave} for more precise rules, but naturally the final cost (computational complexity) will increase. Since in the hypothetical example of the present study, after problem modeling, only six players and their strategies remained in the game (due to Payoff’s criterion per player), so the extraction rules and the strategy equilibrium points for these six players are based on their Payoff based on the amount of support per strategy. The initialization of the BTA will be operationalized so that in the BTA the variables (strategies) and their Support values are close to 1 (repeated in most transactions), i.e., \frac{3}{10} ≤ x_i ≤ 1.
using BTA Algorithm for finding Nash equilibrium problem aiming the extraction of rules... 453

Table 5 Players with candidate strategies for extracting rules based on Payoff value (0.1 ≤ \(I_i\))

| Players | Strategy                  | Payoff |
|---------|---------------------------|--------|
| \(P_2\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_5\) | (0, 1, 0, 0, 1, 0, 0, 0)  | 0.5184 |
| \(P_{10}\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_{14}\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_{17}\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_{21}\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_{25}\) | (0, 1, 0, 0, 1, 0, 0, 0)  | 0.5184 |
| \(P_{30}\) | (0, 1, 0, 0, 1, 0, 0, 0)  | 0.5184 |
| \(P_1\)  | (1, 0, 0, 0, 1, 0, 0, 0)  | 0.29   |
| \(P_3\)  | (1, 0, 0, 0, 1, 0, 0, 0)  | 0.29   |
| \(P_{11}\) | (0, 0, 1, 1, 0, 0, 0, 0)  | 0.2715 |
| \(P_{15}\) | (0, 1, 0, 1, 0, 0, 0, 0)  | 0.2715 |
| \(P_{22}\) | (0, 0, 1, 1, 0, 0, 0, 0)  | 0.2715 |
| \(P_{26}\) | (0, 1, 0, 0, 0, 1, 0, 0)  | 0.2283 |
| \(P_8\)  | (1, 0, 0, 0, 0, 1, 0, 0)  | 0.1851 |
| \(P_{20}\) | (0, 1, 0, 0, 1, 0, 0, 0)  | 0.1604 |

Table 6 Eliminating players with the same strategy and Payoff value (0.1 ≤ \(I_i\))

| Players | Strategy                  | Payoff |
|---------|---------------------------|--------|
| \(P_2\) | (0, 1, 0, 0, 1, 0, 0, 0)   | 0.5184 |
| \(P_1\)  | (1, 0, 0, 0, 1, 0, 0, 0)  | 0.29   |
| \(P_{11}\) | (0, 0, 1, 1, 0, 0, 0, 0)  | 0.2715 |
| \(P_{26}\) | (0, 1, 0, 0, 0, 1, 0, 0)  | 0.2283 |
| \(P_8\)  | (1, 0, 0, 0, 0, 1, 0, 0)  | 0.1851 |
| \(P_{20}\) | (0, 1, 0, 0, 1, 0, 0, 0)  | 0.1604 |

Table 7 Finding Nash equilibrium points by BTA (extracting major rules, the confidence value, and related ranks)

| Super rules | Confidence | Rank |
|-------------|------------|------|
| \(I_2 \rightarrow I_5 \rightarrow I_8\) | 72.72% | 1 |
| \(I_5 \rightarrow I_8 \rightarrow I_9\) | 0 | - |

Table 8 Extracting rules and the confidence values

| Rules | Confidence |
|-------|------------|
| \(I_2 \rightarrow I_5\) | 72.72% |
| \(I_2 \rightarrow I_8\) | 81.81 |
| \(I_5 \rightarrow I_8\) | 68.75 |

resides inside the SSG station. Then, the STSG and MTSG station variables are specified, respectively. The STSG station variables will include strategies with a support value of \(\frac{5}{10} \leq x_i < \frac{7}{10}\). The MSTG station variables will consist of strategies with a support value of \(\frac{1}{4} \leq x_i < \frac{5}{10}\). In the second iteration, the algorithm combines the variables of the three SSG, STSG, and MTSG and the vectors of the variables set at the LLSSG station, provided that each combination of variables (strategies) is present in 60% for each of the previous stations.

4.2 Results

Experimental results showed that the more appropriate the stations’ initialization based on the probabilistic distribution is, the faster the BTA can find the equilibrium points. Appropriate probability distribution means selecting the appropriate range of \(x_i\) based on each strategy’s Support value. Also, for each extracted major rule, a rank can be set relative to the other rules’ Confidence value. And only a subset of the higher-order major rule can be selected as the final rule. This point will also reduce the computational complexity of the method used in the present study. Equilibrium points extract major rules; each subset will be rule with a Confidence value above the specified value (60% in our example). Tables 7 and 8 show the extracted major rules and sub-rules and the Confidence value for each rule in the hypothetical example of the present study. In Table 7, the rank of each major rule specifies that in Table 8, only the subset of major rules in the first rank is selected as the final rule because the second extraction rule has zero Confidence and no validity.

Experimental results after 100 epochs of BTA showed that BTA, in the worst case, could find equilibrium points in only seven epochs (Table 7), which is the same as the extracted major rule. Each subset of this major rule is the final extracted rule. Table 8 shows the extracted sub-rules and related Confidence value. Making use of game theory abstraction, we reduced the computational complexity. When facing processing constraints higher computational complexity is seen when applying a more extensive Payoff range, so controlling computational complexity would be quite smart. Figure 4 shows the BTA convergence rate in achieving the optimal solution and finding equilibrium points. This figure shows the top three executions of the algorithm. In all three executions of BTA, in the worst case, only seven epochs could deliver the answer. Gradually, each epoch took less time and finally reached the optimal answer. The best performance (blue chart) is converged to the answer only after five epochs of performance.

Table 9 shows the fuzzy generated data. This table is the fuzzy mode of Table 3, where each of the data has a membership degree. In fuzzy data, each strategy combination, and the value of \(I_i\), \(\mu_{A_k}(a_k)\) is also defined, which can determine the value \(\sum_k \mu_{A_k}(a_k)\), where \(k\) is the number of strategies used by a player.
Table 9 Support percentage of each item (fuzzy data) and average support

| Items/generated data | Support percentage (%) | Membership degree |
|----------------------|------------------------|-------------------|
| I_1                  | 25                     | 0.2               |
| I_2                  | 36.66                  | 0.37              |
| I_3                  | 30                     | 0.45              |
| I_4                  | 13.33                  | 0.11              |
| I_5                  | 53.33                  | 0.34              |
| I_6                  | 26.66                  | 0.13              |
| I_7                  | 25                     | 0.43              |
| I_8                  | 50                     | 0.12              |
| I_9                  | 33.33                  | 0.32              |
| I_{10}               | 16.66                  | 0.27              |

Figure 4 Top 3 BTA performances in terms of convergence speed to the optimal solution (finding Nash equilibrium points)

Figure 5 shows the BTA convergence rate to reach the optimal solution and finding the Nash equilibrium points when the data are fuzzy. This figure shows the top 3 executions of the algorithm. In all three executions, BTA used only 12 epochs to reach the answer in the worst case. Gradually, each epoch took less time and finally reached the optimal answer. The best performance (blue chart) is converged to the answer only after nine epochs of performance.

Table 10 shows a comparative analysis of the proposed method with improved Apriori and FP-growth algorithms.

Generalization for fuzzy data, reduction of computational complexity, intelligent control of algorithm parameters, and control of search space for big data are the factors of the proposed method’s superiority based on game theory’s abstraction for rule learning.

4.2.1 Comparison with other rule extraction methods

To evaluate the performance of a model, two methods are usually used:
Using BTA Algorithm for finding Nash equilibrium problem aiming the extraction of rules...

### Table 10: Comparative analysis among the proposed approach, Apriori, and FP-growth

| Parameters                                | Proposed approach | Apriori | FP-growth |
|-------------------------------------------|-------------------|---------|-----------|
| Abstraction                               | Game theory       | Probability theory | Tree theory (graph) |
| Ability to process fuzzy data             | Generalizable for fuzzy data | Non-generalizable for fuzzy data | Non-generalizable for fuzzy data |
| Ability to control the search space       | Ability to control search space and algorithm | Uncontrollable search space and algorithm | Uncontrollable search space and algorithm |
| Ability to control computational complexity | Ability to control computational complexity and the possibility of decrease computational complexity | Uncontrollable computational complexity | Uncontrollable computational complexity |

1. Evaluation based on the assumptions to which the model must apply.
2. Evaluation based on the efficiency of the model in predicting new values (not observed).

To evaluate the model based on the first procedure, we rely on the data that have been observed and used to structure the model, which is called cross-validation. We expect the model to have the least sum of squares of error compared to any other given models. If the sum of training data is differentiated randomly into \( k \) sub-sample or fold with the same volume, in each stage of CV, \( k - 1 \) number of these layers can be considered as the training dataset and another one as the validation dataset. The validation method is called \( k \)-Fold which was \( k = 5 \) in the present study. The present study’s method is compared with three other methods (decision tree, eclectic rule extraction from neural network with multi-hidden layer (ERENN-MHL) introduced in Chakraborty et al. (2020), and the method introduced in Zou et al. (2021) that was proposed based on linguistic-valued intuitionistic fuzzy layered concept lattice (LVIFLCL) in the same domain. The reason for this refers to the fact that the other methods benefited from highly efficient abstractions for rule extraction issue. The accuracy and validity of the obtained rules are determined by the criteria of accuracy, precision, recall, and \( f \)-measure. In the field of machine learning, precision means the percentage of model predictions that are relevant, but recall refers to the percentage of total predictions that are correctly categorized by the model. Precision means dividing the number of items detected by the model correctly by the total number of items created by the model, and recall means dividing the number of items detected by the model correctly by the number of items that have been truly detected. Accuracy means how close the measured value is to the actual value. Accuracy should have high precision, but not necessarily the reverse is meant. High bias and variance mean low accuracy. Precision or accurate measurements of a value indicate the closeness of the measurements values. The accuracy index is calculated by the following equation:

\[
\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}.
\]  

(23)

The criterion for measuring the accuracy and precision of the extracted rules is calculated with two precision and recall indexes, which are obtained from the following relations:

\[
\text{Precision} = \frac{tp}{tp + fp},
\]

(24)

\[
\text{Recall} = \frac{tp}{tp + fn},
\]

(25)

in which: \( tp \) is the correctly detected true-positive neighborhoods,

\( fp \) is the correctly detected false-positive neighborhoods,

\( fn \) is the correctly undetected false-negative neighborhoods,

\( tn \) is the correctly undetected true-negative neighborhoods.

The \( f \)-measure criterion, which is actually a balanced combination of accuracy and precision measures, can be used in cases where the cost of false positive and false negative is different. If the cost of false positive and false negative is approximately equal, the same accuracy criteria can be used. The \( f \)-measure criterion is calculated by the following equation:

\[
\text{f - measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}.
\]

(26)

Table 11 shows the comparison of testing accuracy with fivefold cross-validation results with the hypothetical data used in the present study. To compare the accuracy criterion of the present study’s method with other methods, we implemented 30, 60, 90, 500, 1200, 1800, 3200, and 5400 hypothetical randomness together with all four methods; the results of the accuracy criterion are shown in Table 11. The results indicate a significant superiority of the present study’s method over the methods selected for comparison. Chart 6 shows the
accuracy stability of the decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method. As it can be seen from the results of the chart, the present study’s method has better stability in terms of accuracy criterion than other compared methods.

Table 12 shows the average precision for fivefold with the hypothetical data used in the present study. In order to compare the precision criterion of the present research method with other methods, we implemented 30, 60, 90, 500, 1200, 1800, 3200, and 5400 hypothetical randomness together with all four methods. The results of the precision criterion are shown in Table 12. The results indicate a significant superiority of the present study’s method over the methods selected for comparison. Chart 7 shows the precision stability of the decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method. As it can be seen from the results of the chart, the present study’s method has better stability in terms of precision criteria than other compared methods (Figs. 6 and 7).

Table 13 shows the average recall for fivefold with the hypothetical data used in the present study. To compare the recall criterion of the present study’s method with other methods, we implemented 30, 60, 90, 500, 1200, 1800, 3200, and 5400 hypothetical randomness together with all four methods; the results of the recall criterion are shown in Table 13. The results indicate a significant superiority of the present study’s method over the methods selected for comparison. Chart 8 shows the recall stability of the decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method. As it can be seen from the results of the chart, the present study’s method has better stability in terms of recall criterion than other compared methods (Figs. 8 and 9).

Table 14 shows the average f-measure for fivefold with the hypothetical data used in the present study. To compare the precision criterion of the present study’s method with other methods, we implemented 30, 60, 90, 500, 1200, 1800, 3200, and 5400 hypothetical randomness together with all four methods. The results of the f-measure criterion are shown in Table 14. The results indicate a significant superiority of the f-measure method of the present study over the methods selected for comparison. Chart 9 shows the stability of the f-measure of decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method. As it can be seen from the results of the chart, the present study’s method has a better stability in terms of f-measure criterion than other compared methods.

5 Discussion and future work

In the present study, the researchers seek to propose a method with reduced computational capacity which avails more optimal and reliable extracted rules compared to other methods through changing the abstraction of the problem of rule extraction in a rule learning system to game theory. The main idea of the present study is based on finding super rules that can cover the most frequent rules and be modeled based on game theory so that the super rules are capable to be implemented through the concept of Nash equilibrium in game theory. Abstraction based on game theory has been the focus of data mining activities including rule learning for the past two decades. For example, data mining activity can be modeled as a “non-collaborative game” performed by multiple users. The basic elements of a given game include players, actions, the benefits of each action, and information. If no player deviates from their strategy, a Nash equilibrium is achieved. Recently, this kind of view or to be more precise this kind of mathematical modeling based on game theory has become very popular in data mining, so that research on new methods in data clustering, data classification, data pattern extraction, or data prediction models has been presented. Utilizing game theory is also to improve two algorithms, namely to say Apriori and FP-growth, and other algorithms for detecting repetitive patterns to extract rules that rely on previous information. Of course, the use of game theory is based on the initial abstraction of algorithms, and therefore, the use of game theory in extracting rules means utilizing hybrid algorithms that use game theory in some algorithms that use another abstraction and the basis of such algorithms has nothing to do with the abstraction of game theory. Primary methods for learning rules of predicting data are methods that introduce Nash learning. These methods are based on probabilistic abstraction. In methods that have probabilistic abstraction, Nash learning is strongly rejected because of the probabilistic basis of logical behaviors. But they introduce the extraction of the rules of a given game by distributing the probabilities in the actions in the game within a hypothetical game. This probabilistic distribution is updated with each round of play. After these researches, methods based on game theory were proposed to extract association rules. In these studies, the action of each agent in a game is assumed based on predetermined and defined strategies. The set of actions of each agent against another agent is quite clear, and the action of each agent can influence the actions of other agents. Finally, the set of association rules of the studied system is extracted from the effective actions of each of the different agents. Because these methods presuppose definite behavior instead of logical behavior, they can consider Nash equilibrium as a game answer designed to learn the rules of data prediction. But as it was mentioned before, the deterministic Nash equilibrium cannot be found easily for logical behavior that has a probabilistic basis, especially for modeling data learning rules. Because the game space has a wide range of strategies with different probabilistic distributions, and this probability distribution is updated at each round of the game, the Nash equilibrium in these
Table 11  Comparison of testing accuracies with fivefold cross-validation results

| Number of transactions | Present study’s method (%) | Decision tree (%) | ERENN-MHL (%) | LVIFLCL (%) |
|------------------------|----------------------------|-------------------|--------------|------------|
| 30                     | 90.023                     | 78.67             | 88.34        | 89.76      |
| 60                     | 90.067                     | 71.39             | 89.10        | 89.63      |
| 90                     | 90.043                     | 73.78             | 83.26        | 88.18      |
| 500                    | 90.003                     | 70.97             | 81.36        | 56.56      |
| 1200                   | 90.023                     | 68.45             | 83.91        | 83.93      |
| 1500                   | 89.81                      | 63.12             | 77.71        | 79.91      |
| 3200                   | 90.006                     | 61.68             | 73.19        | 75.61      |
| 5400                   | 88.45                      | 57.89             | 70.023       | 73.81      |

Fig. 6  Accuracy stability of the decision tree methods, ERENN-MHL, LVIFLCL, and the present study’s method

Table 12  Average precision for fivefold cross-validation

| Number of transactions | Present study’s method (%) | Decision tree (%) | ERENN-MHL (%) | LVIFLCL (%) |
|------------------------|----------------------------|-------------------|--------------|------------|
| 30                     | 81.43                      | 69.82             | 79.53        | 80.16      |
| 60                     | 81.87                      | 66.53             | 79.65        | 78.36      |
| 90                     | 82.021                     | 65.81             | 77.35        | 81.26      |
| 500                    | 81.98                      | 63.01             | 76.91        | 80.09      |
| 1200                   | 82.11                      | 63.52             | 74.46        | 77.63      |
| 1500                   | 82.08                      | 60.39             | 74.02        | 75.86      |
| 3200                   | 82.21                      | 58.88             | 71.67        | 72.74      |
| 5400                   | 80.99                      | 52.20             | 69.48        | 71.04      |
Fig. 7  Precision stability of the decision tree methods, ERENN-MHL, LVIFLCL, and the present study’s method.

Table 13  Average recall for fivefold

| Number of transactions | Present study’s method (%) | Decision tree (%) | ERENN-MHL (%) | LVIFLCL (%) |
|------------------------|----------------------------|-------------------|---------------|-------------|
| 30                     | 94.78                      | 75.79             | 91.32         | 93.89       |
| 60                     | 94.71                      | 73.78             | 92.18         | 91.57       |
| 90                     | 95.05                      | 74.19             | 90.71         | 92.81       |
| 500                    | 94.67                      | 72.39             | 90.29         | 91.38       |
| 1200                   | 94.11                      | 70.83             | 89.68         | 90.25       |
| 1500                   | 94.90                      | 69.43             | 88.81         | 90.53       |
| 3200                   | 94.33                      | 67.63             | 83.41         | 88.64       |
| 5400                   | 93.93                      | 62.69             | 81.07         | 85.39       |

Table 14  Average f-measure for fivefold

| Number of transactions | Present study’s method (%) | Decision Tree (%) | ERENN-MHL (%) | LVIFLCL (%) |
|------------------------|----------------------------|-------------------|---------------|-------------|
| 30                     | 88.31                      | 61.76             | 86.17         | 87.70       |
| 60                     | 88.36                      | 60.30             | 85.91         | 87.62       |
| 90                     | 88.71                      | 62.42             | 81.73         | 85.76       |
| 500                    | 89.05                      | 60.01             | 80.81         | 84.26       |
| 1200                   | 88.16                      | 58.82             | 79.98         | 83.95       |
| 1500                   | 88.39                      | 54.62             | 77.64         | 80.43       |
| 3200                   | 87.81                      | 52.11             | 76.08         | 79.76       |
| 5400                   | 88.73                      | 50.07             | 75.28         | 77.22       |
Fig. 8 Recall stability of the decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method.

Fig. 9 Stability of the f-measure of decision tree, ERENN-MHL, LVIFLCL methods, and the present study’s method.
systems falls into the TFNP computational complexity class. Nash equilibrium can also be reduced to the problem of 0–1 IP. So we use the BTA which is designed to solve the 0–1 IP problem to approximately find the Nash equilibrium problem in the data learning rules game and extract the learning rules. In the present method the goal is to find rules that cover most repetitive instances. This idea can be modeled on a non-collaborative game, and through the concept of Nash equilibrium, a way can be found to systematically implement the rules for repetitive instances. In other words, the equilibrium point in a designed game is where the search to find the considered rules ends, and because finding the equilibrium point resides in the TFNP complexity class, we use the BTA to find these points. But before that Nash equilibrium problem is reduced to 0–1 IP.

In the present study, using the abstraction of game theory, we addressed the problem of extracting association rules used in rule learning systems. We turned the problem into a non-cooperative game with $N$ players, each having a limited set of strategies. Each player is, in fact, a data source of which the generated data by this source occupy the role of players’ strategies. We introduced Eq. 10 to estimate each player’s $i$ Payoff. It is to say that equation’s efficiency for adequate calculation of each player’s Payoff can be proved by induction. We were looking for its Nash equilibrium points to solve the game because the equilibrium points are extracted major rules that their related subsets also have at least as many as confidence major rules. We reduced the game’s FNE 0–1 IP problem and solved it through the BTA. After abstracting and simulating the problem, the following results could be summarized:

1. The Payoff function (formula 10) introduced in the present study is a fundamental function that in case the value of minsup is not defined correctly, it cannot function correctly in identifying the items that should participate in the final rules. Solving these problem of the introduced function can be a topic for future research.
2. Although the BTA operates well to find the Nash equilibrium—after the problem is reduced to a 0–1 IP problem—like any other meta-heuristic algorithm it cannot guarantee the best responses to the FNE problem. As a suggestion for future research, the use of approximate algorithms, especially algorithms based on computational geometry, can guarantee reliable approximations for FNE.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Agrawal R, Srikant R (1994) Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th international conference on very large data bases, VLDB ’94, San Francisco. Morgan Kaufmann Publishers Inc, pp 487–499
Ai D, Pan H, Li X, Gao Y, He D (2018) Association rule mining algorithms on high-dimensional datasets. Artif Life Robot 23(3):420–427
Ait-Mlouk A, Agouti T, Gharati F (2017) Mining and prioritization of association rules for big data: multi-criteria decision analysis approach. J Big Data 4(1):42
Asghar M, Subhan F, Ahmad H, Khan W, Hakak S, Gadekallu T (2020) Senti-esystem: a sentiment-based esystem-using hybridized fuzzy and deep neural network for measuring customer satisfaction. Softw Pract Exp 51:06
Bell J (2020) Association rules learning, pp 129–142.02
Bhagat A, Sanjay S, Pardasani K (2010) Feed forward neural network algorithm for frequent patterns mining. Int J Comput Sci Inf Secur 8:11
Bodaghi M, Samieefar K (2019) Meta-heuristic bus transportation algorithm. Iran J Comput Sci 2(1):23–32
Carmona CJ, del Jesus MJ, Herrera F (2018) A unifying analysis for the supervised descriptive rule discovery via the weighted relative accuracy. Knowl Based Syst 139:89–100
Chakravarty V, Pandit V, Sabharwal Y (2009) Analysis of sampling techniques for association rule mining, pp 276–283, 01
Chakraborty M, Biswas S, Purkayastha B (2020) Rule extraction from neural network trained using deep belief network and back propagation. Knowl Inf Syst 62:09
Cheng X, Sen S, Shengzhi X, Li Z (2015) Dp-apriori: a differentially private frequent itemset mining algorithm based on transaction splitting. Comput Secur 50:74–90
Using BTA Algorithm for finding Nash equilibrium problem aiming the extraction of rules...

Cózar J, delaOsa LG, ámez José A (2018) Learning compact zero-order tsk fuzzy rule-based systems for high-dimensional problems using an apriori + local search approach. Inf Sci 433–434:1–16

Daskalakis C, Goldberg PW, Papadimitriou CH (2006) The complexity of computing a nash equilibrium. In: Proceedings of the thirty-eighth annual ACM symposium on theory of computing, STOC ’06, pp 71–78, New York. Association for Computing Machinery

Djenouri Y, Comuzzi M (2017) Combining apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. Inf Sci 420:1–15

Durkin J, Durkin J (1998) Expert systems: design and development, 1st edn. Prentice HallPTR, New York

Durlauf Steven N, Blume Lawrence E (2010) Learning and Evolution in Games: An Overview, pages 184–190. Palgrave Macmillan UK, London

Fürnkranz J, Gamberger D, Lavrač N (2012) Supervised descriptive rule learning, pp 247–265.09

Fürnkranz J, Kliegr T (2015) A brief overview of rule learning. 08

Gan W, Lin C-W, Viger PF, Chao H-C, Philip Yu (2019) A survey of parallel sequential pattern mining. ACM Trans Knowl Discov Data 13:1–34, 06

Han J, Pei J, Yin Y (2000) Mining frequent patterns without candidate generation. SIGMOD Rec 29(2):1–12

Hoque S, Mustafa R, Mondal S, Bhuiyan Md (2015) A fuzzy frequent pattern-growth algorithm for association rule mining. Int J Data Min Knowl Manag Process 5:21–33, 09

Huici Q (2020) Improvement parallelization in apriori algorithm. In: Proceedings of the 2020 international conference on computers, information processing and advanced education, CIPAE, New York 2020, Association for Computing Machinery, pp 235–238

Iancu I, Gabrovanu M (2010) Fuzzy logic controller based on association rules. Anele Universitatii din Craiova. Seria Matematica Informatic, 37, 01

Isazadeh A, Pedrycz W, Mahan F (2014) Eca rule learning in dynamic environments. Expert Syst Appl 41(17):7847–7857

Lin K-C, Liao I-E, Chen Z-S (2011) An improved frequent pattern-growth algorithm for mining association rules. Expert Syst Appl 38:5154–5161, 05

Liu H, Chen S-M (2019) Multi-stage mixed rule learning approach for advancing performance of rule-based classification. Inf Sci 495:65–77

Mangalamalli A, Pudi V (2009) Fuzzy association rule mining algorithm for fast and efficient performance on very large datasets. In: 2009 IEEE international conference on fuzzy systems, pp 1163–1168

Millette L (2012) Improving the knowledge-based expert system life-cycle. UNF Grad Theses Diss 407:01

Miyaji A, Rahman MS (2011) Privacy-preserving data mining: a game-theoretic approach. In: Li Y (ed) Data and applications security and privacy XXV. Springer, Berlin, Heidelberg, pp 186–200

Narahari Y (2010) Game theoretic approaches to knowledge discovery and data mining. In: Zaki MJ, Xu J, Yu BR, Pudi V (eds) Advances in Knowledge discovery and data mining. Springer, Berlin, Heidelberg, p 3

Novak PK, Lavrač N, Webb G (2009) Supervised descriptive rule discovery: a unifying survey of contrast set, emerging pattern and subgroup mining. J Mach Learn Res 10:377–403, 01

Novak PK, Lavrač N, Webb GI (2010) Supervised descriptive rule induction. Springer, Boston, pp 938–941

Palshikar GK, Kale MS, Apte MM (2007) Association rules mining using heavy itemsets. Data Knowl Eng 61(1):93–113 (Business Process Management)

Papadimitriou C (2015) Chapter 14—the complexity of computing equilibria. Volume 4 of Handbook of game theory with economic applications. Elsevier, pp 779–810

Piatetsky-Shapiro G (1991) Discovery, analysis, and presentation of strong rules. In: Piatetsky-Shapiro G, Frawley WJ (eds) Knowledge discovery in databases. AAAI/MIT Press, London, pp 229–248

Pierrard R, Poli J-P, Hudelot C (2018) A fuzzy close algorithm for mining fuzzy association rules. Working paper or preprint

Rathinasabapathy R, Bhaskaran R (2009) Performance comparison of hashing algorithm with apriori. In: 2009 International conference on advances in computing, control, and telecommunication technologies, pp 729–733

Saabith S, Sundararajan E, Abu BA (2016) Parallel implementation of apriori algorithms on the Hadoop-Mapreduce platform—an evaluation of literature. J Theor Appl Inf Technol 85(321–351):03

Sabita B, Mishra D, Shruti M, Satapathy S, Rath A, Acharya M (2010) Pattern discovery using fuzzy fp-growth algorithm from gene expression data. Int J Adv Comput Sci Appl 5:11

Shabtay L, Fournier-Viger P, Yafi R, Dattner I (2021) A guided fp-growth algorithm for mining multitude-targeted item-sets and class association rules in imbalanced data. Inf Sci 553:353–375

Singh S, Garg R, Mishra P (2015) Performance analysis of apriori algorithm with different data structures on hadoop cluster. Int J Comput Appl 128:975–8887, 10

Sowan B, Keshav Dahal MA, Hossain LZ, Spencer L (2013) Fuzzy association rule mining approaches for enhancing prediction performance. Expert Syst Appl 40(17):6928–6937

Soysal ÔM, Gupta E, Donepudi H (2016) A sparse memory allocation data structure for sequential and parallel association rule mining. J Supercomput 72(2):347–370

Stahl D (1997) Rule learning in symmetric normal-form games: theory and evidence. Care working papers, The University of Texas at Austin, Center for Applied Research in Economics

Stahl DO (2000) Rule learning in symmetric normal-form games: theory and evidence. Games Econom Behav 32(1):105–138

Telikani A, Gandomi AH, Shabbahrami A (2020) A survey of evolutionary computation for association rule mining. Inf Sci 524:318–352

Thakur S, Ninoria SZ (2017) An improved progressive sampling based approach for association rule mining. Int J Comput Appl 165:27–35

Theocharopoulos G, Bobori C, Vlamos P (2017) Formal models of biological systems. In: Panayiotis V (ed) GeNeDis 2016. Springer, Cham, pp 325–338

Triantaphyllou E, Felici G (2006) Data mining and knowledge discovery approaches based on rule induction. Techniques 6:06

Vasoya A, Koli N (2016) Mining of association rules on large database using distributed and parallel computing. Procedia Computer Science, 79:221–230, 2016, Proceedings of international conference on communication, computing and virtualization (ICCCCV)

Vijayakalshmi V, Pethalakshmi A (2015) An efficient count based transaction reduction approach for mining frequent patterns. Procedia Comput Sci, 47:52–61. Graph Algorithms, high performance implementations and its applications (ICGHIA 2014)

Wang Y (2006) Integration of data mining with game theory. In: Wang K, Kovacs GL, Wozny M, Fang M (eds) Knowledge enterprise: intelligent strategies in product design, manufacturing, and management. Springer, Boston, pp 275–280

Wang C, Zheng X (2020) Application of improved time series apriori algorithm by frequent itemsets in association rule data mining based on temporal constraint. Evol Intel 13(1):39–49

Wang K, Kovács G, Wozny M, Fang M (2006) Knowledge enterprise: intelligent strategies in product design, manufacturing, and management: proceedings of PROLAMAT 2006, IFIP TC5 international conference, June 15–17, 2006, Shanghai, China, vol 207. 01

Wang C-H, Zheng L, Yu X, Zheng X (2017) Using fuzzy fp-growth for mining association rules. In: Proceedings of the 2017 inter-
national conference on organizational innovation (ICOI 2017). Atlantis Press, 2017/07, pp 275–279

Wu Z, Dang C, Karimi HR, Zhu C, Gao Q (2014) A mixed 0–1 linear programming approach to the computation of all pure-strategy nash equilibria of a finite n-person game in normal form. Math Prob Eng 2014:640960

Xiangyang S, Ling Z (2016) Apriori parallel improved algorithm based on mapreduce distributed architecture. pp 517–521

Yu X, Zhan R, Tan G, Chen L, Tian B (2020) An improved apriori algorithm research in massive data environment. In: Xu Z, Raymond CKK, Ali D, Reza P, Mohammad H (eds) Cyber security intelligence and analytics. Springer, Cham, pp 843–851

Yin M, Wang W, Liu Y, Jiang D (2018) An improvement of fp-growth association rule mining algorithm based on adjacency table. MATEC Web Conf 189:10012

Yuan X (2017) An improved apriori algorithm for mining association rules. AIP Conf Proc 1820(1):080005

Zeng Y, Yin S, Liu J, Zhang M (2015) Research of improved fp-growth algorithm in association rules mining. Sci Program 2015:910281

Zhu W, Chang L, Sun J, Wu G, Xu X, Xu X (2021) Parallel multipopulation optimization for belief rule base learning. Inf Sci 556:436–458

Zou L, Lin H, Song X, Feng K, Liu X (2021) Rule extraction based on linguistic-valued intuitionistic fuzzy layered concept lattice. Int J Approx Reason 133:1–16

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.