A Two-Armed Bandit Collective for Hierarchical Examplar Based Mining of Frequent Itemsets with Applications to Intrusion Detection

Vegard Haugland, Marius Kjølleberg, Svein-Erik Larsen, and Ole-Christoffer Granmo(✉)

University of Agder, Grimstad, Norway
ole.granmo@uia.no

Abstract. Over the last decades, frequent itemset mining has become a major area of research, with applications including indexing and similarity search, as well as mining of data streams, web, and software bugs. Although several efficient techniques for generating frequent itemsets with a minimum frequency have been proposed, the number of itemsets produced is in many cases too large for effective usage in real-life applications. Indeed, the problem of deriving frequent itemsets that are both compact and of high quality, remains to a large degree open.

In this paper we address the above problem by posing frequent itemset mining as a collection of interrelated two-armed bandit problems. We seek to find itemsets that frequently appear as subsets in a stream of itemsets, with the frequency being constrained to support granularity requirements. Starting from a randomly or manually selected examplar itemset, a collective of Tsetlin automata based two-armed bandit players – one automaton for each item in the examplar – learns which items should be included in the mined frequent itemset. A novel reinforcement scheme allows the bandit players to learn this in a decentralized and online manner by observing one itemset at a time. By invoking the latter procedure recursively, a progressively more fine granular summary of the itemset stream is produced, represented as a hierarchy of frequent itemsets.

The proposed scheme is extensively evaluated using both artificial data as well as data from a real-world network intrusion detection application. The results are conclusive, demonstrating an excellent ability to find frequent itemsets. Also, computational complexity grows merely linearly with the cardinality of the examplar itemset. Finally, the hierarchical collections of frequent itemsets produced for network intrusion detection are compact, yet accurately describe the different types of network traffic present.

1 Introduction

Over the last two decades, frequent itemset mining has become a major area of research, with applications including indexing and similarity search, as well as mining of data streams, web, and software bugs [5].

N.T. Nguyen (Ed.): TCCI XIV 2014, LNCS 8615, pp. 1–19, 2014.
DOI: 10.1007/978-3-662-44509-9_1

The problem of finding frequent itemsets can be formulated as follows. Consider a set I of n items, $I = \{i_1, i_2, \ldots, i_n\}$. A *transaction* T_i, $1 \leq i \leq m$, is defined as a subset of I, $T_i \subseteq I$, collectively referred to as a transaction set: $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$. When an arbitrary set X is a subset of a transaction T_i, $X \subseteq T_i$, one says that T_i supports X. The *support* of X is then simply the number of transactions T_i in $\mathcal{T}$ that supports X, $support(X) = |\{T_i \in \mathcal{T} | X \subseteq T_i\}|$, with $|\cdot|$ denoting set cardinality. The notion of interest in this paper – the *frequency* of an itemset – can then be defined as follows:

Definition 1 (Itemset Frequency). *The frequency of itemset X, $freq(X)$, is defined as the fraction of transactions T_i in $\mathcal{T}$ that supports X:*

$$freq(X) = \frac{|\{T_i \in \mathcal{T} | X \subseteq T_i\}|}{|\mathcal{T}|}.$$

In all brevity, the goal of frequent itemset mining is to produce the itemsets, X_j, that commonly appear in the transactions, T_i, of a transaction set, $\mathcal{T}$. By exploring all of the possible item subsets, X_j, of each transaction $T_i \in \mathcal{T}$, the candidate frequent itemsets are identified and organized in a search space, so that the frequent ones can be efficiently extracted. The frequent itemsets, in turn, form the basis for building associations between itemsets in the form of rules, $A \Rightarrow B$, meaning that whenever itemset A is contained in a transaction, itemset B appears too (with a certain frequency).

The pioneering work of R. Agrawal et al. [2], introduced a framework for finding all of the itemsets surpassing a minimum frequency, and for building association rules that relates the frequent itemsets. It was here the concepts of *support* and *confidence* (the frequency by which the precedent of a rule follows the antecedent of the rule) were introduced. This work triggered a cascade of new results expanding the foundation laid, as briefly sampled here. For a full treatment of current status and future directions, the reader is referred to one of the many surveys on frequent pattern mining, for instance a recent one by Han et al. [5].

One avenue of research involves different kinds of measures besides *support* and *confidence*, such as *lift* [4], *itemset share* [3] and *collective strength* [1]. Furthermore, different approaches to mining with adaptively set minimum *support* thresholds have been investigated. This includes using the Chi-square test for correlation [4] to measure statistical significance of a rule, which allows rules with arbitrarily small *support* to be extracted, as long as sufficient statistical significance is ensured. Another direction of research concerns mining with *constraints* on the itemsets involved, for instance in the form of rule templates [6] or in the form of a wide range of operators such as absence or presence of items [10], extended with *support* constraints [16]. Other work further involves mining of sequential rules to find sequential patterns, including remarkably efficient algorithms such as SPADE [17]. Recently, a number of closely related problems have been addressed, such as mining of structural patterns, high-dimensional datasets, and closed and maximal frequent itemsets, further explored in [5].

Although several efficient techniques for generating frequent itemsets with a minimum frequency have been proposed, as exposed above, the number of itemsets produced is in many cases too large for effective usage in real-life applications. Indeed, the problem of deriving frequent itemsets that are both compact and of high quality, so that they are tailored to perform well in specific real-life applications, remains to a large degree open [5].

1.1 Paper Contributions

The approach introduced in the present paper is quite distinct from the above families of techniques, as briefly explained here and further clarified in the following. A characteristic and typical property of algorithms for frequent itemset mining is that they perform an exhaustive search of the space of candidate frequent itemsets, ranking the found itemsets according to some measure, like *support*. Research advances through the development of better ways of organizing and pruning the candidate itemset spaces, to deal with larger itemset spaces or to produce the frequent itemsets with less computations. Although itemset spaces often can be intelligently organized to support efficient searches, managing the search requires memory consuming data structures and/or time consuming computations, leading to scalability problems. After all, the number of possible itemsets grows exponentially with the number of unique items in the transaction set.

In this paper we introduce a completely different approach to frequent itemset mining that possesses several unique properties:

- In contrast to being based on extensive and dynamically built data structures, the memory footprint of the approach introduced here is both constant and small in size — at most, the scheme only requires one bit per unique item, $i_j \in I = \{i_1, i_2, \ldots, i_n\}$, to organize the search for frequent itemsets. Additionally, each of the n unique items are associated with a dedicated deterministic Learning Automata [13], which is a finite state machine whose state is represented merely by a single integer.
- Furthermore, instead of enumerating itemsets explicitly through exhaustive search, our scheme is light-weighted when it comes to computation. The Learning Automata simply performs a guided random walk in the space of frequent itemset candidates, with each computational step involving an increment or a decrement of the integers associated with the Learning Automata. Yet, the Learning Automata, as a collective, converge rather rapidly and accurately towards producing itemsets that possess a user specified frequency.
- Relying on the above convergence property, the scheme is invoked recursively, with each recursion step producing a new collection of frequent itemset, thus forming a hierarchy of frequent itemsets. In this manner, a progressively larger part of the transaction set is described.
- Also, as a consequence of the latter properties, our scheme operates online, allowing it to process a single transaction at a time, arbitrarily ordered, and with history remembered as part of the Learning Automata states. This is

ideal for instance for network intrusion detection (one of our application domains), where network packets arrive sequentially, in an endless stream.
– Finally, since our scheme is based on Learning Automata, it also handles itemsets with stochastic items, that is, itemsets whose items are randomly included or excluded from the itemset based on some unknown distribution. Again, this is ideal for network intrusion detection since network packets usually contain some fields that for all practical purposes appear as random from the perspective of a single network packet.

We achieve the above properties by posing frequent itemset mining as a collective intelligence problem, modelled as a collection of interrelated *two-armed bandit* problems. The two-armed bandit problem [11] is a classical optimization problem where a player sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting in a random reward. The reward distributions are unknown, and thus, one must balance between exploiting existing knowledge about the arms, and obtaining new information.

Our proposed scheme can be summarized as follows. Starting from a randomly or manually selected examplar transaction, a collective of so-called Tsetlin automata [13] based bandit players – one automaton for each item in the examplar – aims to learn which items should be included in the mined frequent itemset, and which items should be excluded. A novel reinforcement scheme allows the bandit players to learn this in a decentralized and on-line manner, by observing transactions one at a time, as they appear in the transaction stream. The above procedure is invoked recursively, with each recursion step producing a new collection of frequent itemset, thus forming a hierarchy of frequent itemsets. In this manner, a progressively larger part of the itemset stream is described. Since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present transaction and the corresponding decisions of the bandit players, the resulting memory footprint is minimal. Furthermore, computational complexity grows merely linearly with the cardinality of the examplar transaction.

The above hierarchical Tsetlin automata based formulation of frequent itemset mining provides us with four distinct advantages:

1. Any desired target itemset frequency can be obtained without spending any more memory than what is required by the Tsetlin automata in the collective (one integer per automaton).
2. Itemsets are found by the means of on-line collective learning, supporting processing of on-line data streams, such as streams of network packets.
3. An examplar transaction is used to focus the search towards frequent itemsets that are both compact and of high quality, tailored to perform well in real-life applications.
4. Our hierarchical organization of the frequent itemsets supports a progressively more fine-granular summary of the transaction set at hand, as the depth of the hierarchy is increased. This opens up for frequent itemset based network anomaly detection, as will be discussed later in this paper.

1.2 Example Application — Network Anomaly Detection

Network intrusion detection has been a particularly promising application area for frequent itemset mining [14,15]. In so-called network anomaly detection, huge amounts of network packet data needs to be mined so that the patterns of normal traffic can be found, and so that anomalous traffic can be distilled as deviations from the identified patterns. Although not based on frequent itemset mining, the packet byte based anomaly detection approach of Mahoney [8] is particularly fascinating in this perspective because it achieves state-of-the-art anomaly detection performance simply by inspecting 48 bytes from the header of network packets.

In order to investigate to what degree the properties of our bandit problem based approach to frequent itemset mining can be taken advantage of in network anomaly detection, we also introduce a novel packet byte based anomaly detection scheme in this paper. Formulated as a frequent itemset mining problem, each network packet i is seen as a transaction T_i and each byte value from the network packet is seen as an item belonging to the transaction. In other words, in this application we are looking for frequent itemsets consisting of byte-value pairs, such as $\{dstaddr1 : 24, dstaddr2 : 34, tcpflag : 12\}$, which is an itemset that identifies network packets with destination $24.34.*.*$ and with the tcp-flag set to 12.

1.3 Paper Organization

The paper is organized as follows. First, in Sect. 2 we present our decentralized Tsetlin automata based solution to frequent itemset mining, as well as a novel reinforcement scheme that guides the collective of Tsetlin automata towards a given target itemset frequency. The recursive approach to building a hierarchy of frequent itemsets is introduced in Sect. 3. Then, in Sect. 4 we demonstrate the performance advantages of the introduced scheme, including its ability to robustly identify compact itemsets that are useful for summarizing both artificial as well as real-life data. Finally, in Sect. 5 we offer conclusions as well as pointers to further work.

2 A Collective of Two-Armed Bandit Players for Examplar Based Frequent Itemset Mining

We here target the problem of finding frequent itemsets with a given support by *on-line* processing of transactions, taking advantage of so-called transaction *examplars*. To achieve this, we design a collective of Learning Automata (LA) that builds upon the work of Tsetlin and the linear two-action automaton [9,13]. Generally stated, an LA performs a sequence of actions on an *Environment*. The Environment can be seen as a generic *unknown* medium that responds to each action with some sort of reward or penalty, generated *stochastically*. Based on the responses from the Environment, the aim of the LA is to find the action

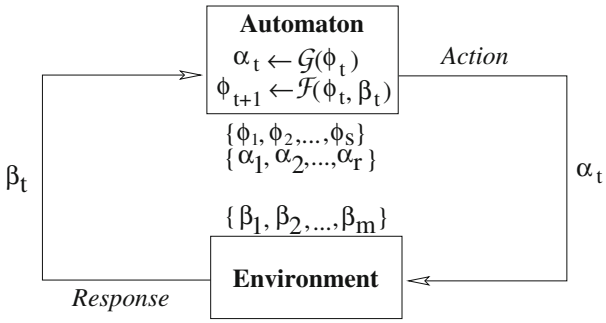

Fig. 1. A Learning Automaton interacting with an Environment

that minimizes the expected number of penalties received. Figure 1 shows the interaction between a LA and the Environment.

As illustrated in the figure, an LA can be defined in terms of a quintuple [9]:

$$\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{G}(\cdot,\cdot)\}.$$

$\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_s\}$ is the set of internal automaton states, $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of automaton actions, and, $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs that can be given to the automaton. An output function $\alpha_t = \mathcal{G}[\phi_t]$ determines the next action performed by the automaton given the current automaton state. Finally, a transition function $\phi_{t+1} = \mathcal{F}[\phi_t, \beta_t]$ determines the new automaton state from the current automaton state as well as the response of the Environment to the action performed by the automaton.

Based on the above generic framework, the crucial issue is to design automata that can learn the optimal action when interacting with the Environment. Several designs have been proposed in the literature, and the reader is referred to [9,12] for an extensive treatment.

2.1 The Item Selector Automaton (ISA)

Our LA based scheme for solving frequent itemset problems is centered around the concept of an *examplar* transaction $T_E \subset I$. With the examplar transaction T_E as a basis, the goal of our scheme is to identify an itemset $X \subseteq T_E$ whose frequency, $freq(X)$, is equal to a specific target frequency γ.

At the heart of our scheme we find an Item Selector Automaton (ISA). In brief, for each item i_j in T_E, a dedicated ISA, based on the Tsetlin automaton [13], is constructed, having:

- States: $\underline{\Phi} = \{-N-1, -N, \ldots, -1, 0, \ldots, N-2, N\}$.
- Actions: $\underline{\alpha} = \{Include, Exclude\}$.
- Inputs: $\underline{\beta} = \{Reward, Penalty\}$.

Figure 2 specifies the $\mathcal{G}$ and $\mathcal{F}$ matrices.

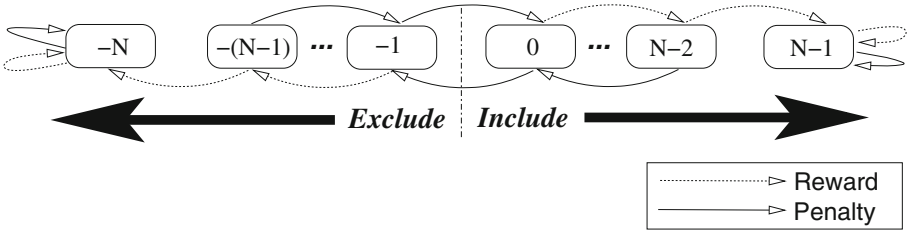

Fig. 2. An ISA choosing between including or excluding an item from candidate frequent itemsets

The $\mathcal{G}$ matrix can be summarized as follows. If the automaton state is positive, then action *Include* will be chosen by the automaton. If on the other hand the state is negative, then action *Exclude* will be chosen. Note that since we initially do not know which action is optimal, we set the initial state of the ISA randomly to either '-1' or '0'.

The state transition matrix $\mathcal{F}$ determines how learning proceeds. As seen in the graph representation of $\mathcal{F}$ found in the figure, providing a *reward* input to the automaton *strengthens* the currently chosen action, essentially by making it less likely that the other action will be chosen in the future. Correspondingly, a *penalty* input *weakens* the currently selected action by making it more likely that the other action will be chosen later on. In other words, the automaton attempts to incorporate past responses when deciding on a sequence of actions.

Note that our ISA described above deviates from the traditional Tsetlin automaton in one important manner: State $-N$ and state $N-1$ are absorbing. This allows the ISA to converge to a single state, rather than to a distribution over states, thus artificially introducing an unambiguous convergence criterion.

2.2 Reinforcement Scheme

Since each item i_j in the transaction examplar T_E is assigned a dedicated ISA, ISA_j, we obtain a collective of ISA. The reinforcement scheme presented here is incremental, processing one transaction at a time at discrete time steps. At each time step s, a transaction $T_i \in \mathcal{T}$ is presented to the collective of ISA, whose responsibility is to propose a candidate itemset $X(s)$ for that time step. By on-line processing of the transactions, the goal of the ISA is to converge to proposing an itemset X^* that is supported with frequency, $freq(X^*) = \gamma$, with probability arbitrarily close to 1.

To elaborate, each automaton, ISA_j, chooses between two options at every time step s: shall its own item i_j be included in $X(s)$ or shall it be excluded? Based on the decisions of the ISAs as a collective, a candidate itemset $X(s)$ for time step s is produced. A response from the Environment is then incurred as follows. First it is checked whether the present transaction T_i supports $X(s)$, and based on the presence or absence of support, each ISA_j is rewarded/penalized according to the following novel reinforcement scheme.

The novel reinforcement scheme that we propose rewards an automaton ISA_j based on the decision of the automaton at time step s and based on whether the present transaction T_i supports the resulting candidate itemset $X(s)$. In brief, if ISA_j decides to include item i_j in $X(s)$, we have two possibilities. If T_i supports $X(s)$, ISA_j is rewarded. On the other hand, if T_i does not support $X(s)$, then ISA_j is randomly penalized with probability $r = \frac{\gamma}{1-\gamma}$ (assuming $\gamma < 0.5$). The other decision ISA_j can make is to exclude item i_j from $X(s)$. For that decision, ISA_j is randomly rewarded with probability $r = \frac{\gamma}{1-\gamma}$ if T_i does not support $X(s) \cup \{i_j\}$. On the other hand, if T_i supports $X(s) \cup \{i_j\}$, then the ISA is penalized.

The above reinforcement scheme is designed to guide the collective of learning automata as a whole towards converging to including/excluding items in $X(s)$ so that the frequency of $freq(X(s))$ converges to γ, with probability arbitrarily close to 1.

Note that because multiple variables, and thereby multiple ISA, may be involved when constructing the frequent itemset, we are dealing with a game of LA [9]. That is, multiple ISA interact with the same Environment, and the response of the Environment depends on the actions of several ISA. In fact, because there may be conflicting goals among the ISA involved, the resulting game is competitive. The convergence properties of general competitive games of LA have not yet been successfully analyzed, however, results exist for certain classes of games, such as the Prisoner's Dilemma game [9].

In order to maximize speed of learning, we initialize each ISA randomly to either the state '-1' or '0'. In this initial configuration, the actions will be switched relatively quickly because only a single state transition is necessary for a switch. Accordingly, the joint state space of the ISA is quickly explored in this configuration. However, as learning proceeds and the ISA move towards their boundary states, i.e., states '$-N$' and '$N-1$', the exploration calms down. Accordingly, the search for a solution to the frequent itemset problem at hand becomes increasingly focused.

Furthermore, note that we keep a time step counter for each ISA. When a certain cut off threshold has been achieved, we force one of the ISA to converge if it has not yet done so. This enforcement resets the counters of the other ISA, allowing them to adapt to the new configuration. The purpose of this mechanism is to increase convergence speed in ambiguous decision making cases where two different actions provide more or less the same feedback.

3 A Hierarchical Collective of Two-Armed Bandit Players

The two-armed bandit problem based scheme proposed in the previous section seeks to produce an itemset X that is supported by a fraction γ of the transactions T_i contained in a transaction set $\mathcal{T}$. In this section, we take advantage of the latter scheme in order to produce a compact summary of the *complete*

transaction set $\mathcal{T}$. The summary is formulated as a hierarchy of frequent itemsets. At each "level" in the hierarchy, every transaction in $\mathcal{T}$ is supported by at least one frequent itemset, and with each level descended, the granularity of the frequent itemsets increases. In this manner, multiple degrees of granularity, from coarse grained to fine grained, are concurrently maintained in the hierarchy. We achieve this by the means of a recursive algorithm, presented below. We explore the expressive power of this hierarchical approach in the next section, using the frequent itemset hierarchy as the basis for a network anomaly detection system.

Algorithm 1. Hierarchy Construction

Input: Transactions $\mathcal{T}$; Maximal coarseness $\gamma_{\max}$
Output: Hierarchy H of frequent itemsets

```
Queue Q; # Queue for organizing breadth first construction of hierarchy
Hierarchy H; # Hierarchy of frequent itemsets
# Empty frequent itemset {} as root, supporting all transactions in T
Q.enqueue([{}, T, γmax]);
H.root({});
# Breadth first expansion of hierarchy nodes, starting from root
while not Q.empty() do
    [X_P, T_P, γ_P] ← Q.dequeue();
    # Identifies candidate children nodes of parent P
    C ← Frequent_Itemset_Generation([X_P, T_P, γ_P]);
    # Expands parent P with children nodes who has coarseness greater than or
    equal to the minimum coarseness sought:
    for [X_C, T_C, γ_C] ∈ C do
        if γ_C ≥ γmin then
            Q.enque([X_C, T_C, γ_C]);
            H.addChild(X_P, X_C);
        end if
    end for
end while
return H;
```

As seen in Algorithm 1, the input to the hierarchy construction algorithm is the transaction set $\mathcal{T}$ to be summarized, as well as the maximum, $\gamma_{\max}$, and the minimum, $\gamma_{\min}$, coarseness sought. Each node P in the hierarchy built is associated with a frequent itemset X_P, a transaction set $\mathcal{T}_P$, and a coarseness γ_P, organized as a triple $[X_P, \mathcal{T}_P, \gamma_P]$. As initialization, the root node is assigned an empty itemset, the complete transaction set $\mathcal{T}$, and the maximum coarseness $\gamma_{\max}$. The hierarchy is then built breadth first, based on a queue of hierarchy nodes, initialized to contain the root node. Each node $[X_P, \mathcal{T}_P, \gamma_P]$ dequeued is expanded producing a collection of candidate children $[X_C, \mathcal{T}_C, \gamma_C] \in \mathcal{C}$. If the coarseness γ_C of a triple is greater than or equal to $\gamma_{\min}$, the triple is enqueued and the frequent itemset X_C is added to the hierarchy H as a child of X_P.

The scheme for expanding a node P to produce candidate children $C \in \mathcal{C}$ is shown in Algorithm 2. As seen, the candidate children are built one by one,

Algorithm 2. Frequent Itemset Generation

Input: Parent itemset X_P; Parent transactions $\mathcal{T}_P$; Parent coarseness γ_P
Output: Collection $\mathcal{C}_P$ of itemset-transactions-coarseness triples $[X_i, \mathcal{T}_i, \gamma_i], i \in \{1, \ldots, n\}$, such that $\mathcal{T}_P = \bigcup_{i=1}^{n} \mathcal{T}_i$ and for $i \neq j : \mathcal{T}_i \cap \mathcal{T}_j = \emptyset$

$\mathcal{C}_P \leftarrow \{\}$; # *Collection of child triples empty initially*
$\mathcal{T}_R \leftarrow T_P$; # $\mathcal{T}_R$ *keeps track of remaining unsupported transactions*
$\gamma_C \leftarrow \gamma_P$; # γ_C *keeps track of current level of coarseness*
# *Keeps producing child triples until all transactions in* $\mathcal{T}_P$ *are supported* ($\mathcal{T}_R \neq \emptyset$)
while $\mathcal{T}_R \neq \emptyset$ **do**
    # *Generates itemsets until one more fine-grained than the parent itemset is found*
    $X_C = \text{ISA\_Collective}(\mathcal{T}_R, \gamma_C)$; # *Invokes ISA collective on remaining transactions*
    if $X_C \subseteq X_P$ **then**
        $\gamma_C \leftarrow 0.999 \cdot \gamma_C$; # *Reduces target coarseness*
    else
        $\mathcal{C}_P \leftarrow \mathcal{C}_P \cup [X_C, \{T_i \in \mathcal{T}_R | X_C \subseteq T_i\}, \gamma_C]$; # *Adds new triple to child set*
        $\mathcal{T}_R \leftarrow \{T_i \in \mathcal{T}_R | X_C \not\subseteq T_i\}$; # *Removes transactions supporting* X_C
    end if
end while
return $\mathcal{C}_P$;

using the ISA collective proposed in the previous section. The ISA collective is repeatedly invoked on the part of the parent transaction set $\mathcal{T}_P$ that is still unsupported by the current collection of generated children. Since the children itemsets are to offer a finer granularity than the parent itemset, a child itemset that is a superset of the parent itemset is rejected, triggering a reduction in coarseness γ_C sought. If the child itemset is a superset of the parent itemset, on the other hand, it is added to the collection of candidate children.

In combination, the above two algorithms ensure that the coarseness, γ_C, of a child C always is less than or equal to the coarseness, γ_P, of its parent, P. Furthermore, all the transactions $T_i \in \mathcal{T}_P$ of a parent is supported by at least one of the frequent itemsets of its children $\mathcal{C}_P$, $\forall T_i \in \mathcal{T}_P, \exists [X_C, \mathcal{T}_C, \gamma_C] \in \mathcal{C}_P : X_C \subseteq T_i$. Note that the children transaction sets $\mathcal{T}_C$ jointly contain all of the transactions in the parent transaction set $\mathcal{T}_P$. These properties are taken advantage of in the next section to build a rule based network packet anomaly detection system.

4 Empirical Results

In this section we evaluate our proposed scheme using both artificial data as well as data from a real-world network intrusion detection application. We first evaluate the ISA collective separately, before we evaluate the hierarchical invocation of ISA collectives, as presented in the previous section.

4.1 Artificial Data

For the evaluation on artificial data we constructed a collection of transactions in a manner that by selecting the correct itemset X, one can achieve a frequency, $freq(X)$, of either $0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0$. The purpose is to challenge the scheme by providing a large number of frequency levels to choose among, with only one of these being the target frequency γ that our scheme must converge to. By varying the target frequency γ in the latter range, we also investigate the robustness of our scheme towards low, medium, and high frequency itemsets.

We here report an ensemble average after conducting 100 runs of our scheme. Given a target frequency γ, each run produces an itemset X^*, with X^* being supported by an actual frequency $freq(X^*)$. By comparing the sought target frequency γ with the achieved frequency $freq(X^*)$, the convergence accuracy of our scheme is revealed.

We first study convergence accuracy when any subset $X \subset T_E$ of the examplar transaction T_E either have a support frequency, $freq(X)$, equal to the target frequency γ, 1, or 0. Then the goal of the ISA collective is to identify the subset $X \subset T_E$ with frequency γ. As seen in Fig. 3, our scheme achieves this goal with remarkable accuracy.

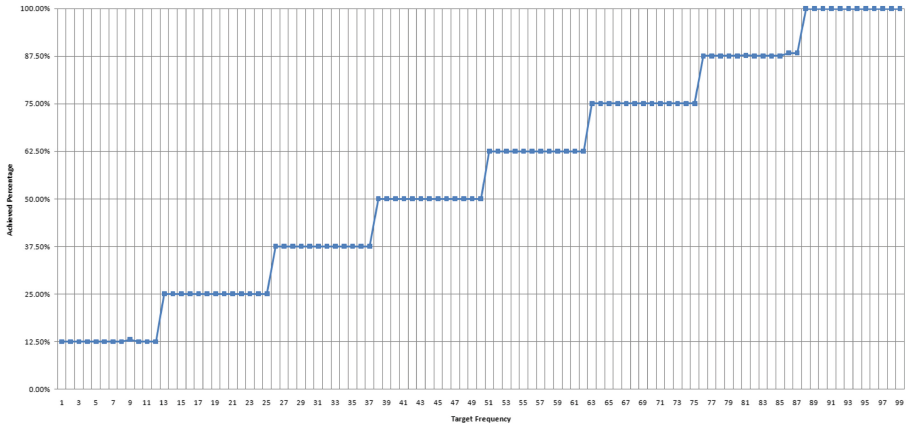

Fig. 3. Achieved percentage of transactions supported by produced itemset (y-axis) using a specific examplar transaction, for varying target frequencies γ (x-axis)

We observe that for any of the target frequencies γ listed in the figure, on average our ISA collective identifies itemsets X^* with frequencies $freq(X^*) \in \{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$ that either equals γ or surpasses γ with the least possible amount: $freq(X^*) \geq \gamma \wedge freq(X^*) - 0.125 < \gamma$.

When using a generic transaction examplar T_E instead — one that contains item subsets $X \subseteq T_E$ of any arbitrary frequency level $freq(X) \in \{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$, the challenge increases. The ISA collective then

also have the option to produce frequencies in close vicinity of the target frequency γ. Figure 4 reports the resulting convergence accuracy, and as seen, it is now more difficult for the collective of ISA to always produce an itemset X with a transaction support frequency exactly equal to γ. Still, the itemsets produced are always close to a nearby neighbor of γ in $\{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$.

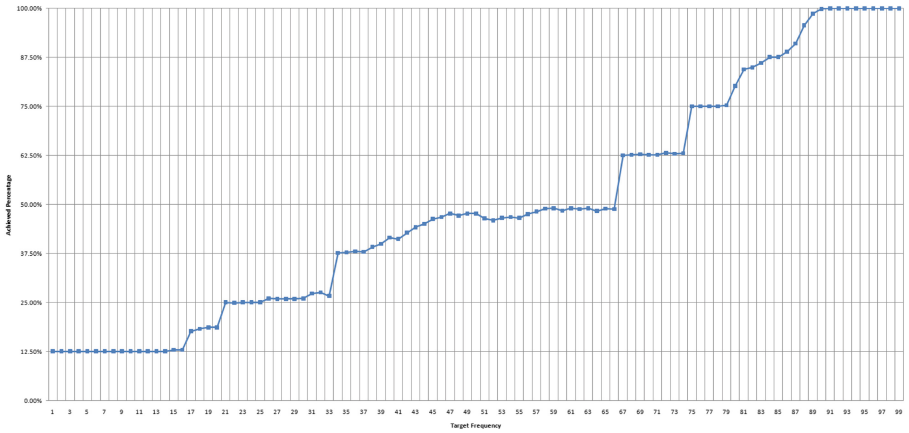

Fig. 4. Achieved percentage of transactions supported by produced itemset (y-axis) using a generic examplar transaction, for varying target frequencies γ (x-axis)

4.2 DARPA Intrusion Detection Evaluation Data Set

To evaluate the ISA collective scheme on a real life application, we have implemented a network intrusion detection system, with the ISA collective at its core. Briefly explained, we analyze the last 40 bytes of each network packet header in combination with the first 8 bytes of the transport layer payload, as also done in NETAD [8].[1] Essentially, we see each network packet as a transaction, and byte-value pairs from a network packet are seen as items.

We intend to detect network attacks by first learning a collection of frequent itemsets that describe the key features of normal network traffic – and based on these frequent itemsets, reporting network packets as anomalous when they do not support any of the learned frequent itemsets. We use the 1999 DARPA Intrusion Detection Evaluation data set [7] for training and testing our system. This data set simulates network traffic occurring in a small US Air Force base that is connected to the Internet. The internal network is connected to the Internet by a CISCO router, and network traffic is captured on both sides of this router. The captured traffic is made available in separate files based on the origin of the network traffic — outside, from the Internet, or inside, from the internal network.

[1] Note that in contrast to NETAD, we analyze both ingoing and outgoing network packets, for greater accuracy.

During training, we use one week of normal traffic data, learning one frequent itemset at a time by randomly picking examplar transactions (network packets) from the normal traffic data. Each time the collective of ISA converges to a new frequent itemset, all network packets that support this itemset are removed from the normal traffic data, and the procedure is repeated on the network packets remaining, to learn each kind of traffic being present. This step produces the first level of children nodes in the hierarchy, the children of the root. Each of the produced children are now associated with a disjoint subset of the network packets, and by recursively invoking the ISA collective on these subsets, a more fine granular representation of the network packets are found. Again, coarseness is controlled by gradually reducing γ from $\gamma_{\max}$ to $\gamma_{\min}$.

As an example, a frequent itemset hierarchy generated from the network packets of Week 1 of the DARPA data set is shown in Fig. 5. In brief, multiple children are needed to support the parent transactions (network packets). Furthermore, node expansion stops when the transaction set of a node contains less than 5 % of the total population of network packets. Notice the hierarchy compactness achieved despite the huge amount of network packets present in the DARPA data set and despite the fine grained target granularity ($\gamma_{\min} = 0.05$) sought.

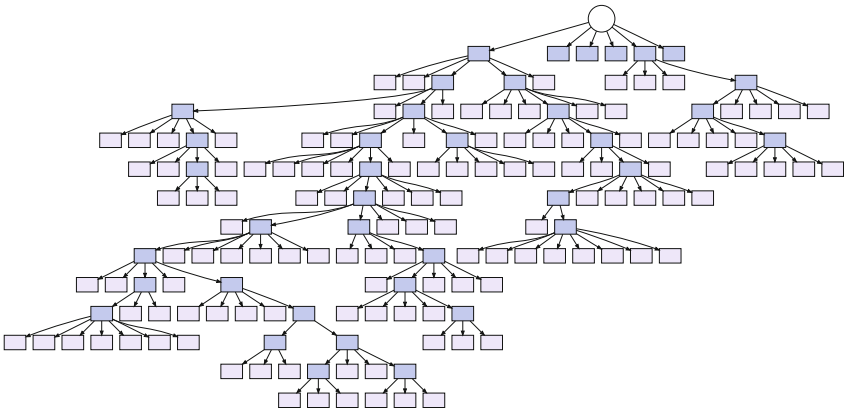

Fig. 5. Example of a frequent itemset hierarchy that was generated based on one week of training data from the 1999 DARPA IDS Evaluation Set. Each node in the hierarchy corresponds to a single frequent itemset

To gain further insight into the hierarchy generation process, Fig. 6 reports the total number of iterations spent by each ISA, grouped by network packet byte position. The figure shows that certain packet bytes and their values are "easier" to learn than others. Basically, we observe that the greater the diversity of values associated with a network packet byte (high entropy), the more iterations are needed to decide upon whether to include the corresponding item in the frequent itemset being constructed by the ISA collective. For instance, the

values contained in the content bytes (byte 40 to 48) of a packet are by nature highly diverse, and thus require more time for learning an appropriate decision.

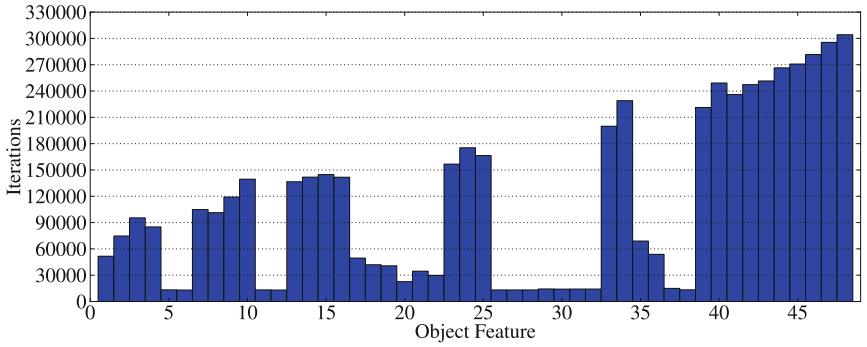

Fig. 6. Total number of iterations performed by ISAs to generate complete hierarchy, grouped by network packet byte position

For testing, the second week of the DARPA data set is used. The network packets from this week contain attacks, interspersed among the normal traffic. If a network packet in the second week of data does not support any of the learned frequent itemsets, it is reported as an anomaly. Table 1 contains a few representative examples of frequent itemsets, called Rules, and which kind of attacks they allow us to detect. As seen, each Rule consists of selected bytes from a packet, combined with a hexadecimal representation of the corresponding byte value. Thus, considering the first row of the table, network packets of the so-called ps-attack do not support the frequent itemset {ver+ihl:0x45, frag1:0x40, frag2:0x00, proto:0x06, srcport1:0x00, tcphl:0x50, urgptr1:0x00, urgptr2:0x00}, and are therefore reported as anomalies.

For computational efficiency, each Rule (frequent itemset) is implemented as a classification tree, as illustrated in Fig. 7. Each node in the classification tree corresponds to a specific item of the frequent itemset. As explained in Sect. 1, for the intrusion detection application an item refers to the combination of packet byte position and packet byte value, represented as a bit string. A non-match marks that the transaction being analyzed is unsupported, and accordingly, can be "blocked" by the network anomaly detection system unless another matching itemset can be found. For the frequent itemset reported in the figure, the target coarseness γ was set to 0.5, and as can be seen, the resulting Rule matches a percentage of 49.9 % of the network packets, i.e., very close to the target.

To further demonstrate the effect minimum coarseness $\gamma_{\min}$ has on anomaly detection sensitivity, Fig. 8 plots the *anomaly scores* for three unique frequent itemset hierarchies, each with different granularity constraints. An anomaly score was introduced to differentiate between different levels of the hierarchy — a packet not being supported by the more coarse frequent itemsets closer to the root is more aggravating than minor deviations closer to the leaf nodes. In order

to capture such effects, we trace a path through the hierarchy, starting from the root, and continuing, level-by-level, with the child that supports the current transaction. When the path reaches a node P whose children do not support the current transaction, the transaction is assigned the anomaly score:

$$48 - 2 \cdot |X_P|.$$

In other words, the smaller the frequent itemset the larger the anomaly score. For instance, a transactions that stops being supported immediately after the root, will have the maximum anomaly score possible, which is 48. The plots in Fig. 8 highlights particularly anomalous packets as spikes, while packets recognized as normal possessing anomaly score less than or equal to zero. Notice that as the hierarchy coarseness increases, from top to bottom in the figure, the number of spikes decreases. Although the number of detected anomalies is at its highest with low coarseness, all of these anomalies are not necessarily *true attacks* (finer granularity may lead to *false* alarms). Thus, to quantify the anomaly detection capability of our scheme, Table 2 provides the estimated detection probability for attacks (detection rate), as well as the average number of false alarms produced for a whole week of network traffic. The table shows that the best detection rates were achieved by the hierarchies that were trained on Week 1 data, and which "leaves" possessed a minimum granularity $\gamma_{\min} = 0.05$.

Table 1. Transaction examplars

Rule	Attacks
ver+ihl:0x45, frag1:0x40, frag2:0x00, proto:0x06, srcport1:0x00, tcphl:0x50, urgptr1:0x00, urgptr2:0x00	ps
ver+ihl:0x45, dscp:0x00, frag1:0x00, frag2:0x00, proto:0x06, tcphl:0x50, urgptr1:0x00, urgptr2:0x00	ps
ver+ihl:0x45, dscp:0x00, len:0x00, frag1:0x40, frag2:0x00, ttl:0x40, proto:0x06, dstaddr1:0xac, dstaddr2:0x10, dstport1:0x00, dstport2:0x17, tcphl:0x50, recwd1:0x7d, recwd2:0x78, urgptr1:0x00, urgptr2:0x00, pld1:0x00, pld5:0x00, pld7:0x00, pld8:0x00, pld9:0x00	ps, guesstelnet, sendmail

To conclude this investigation, Table 3 provides an overview of the attack types that remain undetected by the hierarchies that we produced. Note that only network packets originating from outside the firewall were used in these tests. In other words, the attacks that take place purely on the inside of the firewall were left undetectable as a consequence of the experimental setup. Also undetectable are the locally executed attacks that do not generate any network traffic. This leaves us with only two attacks that could have been detected, but were not: one instance of *guesspop* and one instance of *land* (the other instance is on the inside), which reveals a remarkable detection rate/false positive ratio.

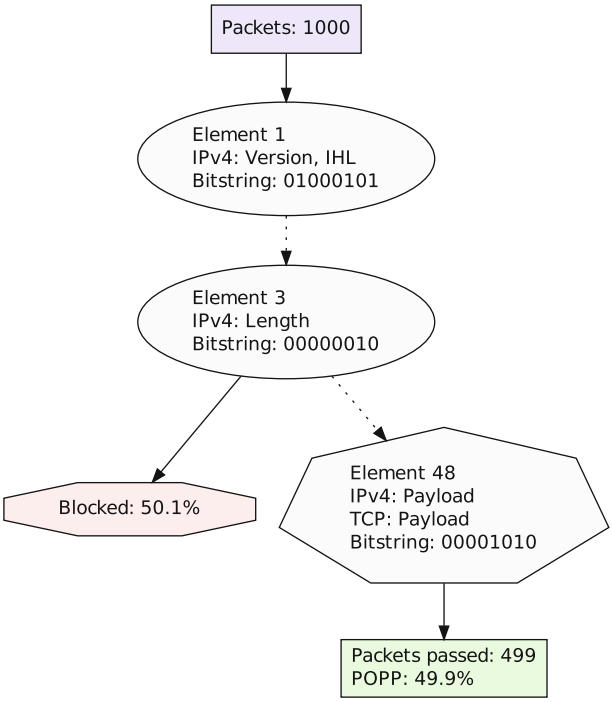

Fig. 7. Classification tree based implementation of frequent itemset transaction support determination

As a final note, observe that since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present itemset and the corresponding decisions of

Table 2. Results from IDS evaluation using the DARPA set

Parameters		Rate			False positives		
Training data	Granularity	avg	min	max	avg	min	max
Week 1 outside	1 %	0.56	0.55	0.57	412.67	392	453
	5 %	0.73	0.72	0.74	123	120	129
	10 %	0.43	0.42	0.44	61.67	52	69
Week 3 outside	1 %	0.54	0.50	0.58	427	426	429
	5 %	0.63	0.57	0.66	88.33	81	96
	10 %	0.54	0.53	0.55	59.67	58	61
Week 1 + 3 outside	1 %	0.54	0.52	0.56	418	404	436
	5 %	0.61	0.57	0.66	82	68	94
	10 %	0.46	0.43	0.53	47.33	40	61

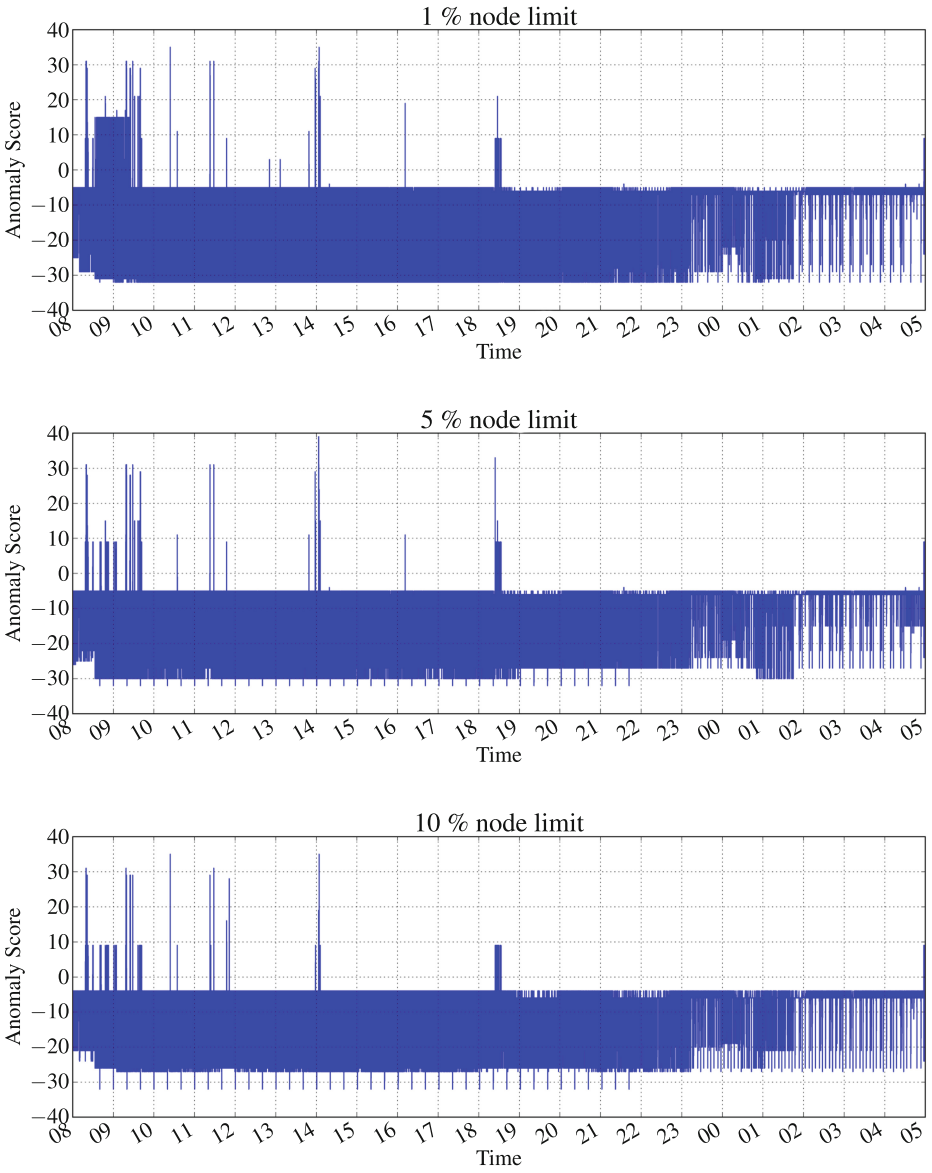

Fig. 8. Comparison of anomaly distribution between three frequent itemset hierarchies of different granularity.

Table 3. Undetected DARPA attack types

Alias	Instances	Console/remote	Inside/outside	Stealthy
anypw	2	Console	–	No
dict	6	Remote	Inside	No
guesspop	1	Remote	Outside	No
illegalsniffer	14	Remote	Inside	Mix
land	2	Remote	Both	No
ntfsdos	3	Console	–	No
resetscan	1	Remote	Inside	Yes
sshprocesstable	12	Remote	Inside	No

the bandit players, the resulting memory footprint is minimal (usually one byte per ISA). Furthermore, computational complexity grows merely linearly with the cardinality of the examplar itemset.

5 Conclusion

In this paper we have addressed frequent itemset mining by the means of a collective of so-called Item Selector Automata (ISA). By on-line interaction with a stream of transactions, the collective of ISA decides which items should be excluded and which should be included in a frequent itemset, with items being chosen from a randomly or manually selected examplar itemset. A novel reinforcement scheme guides the ISA towards finding a candidate itemsets that is supported by transactions with a specified frequency. By invoking the latter procedure recursively, a progressively more fine granular summary of the itemset stream is produced, represented as a hierarchy of frequent itemsets.

Since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present itemset and the corresponding decisions of the bandit players, the resulting memory footprint is minimal. Furthermore, computational complexity grows merely linearly with the cardinality of the examplar itemset.

In extensive evaluation using both artificial data and data from a real-world network intrusion detection application, we find the results quite conclusive, demonstrating that the ISA collective possesses an excellent ability to find frequent itemsets at various levels of support. Furthermore, the sets of frequent itemsets produced for network intrusion detection are compact, yet accurately describe the different types of network traffic present, allowing us to detect attacks in the form of anomalies.

In our further work, we intend to develop formal convergence proofs for the ISA collective. We are also presently investigating a hierarchical scheme for organizing ISA collectives, with the purpose of increased scalability.

References

1. Aggarwal, C.C., Yu, P.S.: A new framework for itemset generation. In: PODS 98, Symposium on Principles of Database Systems, Seattle, WA, USA, pp. 18–24 (1998)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Washington D.C., May 1993, pp. 207–216 (1993)
3. Barber, B., Hamilton, H.J.: Extracting share frequent itemsets with infrequent subsets. Data Min. Knowl. Disc. **7**, 153–185 (2003)
4. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, May 1997, pp. 255–264 (1997)
5. Han, J., Chen, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Min. Knowl. Disc. **15**(1), 55–86 (2007)
6. Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.I.: Finding interesting rules from large sets of discovered association rules. In: Adam, N.R., Bhargava, B.K., Yesha, Y. (eds.) Third International Conference on Information and Knowledge Management (CIKM'94), pp. 401–407. ACM Press (1994)
7. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Comput. Netw. **34**(4), 579–595 (2000)
8. Mahoney, M.V.: Network traffic anomaly detection based on packet bytes. In: Proceedings of ACM-SAC 2003, pp. 346–350. ACM (2003)
9. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice Hall, Englewood Cliffs (1989)
10. Srikant, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Heckerman, D., Mannila, H., Pregibon, D., Uthurusamy, R. (eds.) Proceedings of the 3rd International Conference Knowledge Discovery and Data Mining (KDD-97), pp. 67–73. AAAI Press (1997)
11. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
12. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers, Dordrecht (2004)
13. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, New York (1973)
14. Vaarandi, R., Podins, K.: Network IDS alert classification with frequent itemset mining and data clustering. In: Proceedings of the 2010 IEEE Conference on Network and Service Management. IEEE (2010)
15. Wang, H., Li, Q.-H., Xiong, H., Jiang, S.-Y.: Mining maximal frequent itemsets for intrusion detection. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.) GCC 2004 Workshops. LNCS, vol. 3252, pp. 422–429. Springer, Heidelberg (2004)
16. Wang, K., He, Y., Cheung, D.W.: Mining confident rules without support requirement. In: Proceedings of the Tenth International Conference on Information and Knowledge Management, pp. 89–96. ACM Press, New York (2001)
17. Zaki, M.: Spade: an efficient algorithm for mining frequent sequences. Mach. Learn. **42**(1–2), 31–60 (2001)