Real-time Sensory Pattern Mining for Autonomous Agents

Pedro Sequeira[1] and Cláudia Antunes[2]

[1] INESC-ID / IST, Av. Prof. Dr. Cavaco Silva, 2744-016 Porto Salvo, Portugal,
pedro.sequeira@gaips.inesc-id.pt
[2] Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, Portugal,
claudia.antunes@ist.utl.pt

Abstract. Autonomous agents are systems situated in dynamic environments. They pursue goals and satisfy their needs by responding to external events from the environment. In these unpredictable conditions, the agents' adaptive skills are a key factor for their success. Based on previous interactions with its environment, an agent must learn new knowledge about it, and use that information to guide its behavior throughout time. In order to build more believable agents, we need to provide them with structures that represent that knowledge, and mechanisms that update them overtime to reflect the agents' experience. Pattern mining, a subfield of data mining, is a knowledge discovery technique which aims to extract previously unknown associations and causal structures from existing data sources. In this paper we propose the use of pattern mining techniques in autonomous agents to allow the extraction of sensory patterns from the agent's perceptions in real-time. We extend some structures used in pattern mining and employ a statistical test to allow an agent of discovering useful information about the environment while exploring it.

Key words: autonomous agents, adaptation, learning, pattern mining, knowledge discovery

1 Introduction

There are several definitions of autonomous agents, and several attempts to define the requirements that a designer must meet when building those agents [4]. Nevertheless, there a few characteristics which are common to those definitions of agents, specifically that they are systems situated in dynamic environments, which have and actively pursue some goals and satisfy their needs by autonomously responding to external events from that environment [4][9]. The characteristics of the environment make restrictions on the sensations perceived, and shape the actions performed, the knowledge constructed, and the decisions that are taken by the agent at each moment. Because those environments can be dynamic and unpredictable, the agent must have some mechanisms to distinguish the features that are perceived from it, focusing its attention in those

that seem more promising to achieve its goals, and ignoring others that do not [4]. This ability of adapting to and learning new knowledge from the environment while interacting with it in real-time defines restrictions and requirements when building autonomous agents with such capabilities [9]. Namely, we need to provide them with some structures that represent the acquired knowledge about the environment, and mechanisms that update these representations overtime to reflect the agent's interaction experience.

Data mining encloses a set of techniques to extract previously unknown and possible useful information from data [5]. One of such techniques is transactional pattern mining, which extracts frequent associations and causal structures among sets of objects or items from several transactions.

In this paper we propose the use of pattern mining techniques within the autonomous agents paradigm in order to provide those agents with the ability of discovering associative patterns in their perceptions while they interact with their environment, i.e., in real-time. These patterns constitute the agents' knowledge about its world taken from the regularities that are perceived from its experience. Later on they can be useful for the agent to form concepts about the environment and by setting expectations about future events. To achieve that we extended some structures used in pattern mining to represent the discovered knowledge. We also adopted a statistical test to detect significant sensory information for the agent to discover useful facts about its world, and came up with some heuristics and algorithms to update and maintain the knowledge structures in real-time, while the agent explores it.

This paper is organized as follows: the next section gives an overview of the general idea and motivation behind the work presented, and define the research problem we are trying to solve. In the following section we present several extensions to related work as possible solutions to the problem. In section 5 we present the tests used to validate the proposed solutions and a comparative analysis relating them. Finally we draw some conclusions and possible future extensions for the work.

2 The Problem

In this section we describe the background motivation and research problem behind the current work, introduce the pattern mining problem and explain how the analogy between these problems and the autonomous agents' paradigm can be made. Next we characterize the problem to be solved and define a set of requirements to be satisfied.

2.1 Background

In a previous work [10], Sequeira *et al.* presented *SOTAI* (*Smart ObjecT-Agent Interaction*) framework to help autonomous agents identify the set of possible interactions with unknown objects of the environment, based on previous experiences with other objects. The approach is based on the notion of Gibson's

affordances [6] which can be defined as the interaction opportunities which are transmitted by the objects to the agents taking into account their interaction capabilities. Within the *SOTAI* framework, an agent is provided with a set of sensors from which it perceives its environment. At each moment one sensor can contain a set of sensations which are modeled as symbolic qualities or features of the perceived objects from the environment. For example, if an agent interacts with an orange then it will receive the symbol *orange* in the *color* sensor, *spherical* in the *shape* sensor, the symbol *soft* to the *touch-texture* sensor, etc.

The framework is based in Cohen *et al.*'s *block-building* approach [3]. It allows the construction of small pieces of information called *Base Fluents*, which are pairs of *Sensations* that occur frequent and simultaneously in the agent's sensors. Using a Chi-square [11] statistical test, one can determine whether two monitored *Sensations* are correlated, and also their association strength by calculating their Phi-coefficient [11]. If an association is detected, *Base Fluents* are created and later tested in pairs to determine new associations. The learning process occurs in real-time while the agent interacts with its environment, building new and larger blocks of information (*Fluents*) from smaller structures. More implementation details can be found in [10].

To test the knowledge generation mechanisms, a test application (*Sotai-Tester*) was created where an agent explores and interacts with a simulated environment composed of several objects in a random manner. It selects at each decision moment an object to interact with and receives sensations from the environment. For example, when it *sees* an object, the agent takes its *shape* and *color* features from the object's internal description, when it *touches* it receives the object's tactile information, when it *eats* it takes flavor and energy properties, etc.

2.2 The Pattern Mining Analogy

Pattern mining is a technique that is used to discover frequent patterns, associations, correlations, or causal structures among sets of items or objects [1]. In *transactional pattern mining*, a transaction database (DB) is a set of transactions, each of which containing a set of items describing objects or events that occur simultaneously, at a given point in time. The goal of the pattern mining algorithms is to discover all the maximal sets of items that occur together in a significant number of transactions in the DB. *Itemsets* are considered *patterns* if the number of transactions in the DB where they occur is greater than a minimal threshold value, the pre-established *support*.

Considering the characteristics of the agents being modeled within the *SOTAI* framework, one can envisage a way of applying some of these pattern mining algorithms within an autonomous agent paradigm. In the case of such agents, at each time every sensor will have a certain stimulus represented by some symbol. In the knowledge-discovery language these symbols constitute categorical nominal attribute data or alphabets as they describe discrete values with no ordering between them. Moreover, if we consider the symbols as being items, then at each time the set of all the stimuli in the agent's sensors (it's current perception state)

can be considered as a transaction. In this context, a DB can be defined as the whole record of perception states of the agent during all of its execution time.

2.3 Problem Definition

Having described the perception mechanism of the agent that we are modeling in the context of pattern mining (as a set of items and transactions), we can define the problem of discovering knowledge from the agent's interactions with its environment as a problem of mining patterns from sets of transactions from a database.

More specifically, we can better define the problem as finding *synchronous sensory patterns*: sets of stimuli that occur frequently and simultaneously within the agent's sensors that reveal some regularities of its environment. Because this discovering process directly depends on the perceptions made, the number and set of patterns discovered will depend on each agent's interaction experience even if the environment and its conditions (the objects and its features) are the same. From this definition, the mining algorithms to apply have to follow some requirements:

- Discover sensory patterns in real-time as the agent interacts with the environment;
- Discover the maximum amount of sensory patterns that indicate useful information to be used by the agent at which it should focus its attention;
- Maintain within the knowledge structure the maximum number of patterns so that they can be easily accessed and used by the agent to make decisions;
- Use the minimal amount of resources (both storage and time consumption).

3 Sensory Pattern Mining for Autonomous Agents

In this section we present several approaches to the problem defined in the previous section. We start by a quick overview of the *FP-Growth* algorithm and envisage possible modifications of the algorithm and its structures so that it can be used in mining patterns in real time. Finally we propose the use of the Jaccard index statistic as a way to retrieve more and meaningful patterns from the sensory data.

3.1 Transactional Pattern Mining

As described above, at each instant the agent's perception state describes a set of stimuli that can be defined as a transaction which can be provided to pattern mining algorithms. The best well-known algorithm within the area of transactional pattern mining is the Apriori algorithm introduced by Agrawal *et al.* [1]. It iteratively generates the set of candidate patterns (*itemsets*) of some length k from the set of frequent-patterns of length $k-1$. If the candidate is frequent, i.e. its occurrence is greater than the minimum support threshold,

then it is considered a pattern, and new candidates are generated from this set to be tested in the next step. However, and despite the simplicity of the algorithm, its candidate generation and test philosophy impairs its efficiency, and makes difficult its extension to deal with continuous flows of data and real-time environments.

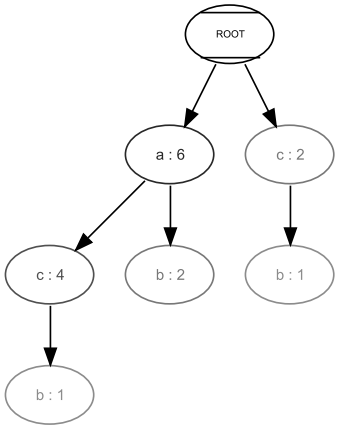
More recently, Han *et al.* proposed an algorithm to surpass some of the problems presented by the Apriori algorithm, namely the necessity of having to generate and test a huge number of candidate sets and execute multiple scans over the entire DB in order to discover the patterns. As such, Han *et al.* developed *FP-Growth*, an algorithm that builds up a compact structure (*FP-Tree*) from the data in a way that avoids scanning the DB multiple times [7]. The algorithm works in three steps: first, it scans the DB once in order to identify frequent items and identifies the best ordering among them; the frequency-descending order among items is then used to reorder each transaction, using the *FP-Tree* to compact all the transactions in the DB; finally, it goes through the tree in order to identify all the patterns in the database. Fig. 1 illustrates the result of applying the *FP-Growth* algorithm (left) to a set of transactions in a DB (right). The algorithm discovers a, b, c and ac as patterns (*min. support* $= 0.4$).

In light of the problem described earlier and taking into account the requirements defined in section 2.3, the *FP-Growth* algorithm presents some drawbacks:

- First by requiring a first scanning over the whole set of transactions from the DB to order items by their occurrence frequency, which is impossible to realize in the autonomous agent's case because we want the mining process to be made online;
- The main objective of the algorithm is to determine all the frequent *itemsets*. Due to the inherent compactness of the *FP-Tree* structure, it is sometimes difficult to determine whether specific combinations of items are considered patterns, having to scan the tree to determine their support;
- Finally, as another consequence of the compactness of the *FP-Tree*, it is difficult to verify if a specific *itemset* is a pattern in real-time. Indeed it would be beneficial for the agent to have an easy and rapid access to all the patterns discovered so far in order to use them to make better evaluations of the current state and also take better decisions upon it.

3.2 Real-time Pattern Mining

One of the requirements stated in section 2.3 was that the algorithm to be developed must be executed in real-time, while the agent interacts with its environment, taking into consideration the past sensory experiences. As such, the first attempt to solve this issue was to try to build an *FP-Tree* structure using the same algorithm before mentioned, but doing it in only one scan over the set of transactions of the DB. To do that we discarded the first scan used to determine the order of the items according to their frequency. Because this order is only a heuristic to minimize the number of nodes generated (not guaranteeing



a, b, c, d
a, c
a, b, d
a, c
c,
b, c, d
a, b
a, c

Fig. 1. The *FP-Tree* (left) resulting from the application of the *FP-Growth* algorithm (*min. support*=0.4) over the set of transactions (right). Lighter nodes represent smaller support values.

that the generated tree is the smallest one possible), we established a fixed order to be used by the algorithm, namely the alphabetic order of the symbols representing the agent's perceptions. Naturally, other kinds of ordering heuristics could be used, namely the ones based on domain knowledge about the environment, for example by determining what stimulus were more likely to be present in the agent's sensors throughout time, estimating the ordering among items. Nevertheless, because the objective was to provide a biased-free mechanism for the agents to explore the environment starting from (almost) zero information about it, and so we used the alphabetical order.

This change of ordering can have an impact on the time used to build the patterns tree, but does not solve the requirement of having the maximum number of known patterns readily accessible to the agent. To solve that problem we changed the way the transactions are inserted in the tree in such a way that now each node represents a specific *itemset* and its count value represents the frequency of that *itemset* so far. The algorithm thus generates all the possible sub-combinations of *itemsets* whenever a transaction is inserted. For example, if we inserted the transaction *abc* in a new tree (Fig. 2), the nodes a, $a \rightarrow b$, b, $a \rightarrow c$, $a \rightarrow b \rightarrow c$ and $b \rightarrow c$ are created with a count value of 1. We call this set of nodes representing all the sub-combinations of an *itemset* its *Dependency Tree* (DT). By making use of the anti-monotone Apriori heuristic [1] we can state that for an *itemset* to be frequent then every node in its DT must also be frequent. The idea behind this way of building the patterns tree is to have the maximum number of frequent patterns stored in the tree's structure without the need of having to search the whole tree for sub-combinations of *itemsets* and determine whether they are frequent.

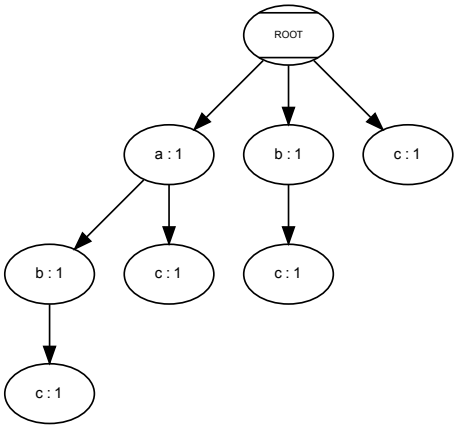


Fig. 2. The patterns tree after inserting transaction *abc*. It also represents the *Dependency Tree* (DT) of the node $a \rightarrow b \rightarrow c$.

Because the number of possible sub-combinations of an *itemset* can be very large, storing all the combinations for every possible transaction seems a very unfeasible and resource-consuming task. To tackle with that problem we defined a heuristic to prune those nodes that do not seem promising in becoming a pattern. This heuristic prunes a node (deletes it from the tree) whenever it's considered as an infrequent node, i.e., when its support (count value) drops below a certain threshold of the pre-established minimum support. In the context of this work we defined this limit as 0.75 times the minimum support[1]. This pruning heuristic is applied when determining the frequent patterns (nodes) to reduce the tree's size overtime. In Fig. 3 we can see the result of applying this new algorithm to the previous example.

3.3 The Jaccard Index

In our opinion, one of the problems that these pattern mining algorithms suffer lies within what they consider to be a frequent pattern. This judgment is based on the statistic of the frequency of an *itemset* in relation to the total number of transactions recorded. In the context of the current work however, sometimes there are some sets of stimuli which frequency is low, but that appear to occur always simultaneously. Looking at the previous example DB, we may notice that most of the times when item b appears in a transaction, a and d also occur in that transaction. Because the support of the *itemsets ab*, *bd* and *abd* is very low,

[1] Through experimentation, a high percentage value should be used as threshold to determine infrequent nodes during the pruning procedure. As such, 0.75 proved to be a value which pruned several nodes without damaging the overall patterns discovered.

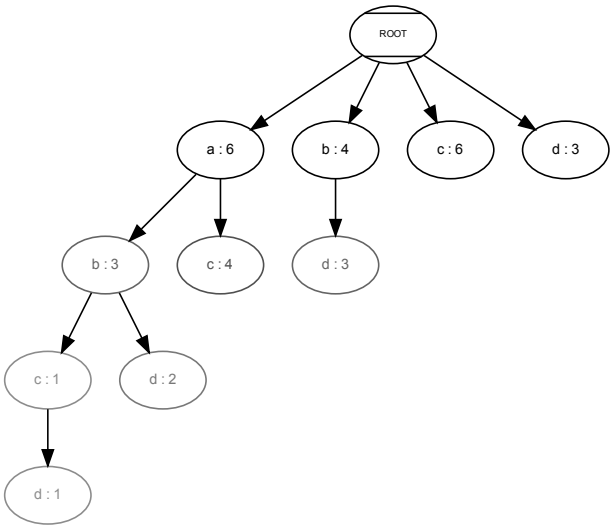


Fig. 3. The tree resulting from the application of the "all-combinations" algorithm (transaction set and support is the same as the above example).

frequency-based algorithms cannot detect them as being patterns. This makes that a lot of useful information about the environment's regularities is being discarded by the agent throughout time.

Due to that fact, we decided to look for a statistic to determine the correlation level between nominal variables (as is the case of the ones here described). Considering the nature and structure of the patterns trees being built, we decided to apply the Jaccard index statistic [8] to determine the frequent patterns. This statistic allows us to determine the level of correlation between several variables as a function of the frequency of their intersection over their union. In the case of two variables A and B (non independent) the index can be expressed using the following formula [11]:

$$Jacc(AB) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1}$$

The index value varies from 0 to 1, so values near 0 indicate a weak correlation between the variables while values near 1 indicate a strong correlation between them. This statistic can be used as a useful heuristic to determine sensory patterns in agents perceptions by allowing us to characterize the level of correlation between the stimuli as a measure of the deviation between their co-occurrence and the sum of all of its occurrences, which at the same time ignores the total number of perceptions made so far. By extending equation (1) to n variables we can informally write the Jaccard index by equation (2).

$$Jacc\ idx = \frac{\text{no. of co-occurrences of the variables}}{\text{no. of co-occurrences} + \text{no. of non-simultaneous occurrences}} \quad (2)$$

If we take the structure of the patterns tree being developed here, we can determine the Jaccard index value of some *itemset* by dividing the count value of the node which represents the *itemset* in the tree (which precisely corresponds to the number of co-occurrences between the items) by the weighted sum of the count values of all the nodes within its DT. The weight determines the sign of the nodes' count value according to the parity of their depth in the tree ($-count$ if odd depth, $+count$ if even).

In this new context, the pruning heuristic is applied right after the insertion of a new transaction, to all the nodes within the DT of the node representing the transaction in the tree, because these were the nodes which count values were updated (for example, if transaction *abc* is inserted, the nodes belonging to the DT of the node $a \rightarrow b \rightarrow c$ are considered). Following the example DB of the previous sections, the application of this new approach results in the tree depicted in Fig. 4. As we can see, for a minimum Jaccard index of 0.4, the algorithm finds the same patterns discovered by the previous algorithms plus the *itemsets ab* and *bd*. *Itemset abd* still does not have enough support to be considered a pattern under the Jaccard index statistic, but it does have minimum support for not being removed under the pruning heuristic.

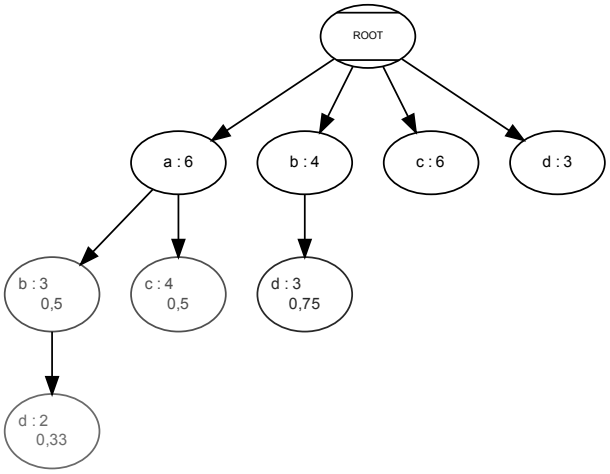


Fig. 4. The tree resulting from the application of the "jaccard-index" algorithm (transaction set and support is the same as the above example). Jaccard index values are represented in the nodes below their support.

4 Experimental Results

In this section we describe the tests carried out over the proposed algorithms to determine their validity and the usefulness of the generated patterns, by presenting the main results and making a comparative analysis between the algorithms performance.

4.1 Test Conditions

To test the efficiency and quantity of patterns generated by the proposed algorithms we decided to test them against the *FP-Growth* [7] algorithm and the *SOTAI* framework algorithm [10], both described earlier. Of course *FP-Growth* is an algorithm which purpose is somewhat different from the type of pattern mining we are aiming at here. Nevertheless, because it served as a base algorithm for the proposed extensions, it provides reference values that we can use when comparing the quantity of patterns generated and the efficiency of the mining processes. In relation to the *SOTAI* framework, we adapted the generated *Fluents* as sensory patterns by considering the set of *Sensations* they represent as *itemsets*. The set of algorithms we ended up testing is the following (short name in brackets):

- *SOTAI* framework knowledge-generation algorithm (***Sotai***);
- *FP-Growth* algorithm (***FP-Growth***);
- *FP-Growth* with alphabetic item order, one scan over the DB (***Alpha-FP-Growth***);
- Algorithm considering all sub-combinations in every transaction with pruning heuristic (***All-Comb***);
- Same algorithm as All-Comb but using the Jaccard index statistic as a heuristic to determine patterns (***Jacc-Index***).

Because we wanted to test the algorithms within an autonomous agent architecture, in a context where the agent continuously interacted with its environment and which perceived sensations were in the form of symbols describing categorical nominal attributes of the objects, we chose *Sotai-Tester* [10] application to generate the transaction data to be input to the algorithms. *Sotai-Tester* agent, as explained before, has a random behavior and at each decision time it chooses an object from the environment to interact with. After that interaction is over, it chooses another object and so on. *Sotai-Tester* environment has a total of 8 objects that the agent can interact with, and the agent has a total of 13 sensors to perceive the environment. There are a total of 70 possible individual symbols to describe the perceptions of the agent. Because some of these symbols can be present at the same time within the agent's sensors, we considered a combination of two or more symbols as a single sensation (for example, if the *color* sensor has *yellow* and *red* symbols at the same update cycle, sensation *color-yellow-red* is considered as being an item). As such, at each update cycle of the agent's process, we recorded in a simple text file all the sensations that were present in the agent's sensors at that time. Each line of the text file thus represents a transaction to be processed by the pattern mining algorithm.

4.2 Metrics

The goal of the tests is essentially to analyze the algorithms' performance and the number and quality of the patterns it finds, i.e., the knowledge generated by the agent from perceiving its environment while interacting with it. As such, the following metrics were adopted and measured for each algorithm at each test execution:

- Time used to build the knowledge structure (tree), reading each transaction one-by-one from the previously generated text files, measured in CPU time (***Build-Time***);
- The number of nodes used to build the tree in order to evaluate the algorithms in terms of memory requirements (***Nodes-Number***);
- Time used to retrieve from the tree every possible pattern according to the minimal threshold established, measure in CPU time (***Pattern-Time***);
- Total number of patterns found. One-item-length patterns were ignored because we are interested in finding correlations in the sensory data (***Pattern-Number***).

Using the *Sotai-Tester* application described earlier we generated a total of 10 text files containing 10K transactions (average length = 5), each representing an update cycle of the agent. To see the relationship between the limit values established to discover patterns and the algorithms' performance, we iterated the minimum support threshold over 10 possible values from 0.1 to 1. In the case of the *Jacc-Index* algorithm, this threshold corresponds to a minimum value for the Jaccard's index associated with a node. In relation to the *Sotai* algorithm, the threshold is the minimum association strength (Phi-coefficient) between the generated structures. After the execution of all the tests, we averaged the values of each metric for each algorithm and each support value, removing mild outliers through quartile estimation.

4.3 Results and Comparative Analysis

The results of the tests described earlier are depicted in Fig. 5. By looking at the algorithms' behavior during the tests we are able to make the following observations about them:

SOTAI framework's knowledge-discovery algorithm didn't perform well at any of the analyzed parameters. It takes too much time to discover associations from the sensory data, maintaining unnecessary information of statistical tests, and discovering few patterns overtime due to its *block-building* approach explained earlier.

Although we introduced *FP-Growth* during the tests only to establish base values for the measures, we found that the algorithm performed very well within its "family". As we expected, by having the transactions sorted by item frequency before its introduction in the tree, less nodes are needed to build the tree and less time is required to search for the patterns than the alphabetical-order algorithm.

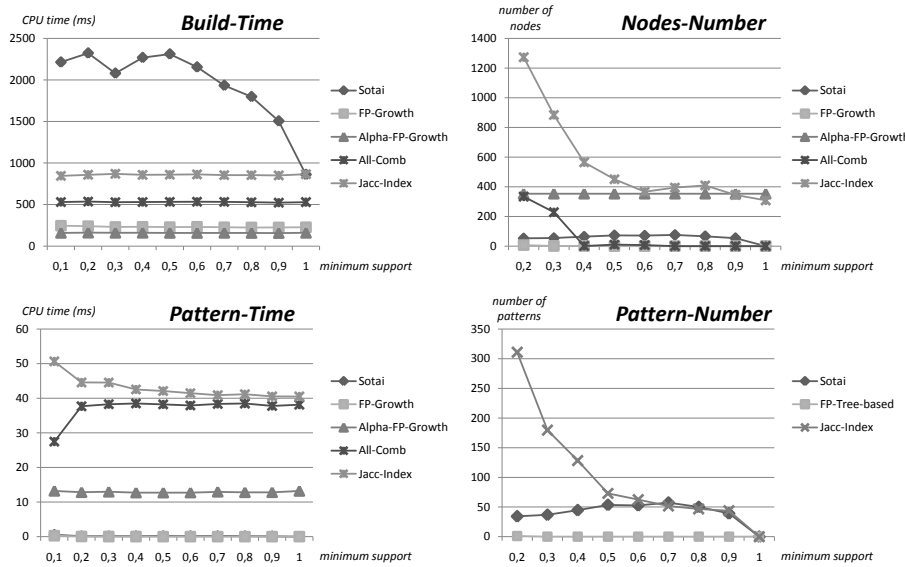


Fig. 5. The graphics containing the results of the tests performed over all the proposed sensory pattern mining algorithms.

By relying on the *itemsets* frequency, *FP-Growth* discovered, as expected, less patterns than the statistic-based association mining algorithms (*Sotai* and *Jacc-Index*). We can also observe that storing all the sub-combinations in memory isn't that useful when mining for frequent patterns: it creates more nodes and requires pruning over the entire tree to find patterns. As a conclusion, none of the *FP-Growth*-based algorithms represents a real solution for the problem being solved.

The first thing we can say about the proposed Jaccard-index-based algorithm is that if we set the minimum index threshold too low, we end up having too many nodes, too many *itemsets* considered as patterns, and also spend too much time scanning the tree for those patterns. A compromise must be made between performance and usefulness of the discovered patterns as to consider how strong an association between stimuli must be for it to be considered a regularity of the environment, a pattern. As expected, it takes more time to build the tree and search for patterns than the *FP-Growth*-based algorithms because it needs to calculate the statistic to determine patterns. However, in this case, the structure of the patterns tree, by storing all the sub-combinations of *itemsets*, enhances its mining performance by maintaining in memory the nodes that are necessary for the index calculation (its Dependency Tree). We believe that the algorithm's performance shows that it can be a solution to our problem: it generates patterns of sensory data from the agent's perceptions throughout time, based on the statistical significance of the correlation between the stimuli they represent; it

also performs this task in a reasonable time to be used by the agent while it interacts with its environment, i.e., in real-time (less than 1 second to process 10K update cycles).

5 Conclusions and Future Work

Autonomous agents are systems inhabiting dynamic and unpredictable environments which they perceive and react to events accordingly. To survive, the agent must learn new facts from the world and perceive its regularities to make better decisions on how to act upon it. In this paper we showed that by modifying some pattern mining algorithms we can provide autonomous agents with mechanisms to discover useful information about its environment while interacting with it in real-time. If we see the agents' perceptions as transactions of a DB, then we can apply those algorithms to discover sensory patterns from the environment. We proposed a new structure to store the sensory information perceived by the agent, which is constructed in a way that facilitates the retrieval of the sensory patterns. We also proposed a new heuristic to discover frequent patterns by using the Jaccard index statistic, which is sensible to some regularities of the environment that are not frequent, but denote particular cases of correlations within the perceptions.

We believe that the use of data mining techniques will provide autonomous agents with capabilities of discovering patterns relating their activities while interacting with the environment. For now, the proposed algorithms discover synchronous sensory patterns, i.e., sets of stimuli that co-occur frequently. In the future we want to extend them for the discovery of *asynchronous patterns*, i.e., sets of stimuli that frequently occur one after the other, revealing causal relations between them. To achieve that, we can adapt algorithms from the sequential pattern mining area [2]. We would also like to test the algorithms in different contexts, testing the pattern mining mechanisms in richer scenarios, possibly involving virtual environments and synthetic characters. Finally another approach would be to test this solution within a multi-agent application. Because different agents interact with the environment in different manners at particular instants of time, they will perceive the world in a singular manner and as such, they will create different sensory patterns throughout time. It would be interesting to check commonalities and differences between the patterns created, and provide them with communication capabilities so that overall social knowledge could be created to reflect the experiences of particular groups.

Acknowledgments

This paper was supported by a scholarship (SFRH / BD / 38681 / 2007) granted by the Fundação para a Ciência e Tecnologia. The authors are solely responsible for the content of this publication. It does not represent the opinion of the Fundação para a Ciência e Tecnologia, which is not responsible for any use that might be made of data appearing therein.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, page 487499. Citeseer, 1994.
2. C. Antunes and A. Oliveira. Sequential pattern mining algorithms: Trade-offs between speed and memory. In *2nd Workshop on Mining Graphs, Trees and Seq.* Citeseer, 2004.
3. P. Cohen, M. Atkin, T. Oates, and C. Beal. Neo: Learning conceptual knowledge by sensorimotor interaction with an environment. *Proceedings of the first international conference on Autonomous agents*, pages 170–177, 1997.
4. S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. *Lecture Notes in Computer Science*, 1193:21–36, 1997.
5. W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. *Ai Magazine*, 13(3):57–70, 1992.
6. J. Gibson. *The ecological approach to visual perception.* Houghton Mifflin, 1979.
7. J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, Janeiro 2004.
8. P. Jaccard. The distribution of the flora in the alpine zone. *New Phytologist*, 1912.
9. P. Maes. Modeling adaptive autonomous agents. *Artificial life*, 1(1-2):135–162, 1994.
10. P. Sequeira, M. Vala, and A. Paiva. What can i do with this?: finding possible interactions between characters and objects. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 5:1–5:7, New York, NY, USA, 2007. ACM.
11. M. J. Warrens. *Similarity coefficients for binary data: properties of coefficients, coefficient matrices, multi-way metrics and multivariate coefficients.* Doctoral thesis, Leiden University, 2008.