1 Introduction

The problem of association rules mining was first introduced by Agrawal et al [Agrawal1993] to find correlations between products purchased by customers in one transaction. The potential of this knowledge was easily recognized and since then association rule and frequent pattern mining were adopted by different domains with good results. Recent progresses in biomedical and biotechnological research were responsible for increasing the amount of data available in these domains. That growing amount of knowledge, coupled with the development of various methods to mine knowledge, generated a new domain whose goal is to store, extract, organize and analyze biological sequences – bioinformatics. In this area of knowledge, identifying a pattern in a data set can be essential to understand the information contained in that data set, as the recurring occurrence of a pattern means that pattern has some relevance in the data context. Many biodata analyze tasks look for structured frequent patterns, such as frequent molecular fragments in proteins or chemical substances. The complex nature of this kind of data forces the adoption of special strategies to represent information, with graphs being the common choice for this purpose. In fact, labeled graphs have been successfully used to represent complex data sets, such as biological data sets and most of the algorithms that mine frequent patterns in this kind of data use them, which converts the problem of finding frequent patterns in a problem of finding frequent subgraphs. Using this abstraction forces the algorithm to deal with problems like graph and subgraph isomorphism and the computational weight of this kind of operations. Independently of the used approach, frequent pattern mining methods tend to obtain a number of results too high for human perception and yield frequently results already known by the user. These problems are common to generic association rules and frequent pattern mining and not exclusive of bioinformatics and that’s the reason why this area of research still presents challenges so that its techniques can exhibit better performance and return quality results.

One possible strategy to allow the user more control over the knowledge mining process is the incorporation of domain knowledge in that process. The majority of works developed in knowledge mining area prefer to study issues related to the interestingness of the results and the definition of interestingness measures. However, introducing domain knowledge in the knowledge mining and discovery process is an option with potential to induce improvements at various levels in that process. Domain knowledge can reduce the space search and contribute to more useful results in the user’s point of view. So, this is a field where there is a lot of work to do, especially in the matter of knowledge representation and use, and bioinformatics is a privileged area to explore the introduction of domain
knowledge in the mining processes, as there is a large amount of specific knowledge available. This text documents the work developed over a set of biochemical data to show it is possible to extract frequent patterns in that data set, using a simple pattern mining approach instead of a more complex approach that uses graphs and frequent subgraph mining. Domain knowledge was introduced in an algorithm similar to Apriori algorithm using Onto4AR framework. The results obtained this way were compared with the results obtained for the same data set using gSpan algorithm, as this is the algorithm commonly used for data sets that can be represented using graphs. This text is structured as follows: section 2 describes briefly the state of art in frequent pattern mining; section 3 introduces the issue of using domain knowledge to improve pattern mining; section 4 describes the work done; section 5 summarizes the results; section 6 shows conclusions.

2 Frequent pattern mining

The problem of frequent pattern mining can be understood as a sub problem of association rules mining, as the first step for solving this problem is to determine all itemsets whose support is equal or superior to a minimum support value.

The term transaction belongs to data mining vocabulary and usually refers a set of data contained in a transaction from the database system point of view. In this context, an item corresponds to one of the values in that transaction and an itemset corresponds to a set of items that belong to the same transactions. When we mention the expression “intra-transaccional pattern”, we refer to frequent patterns at transaction level. There are two main approaches for mining this kind of patterns: candidate generation and test, represented by Apriori algorithm [Agrawal1994], and pattern growth approach, represented by FP-growth algorithm [Han1999]. Several studies [Han1995] show that pattern growth algorithms have better performance because they can eliminate a critical and expensive step as candidate generation.

Although, there are situations where data is more complex and its own structure is part of its semantic. This means that structure can’t be ignored during the knowledge discovery process or it would result in loss of information and probably low quality results. Graphs are often used as a tool to represent that special interactions between the entities that compose a data set. Yan and Han [Yan2002] proposed gSpan algorithm as a way to extract frequent patterns in data sets represented as graphs with using candidate generation. This was one of the first algorithm for mining frequent subgraphs to use depth-first search, which makes it less memory demanding.

3 Using ontologies to mine frequent patterns

Ontologies are a well known and successful way to represent and share knowledge in Artificial Intelligence area and also a viable option to integrate domain knowledge in knowledge discovery process, as shown in works by Češpivová et al [Češpivova2004], Vanzin [Vanzin2004] and Euler and Scholz [Euler2004]. Based on these works, Antunes designed a framework that allows a user to define restrictions over the knowledge mining process using an ontology that expresses knowledge on the
process domain [Antunes2007]. There are two main categories of restrictions: interest restrictions and content restrictions. Interest restrictions impose quantitative conditions over a set of items, while content restrictions impose qualitative conditions over the components of an itemset. Content restrictions can exist at a taxonomic (the item’s characteristics) or non-taxonomic (the relations between the items) level. One of the simplest taxonomic content restrictions that can be defined on this framework is an item restriction, which is satisfied when at least part of the items in an itemset belong to a specific subset. Using ontologies allows the definition of more complex content constraints, based on the relations between concepts. Two items are related if there is a non-taxonomic relations between their classes. Based on this notion, Antunes defines three types of constraints:

- Weakly connected – an itemset is weakly connected if all its items are related with at least another item in that itemset.
- Softly connected – an itemset is softly connected if there is a chain of relations between the items.
- Strongly connected – an itemset is strongly connected if all items are related with all the other items in that itemset.

Using this categorization, the user can specify constraints in any of these categories or even compose constraints to create new types of constraints. The figure below shows a diagram of the above categorization.

![Diagram of the proposed constraints](image)

**Figure 1** – Categorization of the proposed constraints [Antunes2007]

### 4 Identification of frequent molecular fragments

This section describes the work developed to accomplish this work’s goal: to show it is possible to identify frequent molecular fragments using an approach typical of intra-transactional patterns, if we supply the algorithm with domain knowledge contained in an ontology and expressed by means of constraints.
4.1 Problem formalization

For our problem, each molecule corresponds to a transaction, the atoms that compose that molecule are the items in that transaction and an itemset is a set of atoms. However, two carbon atoms can’t be considered as a single item only based on the chemical element, because from a chemical point of view they can perform different roles in the molecule and are probably connected to different atoms. So, it was necessary to define in which conditions two atoms represent the same item. In this context, “to be the same item” doesn’t mean they are the same instance of the atom concept. Instead, it means they satisfy a previously established condition which defines when two atoms are equal in terms of the possibility of adding new bonds. Just because we have two atoms of the same chemical element, it doesn’t mean we can add the same chemical bonds to both and that it matches the real data. So, a carbon atom is only “equal” to another carbon atom if both atoms have the same number and type of bonds.

This work’s goal is to identify molecular fragments using an approach fitted for tabular data, so it’s necessary to define also in which conditions an itemset represents a molecular fragment. An itemset can be considered a molecular fragment when it respects these conditions:

- Each item in the itemset has to establish a chemical bond with at least one of the other items in the itemset.
- Each item in the itemset has a maximum number of chemical bonds it can establish, which is equal to the valence number of the corresponding chemical element. For example, a carbon atom has a valence number of 4, which means each carbon atom can’t establish more than four chemical bonds in one molecular fragment. The total number of bond established by an atom is given by the sum of the bond types in which is involved. This means that a carbon atom can establish four simple bonds (1 + 1 + 1 + 1 = 4) or two simples bonds and a double bond (1 + 1 + 2 = 4) or two double bonds (2 + 2 = 4), but never two double bonds and a simple bond (2 + 2 + 1 = 5).

4.2 Ontology

A simple ontology was designed to represent basic concepts of chemistry, as shown in figure 2 where concepts are denoted by ellipsis, relations is-a by non-labeled arrows and relations between concepts as labeled arrows.
Figure 2 – Ontology used in this work

There are three main concepts: molecule, atom and bond. Each molecule is composed by two or more atoms, as denoted in Atoms relations, and has at least one bond, as denoted in relation Bonds. The concept of bond is related with the concept of atom due to the fact each bond involves two different atoms. Each concept has its own attributes where additional information is kept.

4.3 Algorithm

Apriori was one of the first frequent itemset mining algorithms proposed and is still today one of the more versatile, proving itself competitive when compared with other frequent pattern mining algorithms. When compared with depth-first search algorithms, Apriori scans the search space in a moderate way, which is an advantage when the database is large, and even shows superior performance when the database contains many frequent items but not many frequent itemsets [Bodon2005]. These are some of the reasons that justify the appreciation that frequent itemset mining community shows for this algorithm, from which different implementations and variants were developed for different purposes. These reasons motivated Goethals and Zaki to create a repository to allow the test of different mining algorithms and implementations for a large variety of parameters and datasets [Goethals2004].

In this work, Bodon’s implementation deserved a special attention for being one of the more recent and returning good results. The strong point in this implementation is the use of a data structure similar to a hash tree – a trie. A trie is a tree with the root node at depth 0 and each node at depth level d points at least for a node at depth level d+1. In frequent pattern mining context, root node corresponds to the empty itemset and each k-itemset corresponds to a node, as its k-1-itemsets; all 1-itemsets are linked to the root node and the edges keep the item id, while the nodes store the itemset support.

After data is read from file, the trie is initialized with the frequent 1-itemsets. For each itemset, an edge and respective node are created and the edges are added to the trie by decreasing support order.
Using this heuristic is explained by the fact that candidate generation and test algorithms have their performance tied to the number of candidate itemsets in each generation, as it can contribute to reduce the number of candidate itemsets.

Next, to each edge $i$ is added a new trie which edges are the siblings of edge $I$ and the support is accounted for the itemset composed by all the items denoted in each edge on the path. That value is kept in the node, which is labeled as frequent if it respect the minimum support requisite. This process is repeated for each edge labeled as frequent till there are no more edges to expand or because there are no more frequent itemsets. Due to this depth mechanics, it's not correct to say we used Apriori algorithm as proposed by [Agrawal1994], and for that reason we refer to the algorithm used in this work as FastApriori from this point on.

### 4.4 Definition of constraints on Onto4AR framework

Apriori was initially designed to identify correlations between products purchased by customers in the well-known supermarket basket problem. This means the algorithm doesn’t take into account some data characteristics, as relations that may exist between the atoms, and for certain types of data the simple fact of occurring or not in a transaction may not be enough to obtain valid results. This is what happens with our data: just because four atoms occur together in a frequency superior to a minimum value, that doesn’t mean those atoms are related between them in the molecules where they occur. So, it was necessary to introduce in the algorithm information that isn’t available to it in a way it can use it without disrupting its logic. The following categories of constraints were defined:

- Structural constraint (RestrEst)
- Bond constraint (RestrLig)
- Support constraint (RestrSup)

#### 4.4.1 Structural constraint

This is a non-taxonomic constraint that tries to assure that the algorithm finds molecular fragments and not just sets of frequent items. A structural constraint established that an item can only be added to an itemset if that item has at least one chemical bond with one of the items in that itemset. This constraint can detect two different types of structures:

- When the item that is added to an itemset has chemical bonds only with one of the items in the itemset, the constraint finds linear fragments
- When the item that is added to an itemset has chemical bonds with more than one of the items in the itemset, the constraint finds cyclic fragments.

This constraint is satisfied when the item that is added to an itemset can establish at least one chemical bond with at least one of the items on the itemset. In terms of implementation, two versions of this constraint with different relaxation levels were proposed: $\text{RestrEst}^-$ and $\text{RestrEst}^+$. $\text{RestrEst}^-$ is less severe and accepts an item as long as that item occurs simultaneously in a transaction with the
other items in the itemset. RestrEst+ is less permissive and only accepts an itemset if the item to add occurs simultaneously with the other items in a transaction with the same chemical bonds denoted in the itemset.

4.4.2 Bond constraint
This constraint assures that the items in an itemset don’t establish bonds beyond its maximum capacity by verifying if the number of bonds it establishes in an itemset isn’t superior to the number of bonds that turned it into an unique item. This means that, if a carbon atom is characterized by having a simple bond and a double bond, that item can’t establish four simple bonds as that violates its definition as item.

4.4.3 Support constraint
Support constraint is applied in prune step of the algorithm and defines in which conditions an itemset respects the minimum support requisite. As for structural constraint, there are two versions with different levels of severity: RestrSup- and RestrSup+. The more permissive version (RestrEst-) only verifies if the items that compose the itemset occur together in a transaction and essentially corresponds to the basic definition of support specified on the Onto4AR framework. The less permissive version (RestrSup+) checks not only the presence of the items but also the presence of the same number and type of bonds contained in the itemset.

5 Results
This section shows the results obtained applying the algorithm FastApriori with and without using domain knowledge. Tests were run in a Pentium IV 3 Ghz with 1 Gb of memory.

5.1 FastAprioriStandard
FastAprioriStandard doesn’t have access to domain knowledge. For this algorithm, each molecule is a transaction and each atom of each chemical element is an item; the algorithm goes on composing itemsets with different atoms and checking its support without any criteria on structure or chemical bonds. The results show the loss of information on the bonds between the atoms and often no correspondence between the fragments and the real molecules.

5.2 FastAprioriRich
This is the version of FastApriori that integrates domain knowledge using constraints. The first step is to identify the different items, according to our formalization. Next, the mining step was run with all the possible combinations of constraints for different support values and for two different data sets: one only with linear molecules and other that include molecules with rings. The conclusions shown below were made comparing the results obtained using different combinations of constraints with the results obtained by gSpan.
5.2.1 Results

Using RestrEst- and RestSup- corresponds to the most permissive combination and yields a large number of fragments and duplicates. Many of those fragments don’t occur in the data set with the structural organization reported in the results, although its chemical content is correct as the atoms that compose that fragment respect the minimum support requisite. It is also true that the frequent fragments that really exist in the data set can be found in the results.

Using RestrEst+ and RestrSup+ corresponds to the less permissive combination of constraints. The results obtained this way have less fragments and less duplicates and their quality is higher, as the number of fragments whose structure doesn’t match the one in the data asset is inferior to the number of fragments in that situation on the previous combination of constraints.

Combining RestrEst- and RestrSup+ corresponds to a relaxation in the itemset formation, which results in a higher number of results and more fragments whose structure doesn’t match the structure in the data set. The combination RestrEst+ and RestrSup- corresponds to a relaxation in support accounting and yields results similar to the most permissive combination of constraints, because relaxing the way the support is calculated allows to admit more fragments in the result set.

Then, we can say that FastAprioriRich returns better results when RestrEst+ and RestrSup+ are used together, especially with the data set that contains molecules with rings. For this data set in particular, FastAprioriRich can detect cyclic structures that gSpan doesn’t report, for low minimum support values. In any situation, FastAprioriRich always finds more frequent fragments than gSpan, which can be explained by all the duplicates found. The exception is for low minimum support values, where gSpan can find more frequent fragments than FastAprioriRich. The graph below show this situation when both algorithms are applied to ten molecules with an average of 45 atoms by molecule.

![Graph](image-url)

*Figure 3 – Number of frequent molecular fragments found by FastAprioriRich and gSpan*
5.2.2 Performance
Comparing the execution times of both algorithms, FastAprioriRich always shows higher execution times than gSpan and this difference is higher when minimum support value decreases. This is not a surprise at all, as gSpan is known as the algorithm with better performance when applied to biodata sets [Worlein 2005].

6 Conclusions
Results shows the goal was achieved, as they proved that it is possible to identify frequent molecular fragments using an algorithm designed for tabular data if we supply it domain knowledge. Due to the lack of works on domain knowledge integration in mining algorithms, this work can motivate further investigation in this field and posterior works.

However, there is space for improvements to this work, especially in aspects concerning the quality of the results and the algorithm performance. This last issue is deeply connected to the algorithm, which includes a candidate generation step and consequently makes the algorithm very demanding in terms of memory. As said before, Apriori shows good performance for large data sets when the number of frequent itemsets isn’t high. When we deal with organic molecules, those kind of molecules have many atoms and bonds that have to be considered during the candidate generation step. This means the algorithm performance is affected by the problem features. One strategy to reduce the amount of memory needed could be to not process atoms whose presence doesn't affect the molecule's behavior, as hydrogen atoms.

The quality of the results obtained is also an issue that deserves attention, as the results contain not only the “correct” fragments but also duplicates and fragments that are only chemically correct. We could address this matter by supplying the algorithm extra information with a wider and more complete ontology. For example, in organic chemistry molecules are classified according to its chemical function in groups that share a structural group; if the ontology included information on that classification, it would allow to filter results and that would certainly improve the quality of the results.

7 References


[Cespivova2004] Češpivová, Hana; Rauch, Jan; Vojtěch, Svatec; Kejkula, Martin; Tomčeková, Marie; “Roles of Medical Ontology in Association Mining CRISP-DM Cycle” in Proceedings of the Workshop on Knowledge Discovery and Ontologies (KDO 2004), pp. 1-12, Pisa, Italy, 2004


[Han1999] Han, Jiawei; Pei, Jian; Yin, Yiwen; “Mining Frequent Patterns Without Candidate Generation”, Technical Report TR-99-12, Computing Science Technical Report, Simon Fraser University, Outubro 1999

[Worlein2005] Worlein, Marc; Meinl, Thorsten; Fischer, Ingrid; Philippsen, Michael; “A Quantitative Comparison of the Subgraph Miners MoFa, gSpan, FFSM and Gaston” in Proceedings of the 9th European Conference on Principles and Practices of Knowledge Discovery in Databases, Porto, Portugal, Outubro 2005


[Yan2002] Yan, Xifeng; Han, Jiawei; “gSpan: Graph-Based Substructure Pattern Mining” in Proceedings of 2002 IEEE International Conference on Data Mining (ICDM02), 2002