MRAR: Mining Multi-Relation Association Rules

Reza Ramezani a, *, Mohamad Saraee a, Mohammad Ali Nematbakhsh b

a Electrical & Computer Engineering, Isfahan University of Technology, Iran.
b Department of Computer Engineering, University of Isfahan, Iran.

ARTICLE INFO.

Article history:
Received: 23 December 2013
Revised: 28 April 2014
Accepted: 08 June 2014
Published Online: 13 September 2014

Keywords:
Data Mining, Knowledge Discovery, Association Rules, Multi-Relation Association Rules, MRAR, Copulative Entity, Endpoint Entity, ItemChain

ABSTRACT

In this paper, we introduce a new class of association rules (ARs) named “Multi-Relation Association Rules” which in contrast to primitive ARs (that are usually extracted from multi-relational databases), each rule item consists of one entity and several relations. These relations indicate indirect relationship between entities. Consider the following Multi-Relation Association Rule where the first item consists of three relations live in, nearby and humid: “Those who live in a place which is near by a city with humid climate type and also are younger than 20 → their health condition is good”. A new algorithm called MRAR is proposed to extract such rules from directed graphs with labeled edges which are constructed from RDBMSs or semantic web data. Also, the question “how to convert RDBMS data or semantic web data to a directed graph with labeled edges?” is answered. In order to evaluate the proposed algorithm, some experiments are performed on a sample dataset and also a real-world drug semantic web dataset. Obtained results confirm the ability of the proposed algorithm in mining Multi-Relation Association Rules.

© 2014 JComSec. All rights reserved.

1 Introduction

In KDD process, the problem of finding frequent patterns and association rules (ARs) has been studied in different settings. The association rules mining (ARM) problem has gained considerable interest among the researchers as one of the most important data mining components due to its usage in everyday life. One of the ARM goals is to find frequent patterns from existing data. These patterns show what items occur more frequently with each other. Employing these patterns, the desired ARs would be generated. Each AR shows that if some items or events occur together, some other specific items or events will also occur with a certain probability which is known as confidence.

The problem of mining association rules focuses on discovery episodes in a sequence of events [1, 2], using the hierarchies of items type, search for sequential patterns in the collection of transactions [3–5] and etc. In these cases, the required language to discover a patterns is more complex than market-basket applications and hence specialized algorithms exist for these tasks. ARM studies also have evolved from techniques for discovery of functional dependencies [6, 7], causal rules [8, 9], classification rules [10, 11], strong rules [12], clustering rules [13, 14], etc. to tabular-based [15–17] or graph-based [5, 18–20] efficient methods for ARM in large sets of transaction data.

* Corresponding author.
Email addresses: r.ramezani@ec.iut.ac.ir (R. Ramezani), saraee@cc.iut.ac.ir (M. Saraee), nematbakhsh@eng.ui.ac.ir (M.A. Nematbakhsh)
ISSN: 2322-4460 © 2014 JComSec. All rights reserved.
Each AR has one antecedent part and one consequent part where each part consists of one or more items. Current ARM algorithms generate primitive ARs consisting of only one entity and at most, one relation. For example the following rules are primitive rules:

- **Apple, Tomato → Cucumber**, \(\{C = 0.78\}\)
- **Buy(Apple), Buy(Tomato) → Buy(Cucumber)**, \(\{C = 0.78\}\)
- **AgeYoungerThan(20), MaritalState(Single), FatherSalaryMoreThan(2000$) → GraduateInYears(4)**, \(\{C = 0.78\}\)

In these rules, *italic* words are relations and **bold** words and numbers are entities. The first and the second rules indicate that “Those who buy apple and buy tomato → they also buy cucumber, with probability of 78%”. The last rule indicates that: “Those who are younger than 20 and are bachelor and also their fathers’ salary is more than 2000$ → they are graduated in 4 years, with probability of 78%”. There are 3 items in the antecedent part of this rule and there is only one in the consequent part with each item having only one relation. In this example, **AgeYoungerThan(20)** is an item and **AgeYoungerThan, MaritalState, Father-SalaryMoreThan and GraduateInYears** are relations of the entities “20”, “Bachelor”, “2000$” and “4” respectively.

In this paper, a novel algorithm is proposed to extract **Multi-Relation Association Rules** from RDBMS and semantic web data. These rules are a new class of ARs with more than one relation in each item of each rule. These new rules allow discovering indirect relationships among entities. For example, consider the following rule:

- **LiveIn(NearTo(ClimateType(Humid))), AgeLessThan(20) → HealthCondition(Good)**, \(\{C = 0.78\}\)

This rule indicates that “Those who live in a place which is near to a city with humid climate type and also are younger than 20 → they have a good health condition, with probability of 78%”. There are 3 relations in the first item of the antecedent part of this rule. Details about **Multi-Relation Association Rules** are addressed later in Section 2.3.

To the best of our knowledge, this is the first paper that introduces this new class of ARs and proposes a new algorithm to solve the problem. The proposed algorithm receives a directed graph with labeled edges as input data (these labels differ to edge’s weight) and recursively traverses the graph to extract **Multi-Relation Association Rules**. The input graph is a special graph that the source vertices indicate entities, the destination vertices indicate other entities or values of an attribute of the source entity (source vertex), and each edge indicates a relation between two entities or an attribute of the source entity. Any dataset convertible to this graph can be employed by the algorithm. Such data structure also makes it possible to extract ARs from heterogeneous data. Heterogeneous data, are those data that their entities (with same type or different types) can take different attributes. The input dataset could be heterogeneous semantic web data or existing data in relational databases that in both cases, data should be converted to a directed graph with labeled edges. In this paper these types of datasets and also how they are converted to directed graphs with labeled edges are discussed.

To clarify the task of converting RDBMS and semantic web data to a suitable directed graph with labeled edges and also the problem of mining **Multi-Relation Association Rules**, a simple and overt example will be shown. Finally, to evaluate the proposed algorithm behavior and also to prove its ability in mining **Multi-Relation Association Rules**, several experiments have been done on a real-world drugs dataset. The obtained results show the usefulness of the proposed algorithm and its ability in mining **Multi-Relation Association Rules** from datasets convertible to directed graphs with labeled edges.

The rest of the paper is organized as follows. **Section 2** briefly describes the concepts of association rules, semantic web and **Multi-Relation Association Rules**. **Section 3** introduces a number of related work. **Section 4** investigates two different kinds of input dataset, RDBMS and semantic web data, and shows how to convert them to a directed graph with labeled edges. **Section 5** contains the general methodology and foundations of the proposed method addition to the related concepts and data structures. **Section ??** presents the proposed algorithms pseudo code in detail. **Section 7** shows an example to clarify the proposed algorithm and the employed data structures concepts. **Section 8** gives the experimental evaluations and describes the obtained results and finally the **Section 9** concludes the paper and offers some future work.

## 2 Basic Concepts

This section briefly describes **Association Rules, Semantic Web** and **Multi-Relation Association Rules** concepts which are related to our work.

### 2.1 Association Rules

Frequent item set mining and association rules induction are powerful methods for so-called market basket analysis, which aims at finding regularities in the
co-occurred items, such as sold products, prescript biomedical drugs and etc. The problem of mining association rules was first introduced in 1993 [15]. Let us denote each item with \( I_i \), thus \( I = \{I_1, I_2, \ldots, I_m\} \) is set of all items which sometimes called the item base. Each transaction \( T_i \) is a subset of \( I \) and based on transactions we define database as collection of transactions denoted by \( D = \{T_1, T_2, \ldots, T_n\} \). Based on this definition each transaction contains only items and there is only one relation among items (e.g. bought together) and thus this relation is not shown in the transaction. Each itemset \( (S) \) is a non-empty subset of \( I \) and an association rule \( (R) \) is a rule in the form of \( X \rightarrow Y \) which both \( X \) and \( Y \) are itemsets and the relation among items is implicit. This rule means that if in a transaction the itemset \( X \) occurs, with certain probability the itemset \( Y \) will appears in the same transaction too. We call this probability as confidence and call \( X \) as rule antecedent and \( Y \) as rule consequent.

- **Support of an Item Set**

  The absolute support of the itemset \( S \) is the number of transactions in \( D \) that contain \( S \). Likewise, the relative support of \( S \) is the fraction (or percentage) of the transactions in \( D \) which contain \( S \).

  More formally, let \( S \) be an item set and \( U \) the collection of all transactions in \( D \) that contain all items in \( S \). Then

  \[
  \text{Sup}_{\text{abs}} (S) = |U|
  \]

  \[
  \text{Sup}_{\text{rel}} (S) = \left(\frac{|U|}{|D|}\right) * 100\%
  \]

  For brevity we call \( \text{Sup}_{\text{rel}} (S) \) as \( \text{Sup} (S) \).

- **Confidence of an Association Rule**

  The confidence of an association rule \( R = X \rightarrow Y \) is the support of the set of all items that appear in the rule divided by the support of the antecedent of the rule. That is,

  \[
  \text{Conf} (R) = \left(\frac{\text{Sup} (\{X \cup Y\})}{\text{Sup} (X)}\right) * 100\%
  \]

  Rules are reported as association rules if their confidence reaches or exceeds a given lower limit (minimum confidence, to be specified by a user).

- **Support of an Association Rule**

  As mentioned in [15, 16], the support of the rule is the (absolute or relative) number of cases in which the rule is correct. For example in the association rule \( R: A, B \rightarrow C \), the support of \( R \) is equal to support of \( \{A, B, C\} \).

- **Frequent Itemsets**

  Itemsets with greater support than a certain threshold, so-called minimum support are frequent itemsets. The goal of frequent itemset mining is to find all frequent itemsets.

- **Maximal Itemsets**

  A frequent itemset is called maximal if no superset is frequent, that is, has a support exceeding the minimum support.

- **Items structure**

  In this paper, if there is only one relation between items (such as bought together) each item is equal to an entity (such as beard, cheese and etc.), otherwise if there are different relations among items, each item not only is equal to an entity but also it is equal to an entity and one relation. Figure 1 shows two kinds of Item Structure.

  ![Figure 1. Item Structure](image)

  **2.2 Semantic Web**

  Semantic web data is one of the employed data sources in this papers. Hence in this part some concepts of semantic web are described briefly.

  The Semantic Web (or Web of Data), sometimes called the third generation of the Web, emerges in distinction to the traditional web of documents. The goal of the Semantic Web is to standardize web page formats so that the data becomes machine readable. This data is described by ontologies. A well-known definition by T.R.Gruber in 1995 is “An ontology is an explicit specification of a conceptualization” [21]. The main purpose of the semantic web is to be machine readable so this feature needs to make entities meaningful and also describe entities by standard methods.

  In order to describe entities, some means of entity representation and entity storing are needed. There are several methods for representing and storing semantic web data. The first method is RDF \(^1\) which is based on XML structure. XML is a powerful standard and also is flexible for transmitting structured data. In fact, the RDF documents are descriptions of semantic web data so this data becomes machine readable. Each RDF statement is a triple and each triple consists of three parts: subject, predicate and object. Subjects and predicates are resources that are identified by URI.

---

\(^1\) Resource Description Framework
Objects can be resources and shown by URI or can be constant values (literals) and represented as strings. In each triple, one relation or typed link exists between either two resources or between one resource and one literal. A similar concept to the URL is the IRI, which has been introduced to represent non-Latin text items in order to internationalize DBPedia [22].

RDFS is an extension of RDF which allows to define entities over classes, subclasses and properties. Hence it is possible to apply some inference rules on these RDFS structure entities.

Due to RDF and RDFS limitations the OWL 2 has been introduced which has more powers of deduction. OWL, which is based on DAML+OIL [3], is the most well-known language that applies description logic to the semantic web data. The first version of this language has three versions, OWL Lite, OWL DL and OWL Full, which differ in expressive ability and deductive power. This language also allows transitive, symmetric, functional and cardinality relations between entities.

These three OWL flavors (Lite/DL/Full) are a bit old-fashioned. New profiles have been designed as OWL 2 [24]. OWL 2 profiles are defined by placing restrictions on the structure of OWL 2 ontologies. Syntactic restrictions can be specified by modifying the grammar of the functional-style syntax and possibly giving additional global restrictions. OWL 2 has three subsets (EL, QL and RL). OWL 2 EL is particularly useful in applications employing ontologies that contain very large numbers of properties and/or classes and has polynomial time reasoning complexity with respect to the size of the ontology. OWL 2 QL is aimed at applications that use very large volumes of instance data, and where query answering is the most important reasoning task. This profile is designed to enable easier access and query to data stored in databases. OWL 2 RL is aimed at applications that require scalable reasoning without sacrificing too much expressive power. It is designed to accommodate OWL 2 applications that can trade the full expressivity of the language for efficiency, as well as RDF(S) applications that need some added expressivity.

As with traditional databases, which in order to retrieve information, need an endpoint language (SQL), semantic web datasets need such a language too. For this purpose, the SPARQL 4 [25, 26] language has been introduced which is able to extract information and knowledge from semantic web datasets. DBPedia [27] is an example of a semantic web dataset. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL has capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL also supports extensible queries based on RDF graphs. The results of SPARQL queries can be presented as result sets or RDF graphs.

In addition to RDF, semantic web data can be stored in different kinds of dataset formats, such as Turtle, Jason, NTriples, and etc. Regardless to dataset format, it is possible to extract data from datasets with different format and ontologies by SPARQL commands and convert the extracted data to a directed graph with labeled edges which is the standard input dataset of the proposed algorithm.

### 2.3 Multi-Relation Association Rules

As mentioned in Section 2.1, antecedent part and consequent part of ARs are constructed of itemset. Each itemset consists of one or more items. In the simplest form, each item contains only one entity and has no relation (in fact relations are implicit). These simple items are extracted from those datasets that has only one type relation among entities and thus this relation does not being put in items. For example consider a market basket analyze problem that there is consider a market basket analyze problem that there is

\[
\text{Bread, Cheese } \rightarrow \text{ Cucumber, } \{C = 0.78\}
\]

This rule indicates: “Those customers who buy Bread and Cheese, may also buy Cucumber with probability 78%”. In order to extract Multi-Relation Association Rules, this kind of data are not usable, since more than one type relation among items is required.

There is another form of ARs that each item consists of one entity and one relation. The image of such item has been depicted in the right side of Figure 1. These items can be discovered from those datasets that have more than one type of relation among entities. Relational databases, heterogeneous semantic web data or graph structured data are the most important data sources of such data. For example, in the following rule, each item has one entity and one relation. This rule means “Those who are younger than 20 and are bachelor and also the salary of their father is more than 2000$ \rightarrow \text{ they are graduated in 4 years, with probability 78%}”.

---

2 Ontology Web Language
3 http://www.daml.org/
4 http://www.w3.org/TR/rdf-sparql-query/
In the scientific societies, those rules that are extracted from multiple tables (multiple relation in relational databases) are referred to as “Multi Relational Association Rules” [28–31] and means “those rules that are extracted from multiple tables (multiple relations)” not “those rules that have several relations in their items”.

Here we define “Multi-Relation Association Rules” as “those rules that have more than one relation in at least one of their items”. For example in the following rule, there are three relations in the first item and means “those who live in a place which is near to a city with humid climate type and also are younger than 20, they have a good health condition, with probability of 78%”:

- \( \text{LiveIn}(\text{NearTo}(\text{ClimateType}(\text{Humid}))), \text{AgeLessThan}(20) \rightarrow \text{HealthCondition}(\text{Good}) \)

In this paper a new algorithm named MRAR has been proposed to extract Multi-Relation Association Rules from directed graphs with labeled edges. This graph is constructed from various data sources, such as RDBMS or semantic web data. In order to extract these rules, regardless to the type of input dataset, the input data should be converted to a directed graph with labeled edges in a way that source vertices indicate entities, destination vertices indicate other entities or values of an attribute of the source entity (source vertex), and edges indicate relations between two entities or indicate an attribute of the source entity.

In order to clarify the problem, consider the presented graph in Figure 2.

Table 1 shows the meaning of Figure 2 entities.

Now we define “primitive rules” as those rules that have at most one relation in each item. By traversing and mining the graph presented in Figure 2, these primitive rules would be extracted:

1. Those who \( \text{LiveIn} \) \( \text{Isfahan} \) \( \rightarrow \) \( \text{StudyIn} \) \( \text{IUT} \) too \{Ali, Ahmad\}
2. Those who \( \text{StudyIn} \) \( \text{IUT} \) \( \rightarrow \) \( \text{are Supervised By} \) \( \text{Saraee} \) too \{Ali, Ahmad\}

In these primitive rules, each item has only one relation. Such as \( \text{LiveIn} \) and \( \text{StudyIn} \). In these rules, italic words indicate relations and bold words indicate entities.

Consider the following Multi-Relation Association Rules that have been extracted from the graph presented in Figure 2:

3. Those who their \( \text{Health Condition} \) is \( \text{Good} \) \( \rightarrow \) they \( \text{LiveIn} \) in a place \( \text{NearTo} \) by a city which its \( \text{ClimateType} \) is \( \text{Humid} \) \{Hasan and Reza\}
4. Those who \( \text{StudyIn} \) \( \text{IUT} \) \( \rightarrow \) they are \( \text{SupervisedBy} \) a person who is \( \text{Cooperator} \) with another person who \( \text{WorksOn} \) a project which its \( \text{Patronage} \) is \( \text{MIT} \) University. \{Reza, Ali and Ahmad\}
5. Those who \( \text{LiveIn} \) \( \text{Isfahan} \) \( \rightarrow \) they are \( \text{SupervisedBy} \) a person who is \( \text{Cooperator} \) with another person who \( \text{WorksOn} \) a project which its \( \text{Patronage} \) is \( \text{MIT} \) University. \{Ali and Ahmad\}

In these rules, there is more than one relation in at least one item.

And also if we add \( \text{(Hasan Knows Ali)} \) and \( \text{(Reza Knows Ahmad)} \) to the graph presented in Figure 2, the following rules would be generated too.

6. Those who their \( \text{Health Condition} \) is \( \text{Good} \) \( \rightarrow \)
they Know people who are Supervised By a person who is Cooperator with another person who Works on a project which its Patronage is MIT University. {Hasan and Reza}

(7) Those who Live In a place Near by a city which its Climate Type is Humid → they Know people who are Supervised By a person who is Cooperator with another person who Works on a project which its Patronage is MIT University. {Hasan and Reza}

As rules #3 to #7 show, in at least one item, there is more than one relation. Also the rule #3 can be rewritten in this form:

- \( \text{HealthCondition}(\text{Good}) \rightarrow \text{LiveIn}(\text{Near}(\text{ClimateType}(\text{Humid}))) \)

In a similar way, the rule #7 can be rewritten in the below form:

- \( \text{LiveIn}(\text{Near}(\text{ClimateType}(\text{Humid}))) \rightarrow \text{Know}(\text{SupervisedBy}(\text{Cooperator}(\text{WorksOn}(\text{Patronage}(\text{MIT})))) \)

### 3 Related Work

In the past years many machine-learning algorithms have been applied to traditional datasets successfully in order to discover useful and previously unknown knowledge and patterns. Although these machine learning algorithms are useful, in contrast to our proposed algorithm they are not able to extract ARs with multiple relations.

The ARM problem as first introduced in [15, 32] has goal to find frequent itemsets and to generate primitive and simple ARs. Nowadays there are many ARM algorithms which can work with traditional datasets [33–35]. These algorithms are classified into two main categories: Apriori based [16, 36] and FP-Tree based [18–20]. FP-Tree based approaches extract ARs from graph structured data by using frequent subgraph and frequent sub-tree techniques [18, 37]. The logic behind these algorithms is based on identifying repeated sub graphs in the entire graph. Although this is an interesting approach but it is not appropriate for our work, because these algorithms do not consider relations among entities and also in our proposed scheme, each entity is not replicated in the entire graph more than once.

ARM problem has different settings. The most related work to our problem are categorized into three groups: multi-level ARM methods, several relational database based methods and semantic web data based methods. Among these three settings, multi-level ARM is the most similar problem to the problem of Mining Multi-Relation Association Rules.

The problem of mining multi-level association rules was first introduced in [38]. Many studies on ARM find rules at single concept level. Mining association rules at multiple concept levels may lead to the discovery of more specific and concrete knowledge from data and often carry more specific and concrete information than primitive ARs. Mining multi-level association rules uses concept hierarchies, also called taxonomies and defined as relations of type ‘is-a’ between objects, to extract rules whose items belong to different levels of abstraction. There are applications which need to find associations at multiple concept levels. For example, besides finding “80% of customers that purchase milk, may also purchase bread”, it also could be informative to show that “75% of people buy \( \text{wheat bread} \) if they buy \( 2\% \text{ milk} \)” or even “75% of people buy \( \text{Dairyland 2}\% \text{ milk} \) if they buy \( \text{Wonder wheat bread} \)”. To discover multi-level association rules, one needs to provide data at multiple levels of abstraction, and also provide efficient methods for multiple-level rule mining. Figure 3 shows an example of items taxonomy [38]. Hierarchy levels can be conceptual and attribute based or can be time/place based [39].

Nowadays there are many algorithms and variations for mining multi-level ARs that almost all of them are based on data hierarchy and tree structure with different settings [40, 41]. For instance, [42] proposes a fast and an efficient algorithm (SC-BF Multilevel) with single scan of database for mining multi-level association rules in large databases to finding maximum frequent itemset at lower level of abstraction. Similar to primitive ARs, in multiple-level ARs, datasets can be extracted in a way that only positive and negative rules are extracted [30] or only rules related to special items are extracted [43, 44]. These rules can be restricted to the concepts at same level of a hierarchy or

### Table 1. Meaning of graph example entities

<table>
<thead>
<tr>
<th>Entity</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tehran, Shiraz, Isfahan, Yazd, Kerman</td>
<td>City</td>
</tr>
<tr>
<td>Reza, Nematbakhsh, Saraee, Hasan, Ali, Ahmad</td>
<td>Person</td>
</tr>
<tr>
<td>IUT, MIT</td>
<td>University</td>
</tr>
</tbody>
</table>
Multi-level association rules are different to Multi-Relation Association Rules in the sense that in multi-level association rules, in rules’ items, there is only one relation but there are more than one entities that are derived from data hierarchy. Thus, the proposed algorithms for mining multi-level association rules are not applicable to mining Multi-Relation Association Rules.

Studies for AR discovery in Multi-Relational Data Mining [15, 46] are rooted in the field of Inductive Logic Programming (ILP) [47]. In ILP both relational data and relational patterns are expressed in first-order logic and the logical notions of generality order and of the downward/upward refinement operator on the space of patterns are used to define both the search space and the search strategy. WMRAR [28] and its variants [29] are the most popular approaches that use ILP to extract ARs. However, with larger search spaces and more complex evaluation of a single candidate pattern, these approaches are inherently computation-wise and thus efficient methods such as [48] could be used.

There are many other works on mining ARs from relational databases that are not rely on ILP. Some works use SQL commands to extract ARs [49]. In these work instead of frequent itemsets, frequent queries are used where a query support is the number of tuples that it returns [50]. Other works use other algorithms or use extended SQL to extract ARs [51, 52]. Some improvements on query based ARM techniques have been proposed in [53, 54]. In fact all these work hoard data from multiple relations (tables) by different queries and based on the relations among queries, launch to discover ARs with at most one relation in each item. For example, the following rules are extracted from a multi-relational database [28] (likes, has and prefers are database tables):

- \( \text{likes}(\text{KID}, \text{piglet}), \text{likes}(\text{KID}, \text{ice-cream}) \rightarrow \text{likes}(\text{KID}, \text{dolphin}) \)
- \( \text{likes}(\text{KID}, A), \text{has}(\text{KID}, B) \rightarrow \text{prefers}(\text{KID}, A,B) \)

In many ARM researches, the researchers work on data with tabular structure. In [36, 55, 56] a number of methods have been introduced that receive data in graph structure and extract ARs from these data. Unfortunately, these works are not suitable for our problem since they find only maximal frequent itemsets instead of all frequent itemsets and also, like other ARM algorithms, do not consider relations among entities and generate ARs with at most one relation in each item.

A transaction in a database typically consists of transaction identifier, customer identifier, transaction date (or transaction time), and the items purchased together in the transaction. In semantic web data there is no exact definition of transactions and traditional ARM algorithms are not able to extract ARs from semantic web data directly. In [57], an algorithm has been introduced to extract association rules from semantic web data through mining patterns following an extended SPARQL syntax provided by the end user. In fact, this work converts semantic web data into traditional transactions and then employs traditional algorithms to extract primitive and simple ARs.

As mentioned earlier in section 2.2, in RDF structure each data statement is called a triple and is identified by three values: subject, predicate and object. In order to generate transactions, it is possible to use one of these three values to group transactions (transaction identifier) and use one of the remaining values as
transaction items. Six different combinations of these values along with their usage are shown in Table 2 [58]. For example, grouping triples by predicates and using objects for generating transactions has usage in clustering. This approach has two drawbacks. First, it extracts primitive and simple ARs not Multi-Relation Association Rules, and also it eliminates one part of triples parts and does not consider it in mining process.

Table 2. Combinations of triple parts [58]

<table>
<thead>
<tr>
<th>Row</th>
<th>Context</th>
<th>Target</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Subject</td>
<td>Predicate</td>
<td>Schema discovery</td>
</tr>
<tr>
<td>2</td>
<td>Subject</td>
<td>Object</td>
<td>Basket analysis</td>
</tr>
<tr>
<td>3</td>
<td>Predicate</td>
<td>Subject</td>
<td>Clustering</td>
</tr>
<tr>
<td>4</td>
<td>Predicate</td>
<td>Object</td>
<td>Range discovery</td>
</tr>
<tr>
<td>5</td>
<td>Object</td>
<td>Subject</td>
<td>Topical clustering</td>
</tr>
<tr>
<td>6</td>
<td>Object</td>
<td>Predicate</td>
<td>Schema matching</td>
</tr>
</tbody>
</table>

Linked Data is an effort to implement semantic web data. There are a number of methods to extract ARs from linked data [59, 60]. The first one is based on transactions and extracts primitive ARs in a way that the items of the generated rules do not have any relation. The second one considers the input data as a directed graph with labeled edges and regardless to transactions concept, extracts primitive ARs in a way that the items of the generated rules consist of one entity and one relation.

All of the above work have a common feature: the generated rules do not have several relations in their items. To the best of our knowledge, this paper is the first work that introduce the problem of Multi-Relation Association Rules and proposes an algorithm to solve it.

4 Data Sources

The proposed algorithm receives the required data as a directed graph with labeled edges like the presented graph in Figure 2, in a way that vertices identify entities or values and edges identify a relation between corresponding vertices. It then extracts frequent itemsets by traversing the input graph recursively. The input dataset can be in different structures, such as relational databases or heterogeneous semantic web data, which should be converted to the mentioned graph.

In this section, we will show how relational databases and semantic web data are converted to the required format of the proposed algorithm which is a directed graph with labeled edges. Next sections present the proposed algorithms and related data structures.

4.1 Relational Databases

Relational database refers to those databases that data are distributed over several tables (schemes) and there are some relations among them.

We define a new concept named “Copulative Entity” which refers to those entities which have edges in the input graph from some entities and also have edges to some other entities. In order to extract Multi-Relation Association Rules, the existence of such copulative entities is essential. In relational databases, copulative entities are those entities that are stored in a table and have primary key (independent entities) and also their key is used in other tables as foreign key of other entities. In our approach, each table describes at most one type of copulative entity (e.g. Persons) and the name of fields construct edges between the copulative entity and the value of the field. For example, in Table 4 there are some information about “Kerman” and also in Table 3 “Kerman” is the value of attribute Live in of entity “Reza”. In fact, “Kerman” is a primary key in Table 4 and a foreign key in Table 3 and hence it is a copulative entity. As another example, in Table 3 which describes persons, “Nematbakhsh” is a copulative entity, because it has a primary key and also its key is the foreign key of attribute Knows of entity “Saraee” at the same table.

In the proposed algorithm, the existence of copulative entities is crucial. Because they act as median vertices (connector) between adjacent edges (relations) in the input graph and if they do not exist, mining Multi-Relation Association Rules is impossible.

The process of converting a relational database to a directed graph with labeled edges is as simple as follows: first, for each copulative entity, a vertex is constructed. Afterwards, for each attribute of each copulative entity, an edge is made out from the corresponding vertex. Finally, the value of attribute constructs the target vertex of the edge and the name of the attribute constructs the label of the edge. If the value of the attribute is a copulative entity (foreign key), the edge is connected to the vertex corresponding to that copulative entity.

For example, consider data depicted in Table 3 to Table 5. In these tables, the underlined attributes stand for copulative entities. These tables are equivalent to the graph presented in Figure 2.
Table 3. Example of persons

<table>
<thead>
<tr>
<th>Person</th>
<th>Health Condition</th>
<th>Study in</th>
<th>Live in</th>
<th>Supervised By</th>
<th>Cooperator</th>
<th>Work On</th>
<th>Knows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hasan</td>
<td>Good</td>
<td>Null</td>
<td>Yazd</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
</tr>
<tr>
<td>Reza</td>
<td>Good</td>
<td>IUT</td>
<td>Kerman</td>
<td>Sarae</td>
<td>Null</td>
<td>Null</td>
<td>Hasan</td>
</tr>
<tr>
<td>Ali</td>
<td>Null</td>
<td>IUT</td>
<td>Isfahan</td>
<td>Sarae</td>
<td>Null</td>
<td>Null</td>
<td>Ahmad</td>
</tr>
<tr>
<td>Ahmad</td>
<td>Null</td>
<td>IUT</td>
<td>Isfahan</td>
<td>Nematbakhsh</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
</tr>
<tr>
<td>Sarae</td>
<td>Null</td>
<td>Null</td>
<td>Kerman</td>
<td>Mr A</td>
<td>Null</td>
<td>Mr A</td>
<td>Nematbakhsh</td>
</tr>
<tr>
<td>Nematbakhsh</td>
<td>Null</td>
<td>Null</td>
<td>Isfahan</td>
<td>Null</td>
<td>Mr B</td>
<td>Null</td>
<td>Mr A</td>
</tr>
<tr>
<td>Mr A</td>
<td>Null</td>
<td>Null</td>
<td>Shiraz</td>
<td>Null</td>
<td>Project A</td>
<td>Null</td>
<td></td>
</tr>
<tr>
<td>Mr B</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>Null</td>
<td>Project B</td>
<td>Nematbakhsh</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Example of cities

<table>
<thead>
<tr>
<th>City</th>
<th>Climate Type</th>
<th>Near</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yazd</td>
<td>Null</td>
<td>Tehran</td>
</tr>
<tr>
<td>Kerman</td>
<td>Null</td>
<td>Shiraz</td>
</tr>
<tr>
<td>Tehran</td>
<td>Humid</td>
<td>Null</td>
</tr>
<tr>
<td>Shiraz</td>
<td>Humid</td>
<td>Null</td>
</tr>
</tbody>
</table>

Table 5. Example of projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Patronage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project A</td>
<td>MIT</td>
</tr>
<tr>
<td>Project B</td>
<td>MIT</td>
</tr>
<tr>
<td>Project B</td>
<td>IUT</td>
</tr>
</tbody>
</table>

4.2 Semantic Web Data

As mentioned earlier in Section 2.2, each instance of semantic web data would has a subject-predicate-object format. These data are stored in files with different syntaxes. Regardless to the syntax of semantic web data, they can be extracted by SPARQL commands and be shown in simple subject-predicate-object format. Suppose the semantic web data of Table 6 which are in triple format.

In semantic web data, copulative entities are those entities that are laid in both subject and object parts, hence in this paper only those semantic web data are suitable to be used that some entities are appear in subjects of some triples and also in objects of some other triples too. For example, in the data presented in Table 6 “Sarae” is a copulative entity, because it is located in both subject and object parts.

The conversion of semantic web data to an appropriate directed graph with labeled edges is very simple and straightforward. The subject and object parts of triples construct graph vertices and predicates construct graph edges to connect corresponding subject to corresponding object. The result of converting semantic web data presented in Table 6 to a directed graph with labeled edges, has been depicted in Figure 2.

5 Methodology, Concepts and Data Structures

In this section, the proposed methodology for solving the problem of mining Multi-Relation Association Rules along with related concepts and data structures are discussed in detail.

5.1 Problem Description

In this section, some details and concepts related to the Multi-Relation Association Rules are presented.

As rules #3 to #7 in Section 2.3 show, in Multi-Relation Association Rules only median relations and endpoint entities are shown and median entities are not shown.

For example, consider rule #3 which indicates “Those who their Health Condition is Good → they Live In a place Near by a city which its Climate Type is Humid {Hasan and Reza}”. Figure 4 is a subgraph of Figure 2 and also is the data source of rule #3. In Figure 4, Humid and Good are endpoint entities: LiveIn, Near, ClimateType and HealthCondition are median relations; Yazd, Tehran, Shiraz and Kerman are median entities (copulative entities) and finally Hasan and Reza are the entities which satisfy rule #3. As mentioned before, in Multi-
Table 6. Example of semantic web data in triple format

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hasan</td>
<td>Health Condition</td>
<td>Good</td>
</tr>
<tr>
<td>Hasan</td>
<td>Live in</td>
<td>Yazd</td>
</tr>
<tr>
<td>Reza</td>
<td>Health Condition</td>
<td>Good</td>
</tr>
<tr>
<td>Reza</td>
<td>Live in</td>
<td>Kerman</td>
</tr>
<tr>
<td>Reza</td>
<td>Study in</td>
<td>IUT</td>
</tr>
<tr>
<td>Reza</td>
<td>Supervised By</td>
<td>Saraee</td>
</tr>
<tr>
<td>Reza</td>
<td>Knows</td>
<td>Hasan</td>
</tr>
<tr>
<td>Ali</td>
<td>Study in</td>
<td>IUT</td>
</tr>
<tr>
<td>Ali</td>
<td>Live in</td>
<td>Isfahan</td>
</tr>
<tr>
<td>Ali</td>
<td>Supervised By</td>
<td>Saraee</td>
</tr>
<tr>
<td>Ali</td>
<td>Knows</td>
<td>Ahmad</td>
</tr>
<tr>
<td>Ahmad</td>
<td>Study in</td>
<td>IUT</td>
</tr>
<tr>
<td>Ahmad</td>
<td>Live in</td>
<td>Isfahan</td>
</tr>
<tr>
<td>Ahmad</td>
<td>Supervised By</td>
<td>Nemathbakhsh</td>
</tr>
<tr>
<td>Saraee</td>
<td>Cooperator</td>
<td>Mr A</td>
</tr>
<tr>
<td>Saraee</td>
<td>Live in</td>
<td>Kerman</td>
</tr>
<tr>
<td>Nemathbakhsh</td>
<td>Cooperator</td>
<td>Mr B</td>
</tr>
<tr>
<td>Nemathbakhsh</td>
<td>Knows</td>
<td>Mr A</td>
</tr>
<tr>
<td>Nemathbakhsh</td>
<td>Live in</td>
<td>Isfahan</td>
</tr>
<tr>
<td>Mr A</td>
<td>Work On</td>
<td>Project A</td>
</tr>
<tr>
<td>Mr A</td>
<td>Live in</td>
<td>Shiraz</td>
</tr>
<tr>
<td>Mr B</td>
<td>Work On</td>
<td>Project A</td>
</tr>
<tr>
<td>Mr B</td>
<td>Knows</td>
<td>Nemathbakhsh</td>
</tr>
<tr>
<td>Yazd</td>
<td>Near</td>
<td>Tehran</td>
</tr>
<tr>
<td>Kerman</td>
<td>Near</td>
<td>Shiraz</td>
</tr>
<tr>
<td>Tehran</td>
<td>Climate Type</td>
<td>Humid</td>
</tr>
<tr>
<td>Shiraz</td>
<td>Climate Type</td>
<td>Humid</td>
</tr>
<tr>
<td>Project A</td>
<td>Patronage</td>
<td>MIT</td>
</tr>
<tr>
<td>Project B</td>
<td>Patronage</td>
<td>MIT</td>
</tr>
<tr>
<td>Project B</td>
<td>Patronage</td>
<td>IUT</td>
</tr>
</tbody>
</table>

Relation Association Rules, only median relations and endpoint entities are shown and median entities are not shown. Hence in rule #3, entities Yazd, Kerman, Tehran and Shiraz are not shown, because they are median entities and only endpoint entities Humid and Good along with median relations LiveIn, Near, ClimateType and HealthCondition are shown.

- Rule #3: HealthCondition(Good) → LiveIn(Near(ClimateType(Humid)))

In the proposed algorithms, endpoint entities differ to conventional sink entities. In each iteration of the main algorithm, endpoint entities change and are become the entities where the process of extracting ItemChains is started from. As it will be stated later, the algorithm GenerateItemChains for each vertex is called only once. By calling GenerateItemChain on a vertex, that vertex becomes an endpoint entity (vertex). For example, in graph Figure 2 even if an edge is made out from the entity Humid to the entity IUT; by calling algorithm GenerateItemChain on entity Humid, this entity still remains as an endpoint entity even though it has a relation to another entity.

Each ItemChain is a set of entities that are connected to an endpoint entity with common relations. Details of data structures will be discussed later.

5.2 Working Process

The proposed algorithm is in fact an extended version of Apriori algorithm which extracts Multi-Relation Association Rules from directed graphs with labeled edges. As mentioned earlier, in the proposed problem, there are different relations among entities that must be considered in the mining process. Also the input data are heterogeneous and there are no exact definitions of transactions and hence the Apriori algorithm must be changed so that it does not need transactions and would also generate rules with several relations from heterogeneous data. For this purpose, the proposed algorithm regardless of the concept of transactions and by considering relations among entities, after generating ItemChains generates 2-Large ItemChains and feeds them to the extended Apriori algorithm to generate Larger ItemChains. Finally, Multi-Relation Association Rules are generated from L-Large ItemChains (L ≥ 2).

Figure 5 shows the workflow of the mining Multi-Relation Association Rules process.

5.3 ItemChains

ItemChain is a new and important concept which is employed in this paper. Each ItemChain shows that a set of entities are connected to an endpoint entity via common relations. For example, in Figure 4 (Hasan and Reza) construct an ItemChain, because the entities Hasan and Reza are both connected to...
the endpoint entity Humid through relations LiveIn, Near and ClimateType. Each ItemChain includes a list of connected entities (LOE), a list of median relations (LOR), an endpoint entity and the support value of the ItemChain. In fact, the concept ItemChain is equivalent to the concept Itemset in Apriori algorithm.

Figure 6 shows the structure of ItemChain. Consider rule #3 in Section 2.3 whose data source has been depicted in Figure 4. The consequent part of this rule is an ItemChain:

- “Those who LiveIn a place Near by a city which its Climate Type is Humid” {Hasan and Reza}

This ItemChain contains these parts:

- **ChainID**: a numerical ID for identifying ItemChain. This number starts incrementally from 1.
- **List of Entities (LOE)**: a set of entities which are connected to the endpoint entity with common relations (LOR). In the above example, Hasan and Reza are laid in this part.
- **List of Relations (LOR)**: a set of relations which connect several entities (LOE) to the endpoint entity. In above example, Live in, Near and Climate Type are laid in this part.
- **Endpoint Entity**: identifies an endpoint entity which several entities (LOE) are indirectly connected to it through common relations (LOR). In above example Humid is an endpoint entity.

- **Support**: the frequency rate of ItemChain. That is what percent of entities are connected to the endpoint entity via LOR. This value is equivalent to the number of LOE part’s entities divided by the number of entire graph entities. In above example, \( \text{Support} = \frac{2}{19} \).

### 5.4 2-Large ItemChain

The second step of mining Multi-Relation Association Rules is to generate 2-Large ItemChains from the extracted ItemChains. Two ItemChains that their LOE parts have many common entities are combined to generate a 2-ItemChain. A 2-ItemChain is large when the intersection count of LOE parts of its two ItemChains is equal to or greater than the predefined minimum support value (MinSup) that means these two ItemChains are co-occurred abundantly.

In order to generate 2-Large ItemChains, the proposed algorithm compares all extracted ItemChains two by two and adds those two ItemChains which their LOE parts intersection count is equal to or greater than MinSup value, to the LargeItemChains list. In order to store L-Large ItemChains (L ≥ 2), a data structure is employed that its image is depicted in Figure 7.
5.5 Larger ItemChains

Apriori algorithm generates a (L+1)-candidate itemset by combining two L-large itemsets that their L-1 first items are equal and then makes a candidate itemset with length L+1 [16]. A candidate itemset is large when its occurrence is equal to or greater than $MinSup$ value.

In our approach, each $L$-Large ItemChain has $L$ ChainIDs in its LOC part (see Figure 7). In order to generate $(L+1)$-Large ItemChains, the proposed algorithm employs those two $L$-Large ItemChains that their L-1 first ChainIDs of LOC parts are equal. For example suppose \{1, 2\} and \{1, 3\} are LOC parts of two 2-Large ItemChains. Combining \{1, 2\} and \{1, 3\} results \{1, 2, 3\} as LOC part of a new 3-ItemChains. If ItemChains with ChainID 1, 2 and 3 have many common entities in their LOE part, these three ItemChains construct a new 3-Large ItemChains. Also suppose \{1, 2, 3\} and \{1, 2, 5\} are LOC parts of two 3-Large ItemChains. Combining \{1, 2, 3\} and \{1, 2, 5\} results \{1, 2, 3, 5\} as LOC part of a new 4-ItemChains. Similarly if ItemChains with ChainID 1, 2, 3 and 5 have many common entities in their LOE part, these four ItemChains make a new 4-Large ItemChains. Inter bracket numbers indicate ChainIDs in LOC part. Generating larger ItemChains is continued until generating new candidate ItemChains is not possible.

The image of an $L$-Large ItemChain ($L \geq 2$) is depicted in Figure 7.

Each Large ItemChain has three parts:

- **List of ChainIDs (LOC):** ChainIDs of ItemChains that have many common entities.
- **Intersection Count:** the intersection count of LOE parts of those ItemChains that their ChainIDs are laid in the LOC part.
- **Support:** the frequency rate of this Large ItemChain which is accounted by this formula:

$$Support = \frac{\text{Intersection Count of LOE Parts}}{\text{Entire Graph Entities Count}}$$

5.6 Association Rules

Finally the proposed algorithm generates Multi-Relation Association Rules by employing Large ItemChains. Each generated rule includes several ItemChain and each ItemChain contains one or more relations in LOR part. The algorithm generates rules with only one ItemChain in the consequent part. The logic behind this work is that usually the number of generated rules is enormous, thus with only one item in the consequent part, this number would be reduced. Additionally when complex rules are generated (rules with several items in the consequent part) it is hard to use them in the real world applications. Finally generated rules with confidence equal to or greater than predefined confidence value ($MinConf$) are marked as Multi-Relation Association Rules.

Multi-Relation Association Rules are generated by using Large ItemChains in a way that one ChainID of LOC part makes consequent part and the rest ChainIDs make antecedent part of the rule. For each Large ItemChain this process is repeated until each ChainID is laid in the consequent part once.

Figure 8 shows the structure of a Multi-Relation Association Rule.

Each Multi-Relation Association Rule consists of these fields:

1. **Antecedent:** list of ItemChains as antecedent
2. **Consequent:** an ItemChain as consequent
3. **Rule Confidence** value
4. **Rule Support** value

Rule confidence is equal to the intersection count of existing entities in the LOE parts of all ItemChains in the whole rule divided by the intersection count of existing entities in the LOE parts of all ItemChains in the rule’s antecedent part. Rule support is equal to the support of the Large ItemChain which has been employed to generate this rule.
5.7 EntityInfo Data Structure

The simplest and the fastest way to retain the input graph (which is a directed graph with labeled edges) in main memory is to use cube (3D array) as data structure, in a way that the first dimension stores source vertices, the second dimension stores destination vertices and the third dimension stores relation between two vertices. Each entry value is 0 or 1. If the value of the \((i,j,k)\)th entry is equal 1, that is there is a relation of type \(k\) from the \(i\)th vertex to the \(j\)th vertex. Although cube structure is too fast and easy to use, to retain such cube, a large memory space is required. The solution of this problem is to use linked list as data structure. To store information about each entity (including relations and other entities that are connected to the entity), there is an **EntityInfo** structure with these attributes:

1. **EndpointEntity (Vertex)**: identifies an entity which is a vertex of graph.
2. A **Linked List** that its entries have two parts:
   a. **Relations (Edges)**: identifies relations (edges) that are entered to the **EndpointEntity**.
   b. **Input Entities (Vertices)**: pointer to a list of entities (vertices) which refer to the **EndpointEntity** through corresponding relation (edge).

The image of **EntityInfo** structure is depicted in Figure 9.

By this data structure policy, in fact data are grouped based on destination vertices (endpoint entities), because for each vertex of graph, the algorithm defines an **EntityInfo** instance and then specifies that based on each edge (relation), what other vertices (entities) refer to this vertex (endpoint entity). This grouping reason is to make the mining process faster, based on the proposed algorithm. Finally there is a linked list named **List_EntityInfo** that its entries refer only to **Large EntityInfo** instances. An **EntityInfo** is large when the number of entities connected to it divided by the number of vertices in the input graph is more than **MinSup** value.

5.8 MinLevel and MaxLevel

As mentioned earlier, the proposed algorithm is able to generate **ItemChains** with several relations in LOR part. To determine the number of relations, the algorithm receives the minimum and the maximum number of relations in LOR part as input parameters. These parameters are named **MinLevel** and **MaxLevel** respectively.
6 Algorithms

In this section, the proposed algorithms pseudo codes are described in detail. The name of the main proposed algorithm is MRAR (Multi-Relation Association Rules). The MRAR algorithm (Algorithm 1) calls three other sub-algorithms and its workflow is as follows: First, after constructing large EntityInfo instances, the GenerateItemChains algorithm (Algorithm 2) is called to traverse the input graph recursively and generate all possible ItemChains. Afterwards the Generate2LargeItemChains algorithm (Algorithm 3) is invoked to generate larger ItemChains and feed them to the MRAR algorithm. Then the MRAR algorithm generates larger ItemChains in a repetitive process. Finally the GenerateLargeItemChains algorithm (Algorithm 4) is called to generate Multi-Relation Association Rules based on the generated Large ItemChains. These algorithms are as follow:

6.1 Algorithm 1: MRAR

Algorithm MRAR is the main algorithm that after invoking GenerateItemChains and Generate2LargeItemChains, generates Large ItemChains and finally invokes GenerateRules to generate Multi-Relation Association Rules. The pseudo code of this algorithm is depicted in Algorithm 1.

MRAR algorithm receives a dataset convertible to a directed graph with labeled edges, along with minimum support and minimum confidence values and minimum and maximum number of relations in ItemChains as input parameters. The pre-process step is done in lines 20 and 21. In pre-process, the input data are converted to appropriate graph and Large EntityInfo instances are constructed from the graph. An EntityInfo is large when the number of entities connected to it divided by the number of vertices in the input graph is more than MinSup value. After pre-process, all ItemChains are generated by GenerateItemChains algorithm. This algorithm starts its process from a vertex and an incoming edge of that vertex and discovers all entities that with common edges are directly or indirectly connected to that vertex. This vertex is called EndpointEntity. This process is done in lines 22 to 26. After generating all ItemChains, all Large ItemChains are generated by invoking Generate2LargeItemChains algorithm in line 27. Then the loop between lines 29 to 45 generates all Large ItemChains and this generation is continued until generating larger ItemChains is impossible. In each run of this loop, all Large ItemChains with L ChainIDs in LOC part are assessed and new candidate ItemChains with L+1 ChainIDs in LOC part are generated. Each loop’s run (lines 32-36), uses previous loop’s run results which is stored in LLICs. Lines 32 and 33 state that all Large ItemChains with L ChainIDs in LOC part have to be compared two by two and this comparison is done in line 33. If two Large ItemChains with L ChainIDs in LOC part are combinable (their L-1 first ChainIDs are equal), they are combined with CombineAndSort function generating new candidate ItemChains with L+1 ChainIDs in LOC part. After generating all candidate ItemChains with L+1 ChainIDs in LOC, in lines 38 to 43 all Large ItemChains are selected from candidate ItemChains collection and then added to the Large ItemChains collection (LLICs). Finally, line 44 adds generated Large ItemChains with L+1 ChainIDs in LOC part to the collection of all Large ItemChains (AllLLICs). After generating all possible Large ItemChains, Multi-Relation Association Rules are generated by invoking GenerateRules algorithm in line 46.

6.2 Algorithm 2: GenerateItemChains

Algorithm GenerateItemChains traverses the input graph recursively and generates ItemChains. This algorithm receives a vertex as endpoint entity and one of its incoming edges and then finds entities connected to the endpoint entity directly or indirectly with common relations. If the number of relations from the entities to the endpoint entity is between MinLevel and MaxLevel, the algorithm adds generated ItemChain to the List_ItemChains. This algorithm is depicted in Algorithm 2.

GenerateItemChains generates ItemChains with the number of relations between MinLevel and MaxLevel. This algorithm is invoked by MRAR algorithm and starts its process from a vertex of graph and one of its incoming relations. Parameter EndpointEntity indicates start vertex and parameter Relations_Parameter indicates one or more edges between vertices in parameter Entities_Parameter and the EndpointEntity. In line 16, all vertices that are connected to the EndpointEntity through relations in Relations_Parameter are extracted. In lines 17 and 19, it is determined if the current level is between valid levels and also the support value of connected vertices is equal to or greater than MinSup value. If so, a new ItemChain is generated in line 21. Lines 24 to 29 re-traverse the input graph to generate more ItemChains. Line 24 assesses if the number of current relations (Level) is less than maximum possible relations (MaxLevel). If so, all incoming edges of the connected vertices are extracted in line 25 and unified by UnionIncomingEdgesOf function and then GenerateItemChains algorithm is re-called by adding all of the extracted edges to the current relations set. All generated ItemChains are retained in List_ItemChains.
Algorithm 1 MRAR: Mining Multi-Relation Association Rules

1: function MRAR(DS, MinSup, MinConf, MinLevel, MaxLevel)  
2: Inputs  
3: $DS$ $\triangleright$ a dataset convertible to a directed graph with labeled edges  
4: $MinSup$ $\triangleright$ Minimum support value  
5: $MinConf$ $\triangleright$ Minimum confidence value  
6: $MinLevel$, $MaxLevel$ $\triangleright$ Minimum and maximum number of relations in each ItemChain  
7: EndInputs  
8: Outputs  
9: $AllLICs[]$ $\triangleright$ List of Large ItemChains  
10: $Rules[]$ $\triangleright$ Multi-Relation Association Rules  
11: EndOutputs  
12: Variables  
13: $LLICs[]$ $\triangleright$ List of Large ItemChains  
14: $Candidates[]$ $\triangleright$ Lists that maintain ChainIDs Set  
15: $CIS$ $\triangleright$ Set of ChainIDs  
16: $LIC1$, $LIC2$ $\triangleright$ Large ItemChain  
17: $List_{EntityInfo}[]$ $\triangleright$ List of large EntityInfo instances  
18: $List_{ItemChains}[]$ $\triangleright$ Global list of ItemChains  
19: EndVariables  
20: convert input data to a directed graph with labeled edges  
21: construct Large EntityInfo instances from the input graph and add to List_{EntityInfo}  
22: for all $(EntityInfo$ in List_{EntityInfo}) do  
23: for all $(Relation$ in EntityInfo.Relations) do  
24: GenerateItemChains $(EntityInfo.EndpointEntity, Relation, EntityInfo.EndpointEntity, 1)$ $\triangleright$ adds ItemChains to List_{ItemChains}  
25: end for  
26: end for  
27: $LLICs = AllLICs = Generate2LargeItemChains(List_{ItemChains})$  
28: $L = 1$  
29: repeat  
30: $L = L + 1$  
31: $Candidates = null$  
32: for all $(LIC1, LIC2$ in $LLICs)$ do  
33: if $(LIC1.LOC[1..L-1] = LIC2.LOC[1..L-1])$ then  
34: $Candidates.Add(CombineAndSort(LIC1.LOC[1..L], LIC2.LOC[L]))$  
35: end if  
36: end for  
37: $LLICs = null$  
38: for all $(CIS$ in $Candidates)$ do  
39: Calculate IntersectionCount and Support of CIS  
40: if $(Support(CIS) \geq MinSup \text{ AND all subsets of } CIS \text{ are Large})$ then  
41: $LLICs = LLICs \cup CIS$  
42: end if  
43: end for  
44: $AllLICs = AllLICs \cup LLICs$  
45: until $(Candidates.Length = 0)$  
46: $Rules = GenerateRules(AllLICs)$  
47: return $AllLIC$, $Rules$  
48: end function
Algorithm 2 GenerateItemChains: generating ItemChains

1: function GenerateItemChains(EndpointEntity, Relations_Parameter[], Entities_Parameter[], Level)
2:   Inputs
3:       EndpointEntity \[\text{A vertex of graph which the algorithm starts search from it}\]
4:       Relations_Parameter \[\text{Common relations between several vertices and the EndpointEntity}\]
5:       Entities_Parameter \[\text{Vertices connected to EndpointEntity through Relations_Parameter}\]
6:       Level \[\text{Number of relations in ItemChain, initially it is 1}\]
7:   EndInputs
8:   Outputs
9:       List_ItemChains[] \[\text{List of all ItemChains}\]
10:  EndOutputs
11: Variables
12:       Entities_Var \[\text{List of entities}\]
13:       Relations_Var \[\text{List of relations}\]
14:       Support \[\text{Support value of an ItemChain}\]
15: EndVariables
16: if (Level \(\geq\) MinLevel AND Level \(\leq\) MaxLevel) then
17:   Support = Entities_Var.Count \(\div\) Graph.NumberOfVertices
18: if (Support \(\geq\) MinSup) then
19:   ChainID = ChainID + 1
20:   List_ItemChains.Add(new ItemChain(ChainID, Entities_Var, Relations_Parameter, EndpointEntity, Support))
21: end if
22: end if
23: if (Level < MaxLevel) then
24:   Relations_Var = UnionIncomingEdgesOf(Entities_Var)
25: for all (Relation in Relations_Var) do
26:   GenerateItemChains(EndpointEntity, Relations_Parameter \(\cup\) Relation, Entities_Var, Level + 1)
27: end for
28: end if
29: end function

Algorithm 3 Generate2LargeItemChains: generating 2-Large ItemChains

1: function Generate2LargeItemChains(List_ItemChains[])
2:   Inputs
3:       List_ItemChains[] \[\text{List of all ItemChains}\]
4:   EndInputs
5:   Outputs
6:       LLICs[] \[\text{List of all Large ItemChains with two ChainIDs in LOC part}\]
7:   EndOutputs
8: Variables
9:       IC1, IC2 \[\text{ItemChain}\]
10:      LOE[] \[\text{List of Entities}\]
11: EndVariables
12: for all (IC1, IC2 in List_ItemChains) do
13:   LOE = Intersect(IC1.LOE, IC2.LOE)
14:   Support = LOE.Length \(\div\) Graph.NumberOfVertices
15: if (Support \(\geq\) MinSup) then
16:   LLICs.Add(new LargeItemChain({IC1.ChainID \(\cup\) IC2.ChainID}, LOE.Length, Support))
17: end if
18: end for
19: return LLICs
20: end function
Algorithm 4 GenerateRules: generating ARs by using Large ItemChains

1: function GenerateRules(List_LargeItemChains[])
2: Inputs
3: List_LargeItemChains[]
4: EndInputs
5: Outputs
6: Rules[]
7: EndOutputs
8: Variables
9: LIC
10: Antecedent[]
11: Consequent
12: EndVariables
13: for all (LIC in List_LargeItemChains) do
14: for all (ChainID in LIC.ListOfChainIDs) do
15: Consequent = ChainID
16: Antecedent = LIC.ListOfChainIDs – Consequent
17: Confidence = LIC.Support ÷ Support(Antecedent)
18: if (Confidence ≥ MinConf) then
19: Rules.Add(new Rule(Antecedent, Consequent, Confidence, LIC.Support))
20: end if
21: end for
22: end for
23: return Rules
24: end function

6.3 Algorithm 3: Generate2LargeItemChains

Algorithm Generate2LargeItemChains traverses the List_ItemChains (all ItemChains) and generates all possible Large ItemChains with two ChainIDs in the LOC part. These 2-Large ItemChains are then employed by the MRAR algorithm to generate Larger ItemChains. This algorithm is depicted in Algorithm 3.

This algorithm receives all ItemChains as the input parameter and then generates all possible 2-Large ItemChains. In line 12, all ItemChains are traversed two by two. In line 13, an intersection is made from entities (LOE) of two ItemChains. This intersection returns the common entities of two ItemChains. If the number of common entities divided by the number of all entities is equal to or greater than MinSup value, these two ItemChains generate a 2-Large ItemChain. This algorithm is finished when all ItemChains are compared to each other. After generating all 2-Large ItemChains, the MRAR algorithm generates L-Large ItemChains (L ≥ 3) to be used for generating Multi-Relation Association Rules.

6.4 Algorithm 4: GenerateRules

Algorithm GenerateRules receives all L-Large ItemChains (L ≥ 2) and generates candidate rules with only one ItemChain in the consequence part. If the confidence of a candidate rule is equal to or greater than MinConf value, it is identified as a Multi-Relation Association Rule. This algorithm is depicted in Algorithm 4.

This algorithm receives L-Large ItemChains (L ≥ 2) as input parameter. In line 13, the Large ItemChains are selected one by one. In line 14, all ChainIDs of the selected Large ItemChain are traversed. Lines 15 and 16 construct the antecedent and the consequent parts of a new candidate rule based on the selected Large ItemChain and ChainID, and then line 17 calculates the confidence of this new candidate rule. Line 18 assesses the rule’s confidence. If the confidence value is equal to or greater than MinConf value, then this candidate rule is strong and it is added to the strong rules collection in line 19. Notice that the algorithm puts only one Item in the consequent part in line 15. Finally, all the generated strong rules are returned as Multi-Relation Association Rules.

7 Example

Let us make an example to show how the proposed algorithms work and how the related data structures are filled by corresponding values during the mining process. In this example, the graph depicted in Figure 2 is employed as data source.

7.1 Data Structure

Some parameters of the example are as follows:
In order to generate 3-Large ItemChains, MRAR algorithm combines two by two those 2-Large ItemChains that the first ChainID of their List of ChainIDs (LOC) part are equal. If the number of common entities of the combined ItemChains is equal to or greater than MinSup value and all subsets of the combined ChainIDs are large too, the combination is identified as a 3-Large ItemChain. In Table 8, the combination of \{2, 3\} and \{2, 6\}, would generate a 3-Large ItemChain, because the result of intersecting their entities parts (LOE) is \{Ali, Ahmad\} and its support is 2/19 that is equal to MinSup value. \{2, 3, 6\} constitute the LOC part of this new 3-Large ItemChain. Consider that all subsets of \{2, 3, 6\} are large too.

In order to generate a \((L+1)\)-Large ItemChain, the algorithm combines two \(L\)-Large ItemChains which
their first \textbf{L-1 ChainIDs} of their \textit{List of ChainIDs (LOC)} part are equal and then makes a candidate set with \textbf{L+1 ChainIDs} in the \textit{LOC} part. This candidate set is large if the intersection count of its \textit{ItemChains’} entities divided by the number of all entities is equal to or greater than \textit{MinSup} value and also all the subsets of the \textit{L+1 ChainIDs} are large too.

After generating all \textit{Large ItemChains}, the algorithm generates candidate rules. The candidate rules are identified as \textit{Multi-Relation Association Rules} if their confidence is equal to or greater than \textit{MinConf} value. As mentioned before, the algorithm generates rules with only one \textit{ItemChain} in the consequence part.

For example if \{2, 3, 6\} is the \textit{LOC} part of a \textit{3-Large ItemChain}, the following items are \textit{Multi-Relation Association Rules}:

\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbf{Antecedent} & \textbf{Consequent} & \textbf{Support} & \textbf{Confidence} \\
\hline
2 & 3 & 6 & 2/19 & 1.00 \\
2 & 6 & 3 & 2/19 & 1.00 \\
3 & 6 & 2 & 2/19 & 0.66 \\
\hline
\end{tabular}
\end{center}

For example, the first rule indicates that: “Those who \textit{Live in Isfahan} and also \textit{Study in IUT} \rightarrow they are \textit{Supervised By} a person who is \textit{Cooperator} with another person who \textit{Works on} a project which its \textit{Patronage} is \textit{MIT University}. \{Ali and Ahmad\}”
8 Experimental Results

In order to evaluate the proposed algorithm’s usefulness and its ability in extracting Multi-Relation Association Rules, some experiments have been made on Drugbank dataset that show the proposed method is able to convert the input data to a directed graph with labeled edges, make ItemChains and L-Large ItemChains from the graph and finally generate Multi-Relation Association Rules based on the L-Large ItemChains.

8.1 Dataset

To make experiments on a real-world dataset, Drugbank dataset was used which “is a detailed database on small molecules and biotech drugs. Each drug entry (“DrugCard”) has extensive information on properties, structure, and biology (what the drug does in the body). Each drug can have 1 or more targets, enzymes, transporters, and carriers associated” [61]. Drugbank is a semantic web dataset that has many heterogenous semantic annotations. This dataset has these information:

- Number of triples: 766,920
- Number of entities (graph vertices): 288,871
- Number of relations (predicates): 119

Before feeding the pure extracted data to the algorithm, first they were discretized and then infrequent entities were eliminated. The discretization was applied on objects and it was done by dividing the difference between the minimum value and the maximum value of objects of each predicate into five segments and the value of objects of the predicate was changed to the start value of the segment which they lie in. Then subjects, predicates, and objects that were replicated in less than 10 triples, were identified as infrequent entities and their containing triple was eliminated from the triple set.

After input data were discretized and infrequent entities were eliminated, these new information were obtained:

- Number of triples: 291,082
- Number of entities (graph vertices): 22,952
- Number of relations (predicates): 57
- Number of copulative entities: 546
- Average relation count per endpoint entity: 76

8.2 Experimental set-up

In order to generate ItemChains, the input data should be converted to the algorithm’s standard input format. This conversion is automatically done by our program. The input dataset may be a relational database, a complete semantic web dataset or a subset of it. The input dataset can also be a concatenation of multiple semantic web datasets made by SPARQL commands and linked by data standards [62]. Also, any dataset convertible to the algorithm’s standard format could be employed by the algorithm.

If the input data is a relational database, copulative entities and values of attributes (fields) generate graph vertices and attributes’ names generate graph edges. If the input data is a semantic web dataset, subjects and objects generate graph vertices and predicates generate edges between corresponding subjects and objects.

After providing input data, these steps should be passed to generate Multi-Relation Association Rules:

- Convert data to suitable graph
- Discretize data and eliminate infrequent entities
- Define EntityInfo instances
- Generate ItemChains
- Generate 2-Large ItemChains
- Generate L-Large ItemChains (L ≥ 3)
- Generate Multi-Relation Association Rules

8.3 Results

The proposed algorithm would extract Multi-Relation Association Rules from a directed graph with labeled edges. Since there are no exact definition of transactions in the input graph, the end user should interpret the generated rules and use them in the real world applications himself/herself.

Following are some results obtained by mining Multi-Relation Association Rules from Drugbank dataset [61]. In these results, the range of MinSup values is between 0.04 and 0.28, the MinConf value is 0.8, the MinLevel value is 1 and the MaxLevel value is 3.

Table 9 shows some Multi-Relation Association Rules along with their corresponding confidence and support values discovered by the proposed algorithms from Drugbank dataset. Each generated rule is constructed of several ItemChains. In the ItemChains of the generated rules, the last inner parentheses word identifies an endpoint entity (vertex) and the words before endpoint entity identify relations (LOR). For example, in the antecedent part of the first rule in Table 9, Enzyme and InteractionDrug2 are relations (LOR). This rule indicates the relationship between interactions of two enzymes of some drugs and the interaction type of the drugs. As it was mentioned earlier, in case of generated rules, those entities (LOE) that refer to the endpoint entity via relations in the LOR part, are not shown because their values are not important. The only important goal is to discover similarity of several entities behavior.
Table 9. Examples of discovered association rules along with their confidence and support

<table>
<thead>
<tr>
<th>Rule</th>
<th>Confidence</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type, enzyme, interactionDrug2(enzymes) → Type(drug_interactions)</td>
<td>0.91</td>
<td>0.237</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug2(membrane) → Type(drug_interactions)</td>
<td>0.85</td>
<td>0.212</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug2(membrane) → Type, target, interactionDrug2(targets)</td>
<td>0.86</td>
<td>0.212</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug1(cell) → Type, target, interactionDrug1(targets)</td>
<td>0.80</td>
<td>0.228</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug2(cell) &amp; Type(drug_interactions) → goClassificationComponent, target, interactionDrug2(membrane)</td>
<td>0.79</td>
<td>0.212</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug1(cell) &amp; Type, target, interactionDrug1(targets) &amp; Type(drug_interactions) → goClassificationComponent, target, interactionDrug1(membrane)</td>
<td>0.77</td>
<td>0.221</td>
</tr>
<tr>
<td>goClassificationComponent, target, interactionDrug1(membrane) &amp; Type, target, interactionDrug1(targets) &amp; Type(drug_interactions) → goClassificationComponent, target, interactionDrug1(cell)</td>
<td>0.74</td>
<td>0.221</td>
</tr>
</tbody>
</table>

Table 10. Statistical Results for the Performed Experiments

<table>
<thead>
<tr>
<th>Minimum Support</th>
<th>Number of Large Entities</th>
<th>Number of ItemChains</th>
<th>Number of 2-Large ItemChains</th>
<th>Number of L-Large ItemChains ((L \geq 3))</th>
<th>Number of Rules</th>
<th>Average of Relations Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.04</td>
<td>22</td>
<td>65</td>
<td>470</td>
<td>154780</td>
<td>990186</td>
<td>2.82</td>
</tr>
<tr>
<td>0.06</td>
<td>20</td>
<td>59</td>
<td>387</td>
<td>99227</td>
<td>639413</td>
<td>2.82</td>
</tr>
<tr>
<td>0.08</td>
<td>12</td>
<td>51</td>
<td>278</td>
<td>34449</td>
<td>214643</td>
<td>2.81</td>
</tr>
<tr>
<td>0.1</td>
<td>11</td>
<td>38</td>
<td>178</td>
<td>11825</td>
<td>71863</td>
<td>2.83</td>
</tr>
<tr>
<td>0.12</td>
<td>10</td>
<td>34</td>
<td>147</td>
<td>5406</td>
<td>31684</td>
<td>2.82</td>
</tr>
<tr>
<td>0.14</td>
<td>9</td>
<td>27</td>
<td>111</td>
<td>1262</td>
<td>6181</td>
<td>2.78</td>
</tr>
<tr>
<td>0.16</td>
<td>7</td>
<td>24</td>
<td>66</td>
<td>101</td>
<td>443</td>
<td>2.65</td>
</tr>
<tr>
<td>0.18</td>
<td>6</td>
<td>17</td>
<td>37</td>
<td>32</td>
<td>193</td>
<td>2.61</td>
</tr>
<tr>
<td>0.2</td>
<td>5</td>
<td>17</td>
<td>32</td>
<td>5</td>
<td>93</td>
<td>2.52</td>
</tr>
<tr>
<td>0.22</td>
<td>5</td>
<td>14</td>
<td>23</td>
<td>1</td>
<td>28</td>
<td>2.35</td>
</tr>
<tr>
<td>0.24</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>0</td>
<td>9</td>
<td>2.13</td>
</tr>
<tr>
<td>0.26</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1.75</td>
</tr>
<tr>
<td>0.28</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Some statistical results of the performed experiments have been depicted in Table 10. The following concepts exist in this table:

- **Minimum Support**: shows the MinSup value of different experiments.
- **Number of Large Entities**: shows the number of large endpoint entities. In fact, this column indicates the number of Large EntityInfo instances that have been defined by the program and also have been used to generate ItemChains. As these numbers show, even for little MinSup values, a few number of EntityInfo instances have been defined.
- **Number of ItemChains**: indicates the number of ItemChains generated by GenerateItemChains algorithm. These ItemChains are employed to generate 2-Large ItemChains.
- **Number of 2-Large ItemChains**: indicates the number of 2-Large ItemChains generated by Generate2LargeItemChains algorithm. These 2-Large ItemChains are employed to generate 3-Large ItemChains.
- **Number of L-Large ItemChains (L ≥ 3)**: indicates the number of L-Large ItemChains (L ≥ 3) generated by the main algorithm. For different values of L, L-Large ItemChains are constructed of (L-1)-Large ItemChains. All Large ItemChains are employed to generate Multi-Relation Association Rules.
- **Number of Rules**: shows the number of rules generated by GenerateRules algorithm from all Large ItemChains. The number of generated rules is several times more than the number of Large ItemChains, which is because of permutation of the ItemChain in the rules.
- **Average number of Relations Count**: shows the average number of relations in the ItemChains of the generated rules. As this column shows, the proposed algorithm is able to extract association rules with several relations.

The results presented in Table 10 indicate that as the value of MinSup decreases, the number of generated L-Large ItemChains increase exponentially. That’s because of Large ItemChain definition. A Candidate L-ItemChain is large when L-ItemChain itself and all of its subsets are large too. When the value of MinSup decreases, L-ItemChains and all of their subsets have more chance to become large and generate L-Large ItemChains and as a result, the number of Larger ItemChains (Large ItemChains with more ChainIDs) increases. By increasing the number of ChainIDs, the number of generated rules would also increase.

For different MinSup values, the number of ItemChains generated by GenerateItemChains algorithm has been depicted in Figure 11. In this figure, ItemChains[K] indicates the number of ItemChains that contain K relations in their LOR part. These numbers show an unexpected result: in many cases, the number of ItemChains[3] is more than ItemChains[1] and they are both more than ItemChains[2].

Figure 12 shows two important ratios between the numbers of generated items by the proposed algorithms as:

- **The Ratio of 2-Large ItemChains Count to ItemChains Count**: shows the number of generated 2-Large ItemChains divided by the number of generated ItemChains. As this curve shows, by decreasing the value of MinSup, the combined ItemChains have more chance to become large and generate a 2-Large ItemChain.
- **The Ratio of Rule Count to Large ItemChains Count**: shows the number of generated Multi-Relation Association Rules divided by the number of generated Large ItemChains. In fact, these values show that for different MinSup values, how many rules is generated from each Large ItemChain. This curve shows that by decreasing the value of MinSup, the number of ChainIDs in the LOC part of Large ItemChains increases and as a result the number of generated Multi-Relation Association Rules also increases. This is because rule generation is based on permuting ChainIDs in the antecedent and consequent parts.

Finally, Figure 13 shows the run time of the experiments in seconds. The experiments were done on a Core i5 M450 2.40GHz Laptop with windows 7. As this figure shows, by decreasing the value of MinSup, the required time to generate ItemChains, Large ItemChains and Multi-Relation Association Rules increases. That’s because, as Table 10 shows, whenever the value of MinSup decreases, the number of Large Entities (EntityInfo), ItemChains and Large ItemChains involved in the computations will also increase.

9 Conclusions & Future Work

In the past years many ARM algorithms have been developed which differ in the structure of their input data, their problem solving methodology, and their goal of ARM or the structure of the generated rules.

In this paper, a new class of association rules namely Multi-Relation Association Rules was proposed. The intuition behind this new kind of rules is to employ direct and indirect relations among entities to generate ARs. Each Multi-Relation Association Rule includes several items in which each item is constructed of one entity and several relations concerned to the entity. In
Figure 11. Number of ItemChains[1..3]

Figure 12. Ratio of 2-Large ItemChains and ItemChains, Large ItemChains and Rules

Figure 13. Run Time
contrast to traditional ARs, these rules show indirect events related to entities which cause the occurrence of special patterns in data.

The name of the main proposed algorithm is MRAR which employs a chain of relations or events to generate ARs with several relations. In addition to considering indirect relations among entities, another facet of the proposed algorithms is its ability in mining Multi-Relation Association Rules from heterogeneous datasets with no exact definition of well-defined transactions. Any dataset convertible to a directed graph with labeled edges (such as semantic web and relational databases), can be employed by the proposed algorithm. In this graph, source vertices indicate entities, destination vertices indicate other entities or attribute values of the source entity (source vertex), and edges indicate relations between two entities or indicate an attribute of an entity.

The obtained results show information about the proposed algorithm behavior from different aspects and prove its ability in mining Multi-Relation Association Rules by considering indirect relations among entities from heterogeneous datasets with no exact definition of well-defined transactions. As the results show, the number of generated patterns is usually high, hence selecting and employing suitable rules for real-world applications may be hard. For future work, as in this work we employed ontologies at instance level, proposing a method for mining and selecting the most interested and useful patterns and ARs by considering the semantics of data provided by ontologies is suggested.

References


[18] Michihiro Kuramochi and George Karypis. Fre-


[37] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery, 8(1):53–87, 2004.

[38] Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In VLDB, volume 95, pages 420–431, 1995.


[40] Jiawei Han and AW Fu. Mining multiple-level association rules in large databases. Knowledge and Data Engineering, IEEE Transactions on, 11(5):798–805, 1999.


Reza Ramezani received his B.Sc. degree in software engineering from Shiraz Faculty of Engineering (Bahonar) in July 2010 and M.Sc. degree from Isfahan University of Technology in August 2012 under supervision of Dr. Mohammad Saraee and Prof. Mohammad Ali Nematbakhsh. After having completed the M.Sc. degree in software engineering, he changed his research interest from semantic web mining to fault-tolerant real-time reconfigurable systems. Since August 2012, he is a Ph.D. student of software engineering in Distributed Dependable Embedded Systems (DDEmS) Laboratory, at Ferdowsi University of Mashhad under supervision of Dr. Yasser Sedaghat.

Mohammad Saraee Mohammad Saraee is a Senior Lecturer in Data Mining and Bioinformatics and the Programme Leader for M.Sc course (Databases and Web Based Systems). He holds a PhD in Computer Science from the University of Manchester. He is a member of Informatics Research Center. He co-authored 4 book chapters, 23 research articles in leading ISI / International refereed journals, and 66 papers in IEEE and International conferences.

Mohammad Ali Nematbakhsh received his B.Sc. degree of Electrical engineering from Louisiana Tech University at 1981 and Doctor of Philosophy (Ph.D.) of Electrical and Computer engineering from University of Arizona at 1987. He is now a professor of Computer Software and teaches computer engineering courses (B.Sc., M.Sc., Ph.D.) at University of Isfahan and researches on the same field.