

**EKONOMICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA HOSPODÁRSKEJ INFORMATIKY**

Evidenčné číslo: 103004/I/2014/1398580811

**DISCOVERY OF ASSOCIATIVE KNOWLEDGE  
IN DATABASES VIA EVOLUTIONARY  
ALGORITHMS**

**Diplomová práca**

**2014**

**Bc. Ľudovít Petržala**

**EKONOMICKÁ UNIVERZITA V BRATISLAVE  
FAKULTA HOSPODÁRSKEJ INFORMATIKY**

**DISCOVERY OF ASSOCIATIVE KNOWLEDGE  
IN DATABASES VIA EVOLUTIONARY  
ALGORITHMS**

**Diplomová práca**

**Študijný program:** Manažérske rozhodovanie a informačné technológie

**Študijný odbor:** 6258 Kvantitatívne metódy v ekonómii

**Školiace pracovisko:** Katedra aplikovanej informatiky

**Vedúci záverečnej práce:** Ing. Kamil Krauspe, PhD.

**Bratislava 2014**

**Bc. Ľudovít Petržala**



## ZADANIE ZÁVEREČNEJ PRÁCE

**Meno a priezvisko študenta:** Bc. Ľudovít Petržala  
**Študijný program:** Manažérske rozhodovanie a informačné technológie  
(Jednoodborové štúdium, inžiniersky II. st., denná forma)  
**Študijný odbor:** 3.3.24 Kvantitatívne metódy v ekonómii  
**Typ záverečnej práce:** Inžinierska záverečná práca  
**Jazyk záverečnej práce:** anglický

**Názov:** Discovery of associative knowledge in databases via evolutionary algorithms

**Anotácia:** The thesis utilizes evolution computation techniques to induce association rules based on example data stored in big datasets. This discipline is commonly known as knowledge discovery in databases. Evolutionary algorithm is a generic population-based metaheuristic optimization algorithm that uses solution space search mechanisms inspired by biologic evolution, such as recombination, mutation and evolutionary selection. The thesis focuses on the problem of solution encoding, evaluation and recombination using various evolutionary operators. The thesis describes selected models, approaches and picks the most useful of them for real implementation. A practical output of the thesis is a fully-functional desktop application that evolves association rule models from a given input dataset.

**Vedúci:** Ing. Kamil Krauspe, PhD.  
**Katedra:** KAI FHI - Katedra aplikovanej informatiky FHI

**Dátum zadania:** 08.10.2012

**Dátum schválenia:** 23.11.2012

doc. Ing. Gabriela Kristová, CSc.  
vedúci katedry

### **Čestné vyhlásenie**

**Čestne vyhlasujem, že záverečnú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.**

**Dátum: 24.04.2014**

.....

Ludovít Petržala

**Acknowledgment:**

Hereby I would like to express my gratitude and say thank you to all people who have helped me with my work, who shared their wisdom and guided me so I was able to come that far.

My special thanks go to my supervisor Ing. Kamil Krauspe, PhD. for sharing his experience and precious recommendations that have significantly affected the overall output and helped to shape my work to fully reach its potential.

Last but not least, I would like to thank my family and friends as well, especially for their patience, constructive feedback and never-ending support.

## **ABSTRAKT**

PETRŽALA, Ľudovít: *Objavovanie asociatívnych poznatkov v databázach cez evolučné algoritmy*. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: Ing. Kamil Krauspe, PhD. – Bratislava: FHI EU, 2014, 75 s.

Diplomová práca popisuje evolučné výpočtové techniky na získavanie asociatívnych pravidiel z dát organizovaných do rozsiahlych datasetov. Zameriava sa predovšetkým na genetické algoritmy, ktoré implementujú nástroje inšpirované evolučnou teóriou, ako napríklad mutáciu, evolučnú selekciu a rekombináciu chromozómov. Hlavným cieľom práce je vyvinúť vlastný genetický algoritmus, čo pozostáva z definovania reprezentácie jedincov, ich vyhodnocovania a návrhu evolučného cyklu vrátane genetických operátorov. Práca je rozdelená do 5 kapitol.

V prvej kapitole sa venujeme problematike asociatívnych pravidiel v rámci data miningu a tiež aj evolučným algoritmom ako takým. Popisujeme základné teoretické princípy, metodológie a techniky na získavanie asociatívnych pravidiel. Taktiež z teoretického hľadiska definujeme problémy ktoré sú s touto úlohou spojené. V ďalšej časti riešime zadanie diplomovej práce, podrobnejšie stanovujeme čiastkové ciele a zameriavame sa na stanovenie podmienok, ktoré zabezpečia integritu práce a dosiahnutie stanovených cieľov. Tretia kapitola je zameraná na popis navrhovaného riešenia, charakteristiku jednotlivých blokov genetického algoritmu a priblíženie ich implementácie z procesného hľadiska. V záverečnej kapitole popisujeme už konkrétne bloky programu, ich návrh, implementáciu a špecifikujeme jednotlivé zložky a ich použitie. V rámci diskusie sme vypracovali prípadovú štúdiu, kde prezentujeme použitie navrhnutého algoritmu nad dátami a uvádzame možné príklady použitia vygenerovaných pravidiel k tvorbe pridanej hodnoty z obchodného hľadiska.

Výsledkom riešenia danej problematiky je desktopová aplikácia, ktorá umožňuje dolovanie asociatívnych pravidiel prostredníctvom vlastného genetického algoritmu a tiež aj analýza navrhovaného riešenia.

## **Kľúčové slová**

Asociatívne pravidlá, data mining, evolučný algoritmus, genetický algoritmus

## **ABSTRACT**

PETRŽALA, Ludovít: *Discovery of associative knowledge in databases via evolutionary algorithms*. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Supervisor: Ing. Kamil Krauspe, PhD. – Bratislava: FHI EU, 2014, 75 p.

The thesis utilizes evolution computation techniques to induce association rules based on example data stored in big datasets. The main focus is especially on genetic algorithms, which represent a generic population-based metaheuristic optimization algorithm that uses solution space search mechanisms inspired by biologic evolution, such as recombination, mutation and evolutionary selection. The main goal is to design and develop own genetic algorithm of mining the association rules, this involves the definition of solution representation, their evaluation and specification of whole evolutionary cycle. The work composes of 5 chapters.

In the first chapter we define overall topic of association rules within the field know as data mining and knowledge discovery as well as evolutionary algorithms as such. We describe basic theoretical principles, methodology and techniques for mining association rules. We also outline common problems and challenges that are related to this topic. In the next chapter we analyze the overall task of this thesis and set partial goals in order to assure the integrity of the work and achieving the main goal. The third chapter is focused on the description of proposed solution, characterizing specific blocks of genetic algorithms and outlining their implementation from procedural point of view. In the last chapter we present actual building-blocks of the algorithm, their design, purpose and implementation. Within the final discussion we have developed own case-study, where we present the usage of proposed algorithm over the data and give the examples of possible usage of generated association rules in terms of creating an added value from business point of view.

The practical output of the thesis is a fully-functional desktop application that evolves association rule models from a given input dataset.

### **Key words:**

Association rules, data mining, evolutionary algorithm, genetic algorithm

<b>TABLE OF CONTENTS</b>	p.
<b>List of figures</b> .....	<b>9</b>
<b>List of tables</b> .....	<b>10</b>
<b>List of algorithms and snippets</b> .....	<b>10</b>
<b>Introduction</b> .....	<b>11</b>
<b>1 The current state of the solved problem in Slovakia and abroad</b> .....	<b>12</b>
1.1 Why do we have to mine the information from data?.....	12
1.2 Science of data mining and its purpose.....	13
1.2.1 Data mining as a multidisciplinary science .....	14
1.2.2 Data mining implementation methodologies.....	16
1.2.3 Modelling techniques of data mining .....	18
1.3 Association rules.....	21
1.3.1 Quality evaluation of association rules.....	23
1.3.2 Basic algorithms for mining association rules.....	25
1.4 Evolutionary algorithms in data mining .....	27
1.4.1 An overview of evolutionary algorithms.....	28
1.4.2 Genetic algorithms in data mining .....	30
1.4.3 Genetic algorithms used for discovery of association rules .....	33
<b>2 Goals of work</b> .....	<b>39</b>
<b>3 Methods of work</b> .....	<b>41</b>
3.1 Methods used to develop the evolutionary algorithm .....	41
3.1.1 Building blocks of evolutionary algorithms .....	42
3.2 Methods for the analysis of proposed algorithm .....	44
<b>4 The work results</b> .....	<b>46</b>
4.1 Loading the data and their transformation.....	46
4.1.1 Transformation of phenotype to genotype .....	49
4.1.2 Representation of chromosomes.....	51
4.2 Fitness function.....	53
4.3 Genetic operators .....	55
4.3.1 Mutation .....	55



4.3.2 Crossover .....	56
4.3.3 Selection .....	58
4.4 Workflow of “DataGen” .....	58
<b>5 Discussion .....</b>	<b>60</b>
<b>Conclusion .....</b>	<b>64</b>
<b>Resumé.....</b>	<b>65</b>
<b>References.....</b>	<b>72</b>
<b>Appendixes .....</b>	<b>75</b>

## List of figures

Figure 1-1: Building blocks of the Knowledge pyramid .....	12
Figure 1-2: Multidisciplinarity of data mining .....	15
Figure 1-3: An overview of the steps that compose the KDD process.....	16
Figure 1-4: CRISP methodology .....	17
Figure 1-5: Common analyses and tools of data mining .....	19
Figure 1-6: Comparison of Pittsburgh and Michigan approaches .....	32
Figure 1-7: Crossover using SSOCF .....	38
Figure 3-1: Cycle of evolutionary algorithm .....	43
Figure 4-1: Main window of developed application.....	46
Figure 4-2: Example of search space.....	47
Figure 4-3: Loading window of developed application.....	48
Figure 4-4: Example of the phenotype .....	49
Figure 4-5: Example of created genotype.....	50
Figure 4-6: Example of crossover.....	57
Figure 4-7: Setting the input parameters for analysis .....	59
Figure 5-1: Frequency distribution of the multiplicity of bought items per transactions ....	61
Figure 5-2: Frequency of bought items in % .....	61

## List of tables

Table 1: Data for classification of heart disease .....	20
Table 2: Sample of transactional records for market basket analysis.....	24
Table 3: Relationship of two initial bits of chromosome on position within the rule. ....	51
Table 4: Example of a chromosome .....	52

## List of algorithms and snippets

Algorithm 1: Pseudocode of the Apriori algorithm .....	26
Algorithm 2: Pseudocode of general genetic algorithm .....	31
Algorithm 3: Pseudocode of select used in ARMGA.....	34
Algorithm 4: Pseudocode of crossover used in ARMGA.....	34
Algorithm 5: Pseudocode of mutation used in ARMGA.....	35
Algorithm 6: Constructor of DataStorage class .....	50
Algorithm 7: Method for creating genotype .....	51
Algorithm 8: Overloaded constructor of Chromosome class .....	53
Algorithm 9: Calculation of confidence .....	54
Algorithm 10: Calculation of comprehensibility .....	54
Algorithm 11: Calculation of J-Measure .....	54
Algorithm 12: Calculation of whole fitness function .....	55
Algorithm 13: Method for performing the mutation .....	56
Algorithm 14: Method for performing the crossover .....	57
Algorithm 15: Method for roulette selection .....	58

## **Introduction**

Nowadays we live in the society that is driven by the information, in business the information might be the most significant competitive advantage, our relationships are affected by the information that we gain from social medias, in big cities the traffic is controlled by the systems that derive the information from weather forecasts, actual traffic densities in specific areas or where the accidents had happened. Mentioned examples are just the tip of the iceberg, information are crucial in healthcare, product customization, manufacturing, any kind of industry, social media, decease prevention, and so on and so forth.

Simply said, the confluence of information technology and communication has produced a society that is starving for the information. But how do we get the information, how can we use them? The majority of the information comes in its raw form as data. Digging of the information from data, discovering patterns and formulating different expectations is the task of the discipline known as data mining.

This thesis describes the specific and very important part of data mining, which are association rules. These rules nowadays serve as a tool for understanding the customer preferences, discovering unique patterns of behavior and provide insight to everyday actions. There are plenty methods that use mathematical and statistical tools to discover such rules, but within this paper we are going to show another, rather emerging approach, which is focused at searching in datasets via genetic algorithms.

Such approach looks very promising because it offers more human-like method, hence we believe that generated rules might be more precise, can accommodate to different needs and might offer higher flexibility, which reflects quickly changing environment. Within the scope of this thesis is also to develop own genetic algorithm to mine association rules from datasets.

After the setting of theoretical background and defining the principles of data mining and genetic algorithms, the own implementation of such algorithm is described. In the end we compare advantages and disadvantages of using genetic algorithms for mining association rules and also showcase its usage on one of the most notorious examples, which is market basket analysis.

# 1 The current state of the solved problem in Slovakia and abroad

## 1.1 Why do we have to mine the information from data?

Nowadays each person and organization – business, family or institution – can generate as well as access a large quantity of data and information about itself and about the environment in which exists. This data has the potential to forecast the evolution of interesting variables or tendencies in the outside environment. According to Science Daily a full 90 percent of all the data in the world has been generated over the last two years (SCIENCEDAILY, 2013). In 2012, every day 2.5 quintillion bytes of data (1 followed by 18 zeroes) are created. As a society, we're producing and recording more data each day than was seen by everyone since the beginning of the earth. But data as itself does not bring any kind of direct added value towards the quality of life, prosperity of businesses or any development. They just represent fertile soil for achieving all mentioned goals or visions. Crucial is the transformation of data to information and later on to knowledge, eventually wisdom.

On the following picture, we can see the relation between data, information and knowledge (FROST, 2014).

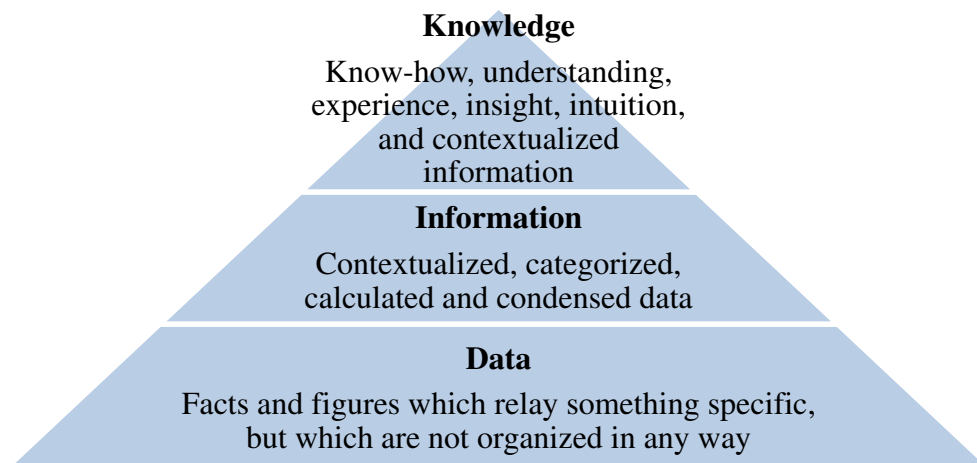


Figure 1-1: Building blocks of the Knowledge pyramid

This pyramid can be also extended by another dimension – wisdom, which is above the knowledge. Wisdom in this understanding can be defined as collective application of knowledge in action and is more experience related. All the relations can be even simplified and we can say that information is equal to understanding relations, knowledge

is about understanding patterns and wisdom is to understand principles. The increasing availability of data and the complicated process of transforming them to relevant outputs led to the need for valid tools for its modelling and analysis. Data mining represent the appropriate tool to extract knowledge from such data.

## **1.2 Science of data mining and its purpose**

Data mining, as a special discipline, is rather young, and as with other youngsters, it is swiftly developing. In the early years of data mining it was just a secondary analysis, focusing solely on large databases which had been used for some other purposes. But nowadays we can find more databases that are collected with the specific aim of subjecting them to a data mining processes. Moreover, we can also perceive formal experimental design being used to decide what data to collect (for example, as with supermarket loyalty cards or bank credit card operations, where different customers receive different cards or coupons).

To understand the expression ‘data mining’ it is useful to look at the literal translation of the whole term. The verb to mine means to extract something; it usually refers to mining operations that extract from the Earth her hidden, precious resources. The connection of this word with data suggests an in-depth search to discover new, precious information which previously were unnoticed within the mass of available data. The basic definition of data mining says that it is an extraction of implicit, previously unknown, and potentially useful information from data (WITTEN et al., 2011). More precise and complex definition of data mining defines it as the process of selection, exploration and modelling of large databases in order to discover models and patterns that are unknown a priori (GIUDICI, 2003).

Data mining is being applied in an increasing variety of areas; hence we can find different mining tools in any field of business, academical and governmental field. Although all mining processes are widely used in scientific applications – for example astrophysics, particle physics and bioinformatics – definitely the major drivers behind its development have been the commercial implementations. This is simply because commercial organizations have recognized the competitive edge added values that proficiency in this area can give mainly in the area of making better-informed and greater decisions. The most familiar business applications contain market basket analysis in the retail and distribution industry (to discover which products are bought at the same time, enabling shelf

arrangements and promotions to be planned properly and effectively), consumer propensity studies (to aim mailshots and telephone calls at customers most likely to respond positively), prediction of attrition (loss of the clients to a competing supplier) in the mobile telephone industry, scoring in financial establishments (to forecast the risk of default by an applicant for credit), automatic fraud detection and the search for the causes of manufacturing flaws. Notorious examples from other fields include analysis of road accidents, decoding of the genome, assistance to medical prognosis, sensory analysis in the food industry, and others.

### *1.2.1 Data mining as a multidisciplinary science*

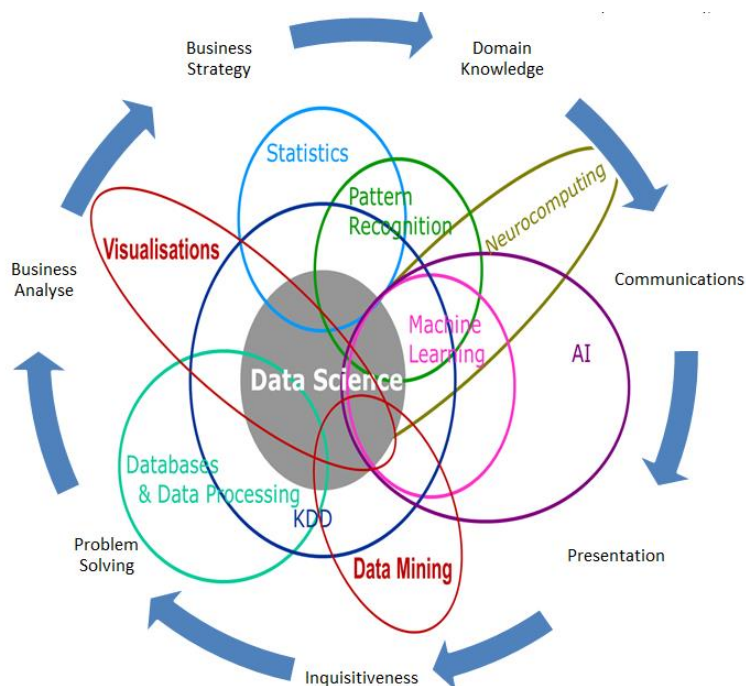
Data mining represents the fusion of various disciplines, most notably statistics and machine learning, because both can be relatively easily applied to the problem of squeezing patterns from large databases. Based on these definitions and the whole scope of data mining we can conclude that statistical apparatus play significant role in transformation of data to knowledge. But they cannot be seen as one and the same tool. Data mining differs from applied statistics mainly in terms of its scope; whereas applied statistics is about the application of statistical methods to the data at hand, data mining is a whole process or flow beginning with data extraction and analysis, finishing with identification of the production of decision rules for specified business goals. In other words, data mining can be understood as a business intelligence process.

Statistics in relation to data mining mainly provides the methods and theoretical concepts. The other substantial element of data mining is information technology, which provides the raw material (data), the computing resources and the communication channels (the output of the results) to other computer applications and to the users. Crucial part is machine learning. The whole aim of any mining tool is to sift through databases automatically, seeking regularities or patterns. Strong patterns, if found, will likely generalize to make accurate predictions on future data. But dealing with real-world data brings some problems. Many of the found patterns will be too simple and uninteresting. Others might be spurious, contingent on accidental coincidences in the particular dataset used. This means that looking for absolute perfection of any discovered pattern or rule is not possible in any way. There will be exceptions to every rule and cases not covered by any rule. Also actual quality of the data might be questionable, often some parts are garbled, and the others are absent. Based on all of these challenges that we have face

during the process of mining leads us to assumption, that used algorithms need to be robust enough to handle the imperfect data and to extract regularities that are inexact but useful.

Machine learning represents the technical backbone of data mining. It is used to extract information from the raw data in databases, more specifically the information that is expressed in a comprehensible form and can be used for a variety of purposes. The process is one of abstraction: taking the data, warts and all, and inferring whatever structure underlies it. Machine learning is interpreted as the acquisition of structural descriptions from instances. The kind of patterns that are recognized can be used for prediction, explanation, and understanding. Forecasting what will happen in new situations is derived from historical data that describe what happened in the past, often by guessing the classification of new cases. Important part is the application of “learning”, that is an actual description of a structure that can be used to classify examples. This structural description provides explanation and understanding as well as prediction. Gained insights by the user are of most interest in the majority of applied data mining tools; indeed, this is one of machine learning’s major benefits over classical statistical modelling.

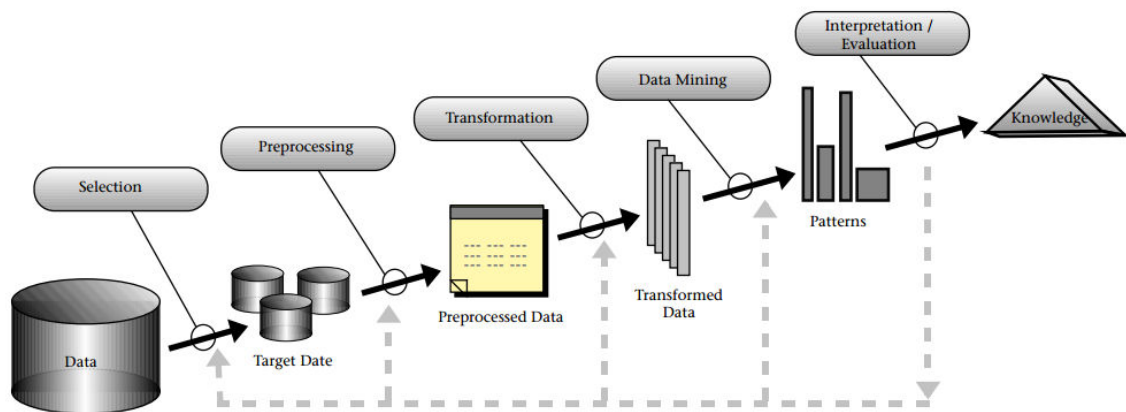
Though statistics and machine learning represent two fundamental elements of data mining, it begins to be obvious that this field is truly multidisciplinary and it combines also many other sciences and fields. In the following scheme by Brendan Tierney from Dublin Institute of Technology is precisely shown true nature of data mining (TIERNEY, 2012).



**Figure 1-2: Multidisciplinary of data mining**



As we can see the data mining exists within much wider area, per se data science and knowledge discovery in databases (KDD). The definition of data mining in relation to KDD is following: “Data mining is a step in KDD process that consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns (or models) over the data” (FAYYAD et al., 1996). The position of data mining within the KDD is quite specific and the most straightforward way is expressed by following scheme:



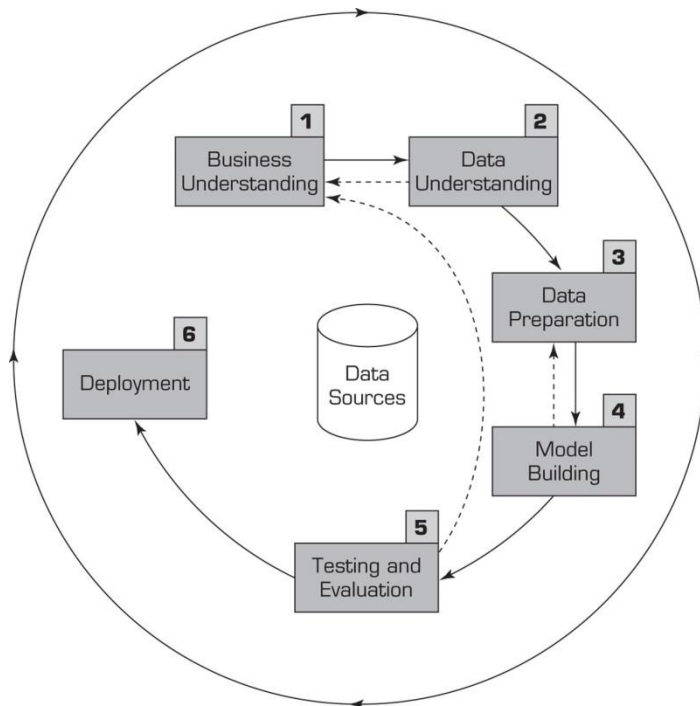
**Figure 1-3: An overview of the steps that compose the KDD process**

The successful implementation of any data mining process within the KDD requires also great skill in areas of neurocomputing, databases and data processing, artificial intelligence and of course in visualization, to represent the outputs in understandable and meaningful ways.

### 1.2.2 Data mining implementation methodologies

Due to the complexity and high demands that are required to achieve the successful implementation of any data mining process, there have been developed many different methodologies. These methodologies or frameworks suggest concrete steps and help to follow the principle of “*divide et impera*”, which suggests to split the complex system into smaller subsystems, that will be easier to understand and coop with. The most famous methodologies include CRISP – DM, SEMMA and 5A.

According to the web platform kdnuggets.com, the most common and the most favorite methodology, that is used by data miners is Cross Industry Standard Process for Data Mining, or also known as CRISP – DM (2007). The specific steps and relations between them are shown in the following picture (CHAPMAN, 2008):



**Figure 1-4: CRISP methodology**

The arrows in the process diagram specify the most significant and common dependencies between stages. The outer circle in the diagram represents the repeating of data mining itself. The discovered patterns from the previous iteration can help to discover new, often more focused ones. Key stages are:

1. Business Understanding - The initial phase, whose aim is to support the understanding of the project objectives and requirements (business rules). Usually the preliminary plan is designed to achieve the overall goals.
2. Data Understanding - This phase begins with an initial data summarization in order to discover first insights, to identify data quality problems and to create initial hypotheses.
3. Data Preparation – The goal of this stage is to construct the final dataset. This task includes all necessary ETL (extraction, transformation and loading) processes, such as attribute selection, cleaning and so on.
4. Model Building – Applying different modelling techniques and setting their parameters to optimal values.

5. Testing and Evaluation – Thorough evaluation of generated models and reviewing executed steps that led to creating of the model. Also the check out with business rules is necessary.
6. Deployment - Gained knowledge needs to be organized and presented in a way that the customer can use it. It can be as simple as generating a report or as complex as implementing a repeatable data mining process.

The first version of this methodology was presented at the 4th CRISP-DM SIG Workshop in Brussels in March 1999. IBM is the main promoter the CRISP-DM process model and it has incorporated it into its SPSS Modeler product, but never the less also many non-IBM data mining practitioners use this framework.

### *1.2.3 Modelling techniques of data mining*

Discovering patterns and transforming data to any kind of knowledge can be done via many modelling tools of data mining. There exists a vast variety of different approaches and every single tool can implement them in slightly modified and customized ways according to the needs of specific industry.

If we will look at these techniques based on their purpose, we can identify two big groups, one is for explaining the past (this group includes mainly descriptive statistical tests) and the second group is for predicting the future (this group includes “typical” mining methods such as clustering, classification and so on). Mining techniques can be also classified according to the approach of discovering the patters. This point of view also splits all techniques to two big groups – direct and indirect data mining. Direct mining acquired the philosophy of “from top to bottom”. In this case we have defined our goal (e.g. some hypothesis) and we are trying to find explanatory variables and attributes that affect (proof or disproof) our assumptions. Direct data mining is mainly used for classification and regression, it is also known as the assumption driven mining. Indirect data mining on the other side looks on the whole process “from bottom to top”. There is no exact goal or hypothesis, which we are trying to proof or disproof. We are basically crawling in the data and trying to find relevant and interesting patterns or rules, so this approach is also called data driven mining. In practice we cannot say which approach is better or worse, both have their limitations as well as advantages. Each method is more suitable for different task and many times all methods just collide and supplement each other. It is common to see for

example the usage of indirect mining over new datasets and after the identification of interesting patterns; they support the creating initial hypothesis for direct mining.

In the following chart, you can see the major techniques and their allegiance according to their purpose and information that their outputs are providing:

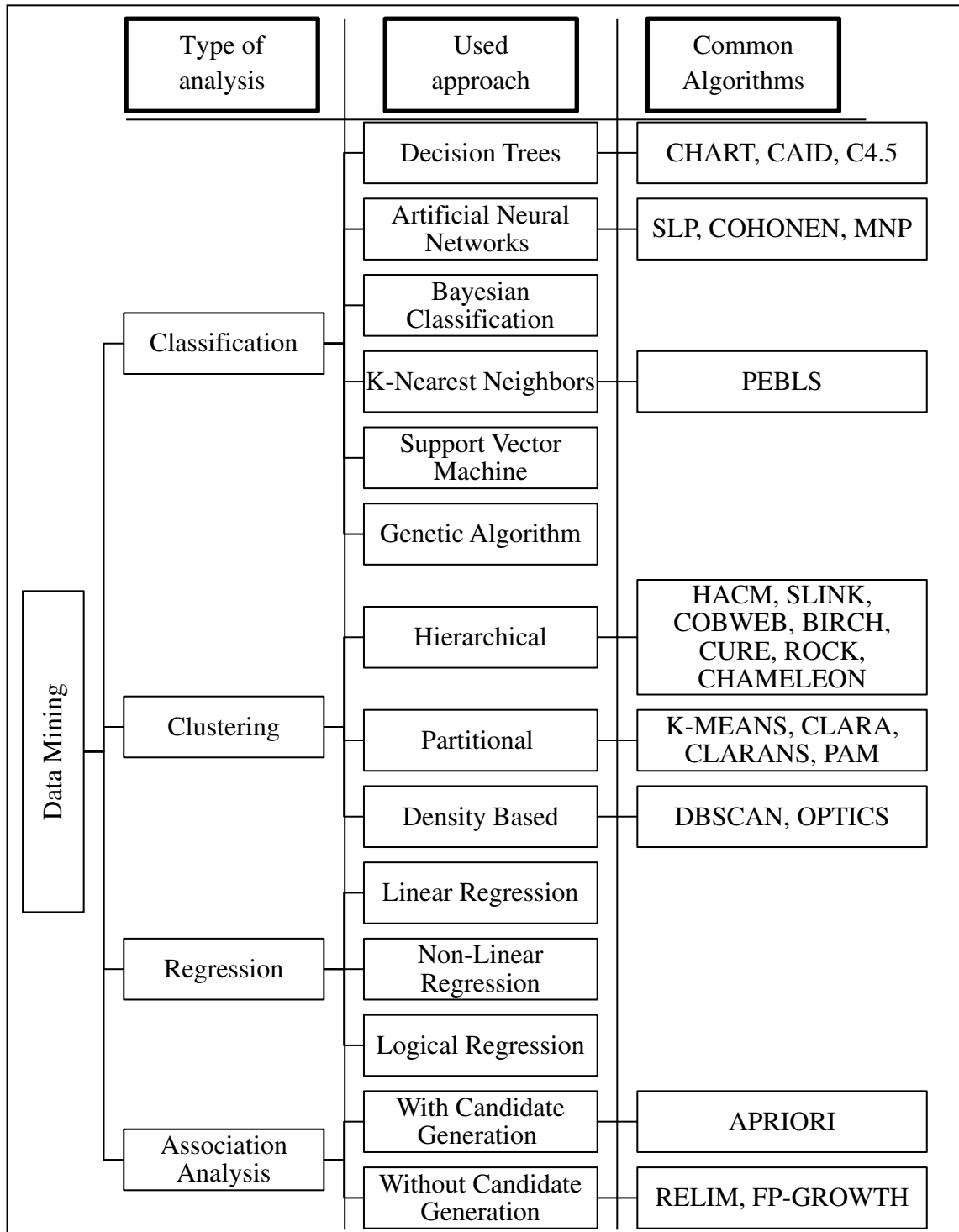


Figure 1-5: Common analyses and tools of data mining

In order to fully understand the usage and capabilities of data mining it is also crucial to become familiar with basic types of analysis, these include mainly classification, clustering, regression and association analysis.

Classification is focusing on predicting a certain outcome based on a given input. General process of classification begins with setting a goal (the explaining variable), which attribute we want to analyze, than follows the processing of a dataset of all attributes. The algorithm tries to discover patterns, more specifically how values of explanatory attributes (all except the one that is our goal) influence the final value of explaining variable. Classical implementations of this analysis are decision trees and neural networks. Typical example for classification can be prediction of the occurrence of heart disease and the dataset can look as following:

**Table 1: Data for classification of heart disease**

<b>Age</b>	<b>Blood pressure</b>	<b>Heart rate</b>	<b>Heart problem</b>
70	145 / 75	75	Yes
35	115 / 78	83	No
30	120 / 80	69	No
71	110 / 70	70	No

Than the explaining variable is occurrence of heart problem (Yes / No) and the rest of attributes will be explanatory variables. When a new patient comes to the doctor his attributes will be recorded and based on their values, it can be predicted, whether he can have some heart problem. The principle of classification is to split values of explanatory attributes into groups that tend to occur together with the specific value of the variable that we are trying to explain.

Regression is trying to find specific equation that explains / forecasts the values of explaining variable. The simplest formula of regression, linear regression, uses a straight line ( $y = ax + b$ ) and determines the suitable values for a and b to predict the value of y based upon a given value of x. Typical example of regression is estimating, whether the client of bank that is asking for credit will be able to pay it back, in this case usually the logical regression is used (with outcome of defaulting or paying it back). Than the explanatory variables can be age, salary, number of children and so on.

Clustering represents typical examples of data driven mining. This means that there is no class to be predicted and the whole aim is to divide instances into “natural” groups. This

means that the goal is to place data elements into related collections without advance information on group descriptions, which means that clustering in a certain way is an unsupervised classification. Popular clustering techniques include k-means clustering and expectation maximization (EM) clustering. Popular example of its usage is as a supporting tool for marketers to discover distinct groups in their customer bases, and then use this knowledge to design targeted marketing programs.

### 1.3 Association rules

The detection of association rules is one of the descriptive methods, which is very popular in data mining. Historically it was developed for the retail industry, where it was used to analyze the products bought by a customer on a single visit. Association is a data mining function that discovers the probability of the co-occurrence of items in a collection. The relationships between co-occurring items are expressed as association rules (ORACLE, 2008). They were originally introduced by prof. Agrawal et al. in paper from 1993 as a tool to analyze customer shopping habits, sales transactions. This explains the alternative name for this method: market basket analysis. Very popular area, where the association rules are used nowadays is web mining, where it is used to analyze the pages visited by a web user (TUFFÉRY, 2011).

Finding association rules is a matter of finding rules of the following type:

*'If, for any one individual, variable  $A = x_A$ , variable  $B = x_B$ , and so on, then, in  $M\%$  of cases, variable  $Z = x_Z$ , and this configuration is found for  $N\%$  of the individuals.'*

In other words, the goal is to find the most frequent combinations of values in a set of variables of a dataset. The first part of the rule is called the “antecedent” or “condition”; the second part is called the “consequent” or “result”; and expressions of the form  $\{A = x_A\}$  are called “items”. This logically means that in an association rule, an item can never be in both the condition and the result concurrently. The value of  $M$  is called confidence and the value of  $N$  is called the support. A rule is therefore an expression of the form:

*If Antecedent, then Consequent.*

Formally we can define the problem of mining the association rules as following: Let  $I = \{a_1, \dots, a_n\}$  be a set of literals, properties or items. A record  $(t_1, \dots, t_n) \in \text{dom}(A_1) \times \dots \times \text{dom}(A_n)$  from our transaction database with schema  $S = (A_1, \dots, A_n)$ ,

can be reformulated as an item set  $T$  by  $a_i \in T \leftrightarrow t_i = 1$ . Hence we call a set  $Z \subseteq I$  an association, if the frequency of occurrences of  $Z$  deviates from our expectation given the frequencies of individual  $X \in Z$  (MAIMON et al., 2010).

This means that we are looking for items in datasets that are frequent (above some minimal level). Now these frequent sets of items need to be translated into rules because preference (or causality) induces a kind of direction in the association. Then

We call  $X \rightarrow Y$  an association rule with antecedent  $X$  and consequent  $Y$ ,  
if  $Z = X \cup Y \subseteq I$  and  $X \cap Y = \{\}$  holds.<sup>1</sup>

Hence it is obvious that association rules are quite similar to classification rules. But in this sense the differences from the classifications represent the added value of associations. So the main differences are:

- Prediction of any attribute, not just the class
- Freedom to predict combinations of attributes
- Rules are not intended to be used together as a set. Different association rules express different regularities that underlie the dataset, and they generally predict different things.

For example during market basket analysis, it might be noted that customers who buy cereal at the grocery store often buy milk at the same time. In fact, association analysis might find that 85% of the checkout sessions that include cereal also include milk. This relationship could be formulated as the following rule:

*Cereal implies milk with 85% confidence.*

Major difficulties linked to searching for association rules arise from the need to process enormous volumes of data and to pick out new and interesting associations from the overwhelming majority of irrelevant or previously known associations.

Association rules are nowadays very popular, it is especially due to their easy interpretation, big variety of possible implementations and ability to relatively accurately describe found patterns with emphasis on trends and forecasting. We can meet with this

---

<sup>1</sup> From the foregoing definition it is clear that it would make sense to additionally require that  $Z$  must be an association. But this is not part of the “classical” problem statement.

analysis in sales promotions, direct marketing, and for discovering business trends. Market-basket analysis can also be used successfully in catalogue design, cross-sell, and for store layout. Another strong domain that is experiencing boom in terms of implementation of association rules is e-commerce. In this field it is used to customize the user experience through “tailored” customization, for example Web page personalization. An association model might find that a user who visits pages A and B is 70% likely to also visit page C in the same session. Based on this rule, a dynamic link could be created for users who are likely to be interested in page C.

### *1.3.1 Quality evaluation of association rules*

The biggest challenge is not to generate rules; the problem is fact that there can be derived vast amount of rules from even a small dataset. So the aim is to identify those that apply to a reasonably large number of instances (support) and have a reasonably high accuracy on the instances to which they apply (confidence).

The support or also called as the coverage describes the probability of instances in which the whole rule is present, ergo correctly predicted:

$$\text{Support} = \text{Prob}(\text{Antecedent and Consequent})$$

The confidence or also called as the accuracy is the probability of instances that it predicts correctly, expressed as a proportion of all instances to which it applies.

$$\text{Confidence} = \text{Prob}(\text{Antecedent and Consequent}) / \text{Prob}(\text{Antecedent})$$

Consequently, the aim is to find association rules for which the support and confidence are above specified minimum thresholds. Identifying the correct values of these parameters is essential for whole analysis. As is show in following example:

Let’s have small “snippet” of simple transactional record from shop. It contains 5 transactions (specific buying of concrete goods) with following items: {Beer, Bread, Cereals, Cheese, Chips, Wine}. Hence each row corresponds to market basket  $T_x$ . Now let’s say that there is found new rule: If customer buys Bread, he also buys Wine (Bread  $\rightarrow$  Wine). The confidence of this rule is  $3/4$  and support is  $3/5$ . Similarly, the confidence of the association Cheese  $\rightarrow$  Bread is  $2/3$  and support is  $2/5$ . One thing is evident: Bread is present in almost all the transactions, or more precisely the a priori probability of having Bread there is 0.8.



Table 2: Sample of transactional records for market basket analysis

Transaction ID	Set of transactions				
T <sub>1</sub>	Beer	Bread	Cheese	Cereals	Wine
T <sub>2</sub>	Bread	Cheese	Wine	Chips	
T <sub>3</sub>	Bread	Wine			
T <sub>4</sub>	Beer	Bread	Cereals		
T <sub>5</sub>	Cheese	Cereals			

This probability is greater than the confidence index for Cheese  $\rightarrow$  Bread and therefore this rule is not helpful for predicting Bread. If we say that a transaction taken at random contains Bread, there is only one chance in five that we will be wrong, as against one chance in three if we follow the rule Cheese  $\rightarrow$  Bread.

The improvement brought by a rule, by comparison with a random response, is called the lift (or simply the ‘improvement’), and is as follows (GIUDICI, 2003):

$$lift(rule) = \frac{Confidence(rule)}{Prob(Result)} = \frac{Prob(Antecedant \text{ and } Consequent)}{Prob(Condition) * Prob(Result)}$$

When the ‘result’ is independent of the ‘condition’, the lift is clearly equal to 1. If the lift is less than 1, the rule does not help. Thus we find that  $lift(Cheese \rightarrow Bread) = \frac{5}{6}$  (useless rule) and  $lift(Bread \rightarrow Wine) = \frac{5}{4}$  (useful rule).

In more complex rules the items can be nested by logical operators such as AND, OR and even NOT. Such rules are then more difficult for understanding, developing business implementations and require higher computational resources. As a benefit they usually achieve significantly better predicting efficiency and bring higher added value.

Negation has interesting characteristics connected to calculating the quality assessment indexes and can many times invert bad rule to the good and precise one. Note that, if the lift of the rule  $Condition \rightarrow Result$  is less than 1, then the lift of the inverse rule, i.e. the rule:

$Condition \rightarrow NOT Result$ ; is greater than 1, since:

- $confidence(inverse\ rule) = 1 - confidence(rule)$
- $Prob(NOT\ Result) = 1 - Prob(Result)$ .

### *1.3.2 Basic algorithms for mining association rules*

There are dozens of algorithms used to mine frequent item sets. Some of them, very well known, started a whole new era in data mining. They made the concept of mining frequent item sets and association rules possible. Others are variations that bring improvements mainly in terms of processing time. We'll go through some of the most important algorithms first briefly in this article, and then in more detail in the subsequent articles. The algorithms vary mainly in how the candidate item sets are generated and how the supports for the candidate item sets are counted.

#### **Apriori Algorithm**

It is by far the most important data mining algorithms for mining frequent item sets and associations. It opened new doors and created new modalities to mine the data. It was introduced by Rakesh Agrawal and Ramakrishnan Srikant in 1994 in their paper named Fast algorithms for mining association rules. Since its inception, many scholars have improved and optimized the Apriori algorithm and have presented new Apriori-like algorithms. The authors became living legends in the data mining communities. They both received masters and PhDs from University of Wisconsin, Madison and both worked for IBM. The IBM's Intelligent Miner was created mainly by them. Once colleagues, they now work for competing companies – Agrawal for Microsoft and Srikant for Google. Apriori uses a breadth-first search strategy to count the support of item sets and uses a candidate generation function which exploits the downward closure property of support (DATA MINING ARTICLES, 2013).

Process of mining AR with Apriori consists of 2 steps:

- It starts by searching for the subsets of items having a probability of appearance (support) above a certain threshold.
- Then it attempts to break down each subset in a form {Antecedent U Consequent} such that the confidence is above a certain threshold.

Expressed in pseudocode Apriori is following:

```

C1 := { {i} | i ∈ I };
k := 1;
while Ck ≠ ∅ do begin
    countSupport (Ck, D);
    Fk := { S ∈ Ck | supp(S) > minsupp };
    Ck+1 := candidateGeneration(Fk);
    k := k+1;
end
F = F1 ∪ F2 ∪ ... ∪ Fk-1; // all frequent itemsets

```

**Algorithm 1: Pseudocode of the Apriori algorithm**

## FP-growth Algorithm

In the year 2000 authors Jiawei Han, Jian Pei, Yiyen Yin introduced new and unique algorithm in their paper Mining Frequent Patterns without Candidate Generation. Its aim was to overcome Apriori's disadvantages. The 2 main limitations are the possible necessity of producing a huge number of candidates if the amount of frequent 1-itemsets is high or if the size of the frequent pattern is big the database has to be scanned repeatedly to match the candidates and determine the support. But FP-growth algorithm brings solution for mining frequent patterns without candidate generation. It performs a depth-first search through all candidate sets and also recursively generates the so called *i*-conditional database  $D^i$ , but instead of counting the support of a candidate set using the intersection based approach, it uses a more advanced technique, which is Frequent-pattern Tree (FPtree). The main idea is to store all transactions in the database in a tree based structure. In this way, instead of storing the cover of every frequent item, the transactions themselves are stored and each item has a linked list linking all transactions in which it occurs together. By using the tree structure, a prefix that is shared by several transactions is stored only once.

Due to divide-and-conquer strategy and a frequent-pattern tree that stores a compressed version of the data, only two passes are required to map the data into an FP-tree. The tree is then processed recursively to “grow” large item sets directly. This avoids generating and testing candidate item sets against the entire database.

Building a frequent pattern tree:

1. First pass over the data – count the number times individual items occur.
2. Second pass over the data – before inserting each instance into the FP-tree, sort its items in descending order of their frequency of occurrence, as found in step 1

- a. Individual items that do not meet the minimum support are not inserted into the tree.
- b. Hopefully many instances will share items that occur frequently individually, resulting in a high degree of compression close to the root of the tree.

The main advantage of this technique is that it can exploit the so-called single prefix path case. That is, when it seems that all transactions in the currently observed conditional database share the same prefix, the prefix can be removed, and all subsets of that prefix can afterwards be added to all frequent sets that can still be found (HAN et al., 2004), resulting in significant performance improvements.

#### **1.4 Evolutionary algorithms in data mining**

Especially due to the growth of computing power the evolutionary algorithms are becoming more and more popular in the area of data mining. The paradigm of Evolutionary Algorithms (EAs) consists of stochastic search algorithms inspired by the process of neo-Darwinian evolution (BACK et al., 2000), (DE JONG, 2002), (EIBEN et al., 2003). Main philosophy of EAs is to generate candidate solution based on a population of individuals, and then through the process of “evolution” improve these solutions until they are the best of best.

This method represents very generic search paradigm. Hence EAs can be used to solve variety of different problems, by precisely specifying what kind of candidate solution an individual represents and how the quality of that solution is evaluated (by a “fitness” function). EAs are strong, adaptive search methods that execute a total search in the space of candidate solutions, which is in essence the reason for implementing EAs to data mining. In contrast, several more conventional data mining methods perform a local, greedy search in the space of candidate solutions. As a result of their global search, EAs tend to cope better with attribute interactions than greedy data mining methods (MAIMON et al., 2010). Hence, naturally EAs can discover interesting knowledge that would be missed by a greedy method.

### *1.4.1 An overview of evolutionary algorithms*

All EAs in general are inspired by the principle of natural selection and natural genetics. Basic idea behind this approach is quite simple. All beings or individuals in any kind of environment are continuously evolving, being more adaptive, so their capability to survive is better and better. In the relation to EA, the “individuals” represent the candidate solution to the problem that needs to be solved and so the problem or task could be considered as an “environment”. The term individual (or candidate solution) is also frequently changed for the expression chromosome<sup>2</sup>. The capability to survive is expressed in the quality of the solution, which is represented by candidate solution, which means that it needs to be evaluated somehow. This evaluation of each individual is done via so called fitness function. Than the individuals with higher values of fitness function have higher probability of being selected for reproduction. The selected individuals undergo operations inspired by natural genetics, such as:

- crossover (where part of the genetic material of two individuals are exchanged)
- mutation (where part of the generic material of an individual is replaced by randomly-generated genetic material)

Afterwards new offspring will replace the parents, creating a new generation of individuals. The whole cycle is repeated until a stopping criterion is satisfied.

Evolutionary algorithms hence implement two basic concepts:

1. Charles Darwin’s theory of the survival of fittest – individuals with lower value of fitness function have lower probability of reproduction; hence their children will not appear in new generation.
2. Gregor Mendel’s law of inheritance – the aim of genetic operators (mutation and crossover) is to generate children with higher fitness function compared to their parents and keep the diversity of whole population, which will help to overcome local extremes in search space and prevent the convergence toward single individual solution.

---

<sup>2</sup> This relates especially to genetic algorithms, where also the operator exchange the genetic information in order to create new individuals, chromosomes.

Based on the different approaches towards the specific parts of whole cycle (representation of individuals, creating new offspring, setting the satisfactory criterion and so on and so forth) there are several kinds of EAs, such as Genetic Algorithms, Genetic Programming, Classifier Systems, Evolution Strategies, Evolutionary Programming, Estimation of Distribution Algorithms, etc. (MAIMON et al., 2010). Within the scope of this work the main emphasis is upon the Genetic Algorithms (GAs). The reasons and consequences will be explained in later chapters.

Although genetic paradigm plays important role in EAs, we cannot forget that they are still just a member of rather larger family of search algorithms. This also puts initial consequences. There are two basic concepts that almost every search algorithm tries to define:

- principle of preference of specific direction and space of searching
- principle of diversity, that allows to change the direction and space of searching (MACH, 2009)

In EAs and especially GAs these two principles play the most significant role and during every single step of designing and implementing of GAs we need to keep this in mind. Reason is quite simple, these two principles are contradictory. When we will focus on one of them too much, the other one will be suppressed and whole algorithm will not be efficient. This means that successful design of genetic algorithm is understood as setting all moving parts in order to find ideal equilibrium of two principles.

GAs obviously represent very strong tool for solving complex, non-linear problems and their usage has been researched and studied for many different fields, the most significant are:

- Function optimizers – difficult, discontinuous, multi-modal, noisy functions
- Combinatorial optimization – layout of VLSI circuits, factory scheduling, traveling salesman problem
- Design and Control – bridge structures, neural networks, communication networks design; control of chemical plants, pipelines
- Machine learning – classification rules, economic modelling, scheduling

Afterwards GAs have proved themselves and have been successfully implemented in portfolio design, optimized trading models, direct marketing models, sequencing of TV advertisements, adaptive agents, data mining, etc.

### *1.4.2 Genetic algorithms in data mining*

Due to general search approach, high level of abstraction and natural flexibility of GAs they can be considered as an ideal solution for data driven mining where also association rules belong. There are two reasons that allow us to accept this presumption:

- GA has the ability to discover interesting, not so obvious knowledge, or so called nuggets, that would be missed by classical greedy methods.
- GA can easier adapt in the real world of imperfect data (missing values, different data types and redundancy in datasets).

The first reason is derived from the whole philosophy of genetic algorithms, because they are generating information, which is strictly based on data. There is no need for any “in-process” interaction with user (except of setting initial conditions that are derived from business rules) or creation of any kind of hypothesis. The second reason is based on whole aspect of probability and randomization that is used to generate new population during iterations. The best explanation is based on Christie-Davies’ Theorem: “If your facts are wrong but your logic is perfect, then your conclusions are inevitably false. Therefore, by making mistakes in your logic, you have at least a random chance of coming to a correct conclusion.” (DAVIES, 2011) This means that imperfection of the data is natural and will never be totally avoided. And so my trying to follow any exact steps of reasoning based on the data will never lead us to correct solutions. This can be avoided by simply inserting some imperfection, keeping a reasonable chance that even weaker individuals will have some chance to become parents for new generation of candidate solutions. But still it needs to be true that an individual with higher value of fitness function will also have higher probability of becoming a parent than the one with lower value of fitness function. So certain imperfection is allowed, even welcomed and it will assure that we have found global maximum of fitness value instead of local one.

On the highest level of abstraction the pseudocode for GA is following:

1. Create initial population of individuals
2. Compute the fitness for each individual
3. repeat
  - a. Select individuals based on fitness
  - b. Use genetic operators to selected individuals, creating new individuals
  - c. Compute fitness of each of the new individuals
  - d. Update the current population (new individuals replace old individuals)
4. until(stopping criteria)

**Algorithm 2: Pseudocode of general genetic algorithm**

Main differences of GAs in comparison to other algorithms<sup>3</sup> from the family of EAs are in:

- Expected output or solution representation – usually a candidate solution consists mainly of values of variables – in essence, data.
- Essential nature of the result that they embody - each individual represents a result to one specific instance of the challenge that is being solved.

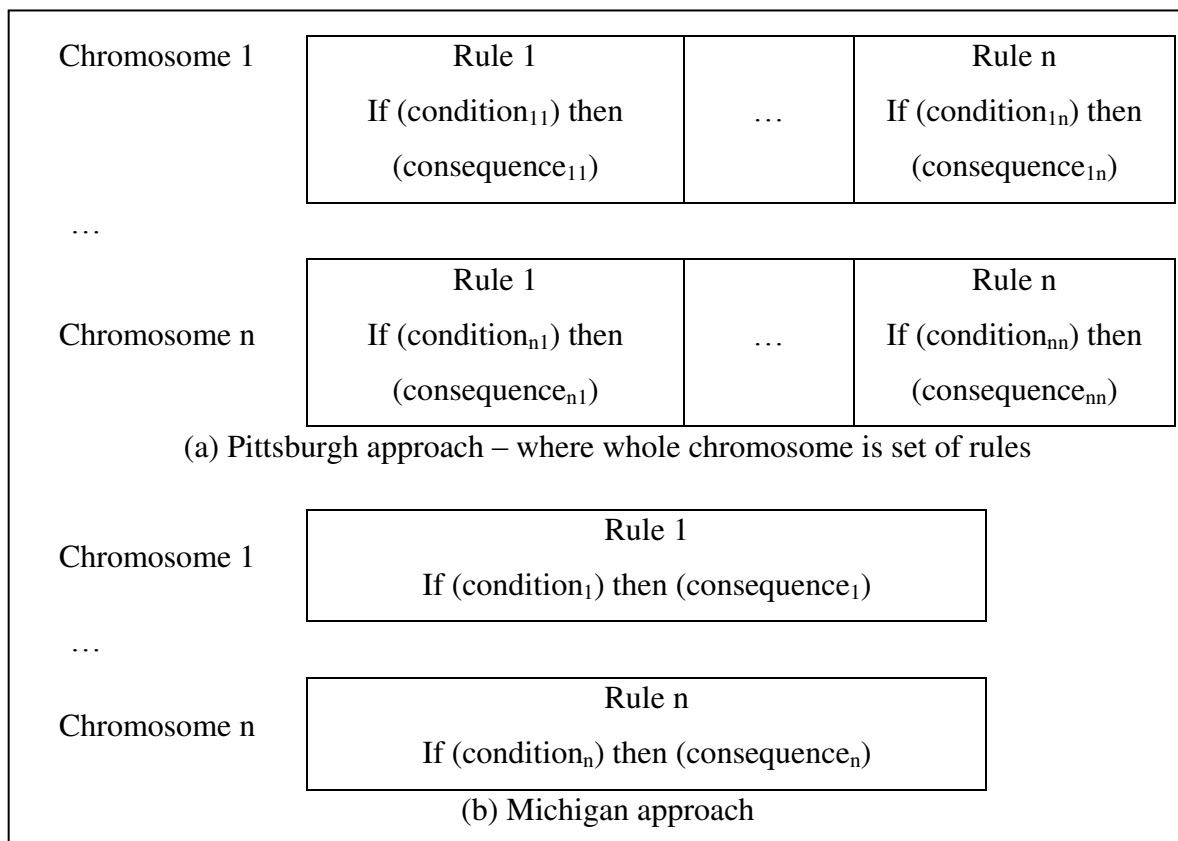
The first specificity causes that the representation of GA individuals tend to be simpler. In the most of the cases GA individuals are usually represented by a fixed-length linear genome. The second one is rather focusing on implementation per se. It means that GAs need to be somehow customized for each specific dataset, but not in the terms of its design but rather their initial parameters (probability of cross-over and mutation, fitness function, selection method and so on and so forth). Anyway the setting of these parameters can be partially automated and ideal configuration can be recommended based on the experience, previous outputs and needs of user. All of this means that GAs are aiming for efficiency (lower required amount of passes through the database) and independency from datasets (standard methods require certain familiarity of dataset to be able to establish minimal thresholds for rules evaluation, usually minimal support and confidence).

Any GA for rule discovery can be in general associated with one of two basic but approaches, the Michigan approach and the Pittsburgh approaches. The biggest distinguishing feature between the two is that in the Michigan approach (also referred to as Learning Classifier Systems) an individual is a single rule, whereas in the Pittsburgh approach each individual represents an entire set of rules. This means that Michigan approach denotes any approach where each GA individual encodes a single prediction rule (WAKABI-WAISWA et al., 2008). Following picture shows mentioned difference:

---

<sup>3</sup> Especially compared to genetic programming which is often misunderstood as a synonym of genetic algorithms.





**Figure 1-6: Comparison of Pittsburgh and Michigan approaches**

Although from implementation point of view both approaches are completely different, it cannot be said that one is better than the other one. The choice which approach will be used strongly depends on the kind of task and what rules we need to find, ergo which kind of data mining task is being addressed. Naturally if we are trying to do some GA for clustering or classification, we need to see all the rules as a whole and asses them as such, which means that Pittsburgh approach is more natural choice, because it allows the interaction among the rules. On the other hand, the Michigan approach might be more natural in other kinds of data mining tasks. Typical situation is when we are trying to find rather smaller set of rules that will have high quality. This means that we are looking for rules that will be evaluated independently of the others.

With the Pittsburgh approach it is usually linked significantly higher need for computation power. Individuals tend to be syntactically longer; hence the calculation of fitness function is more complex and computationally expensive. In addition, standard genetic operators need to be modified because they have to cope with the length and complexity of individuals in reasonable ways.

On the other hand, in the Michigan approach the individuals definitely shorter and simpler. This leads to the reduction of required time for calculating the value of fitness function as well as to possibility to use standardized genetic operators in their generic design. However, this advantage comes with a cost. First of all, since the fitness function expresses the evaluation of rules separately it is more difficult to see the bigger picture - i.e. taking rule interactions into account. Another problem is the convergence of population toward one chromosome, so the diversity of whole population might be suppressed. One of the possible solutions is to running the GA several times, each time discovering a different rule. The drawback of this approach is that it tends to be computationally expensive. The second possibility is to develop some kind of niching method.

### 1.4.3 Genetic algorithms used for discovery of association rules

Due to natural simplicity, high explanatory potential and relatively straightforward outputs GAs are considered as one of the best alternatives in searching for undetermined solutions. But it is still rare to see that genetic algorithm is used to mine association rules. Yet recent years show a big change in this area and the focus of many researchers and practitioners is shifting towards GAs in association mining instead of classification. Hence it has been developed vast variety of such algorithms. In this section we will discuss a few showcases, which will also help to demonstrate what needs to be designed in order to create GA that can be used for discovery of association rules.

#### **ARMGA algorithm**

Basic difference of ARMGA compared to any classical association mining algorithm (e.g. Apriori) is that in ARMGA model, the confidence  $conf(X \rightarrow Y)$  should be larger than, or equal to, support  $supp(Y)$ , because it deals with positive association rules of the form  $X \rightarrow Y$ . And based on the definition of the rule interest  $RI = P(A, B) - P(A)P(B)$ , for  $A \rightarrow B$  (YAN et al., 2005). Hence the positive confidence can be defined as following:

$$pconf(X \rightarrow Y) = \frac{supp(X \cup Y) - supp(X)supp(Y)}{supp(X)(1 - supp(Y))}$$

So the whole task of genetic algorithm will be to find such rules that maximize the function of positive confidence. ARMGA model follows the Michigan strategy. Chromosomes are defined as indexes of all itemsets and one extra bit that defines where condition ends and

consequence begins. It implements three basic operators: select, cross-over, mutation. Select has following pseudocode:

```

Boolean select(c, ps)
begin
    if (frand() * fit(c) < ps then
        return TRUE;
    else
        return FALSE;
    end
end

```

**Algorithm 3: Pseudocode of select used in ARMGA**

Function  $select(c, ps)$  returns  $TRUE$  if chromosome  $c$  is successfully selected with probability  $ps$ , and otherwise  $FALSE$  if failed. In this function,  $frand()$  returns a random real number from interval 0 to 1. Which means that the aspect of randomization brings the possibility that even not so good rule can become a parent for next generation, which will help to keep the diversity in population. Yet, the rules with better fitness function still have higher probability of being selected.

The second operator is cross-over. Its pseudocode is presented in the next snippet. Function  $crossover(pop, pc)$  uses a two-point strategy to create new population from old one ( $pop$ ). Two points that set which segment of chromosomes will be exchanged are randomly generated.

```

population crossover(pop;pc)
begin
    pop_temp ← ∅
    for ∀ c1 = (A10, ..., A1k) ∈ pop do
        begin
            for ∀ c2 = (A20, ..., A2k) ∈ pop ∧ c1 ≠ c2 do
                begin
                    if (frand() < pc) then
                        begin
                            i ← irand(k + 1);
                            j ← irand(k + 1);
                            (i, j) ← (min(i, j), max(i, j));
                            c3 ← (A10, ..., A1,i-1, A2i, ..., A2,j, A1,j+1, ..., A1k);
                            c4 ← (A20, ..., A2,i-1, A1i, ..., A1,j, A2,j+1, ..., A2k);
                            pop_temp ← pop_temp ∪ c3, c4;
                        end
                    end
                end
            end
        end
    return pop_temp;
end

```

**Algorithm 4: Pseudocode of crossover used in ARMGA**

The last operator is for mutation. It has following pseudocode:

```

chromosome mutate(c, pm)
begin
  if (frand() * fit(c) < pm) then
  begin
    c.A0 ← irand(k - 2) + 1;
    i ← irand(k - 1) + 1;
    c.A1 ← irand(n - 1) ;
  end
  return c;
end

```

**Algorithm 5: Pseudocode of mutation used in ARMGA**

Function  $mutate(c, pm)$  considers the fitness of  $c$  as an additional weight in order to makes an occasional change (with probability of  $pm$ ) to genes of chromosome  $c$ . ARMGA uses the mutation to create also the initial population. It is given a seed chromosome and with  $pm=1$ , the whole population  $pop[0]$  is generated. The size of the population is given by the user. Algorithms stops reproduction cycles if and only if at least one of two conditions is fulfilled:

1. The difference between the best and the worst chromosome is less than a given value.
2. The number of iterations is larger than a given maximum number of repetitions.

The ARMGA algorithm is quite efficient and it is proven that it can find relevant rules. But there is still some place for improvement. The authors of this algorithm conclude that very different populations can be obtained each time (YAN et al., 2005), even when there is high number of generations. One of the probable reasons is weak fitness function, because it allows generating many high-quality rules. Hence more strict fitness function might bring better results. One of the possible solutions is to incorporate minimal support threshold. In more recent paper authors provide improved version of ARMGA, which is working with FP-tree, provides improved and more balanced mutation operator and implements more complex fitness function.

### **Genetic algorithm proposed by Wilson Soto and Amparo Olaya–Benavides**

Wilson Soto and Amparo Olaya–Benavides in their paper called A Genetic Algorithm for Discovery of Association Rules proposed slightly different approach for mining association rules via GA. The suggested algorithm is implementing the Java library `lambdaj`. It focuses on manipulation with collections in a pseudo-functional and statically typed way. The main purpose of `lambdaj` is to partially eliminate the burden to

write (often nested and poorly readable) loops while iterating over collections. In particular it allows iterating collections in order to:

- filter its items on a given condition
- convert each item with a given rule
- extract a given property from each item
- sort the items on the values of one of their property
- group or index the items on the value of one or more properties
- invoke a method on each item
- sum (or more generally aggregate) the items or the values of one of their property
- concatenate the string representation of the items or of the values of one of their property

without to write a single explicit loop (GOOGLE, 2010). Hence the overall aim is to increase the readability of the code as well as decrease the number of required passes through the collections of data, which will lead to higher efficiency during whole computation.

The evaluation of chromosomes via fitness function is different compared to ARMGA algorithm. The fitness function in this case is the aggregation of multiple standardized formulas allowing users to define final its form (SOTO et al., 2011):

$$F = (w_1 \times P \times R) + (w_2 \times K) + (w_3 \times I)$$

$w_1, w_2$  and  $w_3$  are user-defined weights for all used metrics.  $P$  is confidence and  $R$  is support. A value  $K$  represents so called comprehensibility. This number expresses is a proportional inversely value, relative to the number of conditions  $N(X)$  in the rule's antecedent  $X$ , while a rule can have at least  $M$  conditions. This means that comprehensibility is defined as  $K = 1 - \frac{N(X)}{M}$ .  $I$  stands for the interestingness of the rule, which is derived from information gain ( $G$ ), that is difference between entropies, namely,

the entropy of whole database ( $H(X)$ ) and expected entropy ( $H(X, A)$ ). This means that whole process of calculating Interestingness is following:

1. entropy H of the database:  $H(X) = -\sum_{i=1}^n p(x_i) \log p(x_i)$

$n$  - The number of different values in the dataset  $X$ ;

$p_i$  - The values frequency  $i$  in the dataset  $X$

2. expected entropy of attribute A:  $H(X, A) = H(X) - \sum_{j=1}^m p(a_j)H(X_{a_j})$

$m$  - Attribute's different values number  $A$  in  $X$ ;

$a_j$ - $j^{\text{th}}$  possible value of  $A$

$H(X_{a_j})$  - subset of  $X$  which contains all the items where the value of  $A$  is  $a_j$

3. The information gain  $G$  from the attribute A:  $G(H(X, A)) = H(X) - H(X, A)$

4. Hence the Interestingness:  $I = \frac{1}{\sum_{j=1}^k G(X, A_j)}$

Fitness function expressed as this allows users to define the kind of rules that he is looking for. If user is looking for rules that are rather simple, easier to understand, the comprehensibility will have higher weight than the rest. Or if user is looking for rules, that are the most helpful and can explain the dataset the best, then the weight of interestingness will be higher. Hence this approach allows user to customize the search for rule more flexibly and enables him to define what kind of rules are important to him, without defining minimal thresholds.

The individual representation is based again on the Pittsburgh approach. In terms of selection, the strategy called tournament ( $\tau$ ) is used. This strategy consists of choosing individuals randomly with uniform distribution from the current population to execute many tournaments. The tournament is the event, where two individuals face each other, and the winner is the one with better fitness function and this individual is selected.

The proposed algorithm applies a specific type of crossover over the chosen attributes. It is called Subset Size-Oriented Common feature Crossover Operator (SSOCF) the advantages of this type of crossover are (SOTO et al., 2011):

- Conservation of the useful information sets

- Better exploration of non-dominated set solutions
- To produce children with the same parents distribution

The common attributes are preserved by the children and the non-common attributes are inherited by  $i^{th}$  father with probability  $\frac{n_i - n_c}{n_u}$ , where  $n_i$  is the number of chosen attributes from the parents,  $n_c$  is the number of common attributes among them and  $n_u$  is the number of non-shared chosen attributes.

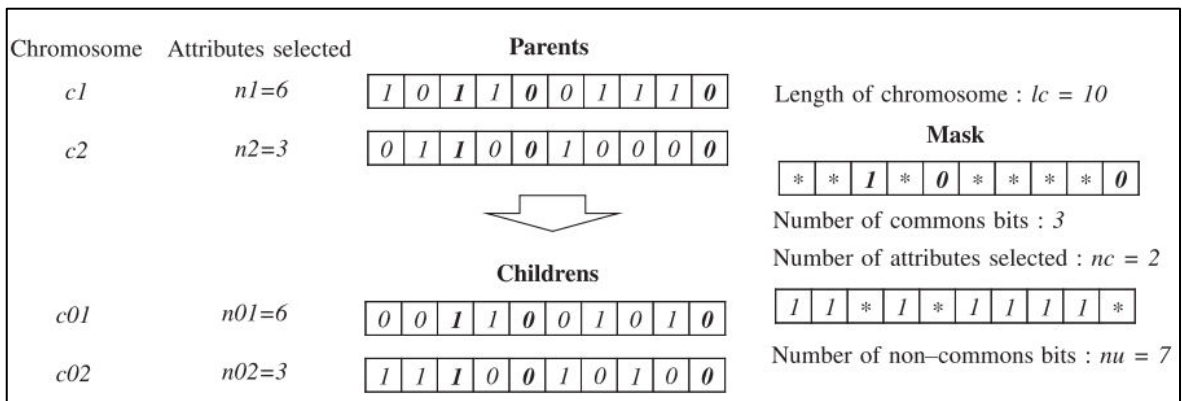


Figure 1-7: Crossover using SSOFC

The mutation operator in this algorithm chooses a number  $n$  of genes for changing. Selected bits are chosen randomly and changed by a non-symmetric probability. To change from 1 to 0 the probability equals 0.4 and to change from 0 to 1 the probability is equal to 1.

As we can see the process of designing a genetic algorithm is really complex and it involves different stages. In following chapters the whole process will be described and implemented step by step.

## 2 Goals of work

The overall goal is to design and develop an evolutionary algorithm, which will be used for mining the association rules in datasets. As it was shown in previous chapters, the association rules embody an important part of data mining tools especially in data driven process of transformation of the data to information and knowledge. Due to their structure and the way they are expressed it makes them easy to understand and implement in various business fields. One of the main limitations of discovering such rules is the fact that it is necessary to deeply understand the dataset from which the rules are discovered. This means that the focus during the mining of association rules in initial phase is mainly on data, their origin, how they were gathered, what they express and descriptive statistics with the emphasis on identification of the minimal thresholds for input parameters, such as minimal support and confidence. Although this approach works we believe that it can be improved. This might be achieved by focusing on business goals and needs, define the characteristics of desired rules and what they should express. This should be manageable with usage of optimization algorithms instead of the greedy ones. Searching the data space and looking for different rules, while maximizing their quality will bring even bigger independence from data as such and will lead to higher abstraction from their imperfection, hence we will be able to focus on business constraints and needs.

To achieve the main goal it is necessary to set partial goals, which will help and support the overall aim in terms of whole execution, while the integrity of content and correct execution are assured. Among the partial goals belongs the selection of appropriate type of evolutionary algorithm. Selected type of evolutionary algorithm will in the end affect the whole structure of the rules, their quality, usability in solving real-life problems and the overall efficiency of mining.

The designing of any evolutionary algorithm is quite complex process, which means that it also requires the identification of the best framework for developing such algorithms. Following standardized and by practice proved steps in scheming the evolutionary algorithms will ensure, that all important aspects will be carefully covered, all requirements on quality, efficiency and targeted behavior will be fulfilled.

Next partial goal involves actual development of the algorithm that will discover association rules. This will include the programming of all the core parts of mining algorithm, the graphical user interface for easy usage and manipulation with the program



and analytical section of the application which will allow us to monitor the overall progress and development of rules during the mining. Hence the output of this section will be an application that will allow us to mine association rules without specifying the minimal thresholds of parameters, application that can be used also by users with minimal programming skills and it will allow analyzing the whole process of rule mining and evaluating its efficiency as well as prediction capabilities of generated rules.

Finally there will be the showcase of the usage of application, which will allow us to analyze the whole process of mining the rules and evaluate the overall efficiency, quality and precision of found patterns. This showcase will simulate real-life process of mining the association rules, which means that we will set some specific business problems, try to analyze them with the developed application, evaluate the whole process of mining and in the end create the recommended solutions for set business challenges.

Before we will describe the methods and tools that will be used to meet all of these goals we feel that it is important to emphasize that the goal of this thesis is not to develop a software product with complete documentation that could be used for commercial purposes and be fully capable to compete with the complex systems that are nowadays used for data mining. But it can be considered as a prototype that will have certain assets, which might be improved and extended in the future. Hence in the last chapter we will also suggest possible improvements and which direction the whole development might take.

### **3 Methods of work**

The whole work consists of two major segments, which will require two different methods as well. Firstly we describe the methods that are used in order to propose new EA that will mine association rules. This will involve the identification of ideal representation; specification of the framework that will be used is programming and actual development of the application. The second part will focus on methods for documentation and evaluation of algorithms. In this section we will describe how proposed algorithm behaves, its efficiency and the quality of identified patterns.

#### **3.1 Methods used to develop the evolutionary algorithm**

Actual EAs can be expressed in many different forms that represent more or less similar algorithms. It is mainly due to many common principles on which every type of EA is based as well as due to implementations of the same algorithmic parts. In this sense the implementation of EA can be reduced on using two basic principles:

- the principle of preferences, that defines the direction of browsing the search space,
- the principle of diversity, that allows to alter the direction of browsing.

It is obvious that these principles are contradictory. And designing any kind of EA is always about searching for the right balance between them. Before developing any kind of EA there have to be answer the fundamental decisions that involve:

- the searching space, which is necessary to browse in order to find rules (the representation and means of calculation define the searching space and its relation to the searching space),
- implementation of specific blocks of algorithm (selecting procedures for realization of algorithm's building blocks define the source for creation of selection preferences and for browsing in search space),
- input parameters of algorithm (values of selected parameters affect overall efficiency of algorithm on defined search space).

### 3.1.1 Building blocks of evolutionary algorithms

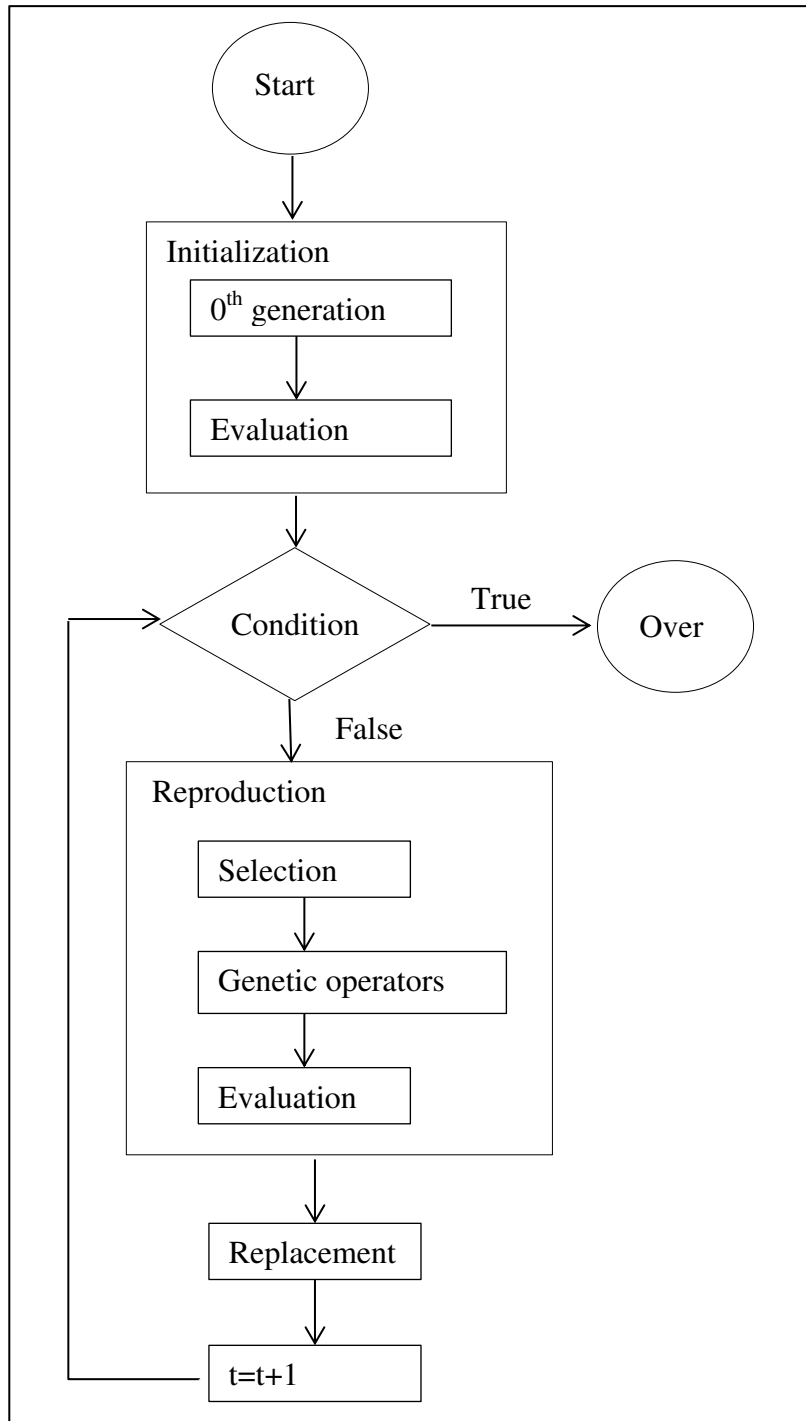
After the setting these initial assumptions the actual programming of evolutionary algorithm can begin. For successful development of our algorithm we have decided to follow the framework of component system for evolutionary algorithms according to Mach (2009). Proposed system covers all stages of any EA (MACH, 2009), as it is expressed on figure 3-1.

In the beginning of the algorithm the initial population of individuals is generated ( $0^{\text{th}}$  generation) and with the flowing time it evolves within every evolutionary cycle. Time  $t$  is measured in discrete units, generations. Each generation represents one transition of evolutionary cycle. It means that in each transition the attention is on the same generation of individuals, which is denoted as population in time  $t$  or population<sup>4</sup> of  $t$ -th generation. Initial individuals create the initial population  $P(t = 0)$ . If  $a_i(t = 0)$  will denote  $i$ -th individual in  $0^{\text{th}}$  generation and the number of required individuals will be  $\mu(t = 0)$ , than it is necessary to generate the set of individuals:  $P(0) = \{a_1(0), a_2(0), \dots, a_{\mu(0)}(0)\}$ . Under the generation of individual we understand the creating of structure for the individual and specification of the values for all its attributes. This means that every individual can become the final solution.

After the initialization of the  $0^{\text{th}}$  generation each individual is evaluated via fitness function. The output of evaluation is a set (distribution) of fitness values  $\{\Phi(a_1(0)), \Phi(a_2(0)), \dots, \Phi(a_{\mu(0)}(0))\}$ . Than this population enters the evolutionary cycle, which begins with the Condition block. It decides whether the whole process will continue at least one more time or it stops. The moment when the condition is satisfied, means that there is no need or sense to continue because solution has been achieved. There are many different kinds of possible conditions but the most common are: setting the maximal number of iterations, setting the minimal thresholds for fitness function, rules begin to converge into one or cumulative fitness function for last few generations did not change significantly.

---

<sup>4</sup> Individual population can, but does not have to be disjoint sets. Some individual can be a part of the population in multiple generations, they might follow close to each other or be separated by other population where the individual is not present.



**Figure 3-1: Cycle of evolutionary algorithm**

When the criterion is not met the population enters the phase or reproduction. It begins with Selection block. Here are identified the individuals who actually enter the process of reproduction (the Genetic operators block), hence they become parents:  $P'(t) = \{b_1(t), b_2(t), \dots, b_{\delta(t)}(t)\}$ .  $\delta$  stands for the amount of selected parents in generation  $t$ , while  $b_j(t) = a_i(t)$ ,  $i \in \langle 1, \mu(t) \rangle$  for all  $j$ . This means that the individual from population  $P'(t)$  can become a parent once, more times or not at all.

The following block is called Genetic operators. This is the place where new individuals are generated from parents  $P'(t)$ . For this purposes one or more operators can be used. The output of this steps are new children:  $P''(t) = \{a_{\mu(t)+1}(t), a_{\mu(t)+2}(t), \dots, a_{\mu(t)+\lambda(t)}(t)\}$ .  $\lambda(t)$  expresses the amount of children in t-th pass through evolutionary cycle, which had the population with  $\mu(t)$  individuals.

Final step of the reproduction process is again the Evaluation of new individuals. Usually it is identical with the block of evaluation in generating the 0<sup>th</sup> population.

The following block in evolution cycle is Replacement. In this step the new population is created. The input for this block is  $\mu(t)$  individuals of current population  $P(t)$  and  $\lambda(t)$  individuals that represent new children,  $P''(t)$ <sup>5</sup>. The output of this block will be  $\mu(t + 1)$  individuals that form new population  $P(t + 1)$ . All the individuals that did not manage to be selected to new generation will extinct. It means that they are forgotten.

The final block of the cycle simply increments the time, which was during all steps of the iteration constant. The newly formed population  $P(t + 1)$  become the initial population  $P(t)$  for next iteration, unless the condition is met. It is also useful to mention that although during the every iteration the parameters of the algorithm can change, it is not very common to see such implementations of any EA. It means that:

- $\mu(t + 1) = \mu(t) = \mu,$
- $\delta(t + 1) = \delta(t) = \delta,$
- $\lambda(t + 1) = \lambda(t) = \lambda.$

### 3.2 Methods for the analysis of proposed algorithm

The most straightforward way how to evaluate the quality of our application will be the simulation of solution for standardized business problem. In our case we will focus on one of the most notorious cases, where association rules are used. It is known as market basket analysis. To successfully evaluate proposed algorithm we will focus purely on quality of generated rules and the process of improvement of each population.

The obvious way how to evaluate the rules will be via fitness function. In proposed algorithm we have developed the fitness function that combines multiple measures,

---

<sup>5</sup> These two sets does not have to be disjoint, parent can be transformed on the children that is the same.

because as it was shown in previous chapters the confidence and support are not sufficient. Whole development of fitness function is mainly the combination of the approaches suggested in (BLANCHARD et al., 2005) and (HAND et al., 2001). The application will record the development of overall fitness function as well as individuals measures. It means that we will be able to track the improvement of each evolutionary cycle. Hence this section will allow us to evaluate the quality of EA as itself.

## 4 The work results

The overall result is an application “DataGen”, which is built in C# in Microsoft Visual Studio Ultimate 2012 on .NET Framework 4.5. Application is composed from three basic parts. The first one is focused on loading data and their transformation to representation that is suitable for discovery of association rules via genetic algorithm. The second component is actual implementation of mining algorithm, whose outputs are association rules. The GAs represent ideal family of algorithms for mining the association rules<sup>6</sup>, hence this approach was used for this purpose. The final component is analytical and it utilizes metadata that were gathered during mining and uses them for visualization of overall process and evaluation of predicting precision of discovered patterns.

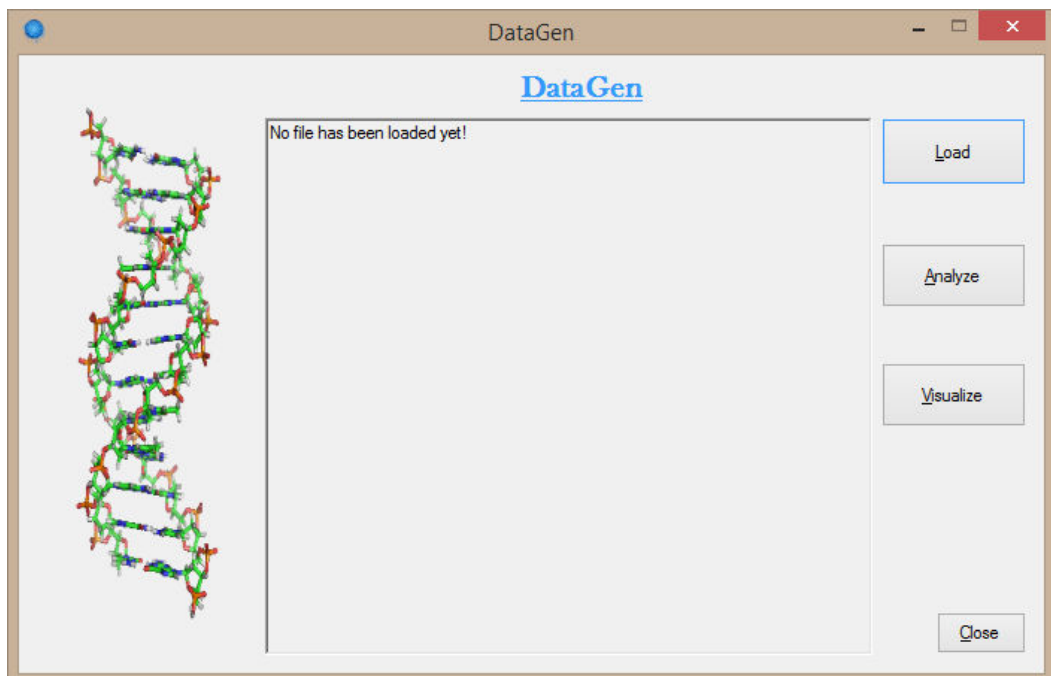


Figure 4-1: Main window of developed application

### 4.1 Loading the data and their transformation

The initial stage of analysis is loading the data and their transformation to representation that is suitable for mining the association rules via genetic algorithms.

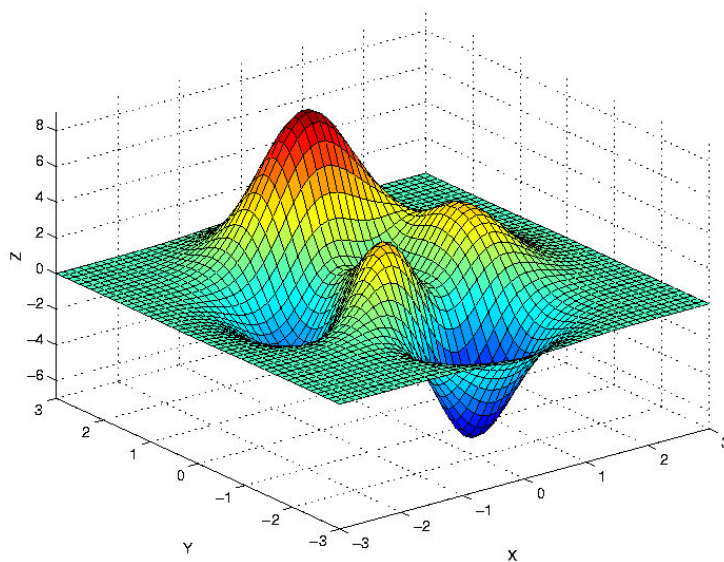
Any kinds of search algorithms, where the GAs belong as well, in general have just a few requirements on solved problems, which allow their application in various tasks. Basically to solve any kind of task with GA, it is just necessary that the solution have two properties. It has to be decomposable and ratable. The decomposition of the solution on

---

<sup>6</sup> Based on the reasons specified in chapter 1.4.2 Genetic algorithms in data mining

smaller parts expresses the natural possibility to join simpler elements to create final solution as well as some kind of purposeful transformation. This solution than can be expressed as a set of attributes and their values. The attributes can be of different data types, for example binary, nominal, real and so on. But because the candidate solutions can have different quality, including absolutely unfitting ones, there is a need for the existence of some measure. The measure will evaluate each candidate and so it will allow us to compare each candidate with others, reaching the final decision, which candidate is the best solution.

When both conditions are fulfilled, the candidate solutions form space. More specifically it is  $(n + 1)$ -dimensional space;  $n$  axes represent  $n$  attributes used for description of the candidate solution (each attribute represents one axis, specific values of given attribute are mapped on corresponding axis) and  $(n + 1)$ th axis expresses the fitness function. The following picture (UNIVERSITY OF UTAH, 2013) shows visualization of candidate solution space, which is built of two attributes (axis  $x$  and  $y$ ) and values of fitness function is mapped to  $z$  axis. Hence the best solution lies in the red area of the space, where fitness reaches its maximum.



**Figure 4-2: Example of search space**

The individuals must be represented in suitable way so the algorithm could work with them efficiently. For this purpose the “DataGen” works only with nominal or qualitative data and not quantitative as such. This means that ideal inputs for our algorithm are attributes that are not continuous (e.g. whether bread was or was not bought or income



that is low, middle, high). The allowed input formats are text files (.txt) or coma-separated values (.csv).

It allows writing the path of an input file directly or via browsing the files on computer. The next is loading button that transforms the data to appropriate representation (with the specification of the separator, which can be any visible character). After the loading, the attributes can be listed with their values. “DataGen” also allows exporting the attributes and whole genotype. The last option is to update data for analysis.

It is becoming more and more obvious that the algorithm does not work directly with the data that are loaded to the system but there has to be done some kind of transformation. Hence in GAs there exist two types of spaces. One is called phenotype and the other one is genotype. Simply said, the phenotype is represented by the data that describe the business side of the problem that is supposed to be analyzed. In our case it will be any .txt or .csv file. On this file the necessary transformations are performed, so the algorithm can work efficiently and can abstract from business related problem and deal clearly with computation. The output of this transformation is the second space, which is called the genotype.

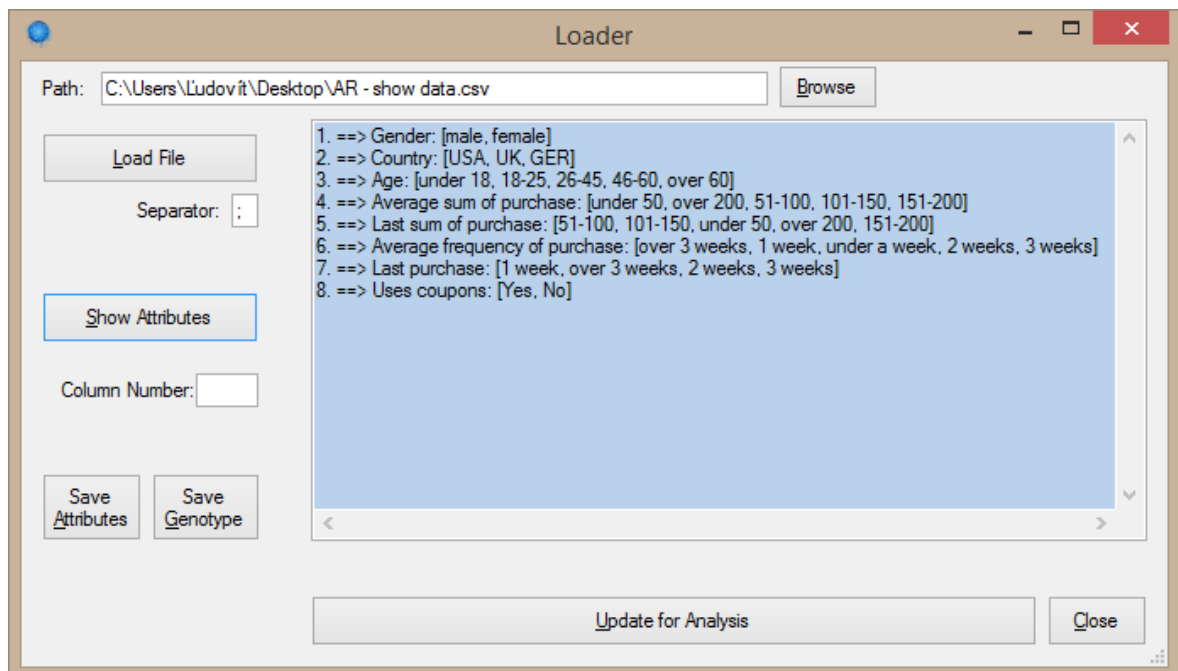


Figure 4-3: Loading window of developed application

### 4.1.1 Transformation of phenotype to genotype

During the selection of representation for the individuals we have decided to implement the binary representation. The phenotype that is built in most cases by strings (e.g.: gender {male, female}, country of origin {USA, UK, GER}) is transformed by “attribute decoding table” to genotype, which is composed of integer.

As the first step we acquire the distinct values for each attribute (column), which for example for the attribute gender is male and female. These attributes are stored and indexed so they create the attribute decoding table. Based on this table the genotype is generated. The values in genotype are integers, which reference to the index of the specific value of each attribute from the table. Indexing is performed on “First come, first serve” basis, hence the data are not sorted; they are processed as they come. This resulted in passing the phenotype only once because then the new value of the attribute is found, the table is extended by this value and new index is assigned to it as well and then this index is used in genotype.

Gender	Country	Age	Average sum of purchase
male	USA	under 18	under 50
female	USA	18-25	over 200
female	UK	26-45	51-100
male	GER	46-60	51-100
male	GER	over 60	101-150
male	UK	26-45	51-100
female	USA	46-60	101-150
male	USA	over 60	under 50
female	UK	18-25	over 200
female	UK	18-25	151-200
male	GER	26-45	151-200

Figure 4-4: Example of the phenotype

This means that the first line of genotype always begins with zeroes. The second line, if it does not contain any of the previous values, has all instances equal to one and so on and so forth. The final genotype of previous figure is in the figure 4-5. The most straightforward example is with the attribute Gender = {male, female} = {0, 1}. On the first line, the values of attribute were empty, hence the indexing began with value male, and it will always be represented by 0. Then the second line comes. There is no female in Gender attribute so it will be added and index of 1 will be assigned to it. In this sense all of the other indexes are generated.

Gender	Country	Age	Average sum of purchase
0	0	0	0
1	0	1	1
1	1	2	2
0	2	3	2
0	2	4	3
0	1	2	2
1	0	3	3
0	0	4	0
1	1	1	1
1	1	1	4
0	2	2	4

Figure 4-5: Example of created genotype

From the implementation point of view we use for genotype own class called `DataStorage`. It contains string array, that stores the names of attributes and so they represent the header of phenotype as well as genotype. This array together with another variable, array of string lists (stores actual values of attributes), form the decoding “table”. The actual genotype is saved in `DataTable`. The constructor of `DataStorage` requires one input parameter that is the array of strings. They form the header of genotype (attribute names of columns). This string is loaded from the first line of input file.

```

public DataStorage(string[] names)
{
    numbAttributes = names.Length;
    data = new DataTable("Genotype");
    attributesHeader = names;
    attributesValues = new List<string>[numbAttributes];

    for (int i = 0; i < numbAttributes; i++)
    {
        attributesValues[i] = new List<string>();
        data.Columns.Add(names[i], typeof(int));
    }
}

```

Algorithm 6: Constructor of `DataStorage` class

Class also implements methods for exporting the genotype, attributes individually as well as all of them. But the most important method is actual transformation of phenotype to genotype. It is following:

```

public void generateGenotype(string[] line)
{
    DataRow row = data.NewRow();
    for (int i = 0; i < numbAttributes; i++)
    {
        if (!attributesValues[i].Contains(line[i]))
            attributesValues[i].Add(line[i]);
        row[i] = attributesValues[i].IndexOf(line[i]);
    }
    data.Rows.Add(row);
}

```

Algorithm 7: Method for creating genotype

### 4.1.2 Representation of chromosomes

The representation of individuals is built upon two-dimensional binary array. For these purposes own class has been designed, its name is **Chromosome**. The first dimension defines the attribute (e.g. gender). The second dimension is focused on actual values of attribute. Hence the binary values (true and false) express whether the specific value of an attribute is or is not present in rule. First two bits describe the position of the attribute within the rule; whether it is part of the condition or consequence. Bits that are setting the position of attribute within the rule cause that any chromosome in given attribute will have its length equal to (count of values) + 2.

Following table explains the position of an attribute within the rule based on two initial bits of second dimension:

Table 3: Relationship of two initial bits of chromosome on position within the rule.

Combination of the first two bits in send dimension	Position of the attribute
True True	Condition
False True	Not present in the rule
True False	Not present in the rule
False False	Consequence

Lets have a following rule:

$$\begin{aligned}
 & \text{gender in (male) and country in (USA, GER)} \\
 & \quad \rightarrow \\
 & \text{average sum of purchase in (under 50, 101-150).}
 \end{aligned}$$

Than based on the data of phenotype, which are in the figure 4-4, the chromosome would look like:

**Table 4: Example of a chromosome**

true, true, true, false
true, true, true, false, true
false, true, true, false, false, true, false
false, false, true, false, false, true, false

Each line of the chromosome represents one attribute. Let's focus on the first line, which describes the gender. The first two values are true, true, it means that this attribute is in conditional part of the rule and the value which is active is the first one that is in phenotype, which is in our case male. Similarly all attributes can be decoded.

To achieve proper generation of the rules some logical requirements had to be ensured. Otherwise the rules could have no meaning. In given representation the two situations could occur:

- condition or consequence might be empty
- attribute which appears in the rule can have all values active or non-active.

These problems have to be faced during the creation of initial population as well as during the realization of genetic operation, namely mutation and crossover. Hence, two methods with the class `Chromosome` were implemented. The generation of the initial population is purely on random basis and it is performed via overloaded constructor of the class. The code is following:

```

public Chromosome(List<string>[] attributes, DataTable data, double compre, double
confi, double jMeasure)
{
    body = new bool[attributes.Count()];

    //sets the position of the attribute within a rule
    bool[][] position = new bool[attributes.Count()];

    for (int i = 0; i < attributes.Count(); i++)
    {
        position[i] = new bool[2];
        for (int j = 0; j < 2; j++)
        {
            if (random.Next(2) == 0) position[i][j] = false;
            else position[i][j] = true;
        }
    }

    this.RepairAttributesPosition(ref position);

    //sets which values of attribute will be in rule
    for (int i = 0; i < attributes.Count(); i++)
    {
        body[i] = new bool[attributes[i].Count + 2];
        body[i][0] = position[i][0];
        body[i][1] = position[i][1];
        for (int j = 2; j < attributes[i].Count + 2; j++)
        {
            if (random.Next(2) == 0) body[i][j] = false;
            else body[i][j] = true;
        }
    }

    body = this.RepairAttributesValues(body);

    fitness = this.CalculateFitness(data, compre, confi, jMeasure);
}

```

Algorithm 8: Overloaded constructor of Chromosome class

## 4.2 Fitness function

As we can see even in the snippet of the code for creating the initial population, the fitness function is also calculated for each chromosome. In our case the fitness function is composed of as weighted average of three measures:

1. Confidence – describes the accuracy of a rule
2. Comprehensibility – describes length and “understandability” of a rule
3. j-Measure – describes an added value of a rule in terms of decreasing information entropy

These measures are aggregated via weighted arithmetic average, while the weights are specified by user. This means user can define his priorities based on business needs, what

is more important for him. User can be looking for rules, that are very accurate but somehow obvious and too long or vice versa. Following snippets show, how each metric is calculated.

```

/*
 * calculating Confidence,
 * occurrence of fulfilling antecedant and consequent / occurrence of
 * fulfilling only antecedant
 */
public double CalculateConfidence(int counterX, int counterXY)
{
    if (counterX == 0) confidence = 0;
    else confidence = ((double)counterXY / counterX);

    if (counterX == 0) return 0;
    else return ((double)counterXY / counterX);
}

```

**Algorithm 9: Calculation of confidence**

```

/*
 * calculation of Comprehensibility
 * log (1 + |consequent|) + log (1 + |whole rule|)
 * | expression | - describes amount of attributes
 */
public double CalculateComprehensibility(List<List<int>>[] rule)
{
    comprehensibility = (Math.Log10(1 + ((double)rule[1].Count)) +
        Math.Log10(1 + ((double)rule[0].Count) + (double)rule[1].Count));

    return comprehensibility;
}

```

**Algorithm 10: Calculation of comprehensibility**

j-Measure:

```

/*
 * calculation of j-Measure
 *  $p(y|x) * \log_2(p(y|x) / p(y)) + (1 - p(y|x)) * \log_2((1 - p(y|x)) / (1 - p(y)))$ 
 */
public double CalculateJMeasure(int counterY, int counterX, int
    counterXY, int numberOfInstances)
{
    double pXY;
    if (counterX == 0) pXY = 0;
    else pXY = (double)counterXY / counterX;
    double pY = (double)counterY / numberOfInstances;

    JMeasure = pXY * Math.Log(pXY / pY, 2) + (1 - pXY) * Math.Log((1 -
        pXY) / (1 - pY), 2);
    if (Double.IsNaN(JMeasure)) JMeasure = 0;
    return JMeasure;
}

```

**Algorithm 11: Calculation of J-Measure**

Then the overall value of fitness function is calculated as following:

```

public double CalculateFitness(DataTable data, double wCompre, double wConfi, double
wJMeasure)
{
    List<List<int>>[] rule = this.GetRule();
    int counterX = 0; int counterXY = 0; int counterY = 0;
    foreach (DataRow row in data.Rows){
        bool evaluatorX = true;
        bool evaluatorY = true;
        /*Counts appearance of antecedant*/
        for (int j = 0; j < rule[0].Count; j++){
            if !(rule[0][j].Skip(1).Contains(row.Field<int>(rule[0][j][0]))){
                evaluatorX = false;
                break;
            }
        }
        /*Counts appearance of consequent*/
        for (int j = 0; j < rule[1].Count; j++){
            if !(rule[1][j].Skip(1).Contains(row.Field<int>(rule[1][j][0]))){
                evaluatorY = false;
                break;
            }
        }
        if (evaluatorX & evaluatorY){
            counterXY++;
            counterX++;
            counterY++;
        }
        else if (evaluatorX) counterX++;
        else if (evaluatorY) counterY++;
    }
    double compre = this.CalculateComprehensibility(rule);
    double confi = this.CalculateConfidence(counterX, counterXY);
    double jMeasure = this.CalculateJMeasure(counterY, counterX, counterXY,
data.Rows.Count);
    return (wCompre * compre + wConfi * confi + wJMeasure * jMeasure) /
(wCompre + wConfi + wJMeasure);
}

```

Algorithm 12: Calculation of whole fitness function

## 4.3 Genetic operators

Mutation and crossover are methods, which are included within the Chromosome class. After their application during the process of generating new individuals, the correction methods are necessary, otherwise the illogical rules might occur.

### 4.3.1 Mutation

Mutation is executed in two steps, firstly there are randomly selected which attributes change their position within the rule. The second step is modification of active and non-active values of selected attribute. The code for mutation is following:



```

public bool[][] DoMutation(DataTable data, double wCompre, double wConfi, double
    wJMeasure)
{
    bool[][] child = copyArr(body);
    for (int i = 0; i < child.GetLength(0); i++)
    {
        /*decides wheather the position of the attribute within rule will be
            changed*/
        switch (random.Next(4))
        {
            case 0:
                child[i][0] = !child[i][0];
                break;
            case 1:
                child[i][1] = !child[i][1];
                break;
            case 2:
                child[i][0] = !child[i][0];
                child[i][1] = !child[i][1];
                break;
        }

        this.RepairAttributesPosition(ref child);

        /*decides which parameter of the attribute will be changed*/
        for (int j = 2; j < child[i].Length; j++)
            if (random.Next(2) == 1) child[i][j] = !child[i][j];

        this.RepairAttributesValues(child);
    }
}

```

**Algorithm 13: Method for performing the mutation**

The probability of initial mutation is set by the user in the beginning of analysis. But because after the each iteration the overall quality of rules is growing, we need to lower this probability of mutation. Hence it is also considered the quality of the rule. The higher value of fitness function causes bigger decrease of probability for the occurrence of mutation. This will allow to cover bigger area of search space and eventually to avoid getting stuck on local maxima. The supplement of probability for mutation is the probability for crossover.

### 4.3.2 Crossover

The “DataGen” uses mainly two-point crossover. This means that there are randomly set two positions for two randomly selected chromosomes, these points define which genetic information will be exchanged. Due to randomization it can happen that these points collide to one and then simply one-point crossover is performed.

The code to perform this operation is following:



### 4.3.3 Selection

To select the chromosomes that will become parents and the genetic operators will be used on them is done via so called roulette selection. This method uses the selection pressure and favors the chromosomes with higher value of the fitness function. In our case the sum of all values of the fitness function represents the wheel of roulette, separated values of the fitness function are the slots of this wheel and random number from the range of 0 to the wheel maximum defines the slot where roulette ball stops. The code that implements roulette selection is following:

```
private int RouletteSelection(double sumFitness)
{
    double roulette = random.NextDouble() * sumFitness;
    double evaluator = 0;
    List<int> indexes = new List<int>();

    for (int i = 0; i < populationSize; i++)
    {
        indexes.Add(i);
    }

    int selector, index;
    do
    {
        selector = random2.Next(indexes.Count);
        index = indexes[selector];
        evaluator += population[index].fitness;
        indexes.RemoveAt(selector);
    } while ((evaluator < roulette) && (indexes.Count != 0));

    return index;
}
```

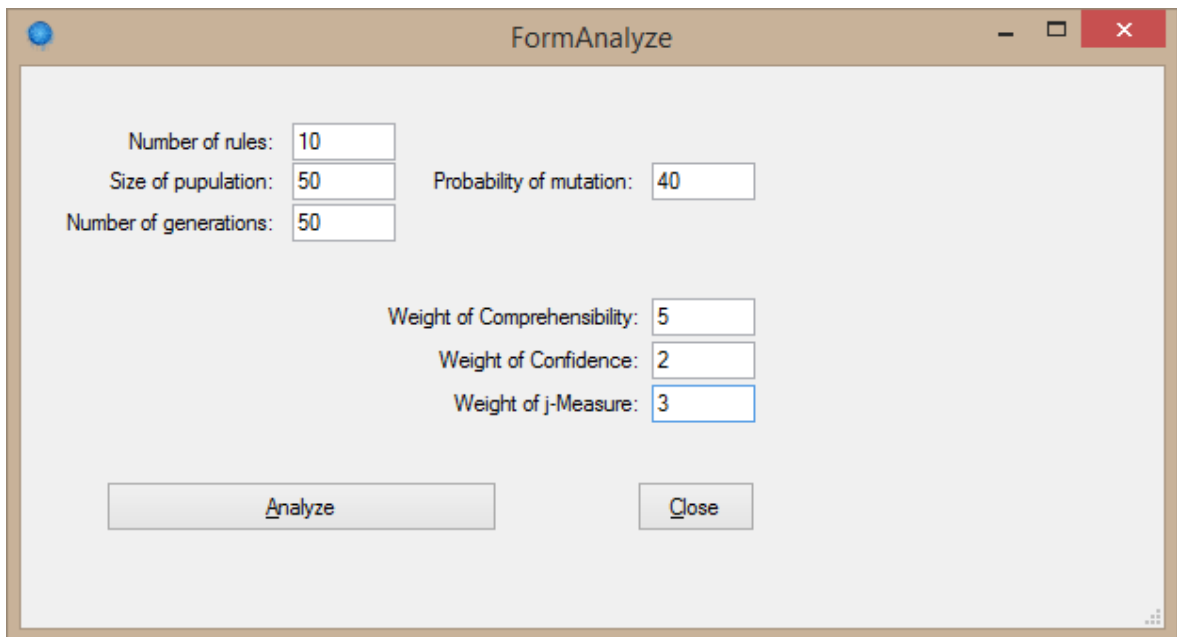
Algorithm 15: Method for roulette selection

## 4.4 Workflow of “DataGen”

Proposed program in terms of its workflow during the analysis have been designed to be as straightforward as possible. Basically the first part is about loading the data. This stage has been specified in previous chapter 4.1 Loading the data and their transformation. The next stage is actual mining of the rules. On the following picture you can see, which input data are required from the user. Otherwise whole mining does not require any other interaction.

During the search for patters there are recorded meta data about the evolution of overall population, combination of mutation and crossover and how fitness function have been changing. Various visualizations and graphs derived from these data can be accessed from the last building block of DataGen, which is called Visualize. This allows the

evaluation of whole mining and eventually helps to modify the initial parameters for future analysis.



The screenshot shows a window titled "FormAnalyze" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains several input fields and two buttons. The parameters are as follows:

Parameter	Value
Number of rules:	10
Size of population:	50
Number of generations:	50
Probability of mutation:	40
Weight of Comprehensibility:	5
Weight of Confidence:	2
Weight of j-Measure:	3

At the bottom of the window, there are two buttons: "Analyze" and "Close".

**Figure 4-7: Setting the input parameters for analysis**

DataGen also keeps the track of the whole process and core outputs from each stage are saved in log file, which is visible on main screen together with the option for its exporting outside of the application.

## 5 Discussion

To be able to evaluate the quality of developed application in terms of the predictive capabilities of discovered association rules we will simulate its usage on one of the most notorious examples, where association rules are used; the marked basket analysis.

This analysis looks at market consumer behavior using the methodology known as market basket analysis. Market basket analysis has the objective of individuating products, or groups of products, that tend to occur together (are associated) in buying transactions (baskets). The knowledge obtained from a market basket analysis can be very valuable; for instance, it can be employed by a supermarket to reorganize its layout, taking products frequently sold together and locating them in close proximity. But it can also be used to improve the efficiency of a promotional campaign: products that are associated should not be put on promotion for the same periods. By promoting just one of the associated products, it should be possible to increase the sales of that product and get accompanying sales increases for the associated products. The databases usually considered in a market basket analysis consist of all the transactions made in a certain sale period (e.g. one year) and in certain sale locations (e.g. a chain of supermarkets). Consumers can appear more than once in the database. In fact, consumers will appear in the database whenever they carry out a transaction at a sales location. The objective of the analysis is to individuate the most frequent combinations of products bought by the customers.

Data originally come pre-prepared along with Weka software. This software is open source mining tool developed at The University of Waikato, New Zealand. Within this tool they also include real-world datasets and assure their appropriate quality. In our case the dataset is called supermarket.arff. Instances are based on true market data gathered from small New Zealand supermarket during approximately one summer week in 2004. The .arff file type is designed for needs of Weka, but DataGen does not support it. So firstly, they will be converted to .csv format and only names of columns will be kept.

Final file for analysis contains 4,627 instances and 37 attributes. It is represented as a matrix of binary values {true, false}, which specify whether the attribute (specific product) was bought in given transaction. In terms of the size of market basket (number of bought items per visit) we have found that maximum is 28, minimum 0, the median is 9, average 9.69 and standard deviation equals 4.48.

Following graph shows the distribution of multiplicity (how many times each product have been bought) according to whole dataset.

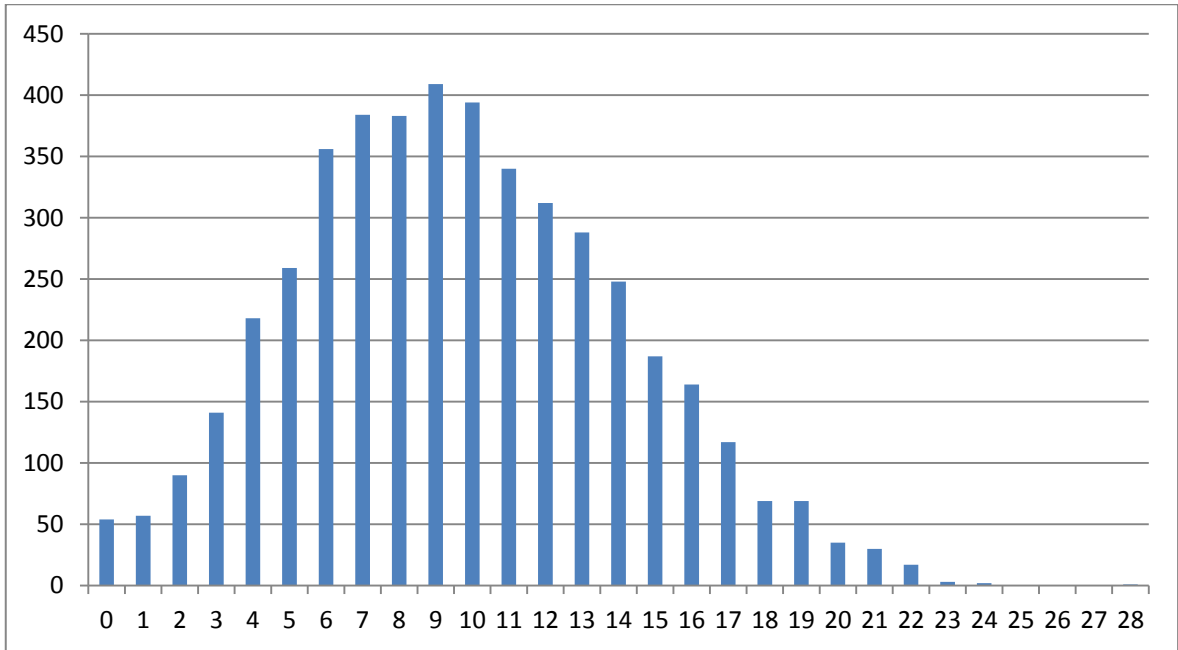


Figure 5-1: Frequency distribution of the multiplicity of bought items per transactions

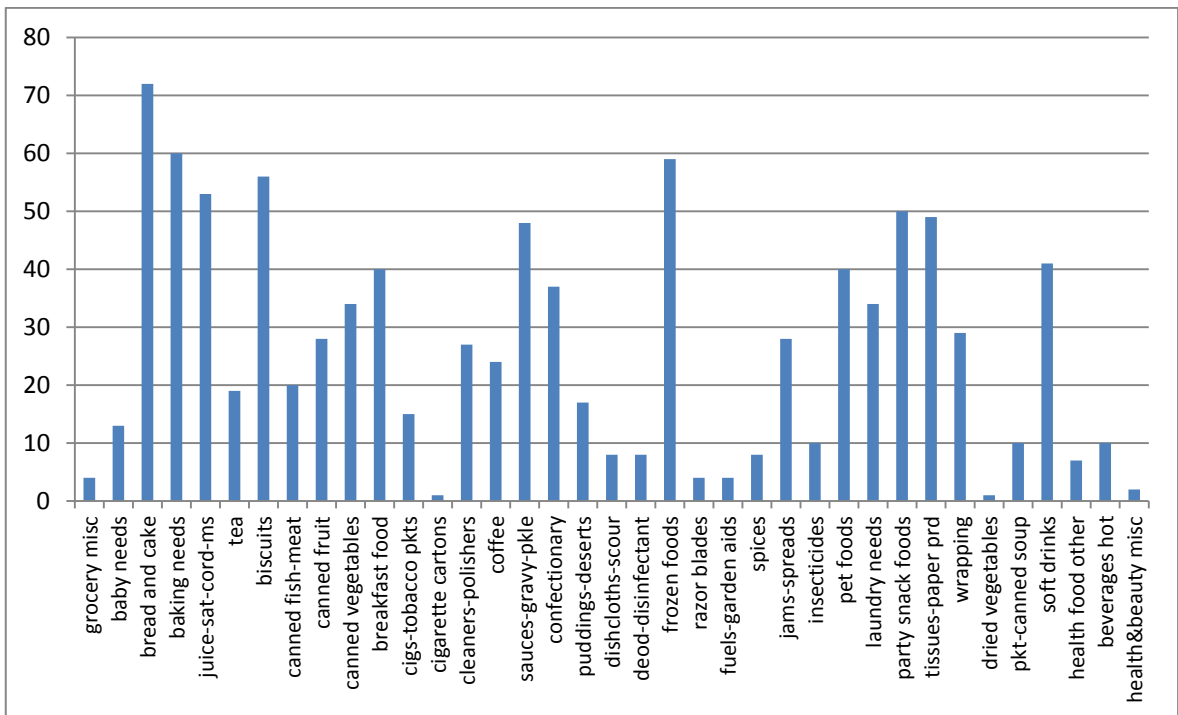


Figure 5-2: Frequency of bought items in %

Based on closer look on dataset, it has been found that the presence of some items (e.g. bread and cake, fruit and canned vegetables) is very strong from 60% to even over 70%. This might cause the problem during mining rules and lead towards not so informative

assumptions. It is expected that rules derived by lift index will be more useful and might contain higher valued information.

The next step after the loading of data and their processing is actual mining of the rules. After a few tries we have decided to mine five final rules out of the population of 30 individuals that was obtained after 20 generations. The probability of mutation was set to 15%. The weight of comprehensibility was 1, confidence 4 and j-Measure 8. Setting such weights should at least partially help to minimize the impact of too frequent attributes.

For easier interpretation, which might not be so complex, we can exclude the negative parts of the rule (items that were not bought). After this we will gain for example from the first rule following one:

*bread and cake in ( TRUE ) and cigs-tobacco pkts in ( TRUE ) and cleaners-polishers in ( TRUE ) and tissues-paper prd in ( TRUE ) → pet foods in ( TRUE ).*

We can interpret it as smokers (bought tobacco products), who frequently eat pastries (bought bread or some kind of cake) and are quite cleanly (bought tissue paper products and polishers); they also own a pet and are buying him pet foods. This means that if we will design a flyer with discounts, we should decrease price of cleaner-polishers and tissue-paper products. This might lead to increased purchases of pet foods. Lowering the price of bread and cake might have no effect on purchases of pet foods because it is very frequent attribute and tobacco products might have, but their price is bound by legislation. Similarly we can interpret also other rules. Including negative parts of the rule might lead to deeper understanding of customers and their preferences (what they do not like), but it would resulted in significantly higher complexity, hence led to more difficult interpretations of rules.

The analysis of generated association rules and metadata showed that the individuals are really improving after each generation and proposed genetic algorithm works. But there are possible improvements, which could help to increase the overall quality of rules. The biggest problem to consider is changeability of generated rules, although the input parameters were the same. This has two possible reasons; the initial population is generated too randomly and without concerns about the data. The second one is in frequent mutation, which randomizes individuals and causes the loss of genetic information. But if we decrease the probability of mutation, the rules tend to converge to very similar ones (with just a few different attributes or values of attributes). Also frequent

occurrence of crossover operator tends to lead to growing length of the rules and extremely long rules have significantly lower j-Measure and confidence, because they are binding to much information and there are not enough instances that would fit into such rules.

Solution for these problems, which also means the possible improvement of DataGen, might be in modifying the initialization of initial population and the need to focus also on the length of generated individuals. This means that the initial population should consist of shorter, more straightforward rules and after each pass of generation allow the rules to be longer, until j-Measure and confidence would begin to decrease. Otherwise the whole fitness function will be driven only by comprehensibility.

From the user perspective it might be useful to work on more complex approach in definition of rules, include the choices to generate only positive rules, negative or mixed. Also extending the fitness function and adding more metrics might bring bigger flexibility to user. And finally the designing extra possibilities for evaluation of the rules, such as modification of confusion matrices or ROC curves might bring better insights about generated rules, their quality and support the implementation that follows afterwards.



## Conclusion

As we have shown association rules represent an important part of data mining, they involve statistic tools as well as machine learning techniques. This combination enables to search for different patterns and rules that might represent the source of competitive advantage in any kind of business. Association rules provide deeper understanding of data and enable their transformation to information and knowledge. Based on this purpose of association rules and their characteristics, such as easy understanding of outputs, fast implementation to business environment and their predictive capabilities, they are frequently used.

Utilization of evolution computation techniques, especially genetic algorithms, to induce association rules evolutionary algorithms has shown as an interesting alternative to typical greedy algorithms, which are commonly used for discovering of association knowledge. Their bring significant benefits, such as better abstraction from actual dataset, possibility to focus more on business needs and define the requirements on rules that reflect it. Due to population-based metaheuristic optimization that uses solution space search mechanisms inspired by biologic evolution we can also overcome some imperfections of input datasets, as well.

Finally we have designed own genetic algorithm and implemented in fully functional desktop application called DataGen. Designing this algorithm included the definition of solution encoding, evaluation of individuals and genetic operators. All parts were successfully developed, which enables evolving the association rule models from a given input dataset. We have also included a module that collects the metadata from whole process of evolution and allows their visualization. Based on the results gained from usage of DataGen we can conclude that it really improves the individuals in each generation and successfully mines the association rules.

We also believe that suggested improvements of DataGen will lead to even better results, more precise rules, better user experience while using our application and will provide bigger flexibility that will allow truly fulfill user needs and expectations. But to achieve the successful implementation of suggested chances further research is definitely necessary, especially in the areas of generating the initial populations, multiple objective fitness functions, multi-pareto-ranking and alternative ways for evaluation of generated rules.

# Resumé

## Úvod

Informácia predstavuje ťažisko a najvýznamnejší zdroj konkurenčnej výhody v znalostnej ekonomii. K ich získaniu a osvojovaniu vždy predchádza spracovanie dát a údajov. Po úspešnej systematizácii vstupných dát je taktiež rovnako dôležitá aj ich implementácia do procesu riadenia a podpory rozhodovania. Preto správne a relevantné informácie, ak sú aj vhodne implementované tvorbu vlastnej, špecifickej pridanej hodnoty, ktorá je ťažko kopírovateľná a napodobniteľná a tak sa informácie pretavia do unikátnej konkurenčnej výhody. To je aj hlavným dôvodom prečo neustále dopyt po informáciách rastie.

Nedielnou súčasťou získavania informácií je aj ich samotná interpretácia. Vedieť sa vyznať alebo orientovať sa v exponenciálne narastajúcom objeme dát, oddeliť tie relevantné od redundantných a v konečnom dôsledku vedieť, na čo sa zamerať už dlho nie je možné bez informačných systémov. Techniky a metódy na získavanie informácií z dát sú čoraz sofistikovanejšie a komplexnejšie, schopnejšie spracovávať nadmerné množstvo údajov. Preto v dnešnej dobe nemôžeme vnímať informačný nedostatok ako následok nedostatočného množstva dát. Problém a teda aj výzva, ktorú sa snaží táto práca riešiť, je skôr v oblasti samotnej interpretácie informácií a odstraňovaní informačnej entropie, čiže získavaní tých relevantných informácií.

Jednou z možností ako objavovať poznatky je práve aj data mining, ktorého úlohou je vyhľadávať relevantné a netriviálne informácie z masívnych údajových základní. Zároveň však platí, že problematika pozorovania akejkoľvek údajovej základne a následná interpretácia poznatkov je veľmi prepojená aj na jednotlivé vedné disciplíny ako také a tie majú mnohokrát svoje špecifické postupy a algoritmy. Úzka špecializácia pri riešení akýchkoľvek problémov objavovania poznatkov zasa môže byť nahradená plytkými metódami, medzi ktoré patria aj evolučné algoritmy.

Preto sa práca sa zaoberá objavovaním asociatívnych pravidiel, ako jedným z prvotných štádií dátami riadeného, nepriameho, data miningu využitím evolučných algoritmov. V rámci práce je aj navrhnutý a implementovaný vlastný genetický algoritmus, ktorého výkonnosť je aj následne analyzovaná.

## **Súčasný stav riešenej problematiky na Slovensku a v zahraničí**

Na dáta ako také je potrebné sa pozerat' ako na surový materiál, ide vyslovene len o alfanumerické, respektíve čisto numerické reťazce. To znamená, že samé o sebe nemajú žiadny veľký význam pre užívateľa alebo ich vlastníka, lebo nepredstavujú konkurenčnú výhodu. Sú to jednoduché a priame fakty, údaje. Čiže tak ako sa musí surový materiál spracovať, aby mohol vzniknúť výsledný tovar, tak aj dáta podliehajú ďalšiemu spracovaniu. Výsledkom týchto procesov sú už samotné informácie. Čiže o informáciách môžeme povedať, že predstavujú organizovaný údaj, pričom musí platiť, že príjemcovi informácie odstraňuje informačnú entropiu. To v praxi znamená, že dobrá informácia musí mať význam, byť zdrojom nových faktov a tieto fakty sú pre danú činnosť relevantné a vecné, v najideálnejšom prípade predtým nepoznané.

Poznanie predstavuje ďalší stupeň alebo vývojovú etapu, lebo je možné vyjadriť ho môžeme ako systematizáciu informácií. V tomto prípade už uvažujeme priamo o schopnosti riešiť problémy. Väčšinu riešení akéhokoľvek problému je možné rozdeliť na menšie, triviálnejšie úlohy a práve tieto jednotlivé kroky riešenia boli získané z informácií, pretože primárnou úlohou informácií je čo najpresnejšie popisovať realitu. Táto podmienka sa odzrkadľuje potom aj na požiadavky, ktoré musia spĺňať aj vstupné dáta. Konkrétne ide o kvality a bezchybnosť dát. Ak máme pracovať s neúplnými alebo zlými vstupnými dátami, potom je vysoko pravdepodobné, ba priam až isté, že aj získané informácie nebudú dostatočne kvalitné a teda nebude možné ich použiť na riešenie problémov.

Ďalšou vývojovou etapou alebo pojmom, ktorý sa v poslednej dobe pridáva dátam, informáciám a poznatkom je múdrosť. V tomto prípade múdrosť samozrejme zahŕňa poznatky o danej realite, ale rozširuje ich aj o aj osobné skúsenosti. Pod skúsenosťami sa môžu rozumieť prvky, ktoré sa nedajú jednoznačne matematicky vyjadriť alebo ohodnotiť. Tieto prvky majú pre každého užívateľa inú váhu pri rozhodovaní a teda skôr ide o rôzne nápady, tendencie správania a podobne. Pridaním individuálnych prvkov k znalostiam, ktoré boli založené na abstrakcii objektívnych informácií o minulosti a prípadnej identifikácii konštantných vzťahov a vlastností rozširujeme práve múdrosť. Výsledkom čoho potom tvoríme rôzne modely správania, či už nákupného, ekonomického alebo sociálneho charakteru. Takto postavené modely následne slúžia nie len na popisovanie historických a aktuálnych stavov, ale aj budúcich očakávaní a rôznych predikcií a stanovovaní cieľov.

Práve techniky data miningu slúžia na to, aby sa z dát, ktoré predstavujú zdroj pre získanie informácií dali „odvodiť“ aj poznatky, ktoré pomôžu identifikovať a vhodne využiť aj menej predpokladané informácie.

### *Data mining a jeho ciele*

Data mining ako taký spadá pod oveľa rozsiahlejšiu vednú disciplínu a to konkrétne Objavovanie Poznatkov v Databázach (Knowledge Discovery in Databases - KDD). Celý proces KDD sa dá rozdeliť na niekoľko základných krokov. Práve ich pochopenie predstavuje základ správneho prístupu k data miningu a očakávaní, čo nám môže priniesť, respektíve, na čo je určený. Prvým krokom je selekcia dát, po nej nasleduje predspracovanie dát. Tieto dva kroky v sebe zahŕňajú práve výber vhodných, relevantných dát, pričom sa často abstrahuje od ich nedostatkov. Tretím krokom je transformácia dát. V tomto bode je pozornosť venovaná samotnej úprave dát pre potreby analýz, tiež sa zameriava na zabezpečenie kvality dát, spracovaniu a minimalizovaniu ich nedostatkov. Ďalším krokom je samotný data mining. V tejto časti práve prebieha transformácia dát na informácie. Výstupom sú modely, ktoré na základe historických dát napríklad z transakčných databáz dokážu identifikovať vzory správania, tendencie, trendy a prípadne poskytnúť rôzne odporúčania. Finálnym krokom je vyhodnotenie a interpretácia. V tomto štádiu je pozornosť venovaná kvalite vyprodukovaných pravidiel, ich relevantnosti a návrhu možných implementácií za cieľom získania konkurenčnej výhody.

Samotný data mining je následne tiež možné rozdeliť do niekoľkých podskupín. Jedným zo základných členení je na priamy a nepriamy. Priamy data mining má za cieľ využiť dostupné dáta na tvorbu predikčného modelu, ktorý popíše premennú záujmu pomocou zvyšných dát (premenných). Ide o prístup „zhora – dolu“ a používa sa vtedy, keď vieme, čo hľadáme. Modely sú vo forme „čiernej skrinky“ (proces spracovania nás nezaujíma) alebo šedej - semitrSPARENTNÉ modely (potrebujeme vedieť ako model pracuje). Modely všeobecne predstavujú spracovanie viacerých vstupov do jedného žiadaného výstupu. Medzi techniky priameho data miningu patria napríklad klasifikácie, odhady, predpovede a regresné modely.

Druhá skupina nástrojov, ktoré patria pod data mining sú nepriame. Cieľom nepriameho data miningu je vytvoriť nejaké vzťahy medzi všetkými premennými. Ide o riešenie „zdola – nahor“. To znamená, že necháme dáta, aby nás viedli, neovplyvňujeme cieľ pravidiel ani ich bližšiu špecifikáciu. Pod nepriamy data mining zaradujeme napríklad

hľadanie podobností, asociatívne pravidlá, zhukovanie (clustering), popisné a vizualizačné metódy.

Na to, aby sa mohli jednotlivé metódy úspešne implementovať data mining požíva rozsiahly aparát, ktorý zahŕňa štatistické metódy, techniky strojového učenia a niektoré postupy, ktoré predstavujú ich prienik. Zo štatistického aparátu sú jednými z najčastejších nástrojov lineárna a logistická regresia, viacrozmerné metódy a analýza časových radov. Avšak vzhľadom na to, že väčšinou sa v data miningu pracuje s rozsiahlym množstvom dát, čo priamo ovplyvňuje komplexnosť a výpočtovú náročnosť daných úloh, používané nástroje sa obohatili aj o oblasť strojového učenia. To zahŕňa napríklad neurónové siete alebo celú skupinu evolučných algoritmov a vektorové učenie. Ďalšou výhodou tejto oblasti je, že boli vyvinuté metriky na vyhodnocovanie modelov, ktoré lepšie odrážajú potreby data miningu. Keďže pracujeme s veľkým objemom dát, tak štandardné testovanie hypotéz s ktorým sa stretáme v deskriptívnej štatistike nemusí byť vždy postačujúce, lebo sa stáva, že takmer každý model sa javí ako štatisticky významný. Medzi nástroje ktoré predstavujú prienik strojového učenia a štatistických metód môžeme napríklad zaradiť klasifikačné a regresné stromy.

### *Charakterizácia asociatívnych pravidiel*

Pod asociatívnymi pravidlami možno chápať určité súvislosti (nájsť produkty, ktoré sa často predávajú spoločne a tieto vhodne umiestniť v predajni). Majú tvar implikácie, čiže *AK (podmienka) POTOM (následok)*. Čo sa dá zapísať aj ako *podmienka*  $\rightarrow$  *dôsledok* ( $X \rightarrow Y$ ). Pričom musí platiť, že prienik podmienky a dôsledku musí byť prázdna množina. Zároveň podmienková a dôsledková časť môžu predstavovať zložené výrazy. Pri základných typoch asociácií sú výrazy pospájané len konjunkciou. Rozšírenia môžu uvažovať aj o zjednotení, negáciách a iných logických operátoroch, ale takéto pravidlá sú komplexnejšie, ťažšie na pochopenie a interpretáciu. Preto sa im v tejto práci nevenujeme.

O objavovaní asociatívnych pravidiel sa dá povedať, že ide o zisťovanie súvislostí učením bez učiteľa. Môže napríklad ísť o pochopenie asociácie medzi rôznymi skupinami tovarov z nákupného hľadiska: zákazník  $\rightarrow$  jeho nákup  $\rightarrow$  databáza. Napríklad s cieľom vhodného umiestnenia v regáloch, akcií na tovar a pod. Asociatívne pravidlo má pravdepodobnostný charakter.

Základným princípom pri tvorbe asociatívnych poznatkov je posúdenie všetkých možných pravidiel a na základe zvolenej štatistiky sa vyberú tie, ktoré opisujú skutočné závislosti. To znamená, že hlavným problémom nie je vytvoriť asociatívne pravidlá, ale navrhnúť proces, ktorý identifikuje tie pravidlá, ktoré ponúkajú zaujímavé asociácie, ktoré sú podložené aj dátami, tak aby užívateľ nebol zahľtený množstvom nepodstatných, zjavných alebo nepresných pravidiel.

Asociatívne pravidlá predstavujú dnes veľmi obľúbený koncept a to najmä vďaka ich jednoduchej interpretácii, širokej škále možností implementácie a schopnosti celkom presne popisovať nájdené vzory s dôrazom na určenie trendov a predpovedí. Asociatívne pravidlá sú najčastejšie používané pri analýze nákupného košíka (za týmto účelom boli aj pôvodne vytvorené), preto majú uplatnenie pri predaji a priamom marketingu. Pomáhajú pri návrhu obsahu katalógov, predaji komplementárnych tovarov alebo návrhu rozmiestnenia predajne. Novou doménou asociatívnych pravidiel je e-commerce, kde sa používajú napríklad na customizáciu webových stránok a cielený marketing. Medzi najpopulárnejšie algoritmy na získavanie asociatívnych pravidiel patria napríklad Apriori a FP-growth.

## **Výsledky práce**

Hlavným výsledkom práce je vlastný genetický algoritmus, ktorý je aj implementovaný v plne funkčnej desktopovej aplikácii s názvom DataGen.

### *Reprezentácia jedincov a genotyp*

Pri výbere reprezentácie jedincov sme využili binárnu reprezentáciu. Textový fenotyp prevádzame pomocou „dekódovacej tabuľky atribútov“ na celočíselný genotyp. Prvým krokom je získanie unikátnych hodnôt pre každý stĺpec, čiže napríklad pre atribút Pohlavie to je {muž, žena}. Po vytvorení dekódovacích tabuliek atribútov sa vytvorí genotyp. Jednotlivé hodnoty genotypu predstavujú celé čísla. Pričom platí, že celé číslo odkazuje na index danej hodnoty atribútu v atribútovej tabuľke. Indexovanie je robené štýlom „First come, first serve“, čiže dáta na netriedia a berú sa tak, ako sú zaznamenané. Čo umožnilo prechádzať vstupné dáta len raz. Ak sa objaví nová, neznáma hodnota daného atribútu, tak je pridaná medzi predchádzajúce hodnoty a následne sa používa jej index pri generovaní genotypu. Preto prvý riadok v genotype vždy začína nulovými hodnotami.

Druhý, ak žiadny z atribútov nenadobúda rovnakú hodnotu ako v prvom riadku bude vyplnený jednotkami a podobne.

Reprezentácia jedincov je riešená na báze dvojrozmerného binárneho poľa. Samotná binárna hodnota *true*, resp. *false* hovorí o tom, či sa daná hodnota atribútu v pravidle nachádza alebo nenachádza. Prvé dva bity zasa hovoria či je daný atribút v podmienkovej, následkovej časti alebo sa v pravidle nenachádza. Prvá dimenzia určuje daný atribút (napríklad, že ide o Pohlavie), druhá dimenzia určuje v ktorej časti pravidla sa atribút nachádza a ktoré z jeho hodnôt sú aktívne.

Keďže dve hodnoty druhej dimenzie hovoria o umiestnení atribútu v rámci pravidla, tak platí, že chromozóm v danom atribúte je o dva polia väčší ako jeho počet unikátnych hodnôt. Pri práci s chromozómami sa museli aj ošetriť možnosti, kedy pravidlo z logického hľadiska nemá význam. Pri danej reprezentácii mohli nastať dve situácie. Prvou je, že podmienková alebo následková časť pravidla boli úplne prázdne a druhou nelogickou situáciou by bolo ak daný atribút má všetky svoje hodnoty aktivované alebo naopak neúčinné.

Generovanie inicializačnej populácie je robené náhodným generovaním hodnôt *true* a *false* do dvojrozmerného poľa, ktoré je následne upravované, aby boli pravidlá zmysluplné.

### *Fitness funkcia*

Pre každý chromozóm sa počíta aj fitness funkcia. Tá je v našom prípade zložená z troch deskriptívnych štatistík vychádzajúcich z teórie informácie. Prvou je confidence, ktorá popisuje prediktívnu presnosť pravidla, ďalej používame comprehensibility, tá popisuje dĺžku a „zrozumiteľnosť“ pravidla a poslednou metrikou je j-Measure, popisuje pridanú hodnotu pravidla (zníženie informačnej entropie). Ich spojenie je docielené váženým aritmetickým priemerom, pričom váhy sú jedným zo parametrov, ktoré užívateľ zadáva. Tým je docielené, že užívateľ nemusí poznať štruktúru dát do hĺbky a môže sa zamerať na typ pravidiel z jeho hľadiska a potrieb.

## *Genetické operátory*

Mutácia a kríženie sú samotné metódy jednotlivých chromozómov. Pri ich uskutočňovaní je potrebné následne pravidlá aj mierne opravovať, aby nevznikali nelogické nové jedince.

### **Mutácia**

Mutácia prebieha v dvoch krokoch, najprv sa náhodne vyberá, ktoré atribúty zmenia svoju pozíciu v rámci pravidla. Druhým krokom je zmena aktivácie konkrétnych hodnôt samotných atribútov. Pravdepodobnosť vykonania mutácie je stanovená samotným užívateľom. Pričom sa však zohľadňuje aj kvalita vstupujúceho rodiča. Čím vyššiu má hodnotu fitness funkcie, tým je pravdepodobnosť mutácie nižšia. Doplnkom je pravdepodobnosť kríženia.

### **Kríženie**

Kríženie je prioritne dvojbodové a vymieňajú sa celé hodnoty atribútov. Ak sa však vplyvom náhodného výberu stane to, že dva body kríženia splynú do jedného, tak ide o jednobodové kríženie.

### **Selekcia**

Výber rodičovských chromozómov, ktoré sú následne transformované genetickými operáciami sa uskutočňuje ruletovým výberom. Výber do novej populácie je silno preferenčný a odstraňuje už vybrané chromozómy, aby bola zaručená diverzita populácie a zároveň vzrastajúca kvalita jedincov. Pričom jedince sa vyberajú z pôvodnej populácie a z vytvorených potomkov.

### **Záver**

Navrhnutý genetický algoritmus na objavovanie asociatívnych pravidiel zlepšuje kvalitu jedincov. Poskytuje aj značnú diverzitu pri úvodnom štádiu prehľadávania stavového priestoru a produkuje pravidlá správne, relevantné a podľa očakávaní. Avšak veríme, že navrhované zmeny programu prispejú k ešte lepším výsledkom, presnejším pravidlám a v konečnom dôsledku DataGen ponúkne ešte väčšiu flexibilitu, ktorá umožní plne uspokojiť požiadavky a očakávania užívateľa. Na ich správnu implementáciu je však potrebný ďalší výskum a to predovšetkým v oblastiach generovania prvotnej populácie, multi-pareto-rankingu a alternatívnych možností vyhodnocovania nájdených pravidiel.



## References

### Books and papers

- [1] BACK, T. – FOGEL, D. B. – MICHALEWICZ Z. 2000. *Evolutionary Computation 1: Basic Algorithms and Operators*. Bristol : Institute of Physics Publishing, 2000. 378 s. ISBN 0-75030-664-5.
- [2] DE JONG, K. A. 2002. *Evolutionary Computation*. Cambridge : Evolutionary Computation, 2002. 256 p. ISBN 0-262-04194-4.
- [3] EIBEN, A. E. – SMITH, J.E. 2003. *Introduction to Evolutionary Computing*. New York : Springer, 2003. 299 p. ISBN 978-3-662-05094-1.
- [4] GIUDICI, P. 2003. *Applied Data Mining: Statistical Methods for Business and Industry*. Chichester : John Wiley & Sons Ltd, 2003. p. 357. ISBN 0-470-84679-8.
- [5] HAND, D. J. – MANILLA, H. and SMITH, P. 2001. *Principles of Data Mining*. Cambridge : MIT Press, 2001. 546 p. ISBN 978-0-262-08290-7.
- [6] MACH, M. 2009. *Evolučné algoritmy: Prvky a princípy*. Košice : VEGA, 2009. 233 p. ISBN 978-80-8086-123-0.
- [7] MAIMON, O. – ROKACH, L. 2010. *Data Mining and Knowledge Discovery Handbook*. 2nd ed. New York : Springer, 2010. 1279 p. ISBN 978-0-387-09822-7.
- [8] SOTO, W. – OLAYA-BENAVIDES, A. 2011. *A Genetic Algorithm for Discovery of Association Rules*. Bogotá : Intelligent systems and spatial information research group (SIGA), 2011.
- [9] TUFFÉRY, S. 2011. *Data Mining and Statistics for Decision Making*. 1st ed. Chichester : John Wiley & Sons Ltd, 2011. 645 p. ISBN 978-0-470-68829-8.
- [10] WITTEN, I. H. – FRANK, E. – HALL, M. A. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. 3rd ed. Burlington : Morgan Kaufmann Publishers, 2011. 607 p. ISBN 978-0-12-374856-0.
- [11] YAN, X. – ZHANG, Ch. and ZHANG, S. 2005. *Argma: Identifying interesting association rules with genetic algorithms*. ISSN: 0883-9514, Sydney : Taylor & Francis Inc., 2005.

### Articles in journals and almanacs

- [1] BLANCHARD, J. – GUILLET, F. – GRAS, R. – BRIAND Henri. 2005. Using Information-theoretic Measures to Assess Association Rule Interestingness. In Houston : *5th IEEE International Conference on Data Mining ICDM'05*. 2005.

- [2] FAYYAD, U. – PIATETSKY-SHAPIRO, G. – SMYTH, P. 1996. From Data Mining to Knowledge Discovery in Databases. In *AI Magazine*. ISSN 0738-4602, 1996, vol. 17, no. 3, p. 37-54.
- [3] HAN, J. et al. 2004. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. In *Data Mining and Knowledge Discovery*. 2004, p. 53–87.
- [4] WAKABI-WAISWA, P. P. – BARYAMUREEBA, V.. 2008. Extraction of interesting association rules using genetic algorithms. In *International Journal of Computing and ICT Research*. ISSN 1818-1139, 2008, vol. 2, no. 1, p. 26-33.

### **Internet sources**

- [1] DATA MINING ARTICLES. 2013. Association Rules Algorithms. In *Association Analysis*. [online]. 2013 [cited 2014-03-25]. Available on the internet: <<http://www.dataminingarticles.com/association-analysis/association-rules-algorithms/>>.
- [2] DAVIES, Ch. 2011. Quote. In *DavesDailyQuotes*. [online]. 2011-07-24 [cited 2014-02-21]. Available on the internet: <<http://davesdailyquotes.com/14595/>>.
- [3] FROST, A. 2014. Defining Knowledge, Information, Data. In *Knowledge Management Tools*. [online]. 2014-04-15 [cited 2014-04-20]. Available on the internet: <<http://www.knowledge-management-tools.net/knowledge-information-data.html>>.
- [4] GOOGLE. 2010. Lambdaj. In *Google code*. [online]. 2010 [cited 2014-04-05]. Available on the internet: <<https://code.google.com/p/lambdaj/>>.
- [5] HAMILTON, H. 2000. Confusion Matrix. In *Computer Science 831: Knowledge Discovery in Databases*. [online]. 2000-09-01 [cited 2014-04-5]. Available on the internet: <[http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion\\_matrix/confusion\\_matrix.html](http://www2.cs.uregina.ca/~dbd/cs831/notes/confusion_matrix/confusion_matrix.html)>.
- [6] CHAPMAN, P. 2008. The CRISP-DM User Guide. In *SMU Lyle School of Engineering*. [online]. 2008-05-31 [cited 2014-04-01]. Available on the internet: <<http://lyle.smu.edu/~mhd/8331f03/crisp.pdf>>.
- [7] KNOWLEDGE NUGGETS. 2007. Data Mining Methodology. In *Pools*. [online]. 2007-08-01 [cited 2014-04-01]. Available on the internet: <[http://www.kdnuggets.com/polls/2007/data\\_mining\\_methodology.htm](http://www.kdnuggets.com/polls/2007/data_mining_methodology.htm)>.
- [8] ORACLE. 2008. Association. In *Oracle® Data Mining Concepts*. [online]. 2008 [cited 2014-04-01]. Available on the internet:

<[http://docs.oracle.com/cd/B28359\\_01/datamine.111/b28129/market\\_basket.htm#DMCON009](http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/market_basket.htm#DMCON009)>.

- [9] SCIENCEDAILY. 2013. Big Data, for better or worse: 90% of world's data generated over last two years. In *Articles*. [online]. 2013-05-22 [cited 2014-04-05]. Available on the internet:  
<<http://www.sciencedaily.com/releases/2013/05/130522085217.htm>>.
- [10] TIERNEY, B. 2012. Data Science Is Multidisciplinary. In *Blog*. [online]. 2012-06-13 [cited. 2014-April-1]. Available on the internet:  
<<http://www.oralytics.com/2012/06/data-science-is-multidisciplinary.html>>.
- [11] UNIVERSITY OF UTAH. 2013. Matlab Tutorial. In *Thermal Geophysics Research Group*. [online]. 2013-07-16 [cit. 2014-04-15]. Available on the internet:  
<<http://thermal.gg.utah.edu/tutorials/matlab/peaks.png>>.

## Appendixes

<b>A</b>	<b>Sample of outputs from DataGen .....</b>	<b>i</b>
A.1	Log file .....	i
A.2	Visualization of metadata generated during mining .....	iii

## A Sample of outputs from DataGen

DataGen allows exporting the log file, that user can see during mining. It contains information about loading and analysis. Also the graphs that are built over generated metadata can be exported, these include the charts of fitness function values, individual metrics from which the fitness function was build and comparison of multiplicities of genetic operators in each generation.

### A.1 Log file

```
|***** D A T A G E N *****|
```

```
==> L O G   F I L E   <==
```

```
|*****|
```

```
-----  
Date and time of start: 04/20/2014, 22:46:34
```

```
-----  
22:46:49 28. 04/20/2014
```

File is ready for analysis:

Generation of genotype lasted: 0 hr, 0 min, 0 sec, 81 ms

Number of attributes: 37

Number of instances: 4627

```
-----  
22:47:11 04/20/2014
```

Rules extraction has been successful:

Rule extraction lasted: 0 hr, 0 min, 2 sec, 547 ms

Number of generations: 20

Size of population: 30

Weight of confidence: 4

Weight of j-Measure: 8

Weight of comprehensibility: 1

Probability of mutation: 10

#### GENERATED RULES:

1. RULE: Fitness = 1,00738835618215

baby needs in ( FALSE ) and bread and cake in ( TRUE ) and juice-sat-cord-ms in ( FALSE ) and tea in ( FALSE ) and biscuits in ( FALSE ) and cigs-tobacco pkts in ( TRUE ) and cigarette cartons in ( FALSE ) and cleaners-polishers in ( TRUE ) and sauces-gravy-pkle in ( FALSE ) and confectionary in ( FALSE ) and spices in ( FALSE ) and jams-spreads in ( FALSE ) and tissues-paper prd in ( TRUE ) and wrapping in ( FALSE ) => coffee in ( FALSE ) and dishcloths-scour in ( FALSE ) and pet foods in ( TRUE ) and laundry needs in ( FALSE ) and pkt-canned soup in ( FALSE ) and health&beauty misc in ( FALSE )

2. RULE: Fitness = 0,638575488792696

juice-sat-cord-ms in ( FALSE ) and tea in ( FALSE ) and biscuits in ( FALSE ) and cigs-tobacco pkts in ( TRUE ) and cigarette cartons in ( FALSE ) and cleaners-polishers in ( TRUE ) and sauces-gravy-pkle in ( FALSE ) and confectionary in ( FALSE ) and spices in ( FALSE ) and jams-spreads in ( FALSE ) and tissues-paper prd in ( TRUE ) and wrapping in ( FALSE ) => coffee in ( FALSE ) and dishcloths-scour in ( FALSE ) and razor blades in ( FALSE ) and pet foods in ( TRUE ) and laundry needs in ( FALSE ) and pkt-canned soup in ( FALSE )

3. RULE: Fitness = 0,624878464838243

juice-sat-cord-ms in ( FALSE ) and tea in ( FALSE ) and biscuits in ( FALSE ) and cigs-tobacco pkts in ( TRUE ) and cigarette cartons in ( FALSE ) and cleaners-polishers in ( TRUE ) and sauces-gravy-pkle in ( FALSE ) and confectionary in ( FALSE ) and spices in ( FALSE ) and jams-spreads in ( FALSE ) and tissues-paper prd in ( TRUE ) and

wrapping in ( FALSE ) => coffee in ( FALSE ) and dishcloths-scour in ( FALSE ) and pet foods in ( TRUE ) and laundry needs in ( FALSE ) and pkt-canned soup in ( FALSE )

4. RULE: Fitness = 0,623215276197977

baby needs in ( FALSE ) and bread and cake in ( TRUE ) and juice-sat-cord-ms in ( FALSE ) and biscuits in ( TRUE ) and coffee in ( TRUE ) and sauces-gravy-pkle in ( FALSE ) and confectionary in ( FALSE ) and insecticides in ( TRUE ) and tissues-paper prd in ( TRUE ) and wrapping in ( FALSE ) => tea in ( FALSE ) and cigs-tobacco pkts in ( FALSE ) and dishcloths-scour in ( FALSE ) and frozen foods in ( TRUE ) and spices in ( FALSE ) and jams-spreads in ( FALSE ) and pet foods in ( TRUE ) and laundry needs in ( FALSE ) and pkt-canned soup in ( FALSE ) and health&beauty misc in ( FALSE )

5. RULE: Fitness = 0,609529447929022

juice-sat-cord-ms in ( FALSE ) and biscuits in ( TRUE ) and coffee in ( TRUE ) and sauces-gravy-pkle in ( FALSE ) and confectionary in ( FALSE ) and insecticides in ( TRUE ) and tissues-paper prd in ( TRUE ) and wrapping in ( FALSE ) => tea in ( FALSE ) and cigs-tobacco pkts in ( FALSE ) and dishcloths-scour in ( FALSE ) and frozen foods in ( TRUE ) and spices in ( FALSE ) and jams-spreads in ( FALSE ) and pet foods in ( TRUE ) and laundry needs in ( FALSE ) and pkt-canned soup in ( FALSE )

-----

## A.2 Visualization of metadata generated during mining

