

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Decision Rule Induction Based on the Graph Theory

Izabela Kutschenreiter-Praszkiewicz

Abstract

The graph theory is a well-known and widely used method of supporting the decision-making process. The present chapter presents an application of a decision tree for rule induction from a set of decision examples taken from past experiences. A decision tree is a graph, where each internal (non-leaf) node denotes a test on an attribute which characterises a decision problem, each branch (also called arc or edge) represents the outcome of a test (attribute value), and each leaf (or terminal) node holds a class label which can be interpreted as a decision type. In the presented approach, the object-attribute-value (OAV) framework will be used for decision problem characteristics. The chapter presents a method of optimal decision tree induction. It discusses the Iterative Dichotomiser 3 (ID3) algorithm and provides an example of the decision tree induction. Also, rules supporting the decision-making in engineering will be developed in this chapter.

Keywords: decision tree, decision rule induction, ID3 algorithm, QFD, dependence network

1. Introduction: decision problems in industry

In order to solve decision problems, we need knowledge, information and data. Data are understood as a set of discrete objectives, facts and events. Information represents a relationship between some pieces of data. For data to be transformed into information, raw data should be cleaned, corrected and processed while considering the context of a decision. Knowledge can be defined as application of the relevant information to address specific problems. Knowledge in decision problems is developed based on the decision-maker's experiences. Hence, data gathering and analysis is one of the necessary stages of decision support systems development with the use of artificial intelligence.

The main issues related to decision support with the use of artificial intelligence are:

- Representing knowledge as close as possible to the human way of thinking.
- Knowledge representation should be easy to use and updated with computer software.

Decision problems in industry may be classified into one of the following decision categories:

- Structured
- Unstructured
- Semi-structured

Structured decisions are routine, and there are usually specific procedures or actions that can be identified to help make the decision. In case of unstructured decisions, decision scenarios often involve new or unique problems, and an individual has little or no programmatic or routine procedure for addressing the problem or making a decision. In semi-structured decisions, decision scenarios involve both some structured and some unstructured components.

According to the data analysed by Iyer et al. [1], in product design, only 20% of parts are new, while 80% of them are reused or modified [2], so knowledge management focused on knowledge reuse related to product and production process is an important field of analysis and can influence the effectiveness of production process preparation.

Knowledge representation is a field of artificial intelligence that focuses on designing computer representations that capture knowledge which can be used to solve problems.

A knowledge representation should be:

- Rich enough to express the knowledge needed to solve the problem
- As close to the problem as possible
- Compact, natural, and maintainable
- Easy to see the relationship between the representation and the domain being represented
- Able to conduct efficient computation, which usually means that it is able to express problem features that can be exploited for computational gain and able to trade off accuracy and computation time
- Able to be acquired from people, data and past experiences

According to formalism used for knowledge representation, knowledge can be classified as:

- Procedural knowledge, which defines algorithms that help to achieve given goals
- Declarative knowledge, which gives the solution without analysing the problem structure

Taking into consideration knowledge transfer to another person, knowledge can be divided into:

- Tacit, which is difficult to express by words or equations
- Explicit, which is easy to formulate and transfer

The object-attribute-value (OAV) model is a knowledge representation framework. Objects are understood in OAV as entities that are items being described. Attributes are understood as features, properties, characteristics or variables of a given object. Value is understood as a numerically discreet or linguistically defined attribute esteem.

OAV can be used in artificial intelligent methods as a framework of data analysis useful for data preparation for applying chosen knowledge representation method. A data analysis schema is presented in **Figure 1**.

Knowledge representation methods include:

- Decision rules—which transform premises (e.g. observations expressed in the OAV framework) into conclusions (e.g. decision expressed in the OAV framework).
- Decision trees—which are graph representations of the decision process. Inspecting the conditions in the decision path starts from the beginning node called the root and to the leaves which give the decision.
- Frames—which are used when information units are characterised by many important features. The structure of a simple frame contains three different lines: a heading with the frame name, a pointer to another frame with appropriate relation and slots defining attribute names and values.
- Semantic networks—which capture knowledge in form of a graph in which nodes represent pieces of information (objects, concepts or situations in the problem domain) and the arcs represent relations or associations between them.
- Artificial neural networks (ANNs)—which are inspired by neurons in the brain and have become a popular target representation for learning. The idea of ANN usage is to create a learning set which includes data characterised by input and output features. During training, ANNs create a model which is able to transform input features into output features of a data set.

Literature presents different methods of formal knowledge representation which have been successfully applied to support various decision problems. Joshi and Dutta [3] use rule-based method to recognise and simplify features in freeform surface models for automatic finite element mesh generation. Cakir et al. [4] apply

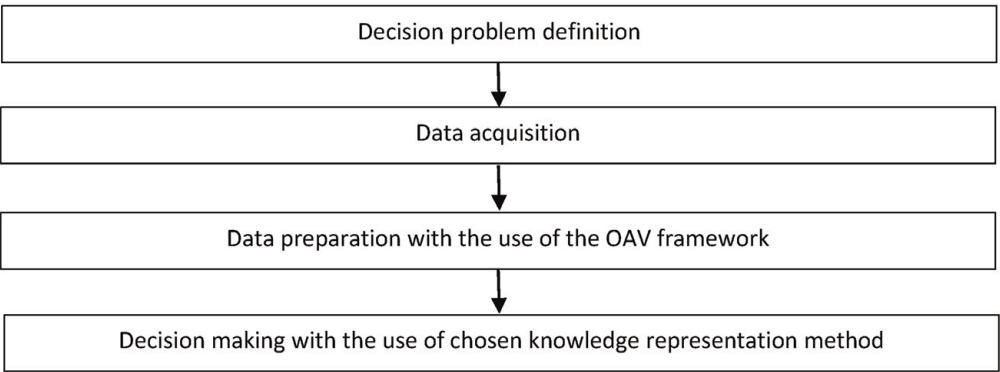


Figure 1.
A data analysis schema.

rule-based expert system for die and mould making. According to Trabelsi et al. [5], decision trees are recognised as effective and efficient machine learning approaches which have been successfully applied to solve real-world problems [5].

2. Data preparation for decision problem solving

One of the phases of knowledge acquisition processes based on a learning data set is the feature selection phase [6].

Obtaining data and selecting features for a given problem can be supported by means of the quality function deployment (QFD) method. QFD was developed to link product users and engineering attributes [7, 8]. QFD can be used for attribute identification not only in product development but also for decision support.

QFD helps to build a partnership between decision-makers and experts who are able to solve a given decision problem.

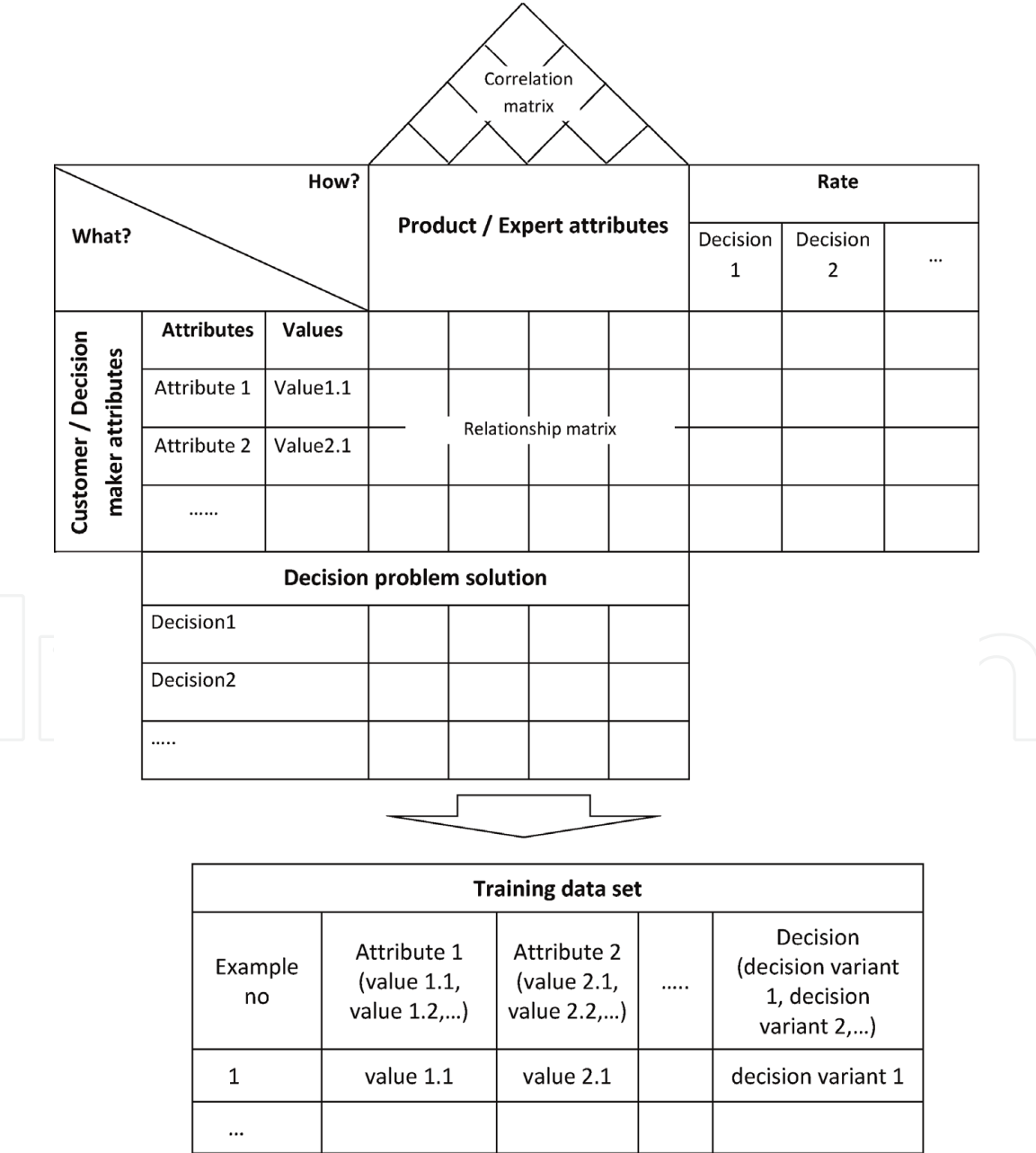


Figure 2. QFD for DSS attribute identification and training data set development.

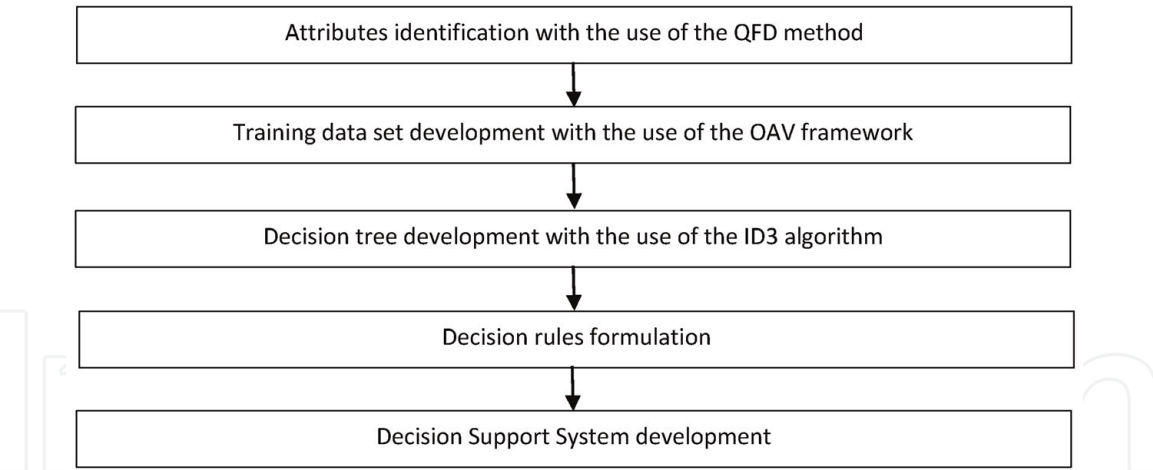


Figure 3.
 Decision-making with the use of chosen knowledge representation methods.

The structure of the QFD matrix was presented in **Figure 2**. The attributes identified in the QFD matrix can be used for obtaining data and developing a training data set.

Decision-making can be supported with the use of an algorithm presented in **Figure 3**.

3. Decision tree development with the use of the ID3 algorithm

Decision tree is a commonly used decision support tool. It is built with the use of training data set which contains examples of a decision problem solved in the past, characterised with the use of the OAV framework. A decision tree example is presented in **Figure 4** [9].

A proper decision tree uses meaningful attributes in the decision-making process. In the presented approach, decisions are generated by a decision tree built with the use of the ID3 (Iterative Dichotomiser 3) algorithm, which helps to feature selection to avoid keeping too many or too few attributes than is appropriate [10–12]. The basic idea of the ID3 family algorithms is to develop decision trees by growing them from the most important attribute, which is on the root, and selecting the next best attribute for each new decision branch added to the tree [12].

The ID3 algorithm [13, 14] uses entropy, which is understood as a measure of the amount of uncertainty in the training data set and which is calculated according to Eq. (1):

$$I = \sum_{i=1}^n (-p_i \log_2 p_i) \tag{1}$$

where:
 p probability that an element from *i* class occurs
 Entropy of a given attribute is calculated according to Eq. (2):

$$I(C / A_k) = \sum_{j=1}^{M_k} p(a_k, j) \bullet \left[- \sum_{i=1}^N (p(c_i / a_{k,j}) \bullet \log_2 p(c_i / a_{k,j})) \right] \tag{2}$$

where *M_k* number of values taken by attribute *A_k*; *N* number of classes; *k* number of attributes; *p(a_k, j)* probability that *a_k* takes value *j*; *p(c_i/a_{k,j})* probability that class *c_i* occurs, when *a_k* = *j*.

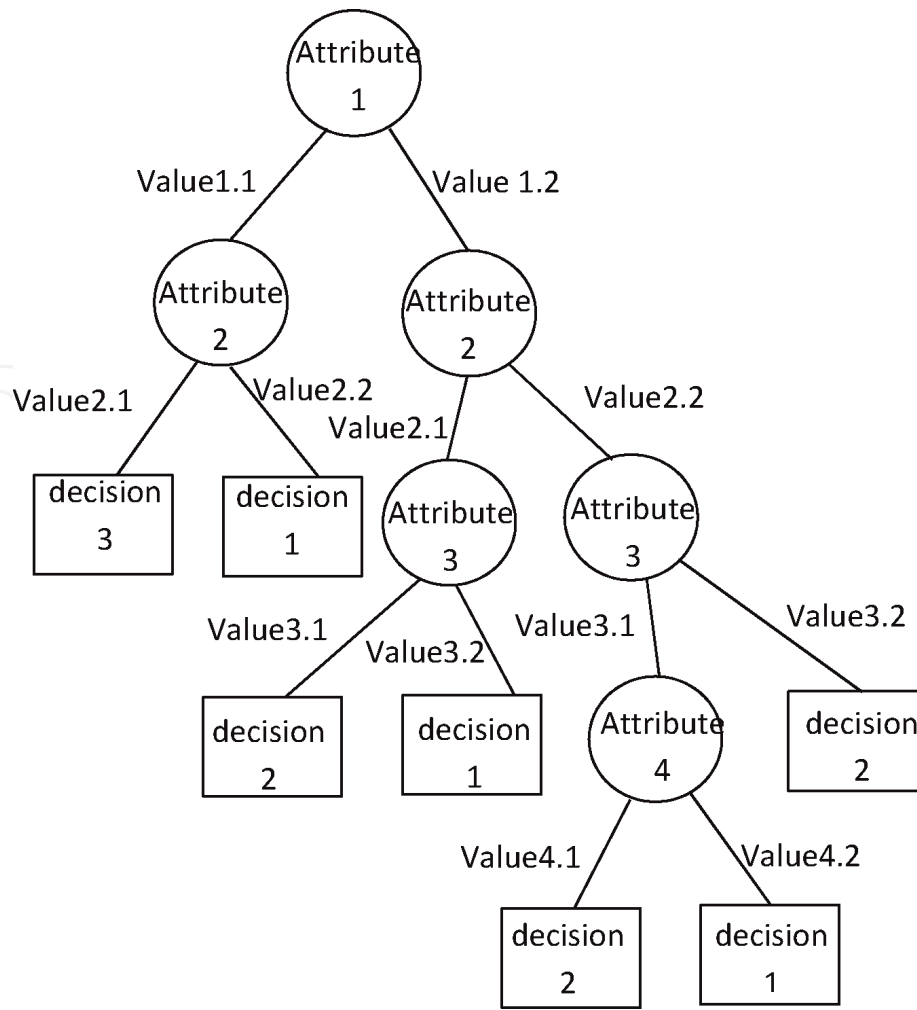


Figure 4.
A decision tree example.

Information gained is calculated according to Eq. (3):

$$\Delta I(A_k) = I - I(C/A_k) \quad (3)$$

A decision tree starts from the root node which is created with the use of the most meaningful attribute for which information gain is the highest. The next node in a decision tree is created with the use of the attribute for which information gain has a high value.

A pseudocode for ID3 algorithm includes the following stages:

- Calculating entropy for the training data set (Eq. 1).
- Calculating entropy for each attribute (Eq. 2).
- Calculating information gain for each attribute (Eq. (3)).
- Creating the root node in the decision tree with the attribute for which information gain is the highest.
- Creating decision tree branches using attribute and their values for which information gain is high.

- Classifying examples from the training data set, taking into consideration the given attribute and its values.
- If all cases for the given attribute and values lead unambiguously a decision, a decision leaf should be added at the end of the decision tree branch.
- If it is not possible to classify all cases from the training data set in a given decision tree branch to a decision, it is necessary to add a new node with the attribute which has high information gain.

The two last steps should be repeated until all branches in a decision tree will end with a decision leaf.

4. Decision rule formulation

4.1 Formulating and transforming rules

Basing on a decision tree, it is possible to formulate decision rules which consist of premises and a conclusion. The rule can be formulated as follows:

If *premise 1 and/or premise 2 and/or premise...* **then** *conclusion*.

where in the OAV framework *premise* is expressed as *attribute... = value...conclusion* is expressed as *decision = decision variant...*

Each premise consists of an attribute (a decision tree node) and a value (a decision tree branch) which create the given path (branch sequence) in a decision tree. A conclusion is understood as a decision leaf at the end of a given path in the decision tree.

The rules directly coming from a decision tree transform each decision tree branch into a given rule, so the obtained set of rules can include some rules which consist of different premises and the same conclusions. In such a case, it is necessary to transform the rules.

The rules which come directly from a decision tree use the operator “and” between the premises. Rule transformation focuses on conjoint premises with the same conclusions and develops a complex rule with the use of the operator “or” between the premise sequences from a given decision tree path. The algorithm for rule formulation is presented in **Figure 5**.

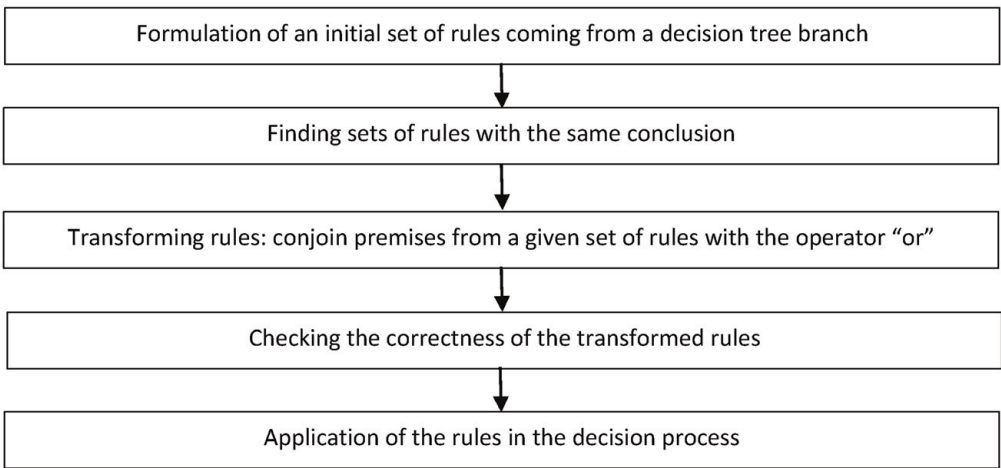


Figure 5.
Rule formulation algorithm.

4.2 An example of decision rule transformation

In the example, a set of initial rules comes from the decision tree presented in **Figure 4** and includes seven rules:

Rule 1:

If attribute 1 = value 1.1 **and** attribute 2 = value 2.1 **then** decision = decision 3.

Rule 2:

If attribute 1 = value 1.1 **and** attribute 2 = value 2.2 **then** decision = decision 1.

Rule 3:

If attribute 1 = value 1.2 **and** attribute 2 = value 2.1 **and** attribute 3 = value 3.1 **then** decision = decision 2.

Rule 4:

If attribute 1 = value 1.2 **and** attribute 2 = value 2.1 **and** attribute 3 = value 3.2 **then** decision = decision 1.

Rule 5:

If attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.1 **and** attribute 4 = value 4.1 **then** decision = decision 2.

Rule 6:

If attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.1 **and** attribute 4 = value 4.2 **then** decision = decision 1.

Rule 7:

If attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.2 **then** decision = decision 2.

The set of rules has to be transformed into complex rules with the use of the operator “or”. The number of complex rules is equal to the number of decision variants.

Rule 1:

If (attribute 1 = value 1.1 **and** attribute 2 = value 2.2) **or** (attribute 1 = value 1.2 **and** attribute 2 = value 2.1 **and** attribute 3 = value 3.2) **or** (attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.1 **and** attribute 4 = value 4.2) **then** decision = decision 1.

Rule 2:

If (attribute 1 = value 1.2 **and** attribute 2 = value 2.1 **and** attribute 3 = value 3.1) **or** (attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.1 **and** attribute 4 = value 4.1) **or** (attribute 1 = value 1.2 **and** attribute 2 = value 2.2 **and** attribute 3 = value 3.2) **then** decision = decision 2.

Rule 3:

If attribute 1 = value 1.1 **and** attribute 2 = value 2.1 **then** decision = decision 3.

4.3 Checking the completeness of the rules

Rule completeness can be checked with the use of a dependence network which consists of the components presented in **Figure 6** [13].

An example of a dependence network is presented in **Figure 7**.

Formal correctness of the rules can be verified by a dependence network, but substantial correctness can be verified by using the rules in the knowledge-based system (KBS).

KBS is essentially composed of two subsystems:

- The knowledge base
- The inference engine

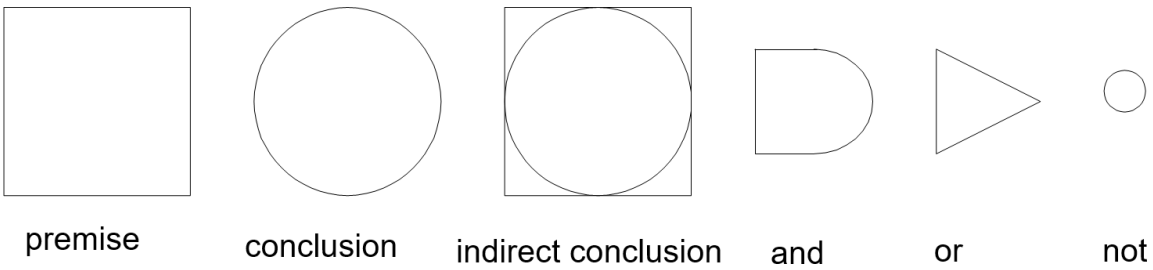


Figure 6.
Components of a dependence network.

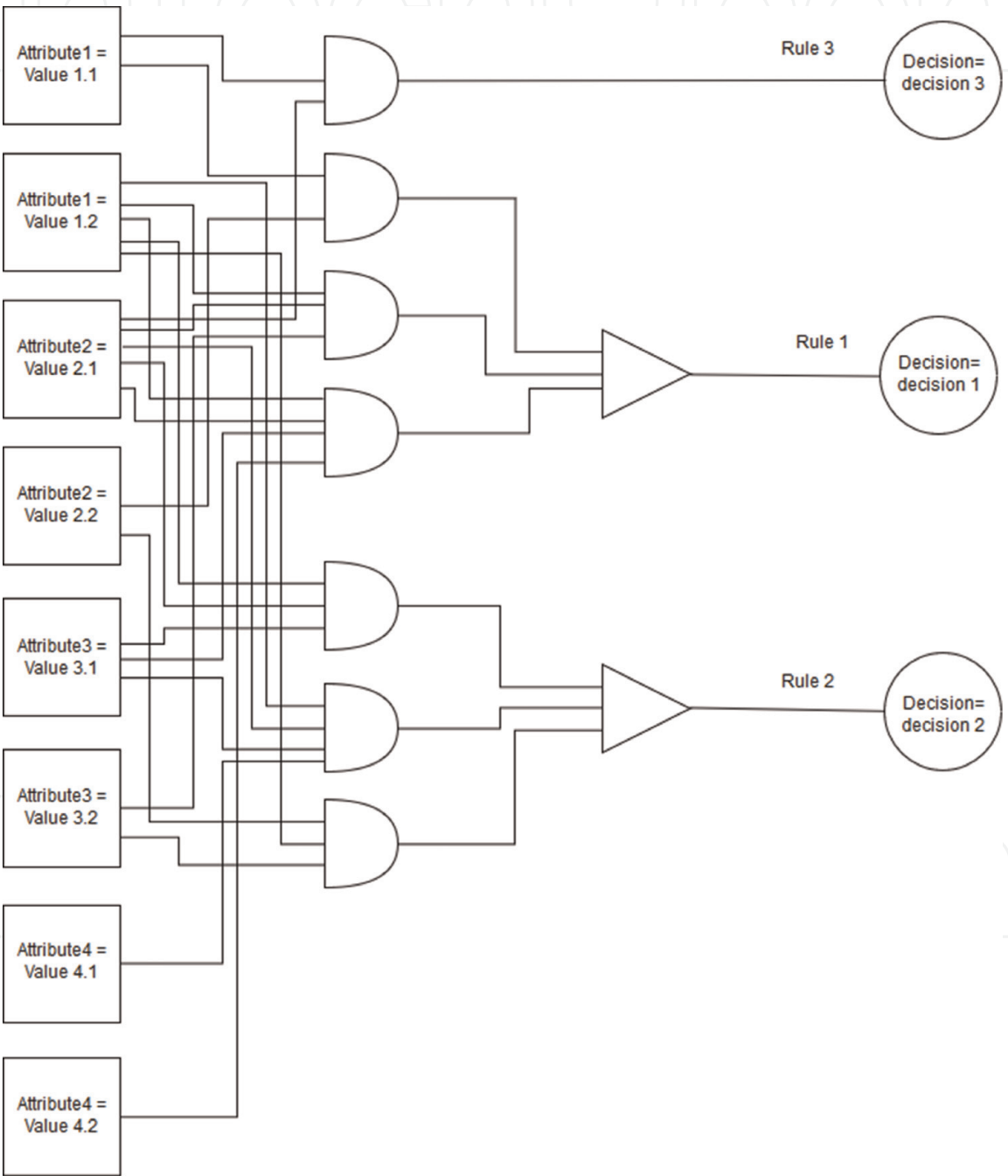


Figure 7.
An example of a dependence network.

The knowledge base consists of rules, whereas the inference engine is an automated reasoning system that evaluates the current state of the knowledge base, applies relevant rules and then asserts new knowledge into the knowledge base.

There are primarily two modes for the inference engine:

- Forward chaining
- Backward chaining

In case of backward chaining, the system looks at possible conclusions and works backwards to see if they might be true.

Forward chaining starts with the available data and uses inference rules to extract more data until a goal is reached.

Software tools for KBS include:

- Programming languages (e.g. Pascal, C++, Java, LISP, Prolog, etc.)
- Expert system shells (e.g. Sphinx)

5. An example

The decision problem analysed in the example is to find the right component—bearing to the toothed gear. The QFD matrix for a decision problem is presented in **Figure 8**.

The data from the matrix presented in **Figure 8** creates the first record in the training data set presented in **Table 1**. The data from the matrix was transformed according to the schema presented in **Figure 9**.

The training data set was presented in **Table 1**.

The meaning of the attributes was set with the use of the ID3 algorithm (**Table 2**).

An example of information gain calculation for A_k = principal dimensions (Eqs. 4–6):

$$I = 3/8 \times \text{LOG}_2 3/8 - 5/8 \times \text{LOG}_2 5/8 = 0,954. \quad (4)$$

$$\begin{aligned} I(C/A_k) &= 3/8 \times (-2/3 \times \text{LOG}_2 2/3 - 1/3 \times \text{LOG}_2 1/3) + 3/8 \\ &\times (-2/3 \times \text{LOG}_2 2/3 - 1/3 \times \text{LOG}_2 1/3) + 2/8 \\ &\times (-1/2 \times \text{LOG}_2 1/2 - 1/2 \times \text{LOG}_2 1/2) = 0,939. \end{aligned} \quad (5)$$

$$\Delta I(A_k) = 0,015. \quad (6)$$

The decision tree for the training data set in the example was presented in **Figure 10**.

Decision rule induction based on the decision tree presented in **Figure 10** is as follows:

Rule 1:

If basic dynamic load rating = small **then** decision = w1.

Rule 2:

If basic dynamic load rating = big **and** basic static load rating = small **then** decision = w2.

Rule 3:

If basic dynamic load rating = big **and** basic static load rating = big **and** delivery time = long **then** decision = w2.

Rule 4:

If basic dynamic load rating = big **and** basic static load rating = big **and** delivery time = short **and** price = high **then** decision = w1.

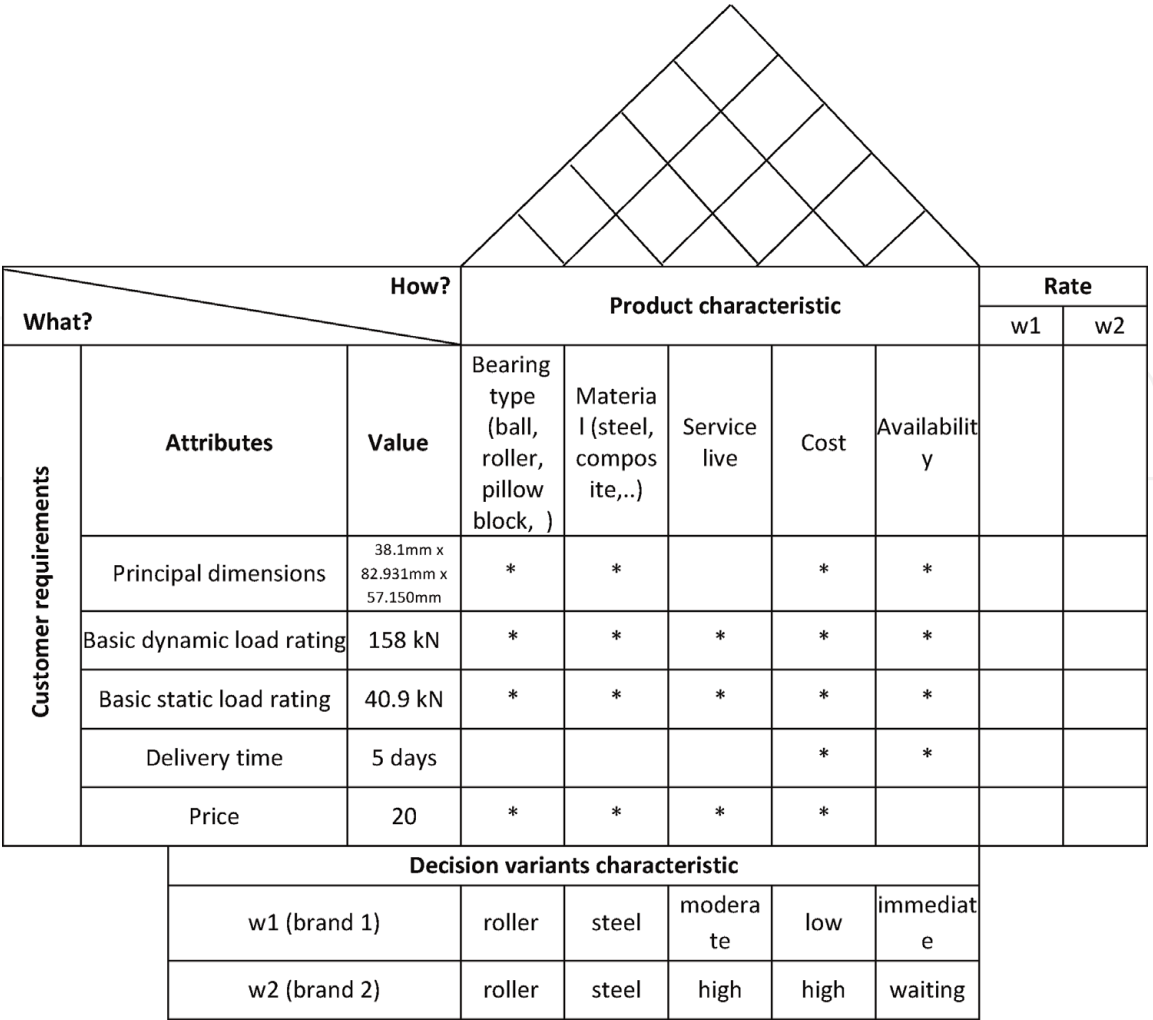


Figure 8.
An example of the QFD matrix.

No	Attribute/(value 1, value 2,...)					Decision
	Principal dimensions (small, medium, big)	Basic dynamic load rating (small, big)	Basic static load rating (small, big)	Delivery time (short, long)	Price (high, low)	
1	Small	Big	Small	Short	High	w2
2	Medium	Big	Big	Short	High	w1
3	Medium	Big	Big	Short	Low	w2
4	Small	Small	Big	Long	High	w1
5	Medium	Small	Small	Short	Low	w1
6	Big	Big	Small	Long	High	w2
7	Big	Small	Big	Long	Low	w1
8	Small	Big	Big	Long	Low	w1

Table 1.
A training data set example.

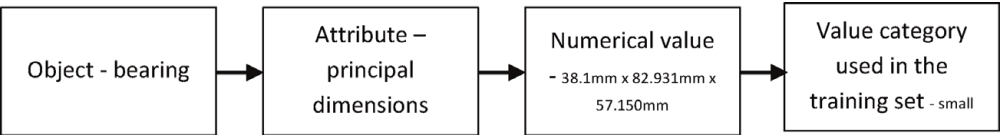


Figure 9.
An example of the OAV framework for a training set.

Entropy of the system I	Entropy of the attribute $I(C/A_k)$				
	Principal dimensions	Basic dynamic load rating (BDLR)	Basic static load rating (BSLR)	Delivery time (DT)	Price
0.954	0.939	0.607	0.796	0.906	0.906
Information gain $I(A_k)$					
	0.015	0.347	0.158	0.048	0.048
Attribute importance (ranking)					
	5	1	2	3	4

Table 2.
Information gain calculation.

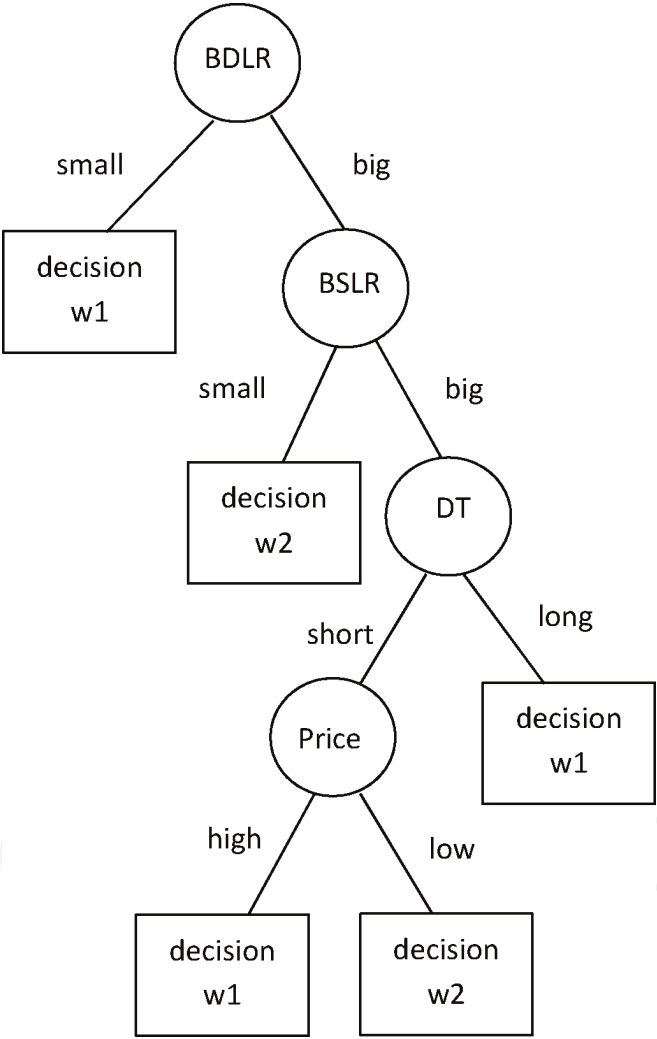


Figure 10.
An exemplary decision tree.

Rule 5:
If basic dynamic load rating = big and basic static load rating = big and delivery time = short and price = low then decision = w2.
The transformed rules:
Rule 1:
If (basic dynamic load rating = small) or (basic dynamic load rating = big and basic static load rating = big and delivery time = short and price = high) then decision = w1.

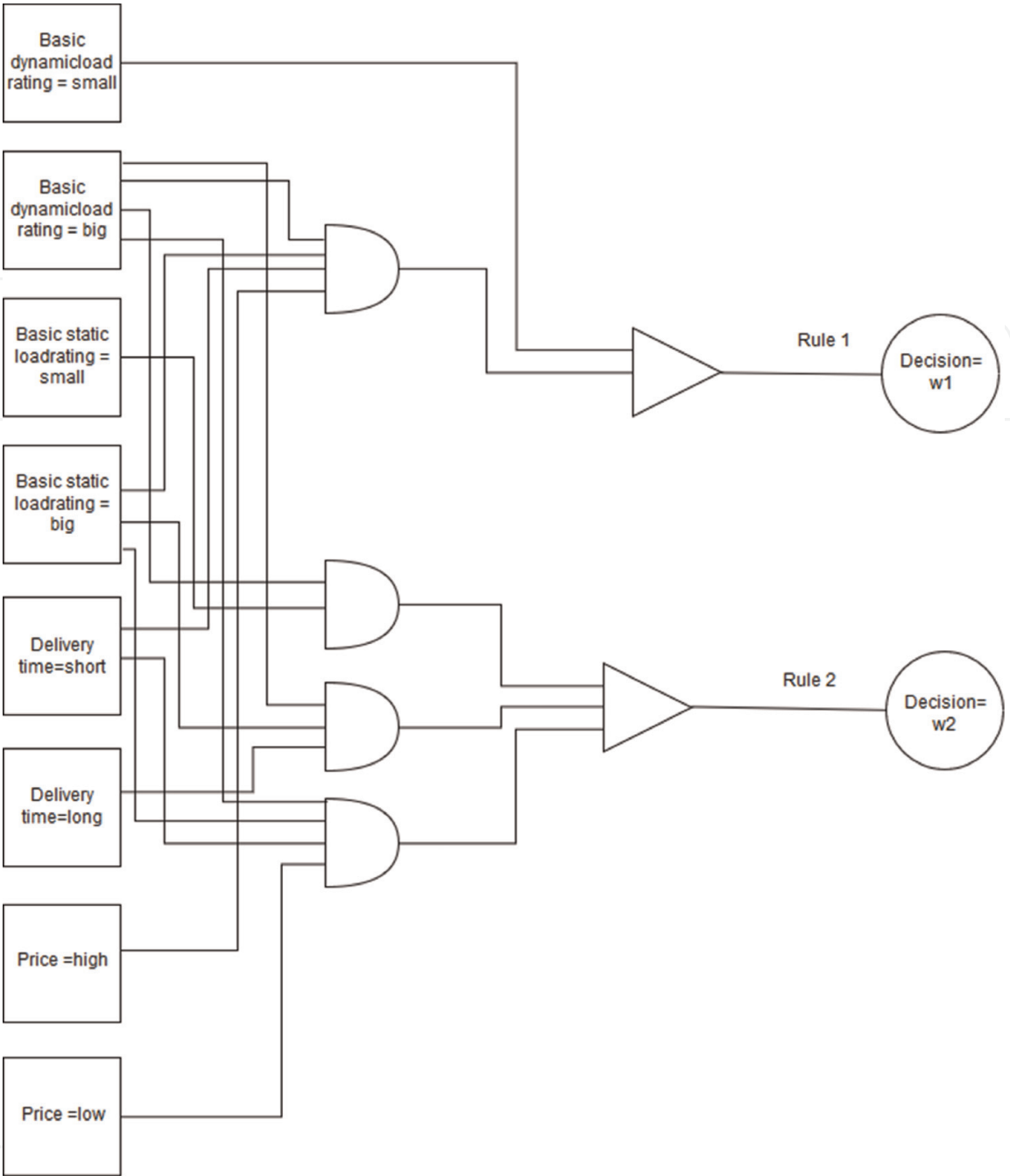


Figure 11.
Dependence network.

Rule 2:

If (basic dynamic load rating = big **and** basic static load rating = small) **or** (basic dynamic load rating = big **and** basic static load rating = big **and** delivery time = long) **or** (basic dynamic load rating = big **and** basic static load rating = big **and** delivery time = short **and** price = low) **then** decision = w2.

The dependence network for checking the transformed rules was presented in **Figure 11**.

6. Conclusions

The decision-making process in enterprises needs knowledge, information and data. A decision-maker's knowledge is developed based on past experiences and can be supported by means of artificial intelligence methods, such as rules which are

the widely used methods of knowledge representation. Decision trees are an efficient method of rule formulation. This chapter presents the methodology of decision tree induction with the Iterative Dichotomiser 3 algorithm, as well as rule formulation and checking with the use of dependence network.

A useful method for data analysis in the decision process is Quality Function Deployment, which organises data both from the customer and expert perspectives. The data was analysed with the object-attribute-value framework in which attributes and their values are used for building a training data set.


An example related to bearing selection was presented to illustrate the decision process.

Author details

Izabela Kutschenreiter-Praszkiewicz
University of Bielsko-Biała, Bielsko-Biała, Poland

*Address all correspondence to: ipraszkiewicz@ath.bielsko.pl

IntechOpen

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Iyer N, Jayanti S, Lou K, Kalyanaraman Y, Ramani K. Three-dimensional shape searching: State-of-the-art review and future trends. *Computer-Aided Design*. 2005;**37**(5): 509-530
- [2] Li X, Zhang S, Huang R, Huang B, Xu C, Zhang Y. A survey of knowledge representation methods and applications in machining process planning. *The International Journal of Advanced Manufacturing Technology*. 2018;**98**:3041-3059
- [3] Joshi N, Dutta D. Feature simplification techniques for freeform surface models. *Journal of Computing and Information Science in Engineering*. 2003;**3**(3):177-186
- [4] Cakir MC, Irfan O, Cavdar K. An expert system approach for die and mold making operations. *Robotics and Computer-Integrated Manufacturing*. 2005;**21**(2):175-183
- [5] Trabelsi A, Elouedi Z, Lefevre E. Decision tree classifiers for evidential attribute values and class labels. *Fuzzy Sets and Systems*. 2019; **366**:46-62
- [6] Moczulski W, Szulim R, Tomasik P, Wachla D. Knowledge discovery in databases. In: Korbicz J, Kościelny JM, editors. *Modeling, Diagnostics and Process Control*. Berlin, Heidelberg: Springer; 2010. pp. 119-152
- [7] Chen W, Hoyle C, Wassenaar H. *Decision-Based Design*. London: Springer-Verlag; 2013. pp. 13-34. DOI: 10.1007/978-1-4471-4036-82
- [8] Molloy O, Warman EA, Tilley S. *Design for Manufacturing and Assembly, Concepts, Architectures and Implementation*. USA: Springer; 1998. DOI: 10.1007/978-1-4615-5785-2
- [9] Kutschenreiter-Praszkiewicz I. Graph-based decision making in industry. In: Sirmacek B, editor. *Graph-Based Decision Making in Industry*. IntechOpen; 2018
- [10] Zhao X, Deng W, Shi Y. Feature selection with attributes clustering by maximal information coefficient. *Procedia Computer Science*. 2013;**17**: 70-79
- [11] Mehrotra L, Saxena PS, Doohan N. Implementation of modified ID3 algorithm. In: Mishra DK et al., editors. *Information and Communication Technology for Sustainable Development. Lecture Notes in Networks and Systems 9*. Springer Nature Singapore Pte Ltd; 2018. DOI: 10.1007/978-981-10-3932-4_6
- [12] Xu M, Wang J-L, Chen T. Improved decision tree algorithm: ID3+. In: Huang D-S, Li K, Irwin GW, editors. *Intelligent Computing in Signal Processing and Pattern Recognition*. Vol. 345. Berlin Heidelberg: Springer-Verlag; 2006. pp. 141-149
- [13] Mulawka J. *Sysytemy Ekspertowe*. Warszawa: WNT; 1997
- [14] Quinlan JR. Induction of decision trees. *Machine Learning*. 1986;**1**:81-106