computing@computingonline.net
www.computingonline.net

ISSN 1727-6209
International Journal of Computing

COMBINING BAYESIAN NETWORKS AND ROUGH SETS: FURTHER STEP TOWARDS REASONING ABOUT UNCERTAINTY

Janusz Zalewski [1], Sławomir T. Wierzchoń [2], Henry L. Pfister [3]

[1] Florida Gulf Coast University, Ft. Myers, FL 33965, USA, zalewski@fgcu.edu, http://www.fgcu.edu/zalewski/
[2] Polish Academy of Sciences, Ordona 21, 01-237 Warsaw, Poland, and University of Gdańsk, 80-953 Gdańsk, Poland, stw@ipipan.waw.pl, http://www.ipipan.waw.pl/staff/s.wierzchon/
[3] Air Force Research Lab, Eglin AFB, FL 32542, USA, henry.pfister@eglin.af.mil, http://www.reef.ufl.edu/Faculty%20pages/Old%20faculty/Pfister/Pfister.htm

Abstract: *This paper discusses a combination of Bayesian belief networks and rough sets for reasoning about uncertainty. The motivation for this work is the problem with assessment of properties of software used in real-time safety-critical systems. A number of authors applied Bayesian networks for this purpose, however, their approach suffers from problems related to calculating the conditional probability distributions, when there is scarcity of experimental data. The current authors propose enhancing this method by using rough sets, which do not require knowledge of probability distributions and thus are helpful in making preliminary evaluations, especially in real-time decision making. The combination of Bayesian network and rough sets tools, Netica and Rosetta, respectively, is used to demonstrate the applicability of this method in a case study of the Australian Navy exercise.*

Keywords: *Bayesian belief networks, rough sets, approximate reasoning, software safety, safety analysis.*

1. INTRODUCTION

Bayesian Belief Networks (BBN's) have been widely used in solving various computational problems with insufficient information and uncertainty. Some of these applications are briefly reviewed in [1], and most of them are collected in the bibliography [2]. Although, in general, BBN's have been very effective, because they allow reasoning and making predictions based on small sets of probabilities with backwards inference, they are still based on probability theory. A significant disadvantage of BBN's is that, in realistic cases, they require extensive computations of the conditional probability values. In most of the previous studies, it has been recognized that this is one of the method's major limitations.

With this in mind, one wants to look at a complementary method of evaluating data in the input data set, which would not rely strictly on probability densities. One of the theories that offer such an approach, with values of data attributes and events measured by likelihoods rather than probabilities, is the rough sets theory [3-4]. Rough sets have been used since the early eighties in a wide range of industries to reason about uncertainty. The most recent discussions of these applications can be found in [5] and in the rough sets bibliography [6].

In a previous paper [1], we gave an outline of the combination of using BBN's and rough sets in decision making under uncertainty, and suggested the enhancement of pure Bayesian reasoning by additional use of rough sets for preliminary evaluation of data. The objective of the current paper is to extend the original concept by using specific techniques to evaluate the missing values in the reasoning process [7]. The paper is structured as follows. In Section 2, the motivation for this project is outlined, which is the automatic assessment of certain critical software properties. Section 3 presents elementary information about rough sets, and Section 4 gives an overview of the method developed for decision making under uncertainty with the use of BBNs and rough sets. General conclusions are derived in Section 5.

2. MOTIVATION: SAFETY-CRITICAL SOFTWARE ASSESSMENT

In recent years, some of the present authors have dealt with various aspects of assessing software quality in real-time safety-critical applications [8-9]. The basic idea for the current project comes from multiple previous attempts to assess various software properties in critical applications. They are briefly outlined below.

A. USE OF BBN'S TO ASSESS SOFTWARE QUALITY

In one of the first studies reported [10], Neil and Fenton addressed the eternal question: "Can we predict the quality of our software *before* we can use it?", by applying BBN's to assess the *defect density* as a measure of software quality. A simplified diagram from their study is presented in Figure 1. The nodes were built based on the understanding of life-cycle processes, from requirements specification through testing.

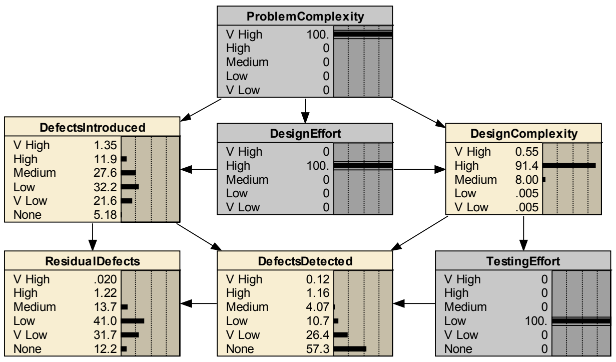

ProblemComplexity: V High 100., High 0, Medium 0, Low 0, V Low 0
DefectsIntroduced: V High 1.35, High 11.9, Medium 27.6, Low 32.2, V Low 21.6, None 5.18
DesignEffort: V High 0, High 100., Medium 0, Low 0, V Low 0
DesignComplexity: V High 0.55, High 91.4, Medium 8.00, Low .005, V Low .005
ResidualDefects: V High .020, High 1.22, Medium 13.7, Low 41.0, V Low 31.7, None 12.2
DefectsDetected: V High 0.12, High 1.16, Medium 4.07, Low 10.7, V Low 26.4, None 57.3
TestingEffort: V High 0, High 0, Medium 0, Low 100., V Low 0, None 0

Fig. 1 – Simplified BBN assessing software density

The probabilities of respective states were based on the analysis of literature and common-sense assumptions about the relations between variables. The node variables are shown on histograms of the predictions obtained by execution of the network after the evidence entered (the evidence is represented by nodes with probabilities equal to 1.0). As the authors say, the advantage of their model is that it "provides a way of simulating different events and identifying optimum courses of action based on uncertain knowledge.

B. BBN'S IN THE ASSESSMENT OF SOFTWARE SAFETY

Dahll and Gran [11] applied BBN's to address safety assessment of software for acceptance purposes, in a more comprehensive way, using multiple information sources, such as complexity, testing, user experience, system quality, etc. Their BBN network for system quality, which is only a part of the entire model, is shown in Figure 3. It involves two root nodes: *UserExperience* and *VendorQuality*, and a number of leaf nodes, corresponding to observable variables, of which *QualityMeasures* is of particular importance. This node shows evidence about the system quality, grouping quality attributes, such as readability, structuredness, etc., and can be expanded further.

Other observable variables include *Failures-InOther Products*, those related to the user experience (*NoOfProducts* and *TotalUseTime*), as well as those related to quality assurance policy. When evidence becomes available, entering respective observation data into these nodes and executing the network provides assessment of the variable in question, which in this case is *SystemQuality.*

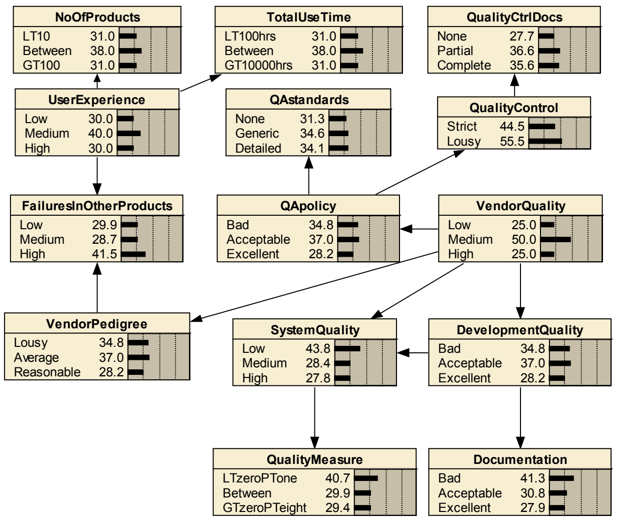

NoOfProducts: LT10 31.0, Between 38.0, GT100 31.0
TotalUseTime: LT100hrs 31.0, Between 38.0, GT10000hrs 31.0
QualityCtrlDocs: None 27.7, Partial 36.6, Complete 35.6
UserExperience: Low 30.0, Medium 40.0, High 30.0
QAstandards: None 31.3, Generic 34.6, Detailed 34.1
QualityControl: Strict 44.5, Lousy 55.5
FailuresInOtherProducts: Low 29.9, Medium 28.7, High 41.5
QApolicy: Bad 34.8, Acceptable 37.0, Excellent 28.2
VendorQuality: Low 25.0, Medium 50.0, High 25.0
VendorPedigree: Lousy 34.8, Average 37.0, Reasonable 28.2
SystemQuality: Low 43.8, Medium 28.4, High 27.8
DevelopmentQuality: Bad 34.8, Acceptable 37.0, Excellent 28.2
QualityMeasure: LTzeroPTone 40.7, Between 29.9, GTzeroPTeight 29.4
Documentation: Bad 41.3, Acceptable 30.8, Excellent 27.9

Fig. 2 – BBN for the system quality in safety assessment

The authors note, however, that their example is intended more as an illustration of the method rather than as a real attempt to compute the quality of the system. Their probability assignments to the node variables were chosen somewhat *ad hoc*, and not based on any deeper analysis of the problem. However, as the authors say in conclusion, the results of the study were positive and showed "that the method reflects the way of an assessor's thinking during the assessment process."

C. DEPENDABILITY AND RELIABILITY ASSESSMENT

Delic et al. [12] used BBNs to formalize reasoning about software dependability to facilitate the software assessment process. They constructed a network for evaluating dependability of a software-based safety system. It used the data associated with two primary assumptions: the excellence in development (called a process argument) and failure-free statistical testing (called a product argument). The network topology includes taking into consideration variables such as: Test Failures, Operational Failures, Initial Faults, Faults Found, Faults Delivered, and System PFD (Probability of Failure per Demand). The probability distributions have been derived from a sample of programs from an academic experiment.

The authors were interested in estimating the probabilities of failure during acceptance testing and during the operational life of the product (represented by two variables mentioned in previous paragraph), given the prior probabilities and observed events. In particular, positive results of an acceptance test allowed deriving numerical estimates about the PFD and operational performance of the product.

Helminen [13] used BBN's to attack the problem of software reliability estimation. His primary motivation to apply BBN's was that they allow all possible evidence (large number of variables, different potential sources, etc.) to be used in the analysis of the reliability of a safety-critical system. The essential characteristic of such systems is that they involve a significant number of variables related to reliability, with very limited evidence.

The reliability of such systems is modeled as a *probability of failure*, that is, the probability that the programmable system fails when it is required to operate correctly. To develop an estimate of probability of failure, the authors built a series of BBN models, using evidence from such sources, as the system development process, system design features, and pre-testing, before the system is deployed. This is later enhanced by data from testing and operational experience.

The essential part of this work was building BBN models for various operational profiles for multiple test cycles, involving continuous probability distributions. As a result, using software combining Bayesian inference with Gibbs sampling, via Markov chain Monte Carlo (MCMC) simulation, it was possible to estimate, how many tests had to be run for a single system in a particular operational environment to achieve certain level of reliability. To decrease the huge number of necessary tests, multiple operational profiles for the same system were used, which required building replicated BBN models to include other profiles' evidence. In essence, by expanding the BBN models further, this approach also allows reliability estimation over the entire lifespan of the software, but respective experiments have not been conducted in this study.

3. ROUGH SETS: AN INTRODUCTION

Since there are essentially no statistical data for making the types of assessments discussed in the previous section, applying BBN's to reason about software properties based on limited information available from experiments is problematic because of the necessity to calculate conditional probability distributions. To deal with this problem, we are proposing the use of rough sets [1]. In this section, a brief introduction to rough sets theory is given.

A. BACKGROUND

Rough Set theory was invented by Zdzisław Pawlak to cope with limited perception of the surrounding world. The theory is especially helpful in dealing with vagueness and uncertainty in decision situations. Its main purpose is the "automated transformation of data into knowledge" [4]. The data are perceived in terms of objects and their features, i.e., values of the attributes used to characterize these objects. The knowledge deduced from these data is expressed in terms of surely and possibly certain statements describing notions of interests. More formally, such descriptions can be divided into so-called *lower* and *upper* approximations of entire notions. Below, we describe a qualitative procedure containing all steps needed to form appropriate description of the concepts under consideration.

We start from a relational database, i.e., a table with rows corresponding to objects and columns corresponding to the attributes. Each entry of the table represents attribute value of a corresponding object (i.e., its feature). In a rough set formalism, the database is considered as an information system, i.e., a quadruple $IS = (U, A, V, f)$, where:

- $U = \{u_1, \dots, ,u_n\}$ stands for a (usually discrete) set of objects
- $A = \{a_1, \dots, ,a_m\}$ is a set of attributes
- $V = V_1 \cup \dots \cup V_m$, where V_i is the domain of an i-th attribute, and
- f: $U \times A \rightarrow V$ is a so-called information function providing the description of objects, that is, $f(u_i, a_j)$ assigns a value of j-th attribute to i-th object.

Usually the set A is decomposed into two disjoint subsets $A = C \cup D$ and the attributes from C are used to characterize objects and form so-called *condition* attributes, while the attributes in D are so-called *decision* attributes and they are used in decision-making or classification tasks.

The above mentioned concepts are illustrated in Table 1, in which:

$U = \{u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8\}$,

$A = \{a_1, a_2, a_3\}$, and

$V = V_1 \cup V_2 \cup V_3$, = {Low, Med, High} $\cup$ {Min, Under, Over, Max} $\cup$ {yes, no}.

Because of the limited knowledge, we cannot fully discern objects, i.e., there are such objects u, v in U that $f(u, c) = f(v, c)$ for all the condition attributes c. This fact leads to the notion of *indiscernibility* relation E being in fact an equivalence relation on U. For example, for the information system in Table 1, objects u_2 and u_8 are indiscernible. So are objects u_5 and u_7.

It appears that in many cases we can identify proper subsets C' of C such that the indiscernibility relation $\mathsf{E}_{C'}$ induced by the attributes in C' is identical with the original relation E. Such sets of attributes are called *reducts*. Existence of reducts proves that not all of the attributes are necessary to form the equivalence classes. In other words, identifying reducts allows more economic description of objects as we need smaller number of descriptors (features) to characterize these objects. Unfortunately, from a computational point of view this is an NP-hard task. No such reducts exist for the example shown in Table 1.

Table 1. Example of an information system.

f: $U \times A \to V$	**Condition attributes C**		**Decision attr. D**
Obj. U	a_1	a_2	a_3
u_1	Low	Max	yes
u_2	Low	Min	no
u_3	Med	Under	no
u_4	Med	Under	yes
u_5	High	Over	no
u_6	Low	Over	yes
u_7	High	Over	no
u_8	Low	Min	no

B. DEFINITION OF A ROUGH SET

Now we are ready to introduce the key concepts of rough set theory. Let B be a subset of the condition attributes and let $[v]_B$ stand for an equivalence class, i.e., a set of objects u in U with identical description (narrowed to the set B) as the object v. The subset X of U can be characterized using information contained in B by means of so-called *B-lower* and *B-upper* approximations defined as

$$B(X)_* = \{u \in U | [u]_B \subseteq X\} \qquad (1a)$$
$$B(X)^* = \{u \in U | [u]_B \cap X \neq \varnothing\} \qquad (1b)$$

The lower approximation of X is the collection of objects which can be viewed *surely* as members of the set X, while the upper approximation of X is the collection of objects that *possibly* are members of X. Obviously $B(X)_* \subseteq B(X)^*$. If $B(X)_* = B(X)^*$ we say that X is B-definable and otherwise it is only partially definable. The set $BN_B = B(X)^* - B(X)_*$ is called *B-boundary* region; it specifies the objects that cannot be classified with certainty to be neither inside X, nor outside X.

There are many grades of partial definability. We say that the set X is [14]:

- roughly B-definable iff $B(X)_* \neq \varnothing$ & $B(X)^* \neq U$
- internally B-indefinable iff $B(X)_* = \varnothing$, $B(X)^* \neq U$
- externally B-indefinable iff $B(X)_* \neq \varnothing$, $B(X)^* = U$
- totally B-indefinable iff $B(X)_* = \varnothing$, $B(X)^* = U$.

Obviously, if $B = C$, i.e., the full set of condition attributes is used, we omit the prefix B- in all above definitions. In such a case a set X is characterized by the pair (X_*, X^*) and we say that X is a rough set (or B-rough set).

To get a numerical characterization of the "roughness" of a set X we introduce a so-called accuracy of approximation [14]

$$\alpha_{B(X)} = |B(X)_*| / |B(X)^*| \qquad (2)$$

where the symbol $|Y|$ stands for the cardinality of the set Y. X is said to be crisp (or precise) with respect to the set of attributes B iff $\alpha_{B(X)} = 1$, and otherwise X is said to be rough (or vague) with respect to B.

Another characterization of the set of objects can be gained by introducing so-called rough membership function $\mu_{B,X}$: $U \to [0,1]$ defined as follows [14]

$$\mu_{B,X}(x) = |[x]_B \cap X| \backslash |[x]_B| \qquad (3)$$

With such a definition a relationship between rough and fuzzy sets theory is established. Further, we can relax the definitions of the lower and upper approximation, namely

$$B_\beta(X)_* = \{u \in U | \mu_{B,X}(x) \geq \beta\} \qquad (4a)$$
$$B_\beta(X)^* = \{u \in U | \mu_{B,X}(x) > 1\text{-}\beta\} \qquad (4b)$$

where $0 \leq \beta \leq 1$. If $\beta = 1$ we obtain original definitions (1a) and (1b).

C. ROUGH RULES

Note that in practical applications of interest are the sets of objects with identical set of decision attributes, that is, we define X as the set of objects satisfying the equality $f(x_1, d) = f(x_2, d)$ for all attributes d in D. If, for example, D is a set of diseases then X is a set of persons suffering on particular disease, and the equivalence classes $[x]_B$ contain patients with identical symptoms (restricted to the set B). Hence, it is natural to find such condition attributes which can be used to discriminate between different diseases. This leads us to the practical aspects of rough set theory: rough rules.

The already mentioned process of "transformation of data into knowledge" translates now into refining the dependencies between sets of attributes. Intuitively, if C and D are two sets of attributes, we say that D depends totally on C if all values of the attributes from D are uniquely determined by values of attributes from C. This is functional dependency known from database theory.

Rough set theory enables relaxing this definition by introducing a dependency in a degree $k \in (0, 1]$. For details, please see [4] and [15]. There are at least two successful computer programs allowing rough data analysis: LERS [14] and Rosetta [16].

Finally, if a new object is introduced into the data set with the decision value missing, one could attempt to determine this value using previously generated rules.

D. HANDLING MISSING VALUES IN A ROUGH SET

Grzymala-Busse [7] describes several algorithms of dealing with missing values in information systems, based on three types of such values:

- those which are lost and no longer available
- totally irrelevant values, and
- partially relevant values.

They are marked in Table 2, using the following symbols: a question mark "?" for not available values, an asterisk "*" for irrelevant values, and a dash "-" for partially relevant values.

Table 2. Information system with missing values.

f: $U \times A \rightarrow V$	**Condition attributes C**		**Decision attr. D**
Obj. U	**a_1**	**a_2**	**a_3**
u_1	?	Max	yes
u_2	Low	Min	no
u_3	Med	Under	no
u_4	-	Under	yes
u_5	High	Over	no
u_6	Low	Over	yes
u_7	High	Over	no
u_8	Low	*	no

The trouble with such systems is that their information function, that assigns a value of j-th attribute to i-th object

$$f: U \times A \rightarrow V$$

is incompletely specified (partial), so the theory developed for total (complete) information functions does not apply here. In such case, however, the indiscernibility relation is replaced by a characteristic relation, and the entire process of calculating lower and upper approximations changes slightly, which is explained below for the information system in Table 2.

To calculate the approximations, one has to start with the meaning of the atomic formulas in a given information system. For the information system in Table 1, these meanings, called also *blocks* in [7], are as follows:

$||a_1 = \text{Low}|| = \{ u_1, u_2, u_6, u_8 \}$
$||a_1 = \text{Med}|| = \{ u_3, u_4 \}$
$||a_1 = \text{High}|| = \{ u_5, u_7 \}$
$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$
$||a_2 = \text{Under}|| = \{ u_3, u_4 \}$
$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7 \}$
$||a_2 = \text{Max}|| = \{ u_1 \}$

These sets have to be modified for an information system with missing values in Table 2, as follows. For the missing value of the attribute a_1, which is not available for object u_1 and marked "?", object u_1 has to be removed from all blocks created for this attribute, that is, block $||a_1 = \text{Low}||$ will change to:

$||a_1 = \text{Low}|| = \{ u_2, u_{6},, u_8 \}$

with two other blocks for a_1 remaining unchanged, because they do not include objects with lost value of a_1.

For the missing value of the attribute a_2, which is irrelevant and marked "*", its corresponding object, u_8, has to be included in blocks for all values of this attribute, which will lead to the following modifications:

$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$
$||a_2 = \text{Under}|| = \{ u_3, u_4, u_8 \}$
$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7, u_8 \}$
$||a_2 = \text{Max}|| = \{ u_1, u_8 \}$

Finally, for the missing value of the attribute a_1, which is marked "-", as partially relevant, respective object u_4 has to be added to the blocks containing objects corresponding to the decision attribute's value the same as the value of this decision attribute for the partially relevant value. In case of Table 2, the partially relevant value of attribute a_1 for object u_4 corresponds to the decision attribute's value "yes".

Thus, this attribute's value is relevant to this particular decision attribute, and this is the meaning of the term "partially relevant". Two other objects exist, which have "yes" as their decision attribute's value: u_1, whose value of attribute a_1 is unavailable, so we drop it from consideration, and u_6, whose value of a_1 equals *Low*; therefore u_4 has to be added to the block, which contains a_1 = Low, because it is partially relevant to a corresponding decision attribute.

So the final list of blocks looks as follows:

$||a_1 = \text{Low}|| = \{ u_2, u_{4},, u_{6},, u_8 \}$
$||a_1 = \text{Med}|| = \{ u_3, u_4 \}$
$||a_1 = \text{High}|| = \{ u_5, u_7 \}$
$||a_2 = \text{Min}|| = \{ u_2, u_8 \}$
$||a_2 = \text{Under}|| = \{ u_3, u_4, u_8 \}$
$||a_2 = \text{Over}|| = \{ u_5, u_6, u_7, u_8 \}$
$||a_2 = \text{Max}|| = \{ u_1, u_8 \}$

Because of the limited length of this paper, we can only mention here that for further computations the so called *characteristic sets* have to be calculated, for each object, which is done as follows:

1) The characteristic set ***K*** of an object is defined as an intersection of blocks for specific values of the attributes for this object.
2) If the value of an attribute is irrelevant "*" or unavailable "?", then the entire universe *U* is taken as a corresponding block for this attribute.
3) If the value of an attribute is partially relevant "-", then for this specific block it is substituted by a union of blocks representing particular values of the attributes for the corresponding decision attribute's value.

A more formal presentation of these concepts, with respective algorithms, is given in [7]. Below we present the computation of characteristic sets for the list of blocks corresponding to Table 2:

$$K_{u1} = U \cap \{ u_1, u_8 \} = \{ u_1, u_8 \}$$
$$K_{u2} = \{ u_2, u_4, u_6, u_8 \} \cap \{ u_2, u_8 \} = \{ u_2, u_8 \}$$
$$K_{u3} = \{ u_3, u_4 \} \cap \{ u_3, u_4, u_8 \} = \{ u_3, u_4 \}$$
$$K_{u4} = \{ u_2, u_4, u_6, u_8 \} \cap \{ u_3, u_4, u_8 \} = \{ u_4, u_8 \}$$
$$K_{u5} = \{ u_5, u_7 \} \cap \{ u_5, u_6, u_7, u_8 \} = \{ u_5, u_7 \}$$
$$K_{u6} = \{u_2, u_4, u_6, u_8\} \cap \{ u_5, u_6, u_7, u_8 \} = \{u_6,u_8\}$$
$$K_{u7} = \{ u_5, u_7 \} \cap \{ u_5, u_6, u_7, u_8 \} = \{ u_5, u_7 \}$$
$$K_{u8} = \{ u_2, u_4, u_6, u_8 \} \cap U = \{ u_2, u_4, u_6, u_8 \}$$

As explained in [7], computation of lower and upper approximations, depends on their definitions. The author presents three such definitions and for one of them:

$$B(X)_* = \{ u_1, u_4, u_6, u_8 \}$$
$$B(X)^* = \{ u_1, u_2, u_4, u_6, u_8 \}$$

The interpretation of this result is such that the missing values cause broadening of the potential span for the lower approximation, because they have to be inferred from the rest of the set. The upper approximation can change either way, because the missing values change the entire structure of a set

4. COMBINING BAYESIAN NETWORKS WITH ROUGH SETS: A CASE STUDY

As mentioned earlier, Bayesian belief networks are models that depict variables with probabilistic descriptions and their dependencies among each other. In general, the probability distribution function that reflects the state of a node is a conditional distribution that depends on the multidimensional distribution consisting of the node's parents (each state of the node has a probability for every combination of states the parent nodes may take). When evidence about the states of one of the nodes is found, the rest of the network is also updated according to the conditional probability tables and dependency relations of the nodes. However, the whole updating process becomes a problem, if the new evidence is distorted or missing.

This situation may not be a problem with off-line computations, such as assessment of software properties, which was outlined in Section 2. But if one wants to use BBN's for situation assessment in real time, when missing or distorted data come into play, as in circumstances such as sensor noise or sensor failure, especially over extended period of time, then the value of Bayesian reasoning becomes problematic. The cure for this, proposed in current work, is to use rough sets to infer the missing or distorted data.

A related problem is finding initial probabilities for a Bayesian network, which are often derived using expert knowledge. Uncertainty and vagueness in collecting data often create difficulties in giving meaningful values to the initial probabilities. In this project, we attempt to address both problems using rough set theory.

A. SOFTWARE TOOLS

To check the feasibility of the proposed idea, we decided to automate the entire inference process by using the public domain tools and a comprehensive case study. One of several software packages that can be used for BNN computations is called Netica [17]. In Netica, networks can be easily constructed and compiled to use for inference. In particular, the conditional probability tables (CPT's) may be generated from data in case files or the probabilities may be manually entered by the user. In real applications, probabilities and data may change in real time, however, the CPTs may be updated automatically as new data become available. Bayesian networks could then be used in situations, which change in real time, while the program is operating.

In this project, for a problem defined in terms of BBN's, with new data appearing in real time, Netica calculates the probabilities and updates them in the CPT's, reading from a case file, updating the nodes of a Bayesian network, and displaying the results. The logic of Netica code for reading a Bayesian network that updates it in real time is shown below.

```
while (true){
    // Remove CPTables of nodes in net,
    // so new ones can be learned.
    for (int n = 0;  n < numNodes;  n++) {
        Node node = (Node) nodes.get (n);
        node.deleteTables();
    }
    nd = (Node)nodes.get(node index);
    net.reviseCPTsByCaseFile(caseFile,
                             nodes, 1.0);
    net.write (new Streamer ("BName.dne"));
    net.compile();
    float tab[] = nd.getBeliefs();
    //Print the probabilities for a node.
    System.out.println(tab[0]+","+tab[1]);
}
```

The entire implementation depicted in Figure 3 works as follows. One program (the rough set tool that acts as a preprocessor) generates data for the case file and the Netica is able to read the data from this file consistently, even when it is being updated during operation. If the data are written into a case file after it is computed by rough set preprocessor, then the Bayesian network may be systematically updated with new data as they arrive.

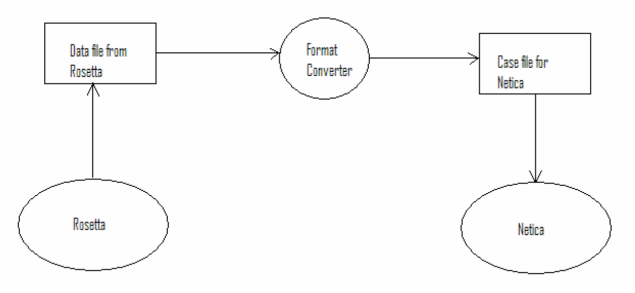


Fig. 3 – Implementation outline

The rough set tool we use, Rosetta [16], is able to import and export files in several different formats, but none of them are compatible with the case files used by Netica. Therefore, a text converter has been written to convert text files (a format that Rosetta exports) to case files that Netica software reads.

B. CASE STUDY

An application involving naval warfare, originating from the Australian Navy research [18], is used as a case study to demonstrate the implementation. Briefly speaking, there are two military forces called the Blue and Orange forces that are hostile towards each other and a country that the Orange forces obtain fuel supplies from and the Blue forces treat as neutral. The Blue forces have communications and surveillance facilities that the Orange forces want to destroy. Blue has set up a restricted area that contains the communication facilities and will consider any military activity or transportation of supplies to be hostile. Orange has a supply route that passes through the restricted area that it wants to defend.

Blue monitors the restricted area sensors and reconnaissance. Orange vessels that are likely to be detected are Guided Missile Frigates (FFG), Free Mantle Class Patrol Boats (FCPB), and Communication vessels. Oil tankers from the neutral country may also be detected. The position, mobility, and communications activity of the vessel are also recorded to try to determine the intent of the Orange Forces.

The Bayesian Network in Figure 4 is used to try to determine what the intentions of the Orange Forces are and how to respond to it by entering the findings from the sensors and reconnaissance into the appropriate nodes. However the probability distributions of all variables have just been initialized to uniform distributions and values for the conditional probability tables are needed.

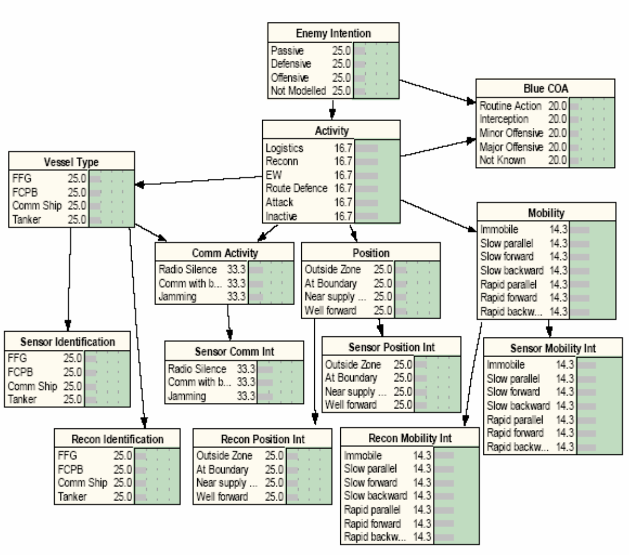
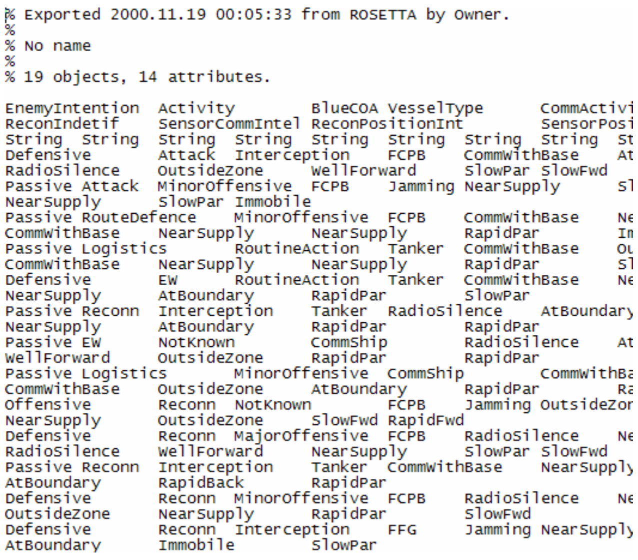


Fig. 4 – Bayesian network for the case study

The file whose contents is partially shown in Fig. 5 contains a table of several cases that are used to calculate the initial values for the conditional probability tables of the nodes. While Rosetta adds data to this file, based on real-time sensor computations, Netica takes input for each node and recalculates periodically the entire network, reducing uncertainty in case some sensor values are corrupted or missing.



```
% Exported 2000.11.19 00:05:33 from ROSETTA by Owner.
%
% No name
%
% 19 objects, 14 attributes.

EnemyIntention  Activity        BlueCOA VesselType      CommActivi
ReconIndetif    SensorCommIntel ReconPositionInt        SensorPosi
String  String  String  String  String  String  String  String  St
Defensive       Attack  Interception    FCPB    CommWithBase    At
RadioSilence    OutsideZone     WellForward     SlowPar SlowFwd
Passive Attack  MinorOffensive  FCPB    Jamming NearSupply      Sl
NearSupply      SlowPar Immobile
Passive RouteDefence    MinorOffensive  FCPB    CommWithBase    Ne
CommWithBase    NearSupply      NearSupply      RapidPar        Im
Passive Logistics       RoutineAction   Tanker  CommWithBase    Ou
CommWithBase    NearSupply      NearSupply      RapidPar        Sl
Defensive       EW      RoutineAction   Tanker  CommWithBase    Ne
NearSupply      AtBoundary      RapidPar        SlowPar
Passive Reconn  Interception    Tanker  RadioSilence    AtBoundary
NearSupply      AtBoundary      RapidPar        RapidPar
Passive EW      NotKnown        CommShip        RadioSilence    At
WellForward     OutsideZone     RapidPar        RapidPar
Passive Logistics       MinorOffensive  CommShip        CommWithBa
CommWithBase    OutsideZone     AtBoundary      RapidPar        Ra
Offensive       Reconn  NotKnown        FCPB    Jamming OutsideZor
NearSupply      OutsideZone     SlowFwd RapidFwd
Defensive       Reconn  MajorOffensive  FCPB    RadioSilence    Ne
RadioSilence    WellForward     NearSupply      SlowPar SlowFwd
Passive Reconn  Interception    Tanker  CommWithBase    NearSupply
AtBoundary      RapidBack       RapidPar
Defensive       Reconn  MinorOffensive  FCPB    RadioSilence    Ne
OutsideZone     NearSupply      RapidPar        SlowFwd
Defensive       Reconn  Interception    FFG     Jamming NearSupply
AtBoundary      Immobile        SlowPar
```

Fig. 5 – Bayesian network for the case study

5. CONCLUSION

BBN's proved to be a valuable tool when making decisions in systems with uncertain information. They loose their power, however, when some pieces of the evidence (i.e., input information) are corrupted or missing, which becomes especially critical in real time. A possible solution to this problem consists of two steps. First, a probability distribution over possible outcomes of the unknown/uncertain quantity is defined, and next this new information is communicated to the original BBN by additional nodes pointing to the nodes of interest. Surely, the network grows in this case, and what is more important, the process of assessing probabilities may be non-trivial.

Hopefully, rough sets theory provides a number of tools allowing coping with incomplete information. That is why we postulate to combine the two formalisms to obtain a universal machinery supporting the process of reasoning under uncertainty, in case of missing values of certain attributes of objects. To implement this idea, the use of two easily available tools: Netica (for BBN's) and Rosetta (for rough sets), has been proposed and tested to work cooperatively when solving instances of two real-life problems.

The proposed approach can be extended further towards mining interesting relationships among the entities constituting a problem under consideration (see, for example [19]). BBN's can be viewed as a concise summarization of a large data collection. On the other hand, rough sets offer additional tools to analyze possible dependencies among the data.

6. ACKNOWLEDGEMENT

The authors would like to thank student David Hatfield from Florida Gulf Coast University for performing the calculations using Netica and Rosetta.

Janusz Zalewski gratefully acknowledges the fellowship under the 2008 Air Force Summer Faculty Fellowship Program that enabled him work on this project.

7. REFERENCES

[1] J. Zalewski, S. Wierzchon, Combining Bayesian Networks and Rough Sets: A New Approach to Reasoning about Uncertainty, *Proc. ICNNAI2008, 5th Int'l Conference on Neural Networks and Artificial Intelligence*, Minsk, Belarus, May 27-30, 2008, pp. 22-27.

[2] Agena Ltd., *Bayesian Net References. Version 4*, London, 13 July. 2008, 39 pp., URL: http://agena-risk.com/resources/*BN_refs.doc*

[3] Z. Pawlak, Rough Sets, *International Journal of Computer and Information Sciences*, vol. 11, no. 5, pp. 341-356, 1982.

[4] Z. Pawlak, *Rough Sets – Theoretical Aspects of Reasoning about Data*, Dordrecht: Kluwer Academic Publishers, 1991.

[5] Z. Pawlak, AI and Intelligent Industrial Applications: The Rough Set Perspective, *Cybernetics and Systems: An International Journal*, vol. 31, pp. 227-252, 2000.

[6] *Rough Set Database System – A Bibliographic Database on Wide Aspects of Rough Sets*, University of Rzeszów, Poland, April 2008, URL: http://rsds.univ.rzeszow.pl/

[7] J. Grzymala-Busse, Three Approaches to Missing Attribute Values – A Rough Set Perspective. *Proc. Workshop on Foundation of Data Mining at the 4th IEEE Int'l Conference on Data Mining*, Brighton, UK, November 1-4, 2004

[8] A. Kornecki, J. Zalewski, Experimental Evaluation of Software Development Tools for Safety-Sritical Real-Time Systems, *Innovations in Systems and Software Engineering: A NASA Journal*, vol. 1, pp. 176-188, 2005.

[9] J. Zalewski, A.J. Kornecki, H.L. Pfister, Numerical Assessment of Software Development Tools in Real-Time Safety Critical Systems Using Bayesian Belief Networks, *Proc. Int'l Multiconference on Computer Science and Information Technology*, Wisla, Poland, 6-10, November 2006, pp. 351-360.

[10] M. Neil, N. Fenton, Predicting Software Quality Using Bayesian Belief Networks, in *Proc. SEW-21, Annual NASA Goddard Software Engineering Workshop*, Washington, DC, 4-5 December 1996, pp. 217-230.

[11] G. Dahll, B.A. Gran, The Use of Bayesian Belief Nets in Safety Assessment of Software Based Systems, *International Journal of General Systems*, vol. 29, no. 2, pp. 205-229, 2000.

[12] K.A. Delic, F. Mazzanti, L. Strigini, Formalising Engineering Judgement on Software Dependability via Belief Networks, in *Proc. DCCA-6, 6th IFIP International Working Conference on Dependable Computing for Critical Applications*, M. Dal Cin, C. Meadows, W.H. Sanders, (Eds.), Los Alamitos, CA: IEEE Computer Society, 1998, pp. 291-305.

[13] A. Helminen, *Reliability Estimation of Safety-Critical Software-Base Systems Using Bayesian Networks*, Report STUK-YTO-TR 178, Radiation and Nuclear Safety Authority, Helsinki, Finland, June 2001.

[14] J. Grzymała-Busse, LERS – A System for Learning from Examples Based on Rough Sets," in R. Słowiński, ed., *Intelligent Decision Support: Handbook of Applications and Advances of Rough Set Theory*, Dordrecht: Kluwer, 1992, pp. 3-18.

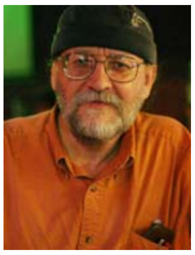
[15] J. Komorowski, L. Polkowski, A. Skowron, Rough Sets: A Tutorial, in S.K. Pal and A. Skowron, Eds., *Rough-Fuzzy Hybridization: A New Method for Decision Making*, Berlin: Springer-Verlag, 1998.

[16] *Rosetta*, The Linnaeus Centre for Bioinformatics, Uppsala Univ., Sweden, URL: http://rosetta.lcb.uu.se/general/download/

[17] *Netica*, Norsys Software Corporation, Vancouver, BC, Canada, URL: http://www.norsys.com/

[18] B. Das, *Representing Uncertainties Using Bayesian Networks*, Report No. DSTO-TR-0918, Defence Science and Technology Organization, Information Technology Division Electronics and Surveillance Research Lab, Sydney, Australia, December 1999.

[19] J. Shao, Knowledge Discovery in Alarm Data Analysis. *Proc. SOFSEM'96, 23rd Seminar on Current Trends in Theory and Practice of Informatics*, Milovy, Czech Republic, November 23-30, 1996, Springer-Verlag, pp. 433-440.



Janusz Zalewski *is a professor of computer science and engineering at Florida Gulf Coast University. His research interests include real-time embedded systems, safety-critical systems and computing education. He worked on projects for nuclear research institutes and several industrial companies, including Harris, Boeing, Lockheed Martin and others.*

Sławomir T. Wierzchoń *is a professor of computer science at the University of Gdańsk and at the Polish Academy of Sciences, where he leads a group on artificial intelligence. His current research interests include artificial immune systems, fuzzy sets and rough sets, theory of evidence, evolutionary computations and decision making under uncertainty.*

Henry Pfister *is a research scientist at the Air Force Research Lab in the Eglin Air Force Base and an adjunct professor at the Research and Engineering Education Facility of the University of Florida. His research interests include modeling, analysis, simulation and optimization of engineering systems, as well as the analysis of uncertainty using fuzzy sets and other non-traditional techniques.*