The Journal of Supercomputing (2024) 80:75–97
https://doi.org/10.1007/s11227-023-05458-y

A generalized approach to construct node probability table for Bayesian belief network using fuzzy logic

Chandan Kumar[1] · Sudhanshu Kumar Jha[2] · Dilip Kumar Yadav[3] · Shiv Prakash[2] · Mukesh Prasad[4]

Accepted: 29 May 2023 / Published online: 22 June 2023

Abstract
The cause–effect relationship has tremendous role in interpreting the engineering and scientific problems which basically deals with the identifying potential causes of problem. Bayesian belief networks (BBN) also referred as Bayesian casual probabilistic network used widely to deal with probabilistic events to elucidate the complications having uncertainty. A major challenge in BBN is to construct a node probability table (NPT), which grows exponentially with the rising number of variables. Various approaches exist for NPT construction, including expert elicitation, data analysis, survey and weighted functions, noisy-OR, noisy-MAX, recursive noisy-OR (ROR), extended recursive noisy-OR, and ranked nodes. However, these methods are problem-specific and lacking behind a generalized approach applicable to all problem types. To address this issue, this paper proposes a generalized universal approach for constructing the NPT using fuzzy logic. The suggested strategy has been validated by applying it to a BBN prototype for software design and development. The proposed strategy has been evaluated with best-case and worst-case software metrics.

Keywords Fuzzy logic · Bayesian belief network (BBN) · Software metrics · Node probability table (NPT)

1 Introduction

The "Bayesian belief network (BBN)" was introduced by Judea Pearl, in 1985, which is basically based on Bayes theorem proposed by Rev. Thomas Bayes [3]. Bayesian belief network combines graph theory with probability theory. Probability theory and graph theory are used to explain the system's uncertainty, and they help to show that the probability distribution has an independent structure that can be broken down into more manageable components [12]. Cooper has shown that [4]

Extended author information available on the last page of the article an inference in BBN is a kind of NP-Hard problem, where inference complexity to calculate the queries involves $O(d^n)$ steps for given n number of nodes with d number of variables in the network. In order to store full-joint distribution, a total space complexity requires nearly to $O(d^n)$.

A Bayesian belief network has been effectively used in a number of fields, including software development, medical diagnostic systems, weather forecasting, agriculture area problems, stakeholder engagement, project management, reliability engineering and safety engineering, signal processing, and other domains of engineering. In the last twenty years, BBN has been used widely in software engineering such as software quality prediction [18], software defect prediction [6, 8, 14, 23, 27], and software risk assessment [32]. It has been established that BBN can express uncertainty wherever it is used. However, there are two major obstacles to the construction of a substantial BBN: (i) construction of causal model and (ii) construction of node probability table (NPT).

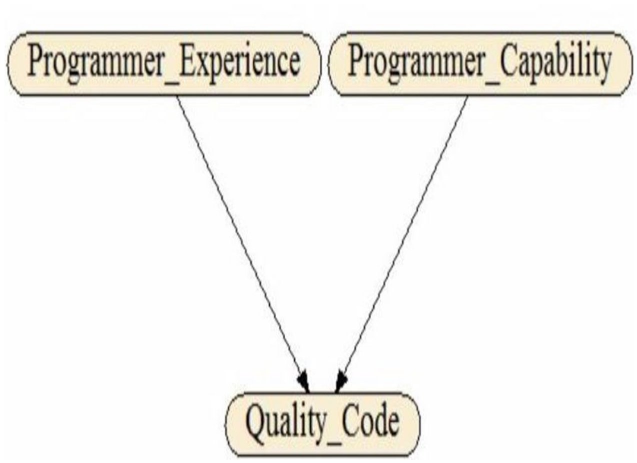
Nadkarni and Shenoy [19] have mentioned and described the two different methods for the construction of a causal model named data-based and knowledge-based approach. These two methods are widely used in the construction of a causal model. This paper focuses purely on the challenge of constructing the NPT.

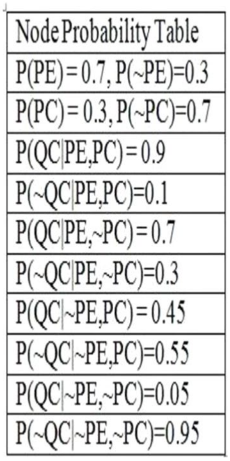
NPT specifies the probability distributions of the child node, the descendent, for all possible combinations of the parent node's states. In the literature, there are different methods/approaches available for constructing the NPT of BBN, such as Expert elicitation based [22, 29], Data analysis based [20, 30], Data and Expert based [35], Survey and weighted functions [25], Noisy-OR [9, 24, 34], Noisy-MAX [5], "Recursive Noisy-OR (RNOR)" [17], and Extended Recursive Noisy-OR (ERNOR) [26], Ranked nodes [7], Improved versions of ranked nodes [15, 16], and so on. The most difficult aspect of building a BBN is often constructing NPTs manually through domain experts; however, outmost care should be taken in constructing of NPTs as when the participating number of nodes increases, NPT's size grows exponentially. In some cases, surprisingly even for single NPT, there are tens or hundreds of probabilities that need to be given. A method based on data analysis has been suggested [20, 30] to address the elicitation difficulty of expert-basedNPT construction. However, major challenge with the real-world applications has insufficient sample data. Therefore, in this situation, the pure data-based approach of NPT construction may not work well. Zhou et al. [35] proposed an amalgam of partial data and an expert discernment-based approach for NPT construction to reduce the complexity of expert-based as well as data-based NPT construction. Perkusich et al. [25] proposed a technique that creates NPTs using weighted expressions created using information gathered from subject-matter experts through a survey. This method's benefit over expert-based ones is that it generates NPTs using weighted expressions. A technique called Noisy-OR was proposed by Pearl [24]; however, their method considers the Boolean nodes only ignore the interactions among variables [13]. Diez and Galan [5] have proposed Noisy-MAX to overcome this drawback. However, the heavy assumptions of independence and some approximations might not be accurate if standard deviations are large, and the observed evidence greatly deviates from the expected values. An improved version of Noisy-OR has been proposed by Lemmer & Gossink [17] which is known as RNOR. However, in RNOR when there are more than three causes, there is an asymmetry issue. To overcome this issue ERNOR has been proposed by Quintanar-Gago & Nelson [26]. Fenton et al. [7] developed the ranking nodes strategy especially for the parent nodes which is basically using probabilistic distribution as weighting function and describes continuous values in discretized intervals, enabling the estimation of huge CPTs. This approach has been improved by Laitila and Virtanen [15, 15], and the viability of applying it to real-world issues has been established. According to the literature, all of the NPT generating techniques currently in use are problem-specific and cannot be used to solve all kinds of problems. Typically, a BBN falls under the category NP-Hard problem, which require efficient solution; however, no distinct solution is available in the literature. Therefore, in this paper, a generalized approach to constructing the NPT has been proposed using fuzzy logic to provide a near-optimal solution.

Additionally the paper is presented as follows: Sect. 2 briefly presents BBN. The Sect. 3, described the proposed method to construct the NPT of BBN model. An illustrative example of applying the proposed methodology is described in Sect. 4. The validation of the proposed method is presented in Sect. 5, followed by the conclusion in Sect. 6.

2 Background

A BBN primarily gives a visual depiction of causal links among variables based on conditional probabilities to convey the uncertainties in the dependences among variables in a given "Directed Acyclic Graph (DAG)," $G(V,E)$, where E represents the edges and V symbolizes the vertices. In the BBN each vertex denotes a variable, and causal connections between variables being denoted by each edge. Based on the evidence of the information given, the posterior probability in BBN is computed using the Bayes theorem. Figure 1 shows a simple BBN where the causal relationships





Node Probability Table
P(PE) = 0.7, P(~PE) = 0.3
P(PC) = 0.3, P(~PC) = 0.7
P(QC|PE,PC) = 0.9
P(~QC|PE,PC) = 0.1
P(QC|PE,~PC) = 0.7
P(~QC|PE,~PC) = 0.3
P(QC|~PE,PC) = 0.45
P(~QC|~PE,PC) = 0.55
P(QC|~PE,~PC) = 0.05
P(~QC|~PE,~PC) = 0.95

Fig. 1 Example of BBN

among 3 variables are shown. Authors would like to investigate the effect of programmer experience (PE) and programmer capability (PC) on quality of code (QC).

The node probability table of all variables is also presented in Fig. 1, where high and low are the two states of each node, *i.e.*, P(PE) = High, P(~PE) = Low. Similarly for P(PC), P(~PC), P(QC) and P(~QC). With the available values in NPT, next step is to calculate the probability of each vertex. For an example, the probability of given vertex in Fig. 1, i.e., programmer capability (PC) and quality of code (QC), is P (PC|QC). Based on the Bayes theorem, this can be calculated by the mathematical expression shown in Eq. 1.

$$P(\text{PC}/\text{QC}) = \frac{P(\text{QC}/\text{PC}) * P(\text{PE})}{P(\text{QC})} \tag{1}$$

The same can also be written as:

$$\begin{aligned} P(\text{QC}) &= P(\text{QC}/\text{PE}, \text{PC}) * P(\text{PE}) \\ &\quad + P(\text{QC}/\text{PE}, \sim \text{PC}) * P(\text{PE}) * P(\sim \text{PC}) \\ &\quad + P(\text{QC}/ \sim \text{PE}, \text{PC}) * P(\sim \text{PE}) * P(\text{PC}) \\ &\quad + P(\text{QC}/ \sim \text{PE}, \sim \text{PC}) * P(\sim \text{PE}) * P(\sim \text{PC}) \end{aligned} \tag{2}$$

Since *P* (QC|PE, PC), *P* (QC|PE, ~PC), *P* (QC|~PE, PC), *P* (QC|~PE, ~PC) are being calculated based on node probability table, thus the diagnosis probability *P* (PC|QC) may also be determined.

A BBN has several other advantages [12], Nadkarni and Shenoy [19] such as method is probabilistic, it can be created using a modest dataset, BBN manages circumstances in which some data entries are inaccessible or missing. Further, it is also possible to model causal links using BBN, BBN can readily include expert data, development of group model building is also possible, and, whenever new information becomes available, updating is simple.

3 Proposed approach

Here, a generalized approach for constructing the NPT using fuzzy logic is being presented. The suggested strategy has been validated by applying it to a BBN model of software design and development and evaluating it with best-case and worst-case software metrics.

Proposed approach starts with identifying a casual concept. The causal concepts refer to the variables or factors within a specific domain that are believed to have causal relationships with each other. These concepts represent the cause–effect relationships among different elements in the system being modelled. Each causal concept is typically represented as a node in the BBN. To identify the causal concepts in a BBN, a domain knowledge is required along with understanding of system under consideration. It is important to note that identifying causal concepts in a BBN is an iterative process that involves a combination of domain knowledge, empirical evidence, and expert input. The process may require adjustments and refinements so that a deeper understanding of the system and its causal dynamics may be obtained.

In the next step a casual relationship is being generated which basically refers to the cause–effect connections between variables represented as nodes in the network. These relationships indicate how changes in one variable can influence or cause changes in another variable within the system being modelled. As suggested by Nadkarni and Shenoy [19] there are mainly two ways to generate a causal relationships data-based approach and knowledge-based approach. However, it is important to note that generating causal relationships in a BBN sometimes requires a combination of domain knowledge, expert input, and empirical evidence.

In the next step it is necessary to define a membership function for each vertex (input and output node), which basically assigns a degree of membership to an element in the given fuzzy set. It represents the extent to which an element fits in a particular fuzzy set or category. Membership functions are essential in fuzzy logic because they allow for the representation of uncertainty and partial truth. It is important to note that defining membership functions in fuzzy logic involves both subjective and objective considerations. Subjective aspects include expert opinions and linguistic interpretations, while objective aspects can involve statistical analysis or data-driven approaches to determining membership values.

In the following step, a fuzzy "IF–Then" rule has been shown that is the fundamental component of fuzzy logic system, which basically prompts a relationship among input (antecedents) and output (consequents) variables using fuzzy logic terms. These rules help in making decisions or performing actions based on the input conditions. Designing fuzzy "IF–Then" rules require a combination of empirical data, domain understanding, and expert knowledge. It is an important to ensure the rules adequately capture the relationship among input and output variables and reflect the desired behavior of the fuzzy logic system.

In subsequent step a fuzzy inference and defuzzification using MATLAB tool have been performed. Fuzzy inference and defuzzification are two key steps in a fuzzy logic system that converts fuzzy inputs into crisp outputs. Fuzzy inference mainly determines the degree of membership or the fuzzy output based on fuzzy "IF–Then" rule and the input variables. It involves evaluating the rules and combining their activations to obtain a fuzzy output. Defuzzification is a procedure to convert the fuzzy output (obtained from the fuzzy inference) into a crisp or numerical value that represents final output of fuzzy logic system.

Finally a NPT with the help of defuzzified area has been obtained. The defuzzified shape obtained from the fuzzy inference and defuzzification process has been further used to calculate the NPT using geometrical or definite integration method.

Algorithm NPT_fuzzy_logic summarizes the steps described above in formal way.

```
Algorithm: NPT_ fuzzy_logic (node, node_link)
        Input:  Given random variable $x_i$ $\forall$ $1 \le i \le n$ at each node (vertex) in a directed acyclic graph $G$
        Output: Node Probability Table
while(true)
{
        step 1: procedure identify_casual_concepts (node, node_link);
        step 2: $\forall$ node $n_i$ $1 \le i \le n$
                    generate_causal_relationships($n_i$)
        step 3: $\forall$ node, Input_Node $n_i$ $1 \le i \le n$ and $\forall$ Output_Node $o_i$
                    define membership_function ($n_i$, $o_i$) used in BBN.
        step 4: Design fuzzy ''IF–THEN'' rules.
        step 5: Perform fuzzy inference and defuzzification
        step 6: Calculate NPT(defuzzified area obtained in step 5)
}
```

For sake of simplicity and reader perspective, an example in the next section elaborates the entire process of proposed method.

4 A descriptive example

This section demonstrates the procedure of proposed algorithm with help of an example. The BBN model of software design and development [8] as presented in Fig. 2 has been taken anonymously to explain the working of proposed approach.

4.1 Identify the causal concepts.

One of the most crucial steps in BBN creation is the identification of causal notions. A causal concept can be defined as cause-and-effect relationships among

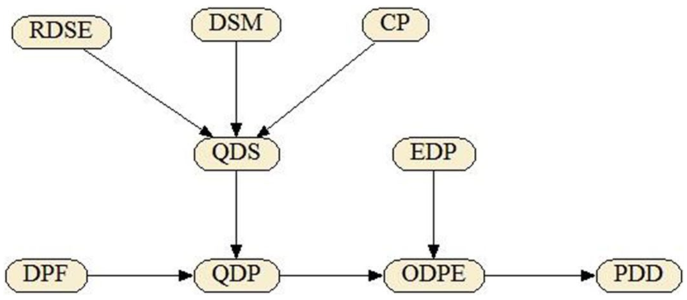


Fig. 2 BBN model of software design and development

the development activities and endpoints in a graphical model. It may be an attribute, issue, factor, assumptions or variable of a domain and typically represented by a node in BBN. Fenton et al. [8] have identified the 9 causal concepts for the BBN model of software design and development. The identified causal concepts are RDSE (Relevant Development Staff Experience), DSM (Development Staff Motivation), CP (Capability of Programmer), QDS (Quality of Development Staff), DPF (Defined Process Followed), QDP (Quality of Development Process), EDP (Effort in Development Process), ODPE (Overall Development Process Effectiveness), PDD (Probability of Defects in Development).

- **RDSE (Relevant Development Staff Experience)** RDSE is the first input metric of QDS. The impact of staff having strong technical backgrounds and experience on QDS is significant.
- **DSM (Development Staff Motivation)** DSM is the second metric of QDS. The employees who work on software development are positive people who do their utmost to generate high-quality design and code.
- **CP (Capability of Programmer)** CP is the last metric of QDS. The ability of a programmer is influenced by their education, background, intelligence, and domain expertise.
- **QDS (Quality of Development Staff)** QDS is one of the output metrics of the BBN model of software design and development. The output of QDS is depending on the evidence of RDSE, DSM, and CP.
- **DPF (Defined Process Followed)** DPF is the input metrics of QDP. The defined process must be followed to achieve a good-quality development process.
- **QDP (Quality of Development Process)** QDP is the 2nd output metric of the BBN model of software design and development. The output of QDP is depending on the evidence of DPF and QDS.
- **EDP (Effort in Development Process)** EDP is the input metric for ODPE. More effort spent in the development process will increase the chances of overall development process effectiveness.
- **ODPE (Overall Development Process Effectiveness)** ODPE is the 3rd output metric of the BBN model of software design and development. The output of ODPE is depending on the evidence of EDP and QDP.
- **PDD (Probability of Defects in Development)** PDD is the desired/last output metric of the BBN model of software design and development. The outcome of PDD is depending on the evidence of ODPE.

4.2 Generate the causal relationships

Causal relationships among nodes can be achieved with the help of causal connections. A unidirectional arrow is used to indicate a causal connection, which is a knot connecting two or more causal notions. Positive or negative causal relationships are possible. A positive connection means that increasing the causative concept causes the effect concept to increase, whereas a negative connection

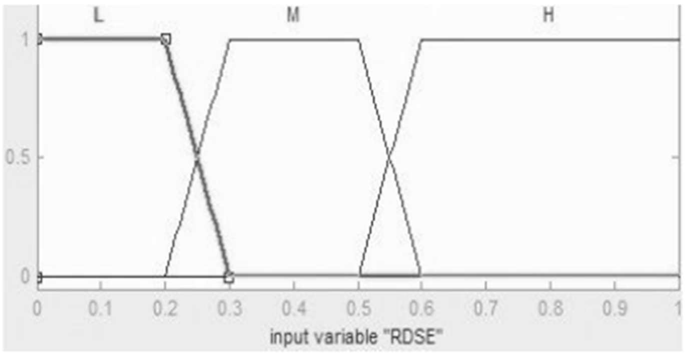


Fig. 3 Relevant development staff experience

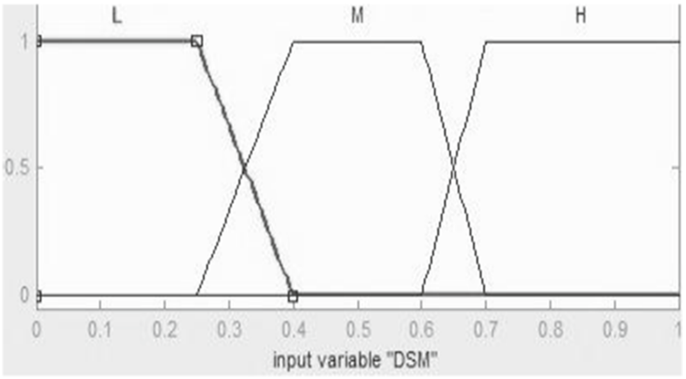


Fig. 4 Development staff motivation

means that increasing the causal concept causes the effect concept to decrease. For example, in Fig. 2, “RDSE,” “DSM,” and “CP” have a positive influence on “QDS.” Thus, the higher RDSE, DSM, and CP, the higher will be the QDS. On the other hand, “ODPE” has a negative influence on “PDD.” Thus, the higher ODPE, the lower PDD.

4.3 Outline the membership function

In order to build membership functions, either domain experts or actual data might be used. [28, 33]. Numerous geometries, including trapezoidal, triangular, polygonal, and others, are possible for membership purposes. [28]. However, triangular and trapezoidal forms being preferred as it provides a useful depiction of domain expert knowledge and subsequently make calculation easier [10, 31]. Domain experts are used to define the membership functions for all input and

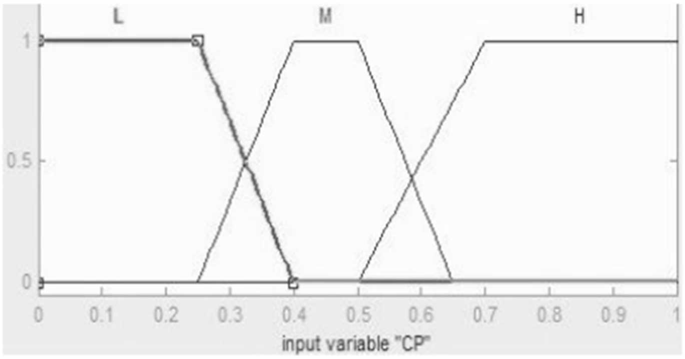


Fig. 5 Capability of programmer

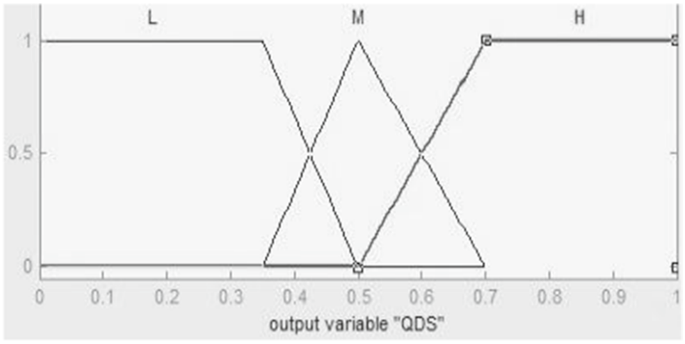


Fig. 6 Quality of development staff

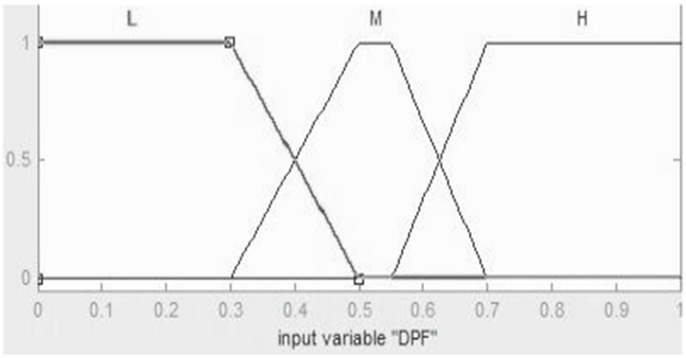


Fig. 7 Defined process followed

output metrics taken into account in Fig. 2 and subsequently in Figs. 3, 4, 5, 6, 7, 8, 9, 10, 11 of the BBN model.

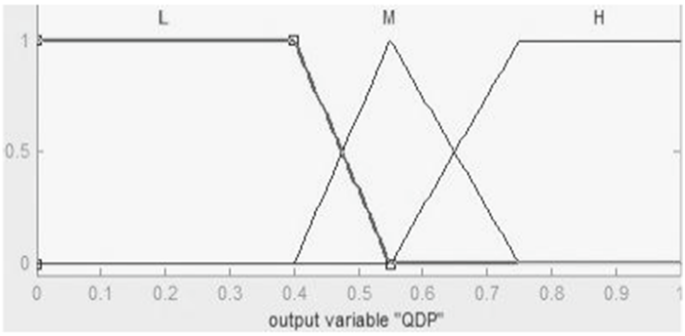


Fig. 8 Quality of development process

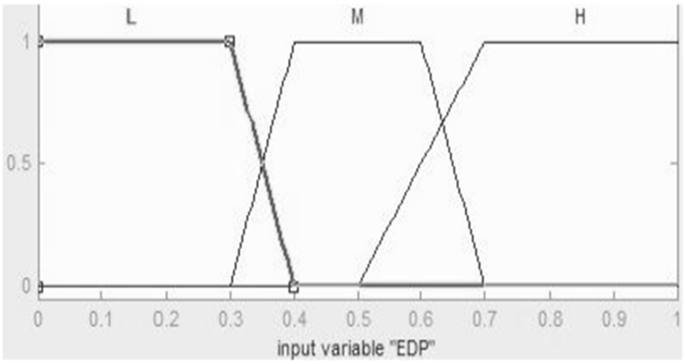


Fig. 9 Effort in development process

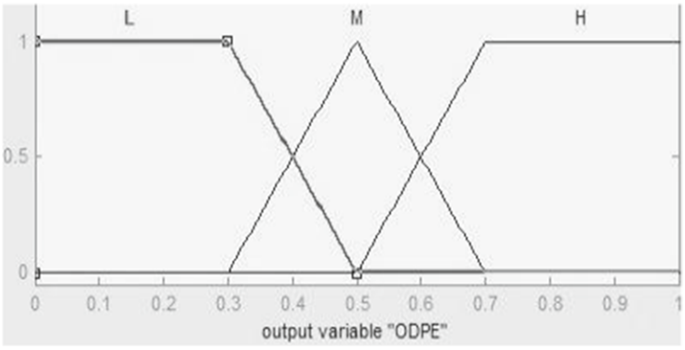


Fig. 10 Overall development process effectiveness

4.3.1 Design "IF–THEN" fuzzy rules

Different sources, including subject matter experts, knowledge engineering, and historical data analysis, from existing literature, can be used to create a fuzzy

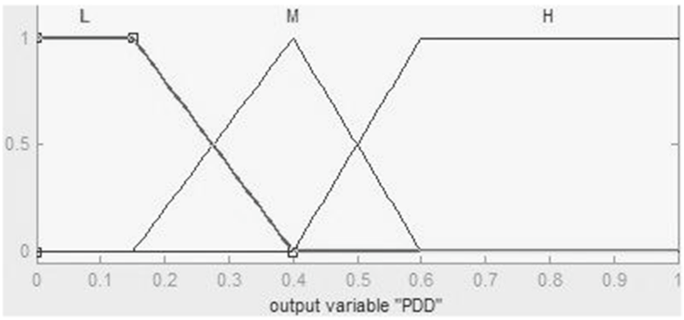


Fig. 11 Probability of defects in development

"IF–THEN" rule. [28]. Typically, domain specialists assist in the formulation of fuzzy rules. The designed "IF–THEN" fuzzy rules concerning individual output node of the BBN model are explained below:

- **Quality of Development Staff (QDS)** If RDSE, DSM, and CP are high, then QDS will be high. Similarly, if RDSE, DSM, and CP are low, then QDS will be low. Three input nodes, each with following three linguistic states low (L), medium (M), and high (H), make up this output node. Consequently, there are 27 rules in total. The following interpretation is given to the fuzzy rules.

  Rule 1:- If RDSE is *H* and DSM is *H* and CP is *H*, then QDS is *H*.
  Rule 2:- If RDSE is *H* and DSM is *H* and CP is *M*, then QDS is *H*.
  …
  Rule 26:- If RDSE is *L* and DSM is *L* and CP is *M*, then QDS is *L*.
  Rule 27:- If RDSE is *L* and DSM is *L* and CP is *L*, then QDS is *L*.

- **Quality of Development Process (QDP)** There are two input nodes and three linguistic states in each of the input nodes in this output node: low (*L*), medium (*M*), and high (*H*). Consequently, there are nine rules in total. Thus following fuzzy rules have been designed:

  Rule 1:- If QDS is *H* and DPF is *H*, then QDP is *H*.
  Rule 2:- If QDS is *H* and DPF is *M*, then QDP is *H*.
  …
  Rule 8:- If QDS is *L* and DPF is *M*, then QDP is *L*.
  Rule 9:- If QDS is *L* and DPF is *L*, then QDP is *L*.

- **Overall Development Process Effectiveness (ODPE)** In this output node, there are also two input nodes and three linguistic states in each of the input nodes: low (*L*), medium (*M*), and high (*H*) and thus there are total 9 rules. Following fuzzy rules are developed.

Rule 1:- If QDP is *H* and EDP is *H*, then ODPE is *H*.
Rule 2:- If QDP is *H* and EDP is *M*, then ODPE is *H*.
…
Rule 8:- If QDP is *L* and EDP is *M*, then ODPE is *L*.
Rule 9:- If QDP is *L* and EDP is *L*, then ODPE is *L*.

- **Probability of Defects in Development (PDD)** This output node has only one input node and three linguistic states, such as low (*L*), medium (*M*), and high (*H*) and thus only 3 fuzzy rules are being developed.

Rule 1:- If ODPE is *H*, then PDD is *L*.
Rule 2:- If ODPE is *M*, then PDD is *M*.
Rule 3:- If ODPE is *L*, then PDD is *H*.

4.4 Fuzzy inference and defuzzification

The output of each fuzzy rule is evaluated and combined by the fuzzy inference engine. An algorithm for fuzzy inference converts one fuzzy set into another. The crisp value must be retrieved as an output in various applications, and fuzzy set is mapped into crisp value using defuzzification techniques like centroid, max–min, bisection, etc. The proposed approach calculates the crisp value using centroid method of defuzzification, commonly known as center of area or center of gravity. This method of defuzzification is the most popular and physically pleasing method available [28]. The output of defuzzification for the first evidence of node QDS is shown in Fig. 12. The MATLAB Fuzzy Logic Toolbox is used to perform the fuzzy inference and defuzzification process.

4.5 Construct NPT with the help of defuzzified area

The NPT of a node can be constructed with the help of a defuzzified shape obtained from the fuzzy inference and defuzzification process. For better visualization and understanding purpose the graphical representation for the output of the first evidence of node QDS is drawn, and it is shown in Fig. 13. In our case, triangular and trapezoidal membership functions have been used. So, the obtained defuzzified shape is also in triangular and trapezoidal form. For calculating the defuzzified area of triangular and trapezoidal shapes, the geometrical method or definite integration method can be applied. However, for the membership functions like Gaussian, Sigmoidal, *Z* curves, *S* curves, Pi curves, etc., only the definite integration method can be applied for calculating the defuzzified area. Therefore, in our case, both methods (geometrical and definite integration) can be applied.

First Method Geometrical method.

- Defuzzified area of low (DV_L): From Fig. 13, it is found that the shape is in triangular form.

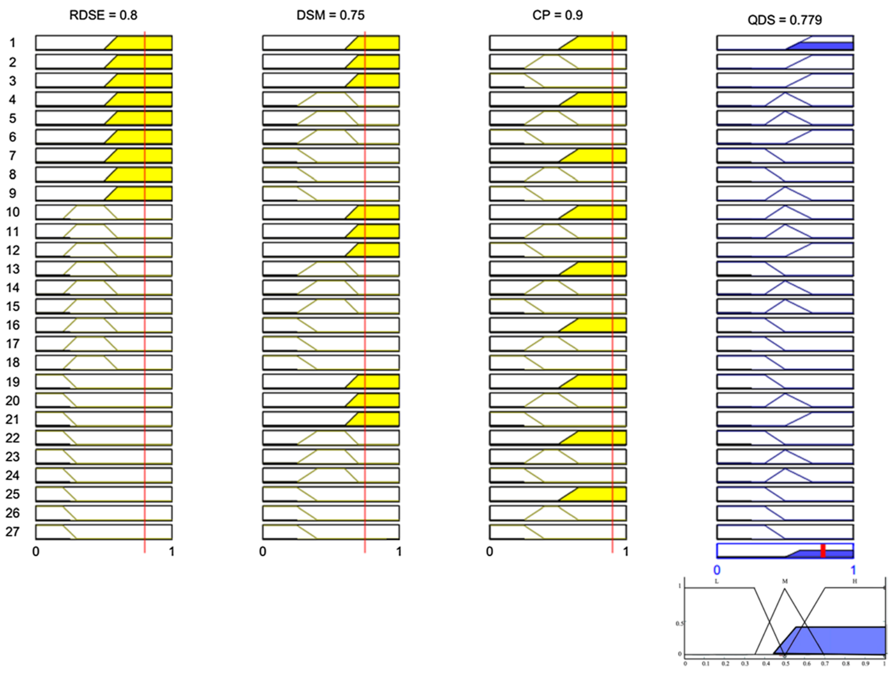


Fig. 12 Output of defuzzification for first evidence of node QDS

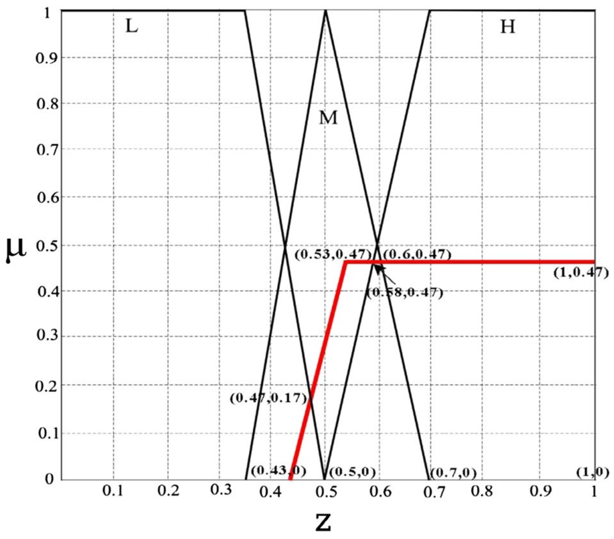


Fig. 13 Graph of defuzzified area of first evidence

$$\begin{aligned}\mathrm{DV_L} &= 0.5 * \text{Base} * \text{Height}\\ &= 0.5 * 0.07 * 0.17\\ &= 0.00595\end{aligned}$$

- Defuzzified area of medium ($\mathrm{DV_M}$): From Fig. 13, it is found that the shape is in trapezoidal form, but the area of low is overlapping.

$$\begin{aligned}\mathrm{DV_M} &= \text{Total defuzzified area} - \text{overlapping area of low}\\ &= \left[(\text{Area of trapezium}) - (\text{Area of triangle})\right]\\ &= \left[(0.5(\text{Sum of parallel side) Height}) - (0.5 * \text{Base} * \text{Height})\right]\\ &= [(0.5(0.27 + 0.07)0.47) - (0.5 * 0.07 * 0.17)]\\ &= 0.07\end{aligned}$$

- Defuzzified area of high ($\mathrm{DV_H}$): From Fig. 13, it is found that the shape is in trapezoidal form.

$$\begin{aligned}\mathrm{DV_H} &= 0.5(\text{Sum of parallel side) Height}\\ &= 0.5(0.3 + 0.4)0.47\\ &= 0.1645\end{aligned}$$

Now, the probability of low, medium, and high can be calculated using the formula given in Eq. 3.

$$\text{Probability of } L \text{ or } M \text{ or } H = \frac{\text{Defuzzified area of } L \text{ or } M \text{ or } H}{\text{Total Defuzzified area}} \tag{3}$$

Probability of low:

$$= \frac{0.00595}{0.00595 + 0.07 + 0.1645} = 0.025 = 0.03$$

Probability of medium:

$$= \frac{0.07}{0.00595 + 0.07 + 0.1645} = 0.29$$

Probability of high:

$$= \frac{0.1645}{0.00595 + 0.07 + 0.1645} = 0.6839 = 0.68$$

Second Method Definite Integration Method (DIM).

The graph shown in Fig. 13 is between the possibility (μ) and the point (z) which is a straight line. The equation of a straight line is shown in Eq. 4.

$$\mu - \mu_1 = m\left(z - z_1\right) \tag{4}$$

Here m is the slope of a straight line and its formula is shown in Eq. 5.

$$\text{slope of a straight line} = \frac{\mu_2 - \mu_1}{z_2 - z_1} \tag{5}$$

where (z_1, μ_1) and (z_2, μ_2) are the known points of a straight line

From this graph (Fig. 13), it is easy to calculate defuzzified area of low, medium, and high using definite integration method (DIM) by eliminating the overlapping area. Equation for calculating the defuzzified area of low, medium, and high is shown in Eq. 6.

$$\begin{aligned}\text{DIM} = &\left[\int_{a_1}^{b_1} \mu_1(z)\mathrm{d}z + \int_{a_2}^{b_2} \mu_2(z)\mathrm{d}z + \ldots + \int_{a_n}^{b_n} \mu_n(z)\mathrm{d}z\right] \\ &- \left[\int_{c_1}^{d_1} \mu_{k_1}(z)\mathrm{d}z + \int_{c_2}^{d_2} \mu_{k_2}(z)\mathrm{d}z + \ldots + \int_{c_n}^{d_n} \mu_{k_n}(z)\mathrm{d}z\right]\end{aligned} \tag{6}$$

where $a_i, b_i, c_i, d_i \forall 1 \le i \le n$ are the limits of integration.

- Defuzzified area of low (DV_L): The defuzzified area of low is lying in the interval $z \in [0.43, 0.5]$; it is divided into two parts, i.e., $z \in [0.43, 0.47]$ and $z \in [0.47, 0.5]$. By putting the value of $z_1 = 0.43$, $\mu_1 = 0$, $z_2 = 0.47$, $\mu_2 = 0.17$ in Eqs. 4 and 5 for $z \in [0.43, 0.47]$ the equation of straight line is:

$$\mu = 4.25z - 1.827 \tag{7}$$

Similarly, by putting the value of z_1=0.5, μ_1=0, z_2=0.47, μ_2=0.17 in Eqs. 4 and 5 for $z \in [0.47, 0.5]$ the equation of straight line is:

$$\mu = -5.67z + 2.835 \tag{8}$$

The DV_L can be calculated using Eqs. 6, 7, and 8 as:

$$\begin{aligned}\text{DV}_\text{L} &= \int_{0.43}^{0.47} 4.25z - 1.827\mathrm{d}z + \int_{0.47}^{0.5} -5.67z + 2.835\mathrm{d}z \\ &= 0.00597\end{aligned}$$

- Defuzzified area of medium (DV_M): The defuzzified area of medium is lying in the interval $z \in [0.43, 0.7]$; it is divided into four parts, i.e., $z \in [0.5, 0.47]$, $z \in [0.47, 0.53]$, $z \in [0.53, 0.6]$, and $z \in [0.6, 0.7]$. By putting the value of $z_1 = 0.5$, $\mu_1 = 0$, $z_2 = 0.47$, $\mu_2 = 0.17$ in Eqs. 4 and 5 for $z \in [0.5, 0.47]$ the equation of straight line is:

$$\mu = -5.67z + 2.835 \tag{9}$$

Similarly, by putting the value of z_1=0.43, μ_1=0, z_2=0.47, μ_2=0.17 for $z \in$ [0.47, 0.53], z_1=0.53, μ_1=0.47, z_2=0.6, μ_2=0.47 for $z \in$ [0.53, 0.6], and z_1=0.7, μ_1=0, z_2=0.6, μ_2=0.47 for $z \in$ [0.6, 0.7] in Eqs. 4 and 5.

The equation of straight line for $z \in$ [0.47, 0.53], $z \in$ [0.53, 0.6], and $z \in$ [0.6, 0.7] is as follows:

$$\mu = 4.25z - 1.827 \tag{10}$$

$$\mu = 0.47 \tag{11}$$

$$\mu = -4.7z + 3.29 \tag{12}$$

The DV_M can be calculated using Eqs. 6, 9, 10, 11, and 12 as:

$$DV_M = \left[\int_{0.5}^{0.47} -5.67z + 2.835dz + \int_{0.47}^{0.53} 4.25z - 1.827dz + \int_{0.53}^{0.6} 0.47dz + \int_{0.6}^{0.7} -4.7z + 3.29dz \right]$$
$$= 0.0717$$

- Defuzzified area of high (DV_H): The defuzzified area of high is lying in the interval $z \in [0.5, 1]$; it is divided into two parts, i.e., $z \in [0.5, 0.58]$ and $z \in [0.58, 1]$. By putting the value of $z_1 = 0.5$, $\mu_1 = 0$, $z_2 = 0.58$, $\mu_2 = 0.47$ in Eqs. 4 and 5 for $z \in [0.5, 0.58]$ the equation of straight line is:

$$\mu = 5.875z - 2.937 \tag{13}$$

Similarly, by putting the value of z_1=0.58, μ_1=0.47, z_2=1, μ_2=0.47 in Eqs. 3 and 4 for $z \in$ [0.58, 1] the equation of straight line is:

$$\mu = 0.47 \tag{14}$$

The DV_H can be calculated using Eqs. 6, 13, and 14 as:

$$DV_H = \left[\int_{0.5}^{0.58} 5.875z - 2.937dz + \int_{0.58}^{1} 0.47dz \right] - \left[\int_{0.5}^{0.58} 5.875z - 2.937dz + \int_{0.58}^{0.6} 0.47dz + \int_{0.6}^{0.7} -4.7z + 3.29dz \right]$$

$$= 0.1645$$

Now, the probability of low, medium, and high can be calculated using the formula given in Eq. 3.

Probability of low:

$$= \frac{0.00597}{0.00597 + 0.0717 + 0.1645} = 0.025 = 0.03$$

Probability of medium:

$$= \frac{0.0717}{0.00597 + 0.0717 + 0.1645} = 0.29$$

Probability of high:

$$= \frac{0.1645}{0.00597 + 0.0717 + 0.1645} = 0.679 = 0.68$$

Here, the probability of low, medium, and high from both the methods for the first evidence of node QDS has been calculated. But for the rest 26 evidence, only first method is applied because in our experiment triangular and trapezoidal membership functions have been used. The complete NPT of node QDS is shown in Table 1. The NPTs of numerical values are expressed in percentage (100 of scale). Similarly, the NPT of nodes QDP, ODPE, and PDD have been constructed and are shown in Tables 2, 3, 4.

Table 1 NPT of node QDS

RDSE	DSM	CP	High	Medium	Low
High	High	High	68	29	3
High	High	Medium	59	33	8
High	High	Low	45	35	20
High	Medium	High	58	32	10
High	Medium	Medium	45	39	16
High	Medium	Low	37	41	22
High	Low	High	44	35	21
High	Low	Medium	33	42	25
High	Low	Low	15.5	28.5	56
Medium	High	High	56	33	11
Medium	High	Medium	46	37	17
Medium	High	Low	21	42	37
Medium	Medium	High	34	52	14
Medium	Medium	Medium	31	46	23
Medium	Medium	Low	12	40	48
Medium	Low	High	35	44	21
Medium	Low	Medium	14	41	45
Medium	Low	Low	7.5	24	68.5
Low	High	High	41	36	23
Low	High	Medium	32	41	27
Low	High	Low	11	38	51
Low	Medium	High	35	44	21
Low	Medium	Medium	17	54	29
Low	Medium	Low	10	21	69
Low	Low	High	13.5	26	60.5
Low	Low	Medium	7	22	71
Low	Low	Low	1	10	89

Table 2 NPT of node QDP

QDS	DPF	High	Medium	Low
High	High	71	21	8
High	Medium	45	42	13
High	Low	21.6	37.4	41
Medium	High	34	48	18
Medium	Medium	20	66	14
Medium	Low	8.6	26.4	65
Low	High	23.7	35.3	41
Low	Medium	7.5	23	69.5
Low	Low	1	11	88

Table 3 NPT of node ODPE

QDP	EDP	High	Medium	Low
High	High	71	21	8
High	Medium	52	33	15
High	Low	25	55	20
Medium	High	34	48	18
Medium	Medium	20	66	14
Medium	Low	8	27	65
Low	High	15	26	59
Low	Medium	7	29	64
Low	Low	5	21	74

Table 4 NPT of PDD

ODPE	High	Medium	Low
High	5.76	10.38	83.86
Medium	11.45	65.62	22.93
Low	78.78	12.24	8.98

5 Validation of proposed method

To validate the constructed NPTs the following steps have been applied:

Step 1 Use of BBN tool Netica [21].

Step 2 Model construction in Netica.

Step 3 Apply the constructed NPTs to all the output nodes.

Step 4 Compilation of BBN model.

Step 5 Apply evidence on output of step 4, i.e., compiled mode of BBN model.

Step 6 Analyze obtained result from BBN model.

With the help of Netica tool, BBN model of software design and development sub-net as shown in Fig. 2 has been constructed. After model construction the constructed

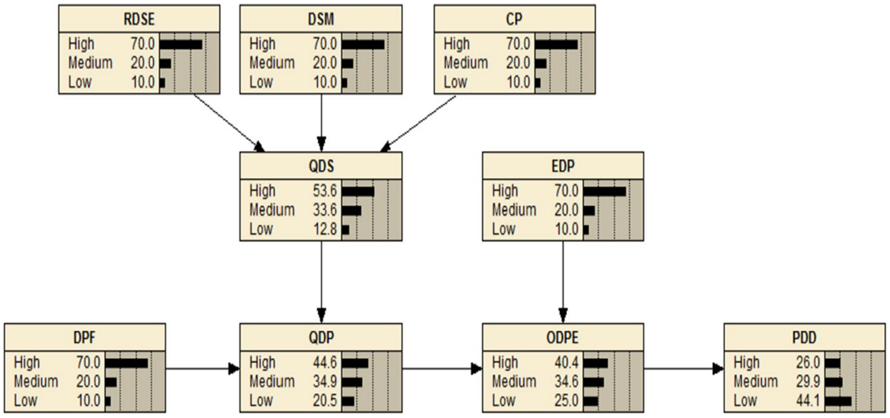


Fig. 14 Compile mode of BBN

Table 5 Qualitative value of software metrics

Case	RDSE	DSM	CP	DPF	EDP
Best case	High	High	High	High	High
Worst case	Low	Low	Low	Low	Low

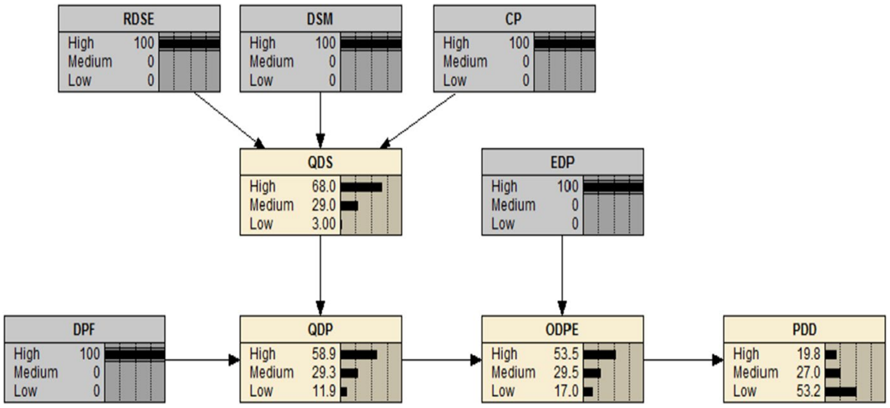


Fig. 15 Outcome of best-case scenario

NPTs are inserted to all the output nodes. The compilation process of BBN model is started using the Netica tool. The obtained complied mode of BBN model of software design and development subnet from the Netica tool is shown in Fig. 14.

Next, the best case and worst case of software metrics have been applied which are shown in Table 5 on compiled mode of BBN model to validate the correctness and

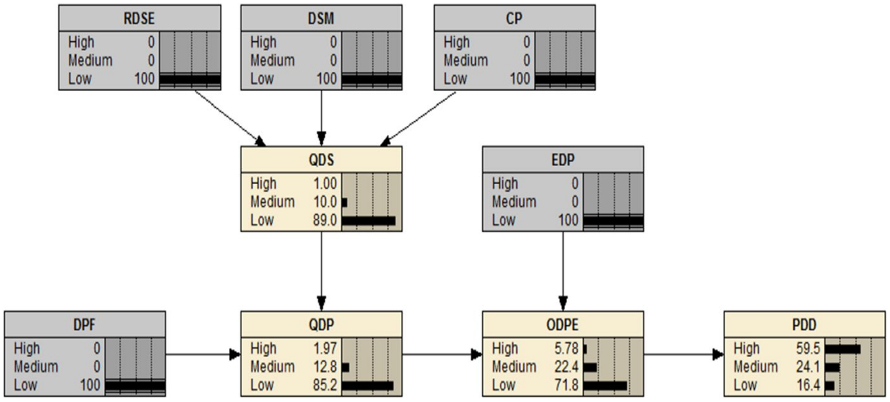


Fig. 16 Outcome of worst-case scenario

applicability of the proposed approach. The obtained outcomes are shown in Figs. 15 and 16.

5.1 Result analysis

Outcomes of BBN model after applying best- and worst-case scenario are shown in Figs. 15 and 16 which further shows the probability of defects in design and development is low (53.2) in the best-case scenario, whereas the probability of defects in design and development is high (59.5) in the worst-case scenario. Further, when the evidence of RDSE, DSM, and CP is applied, the probabilistic outcome of QDS is high (68.0%) in best-case scenario, whereas it is low (89%) in worst-case scenario. Similarly, when evidence of DPF is applied, the outcome of QDP is high (58.9%) in best-case scenario, whereas it is low (85.2%) in worst-case scenario. Next, applied the evidence of EDP the outcome of ODPE is high (53.5%) in best-case scenario, whereas it is low (71.8%) in worst-case scenario.

From the above observation it is clear to observe that the constructed NPTs are correct because the outcomes of all the output nodes (QDS, QDP, ODPE, and PDD) are affirmative in best-case scenario and adverse in worst-case scenario.

6 Conclusion and future scope

Generating a node probability table (NPT) in Bayesian belief networks (BBN) has been considered as NP-Hard problem, as the time complexity grows exponentially with increasing number of variables. Rich articles are available in scientific community that suggest various methods to solve this issue, however, mostly are problem specific and designed by considering special case(s). This paper presented a novel universal approach *NPT_fuzzy_logic*() to generate a NPT in BBN using fuzzy logic technique. The method proceeds with identifying casual concepts among the nodes in given graph $G(V, \underline{E})$, next generating causal relationship among nodes, defining membership function among input and output nodes, defining fuzzy "IF–THEN" rules, then performing fuzzy inference and defuzzification procedure to find the defuzzified area followed by calculating NPT based on premeditated defuzzified area. Fuzzy inference and defuzzification have been performed using Fuzzy Logic Toolbox™ provided in MATLAB®. For sake of simplicity and easy understanding the proposed method has been demonstrated with an illustrative example by considering the well-known BBN model of software design and development [8]. The proposed method has been validated by applying the BBN tool of Netica®. The result analysis section shows the significance of proposed method by considering the outcome of BBN model, after applying best- and worst-case scenario followed by probabilistic outcomes among causal concepts. The correctness of constructed NPT has been shown by considering the consequence of output node. The proposed approach will also be useful in domain experts-based NPT construction because fuzzy logic represents qualitative perception-based reasoning by "IF–THEN" fuzzy rules, which makes it easier for experts to express their judgment of NPT. As the proposed approach is not problem specific, thus may be applied universally in other problem domain. Further, in future the work can also be extended to other problem domain such as software development problems, agriculture area problems, environmental area based on availability of real-time dataset. Due to enormous applications of BBN in decision-making system, nature-based solutions, stakeholder engagement, reliability engineering and safety engineering, etc., the work will be extended to analyze and support the various circumstances such as analyzing the pipe failure in water/oil distribution system [11], stakeholder's knowledge to support nature-based solution implementation [2], software project management [1] etc.

Author contributions All authors contributed to the conception and design of the task performed in this paper. CK contributed by proposed scheme and performed the experiments. DKY contributed significantly to the study of related works. SKJ contributed to the writing, editing, and finalizing the concepts. SP and MP worked with the validation and result analysis. Code availability: The code generated during the current study is available from the corresponding author on reasonable request.

Availability of data and materials The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest There is no conflict of interest for the article.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication All authors reviewed the manuscript.

Informed consent There is no informed consent for the article.

References

1. Abraham de Sousa LR, de Souza CRB, Reis RQ (2022) A 20-year mapping of Bayesian belief networks in software project management. IET Softw 16(1):14–28

2. Albert S, Alessandro P, Rosa Coletta V, Umberto F, Raffaele G (2021) Bayesian Belief networks for integrating scientific and stakeholders' knowledge to support nature-based solution implementation. Front Earth Sci 9
3. Bellhouse DR (2004) The reverend Thomas Bayes, FRS: a biography to celebrate the tercentenary of his birth. Stat Sci 19(1):3–43
4. Cooper GF (1990) The computational complexity of probabilistic inference using Bayesian belief networks. Artif Intell 42:393–405
5. Díez FJ, Galán SF (2003) Efficient computation for the noisy MAX. Int J Intell Syst 18(2):165–177
6. Fenton NE, Neil M (1999) A critique of software defect prediction models. IEEE Trans Softw Eng 25(5):675–689
7. Fenton NE, Neil M, Caballero JG (2007) Using ranked nodes to model qualitative judgments in Bayesian networks. IEEE Trans Knowl Data Eng 19(10):1420–1432
8. Fenton N, Neil M, Marsh W, Hearty P, Radliński Ł, Krause P (2008) On the effectiveness of early life cycle defect prediction with Bayesian Nets. Empir Softw Eng 13(5):499–537
9. Huang K, Henrion M (2013) Efficient search-based inference for noisy-OR belief networks: TopEpsilon. arXiv preprint arXiv:1302.3584
10. Kaya M, Alhajj R (2003) A clustering algorithm with genetically optimized membership functions for fuzzy association rules mining. In: The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ'03, vol 2. IEEE, pp 881–886
11. Kayu T, David JP, Simon J (2019) Comparison of automatic and guided learning for Bayesian networks to analyse pipe failures in the water distribution system. Reliab Eng Syst Saf 186:24–36
12. Koski T, Noble JM (2009) Bayesian networks an introduction. Wiley, UK
13. Kumar C, Yadav DK (2015) A method for developing node probability table using qualitative value of software metrics. In: Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT). IEEE, pp 1–5
14. Kumar C, Yadav DK (2017) Software defects estimation using metrics of early phases of software development life cycle. Int J Syst Assur Eng Manag 8(4):2109–2117
15. Laitila P, Virtanen K (2016) Improving construction of conditional probability tables for ranked nodes in Bayesian networks. IEEE Trans Knowl Data Eng 28(7):1691–1705
16. Laitila P, Virtanen K (2022) Advancing construction of conditional probability tables of Bayesian networks with ranked nodes method. Int J Gen Syst 1–33
17. Lemmer JF, Gossink DE (2004) Recursive noisy OR-a rule for estimating complex probabilistic interactions. IEEE Trans Syst Man Cybern Part B (Cybern) 34(6):2252–2261
18. Mohanta S, Vinod G, Ghosh AK, Mall R (2011) A technique for early prediction of software reliability based on design metrics. Int J Syst Assur Eng Manag 2:261–281
19. Nadkarni S, Shenoy PP (2004) A causal mapping approach to constructing Bayesian networks. Decis Support Syst 38(2):259–281
20. Neapolitan RE (2004) Learning Bayesian networks. Pearson Prentice Hall, Upper Saddle River
21. Netica (2010) Available at http://www.norsys.com
22. O'Hagan A, Buck CE, Daneshkhah A, Eiser JR, Arthwaite PH, Jenkinson GDJ, Oakley JE, Rakow T (2006) Uncertain judgements: eliciting experts' probabilities. Wiley
23. Okutan A, Yıldız OT (2014) Software defect prediction using Bayesian networks. Empir Softw Eng 19(1):154–181
24. Pai GJ, Dugan JB (2007) Empirical analysis of software fault content and fault proneness using Bayesian methods. IEEE Trans Softw Eng 33(10):675–686
25. Pearl J (1988) Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan kaufmann
26. Perkusich M, Perkusich A, de Almeida HO (2013) Using survey and weighted functions to generate node probability tables for Bayesian networks. In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. IEEE, pp 183–188
27. Quintanar-Gago DA, Nelson PF (2021) The extended recursive noisy OR model: static and dynamic considerations. Int J Approx Reason 139:185–200
28. Ross TJ (2009) Fuzzy logic with engineering applications. Wiley
29. Santos E Jr, Wilkinson JT, Santos EE (2011) Fusing multiple Bayesian knowledge sources. Int J Approx Reason 52(7):935–947
30. Tang Z, McCabe BY (2007) Developing complete conditional probability tables from fractional data for Bayesian belief networks. ASCE

31. Yadav HB, Yadav DK (2017) Early software reliability analysis using reliability relevant software metrics. Int J Syst Assur Eng Manag 8(4):2097–2108
32. Yong H, Xiangzhou Z, Ngai EWT, Ruichu C, Mei L (2013) Software project risk analysis using Bayesian networks with causality constraints. Decis Support Syst 56:439–449
33. Zadeh LA (1996) Knowledge representation in fuzzy logic. In: Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi a Zadeh, pp 764–774
34. Zagorecki A, Druzdzel MJ (2004) An empirical study of probability elicitation under noisy-OR assumption. In: Flairs Conference, pp 880–886
35. Zhou Y, Fenton N, Neil M (2014) Bayesian network approach to multinomial parameter learning using data and expert judgments. Int J Approx Reason 55(5):1252–1268

Authors and Affiliations

Chandan Kumar[1] · Sudhanshu Kumar Jha[2] · Dilip Kumar Yadav[3] · Shiv Prakash[2] · Mukesh Prasad[4]

✉ Sudhanshu Kumar Jha
sudhanshukumarjha@gmail.com

Chandan Kumar
chandan.jha150286@gmail.com

Dilip Kumar Yadav
dkyadav1@gmail.com

Shiv Prakash
shivprakash@allduniv.ac.in

Mukesh Prasad
muskesh_prasad@uts.edu.au

[1] School of Computing, Amrita Vishwa Vidyapeetham, Amaravati, Andhra Pradesh 522503, India

[2] Department of Electronics and Communication, University of Allahabad, Prayagraj 211002, India

[3] Department of Computer Science and Engineering, NIT Jamshedpur, Jamshedpur 831014, India

[4] Faculty of Engineering and Information Technology, Australian Artificial Intelligence Institute, University of Technology, Sydney, PO Box 123, Broadway, Sydney, NSW 2007, Australia