applied sciences

Article

# Reliability of Social Networks on Activity-on-Node Binary-State with Uncertainty Environments

**Wei-Chang Yeh [1], Wenbo Zhu [2,*] and Chia-Ling Huang [3]**

1 Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu 300044, Taiwan
2 School of Mechatronical Engineering and Automation, Foshan University, Foshan 528000, China
3 Department of International Logistics and Transportation Management, Kainan University, Taoyuan 33857, Taiwan
* Correspondence: zhuwenbo@fosu.edu.cn

**Abstract:** Social networks (SNs) and many other industrial types of networks, structured by many nodes and relationships between nodes, have become an integral part of our daily lives. A binary-state network (BN) is often used to model structures and applications of SNs and other networks. The BN reliability is the probability that a BN functions continuously, i.e., that there is always a path between a specific pair of nodes. This metric is a popular index for designing, managing, controlling, and evaluating networks. The traditional BN reliability assumes that the network is activity-on-arc, and the reliability of each arc is known in advance. However, this is not always the case. Functioning components operate in different environments; moreover, a network might have newly installed components. Hence, the reliability of these components is not always known. To resolve the aforementioned problems, in which the reliability of some components of a network is uncertain, we introduce the fuzzy concept for the analysis of these components and propose a new algorithm to solve this uncertainty-component activity-on-node BN reliability problem. The time complexity of the proposed algorithm is analyzed, and the superior performance of the algorithm is demonstrated through examples.

**Keywords:** social networks (SNs); binary-state network (BN); network reliability; uncertainty environments; activity-on-node; binary-addition-tree algorithm (BAT)

check for updates

**Citation:** Yeh, W.-C.; Zhu, W.; Huang, C.-L. Reliability of Social Networks on Activity-on-Node Binary-State with Uncertainty Environments. *Appl. Sci.* **2022**, *12*, 9514. https://doi.org/10.3390/app12199514

Academic Editors: Christos Bouras and Dimitris Mourtzis

Received: 18 July 2022
Accepted: 19 September 2022
Published: 22 September 2022

## 1. Introduction

In social networks and many other industrial types of networks based on binary-state networks (BNs), each unreliable component has only two states. The BN architecture is one of the most common structures used to model issues in not only SNs [1,2] but also numerous industrial types of networks that have become an integral part of our daily life, such as networks involving multi-terminal network [3], transportation [4], grid/cloud computing [5,6], general network [7], Internet of things [8,9], network resilience problems [10,11], directed acyclic network [12], transition [13], and wireless sensor network [14]. Hence, an increasing amount of BN applications have recently emerged [15,16]. BNs thus play an important role in the planning, design, execution, management, and control of all the aforementioned systems [17–19].

Assume that $G(V, E, \mathbf{D})$ is a BN, where $\mathbf{D}$ is the state distribution of the components, and $V = \{1, 2, \ldots, n\}$ and $E = \{a_1, a_2, \ldots, a_m\}$ are the sets of nodes and arcs, respectively [20,21]. In the activity-on-arc (AOA) BN, each node is considered perfect, whereas each arc may fail based on a predefined probability in $\mathbf{D}$ [22,23]. On the contrary, in the activity-on-node (AON) BN, each arc is considered perfect, whereas each node (except the source node 1 and the sink node $n$) may fail based on a predefined probability in $\mathbf{D}$ [24–28].

Note that a BN can be a mixture of AOA and AON; and a component can be a node, an arc, cable, the Internet, machine, process station, workstation, tool, unit, or worker [29]. In this study, we focus on the AON—-and the component is a node in the AON, and an arc in the AOA.

For example, Figure 1 shows an AOA BN $G(V, E, \mathbf{D})$, where $V = \{1, 2, 3, 4, 5\}$, $E = \{a_1 = e_{1,2}, a_2 = e_{1,3}, a_3 = e_{1,4}, a_4 = e_{2,4}, a_5 = e_{2,5}, a_6 = e_{3,5}\}$. Node 1 is the source node and node 5 is the sink node; the state distribution **D** is listed in Table 1.

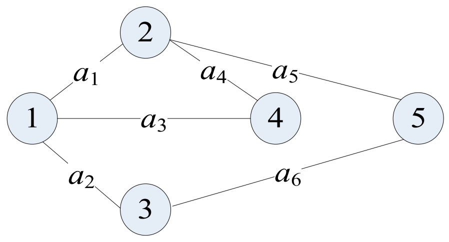

**Figure 1.** Example of an AOA BN with 5 nodes.

**Table 1.** State distributions in Figure 1.

| $a$ | $\mathbf{D}(a)$ |
|---|---|
| $a_1$ | 0.85 |
| $a_2$ | 0.80 |
| $a_3$ | 0.85 |
| $a_4$ | 0.80 |
| $a_5$ | 0.75 |
| $a_6$ | 0.90 |

The traditional reliability, $R$, of BNs is defined as the success probability that there is at least one simple path from node 1 to node $n$. It is one of the most popular indices for evaluating the performance of real-world BNs. For example, the reliability of propagation is studied in SNs based on BN [1]. An algorithm is proposed for the transportation BN [4]. A binary-code genetic algorithm and the simplified swarm optimization (SSO) is hybrid to optimize the grid BN reliability [6]. The SSO is employed to optimize the BN reliability in the Internet of Things [8]. The telecom BN reliability is evaluated by considering network resiliency [10]. The BN reliability is solved using the unscented transformation [13]. Additionally, a squeezed artificial neural network is proposed to evaluate the BN reliability [25].

In the traditional BN, the reliability of each component must be known in advance [19,22], for example, the reliability $\mathbf{D}(a)$ is already known in Table 1 for all $a \in E$. It is NP-Hard and #P-Hard to calculate the BN reliability [17–19]. Various algorithms have been proposed to calculate BN reliability and can be categorized into direct and indirect algorithms [30–33]. The former, for example, the quick binary-addition-tree algorithm (BAT) [34] and binary-decision diagram (BDD) [35], can calculate the BN reliability directly; the latter is separated further into two main steps:

1. Find all simple paths or cuts based on path-based algorithms [5,8,9,21] or cut-based algorithms [20,22,23], respectively; and
2. Calculate the BN reliability using inclusion-exclusion technology (IET) [36–38] or the sum-of-disjoint product (SDP) [39–42].

The direct algorithm is more efficient than the indirect algorithm in calculating the BN reliability because the two main steps in all indirect algorithms are both NP-hard and #P-hard [34]. In addition, regardless of whether we use direct or indirect algorithms, the component is an arc, and the reliability of each component must be known in advance [29–31,38].

In real life, it is not practical or reasonable to assume that the reliability of each component is known precisely beforehand [43–45]. For example, the Mars Rover is a novel vehicular system, and we cannot know the exact reliability of each component in an environment that is completely different from Earth. Hence, the goal of this study is to overcome the aforementioned obstacle of uncertainty by proposing a new algorithm to deal with uncertain nodes using fuzzy set theory [46,47] and the binary-addition-tree algorithm (BAT) [11,29,48–52].

It is well known that fuzzy set theory can effectively resolve uncertain and imprecise problems. The BAT, first proposed by Yeh [48], can find all feasible and required vectors simply based on binary addition. Its current version, quick BAT [34], is more efficient compared to BDD and all other indirect algorithms in calculating BN reliability. Hence, the proposed algorithm is based on BAT combined with fuzzy set theory to solve the proposed uncertainty-component BN reliability problem owing to its importance and practicality.

The paper's contribution is highlighted in the following points:

1. An activity-on-node (AON) BAT-based algorithm is proposed for computing the reliability of a binary-state network instead of the traditional BN reliability assuming that the network is activity-on-arc (AOA).
2. In real life, it is not practical or reasonable to assume that the reliability of each component is known precisely beforehand. Nevertheless, in the traditional BN, the reliability of each component must be known in advance. Thus, one of this study's contributions is to overcome the aforementioned obstacle of uncertainty by proposing a new algorithm to deal with uncertain nodes using fuzzy set theory and the binary-addition-tree algorithm (BAT).

The remainder of this paper is organized as follows. All acronyms, notations, nomenclatures, and assumptions are introduced in Section 2. A review of AOA BAT, AOA path-based layered-search algorithm (PLSA), and some background material on fuzzy set theory is given in Section 3. The proposed BAT-based algorithm is introduced formally in Section 4, including the preprocessing steps of transferring the linguistic variable via many stages to the unreliability, and the proposed AON PLSA to verify the connectivity of each vector obtained from the proposed BAT. The performance of the proposed algorithm in solving the proposed problem is demonstrated as an example. Finally, Section 5 concludes the study.

## 2. Acronyms, Notations, Nomenclatures, and Assumptions

All needed acronyms, notations, assumptions, and nomenclatures are defined, provided, introduced, and explained here.

### 2.1. Acronyms, Notations, and Nomenclatures

| Acronyms | |
|---|---|
| BAT: | Binary-addition-tree algorithm, which is an algorithm to search all possible binary-state (0 or 1) vectors in the network [48]. |
| BN: | Binary-state flow network that each unreliable component has only two states (0 or 1) in the network. |
| MP: | Minimal path means the path from the source node to the sink node is possible, but it is a failure if deleting any arc from this path. |
| MC: | Minimal cut, which is a cut set and satisfies minimality, that is, it is not a cut set after removing one element (arc) from this cut set. |
| DFS: | Depth-search-first algorithm, which is preferential depth search of the branches of the tree for the searching tree. |
| BFS: | Breadth-search-first algorithm, which is preferential breadth search of the branches of the tree for the searching tree. |
| BDD: | Binary-decision-diagram, which is a procedure made up of a series two state shift directions. |
| IET: | Inclusion-exclusion technology, which is an approach to directly compute the solution by including the relations of elements and consider excluding the relationship between elements that affects the solutions obtained. |

| Acronyms | |
|---|---|
| UGFM: | Universal generating function methodology, which is an intuitionally recursive methodology to find out the solutions by node UGF and subnet UGF. |
| PLSA: | Path-based layered-search algorithm, which is an algorithm to verify the connectivity of each vector obtained from BAT. |
| AOA: | Activity-on-arc that the nodes present the events connecting by the arcs. |
| AON: | Activity-on-node that the arcs present the events connected by the nodes. |
| AFN: | Average fuzzy number, which is a method to combine the linguistic variables into one fuzzy number. |
| FPS: | Fuzzy possibility score, which represents the experts' belief of the most likely score of an event occurring by transferring AFN to FPS based on the left and right fuzzy ranking method. |
| FFR: | Fuzzy failure rating = (frequency of error)/(total chance that an event has an error). |
| **Notations** | |
| $\vert \bullet \vert$: | Number of elements in set $\bullet$. |
| $n$: | Number of nodes in network. |
| $m$: | Number of arcs in network. |
| $a_k$: | The $k$th undirected arc in network. |
| $V$: | Set of nodes $V = \{1, 2, \ldots, n\}$ and $\vert V \vert = n$ in network. |
| $E$: | Set of arcs $E = \{a_1, a_2, \ldots, a_m\}$ and $\vert E \vert = m$ in network. |
| $e_{i,j}$: | $e_{i,j} = a_k \in E$, $i, j \in V$, and for one and only one $k$. |
| $\Pr(\bullet)$: | Occurrence probability of set $\bullet$. |
| $\mathbf{D}$: | State distributions listing all components' states and their probabilities, e.g., Table 1 is the arc state distribution of Figure 1. |
| $G(V, E)$: | A graph created by $V$ and $E$. For example, Figure 1 shows a network constituted by set of arcs $E = \{a_1, a_2, \ldots, a_6\}$ and set of nodes $V = \{1, 2, 3, 4, 5\}$. |
| $G(V, E, \mathbf{D})$: | A network constituted by $G(V, E)$ and $\mathbf{D}$. For example, $G(V, E)$ shown in Figure 1 and $\mathbf{D}$ in Table 1 form a network. |
| $X$: | Binary state vector: its $k$th coordinate is the state of the $k$th component. |
| $x_k$: | State of the $k$th coordinate in $X$. |
| $\Pr(X)$: | Occurrence probability of vector $X$. |
| $R$: | Reliability of a BN |
| $G(X)$: | $G(X) = G(V, \{a \in E \mid \text{each arc } a \text{ with } X(a) > 0\}, \mathbf{D})$. |
| $X << Y$: | If vector $X$ is obtained before $Y$ in the BAT. |
| **Nomenclatures** | |
| MP/MC: | An MP/MC is an arc subset; each of its proper subsets is not an MP/MC [17,19,25]. For example, $\{a_1, a_3, a_6\}$ is an MC, and $\{a_3, a_4, a_5\}$ is an MP in Figure 1. |
| Uncertainty component: | The reliability of the component is uncertain. |
| Crisp component: | The reliability of the component is certain. |
| $R$: | Success probability of at least a directed path from nodes 1 to $n$. |

### *2.2. Assumptions*

1. The networks have no loops or parallel arcs.
2. Each node and arc are perfectly reliable in AOA BN and AON BN, respectively.
3. The source node and the sink node, that is, nodes 1 and $n$, are perfectly reliable.
4. Each arc is undirected.
5. The states of the components are statistically independent, and its state follows $\mathbf{D}$ to be zero or one.

## 3. Overview of BAT and Fuzzy Set

The proposed AON BAT is modified from the traditional AOA (binary-state) BAT [45] to find each vector, say $X$. It is also used to verify whether $X$ is connected using PLSA [45] after transferring each uncertainty component with fuzzy and incomplete information to a crisp component to calculate the uncertainty-component binary-state AON reliability. Hence, the traditional AOA BAT, AOA PLSA, and fundamental fuzzy set-related parts are discussed in this section.

### *3.1. AOA BAT*

(AOA) BAT was first proposed by Yeh [48]. Based on binary addition, it is used to solve AOA binary-state network reliability problems. Owing to its simplicity, efficiency, and memory saving capability, BAT has been implemented in various areas. It has recently become a new search method for finding all possible solutions or vectors. For example, BATs have been implemented to solve rework problems [29], network reliability problems [34,48,51,52], wildfire spread prediction [49], and forecasting of computer virus propagation [50].

From the results of numerical experiments evaluating the running time and computer memory, BAT [48] was found to outperform depth-search-first algorithms (DFS), for example, the quick IET [37], which is the best-known algorithm for calculating MFN reliability; the breadth-search-first algorithm (BFS), the universal generating function methodology (UGFM) [19], which is the main algorithm for solving information network reliability problems [53]; and binary-decision diagram (BDD) [35], which outperforms other algorithms in binary-state network reliability problems [17,18].

The purpose of the original BAT is to search all possible vectors for related AOA problems [48]. BAT starts from a vector zero, say $X$, and adds one to $X$ to update $X$ iteratively. Let us briefly explain the basic principles of BAT. The entire original BAT is processed in a backward manner from the last coordinate of $X$ to the first coordinate [48]. Let $x_k$ be the current coordinate. If $x_k = 0$, $x_k$ is updated by letting $x_k = 1$. A new vector is formed, and the current coordinate is reset to $m$, e.g., (1, 0, 0) is updated to (1, 0, 1). If $x_k = 1$, $x_k$ is updated by letting $x_k = 0$. The process is then moved to the $(k - 1)$th coordinate to repeat the aforementioned procedure until the new current coordinate is zero, e.g., (1, 0, 1) is updated to (1, 1, 0).

The original AOA BAT pseudocode is as following Algorithm 1 [48]:

**Algorithm 1:** AOA BAT

```
Input:    $m$.
Output:   All solutions $X$ of the problem.
          Let SUM = 0, $X$ = $\mathbf{0}$, and $k = m$.
          while SUM < $m$
            if $X(a_k) = 1$
              SUM = SUM − 1
              $X(a_k) = 0$
            else
              SUM = SUM + 1
              $X(a_k) = 1$
              if $k > 1$
                $k = k - 1$
              end
            end
          end
```

From the aforementioned five lines of pseudocode, it can be observed that BAT is simple to program, efficient to run, easy to understand, economical in RAM, and suitable for make-to-fit [11,29,48–52].

The number of all obtained vectors is $2^m$ for an $m$-tuple vector zero in STEP B0. The time complexity of BAT is $O(2^{m+1})$, as proved in [52]. Furthermore, the total computer memory is only $O(m)$ because the current vector $X$ is used repeatedly, and a new updated vector needs to compute only its related values or predefined functions [48].

The following table shows how the AOA BAT is implemented to find all the state vectors in Figure 1. Because $|E| = 6$ in Figure 1, there are $2^6 = 64$ vectors at the end. To recognize the sequence of the obtained vectors easily, the notation $X_i$ is used in Table 2. However, in the actual BAT procedure, there is always one vector $X$ in the complete process; there is no need to discriminate the sequence of the obtained $X$.

**Table 2.** Complete results obtained from the BAT in Figure 1.

| $i$ | $X_i$ | $i$ | $X_i$ | $i$ | $X_i$ | $i$ | $X_i$ |
|---|---|---|---|---|---|---|---|
| 1 | (0, 0, 0, 0, 0, 0) | 17 | (0, 0, 0, 0, 1, 0) | 33 | (0, 0, 0, 0, 0, 1) | 49 | (0, 0, 0, 0, 1, 1) |
| 2 | (1, 0, 0, 0, 0, 0) | 18 | (1, 0, 0, 0, 1, 0) | 34 | (1, 0, 0, 0, 0, 1) | 50 | (1, 0, 0, 0, 1, 1) |
| 3 | (0, 1, 0, 0, 0, 0) | 19 | (0, 1, 0, 0, 1, 0) | 35 | (0, 1, 0, 0, 0, 1) | 51 | (0, 1, 0, 0, 1, 1) |
| 4 | (1, 1, 0, 0, 0, 0) | 20 | (1, 1, 0, 0, 1, 0) | 36 | (1, 1, 0, 0, 0, 1) | 52 | (1, 1, 0, 0, 1, 1) |
| 5 | (0, 0, 1, 0, 0, 0) | 21 | (0, 0, 1, 0, 1, 0) | 37 | (0, 0, 1, 0, 0, 1) | 53 | (0, 0, 1, 0, 1, 1) |
| 6 | (1, 0, 1, 0, 0, 0) | 22 | (1, 0, 1, 0, 1, 0) | 38 | (1, 0, 1, 0, 0, 1) | 54 | (1, 0, 1, 0, 1, 1) |
| 7 | (0, 1, 1, 0, 0, 0) | 23 | (0, 1, 1, 0, 1, 0) | 39 | (0, 1, 1, 0, 0, 1) | 55 | (0, 1, 1, 0, 1, 1) |
| 8 | (1, 1, 1, 0, 0, 0) | 24 | (1, 1, 1, 0, 1, 0) | 40 | (1, 1, 1, 0, 0, 1) | 56 | (1, 1, 1, 0, 1, 1) |
| 9 | (0, 0, 0, 1, 0, 0) | 25 | (0, 0, 0, 1, 1, 0) | 41 | (0, 0, 0, 1, 0, 1) | 57 | (0, 0, 0, 1, 1, 1) |
| 10 | (1, 0, 0, 1, 0, 0) | 26 | (1, 0, 0, 1, 1, 0) | 42 | (1, 0, 0, 1, 0, 1) | 58 | (1, 0, 0, 1, 1, 1) |
| 11 | (0, 1, 0, 1, 0, 0) | 27 | (0, 1, 0, 1, 1, 0) | 43 | (0, 1, 0, 1, 0, 1) | 59 | (0, 1, 0, 1, 1, 1) |
| 12 | (1, 1, 0, 1, 0, 0) | 28 | (1, 1, 0, 1, 1, 0) | 44 | (1, 1, 0, 1, 0, 1) | 60 | (1, 1, 0, 1, 1, 1) |
| 13 | (0, 0, 1, 1, 0, 0) | 29 | (0, 0, 1, 1, 1, 0) | 45 | (0, 0, 1, 1, 0, 1) | 61 | (0, 0, 1, 1, 1, 1) |
| 14 | (1, 0, 1, 1, 0, 0) | 30 | (1, 0, 1, 1, 1, 0) | 46 | (1, 0, 1, 1, 0, 1) | 62 | (1, 0, 1, 1, 1, 1) |
| 15 | (0, 1, 1, 1, 0, 0) | 31 | (0, 1, 1, 1, 1, 0) | 47 | (0, 1, 1, 1, 0, 1) | 63 | (0, 1, 1, 1, 1, 1) |
| 16 | (1, 1, 1, 1, 0, 0) | 32 | (1, 1, 1, 1, 1, 0) | 48 | (1, 1, 1, 1, 0, 1) | 64 | (1, 1, 1, 1, 1, 1) |

### 3.2. AOA PLSA

To solve network reliability problems in the traditional BAT, the connectivity of each obtained vector found in the BAT procedure is verified using AOA PLSA [48]. Consequently, the summation of the probabilities of all connected vectors is related to network reliability.

Each vector, say $X$, obtained from BAT is a subgraph $G(V, X)$ of $G(V, E)$. For example, the subgraph corresponding to $X_{28} = (1, 1, 0, 1, 1, 0)$ is depicted in Figure 2.

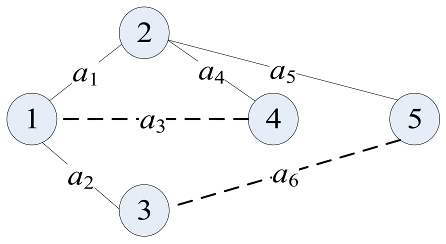

**Figure 2.** Subgraph corresponding to $X_{28} = (1, 1, 0, 1, 1, 0)$.

The layered-search algorithm (LSA) was first proposed in [54] and revised to PLSA to verify the connectivity of each vector obtained from BAT [48].

To apply the PLSA to test whether $X$ is connected, i.e., whether there is a simple path from nodes 1 to $n$ in $G(V, X)$, an empty queue is initialized by adding node 1 to the queue. This queue stores the nodes that are adjacent to any node in the queue by at least one arc repeatedly. If node $n$ is included, a simple path is found from nodes 1 to $n$, that is, such a vector is connected. Otherwise, $X$ is a disconnected vector. The pseudocode of the AOA PLSA is as following Algorithm 2:

**Algorithm 2:** AOA PLSA.

```
Input:    A vector $X$ obtained from BAT.
Output:   Whether $X$ is connected or disconnected
            while $Q_i \neq \emptyset$
            V* = $Q_0$ = {1}
            $i$ = 1
            $Q_i$ = { v ∈ V* | for all $e_{u,v}$ ∈ E, and u ∈ V*}
            if $n \in Q_i$ or $Q_i = \emptyset$
            halted
            else
                V* = V* ∪ $Q_i$
                $i = i + 1$
            end
          end
```

The time complexity of the PLSA to verify whether a vector is connected is $O(n)$. For example, in Figure 2, the PLSA determines $X$ = (1, 1, 0, 1, 1, 0) to be connected, and the procedure is listed in Table 3.

**Table 3.** Process from the proposed PLSA for Figure 2.

| $i$ | $Q_i$ | $Q_{i+1}$ | $V^*$ | Remark |
|---|---|---|---|---|
| 0 | {1} | {2, 3} | {1, 2, 3} | |
| 1 | {2, 3} | {5} | | $X$ is connected |

## 3.3. Fuzzy Set Theory

The fuzzy set theory introduced by Zadeh has become a common technique for handling uncertain problems with imprecise information [46]. The basic principle of fuzzy set theory is briefly presented in this section.

### 3.3.1. Fuzzy Number

Let $x$ be a real number allocated to $\underset{\sim}{A}$, and $\underset{\sim}{A} \subset$ the universe set $U$. The fuzzy set $\underset{\sim}{A}$ is defined as follows [43]:

$$\underset{\sim}{A} = \{(x, u_{\underset{\sim}{A}}(x)) | \ x \in X\} \tag{1}$$

where $u_{\underset{\sim}{A}}(x)$: $X \to [0, 1]$ denotes the membership function of fuzzy set $A$.

There are many forms of fuzzy numbers that represent fuzzy sets. In this study, triangular fuzzy numbers are applied. A triangular fuzzy number $\underset{\sim}{A}$ = $(a, b, c)$, where $a \leq b \leq c$, is defined in Equation (2); it is depicted in Figure 3 [46]:

$$u_{\underset{\sim}{A}}(x) = \begin{cases} (x-a)/(b-a), & a \leq x \leq b, \\ (x-c)/(b-c), & b \leq x \leq c, \\ 0, & \text{otherwise}, \end{cases} \tag{2}$$

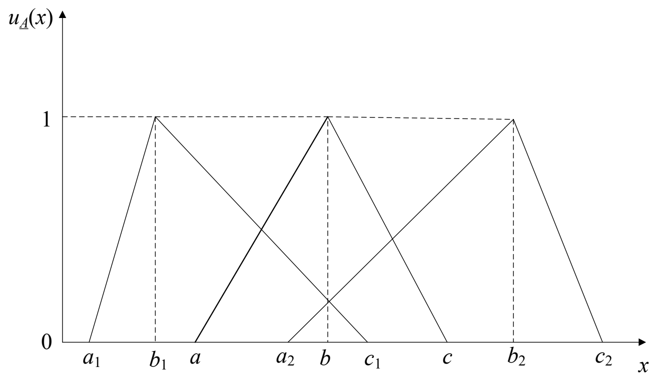

**Figure 3.** Example of the triangular fuzzy number: $\underset{\sim}{A}_1 = (a_1, b_1, c_1)$, $\underset{\sim}{A} = (a, b, c)$, and $\underset{\sim}{A}_2 = (a_2, b_2, c_2)$.

Let $\underset{\sim}{A}_1 = (a_1, b_1, c_1)$ and $\underset{\sim}{A}_2 = (a_2, b_2, c_2)$ be two triangular fuzzy numbers. The fuzzy addition, subtraction, multiplication, and division of triangular fuzzy numbers are as shown in Equations (3)–(6), respectively [46]:

$$\underset{\sim}{A}_1 \oplus \underset{\sim}{A}_2 = (a_1 + a_2, b_1 + b_2, c_1 + c_2) \tag{3}$$

$$\underset{\sim}{A}_1 \ominus \underset{\sim}{A}_2 = (a_1 - c_2, b_1 - b_2, c_1 - a_2) \tag{4}$$

$$\underset{\sim}{A}_1 \otimes \underset{\sim}{A}_2 = (a_1 \times a_2, b_1 \times b_2, c_1 \times c_2) \tag{5}$$

$$\underset{\sim}{A}_1 \odot \underset{\sim}{A}_2 = (a_1/c_2, b_1/b_2, c_1/a_2) \tag{6}$$

The $\alpha$-cut is another common way to represent the fuzzy set; it is also an important approach for transforming a fuzzy membership function into a crisp subset [46]. The $\alpha$-cut is a horizontal representation of fuzzy sets. Let $\underset{\sim}{A}_{\alpha,UB}$ and $\underset{\sim}{A}_{\alpha,LB}$ be the upper and lower bounds of the $\alpha$-cut, respectively. The $\alpha$-cuts of fuzzy set $\underset{\sim}{A}$ can be defined as in Equation (3) [46]:

$$\underset{\sim}{A}_\alpha = \{(x, u_{\underset{\sim}{A}}(x) \geq \alpha) | x \in U\} = [\underset{\sim}{A}_{\alpha,LB}(\alpha)\, \underset{\sim}{A}_{\alpha,UB}(\alpha)], \tag{7}$$

For example, as shown in Figure 4, the $\alpha$-cut for the triangular fuzzy number $\underset{\sim}{A} = (a, b, c)$ is:

$$\underset{\sim}{A}_\alpha = [\alpha(b - a) + a, \alpha(b - c) + c]. \tag{8}$$

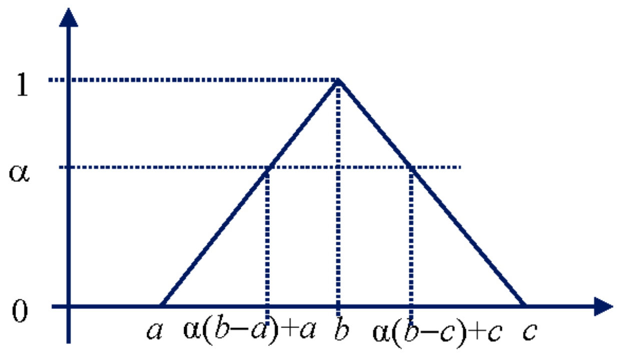

**Figure 4.** $\alpha$-cut for a triangular fuzzy number $\underset{\sim}{A} = (a, b, c)$.

### 3.3.2. Linguistic Variable

It is natural to use linguistic variables to estimate, rate, and judge subjective events. In this study, each uncertainty component had to be rated by experts using linguistic variables.

This is shown in the first column of Table 4. To fuzzify the experts' ratings to measure the reliability of each uncertainty component, the linguistic variables presented in Table 4 are transferred to their approximate reasoning of triangular fuzzy numbers, as shown in the second column of Table 4 [46,47].

**Table 4.** Linguistic variables and their fuzzy numbers.

| Linguistic Variables | Triangular Fuzzy Numbers |
|---|---|
| Very Low (VL) | (0.0, 0.0, 0.1) |
| Low (L) | (0.0, 0.1, 0.3) |
| Fairly Low (FL) | (0.1, 0.3, 0.5) |
| Medium (M) | (0.3, 0.5, 0.7) |
| Fairly High (FH) | (0.5, 0.7, 0.9) |
| High (H) | (0.7, 0.9, 1.0) |
| Very High (VH) | (0.9, 1.0, 1.0) |

For example, assume that components 3, 4, and 5 are uncertainty nodes, and six experts are asked to rate these components using linguistic variables. The results of these linguistic variables are listed in Table 5.

**Table 5.** Linguistic variables of six experts.

| Component | Linguistic Variables |
|---|---|
| 3 | VL, L, L, VL, L, L |
| 4 | VL, L, H, VH, L, H |
| 5 | VH, H, L, VH, H, L |

### 3.3.3. Average Fuzzy Number

After obtaining the linguistic variables for these components with uncertain information, we need to aggregate the experts' rates; that is, combine the linguistic variables into one fuzzy number called the average fuzzy number (AFN), by averaging the fuzzy number of all experts' linguistic variables based on the $\alpha$-cut [47].

Assume that $\utilde{A}_{i,j} = (a_{i,j}, b_{i,j}, c_{i,j})$ is the linguistic variable provided by expert $j$ for uncertainty component $i$. The average fuzzy number of the uncertainty component $i$ is defined as:

$$\utilde{A}_i = (\utilde{A}_{i,1} \oplus \utilde{A}_{i,2} \oplus \ldots \oplus \utilde{A}_{i,h})/h \tag{9}$$

where $h$ is the number of experts.

For example, six experts rated the uncertainty component 3: VL, L, L, VL, L, and L. Based on Equation (3), Equation (9), and Tables 4 and 5, we obtain the AFN for the uncertainty component 3:

$$\utilde{A}_3 = [2(0.0,\ 0.0,\ 0.1) + 4(0.0,\ 0.1,\ 0.3)]/6 = (0.0,\ 0.4/6,\ 1.4/6). \tag{10}$$

### 3.3.4. Defuzzification to Convert AFN to FPS

The AFN is still a fuzzy number and requires further defuzzification to convert it to a crisp number called the fuzzy possibility score (FPS), which represents the experts' belief of the most likely score of an event occurring. The method of transferring AFN to FPS is based on the left and right fuzzy ranking method proposed in [55]. To implement this method, we need to define the fuzzy maximizing and minimizing sets as:

$$f_{\max}(x) = \begin{cases} x & x \in [0,1] \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$$f_{\min}(x) = \begin{cases} 1-x & x \in [0,1] \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Based on the α-cut, we have $\text{FPS}_\text{R}(k)$ and $\text{FPS}_\text{L}(k)$ for uncertainty component $k$ [55]:

$$\text{FPS}_\text{R}(k) = \sup_x [f_k(x) \wedge f_{\max}(x) = \alpha, \text{ where } \alpha = \underset{\sim}{A}_{\alpha,\text{UB}}(\alpha). \tag{13}$$

$$\text{FPS}_\text{L}(k) = \sup_x [f_k(x) \wedge f_{\min}(x) = \alpha, \text{ where } (1-\alpha) = \underset{\sim}{A}_{\alpha,\text{LB}}(\alpha), \tag{14}$$

After solving α and bringing α into the following equation, we can obtain the FPS for the uncertainty component $k$ [55]:

$$\text{FPS}(k) = |\text{FPS}_\text{R}(k) + 1 - \text{FPS}_\text{L}(k)|/2. \tag{15}$$

For example, the α-cut for $\underset{\sim}{A}_3 = (0.0, 0.4/6, 1.4/6)$ is:

$$\underset{\sim}{A}_{3,\alpha} = [\alpha(0.4/6 - 0) + 0, \alpha(0.4/6 - 1.4/6) + 1.4/6] = [0.4\alpha/6, 1.4/6 - \alpha/6]. \tag{16}$$

After solving Equations (13) and (14), we have:

$$\alpha = \underset{\sim}{A}_{3,\alpha,\text{UB}}(\alpha) \Rightarrow \alpha = 1.4/6 - \alpha/6 \Rightarrow \alpha = \text{FPS}_\text{R}(3) = 0.2 \tag{17}$$

$$(1-\alpha) = \underset{\sim}{A}_{3,\alpha,\text{LB}}(\alpha) \Rightarrow (1-\alpha) = 0.4\alpha/6 \Rightarrow \alpha = \text{FPS}_\text{L}(3) = 0.9375. \tag{18}$$

Hence,

$$\text{FPS}(3) = |\text{FPS}_\text{R}(3) + 1 - \text{FPS}_\text{L}(3)|/2 = 0.86875. \tag{19}$$

### 3.3.5. Further Conversion of FPS to FFR

To confirm the affinity between the non-fuzzy failure rate of hardware and experts' FPS, the FPS must be converted into fuzzy failure rates (FFRs), which are an error rate [56]:

$$\text{FFR} = (\text{frequency of error})/(\text{total chance that an event has an error}). \tag{20}$$

$$= \begin{cases} 10^{-k} & \text{FPS} \neq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{21}$$

where:

$$k = |(1 - \text{FPS})/\text{PFS}|^{1/3} \times 2.301. \tag{22}$$

The FFR of an uncertainty component, say $i$, is treated as the unreliability; that is, $R(i) = 1 - \text{FFR}(i)$. For example, because:

$$k = |(1 - 0.86875)/0.86875|^{1/3} \times 2.301 = 1.225514. \tag{23}$$

We have the FFR of uncertainty component 3:

$$\text{FFR}(3) = 10^{-k} = 10^{-1.225514} = 0.059496, \tag{24}$$

and:

$$\text{R}(3) = 1 - \text{FFR}(3) = 0.940504. \tag{25}$$

## 4. Proposed AON BAT-Based Algorithm

The proposed AON BAT-based algorithm for computing the reliability of a binary-state network with uncertainty nodes is presented in this section. The preprocessing steps for transferring uncertainty nodes into crisp nodes are described in Section 4.1. Moreover, the first AON BAT is proposed by modifying the traditional AOA BAT in Section 4.2. An example of the proposed AON BAT is presented in Section 4.3.

### 4.1. Preprocessing Operations: Transfer of Uncertainty Components into Crisp Components

The preprocessing steps of the proposed algorithm include dealing with uncertainty nodes whose reliabilities are unknown. The basic concept of transferring uncertainty nodes into crisp nodes is based on fuzzy set theory [46], as mentioned in Section 3.2. The pseudocode for the proposed preprocessing operation is as following Algorithm 3 [47]:

```
Algorithm 3: Preprocess Uncertainty Components
Input:    An AON BN $G(V, E, \mathbf{D})$ with uncertainty nodes.
Output:   Convert each uncertainty node into a crisp node.
          Each expert rates the uncertainty nodes using linguistic variables.
          Transfer each linguistic variable into a fuzzy number.
          Average experts' fuzzy numbers and calculate their related $\alpha$-cuts for each
          uncertainty node to have an AFN [47].
          Convert AFN to FPS with defuzzification [55].
          Convert FPS to FFR [56]—-which is the unreliability.
```

Taking Table 6 as an example of which nodes 3–5 are uncertainty nodes, in STEP P0, experts first rate these uncertainty nodes using linguistic variables, as shown in Table 4 and the second column of Table 6. Then, STEP P2 calculates the average fuzzy number (AFN) based on Table 4 and STEP P1, as shown in the third column of Table 6.

**Table 6.** Example of STEPs P0, P1, and P2.

| Node *i* | Linguistic Variables | Average Fuzzy Number | $\alpha$-Cut |
|---|---|---|---|
| 3 | VL, L, L, VL, L, L | (0, 0.4/6, 1.4/6) | $[0.4\alpha/6, 1.4/6 - \alpha/6]$ |
| 4 | VL, L, H, VH, L, H | (2.3/6, 3/6, 3.7/6) | $[0.7\alpha/6 + 2.3/6, 3.7/6 - 0.7\alpha/6]$ |
| 5 | VH, H, L, VH, H, L | (3.2/6, 4/6, 4.6/6) | $[0.8\alpha/6 + 3.2/6, 4.6/6 - 0.1\alpha]$ |

STEP P2 mainly focused on aggregating experts' ratings into an AFN. Subsequently, each $\alpha$-cut of the related AFN is also required to be calculated, as shown in the last column of Table 6.

After many fuzzy numbers are aggregated into an AFN and the related $\alpha$-cut, the next step is to defuzzify the values. Based on the left and right fuzzy ranking method proposed in [55], STEP P3 defuzzifies AFN to FPS, as shown in columns 2 to 4 of Table 7. The last step, that is, STEP P4, transfers APS to FFR based on Equation (21), as listed in the last two columns of Table 7. Consequently, the FFR is treated as the unreliability of the related uncertainty node.

**Table 7.** Example of STEPs P3 and P4.

| Node *i* | $\text{FPS}_L$ (*i*) | $\text{FPS}_R$ (*i*) | FPS (*i*) | *k* | FFR (*i*) | R (*i*) |
|---|---|---|---|---|---|---|
| 3 | 0.2 | 0.9375 | 0.86875 | 1.225514 | 0.059496 | 0.940504 |
| 4 | 0.552239 | 0.552239 | 0.5 | 2.301 | 0.005 | 0.995000 |
| 5 | 0.411764 | 0.696970 | 0.642603 | 1.892283 | 0.012815 | 0.987185 |

### 4.2. AON BAT

The proposed AON BAT is based on the traditional AOA BAT proposed in [48]. In the proposed AON BAT, the values of the first and last coordinates, that is, the source node and the sink node, are always 1 because the BN fails if any one of these two nodes fails.

Unlike the traditional AOA BAT that is implemented backward from the last to the first coordinate, the proposed AON BAT is implemented forward; that is, from the first to the second last node. Note that the states of the first and last nodes are always one and do not need to be changed.

Moreover, in the traditional AOA BAT, there is a variable "SUM" to count the number of coordinates—of which the values are one—to decide when to stop the BAT procedure. To improve the efficiency, the proposed AON BAT adopts another method by checking whether the current coordinate is the last coordinate; that is, the coordinate $n$. If the answer is yes, then the algorithm is stopped.

The number of all obtained vectors from the proposed AON BAT is $2^{n-2}$ because the 1st and $n$th coordinates are unchanged. In addition, the time complexity of the traditional AOA BAT is $O(2^{m+1})$ with the space complexity being $O(m)$ [52], whereas the time complexity and space complexity of the AON BAT are $O(2^{n-2+1}) = O(2^{n-1})$ and $O(n-1)$, respectively. Moreover, to calculate the AON BN reliability, an additional step is needed.

The pseudocode of the proposed AON BAT is presented in the following Algorithm 4:

**Algorithm 4:** AON BAT

```
Input:    A $n$-tuple vector $X$.
Output:   All solutions in vector form are found in the related AON problems.
          $X = 0$, except that both the first and last are 1s, and $k = 2$.
          while $k < n$
            if $X(a_k) = 0$
              $X(a_k) = 1$
              $k = 1$
              if $X$ is connected
                $R = R + \Pr(X)$
              endif
            else
              $X(a_k) = 0$
              $k = k + 1$
            end
          end
```

*4.3. AON PLSA*

The proposed AON PLSA is based on the traditional AOA PLSA proposed in [48,54] to verify the connectivity of a vector obtained from the proposed AON BAT. The basic idea of the proposed AON PLSA is very similar to that of the AOA PLSA. The pseudocode of the proposed AON PLSA is similar to that of the AOA PLSA and is listed as following Algorithm 5:

**Algorithm 5:** AOA PLSA.

```
Input:    A vector $X$ obtained from BAT.
Output:   Whether $X$ is connected or disconnected
          $V^* = Q_0 = \{1\}$ and $i = 1$.
          while $n \notin Q_i$ and $Q_i \neq \emptyset$
              $Q_i = \{ v \notin V^* \mid$ for all $e_{u,v} \in E$, $X(v) = 1$, and $u \in V^*\}$.
              $V^* = V^* \cup Q_i$
              $i = i + 1$
          end
```

The only difference between the proposed AON PLSA and AOA PLSA is the second step. Hence, the time complexity of the proposed AON PLSA is $O(n)$ [48].

*4.4. Example*

The general procedure of the proposed AON BAT-based algorithm is best demonstrated with examples. All types of networks are NP-hard and #P-hard. To reduce the complexity, a middle-sized SNs on AON BN, as shown in Figure 5, was implemented to better demonstrate how the proposed algorithm solves the SNs on AON BN reliability with the uncertainty components problem.

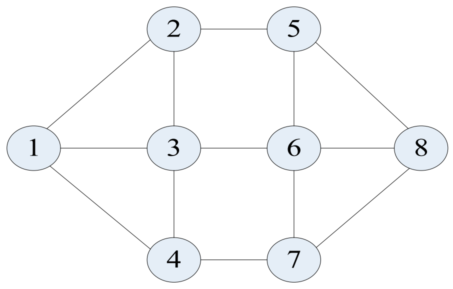

**Figure 5.** Example of a middle-sized SN on AON BN.

The information of nodes 3, 4, and 5 is unknown, and their ratings, that is, linguistic variables, are provided by six experts; the reliability of nodes 2, 6, and 7 is listed in Table 8. Note that nodes 1 and 8 are perfectly reliable.

**Table 8.** State distributions in Figure 5.

| ***a*** | **D (*a*)** | **Linguistic Variables** |
|---|---|---|
| $a_2$ | 0.80 | |
| $a_3$ | unknown | VL, L, L, VL, L, L |
| $a_4$ | unknown | VL, L, H, VH, L, H |
| $a_5$ | unknown | VH, H, L, VH, H, L |
| $a_6$ | 0.90 | |
| $a_7$ | 0.88 | |

After the preprocessing steps provided in Section 4.1, all uncertainty components are transferred to crisp components, as shown in Tables 6 and 7. The new state distribution after transformation is shown in Table 9.

**Table 9.** New state distributions of Figure 5.

| ***a*** | **D (*a*)** |
|---|---|
| $a_2$ | 0.80 |
| $a_3$ | 0.940504 |
| $a_4$ | 0.995000 |
| $a_5$ | 0.987185 |
| $a_6$ | 0.90 |
| $a_7$ | 0.88 |

Note that in Table 10, if Pr($X_i$) is not empty, $X_i$ is a connected vector. For example, Pr($X_1$) is empty and Pr($X_{64}$) = 0.585325 indicates that $X_1$ is a disconnected vector, and $X_{64}$ is a connected vector.

**Table 10.** Result obtained from the proposed AON BAT.

| $i$ | $X_i$ | Pr($x_2$) | Pr($x_3$) | Pr($x_4$) | Pr($x_5$) | Pr($x_6$) | Pr($x_7$) | Pr($X_i$) |
|---|---|---|---|---|---|---|---|---|
| 1 | (1, 0, 0, 0, 0, 0, 0, 1) | 0.2 | 0.059496 | 0.005 | 0.012815 | 0.1 | 0.12 | |
| 2 | (1, 1, 0, 0, 0, 0, 0, 1) | 0.8 | 0.059496 | 0.005 | 0.012815 | 0.1 | 0.12 | |
| 3 | (1, 0, 1, 0, 0, 0, 0, 1) | 0.2 | 0.940504 | 0.005 | 0.012815 | 0.1 | 0.12 | |
| 4 | (1, 1, 1, 0, 0, 0, 0, 1) | 0.8 | 0.940504 | 0.005 | 0.012815 | 0.1 | 0.12 | |
| 5 | (1, 0, 0, 1, 0, 0, 0, 1) | 0.2 | 0.059496 | 0.995 | 0.012815 | 0.1 | 0.12 | |
| 6 | (1, 1, 0, 1, 0, 0, 0, 1) | 0.8 | 0.059496 | 0.995 | 0.012815 | 0.1 | 0.12 | |
| 7 | (1, 0, 1, 1, 0, 0, 0, 1) | 0.2 | 0.940504 | 0.995 | 0.012815 | 0.1 | 0.12 | |
| 8 | (1, 1, 1, 1, 0, 0, 0, 1) | 0.8 | 0.940504 | 0.995 | 0.012815 | 0.1 | 0.12 | |
| 9 | (1, 0, 0, 0, 1, 0, 0, 1) | 0.2 | 0.059496 | 0.005 | 0.987185 | 0.1 | 0.12 | |
| 10 | (1, 1, 0, 0, 1, 0, 0, 1) | 0.8 | 0.059496 | 0.005 | 0.987185 | 0.1 | 0.12 | $2.82 \times 10^{-6}$ |

**Table 10.** *Cont.*

| $i$ | $X_i$ | Pr($x_2$) | Pr($x_3$) | Pr($x_4$) | Pr($x_5$) | Pr($x_6$) | Pr($x_7$) | Pr($X_i$) |
|---|---|---|---|---|---|---|---|---|
| 11 | (1, 0, 1, 0, 1, 0, 0, 1) | 0.2 | 0.940504 | 0.005 | 0.987185 | 0.1 | 0.12 | |
| 12 | (1, 1, 1, 0, 1, 0, 0, 1) | 0.8 | 0.940504 | 0.005 | 0.987185 | 0.1 | 0.12 | $4.46 \times 10^{-5}$ |
| 13 | (1, 0, 0, 1, 1, 0, 0, 1) | 0.2 | 0.059496 | 0.995 | 0.987185 | 0.1 | 0.12 | |
| 14 | (1, 1, 0, 1, 1, 0, 0, 1) | 0.8 | 0.059496 | 0.995 | 0.987185 | 0.1 | 0.12 | 0.000561 |
| 15 | (1, 0, 1, 1, 1, 0, 0, 1) | 0.2 | 0.940504 | 0.995 | 0.987185 | 0.1 | 0.12 | |
| 16 | (1, 1, 1, 1, 1, 0, 0, 1) | 0.8 | 0.940504 | 0.995 | 0.987185 | 0.1 | 0.12 | 0.008869 |
| 17 | (1, 0, 0, 0, 0, 1, 0, 1) | 0.2 | 0.059496 | 0.005 | 0.012815 | 0.9 | 0.12 | |
| 18 | (1, 1, 0, 0, 0, 1, 0, 1) | 0.8 | 0.059496 | 0.005 | 0.012815 | 0.9 | 0.12 | |
| 19 | (1, 0, 1, 0, 0, 1, 0, 1) | 0.2 | 0.940504 | 0.005 | 0.012815 | 0.9 | 0.12 | $1.3 \times 10^{-6}$ |
| 20 | (1, 1, 1, 0, 0, 1, 0, 1) | 0.8 | 0.940504 | 0.005 | 0.012815 | 0.9 | 0.12 | $5.21 \times 10^{-6}$ |
| 21 | (1, 0, 0, 1, 0, 1, 0, 1) | 0.2 | 0.059496 | 0.995 | 0.012815 | 0.9 | 0.12 | |
| 22 | (1, 1, 0, 1, 0, 1, 0, 1) | 0.8 | 0.059496 | 0.995 | 0.012815 | 0.9 | 0.12 | |
| 23 | (1, 0, 1, 1, 0, 1, 0, 1) | 0.2 | 0.940504 | 0.995 | 0.012815 | 0.9 | 0.12 | 0.000259 |
| 24 | (1, 1, 1, 1, 0, 1, 0, 1) | 0.8 | 0.940504 | 0.995 | 0.012815 | 0.9 | 0.12 | 0.001036 |
| 25 | (1, 0, 0, 0, 1, 1, 0, 1) | 0.2 | 0.059496 | 0.005 | 0.987185 | 0.9 | 0.12 | |
| 26 | (1, 1, 0, 0, 1, 1, 0, 1) | 0.8 | 0.059496 | 0.005 | 0.987185 | 0.9 | 0.12 | $2.54 \times 10^{-5}$ |
| 27 | (1, 0, 1, 0, 1, 1, 0, 1) | 0.2 | 0.940504 | 0.005 | 0.987185 | 0.9 | 0.12 | 0.0001 |
| 28 | (1, 1, 1, 0, 1, 1, 0, 1) | 0.8 | 0.940504 | 0.005 | 0.987185 | 0.9 | 0.12 | 0.000401 |
| 29 | (1, 0, 0, 1, 1, 1, 0, 1) | 0.2 | 0.059496 | 0.995 | 0.987185 | 0.9 | 0.12 | |
| 30 | (1, 1, 0, 1, 1, 1, 0, 1) | 0.8 | 0.059496 | 0.995 | 0.987185 | 0.9 | 0.12 | 0.005049 |
| 31 | (1, 0, 1, 1, 1, 1, 0, 1) | 0.2 | 0.940504 | 0.995 | 0.987185 | 0.9 | 0.12 | 0.019954 |
| 32 | (1, 1, 1, 1, 1, 1, 0, 1) | 0.8 | 0.940504 | 0.995 | 0.987185 | 0.9 | 0.12 | 0.079817 |
| 33 | (1, 0, 0, 0, 0, 0, 1, 1) | 0.2 | 0.059496 | 0.005 | 0.012815 | 0.1 | 0.88 | |
| 34 | (1, 1, 0, 0, 0, 0, 1, 1) | 0.8 | 0.059496 | 0.005 | 0.012815 | 0.1 | 0.88 | |
| 35 | (1, 0, 1, 0, 0, 0, 1, 1) | 0.2 | 0.940504 | 0.005 | 0.012815 | 0.1 | 0.88 | |
| 36 | (1, 1, 1, 0, 0, 0, 1, 1) | 0.8 | 0.940504 | 0.005 | 0.012815 | 0.1 | 0.88 | |
| 37 | (1, 0, 0, 1, 0, 0, 1, 1) | 0.2 | 0.059496 | 0.995 | 0.012815 | 0.1 | 0.88 | $1.34 \times 10^{-5}$ |
| 38 | (1, 1, 0, 1, 0, 0, 1, 1) | 0.8 | 0.059496 | 0.995 | 0.012815 | 0.1 | 0.88 | $5.34 \times 10^{-5}$ |
| 39 | (1, 0, 1, 1, 0, 0, 1, 1) | 0.2 | 0.940504 | 0.995 | 0.012815 | 0.1 | 0.88 | 0.000211 |
| 40 | (1, 1, 1, 1, 0, 0, 1, 1) | 0.8 | 0.940504 | 0.995 | 0.012815 | 0.1 | 0.88 | 0.000844 |
| 41 | (1, 0, 0, 0, 1, 0, 1, 1) | 0.2 | 0.059496 | 0.005 | 0.987185 | 0.1 | 0.88 | |
| 42 | (1, 1, 0, 0, 1, 0, 1, 1) | 0.8 | 0.059496 | 0.005 | 0.987185 | 0.1 | 0.88 | $2.07 \times 10^{-5}$ |
| 43 | (1, 0, 1, 0, 1, 0, 1, 1) | 0.2 | 0.940504 | 0.005 | 0.987185 | 0.1 | 0.88 | |
| 44 | (1, 1, 1, 0, 1, 0, 1, 1) | 0.8 | 0.940504 | 0.005 | 0.987185 | 0.1 | 0.88 | 0.000327 |
| 45 | (1, 0, 0, 1, 1, 0, 1, 1) | 0.2 | 0.059496 | 0.995 | 0.987185 | 0.1 | 0.88 | 0.001029 |
| 46 | (1, 1, 0, 1, 1, 0, 1, 1) | 0.8 | 0.059496 | 0.995 | 0.987185 | 0.1 | 0.88 | 0.004114 |
| 47 | (1, 0, 1, 1, 1, 0, 1, 1) | 0.2 | 0.940504 | 0.995 | 0.987185 | 0.1 | 0.88 | 0.016259 |
| 48 | (1, 1, 1, 1, 1, 0, 1, 1) | 0.8 | 0.940504 | 0.995 | 0.987185 | 0.1 | 0.88 | 0.065036 |
| 49 | (1, 0, 0, 0, 0, 1, 1, 1) | 0.2 | 0.059496 | 0.005 | 0.012815 | 0.9 | 0.88 | |
| 50 | (1, 1, 0, 0, 0, 1, 1, 1) | 0.8 | 0.059496 | 0.005 | 0.012815 | 0.9 | 0.88 | |
| 51 | (1, 0, 1, 0, 0, 1, 1, 1) | 0.2 | 0.940504 | 0.005 | 0.012815 | 0.9 | 0.88 | $9.55 \times 10^{-6}$ |
| 52 | (1, 1, 1, 0, 0, 1, 1, 1) | 0.8 | 0.940504 | 0.005 | 0.012815 | 0.9 | 0.88 | $3.82 \times 10^{-5}$ |
| 53 | (1, 0, 0, 1, 0, 1, 1, 1) | 0.2 | 0.059496 | 0.995 | 0.012815 | 0.9 | 0.88 | 0.00012 |
| 54 | (1, 1, 0, 1, 0, 1, 1, 1) | 0.8 | 0.059496 | 0.995 | 0.012815 | 0.9 | 0.88 | 0.000481 |
| 55 | (1, 0, 1, 1, 0, 1, 1, 1) | 0.2 | 0.940504 | 0.995 | 0.012815 | 0.9 | 0.88 | 0.0019 |
| 56 | (1, 1, 1, 1, 0, 1, 1, 1) | 0.8 | 0.940504 | 0.995 | 0.012815 | 0.9 | 0.88 | 0.007598 |
| 57 | (1, 0, 0, 0, 1, 1, 1, 1) | 0.2 | 0.059496 | 0.005 | 0.987185 | 0.9 | 0.88 | |
| 58 | (1, 1, 0, 0, 1, 1, 1, 1) | 0.8 | 0.059496 | 0.005 | 0.987185 | 0.9 | 0.88 | 0.000186 |
| 59 | (1, 0, 1, 0, 1, 1, 1, 1) | 0.2 | 0.940504 | 0.005 | 0.987185 | 0.9 | 0.88 | 0.000735 |
| 60 | (1, 1, 1, 0, 1, 1, 1, 1) | 0.8 | 0.940504 | 0.005 | 0.987185 | 0.9 | 0.88 | 0.002941 |

**Table 10.** *Cont.*

| $i$ | $X_i$ | Pr($x_2$) | Pr($x_3$) | Pr($x_4$) | Pr($x_5$) | Pr($x_6$) | Pr($x_7$) | Pr($X_i$) |
|---|---|---|---|---|---|---|---|---|
| 61 | (1, 0, 0, 1, 1, 1, 1, 1) | 0.2 | 0.059496 | 0.995 | 0.987185 | 0.9 | 0.88 | 0.009257 |
| 62 | (1, 1, 0, 1, 1, 1, 1, 1) | 0.8 | 0.059496 | 0.995 | 0.987185 | 0.9 | 0.88 | 0.037028 |
| 63 | (1, 0, 1, 1, 1, 1, 1, 1) | 0.2 | 0.940504 | 0.995 | 0.987185 | 0.9 | 0.88 | 0.146331 |
| 64 | (1, 1, 1, 1, 1, 1, 1, 1) | 0.8 | 0.940504 | 0.995 | 0.987185 | 0.9 | 0.88 | 0.585325 |
| | SUM | | | | | | | 0.995984 |

From the last row in Table 10, we have $R$ = 0.995984, which is the summation of the reliability of all connected vectors in Figure 5.

## 5. Conclusions

Various social networks (SNs) are necessary for modern daily life. The reliability of these SNs is defined as the success probability that the BN is working, that is, the source and sink nodes are connected by a directed path in the BN of SNs. Hence, reliability can be implemented in evaluating, managing, and designing the performance of all types of BNs. However, most reliability problems are focused on the AOA network structure; moreover, they assume that the reliability of each component is known in advance. This paper presents a novel AON BN of the SN reliability problem with uncertainty components and presents an AON BAT to solve the problem.

To solve the proposed novel problem, we first invited experts to provide their linguistic variables to rate these uncertainty components. A fuzzy set was then applied. Each linguistic variable was transferred to a fuzzy number, and these fuzzy numbers were aggregated into an AFN corresponding to the same uncertainty component. Each AFN was transferred further to the FPS and FFR, which is considered to be an unreliability in the proposed problem.

After transferring each uncertainty component variable to a crisp component, the proposed AON BAT was implemented to calculate the reliability of the AON BNs of SNs together with the proposed AON PLSA to verify the connectivity of each obtained vector. Moreover, the time complexity of the proposed AON BAT was $O(2^{n-1})$, with the space complexity being $O(n-1)$, both of which are less than that of AOA BAT, where $n$ is the number of nodes. In addition, from the example demonstration, it can be observed that the proposed AON BAT can calculate the AON BN of SN reliability with uncertainty components for the 8-node 13-arc 3-uncertainty-component BN. Thus, the proposed algorithm has the potential to solve larger-sized problems, the propagation of information in larger-sized problems, or the assumption of each node and arc as perfectly reliable. Furthermore, a real-world application of the proposed ANO BAT is planned to be studied.

In future works, the proposed AON BN of SNs reliability problem with uncertainty components will be further extended to AON multi-state networks of SNs and AON multi-commodity multi-state networks. The proposed algorithm will be improved correspondingly for larger-sized problems.

**Author Contributions:** Conceptualization, W.-C.Y., W.Z. and C.-L.H.; methodology, W.-C.Y., W.Z. and C.-L.H.; software, W.-C.Y., W.Z. and C.-L.H.; validation, W.-C.Y. and W.Z.; formal analysis, W.-C.Y., W.Z. and C.-L.H.; investigation, W.-C.Y. and W.Z.; resources, W.-C.Y. and W.Z.; data curation, W.-C.Y. and W.Z.; writing—original draft preparation, W.-C.Y., W.Z. and C.-L.H.; writing—review and editing, W.-C.Y., W.Z. and C.-L.H.; visualization, W.-C.Y. and W.Z.; supervision, W.-C.Y. and W.Z.; project administration, W.-C.Y.; funding acquisition, W.Z. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported in part by Guangdong Province Key Field R&D Program Project: Grant No. 2021B0101410002, Natural Science Foundation of China: Grant No. 62106048 and Ministry of Science and Technology, R.O.C (MOST 107-2221-E-007-072-MY3, MOST 110-2221-E-007-107-MY3, MOST 109-2221-E-424-002 and MOST 110-2511-H-130-002).

**Acknowledgments:** We wish to thank the anonymous editor and the referees for their constructive comments and recommendations, which significantly improved this paper. This article was once submitted to arXiv as a temporary submission that was only for reference and did not provide the copyright.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yeh, W.C.; Zhu, W.; Huang, C.L.; Hsu, T.Y.; Liu, Z.; Tan, S.Y. A New BAT and PageRank Algorithm for Propagation Probability in Social Networks. *Appl. Sci.* **2022**, *12*, 6858. [CrossRef]
2. Korányi, R.; Mancera, J.A.; Kaufmann, M. GDPR-Compliant Social Network Link Prediction in a Graph DBMS: The Case of Know-How Development at Beekeeper. *Knowledge* **2022**, *2*, 17. [CrossRef]
3. Yeh, W.C.; Tan, S.Y.; Forghani-elahabad, M.; Khadiri, M.E.; Jiang, Y.; Lin, C.S. New binary-addition tree algorithm for the all-multiterminal binary-state network reliability problem. *Reliab. Eng. Syst. Saf.* **2022**, *224*, 108557. [CrossRef]
4. Aven, T. Availability evaluation of oil/gas production and transportation systems. *Reliab. Eng.* **1987**, *18*, 35–44. [CrossRef]
5. Levitin, G.; Xing, L.; Dai, Y. Optimal Spot-Checking for Collusion Tolerance in Computer Grids. *IEEE Trans. Dependable Secur. Comput.* **2017**, *16*, 301–312. [CrossRef]
6. Yeh, W.C.; Wei, S.C. Economic-based resource allocation for reliable Grid-computing service based on Grid Bank. *Future Gener. Comput. Syst.* **2012**, *28*, 989–1002. [CrossRef]
7. Yeh, W.C.; Tan, S.Y.; Zhu, W.; Huang, C.L.; Yang, G. Novel Binary Addition Tree Algorithm (BAT) for Calculating the Direct Lower-Bound of the Highly Reliable Binary-State Network Reliability. *Reliab. Eng. Syst. Saf.* **2022**, *223*, 108509. [CrossRef]
8. Yeh, W.C.; Lin, J.S. New parallel swarm algorithm for smart sensor systems redundancy allocation problems in the Internet of Things. *J. Supercomput.* **2018**, *74*, 4358–4384. [CrossRef]
9. Wang, J.; Yeh, W.C.; Xiong, N.N.; Wang, J.; He, X.; Huang, C.L. Building an improved Internet of things smart sensor network based on a three-phase methodology. *IEEE Access* **2019**, *7*, 141728–141737. [CrossRef]
10. Kakadia, D.; Ramirez-Marquez, J.E. Quantitative approaches for optimization of user experience based on network resilience for wireless service provider networks. *Reliab. Eng. Syst. Saf.* **2020**, *193*, 106606. [CrossRef]
11. Su, Y.Z.; Yeh, W.C. Binary-Addition Tree Algorithm-Based Resilience Assessment for Binary-State Network Problems. *Electronics* **2020**, *9*, 1207. [CrossRef]
12. Yeh, W.C. New method in searching for all minimal paths for the directed acyclic network reliability problem. *IEEE Trans. Reliab.* **2016**, *65*, 1263–1270. [CrossRef]
13. Ramirez-Marquez, J.E. Assessment of the transition-rates importance of Markovian systems at steady state using the unscented transformation. *Reliab. Eng. Syst. Saf.* **2015**, *142*, 212–220.
14. Singh, A.P.; Luhach, A.K.; Gao, X.Z.; Kumar, S.; Roy, D.S. Evolution of wireless sensor network design from technology centric to user centric: An architectural perspective. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720949138. [CrossRef]
15. Coit, D.W.; Zio, E. The evolution of system reliability optimization. *Reliab. Eng. Syst. Saf.* **2018**, *192*, 106259. [CrossRef]
16. Aven, T. Some considerations on reliability theory and its applications. *Reliab. Eng. Syst. Saf.* **1988**, *21*, 215–223. [CrossRef]
17. Colbourn, C.J. *The Combinatorics of Network Reliability*; Oxford University Press, Inc.: Oxford, UK, 1987.
18. Shier, D.R. *Network Reliability and Algebraic Structures*; Clarendon Press: Oxford, UK, 1991.
19. Levitin, G. *The Universal Generating Function in Reliability Analysis and Optimization*; Springer: Berlin/Heidelberg, Germany, 2005.
20. Yeh, W.C. Search for MC in modified networks. *Comput. Oper. Res.* **2001**, *28*, 177–184. [CrossRef]
21. Yeh, W.C. A simple algorithm for evaluating the k-out-of-n network reliability. *Reliab. Eng. Syst. Saf.* **2004**, *83*, 93–101. [CrossRef]
22. Yeh, W.C. A new algorithm for generating minimal cut sets in k-out-of-n networks. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 36–43. [CrossRef]
23. Ahmad, S.H. Simple enumeration of minimal cutsets of acyclic directed graph. *IEEE Trans. Reliab.* **1988**, *37*, 484–487. [CrossRef]
24. Yeh, W.C. A MCS-RSM approach for network reliability to minimise the total cost. *Int. J. Adv. Manuf. Technol.* **2003**, *22*, 681–688. [CrossRef]
25. Yeh, W.C. A squeezed artificial neural network for the symbolic network reliability functions of binary-state networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 2822–2825. [CrossRef] [PubMed]
26. Yeh, W.C. The k-out-of-n acyclic multistate-node networks reliability evaluation using the universal generating function method. *Reliab. Eng. Syst. Saf.* **2006**, *91*, 800–808. [CrossRef]
27. Yeh, W.C. A new branch-and-bound approach for the n/2/flowshop/αF+ βCmax flowshop scheduling problem. *Comput. Oper. Res.* **1999**, *26*, 1293–1310. [CrossRef]
28. Yeh, W.C. A novel boundary swarm optimization method for reliability redundancy allocation problems. *Reliab. Eng. Syst. Saf.* **2019**, *192*, 106060. [CrossRef]
29. Hao, Z.; Yeh, W.C.; Tan, S.Y. One-Batch Preempt Multi-State Multi-Rework Network Reliability Problem. *Reliab. Eng. Syst. Saf.* **2021**, *215*, 107883. [CrossRef]
30. Kvassay, M.; Levashenko, V.; Zaitseva, E. Analysis of minimal cut and path sets based on direct partial Boolean derivatives. *Proc. Inst. Mech. Eng. Part O J. Risk Reliab.* **2016**, *230*, 147–161. [CrossRef]

31. Zuo, M.J.; Tian, Z.; Huang, H.Z. An efficient method for reliability evaluation of multistate networks given all minimal path vectors. *IIE Trans.* **2007**, *39*, 811–817. [CrossRef]
32. Niu, Y.F.; Shao, F.M. A practical bounding algorithm for computing two-terminal reliability based on decomposition technique. *Comput. Math. Appl.* **2011**, *61*, 2241–2246. [CrossRef]
33. Chen, S.G.; Lin, Y.K. An Improved Merge Search Approach to Evaluate Reliability in Multistate Network Systems. *IEEE Trans. Reliab.* **2022**, *71*, 382–389. [CrossRef]
34. Yeh, W.C. A Quick BAT for Evaluating the Reliability of Binary-State Networks. *Reliab. Eng. Syst. Saf.* **2021**, *216*, 107917. [CrossRef]
35. Lee, C.Y. Representation of switching circuits by binary-decision programs. *Bell Syst. Tech. J.* **1959**, *38*, 985–999. [CrossRef]
36. Yeh, W.C. A greedy branch-and-bound inclusion-exclusion algorithm for calculating the exact multi-state network reliability. *IEEE Trans. Reliab.* **2008**, *57*, 88–93.
37. Hao, Z.; Yeh, W.C.; Wang, J.; Wang, G.G.; Sun, B. A quick inclusion-exclusion technique. *Inf. Sci.* **2019**, *486*, 20–30. [CrossRef]
38. Niu, Y.; Gao, Z.; Sun, H. An improved algorithm for solving all *d*-MPs in multi-state networks. *J. Syst. Sci. Syst. Eng.* **2017**, *26*, 711–731. [CrossRef]
39. Bryant, R.E. Graph-based algorithms for boolean function manipulation. *IEEE Trans. Comput.* **1986**, *100*, 677–691. [CrossRef]
40. Yeh, W.C. An improved sum-of-disjoint-products technique for the symbolic network reliability analysis with known minimal paths. *Reliab. Eng. Syst. Saf.* **2007**, *92*, 260–268. [CrossRef]
41. Yeh, W.C. An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability. *IEEE Trans. Reliab.* **2015**, *64*, 1185–1193. [CrossRef]
42. Bai, G.; Zuo, M.J.; Tian, Z. Search for all *d*-MPs for all *d* levels in multistate two-terminal networks. *Reliab. Eng. Syst. Saf.* **2015**, *142*, 300–309. [CrossRef]
43. Wang, Y.; Xing, L.; Wang, H.; Coit, D.W. System reliability modeling considering correlated probabilistic competing failures. *IEEE Trans. Reliab.* **2017**, *67*, 416–431. [CrossRef]
44. Wang, M.; Yeh, W.C.; Chu, T.C.; Zhang, X.; Huang, C.L.; Yang, J. Solving multi-objective fuzzy optimization in wireless smart sensor networks under uncertainty using a hybrid of IFR and SSO algorithm. *Energies* **2018**, *11*, 2385. [CrossRef]
45. Gurvich, V.; Khachiyan, L. On generating the irredundant conjunctive and disjunctive normal forms of monotone Boolean functions. *Discret. Appl. Math.* **1999**, *96–97*, 363–373. [CrossRef]
46. Zadeh, L.A. Fuzzy sets. *Inf. Control* **1965**, *8*, 338–353. [CrossRef]
47. Lin, C.T.; Wang, M.J. Hybrid fault tree analysis using fuzzy sets. *Reliab. Eng. Syst. Saf.* **1997**, *58*, 205–213. [CrossRef]
48. Yeh, W.C. Novel binary-addition tree algorithm (BAT) for binary-state network reliability problem. *Reliab. Eng. Syst. Saf.* **2021**, *208*, 107448. [CrossRef]
49. Yeh, W.C.; Kuo, C.C. Predicting and Modeling Wildfire Propagation Areas with BAT and Maximum-State PageRank. *Appl. Sci.* **2020**, *10*, 8349. [CrossRef]
50. Yeh, W.C.; Lin, E.; Huang, C.L. Predicting Spread Probability of Learning-Effect Computer Virus. *Complexity* **2021**, *2021*, 6672630. [CrossRef]
51. Yeh, W.C.; Hao, Z.; Forghani-elahabad, M.; Wang, G.G.; Lin, Y.L. Novel Binary-Addition Tree Algorithm for Reliability Evaluation of Acyclic Multistate Information Networks. *Reliab. Eng. Syst. Saf.* **2021**, *210*, 107427. [CrossRef]
52. Yeh, W.C. Novel Algorithm for Computing All-Pairs Homogeneity-Arc Binary-State Undirected Network Reliability. *Reliab. Eng. Syst. Saf.* **2021**, *216*, 107950. [CrossRef]
53. Yeh, W.C. A simple universal generating function method to search for all minimal paths in networks. *IEEE Trans. Syst. Man Cybern.-Part A Syst. Hum.* **2009**, *39*, 1247–1254.
54. Yeh, W.C. A revised layered-network algorithm to search for all d-minpaths of a limited-flow acyclic network. *IEEE Trans. Reliab.* **1998**, *47*, 436–442.
55. Chen, S.J.; Hwang, C.L. *Fuzzy Multiple Attribute Decision Making Methods and Applications*; Springer: New York, NY, USA, 1992.
56. Swain, A.D.; Guttmann, H.E. *Handbook of Human Reliability with Emphasis on Nuclear Power Plant Applications*; NUREG/CR-1278-F; Sandia National Labs: Albuquerque, NM, USA, 1983.