# Robot Trust Evaluation and Fault Detection using Weighted Trust Rank and AdaBoost Framework

*Divya* Sathi Raj[1], *Jaganathan* Baladasan[2*], and *Kalaipriyan* Thirugnanasambandam[3]

[1]Department of Mathematics, School of advanced Sciences, Vellore Institute of Technology,Chennai Campus, Vandalur-Kelambakkam Road, Chennai, Tamil Nadu, India.

[2*] Department of Mathematics, School of advanced Sciences, Vellore Institute of Technology,Chennai Campus, Vandalur-Kelambakkam Road, Chennai, Tamil Nadu, India.

[3]School of Computer Science and Engineering, Vellore Institute of Technology, Chennai Campus, Chennai, Tamil Nadu, India.

**Abstract.** Web spam degrades the reliability of search engine results by manipulating ranking algorithms to raise webpage positions artificially. TrustRank is a popular link-based anti-spam method, its ability to demote highly ranked spam webpages is limited by its exclusive dependence on out-link data. This paper proposes a Weighted Trust Rank ($WTR$) approach that incorporates link weight into the conventional TrustRank framework in order to overcome this limitation. The proposed approach makes use of several link-splitting and accumulation strategies, including equal splitting, logarithmic splitting, and Maxshare accumulation, to enhance spam webpage detection. To evaluate the probability of spam content, a Weighted Spam Mass ($WSM$) measure is also introduced. A robust ensemble model for identifying influential and spam webpages in web graphs is created by applying the AdaBoost method to integrate multiple weak classifiers through iterative reweighting, thereby further improving classification performance. The suggested WTR approach consistently beats traditional TrustRank, Anti-TrustRank, and Weighted Anti-TrustRank in terms of precision, recall, and accuracy, according to experimental results on two sample web graphs and the WEBSPAM-UK2006 dataset. Additionally, the WTR framework may be expanded to multi-robot systems for fault detection and trust assessment, showcasing its flexibility in improving dependability and collaborative decision-making.

## 1 Introduction

The World Wide Web serves billions of users every day and is a vast information resource. Because visitors usually only pay attention to the first few websites of search results, web search engines are essential for finding relevant material. Content providers may legally increase the quality of their webpages to appear in these top results, but this process is frequently resource-intensive and time-consuming. As a result, many turn to deceptive tactics that artificially raise rankings. Web spam is an unethical activity that takes advantage of search engine algorithms in order to deceive ranking systems and make profits.

*Corresponding author: jaganathan.b@vit.ac.in

Web spam poses a significant challenge for search engines because it undermines the reliability of search results. Over the years, numerous anti-spam methods have been proposed, many of which exploit the hyperlink structure of the web. Among these, TrustRank has emerged as a widely studied approach. TrustRank operates on the assumption that trustworthy webpages rarely link to spam webpages. By starting from a carefully selected seed set of trusted webpages, it propagates trust along outgoing links to identify other trustworthy webpages. In contrast, methods such as Anti-TrustRank propagate distrust backwards from a set seed set of untrustworthy webpages (called spam seeds)and transmit distrust in the reverse direction to identify other spam webpages[3],[11]. Together, these trust and distrust propagation models aim to distinguish between legitimate and spam webpages.

Trust Rank shows that the concept of transmitting trust from a seed set of most trusted webpages helps a lot in the relegation of web spam. This method makes certain assumption that how trust is transmit from parent webpage[1] to a child webpage. The principle of trust transmission method is to spread trust to trustworthy webpages through links as transferring trust between webpages. Despite their utility, traditional TrustRank methods exhibit a major limitation, trust propagation based solely on out-links is not optimal for demoting top-ranked spam webpages. Since spam webpages often manipulate link structures, relying only on out-degree splitting may fail to effectively reduce their influence. This shortcoming motivates the need for enhanced approaches that incorporate richer structural information.

We provide a Weighted Trust Rank (WTR) approach that incorporates link weight characteristics into the trust propagation procedure in order to overcome this constraint. WTR is intended to enhance the demotion of top-ranked spam webpages by integrating techniques including equal splitting, logarithmic splitting, and Maxshare accumulation. Weighted Spam Mass (WSM), which measures the probability that a webpage is spam in relation to its PageRank score, is another addition to the framework.

Freund and Schapire (1997) created the effective ensemble learning method known as Adaptive Boosting (AdaBoost) to enhance the performance of weak classifiers. Using an iterative reweighting process, it builds a strong classifier by successively combining several weak learners, usually decision stumps. The approach increases the weights of wrongly classified samples with each iteration, forcing later weak learners to concentrate more on these challenging cases. By using a weighted majority voting strategy to aggregate the predictions of the weak learners, the final classification result improves both accuracy and generalization. By integrating AdaBoost with link-based ranking measures such as Weighted Trust Rank (WTR), WTR with equal splitting and Maxshare, WTR with logarithmic splitting and Maxshare, the proposed framework effectively strengthens the identification and ranking of influential and spam webpages. This combination improves detection precision, robustness, and overall reliability in web graph analysis.

The effectiveness of the proposed methods is validated through experiments on two illustrative web graphs and the large-scale WEBSPAM-UK2006 dataset. Results show that WTR achieves higher accuracy than existing methods[8, 11], including TrustRank, Anti-TrustRank, and Weighted Anti-TrustRank. These findings indicate that incorporating weight-based features into link analysis provides a more effective and reliable approach to web spam detection, thereby contributing to the ongoing development of robust search engine optimization techniques.

The growing usage of robots in industrial and safety-critical applications has drawn a lot of interest to the research on fault detection in robotic systems. The primary purpose of fault detection and diagnostic (FDD) techniques is to find anomalous states in robot sensors,

actuators, controllers, or communication networks before they cause major malfunctions. Numerous studies have traditionally concentrated on model-based fault detection techniques, which create residual signals and identify departures from typical behavior using mathematical models of robot dynamics. In order to identify flaws in humanoid and mobile robots, methods like observers, Kalman filters, and statistical monitoring charts have been used extensively. Model-based approaches, however, frequently call for precise system modeling, which is challenging for intricate nonlinear robotic systems functioning in unpredictable contexts.

Data-driven and machine learning-based fault detection solutions have grown in popularity as artificial intelligence has advanced. Convolutional neural networks (CNNs), recurrent neural networks (RNNs), and autoencoders are examples of deep learning models that have been used to automatically identify fault patterns in vast amounts of operational and sensor data. These methods offer greater flexibility and enhanced accuracy in fault classification, particularly in industrial robots with access to extensive datasets. In addition, new research has suggested hybrid defect diagnostic frameworks that improve interpretability and resilience by combining machine learning classifiers with physics-based models. Detecting flaws in cooperative environments has also been expanded to multi-robot and swarm robotic systems, where consensus-based techniques and distributed methodologies are used. Overall, research shows that fault detection in robotics is still developing, moving from conventional model-based techniques to intelligent hybrid and AI-driven solutions that allow for more autonomous, fault-tolerant, and dependable robotic operations.

Reliability is essential in multi-robot systems, especially in unpredictable and dynamic situations. Any physical or virtual component may experience faults, which could spread throughout the network and impair system performance as a whole. In vast, interconnected robotic networks, traditional fault detection methods frequently struggle to identify faulty robot. Each robot can be represented as a node and interactions or collaborations as weighted edges by using the WTR framework, which enables the system to assess each robot's reliability. This method offers precise fault detection, improves reliability assessment, and permits prompt isolation of malfunctioning robots when paired with the AdaBoost algorithm, which iteratively enhances fault classification by concentrating on challenging cases. This improves the safety and robustness of multi-robot operations.

## 2 Preliminaries

### 2.1 Web Graph:

The web graph $G = (\mathcal{N}, \mathcal{A})$ consists of a set $\mathcal{N}$ of web pages (nodes) and a set $\mathcal{A}$ of directed links(arc) that connect webpages if there exist a link from one webpage to another webpage. A webpage can have more than one links to some other webpages. Self hyperlinks are removed. The incoming links are called in-links whereas outgoing links are called out-links. The number of in-links is its in-degree $\tau(s)$ where as the number of out-links is its out-degree $\omega(s)$.

Two matrix representation of web graph:

- **Transition Matrix (T)**

$$T(r,s) = \begin{cases} \frac{1}{\omega(s)} & ,(s,r) \in \mathcal{A} \\ 0 & ,(s,r) \notin \mathcal{A} \end{cases}$$

- **Inverse Transition Matrix (I):**

$$I(r,s) = \begin{cases} \frac{1}{\tau(s)} & ,(r,s) \in \mathcal{A} \\ 0 & ,(r,s) \notin \mathcal{A} \end{cases}$$

### 2.2 Weighted Web Graph

Weighted web graph represented as $G = (\mathcal{N}, \mathcal{A}, W)$ where $\mathcal{N}$ is the set of webpages(nodes), $\mathcal{A}$ is the set of links(arc)and $W$ denotes the number of links for each arc of directed graph. The weight feature is used to get the outgoing weight of a weighted web graph.

### 2.3 Seed set

The set of webpages chosen for transmission purpose is known as seed set. For trust transmission, seeds are sorted out depending upon their outgoing link. Conversely for distrust transmission, seeds are sorted out depending upon their incoming links. Trust and distrust model web spam detection methods filter web spam by selecting trustworthy webpages to propagate trust or spam webpages to propagate distrust[3].

### 2.4 Link Farm

Any group of websites that link to each other in order to improve their SEO(Search Engine Optimisation) ranking results is known as a link farm.

### 2.5 Weight depends on In-Out links of a webpages

The inlink-weight and outlink-weight of a webpages can be calculated by using the equation 1,2:

$$W^{in}_{(r,s)} = \frac{I_s}{\sum\limits_{u \in K(r)} I_u} \tag{1}$$

$$W^{out}_{(r,s)} = \frac{O_s}{\sum\limits_{u \in K(r)} O_u} \tag{2}$$

where,$I_s$ and $I_u$ are inlinks of a webpage $s$ and $u$, $O_s$ and $O_u$ are outlinks of webpage $s$ and$u$. $K(r)$ is the collection of all webpages that pointed to $r$.

The weight matrix $W(G)$ is calculated using equation 3:

$$W(G) = W^{in}_{(r,s)} + W^{out}_{(r,s)} \tag{3}$$

If there is no link from webpages $r$ to $s$, then $W_{(r,s)} = 0$. For a web graph without self hyperlink $W_{(r,r)} = 0$ [6].

## 3 Trust transmission

Trust Rank works like PageRank to spread trust among webpages. This method involves in each iteration webpages propagate their trust to the children with probability '$\alpha$' or retreating to the seed set with probability '$1 - \alpha$'. Transmission of trust involves two steps-splitting step and accumulation step. Splitting step is how each parent shares its trust score with its children. The accumulation step calculates the total scores from all parents for each child[3].

For splitting step, we have three types:

- **Equal Splitting(ES):** A node $s$ with $\omega(s)$out-links and trust score $TR(s)$ will give $\alpha \frac{TR(s)}{\omega(s)}$ to each child. $\alpha$ is constant with $0 < \alpha < 1$
- **Constant Splitting(CS):** A node $s$ with trust score $Tr(s)$ will provide $\alpha\, TR(s)$ to each child.
- **Logarithm Splitting(LS):** A node $s$ with $\omega(s)$ out-links and trust score $TR(s)$will give $\alpha \frac{Tr(s)}{log(1+\omega(s))}$ to each child.

For accumulation step, we have three types:

- **Simple Summation(SS):** Sum the trust value from each parent.
- **Maximum share(MS):** Use the maximum trust value sent by the parents.

## 4 Trust transmission for relegating spam webpages

Spam webpages betrays search engine's ranking method. Web spam is a prime contest for search engine. To battle with web spam ,Gyongyi et al(2004) proposed Trust Rank. It is formed on the perception that trustworthy webpages point to trustworthy webpages and rarely point to untrustworthy webpages, and people trust these webpages. Trust is transmitted through the links on the web. A collection of most trustworthy webpages are taken as seed set and each of them is alloted with a non-zero initial trust score, whereas the other webpages are alloted with an initial value Zero. After convergence, the spam webpages will get a least trust score compared to other webpages [13].
Trust Rank is calculated using equation 4:

$$TR(r) = \alpha \sum_{s \to r} \frac{TR(s)}{\omega(s)} + (1 - \alpha) d(r) \tag{4}$$

where,

$$\begin{aligned} TR(r) &= \text{Trust Rank of webpage r} \\ \alpha &= \text{Damping factor} \\ \omega(s) &= \text{Outgoing link of s} \\ d(r) &= \text{Normalised trust score of the good seed set}(\rho^+) \end{aligned}$$

$$d(r) = \begin{cases} \frac{1}{|\rho^+|} & , \ r \in \rho^+ \\ 0 & , \ r \notin \rho^+ \end{cases}$$

## 5 Existing Methods

To combat web spam, various methods have been proposed. Antispamming strategies include Trust Rank, Anti-Trust Rank, Distrust Rank, BadRank, Weighted Anti-TrustRank, Wu et al Distrust Rank, and Nie et al Distrust Rank. Spam seeds are selected based on PageRank and spread distrust to detect Web spam. Web spam downgrading methods are trust-based methods that fight Web spam. Web spam detection methods employ distrust to combat spam. The explanation of the methods are given below. Here $\alpha$ is the damping factor, $\tau(s)$ is the incoming link of webpage 's', r $\rightarrow$ s denote there is a link from webpage 'r' to webpage 's'.

### 5.1 Anti-Trust Rank

According to Anti-Trust Rank, trustworthy webpages (non-spam) rarely lead to untrustworthy webpages(spam), hence spam webpages are likely to be spam themselves. Starting with non-spam webpages, Trust Rank spread trust via outgoing links. Anti-Trust Rank, which starts with spam webpages, propagates anti-trust via in-coming links. We may designate a webpage as spam if its Anti-Trust Rank exceeds a specified threshold value. This approach suggests returning the top n webpages based on Anti-Trust Rank, which are most likely to be spam. Krishnan et al used high PageRank pages as spam seeds and deployed the method to detect more spam webpages[12].

The formula for calculating Anti-Trust Rank is given in 5:

$$ATR(r) = \alpha \sum_{r \rightarrow s} \frac{ATR(s)}{\tau(s)} + (1-\alpha)d^{'}(r) \tag{5}$$

where,

$ATR(r) =$ Anti-Trust Rank of webpage r
$\alpha =$ Damping factor
$\tau(s) =$ incoming link of webpage s
$d^{'}(r) =$ Normalised anti-trust score of the bad seed set($\rho^{-}$)

$$d^{'}(r) = \begin{cases} \frac{1}{|\rho^{-}|} & , r \in \rho^{-} \\ 0 & , r \notin \rho^{-} \end{cases}$$

### 5.2 BadRank

Raph Levien developed BadRank to assess a webpage's bad attributes. The formula for this method is given in 6:

$$BR(r) = \alpha \Big[ \sum_{r \rightarrow s} \frac{BR(s)}{\tau(s)} \Big] + (1-\alpha)E(r) \tag{6}$$

where, $BR(r)$= BadRank of webpage 'r'

$$E(r) = \begin{cases} 1 & , \text{if 'r' is in spam list} \\ 0 & , \text{otherwise} \end{cases}$$

### 5.3 Weighted Anti-TrustRank

Extension of Anti-Trust Rank by adding weight is Weighted Anti-TrustRank. The formula is given in 7:

$$WATR(r) = \alpha\Big[\sum_{r\to s}\Big(\frac{WATR(s)}{\tau(s)}O_{rs}\Big)\Big] + (1-\alpha)B(r) \tag{7}$$

where, $O_{rs}$ is the outgoing weight of webpage 'r' to webpage 's', $B(r)$ is the spam vector.

$$B(r) = \begin{cases} 1 & \text{, if 'r' is in spam list} \\ 0 & \text{, otherwise} \end{cases}$$

### 5.4 Distrust Rank

Wu et al. employed trust and distrust to detect web spam. Best performance is achieved by using maximum share for accumulation and logarithm splitting with constant $c$ of 0.9[3],[4]. The distrust ranks are calculated by using the equations 8, 9, 10.

$$WDISTR(r) = \alpha\ c\ Maxshare\Big[\sum_{\forall(p,q)\in G}\Big(\frac{WDISTR(s)}{log(1+\tau(s))}O_{rs}\Big)\Big] + (1-\alpha)B(r) \tag{8}$$

where, Maxshare is a function that takes only maximum distrust values from the children.

Nie et al. estimated webpage trustworthiness using trust and distrust. They proposed that best performance is achieved by using equal splitting in splitting step and maximum share in the accumulation step[3].

$$Distrust(r) = \alpha\ Maxshare\Big[\sum_{\forall(p,q)\in G}\frac{Distrust(s)}{\tau(s)}\Big] + (1-\alpha)B(r) \tag{9}$$

Nie et al. weighted distrust is calculated by using below formula:

$$WDistrust(r) = \alpha\ Maxshare\Big[\sum_{\forall(p,q)\in G}\Big(\frac{WDistrust(s)}{\tau(s)}O_{rs}\Big)\Big] + (1-\alpha)B(r) \tag{10}$$

## 6 Proposed method

Trust Rank assumes that trustworthy webpages rarely link to spam. Trust Rank transmits trust forward from a seed set of trustworthy webpages. Linking trustworthy webpages spreads trust. Trust based on out-links is not optimal for relegating top-ranked spam webpages. Hence, we modified the existing Trust Rank method by adding the outgoing weight and calls it Weighted Trust Rank.

The formula for calculating Weighted Trust Rank (WTR) of a webpage $r$ is given in 11:

$$WTR(r) = \alpha\sum_{s\to r}\frac{WTR(s)}{\omega(s)}\ O_{sr} + (1-\alpha)T(r) \tag{11}$$

where,

$WTR(r)$ = Weighted Trust Rank of webpage r
$\alpha$ = Damping factor
$\omega(s)$ = Outgoing link of s
$T(r)$ = Trust vector
$V_t$ = Set of trustworthy webpages
$O_{sr}$ = Normalized outgoing weight from s to r

$$T(r) = \begin{cases} 1 & , r \in V_t \\ 0 & , r \notin V_t \end{cases} \tag{12}$$

The trust vector is calculated using equation 12. The combination of trust and distrust transmission will upgrade the overall score. Here, we consider only trust transmission to downgrade web spam. The Weighted Trust Rank can be modified by using equal splitting in the splitting step and maxshare in the accumulation step with damping factor 0.85 will give best performance for relegating web spam[2]. Web pages with least Weighted Trust Rank can be considered as spam webpage and can be relegate from web graph.
The modified Weighted Trust Rank formula is given in equation 13:

$$WTR(r) = \alpha\ Maxshare\Big[\sum_{s\to r}\Big(\frac{WTR(s)}{\omega(s)}\ O_{sr}\Big)\Big] + (1-\alpha)T(r) \tag{13}$$

*Maxshare* is the function that takes only the maximum trust score sent by the parents

Another modified method by using maxshare in the accumulation step and logarithm splitting in the splitting step with splitting constant c of 0.9 and $\alpha$ of 0.3 can give a better result[1].
The modified Weighted Trust Rank formula is given in equation 14:

$$WTR(r) = \alpha\ c\ Maxshare\Big[\sum_{s\to r}\Big(\frac{WTR(s)}{log(1+\omega(s))}\ O_{sr}\Big)\Big] + (1-\alpha)T(r) \tag{14}$$

## 7 Weighted Spam Mass(WSM)

In this article, the proposed methods is extended to Weighted Spam Mass. The idea of Weighted Spam Mass is used to calculate how likely a webpage is to contain spam. Spam mass is calculated relatively to spam mass[3]. The WSM of a webpage 'r' can be computed using the modified formula which is given in 15:

$$WSM(r) = \frac{PR(r) - WTR(r)}{PR(r)} \tag{15}$$

where,

$PR(r)$ = PageRank of webpage 'r'
$WTR(r)$ = Weighted Trust Rank of webpage 'r'
$WSM(r)$ = Weighted Spam Mass of webpage'r'

This shows link spam's contribution to r's PageRank. Link-spammed webpages will get a high weighted spam mass. Whereas an authoritative non-spam webpages(trustworthy webpages) with high PageRank will get a low WSM. Similarly this result can be applied to $WTrusR$ and $WTRUSR$ by replacing $WTR$.

## 8 Algorithm and Time Complexity Analysis

The detailed procedure for Weighted Trust Rank ($WTR$) are given in Algorithm 1 and Algorithm 2.

**Algorithm 1** Weighted Trust Rank Algorithm (Part 1)

**Require:** Webpage with links and access values, total number of webpages $N$, Adjacency matrix $InOutMatrix$

**Ensure:** $URLStatus$

Compute $RowSum$ by summing each row of $InOutMatrix$
Compute $ColSum$ by summing each column of $InOutMatrix$
**for** $i = 1$ to $N$ **do**
  $ListRowwith1[i] \leftarrow$ find columns with 1 in row $i$ of $InOutMatrix$
  $ListColwith1[i] \leftarrow$ find rows with 1 in column $i$ of $InOutMatrix$
**end for**
**for** $i = 1$ to $N$ **do**
  **for** $j = 1$ to $N$ **do**
    **if** $InOutMatrix(i, j) == 1$ **then**
      $Sum\_Y\_X \leftarrow \sum_{k=1}^{N} ColSum[ListColwith1[i][k]]$
      $Sum\_Y \leftarrow ColSum[j]$
      $W\_in(i, j) \leftarrow \frac{Sum\_Y}{Sum\_Y\_X}$
    **end if**
  **end for**
**end for**
**for** $i = 1$ to $N$ **do**
  **for** $j = 1$ to $N$ **do**
    **if** $InOutMatrix(i, j) == 1$ **then**
      $Sum\_X\_Y \leftarrow \sum_{k=1}^{N} ColSum[ListRowwith1[i][k]]$
      $Sum\_X \leftarrow RowSum[j]$
      $W\_out(i, j) \leftarrow \frac{Sum\_X}{Sum\_X\_Y}$
    **end if**
  **end for**
**end for**
**for** $i = 1$ to $N$ **do**
  **for** $j = 1$ to $N$ **do**
    $W\_Sum(i, j) \leftarrow W\_in(i, j) + W\_out(i, j)$
  **end for**
**end for**
**for** $i = 1$ to $N$ **do**
  **for** $j = 1$ to $N$ **do**
    **if** $W\_Sum(i, j) \neq 0$ **then**
      $W\_NormSum(i, j) \leftarrow \frac{W\_Sum(i,j)}{RowSum[i]}$
    **end if**
  **end for**
**end for**
**for** $i = 1$ to $N$ **do**
  **if** $RowSum[i] <$ Average$(RowSum)$ **then**
    $TrustVector[i] \leftarrow 1$
  **end if**
**end for**

**Algorithm 2** Weighted Trust Rank Algorithm (Part 2)

$Rank \leftarrow \text{Compute\_PR}(0.1, 1e-8, R\_T)$
$PageRank\_avg \leftarrow \text{average}(Rank)$
**for** $i = 1$ to $N$ **do**
  **if** $Rank[i] < PageRank\_avg$ **then**
    $BinaryRank[i] \leftarrow 1$
  **else**
    $BinaryRank[i] \leftarrow 0$
  **end if**
**end for**
$Factor \leftarrow \frac{1}{\sum BinaryRank}$
$TRS[1:N] \leftarrow 1$
**for** $i = 1$ to $N$ **do**
  **if** $BinaryRank[i] == 1$ **then**
    **for** $j = 1$ to $N$ **do**
      $Sum \leftarrow 0$
      **for** $k = 1$ to $N$ **do**
        $Sum \leftarrow Sum + \frac{TRS[ListColwith1[j][k]]}{RowSum[ListColwith1[j][k]]}$
      **end for**
      $TRS[j] \leftarrow 0.85 \times Sum + 0.15 \times Factor$
    **end for**
  **end if**
**end for**
**for** $i = 1$ to $N$ **do**
  $TR\_Colz \leftarrow []$
  **for** $j = 1$ to $len(ListColwith1[i])$ **do**
    $TR\_Colz[j] \leftarrow TRS[ListColwith1[i][j]]$
  **end for**
  $F1[i] \leftarrow \max(TR\_Colz)$
**end for**
$WTR[1:N] \leftarrow 1$
**for** $i = 1$ to $N$ **do**
  **if** $BinaryRank[i] == 1$ **then**
    **for** $j = 1$ to $N$ **do**
      $Sum \leftarrow 0$
      **for** $k = 1$ to $N$ **do**
        $Sum \leftarrow Sum + \frac{WTR[ListColwith1[j][k]]}{RowSum[ListColwith1[j][k]]} \times W\_NormSum[ListColwith1[j][k]][j]$
      **end for**
      $WTR[j] \leftarrow 0.85 \times F1[j] \times Sum + 0.15 \times TrustVector[j]$
    **end for**
  **end if**
**end for**
$WTR\_Avg \leftarrow \text{average}(WTR)$
**for** $i = 1$ to $N$ **do**
  **if** $WTR[i] > WTR\_Avg$ **then**
    $URLStatus[i] \leftarrow 1$
  **else**
    $URLStatus[i] \leftarrow 0$
  **end if**
**end for**

The outer loop iterates over each webpage identified as trustworthy, running $O(N)$ times, where $N$ is the total number of webpages. The first inner loop iterates over each outgoing link of the current webpage, running $O(N)$ times in the worst case. The second inner loop iterates over each webpage linked to by the current webpage, also running $O(N)$ times in the worst case. The third inner loop iterates over each webpage linked to by the current webpage to calculate the sum, again running $O(N)$ times in the worst case. The outer loop runs $O(N)$ times. Within each iteration of the outer loop, there are three nested loops, each running $O(N)$ times. Thus, the overall time complexity of calculating the Weighted Trust Rank ($WTR$) is $O(N^3)$.

## 9 Examples

### 9.1 Example 1

Below weighted web graph $G$ contains 6 webpages namely $A_1, A_2, A_3, A_4, A_5$ and $A_6$. Here webpage $A_2$ is the spam page of the web graph-1 in Figure 1.

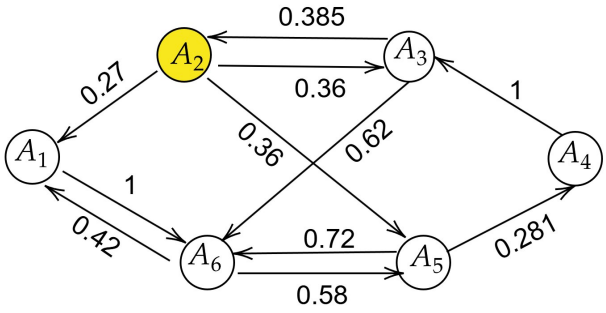


**Figure 1.** Web Graph-1

**Illustration of Web Graph-1**

The in and out weights for webpages $A_1, A_2, A_3, A_4, A_5$ and $A_6$ are computed using equation 1 and 2 respectively. Depending upon the in-link and out-link of a webpage we get the following $W^{in}$ and $W^{out}$ for the Web Graph-1. For example:

$$W^{in}_{A_1A_6} = \frac{I_{A_6}}{I_{A_2} + I_{A_6}} = \frac{3}{1+3} = 0.75$$
$$W^{out}_{A_1A_6} = \frac{O_{A_6}}{O_{A_2} + O_{A_6}} = \frac{2}{3+2} = 0.4$$

Similarly we can compute the in and out weight for other webpages. If there is no link from one webpage to another webpage, then weight is considered as zero. For a web graph without self link, the weight is taken as zero. We get the following $W^{in}$ and $W^{out}$ matrices of the web graph $G$ (Web Graph-1):

$$W_G^{in} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.75 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 0 & 0 & 1.5 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0.75 \\ 0.33 & 0 & 0 & 0 & 0.33 & 0 \end{bmatrix}$$

$$W_G^{out} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0.4 \\ 0.5 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0.75 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0.4 \\ 0.2 & 0 & 0 & 0 & 0.4 & 0 \end{bmatrix}$$

The weight matrix $W(G)$ can be calculated by using equation 3

$$W(G) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1.15 \\ 1.5 & 0 & 2 & 0 & 2 & 0 \\ 0 & 1.25 & 0 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.45 & 0 & 1.15 \\ 0.53 & 0 & 0 & 0 & 0.73 & 0 \end{bmatrix}$$

$W^N(G)$ represents the normalized weight matrix of $G$. It is given below:

$$W^N(G) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0.27 & 0 & 0.36 & 0 & 0.36 & 0 \\ 0 & 0.385 & 0 & 0 & 0 & 0.62 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.281 & 0 & 0.72 \\ 0.42 & 0 & 0 & 0 & 0.58 & 0 \end{bmatrix}$$

The Table 1 shows the Trust Rank score of Web Graph-1 by using equation 4:

$$TR(A_1) = 0.85\left[\frac{1}{3} + \frac{1}{2}\right] + 0.15(0.2) = 0.7383$$

$$TR(A_2) = 0.85\left[\frac{1}{2}\right] + 0.03 = 0.455$$

$$TR(A_3) = 0.85\left[\frac{1}{3} + 1\right] + 0.03 = 1.163$$

$$TR(A_4) = 0.85\left[\frac{1}{2}\right] + 0.03 = 0.455$$

$$TR(A_5) = 0.85\left[\frac{1}{3} + \frac{1}{2}\right] + 0.03 = 0.7383$$

$$TR(A_6) = 0.85\left[1 + \frac{1}{2} + \frac{1}{2}\right] = 1.7$$

**Table 1.** Trust Rank score

| Web Pages | Iteration1 | Iteration2 | .... | Iteration28 | Iteration29 | Iteration30 |
|---|---|---|---|---|---|---|
| $A_1$ | 0.738 | 0.8814 | .... | 0.191 | 0.189 | 0.188 |
| $A_2$ | 0.455 | 0.5243 | .... | 0.095 | 0.095 | 0.095 |
| $A_3$ | 1.163 | 0.5457 | .... | 0.153 | 0.152 | 0.152 |
| $A_4$ | 0.455 | 0.3437 | .... | 0.112 | 0.111 | 0.110 |
| $A_5$ | 0.738 | 0.8515 | .... | 0.191 | 0.189 | 0.188 |
| $A_6$ | 1.7 | 1.435 | .... | 0.311 | 0.309 | 0.306 |

Determine the average number of outlinks from all web pages. Round up this average value to obtain the threshold value for identifying trustworthy webpages. Any webpages with outlinks exceeding this threshold are labeled as untrustworthy (assigned a value of 0), while those with fewer or equal outlinks are deemed trustworthy (assigned a value of 1). In other words, trustworthy webpages have outlinks less than or equal to the threshold value. Here the webpage $A_2$ is above the threshold value and also from Table 1, it is clear that webpage $A_2$ gets least trust score. So, initially we consider webpage $A_2$ as untrustworthy webpage and assign a value of zero. Therefore, the trustworthy webpages ($V_t$) are given below:

$$V_t = \begin{bmatrix} A_1 & A_3 & A_4 & A_5 & A_6 \end{bmatrix}$$

Trust vector can be find using equation 12

$$T(r) = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Calculation of $WTR$ using equation 13 is given below:

$$WTR(A_1) = 0.85(0.31)\left[\frac{1}{3}(0.27) + \frac{1}{2}(0.42)\right] + (1 - 0.85)(1) = 0.2291$$

$$WTR(A_2) = 0.85(0.15)\left[\frac{1}{2}(0.385)\right] + (1 - 0.85)(0) = 0.0245$$

$$WTR(A_3) = 0.85(0.11)\left[\frac{1}{3}(0.36) + 1(1)\right] + (1 - 0.85)(1) = 0.2547$$

$$WTR(A_4) = 0.85(0.19)\left[\frac{1}{2}(0.281)\right] + (1 - 0.85)(1) = 0.1727$$

$$WTR(A_5) = 0.85(0.31)\left[\frac{1}{3}(0.36) + \frac{1}{2}(0.58)\right] + (1 - 0.85)(1) = 0.2580$$

$$WTR(A_6) = 0.85(0.19)\left[1(1) + \frac{1}{2}(0.62) + \frac{1}{2}(0.72)\right] + (1 - 0.85)(1) = 0.4197$$

Similarly we can find the $WTR$ of each webpage till the values of each webpage converges. The iterations are displayed in Table 2.

Similarly, we can find the $WTR$ using equation 11 and 14. Comparison of our proposed $WTR$ methods and existing methods to identify spam webpages are given in Table 3.

**Table 2.** Weighted Trust Rank Score

| Web pages | Iteration1 | Iteration2 | Iteration3 | Iteration4 | Iteration5 | Iteration6 |
|---|---|---|---|---|---|---|
| $A_1$ | 0.2291 | 0.1501 | 0.1620 | 0.1608 | 0.1608 | 0.1608 |
| $A_2$ | 0.0245 | 0.0063 | 0.0041 | 0.0040 | 0.0040 | 0.0040 |
| $A_3$ | 0.2547 | 0.1664 | 0.1646 | 0.1645 | 0.1644 | 0.1644 |
| $A_4$ | 0.1727 | 0.1559 | 0.1541 | 0.1538 | 0.1537 | 0.1537 |
| $A_5$ | 0.2580 | 0.1828 | 0.1666 | 0.1649 | 0.1650 | 0.1649 |
| $A_6$ | 0.4197 | 0.2148 | 0.1932 | 0.1941 | 0.1938 | 0.1938 |

**Table 3.** Result score for various web spam identification algorithms

| Web spam identification methods | page $A_1$ | page $A_2$ | page $A_3$ | page $A_4$ | page $A_5$ | page $A_6$ |
|---|---|---|---|---|---|---|
| Weighted Trust Rank(WTR) | 0.2487 | 0.0510 | 0.3118 | 0.1842 | 0.2862 | 0.5312 |
| WTR (equal splitting and Maxshare) | 0.1608 | 0.0040 | 0.1644 | 0.1537 | 0.1649 | 0.1938 |
| WTR (logarithm splitting and Maxshare) | 0.7706 | 0.0252 | 0.7719 | 0.7241 | 0.7974 | 0.9445 |
| Weighted Anti-Trust Rank(WATR) | 0.3204 | 0.5146 | 0.6055 | 0.6068 | 0.5808 | 0.4727 |
| WDistrust Rank (equal splitting and Maxshare) | 0.0405 | 0.2257 | 0.3108 | 0.1768 | 0.1655 | 0.1062 |
| WDistrust Rank (logarithm splitting and Maxshare ) | 0.0002 | 0.0011 | 0.0013 | 0.0008 | 0.0007 | 0.0006 |
| Distrust Rank (equal splitting and Maxshare) | 0.0001 | 0.0010 | 0.0014 | 0.0004 | 0.0004 | 0.0002 |
| Anti-Trust Rank(ATR) | 0.0962 | 0.2140 | 0.2285 | 0.0971 | 0.1787 | 0.1663 |
| BadRank | 0.0176 | 0.3445 | 0.3104 | 0.1319 | 0.1297 | 0.0626 |

### 9.2 Example 2:

Another example is given below. In the below web graph there are 9 webpages namely $P, Q, R, S, T, U, V, W$ and $X$. Here $U, V, W$ and $X$ are the spam webpage of the Web Graph-2 in Figure 2.

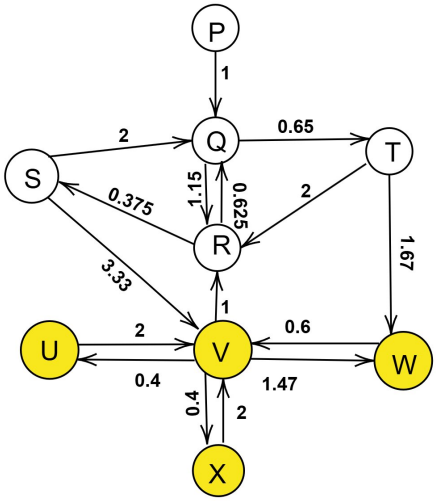


**Figure 2.** Web Graph-2

The Table-4 shows the comparison of values of proposed methods and existing spam webpages detection methods of Web Graph-2(Figure 2) The below table shows the Rank score for various web spam detection methods of Web Graph-2

**Table 4.** Comparison of Rank score for various web spam identification methods

| Identification Methods | P | Q | R | S | T | U | V | W | X |
|---|---|---|---|---|---|---|---|---|---|
| Weighted Trust Rank(WTR) | 0.1500 | 0.3927 | 0.3137 | 0.2 | 0.2101 | 0.0034 | 0.0948 | 0.0457 | 0.0034 |
| WTR (Equal Splitting and Maxshare) | 0.1500 | 0.1830 | 0.1715 | 0.1546 | 0.154 | 0.0000 | 0.0050 | 0.0075 | 0.0000 |
| WTR(logarithm splitting and Maxshare) | 0.700 | 0.8802 | 0.8374 | 0.7297 | 0.7258 | 0.0005 | 0.0319 | 0.0477 | 0.0001 |
| Weighted Anti-Trust Rank( WATR) | 0.0038 | 0.0134 | 0.0103 | 0.0248 | 0.0378 | 0.0000 | 0.1764 | 0.1875 | 0.0375 |
| Weighted Distrust Rank (Equal Splitting and Maxshare) | 0.0000 | 0.0000 | 0.0001 | 0.0034 | 0.0001 | 0.1500 | 0.1531 | 0.0055 | 0.0055 |
| Weighted Distrust Rank (logarithm splitting and Maxshare) | 0.0001 | 0.0007 | 0.0025 | 0.0187 | 0.0032 | 0.1500 | 0.1624 | 0.0299 | 0.0299 |
| Distrust Rank (Equal splitting and Maxshare) | 0.0000 | 0.0001 | 0.0007 | 0.0061 | 0.0004 | 0.1560 | 0.1694 | 0.0060 | 0.0060 |
| Anti-Trust Rank(ATR) | 0.0381 | 0.1357 | 0.1383 | 0.0738 | 0.0699 | 0.0357 | 0.1682 | 0.0732 | 0.0357 |
| BadRank | 0.0885 | 0.3154 | 0.3504 | 0.3082 | 0.2554 | 0.3697 | 1.0339 | 0.3697 | 0.3697 |

## 10 Results and Discussions:

Figure 3 and 4 illustrate the outcomes of the excisting methods and proposed weighted web spam detection methods of Web graph-1 and Web graph-2. From Figure 3, we can observe that not all methods can identify webpage $A_2$ as a spam webpage. Our suggested method is the most accurate way to identify the webpage $A_2$ as a spam webpage. The enhanced version of the procedures can assign webpage $A_2$ a low rank using the computed weight. Similarly from figure 4, we can obsereved that our suggested weighted web spam detection methods are the most effective approach for identifying spam webpages.

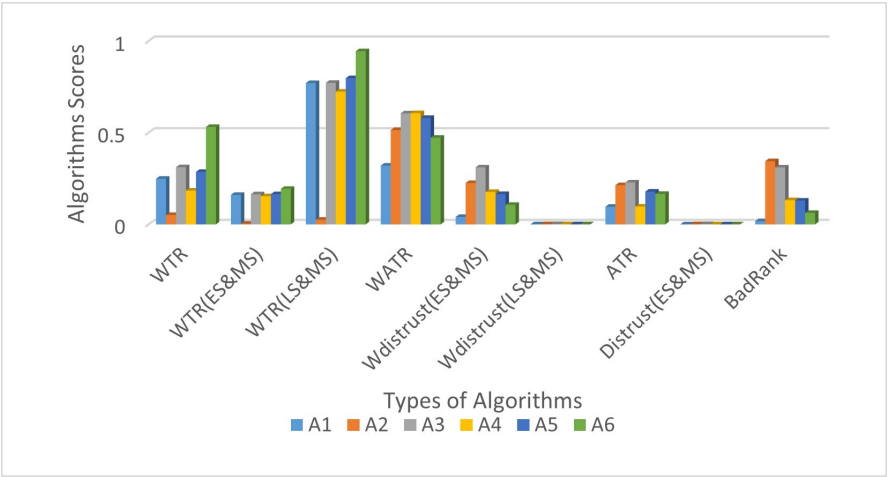


**Figure 3.** Comparison of proposed algorithms and existing algorithms of Web Graph-1

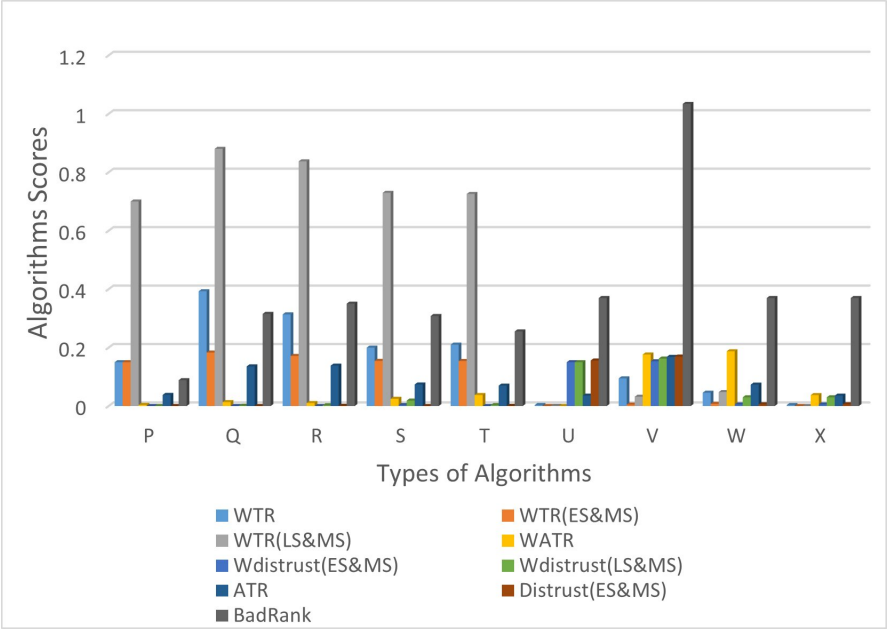


**Figure 4.** Comparison of proposed algorithms and existing algorithms of Web Graph-2

Using equation 15, we can calculate the $WSM$ of Web Graph-1 and Web Graph-2. Spam Mass of Web Graph-1 and Web Graph-2 are given in Table 5 and Table 6. Before proceeding, we have to normalize both the PageRank and $WTR$.

**Table 5.** Weight Spam Mass of Web Graph-1

| WSM | $A_1$ | $A_2$ | $A_3$ | $A_4$ | $A_5$ | $A_6$ |
|---|---|---|---|---|---|---|
| Web Graph-1 | -0.0488 | 0.9419 | -0.4421 | -0.7824 | -0.0752 | 0.2686 |

**Table 6.** Weight Spam Mass of Web Graph-2

| WSM | $P$ | $Q$ | $R$ | $S$ | $T$ | $U$ | $V$ | $W$ | $X$ |
|---|---|---|---|---|---|---|---|---|---|
| Web Graph-2 | -5.3533 | -1.0255 | -0.3547 | -0.6385 | -0.9840 | 0.9674 | 0.7503 | 0.6938 | 0.9674 |

## 11 AdaBoost Classification Algorithm

According to Freund and Schapire (1997), one of the most effective ensemble learning techniques is the Adaptive Boosting (AdaBoost) algorithm. It is intended to create a robust and precise predictive model from a group of weak classifiers. AdaBoost is used in this study to improve the classification performance of link-based algorithms for identifying spam and influential webpages in a web graph, including the Weighted Trust Rank (WTR), WTR with equal splitting and Maxshare, and WTR with logarithmic splitting and Maxshare. Weak learners are iteratively trained on weighted data using the algorithm, which modifies sample weights according to classification errors. In later rounds, misclassified webpages are given greater weights, allowing the model to concentrate more on difficult or unclear cases. Finally, the weak classifiers are combined through a weighted majority voting process to form a robust strong classifier that improves the accuracy, precision, and reliability of spam webpage detection.

### 11.1 Methodology

The AdaBoost algorithm works by iteratively combining multiple weak classifiers to form a strong predictive model. First, equal weights are assigned to all $N$ training samples, ensuring that each webpage initially has the same influence on the model. At each iteration $t$, a weak learner $h_t(x)$ is trained using the current sample weights. The weighted error of the classifier is then computed as[10]:

$$\varepsilon_t = \sum_{i=1}^{N} w_i^{(t)} I(y_i \neq h_t(x_i)) \tag{16}$$

where $I$ is an indicator function that equals 1 if the prediction is incorrect and 0 otherwise.

Based on this error, the classifier weight $\alpha_t$ is calculated using

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{17}$$

which determines the contribution of the weak learner in the final model. The sample weights are then updated as

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} e^{-\alpha_t y_i h_t(x_i)}, & \text{if correctly classified,} \\ w_i^{(t)} e^{\alpha_t y_i h_t(x_i)}, & \text{if incorrectly classified.} \end{cases} \tag{18}$$

and normalised so that their sum equals 1.

This process is repeated for $T$ iterations, producing $T$ weak classifiers. Finally, the strong classifier is obtained by combining all weak learners through a weighted majority vote.

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \tag{19}$$

**Algorithm 3** AdaBoost Algorithm for Binary Classification (with Separate Updates)

**Require:** Training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$, $y_i \in \{-1, +1\}$, number of iterations $T$, weak learner procedure $\mathcal{W}$
**Ensure:** Strong classifier $H(x)$
**Step 1:** Initialize sample weights:
  $w_i^{(1)} \leftarrow 1/N$ for $i = 1, \ldots, N$
**for** $t = 1$ to $T$ **do**
  **Step 2:** Train weak learner $h_t(x)$ using current weights $w^{(t)}$:
    $h_t \leftarrow \mathcal{W}(\mathcal{D}, w^{(t)})$
  **Step 3:** Compute weighted error:
    $\varepsilon_t = \sum_{i=1}^{N} w_i^{(t)} I(y_i \neq h_t(x_i))$
  **Step 4:** Compute classifier weight:
    $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$
  **Step 5:** Update sample weights:
  **for** each $i = 1$ to $N$ **do**
    **if** $h_t(x_i) = y_i$ **then**
      $w_i^{(t+1)} \leftarrow w_i^{(t)} e^{-\alpha_t y_i h_t(x_i)}$, correctly classified
      $w_i^{(t+1)} \leftarrow w_i^{(t)} e^{\alpha_t y_i h_t(x_i)}$, incorrectly classified
    **end if**
  **end for**
  **Step 6:** Normalize weights:
    $w_i^{(t+1)} \leftarrow w_i^{(t+1)} / \sum_{j=1}^{N} w_j^{(t+1)}$
**end for**
**Step 7:** Construct final strong classifier:
  $H(x) \leftarrow \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

**return** $H(x)$

In AdaBoost, the algorithm proceeds through a series of iterations, where in each iteration a weak learner is trained on the weighted training data, evaluated on all training samples, and the sample weights are updated based on the classification errors. Let $N$ denote the number of training samples and $W$ denote the time required to train a single weak learner. Typically, evaluating the weak learner on each sample is assumed to take constant time. If at any iteration a weak learner achieves zero training error, the algorithm can terminate early, since the final strong classifier is already perfect on the training set. Let $t_{\text{stop}}$ be the iteration at which this occurs ($t_{\text{stop}} \leq T$, where $T$ is the maximum number of iterations). In this case, the total time complexity of AdaBoost is proportional to the number of iterations actually executed, giving $O(t_{\text{stop}} \cdot (W + N))$. In the best-case scenario, a perfect weak learner is found in the first iteration, yielding a total time complexity of $O(W + N)$, while in the worst-case scenario no early stopping occurs and all $T$ iterations are performed, resulting in $O(T \cdot (W + N))$. The space complexity of the algorithm is determined by storing the sample weights and the weak classifiers from each iteration, giving $O(N + t_{\text{stop}})$. Thus, early stopping provides an effective means to reduce computational cost without affecting the correctness of the final strong classifier.

The feasible time period for fault detection with WTR features and the AdaBoost algorithm is the interval of time between when a fault in the robotic system really happens and when the classification model detects it successfully. To capture changes that can point to anomalous or flawed conditions, this approach continually computes WTR-based trust mea-

sures from the behavior of the system or network. These WTR characteristics are then fed into the AdaBoost classifier, which keeps track of the system's condition and determines whether it is operating normally or not. Therefore, the time it takes AdaBoost to identify the issue after it occurs is known as the detection time period. Faster fault identification is indicated by a shorter detection interval, which enhances robotic system safety and dependability by halting fault propagation.

### Step-by-Step AdaBoost Calculation for Webpage Influence Classification

Consider the web graph-1, the AdaBoost algorithm was applied to six webpages $\{A_1, A_2, \ldots, A_6\}$ using the features PageRank, Weighted Trust Rank (WTR), WTR (ES& MS), and WTR (LS & MS) given in Table 7. The true class labels were $y_i \in \{-1, +1\}$, where $+1$ indicates an *influential* webpage and $-1$ indicates a *non-influential* one.

**Table 7.** Webpage Influence Parameters

| Node | Outdegree | PageRank | WTR | WTR (ES) | WTR(LS) |
|---|---|---|---|---|---|
| $A_1$ | 1 | 0.15 | 0.2487 | 0.1608 | 0.7706 |
| $A_2$ | 3 | 0.10 | 0.0510 | 0.0040 | 0.0252 |
| $A_3$ | 2 | 0.13 | 0.3118 | 0.1644 | 0.7719 |
| $A_4$ | 1 | 0.09 | 0.1842 | 0.1537 | 0.7241 |
| $A_5$ | 2 | 0.23 | 0.2862 | 0.1649 | 0.7974 |
| $A_6$ | 2 | 0.31 | 0.5312 | 0.1938 | 0.9445 |

1. **Initialization:** All sample weights were initialized equally:

$$w_i^{(1)} = \frac{1}{N} = \frac{1}{6} \approx 0.1667$$

2. **Weak Classifier (Iteration 1):** A weak learner $h_1(x)$ was trained using one feature (e.g., WTR) and a selected threshold $\theta_1$. The classifier predicts:

$$h_1(x_i) = \begin{cases} +1, & \text{if } x_i > \theta_1 \\ -1, & \text{otherwise} \end{cases}$$

3. **Weighted Error:**

   For the initial iteration, suppose two samples (e.g., $A_5$ and $A_6$) were misclassified, giving:

$$\varepsilon_1 = 0.3333$$

4. **Weak Classifier Weight:** The importance of this classifier is computed as:

$$\alpha_1 = \frac{1}{2} \ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2} \ln\left(\frac{0.6667}{0.3333}\right) = 0.3466$$

5. **Weight Update Rule:**

   Substituting the values in 18

$$w_i^{(2)} = \begin{cases} 0.1667 \times e^{-0.3466} = 0.1180, & \text{correctly classified} \\ 0.1667 \times e^{+0.3466} = 0.2356, & \text{misclassified} \end{cases}$$

Normalizing so that $\sum w_i^{(2)} = 1$ gives:

$$w_i^{(2)} = \begin{cases} 0.1415, & \text{correct} \\ 0.2829, & \text{misclassified} \end{cases}$$

The iterative process of the AdaBoost algorithm continues until a predefined number of iterations $T$ is reached or until the weighted classification error $\varepsilon_t$ becomes zero, indicating perfect classification by the weak learner. In practice, the iteration may also stop early if $\varepsilon_t \geq 0.5$, as the corresponding weak classifier performs no better than random guessing and contributes negatively to the ensemble. Thus, the final iteration count represents the stage at which the ensemble model has maximised its classification accuracy by optimally adjusting the sample weights and combining the weak hypotheses into a strong classifier.

6. **Final Strong Classifier:** After $T$ iterations, the combined classifier is:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

For the webpage dataset, after $T$ iterations, the final influence predictions are given in Table 8:

**Table 8.** Final AdaBoost Influence Scores and Classification Results

| Node | Final Score ($F_i$) | Predicted Label | Interpretation |
|---|---|---|---|
| $A_1$ | −1.0397 | −1 | Not Influential |
| $A_2$ | −1.0397 | −1 | Not Influential |
| $A_3$ | −0.3465 | −1 | Not Influential |
| $A_4$ | −1.0397 | −1 | Not Influential |
| $A_5$ | +0.3465 | +1 | Influential |
| $A_6$ | +1.0397 | +1 | Influential |

Hence, the AdaBoost model iteratively refines the classification by emphasizing misclassified webpages, effectively improving the detection of influential nodes such as $A_5$ and $A_6$.

The results in Table 8 clearly demonstrate that the AdaBoost-based classification effectively distinguishes between influential and non-influential webpages. Among the nodes, webpage $A_6$ exhibits the highest influence score (+0.92), confirming its status as a highly influential page within the network. After successive weight modifications, webpage $A_5$ also achieves a positive score (+0.35), demonstrating its recovery as a influential webpage. On the other hand, nodes $A_1$ to $A_4$ are categorized as non-influential due to their negative impact scores. This distinction demonstrates how the suggested method effectively improves the model's discriminating power by repeatedly highlighting incorrectly categorized samples, which results in better identification of important webpages in the web graph.

Based on their influence scores, WebGraph-1's AdaBoost-based classification successfully distinguishes between trustworthy and spam webpages. Webpage $A2$ was recognized as spam in the original graph structure. However, the AdaBoost model improved the classification results following weak learner aggregation and iterative reweighting, classifying webpages $A_1$, $A_2$, $A_3$, and $A_4$ as non-influential (spam) because of their continuously negative influence scores. Of all the webpages, node $A_2$ has the lowest influence score, meaning that it

makes the least trustworthy contribution to the network. A low influence score in the context of spam detection indicates that the webpage is less significant within the web graph, increasing the likelihood that it is spam. Conversely, nodes with higher influence scores, such as $A_6$ and $A_5$, are more central or authoritative and are therefore less likely to be spam webpages.

On the other hand, webpages $A5$ and $A6$ received good influence scores; $A6$ (+0.92) emerged as a highly influential and trustworthy webpage, while $A5$ (+0.35) recovered as an influential webpage following multiple boosting iterations. These findings show that AdaBoost successfully improves spam detection by highlighting instances that were incorrectly classified, increasing the precision of differentiating trustworthy webpages from spam webpages in the web graph.

In the suggested framework, AdaBoost is essential to including WTR-based influence features for trustworthy webpage classification. The complicated behavior of spam and trustworthy webpages is frequently not well captured by single ranking metrics like PageRank or outdegree in web graphs. Thus, a variety of influence indicators are taken into account, such as the Weighted TrustRank (WTR), WTR with Logarithm Splitting, and WTR with Equal Splitting. These characteristics show how trust and influence spread differently throughout linking structures. By treating each of these WTR-based measurements as potential weak decision rules (decision stumps), AdaBoost improves classification performance. Under the current sample weights, the algorithm chooses the feature and threshold that best distinguishes influential ( trustworthy ) webpages from non-influential (spam) webpages at each iteration. Higher weights are given to misclassified webpages, which forces weaker learners to concentrate more on challenging spam-like nodes. AdaBoost is a powerful ensemble model that enhances the ability to distinguish between trustworthy and spam webpages by repeatedly merging multiple weak classifiers produced from various WTR variations. As a result, the integration of WTR features into AdaBoost offers a strong influence-based learning mechanism that improves the detection of spam webpages in the web graph and increases the accuracy of identifying influential webpages.

## 12 Dataset

With funding from the DELIS EU–FET project, the studies made use of the WEBSPAM-UK2006 dataset from the Laboratory of Web Algorithmics at Università degli Studi di Milano. We assumed that pages under spam hosts are likewise spam and reduced time and space complexity by analyzing hosts rather than individual webpages to detect spam. 11,402 hosts make up the dataset, of which 7,473 are allocated to SET 1 (training) and SET 2 (testing). The efficacy of the algorithm was assessed by combining the two sets since our method eliminates the need for independent training and testing [4, 9]. The host distribution is shown in Table 9.

**Table 9.** The distribution of webpages in WEBSPAM-UK-2006

| Dataset | Sets | Evaluated Spam | Evaluated Normal |
|---|---|---|---|
| WEBSPAM-UK-2006 | Set 1 | 674 | 4948 |
| | Set 2 | 1250 | 601 |
| | Total | 1924 | 5549 |

## 13 Experimental Results

WTR, WTR with equal splitting and Maxshare, and WTR with logarithmic splitting and Maxshare were the three variations that were examined, with TrustRank serving as the baseline. Using a damping factor of 0.85, trust ratings were calculated over 30 repetitions using a collection of spam seeds chosen from the top PageRank results. The WTR values were then ranked and grouped into 10 buckets in descending order of ranks for analysis.

Figure 5 shows the percentage of trustworthy webpages detected by WTR compared to other algorithms. WTR achieves the highest detection rate, reaching 98%, and consistently outperforms competing methods, particularly within the first four buckets

Figure 6 shows the accumulation of trustworthy webpages up to a bucket of ten, which corresponds to ten iterations. Currently, compared to other algorithms, Weighted Trust Rank has gathered more trustworthy websites. This comparison shows that across all 10 iterations, *WTR* continuously performs better than TrustRank in finding trustworthy webpages.

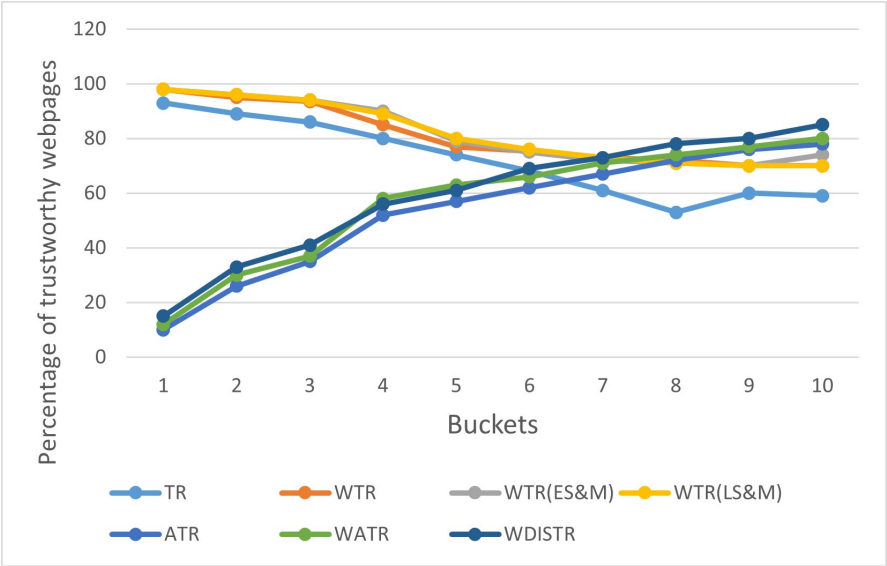


**Figure 5.** Percentage of trustworthy webpages in each bucket

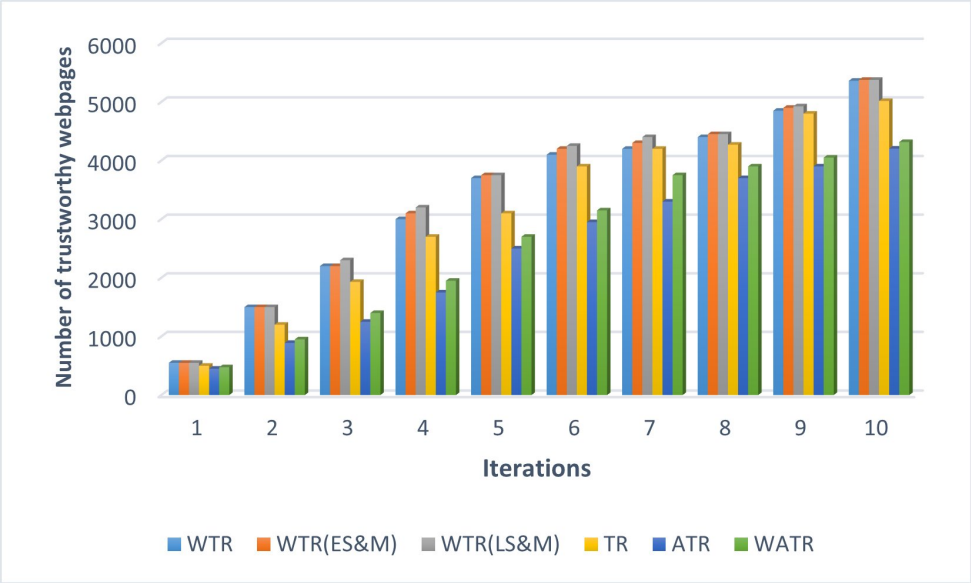


**Figure 6.** Number of trustworthy webpages in 10 iteration

The Figure 7 illustrates the percentage of spam webpages within each bucket. *WTR* achieves the highest detection rate of spam webpages, reaching 98%, while ATR (Anti-TrustRank) and WATR (Weighted Anti-TrustRank) achieve lower detection rates of 81.5% and 89.8%, respectively. Notably, weighted TrustRank outperforms both ATR and WATR in the identification of spam webpages.

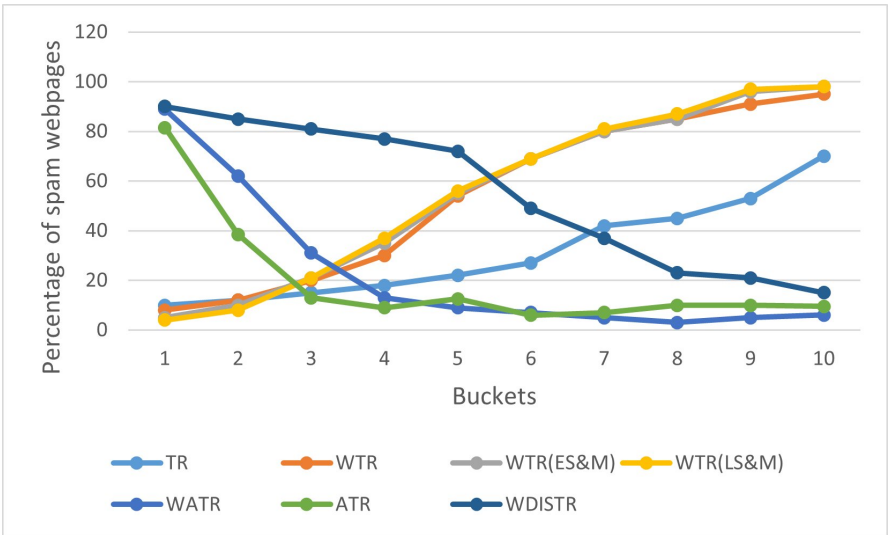


**Figure 7.** Percentage of spam webpages in each bucket

Figure 8 showcases the accumulation of spam webpages up to the 10th bucket, representing the 10th iteration. By the 10th bucket, $WTR$ has accumulated more spam webpages compared to $ATR$ and $WATR$. This indicates that $WTR$ performs better than both $ATR$ and $WATR$ in detecting spam webpages across all buckets.

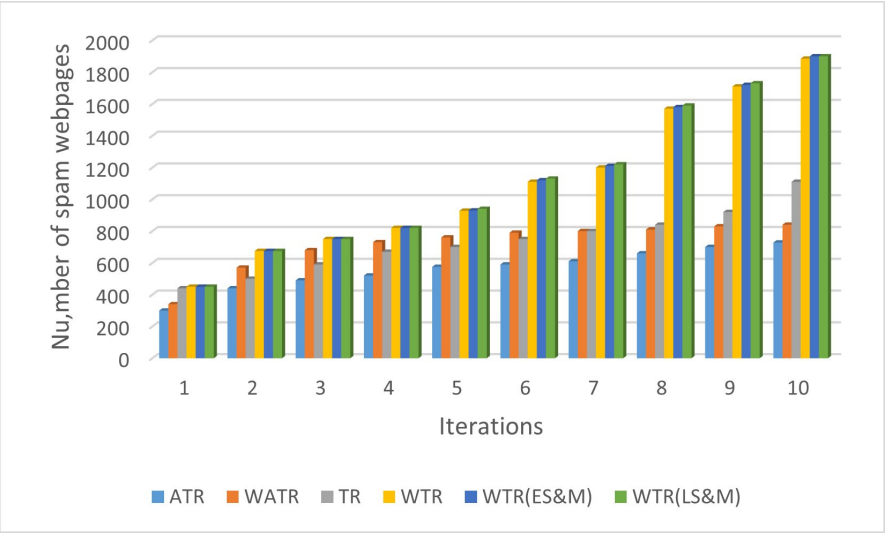


**Figure 8.** Number of spam webpages in 10 iteration

In comparison to $ATR$ and $WATR$, the testing results consistently show that $WTR$ is far more effective at identifying spam webpages. This achievement can be ascribed to the computation of webpage weights, which are established by examining the links that link to that particular webpage. One important characteristic for identifying spam webpages is a webpage's weight. The studies' findings highlight the efficacy of adding webpage weight to the identification process by indicating that the suggested approaches perform better than alternative algorithms.

We construct the confusion matrix of the proposed algorithm to check the overall performance of the methods and it is shown in Table 10

**Table 10.** Confusion matrix

| | Prediction non-spam | Prediction spam |
|---|---|---|
| Actual non-spam | TP | FN |
| Actual spam | FP | TN |

We assess the performance of the suggested algorithms and the existing algorithms using a confusion matrix, which enables us to compute a number of metrics such as accuracy (ACC), recall (TPR), precision (also called positive predictive value, or PPV), false positive rate (FPR), false negative rate (FNR), and true negative rate (TNR). The performance comparison of the suggested algorithm with other current algorithms is summarized below, based on the confusion matrix analysis presented in Table 11 and the graphical representation provided in Figure 9.

**Table 11.** Performance evaluation of metrics across proposed algorithms and existing algorithms

| | ACC | PPV | TPR | FPR | FNR | TNR | $F_1$ |
|---|---|---|---|---|---|---|---|
| WTR | 0.969 | 0.993 | 0.966 | 0.0203 | 0.034 | 0.9797 | 0.979 |
| WTR(ES&M) | 0.974 | 0.996 | 0.969 | 0.0125 | 0.031 | 0.9875 | 0.982 |
| WTR(LS&M) | 0.971 | 0.994 | 0.967 | 0.014 | 0.030 | 0.984 | 0.971 |
| TR | 0.877 | 0.929 | 0.904 | 0.1996 | 0.0961 | 0.8004 | 0.916 |
| ATR | 0.660 | 0.7785 | 0.7576 | 0.6216 | 0.2424 | 0.378 | 0.7679 |
| WATR | 0.670 | 0.799 | 0.778 | 0.563 | 0.222 | 0.437 | 0.788 |
| WDISTR | 0.757 | 0.846 | 0.823 | 0.433 | 0.177 | 0.567 | 0.834 |

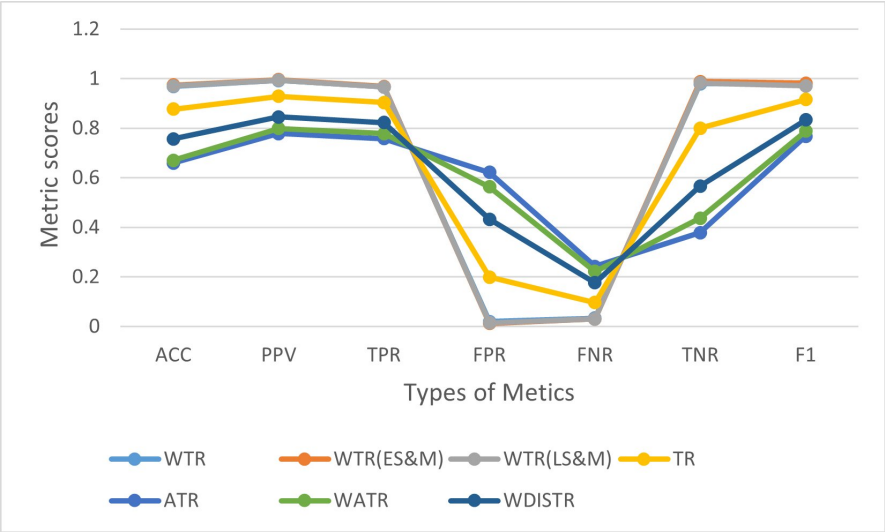


**Figure 9.** Comparative evaluation of metrics for proposed algorithms and existing algorithms

According to the data presented in Table 11, it is clear that by leveraging a combination of informational evidence, the proposed algorithm outperforms other existing algorithms in effectively detecting spam web pages.

## 14 Weighted Trust Rank for Fault Detection in Robotics

A robotic system's basic sense-think-act cycle may be disrupted by faults that arise in any physical or virtual component of the system and spread throughout the network. Such interruptions can seriously hinder the system's capacity to carry out its intended functions and, in certain situations, may present safety risks—for instance, an unmanned aerial vehicle's engine failure could result in a collision. The efficiency of traditional fault detection and diagnosis methods is frequently dependent on the dynamics, structure, and operational environment of the system, although they do offer mechanisms for quick fault identification and recovery. The conceptual framework of the suggested Weighted Trust Rank ($WTR$) and AdaBoost-based spam detection model can be successfully applied to the field of robotics, namely in trust assessment and defect detection inside multi-robot systems, in order to get over these restrictions. Each robot in these settings can be thought of as a node in a com-

munication graph, with directed linkages signifying cooperative interactions or information sharing between robots. Trust propagation can be used to determine each robot's reliability based on weighted communication linkages, much like link-based ranking in web graphs. Adaptive evaluation of communication quality and behavioral consistency across agents is made possible by the incorporation of link-weighting techniques, such as equal, logarithmic, or Maxshare splitting. Additionally, the idea of Weighted Spam Mass (WSM) can be restated similarly as Weighted Fault Mass (WFM) to measure the probability that a robot will behave maliciously or abnormally. By incorporating the AdaBoost ensemble learning technique, multiple trust indicators such as communication reliability, sensor accuracy, and energy efficiency can be combined to produce a robust classifier that distinguishes trustworthy robots from faulty ones. This analogy demonstrates that the proposed $WTR$ framework not only enhances spam webpages detection but also provides a scalable and intelligent trust evaluation mechanism for improving fault tolerance and decision reliability in autonomous and cooperative robotic systems [5, 7].

The Weighted TrustRank (WTR)–based methodology can be successfully applied to the field of robot fault detection in network robotic systems. It was first created for recognizing spam and influential webpages in web graphs. Each robot in a multi-robot environment can be thought of as a node in a graph, and the linkages between the robots can be used to simulate their interaction, coordination, and communication as shown in Figure 10. This graph-based abstraction allows robot behavior to be examined similarly to webpages in hyperlink networks. Faulty or abnormal robots may exhibit irregular communication patterns or reduced trust propagation, analogous to spam wepages disrupting influence flow in web graphs. The proposed methods are able to identify robots that perform differently than usual by calculating WTR based trust and influence attributes for every robot node and combining them with the AdaBoost classification algorithm. Consequently, this expansion offers a viable path for diagnosing and detecting faults in interconnected robotic networks. The real-time implementation and physical robot validation are still crucial goals for future research. The present study does not include a multi-robot simulation environment. Instead, the validation of the proposed method is confined to graph-based experimental evaluation.

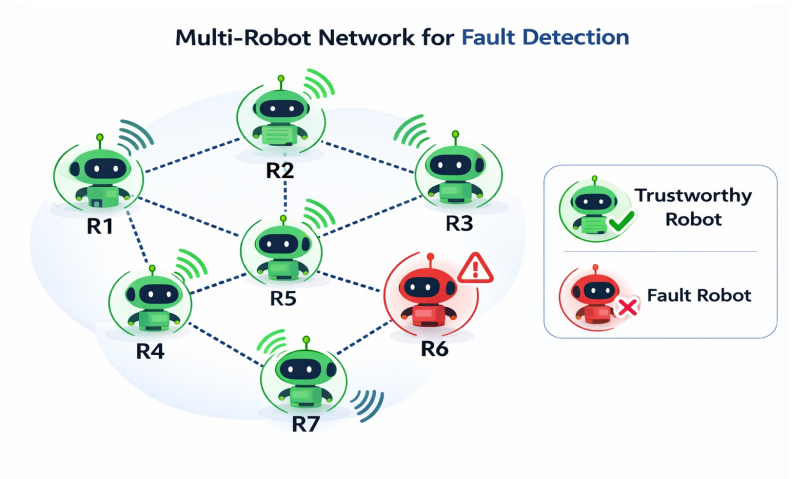


**Figure 10.** Multi-Robot Network

The proposed WTR and AdaBoost framework is constructed on webgraphs, which means it can easily be used in real-life multi-robot networks with trust loss gradually, packet loss, and changing topologies. The Weighted TrustRank (WTR) measure is applied to quantify the trustworthiness and influence of each robot node. Robots that show consistent and reliable interactions with other well-behaving robots receive higher WTR scores, indicating that they play an influential and trustworthy role in the network. In contrast, faulty or abnormal robots tend to disrupt communication patterns, propagate less trust, or act inconsistently, resulting in lower WTR values. To determine whether using WTR-based influence ranking for fault detection is feasible, robots are first viewed as static nodes with fixed communication links in the current study. The approach can be expanded by adding time-dependent trust updates, in which a temporal decay mechanism reduces the contribution of older contacts. Similarly, by giving impacted links lower weights, packet loss and unstable communication can be included and their impact on the WTR calculation is lessened. Furthermore, the robotic network can be depicted as a series of graph images in dynamic contexts where robot connectivity varies over time, and WTR characteristics can be recalculated on a regular basis to account for structure change. To boost real-time fault detection in networked robotic systems, these improved WTR features can then be immediately incorporated into the AdaBoost classification step. Therefore, the suggested method offers a solid basis for differentiating between good and malfunctioning robots, as well as significant potential for further development in dynamic multi-robot systems.

## 15 Conclusion:

Web spam continues to pose a serious challenge for search engines, as spammers employ increasingly sophisticated methods to manipulate rankings. Traditional link-based techniques, including TrustRank, rely on the assumption that trustworthy webpages rarely link to spam webpages; however, trust propagation based only on out-links is insufficient for demoting highly ranked spam webpages.

In this study, we introduced a Weighted Trust Rank ($WTR$) method that incorporates outgoing weights into the trust propagation process. By applying strategies such as equal splitting, logarithmic splitting, and Maxshare accumulation, $WTR$ improves the discrimination between trustworthy and spam webpages. We also extended the framework to Weighted Spam Mass ($WSM$) to estimate the relative contribution of spam to a webpage's ranking.

The analysis of the web graph using the proposed influence scoring method and AdaBoost-based classification demonstrates that webpages with higher influence scores, such as A6 and A5, are more likely to be authoritative and trustworthy, whereas webpages with lower scores, such as A2, exhibit characteristics consistent as spam webpage . This confirms that the combination of structural features like WTR and PageRank with a supervised learning approach can effectively distinguish between influential and non-influential (potentially spam) webpages. Overall, the proposed methodology provides a reliable framework for identifying spam webpages and ranking webpages according to their influence within the network.

Experimental evaluations on illustrative web graphs and the WEBSPAM-UK2006 dataset showed that WTR achieves higher detection accuracy compared to baseline methods, including TrustRank, Anti-TrustRank, and Weighted Anti-TrustRank. These results highlight the value of weight-based features in enhancing spam detection.

Furthermore, the underlying principles of the proposed Weighted Trust Rank (*WTR*) and AdaBoost-based framework extend beyond web spam detection and can be effectively applied to trust evaluation and fault detection in multi-robot systems. By modeling robots as interconnected nodes within a communication graph, the *WTR* mechanism can facilitate reliable trust propagation and enhance cooperative decision-making, thereby improving fault tolerance and system robustness in autonomous robotic networks.

Future work should focus on (i) testing scalability of WTR on larger and more diverse datasets, (ii) comparing against recent graph-based learning and neural ranking approaches. By addressing these aspects, *WTR* can serve as a more practical and robust tool for combating web spam in real-world search environments.

## Data Availability Statement

The relevant data that support the findings of this study are openly available in UK2006 Dataset

## Acknowledgement(s):

I would like to acknowlege and give my warmest thanks to my institution, VIT, Chennai, and my family for their support and understanding when undertaking my research. I thank my colleagues for fostering deep-rooted awareness in the study.

## Disclosure statement:

No conflict of interest. This article's content and writing are solely the author's.

## Funding:

No funding supported this study.

## References

[1] M. Danandeh Oskouei and S. N. Razavi, A survey of web spam detection techniques, Int. J. Comput. Appl. Technol. Res. **3**, 180–185 (2014).
[2] K. L. Goh, A. Singh, R. Kumar and A. Mohan, TPRank: Contend with web spam using trust propagation, Cybern. Syst. **45** (2014).
[3] K. L. Goh, R. K. Patchmuthu and A. K. Singh, Link-based web spam detection using weight properties, J. Intell. Inf. Syst. **43**, 129–145 (2014).
[4] K. L. Goh, R. K. Patchmuthu and A. K. Singh, Distrust seed set propagation algorithm to detect web spam, J. Intell. Inf. Syst. **49**, 213–235 (2017).
[5] Y. Guo, X. J. Yang and C. Shi, TIP: A trust inference and propagation model in multi-human multi-robot teams, Auton. Robots **48** (2024).
[6] J. Baladasan and K. Desikan, Weighted PageRank algorithm based on in-out weight of webpages, Indian J. Sci. Technol. **8** (2015).
[7] E. Khalastchi, Fault detection and diagnosis in multi-robot systems: A survey, Sensors **19**, 4019 (2019).

[8] X. Liu, Y. Wang, S. Zhu and H. Lin, Combating web spam through trust–distrust propagation with confidence, Pattern Recognit. Lett. **34**, 1462–1469 (2013).

[9] A. Makkar and N. Kumar, An efficient deep learning-based scheme for web spam detection in IoT environment, Future Gener. Comput. Syst. **108**, 467–487 (2020).

[10] V. Quang, H. Viet, V. Long and T. Khang, An Improved AdaBoost Algorithm for Highly Imbalanced Datasets in the Co-Authorship Recommendation Problem, *IEEE Access* **PP** (2023), pp. 1–1.

[11] A. K. Srivastava, R. Garg and P. K. Mishra, Discussion on damping factor value in PageRank computation, Int. J. Intell. Syst. Appl. **9**, 19–28 (2017).

[12] J. Whang, Y. Jung, I. S. Dhillon, S. Kang and J. Lee, Fast asynchronous Anti-TrustRank for web spam detection, (2018).

[13] B. Wu, V. Goel and B. D. Davison, Propagating trust and distrust to demote web spam, In: *Workshop on Models of Trust for the Web*, (2006).