Received August 9, 2021, accepted August 18, 2021, date of publication August 27, 2021, date of current version September 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3108427

Detecting Asset Cascading Failures Using Complex Network Analysis

JAYMIN MOFFATT[1], AYHAM ZAITOUNY[2,3], MELINDA R. HODKIEWICZ[1,3], (Member, IEEE), AND MICHAEL SMALL[2,3,4], (Senior Member, IEEE)
[1]School of Engineering, The University of Western Australia, Crawley, WA 6009, Australia
[2]Department of Mathematics and Statistics, The University of Western Australia, Crawley, WA 6009, Australia
[3]ARC Industrial Transformation Training Centre (Transforming Maintenance through Data Science), Curtin University, Bentley, WA 6102, Australia
[4]CSIRO, Kensington, WA 6151, Australia

Corresponding author: Ayham Zaitouny (ayham.zaitouny@uwa.edu.au)

This work was supported by the Australian Government through the Australian Research Council Centre for Transforming Maintenance through Data Science (Industrial Transformation Research Program) under Grant IC180100030.

ABSTRACT Experienced process plant personnel observe that corrective maintenance work on one asset may often be followed by corrective work on the same asset or connected assets within a short amount of time. This problem is referred to as a cascading failure. Confirming if these events are chronic is difficult given the number of assets and the volume of maintenance and operation data. If cascading events can be identified, preventative measures can be implemented to prevent those cascades, eliminating unnecessary corrective work. This project uses complex network analysis to identify cascading events and where co-occurrence of work is most frequent, in a process plant. Data is drawn from over 50,000 work orders for 5,655 pumps in a mining company over a five-year period. A complex network is produced by connecting assets based on the frequency of co-occurrence of work. Beside the advantages of the visualisation of complex networks, the method produces quantified measures, normalised degree, eigenvector centrality and betweenness centrality, which are used to identify assets with significant impact on other assets. Affected pumps are apparent as communities in the network. This analysis identifies pumps that are ''super-spreaders'': pumps who experience corrective maintenance events which lead to corrective maintenance events on other pumps. The model can be tuned to different time windows, for example events within one or seven days. From these insights, changes can be made to operational, maintenance and recording practices to prevent re-occurrence. Of particular note in this data was the occurrence of self–loops in certain pumps and the prevalence of hidden failures in standby pumps.

INDEX TERMS Complex networks, cascading failures, network science, eigenvector centrality, betweenness centrality, community detection, asset management, corrective maintenance.

I. INTRODUCTION

Maintenance is a collection of actions, intended to retain an item in, or restore it to, a state in which it can perform a required function [1], [2]. Maintenance work is critical to ensure assets within a company are repaired when a failure occurs, and to implement preventative measures to prevent failures from occurring. Maintenance personnel will inspect, modify, replace and repair different assets, or parts of assets, with the goal of returning the asset to a state in which it can perform its required function.

The associate editor coordinating the review of this manuscript and approving it for publication was Derek Abbott.

Maintenance work will normally fall under one of two categories: proactive and reactive maintenance, often referred to as preventative and corrective maintenance respectively [3]. Proactive maintenance includes tasks such as scheduled inspections or fixed interval replacement. This type of work is implemented to prevent future failures from occurring. Proactive maintenance is generally preferred as it normally results in a lower total cost, and minimal downtime on equipment [4]–[6]. Corrective maintenance is required when a failure occurs, or is imminent, and a restoring function is needed immediately [7].

Maintainers and operators often observe that a corrective work order on a specific asset is followed by the generation of a corrective work order on the same asset or another asset within a short amount of time. In situations where this cascading of failures exist, the failure of one asset can cause many failures amongst other assets. Confirming if these events are chronic is difficult given the number of assets and the volume of maintenance and operation data generated. If cascading events can be identified, preventative measures can be implemented to prevent the cascades, eliminating unnecessary corrective work.

Detecting and preventing cascading events is not a simple task. Besides the conventional approaches of engineering and risk management, the most current methods for preventing such cascading events rely on redesigning equipment to be less likely to cause other equipment to fail, such as resilience engineering that is based on designing resilient systems so that they can quickly recover their functions from failure or damage conditions [8], [9]. Other methods rely heavily on the engineering context of the system being analysed [10] and as such cannot be applied to different contexts without a significant change to the method. This is where complex network analysis is proposed to be applied to analyse a system which has cascading events present.

A network is a mathematical abstraction of a group of objects, in which some of the objects are related in some way. Typically, the objects are represented by nodes (also called a vertex or a point) in the network. The objects that exhibit the specified relationship will have an edge between their corresponding nodes within the network. The relationship between the nodes can be a unidirectional connection or bidirectional connection, these different types of relationships can be modelled in a network using directed or undirected edges [11]. A complex network is a graph that departs substantially from regular or statistically regular graphs [12]. A regular graph is a network where each node has the same number of edges, referred to as the degree of the node [13]. While a complex network is a graph that does not have a simple structure. Typically, complex networks have a large number of nodes and edges and do not contain repeating structure. A similar network approach has been utilised to model and analyse different systems, to name a few, modelling train routes [14], modeling the world wide web [15], modeling the mobile communication between people [16], analysing the transmission of a virus through a social network [17], modeling a social network for actors [18], modelling and predicting failure of a metropolitan water distribution network [19], and in neuroscience, functional networks are used to identify anomaly functional patterns after a brain stroke [20]. A comprehensive review of using complexity science and complex networks to model several complex systems such as crowd disasters, crime, terrorism, war, and the spread of disease can be found in [21].

A network can be represented visually by drawing the nodes and the edges connecting them, however, these networks are derived mathematically by an adjacency matrix. An adjacency matrix is a square matrix with each element in the matrix representing if an edge is present between two nodes in the network [11]. The elements of the adjacency matrix are defined as follows:

$$A_{ij} = \begin{cases} 1 & \text{if an edge exists from node } i \text{ to node } j, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The benefits of using an adjacency matrix are that it allows for storage of networks, and can be used to calculate network properties which infer hidden information about the system under consideration.

Complex networks of nodes and edges can be constructed using different methods. The simplest of these methods is when the structure of the network is predetermined. An example of this is a power network [22]. The power network has a known structure, where all the nodes and edges are predetermined by the layout of the power grid. The opposite of this situation is when a complex network needs to be discovered. In cases where network discovery is used, no existing network structure, or underlying network is known. For instance, network discovery was used to construct a complex network based on the number of passes between players in Australian Rules Football [23]. In this case the nodes represented the players and were known, whereas the edges of the network were discovered throughout a match when the ball was passed between two players. At the end of the match the whole network had been discovered and properties of the network were found and used to predict the outcomes of future games.

Typical complex network approaches for simulating or detecting cascading events include examining the existing network structure or testing how the network structure reacts to certain changes [22], [24], [25]. These approaches rely heavily on the engineering context of the system being analysed. These methods, when applied to a different system, even when similar, normally require a redesign to match the new context. A new network will need to be used, which can pose challenges if the network being analysed is not fully understood or if the elements of the network differ in abstraction. If the engineering context differs significantly, the method may not be applicable, due to factors such as how the assets interact. For example, the methods used to analyse cascading events on a power grid [22], cannot easily be adapted to analyse cascading failures on a road network. Road networks typically exhibit edge failures (roads being closed), whereas the power grid network analyses the impact on the network in the event of node failure (power substation failing). This lack of adaptability is undesirable as the underlying structure of the network is not always known and the method of failure propagation is not always obvious.

Cascading events have been modelled using complex networks. However, these networks all have a predefined structure. This predefined structure allows case specific methods to be utilised. In a system without a predefined relationship between nodes, a discovery method must be used to construct a complex network. This is where the literature is lacking. Cascading failures have not been analysed using a discovery method to construct a complex network. In this study, we are proposing a discovery complex network

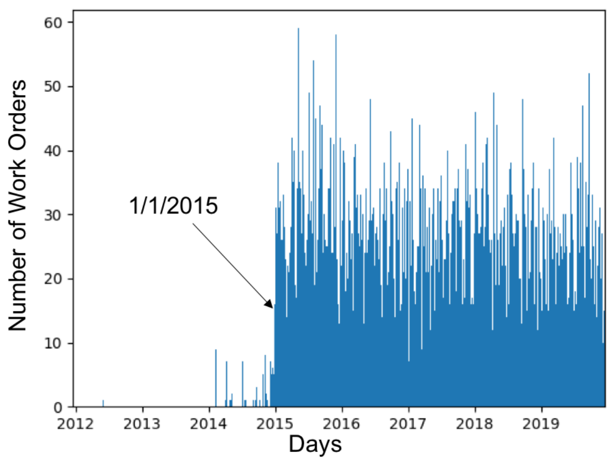


FIGURE 1. Distribution of the number of work orders per 24 hour period. Between the period of January 2015 and December 2019 seem normal but before this period only a few work orders exist. These work orders were removed as they do not include all the work done during that period.

approach that uses maintenance work order categorical data to construct a mathematical framework–complex network, which is analysed visually and quantitatively to infer hidden information and connections within the system, and detect cascading events in a process plant.

II. CASE STUDY DATA

The data in this project is provided by a mining company for work orders describing maintenance work on their pumps. As described in [3], [26], Maintenance work orders are records with a number of fields including creation date, work required, type of work (corrective, preventative etc.), desired start and end dates, costs and other information relevant to the planning and execution of maintenance work. All maintenance work should be associated with a work order. There are 88,545 work orders in this database for 5,655 different pumps. These work orders come from a period of eight years between December 2011 and December 2019. However, as can be seen in Figure 1 most of the work orders are between 1/1/2015 to 31/12/2019. Work orders not in this five year period are removed from the data as it appears to be incomplete data. This removed 1,175 work orders from the data leaving 86,370 to be used for analysis. Consequently, we restrict our analysis to data recorded between the period 1/1/2015 and 31/12/2019. We exclude data prior to 1/1/2015 because the frequency of recording is low and we infer that the company did not have the work order system in full operation. That is, we believe that we have complete (as far as possible) data for the period of analysis from 1/1/2015 to 31/12/2019. There are many results concerning the sampling of scale free networks. Our data excludes approximately 1% of the data provided to us (on the basis that it falls outside the census interval), this will not significantly affect our conclusions as they do not directly depend on the scale–free nature of the data.

TABLE 1. Summary of work order data acquired from a mining company for work on pumps. 5,655 pumps from three different sites are used in this investigation and only corrective maintenance work orders are considered for the failure cascading application.

Number of Work Orders	88,545
Number of Corrective Maintenance Work Orders	51,571
Number of Assets	5,655
Average Number of Work Orders Per Asset	15.27
Maximum Number of Work Orders Per Asset	130
Minimum Number of Work Orders Per Asset	1
Number of Days of Data	2,940
Number of Days of Data Used	1,826
Number of Assets with <5 Work Orders	2,949
Number of Assets with <10 Work Orders	4086
Number of Assets with <20 Work Orders	5016
Number of Assets with >50 Work Orders	87

On average each asset has 15.27 work orders in the five year period. Table 1 outlines a brief statistical description of the work order data at hand and the associated data which is used. It is notable that the work orders are log–normally distributed, that is, there is a high probability for any given asset in the system to have a low number of work orders and there is a low probability of an asset having a high number of work orders. Observe from the table how the range of the number of work orders per asset is between 1 and 130, while the number of the assets with more than 50 work orders is only 87 out of 5,655 assets, however, there are 2,949 assets with less than 5 work orders.

III. METHOD

This project uses complex network analysis techniques to analyse maintenance work orders with the aim to reveal cascading failure events within the system. For this project, Assets (in our application, pumps identified by their asset numbers) were chosen to be represented by the nodes, and an edge is placed between the nodes (pumps) if the corresponding assets are operationally (mechanically or physically) correlated and a work order was created for both assets, within a specified time window (co-occurrence of work). Additionally, edges are assigned weights based on the number of times co-occurrence of work is present in the data. For example, if the time window for connection is set to 24 hours and if asset A and asset B both had work orders created within 24 hours of each other on two different occasions, then the edge between the two nodes, representing these assets, have a weight of two. This is a discovered network as the nodes (pumps) are known and an edge is placed between the nodes when an event (co-occurrence of work) occurs.

This method uses discovery approach to build a network from data and as such can be applied to any context where failures occur and where there is the potential for a cascade of these failures to happen. The case study seen in this paper is for work order data for pumps, but the same method can be used to analyse other maintenance operations without redesigning the process.

The general model of constructing a complex network to infer hidden failure correlation between different assets and identify cascading events is defined as follows:

$$\text{Two assets are connected} \iff \text{“operationally correlated”} \\ \text{AND “temporally correlated”}$$

In our case for the pumping system and the provided information:

$$\text{“operationally correlated”} \equiv \text{“geographically co–situated”}$$

Mathematically this process can be represented as:

- Let N be the number of assets (nodes).
- Let M be the number of work orders.
- Let $\mathcal{A} = \{\alpha_i : i = 1, \ldots, N\}$ be the Asset Space.
- Let $\mathcal{T} = \{t_i : i = 1, \ldots, K \leq M\}$ be the Time Space of the work orders.
- $\mathrm{WO}(\alpha_i, t_m)$ denotes to the work order of asset α_i at time t_m.
- $L(\alpha_i)$ denotes to the geographical location (site) of asset α_i.

Accordingly, we define a Cascading Event (CE) between two work orders as follows:

$$\mathrm{CE}\Big(\mathrm{WO}(\alpha_i, t_m), \mathrm{WO}(\alpha_j, t_n)\Big) = \begin{cases} 1; & L(\alpha_i) = L(\alpha_j) \wedge 0 \leq (t_n - t_m) \leq \epsilon \\ 0; & \text{otherwise} \end{cases} \quad (2)$$

where ϵ is the time-window threshold.

Consequently, we define the weighted adjacency matrix as follows:

$$A_{ij} = \sum_{t_m \in \mathcal{T}_{\alpha_i}, t_n \in \mathcal{T}_{\alpha_j}} \mathrm{CE}\Big(\mathrm{WO}(\alpha_i, t_m), \mathrm{WO}(\alpha_j, t_n)\Big) \quad (3)$$

where $\mathcal{T}_{\alpha_i} \subset \mathcal{T}$ is the time subspace of work orders on asset α_i.

Different networks are created using this approach[1] by altering the criteria of connection to include a longer or shorter length of time, or by looking at only work orders from a specific point in time, for example, only looking at work orders from 2019. Furthermore, the networks can be altered by removing weak connections. By removing connections with a weight less than a defined threshold, the aspect of random chance can be minimised. Assuming there is no relationship between the occurrence of work between two assets, the connection approach discussed above still has a chance to produce a connection between the two assets if work co-occurs by chance. This random chance should have a low probability of occurring and so the connection will be comparatively weak. By removing the weaker connections in the network, this random chance can be minimised. For example, if a complex network is produced for the whole data set and then all edges with a weight less than one are removed, this results in over one million, or approximately 83% of the connections being removed. This has the added benefit of making the graph easier to visualise.

[1]The networks are constructed using the NetworkX package in Python [27]. The figures are obtained using Gephi [28] and its associated layouts [29], [30].

Only corrective work orders are included in this approach. This is because not all work that is recorded contributes to cascading failures amongst assets. Work such as inspections are unlikely to result in work on a different asset than the one inspected. Corrective work was identified as having the highest chance of causing cascading failures and as such, is the focus for this investigation. Saying that, it is needed to emphasise the fact that a corrective work order is generated when an event has occurred or is likely to occur in the future. For instance, if an asset's condition is degraded then a corrective work order would be generated while it may still be functioning. Therefore, the proposed model in Eq. 3 is identifying the connection between the creation of corrective maintenance work orders not between actual failure events. However, co-occurrence events are analysed between the work orders to infer potential actual cascading failures between assets. Options for including different types of work orders or revealing other questions are discussed further in Section VI.

IV. RESULTS

A. COMMUNITY DETECTION

A community in a network is a subset of nodes which are densely connected to each other whilst being sparsely connected to nodes outside the community [11]. For the context of this project a community in the network represents a group of assets that commonly have co-occurrence of corrective work within a specified time window of each other.

The communities of the networks are determined by using an iterative modularity–based algorithm [31]. Modularity of a community partition compares the density of edges inside the communities with the density of edges between the communities [11] and it is defined as follows:

$$Q = \frac{1}{2m} \sum_{ij} \Big(A_{ij} - \frac{k_i k_j}{2m}\Big) \delta(C_i, C_j), \quad (4)$$

where A_{ij} is the edge weight between assets α_i and α_j as expressed in the network's adjacency matrix, $k_i = \sum_j A_{ij}$ is the degree of node α_i (the sum of the weights of the links connected to asset α_i), m is the total number of the weights of the edges in the network, C_i denotes the community to which the asset α_i is assigned to, finally, the function $\delta(C_i, C_j) = 1$ if nodes α_i and α_j are within the same community and 0 otherwise. The algorithm starts by making a community for each node. For each node in the network, the algorithm checks if merging that node into a community of a node it is connected to, increases the modularity of the network. Each node is merged with a community of a neighboring node until no increase in modularity occurs. At this point a new network is constructed. The communities in the network are treated as individual nodes with weighted links between these nodes based on the number

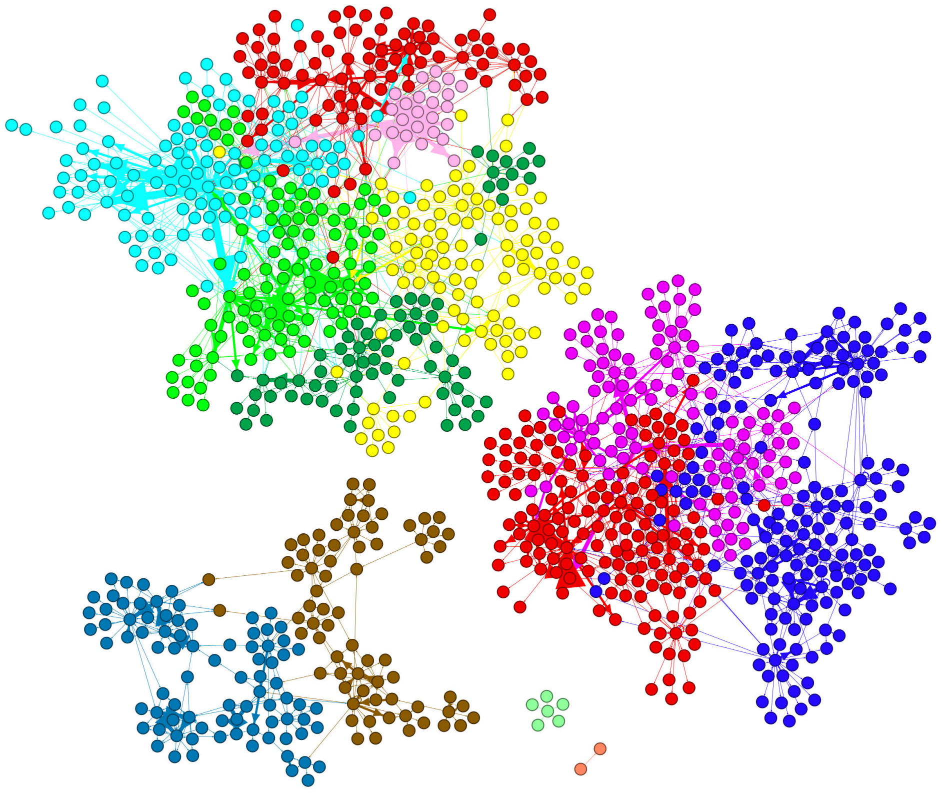


FIGURE 2. Community structure of the network produced by using data from the first three months of 2019 and using a 24 hour connection criteria. Nodes refer to assets (pumps) and edges indicate that the two assets are geographically co–located and have failure correlation, edges with a weight less than two have been removed. Colours indicate different communities, the network shows 13 communities within the network. If a failure occurs to an asset from a community then it would potentially cascade this failure to a neighbour asset within the same community.

of connections between nodes in the two corresponding communities. Self–loops existed with a weight showing the total sum of the nodes inside the community. The process is then repeated for this new network, merging nodes until no increase in modularity occurs. These two steps of merging communities and then creating a new simplified network, are repeated until no increase of modularity occurs. At this point the communities present are the final communities of the network.

Figure 2 shows a subset of the work order data taken from the first three months of 2019, from 1/1/2019 to the 30/3/2019. The time window for connection was set to 24 hours. Edges of weight one are removed to account for random low weight edges. The different communities within the network have been individually coloured. In total, there were 13 communities of assets detected with an average of 78 assets in each community. An example of a single community from Figure 2 can be seen in Figure 3. It can be seen in Figure 2 how the community detection method has grouped the assets into three large clusters related to the sites (due to the geographical location constraint from Eq. 3), and within these clusters it assigned the nodes into smaller communities.

If the time window for connection is increased, the network constructed will contain many more connections. Due to this reason, the community detection method produces fewer communities of assets. Figure 4 shows the communities present for the same network seen in Figure 2 where the altered connection criteria of seven days is used. The community detection algorithm has detected only 3 communities (related to the different sites due to the geographical location constraint) within this network, as opposed to the 13 seen in Figure 2. To reveal the inner structure within these large communities, one can investigate the sub–communities by analysing the data of each site separately. Alternatively, the number of communities can be altered by adjusting the resolution of the community detection algorithm used [31], however, in this project a constant resolution of 1.0 was used.

Using a larger connection window increases the probability that two work orders are in the same time window, hence the number of connections each node has is increased. This results in more connections between assets that exhibit corrective work unrelated to each other. This limits the capability of the community detection to adequately group the nodes, as unrelated nodes can be connected due to random chance. This is still a potential outcome in the original case

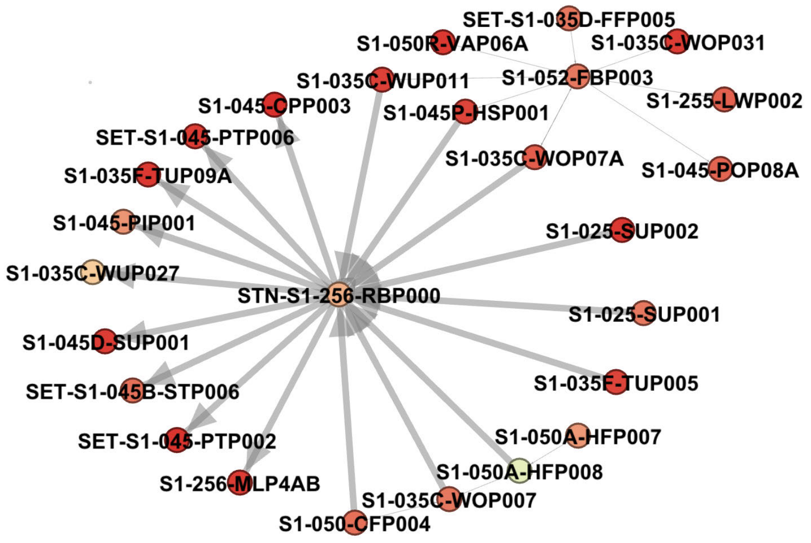


FIGURE 3. Single community from Figure 2 containing 26 nodes and 31 edges. Contains some nodes (assets) with a high number of connections, such assets play an essential role in cascading the failure to other assets within the community.

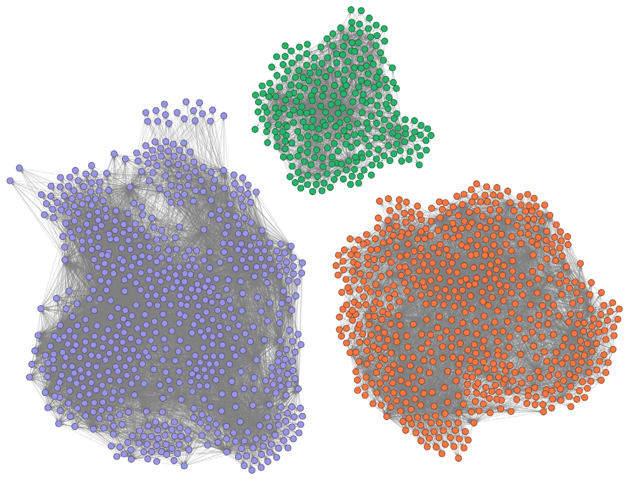


FIGURE 4. Community detection on the network made using a connection window of seven days from work orders during the first three months of 2019. Nodes are assets (pumps) and edges indicate geographical and failure correlation between assets. Edges of weight two or less have been hidden for clarity. Colours indicate different communities, this graph is much more interconnected and exhibits only 3 communities that are related to the site geographical location of the assets.

examined in Figure 2, however these random connections occur less often when a shorter time window is used, resulting in more distinct communities. Alternatively, when a longer connection window is used, edges with a relatively low weight can be removed before community detection is done to reduce the number of unrelated, random connections between assets.

B. NORMALISED DEGREE RANKING

A common way to rank a node's importance in a complex network is by ranking them by degree. The degree of a vertex in a graph is the number of edges connected to it. In a weighted graph the degree of a node is the sum of the weights of all the edges connected to that node [11]. In a directed network the degree can be separated into two categories, in–degree and out–degree, which is the sum of the weight of edges entering and leaving the node respectively. More formally this is defined as:

$$D^{\text{out}}(\alpha_i) = \sum_j A_{ji} \tag{5}$$

$$D^{\text{in}}(\alpha_i) = \sum_j A_{ij} \tag{6}$$

In our application, the out–degree reflects how an asset impacts on neighbour assets when it fails, while the in–degree indicates how an asset could be affected by failures of neighbour assets. If degree ranking is applied to the context of asset maintenance, the measure will rank nodes based on the sum of the number of work orders created within a time window, when a work order is created for that asset. This identifies the hubs of the network. This approach gives a bias to assets that frequently have corrective work done, as the total sum is generally large, simply due to random connections. To overcome this a normalised approach can be used. This is done using the following equations:

$$D_n^{\text{out}}(\alpha_i) = \frac{\sum_j A_{ji}}{N(\alpha_i)} \tag{7}$$

$$D_n^{\text{in}}(\alpha_i) = \frac{\sum_j A_{ij}}{N(\alpha_i)} \tag{8}$$

where $D_n^{\text{out}}(\alpha_i)$, $D_n^{\text{in}}(\alpha_i)$ are the normalised out–degree and in–degree of asset α_i respectively, and $N(\alpha_i)$ is the number of times that asset α_i occurs in the work order data.

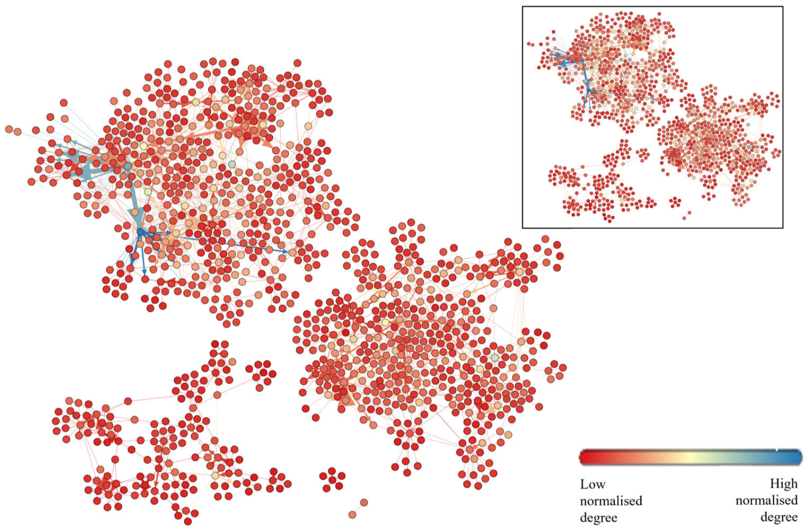


FIGURE 5. In–normalised–Degree (full size network) and out–normalised–degree (small size figure in the right top corner) of the network produced by using data from the first three months of 2019 and using a 24 hour connection criteria. Nodes are assets (pumps) and edges indicate geographical and failure correlation between assets, edges with a weight less than two have been removed. Nodes ranked highly by normalised degree appear to be grouped in communities.

TABLE 2. Summary of network centrality measures showing the top 5 most important nodes in each network described. The assets are ordered from most important to least important in descending order.

	3 month of 2019 - 24h connection window	All data - 24h connection window	All data - 7 day connection window
In Degree Ranking	S1-045P-FFP001	S2-025-MPP031	S2-025-MPP031
	S1-257-SRP02B	STN-S1-256-RBP000	S2-025-MPP022
	S1-035D-FFP009	S2-025-MPP022	S2-025-MPP021
	S2-050-LFP011	S2-025-MPP012	STN-S1-256-RBP000
	S1-257-SRP01A	S2-025-MPP011	S2-025-MPP012
Out Degree Ranking	S1-045P-FFP001	S2-025-MPP031	S2-025-MPP031
	S1-257-SRP02B	S2-025-MPP012	S2-025-MPP021
	S2-050-LFP011	STN-S1-256-RBP000	S2-025-MPP022
	S1-035D-FFP009	S2-025-MPP032	STN-S1-256-RBP000
	S1-035D-SUP003	S1-257-SBP001	S1-257-SBP001
Eigenvector Centrality	S1-045P-FFP001	S2-025-MPP031	S2-025-MPP031
	S1-257-SRP02B	S2-025-MPP022	S2-025-MPP022
	S1-035D-FFP009	S2-025-MPP071	S2-025-MPP021
	S1-257-SRP01A	S2-025-MPP021	S2-025-MPP012
	S1-256-MLP06A	S2-025-MPP072	S2-025-MPP032
Betweenness Centrality	S1-045P-FFP001	S2-025-MPP031	S2-025-MPP031
	S1-257-SRP02B	STN-S1-256-RBP000	STN-S1-256-RBP000
	S2-050-LFP011	S2-025-MPP012	S2-025-MPP021
	S1-035D-FFP009	S2-025-MPP022	S1-257-SBP001
	S-S3-110	S1-257-SBP001	S2-025-MPP022

Using these formulas results in normalised degrees for each node, ranking assets based on the average number of other assets that have work at the same time as the asset in question. The highest ranked assets are the normalised hubs of the network. These hubs are the potential assets in which a failure typically results in numerous other failures at the same time (in the case of out–degree ranking), or they are the potential assets which have higher probability to fail when failures occur on other assets (in the case of in–degree ranking). The identification and ranking of these hubs are summarised in Table 2 for different network settings.

The results of getting these normalised degrees for the work order data can be seen in Figure 5. This network is identical to the one seen in Figure 2 with the colour representing the normalised degrees ranking as opposed to the community structure. In comparing Figure 2 and 5 an interesting relationship can be seen. Nodes that have a high normalised in–degree (or out–degree) in Figure 5, tend to be grouped together in the communities of Figure 2. This is beneficial as the community detection method indicates which assets potentially fail together but does not indicate in which assets this co-occurrence of failure is most common. By combining the two methods, insight is found as to which

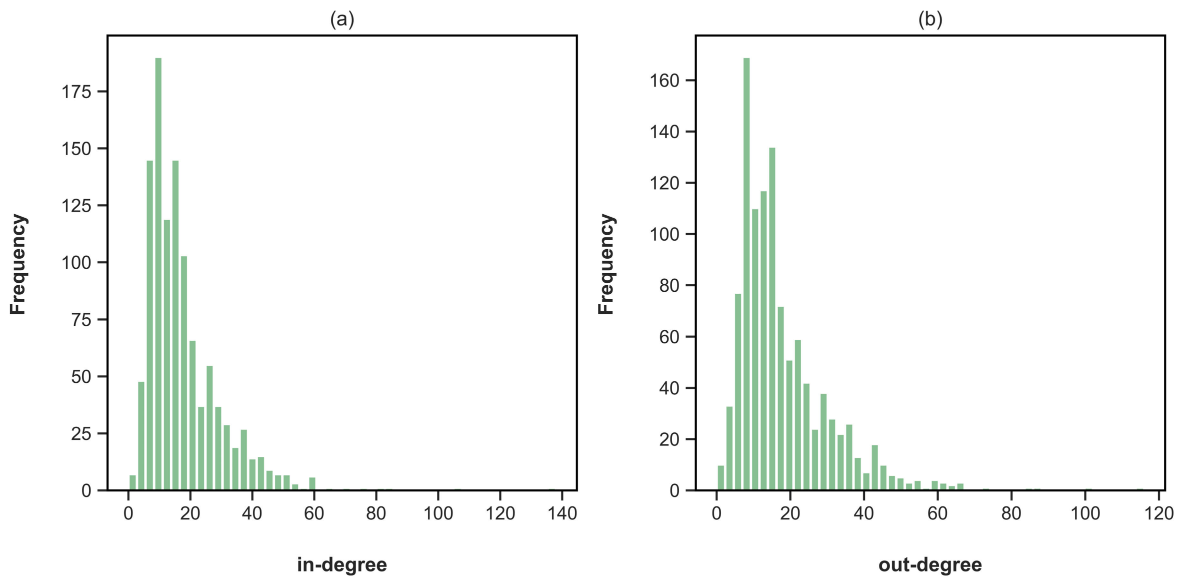


FIGURE 6. The distributions of the normalised degrees ranking of nodes within the network produced using three months of data from 2019. Both types of degree have a log–normal distribution with mean ≈18.

nodes potentially fail together and which nodes within each group are correlated with many other events. This identifies the assets that should be looked at first when attempting to correct the cascading of failures within this system.

Figure 6 shows the distribution of the normalised degrees ((a) for in–degree and (b) for out–degree) amongst the nodes in the respective network. These appear to follow a log–normal distribution with a mean of approximately 18. This means on average when there is corrective work on one asset then there is 18 different corrective work orders made within 24 hours. Obviously most of these work orders will be unrelated to the initial work, only in cases where repeated co-occurrence of work is frequent we can say there is likely cascading work occurring. This method of analysing the strongest connections in the network is discussed further in Section IV-C.

C. STRONG CONNECTIONS

In this section, we analyse the constructed network visually by looking at sub–networks with strong connections. This is done by removing edges below a threshold weight and then removing nodes that are isolated (nodes with no connections), hence, a sub–network can be formed that shows the assets in the data which have the highest frequency of co-occurrence of work.

Examples of these strong connection sub–networks can be seen in Figures 7, 8, and 9. These three Figures show different scales of the network with the only difference being the minimum weight of the edges shown. In Figure 7 several chains of connections exist that indicate long cascade events (among several assets) and some assets behave as super–spreaders as they start different cascades (e.g. see assets S2–025–MPP031, S2–025–MPP021), whereas, in Figures 8 and 9 these chains disappear and only dual connections or self–loops exist. This shows that it is important to check many different minimum weights and investigate different scales, when looking at the strong connections of the network. The failure of the super-spreader (hub) asset may lead to cascading events amongst many other assets, but if a minimum weight that is too large is used, the asset will seem relatively unimportant. An alternative and somewhat related approach to identify super-spreader events, that is similar to the strong connection analysis, is the k–shell decomposition approach [32]. However, in our application the weight of the connection also plays a significant role in identifying the cascading event rather than the degree of the node alone.

One interesting observation is the presence of chains of connected assets in the sub–network with minimum weight 10 (Figure 7). These chains indicate that when corrective work is done on one asset in the chain, the next asset in the chain is likely to have a corrective work order created within the set time window. This cascades on until the end of the chain. By identifying these chains, preventative measures can be implemented to interrupt this process and reduce the amount of future work.

An argument can be made that this method does not necessarily detect where cascading failures are occurring, but just the assets that tend to fail at the same time. The presence of a connection does not imply that the failure in the preceding asset causes the failure of the proceeding one. It is possible that an external aspect causes the failure in both pumps. However, if assets are affected in this way, a connection would be made that has an equal strength in both directions and not be weighted heavily in one direction. Additionally, if an external force is causing a failure on three or more assets, the graph would be expected to contain cliques

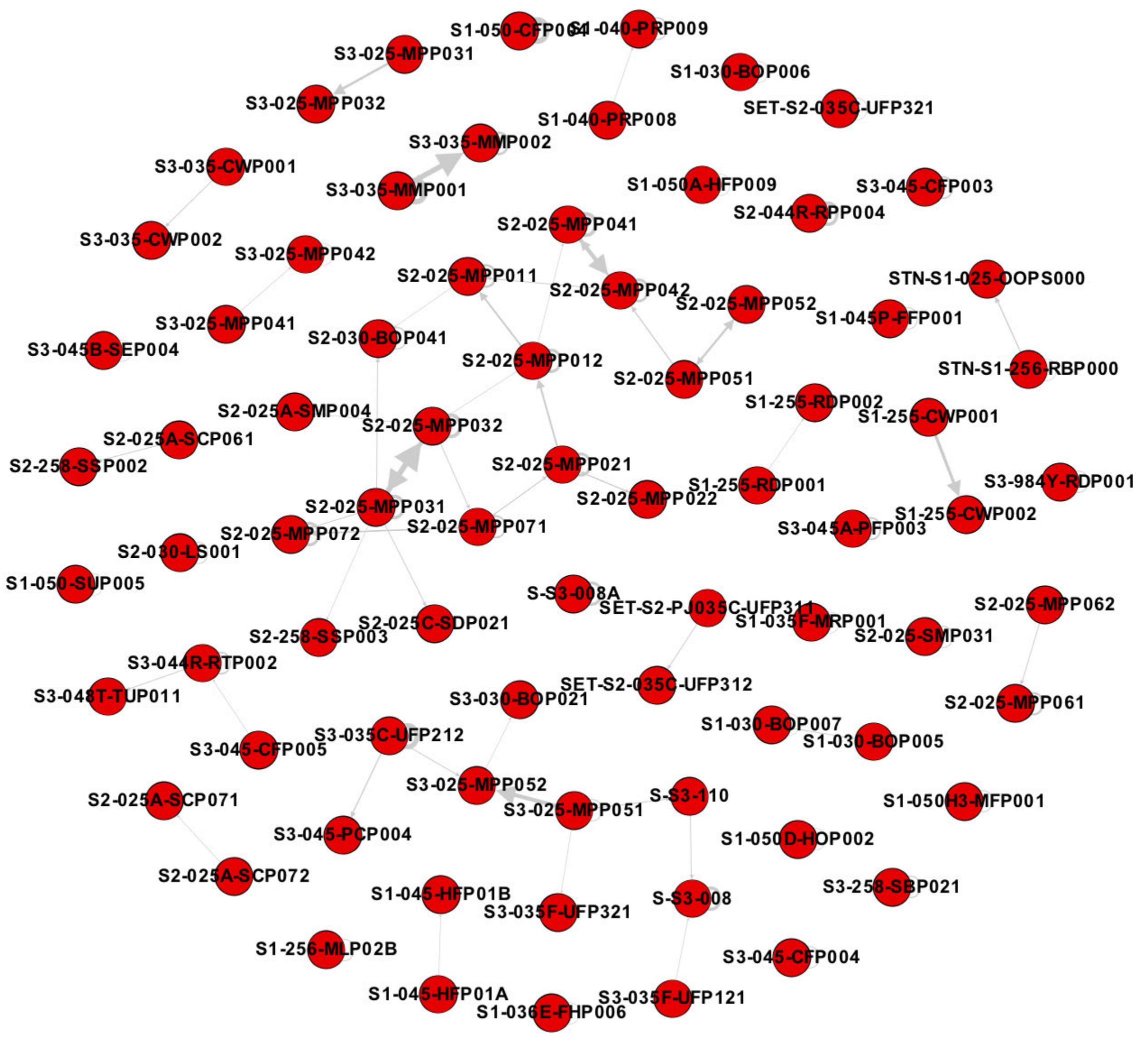


FIGURE 7. Strong connections, all work orders are considered with a 24–hour Connection window. Nodes are assets (pumps) and edges indicate geographical and failure correlation between assets. Thickness of the edges indicate the edge's weight, minimum weight considered is 10. Some long cascade events are identified as chains of connections and some potential super–spreader assets are detected.

of third order or higher.[2] Typically, when an external aspect is causing the failures, the assets fail at the same time, creating the clique structure. There are no such cliques of third–order or higher present in these networks (Figures 7, 8, 9). The absence of cliques in the network supports the cascading structure suggested in this paper, as the failures appear to occur in stages. Note should be taken that the presences of high order cliques does not necessarily imply that a cascade is not occurring. Fast moving cascades also create clique structures, as many assets fail within a short period of time.

The strong connection results seen in this Section were presented to the mining company that supplied the data, to see if the identified connections matched up with expectations. The company identified that some of the connections present were already known and corrective action has been taken. Furthermore, the strong connection present in Figure 9 between pump S3-035-MMP001 and S3-035-MMP002 was of particular interest. The MMP002 pump is the backup pump for MMP001. This means that when pump MMP001 fails, the backup pump is put into use. Soon or immediately after pump MMP002 is used, the pump exhibits a failure as well. This is known as a hidden failure. The pump has been unused, and whilst it is not in use a failure has occurred. This failure remained undetected, until pump MMP001 failed at which point its failure was made apparent. Without using knowledge that these pumps were related, using only the work order data provided, complex network analysis identified a strong connection between the work done on these two pumps.

A further interesting observation is that a large proportion of the edges correspond to self–loops (see Figure 8).

[2] A clique in a network is a subset of nodes, in which all the nodes in the subset are connected to all the other nodes in the subset. A clique of size two is simply a line, which is frequent in the strong connection networks. A clique of size three is a triangle in the network.

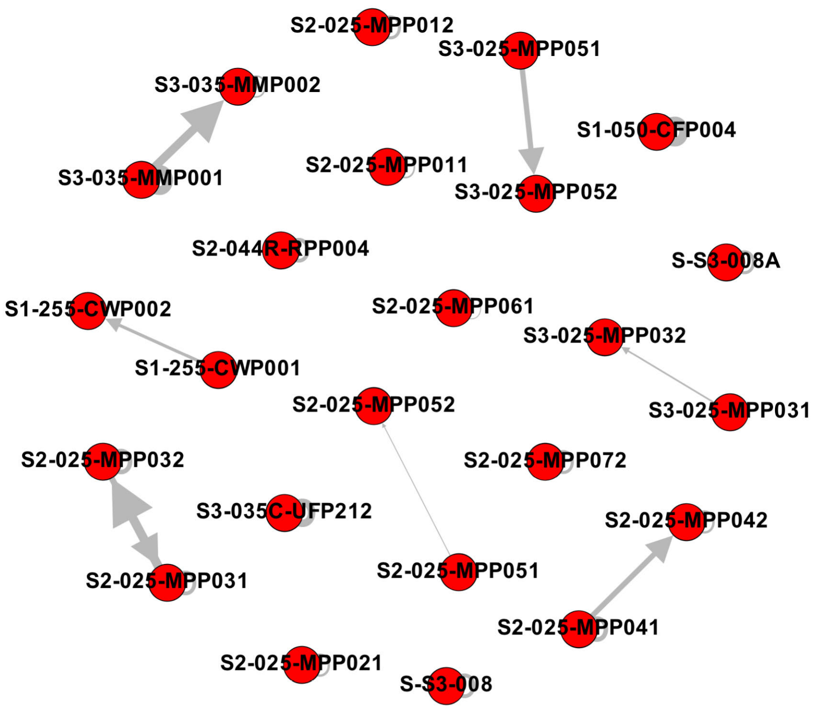


FIGURE 8. Strong Connections, similar to Fig. 7, all work orders, 24–hour Connection window, but minimum weight is 15. Only strong dual connections appear here that indicate potential failure cascade events from one asset to another.

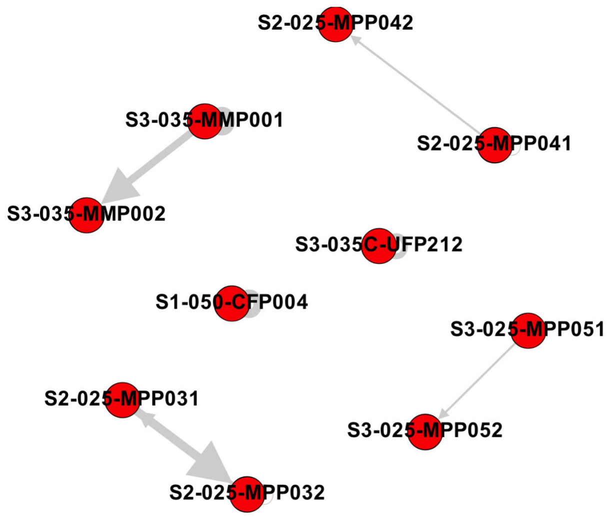


FIGURE 9. Strong connections, all work orders, 24–hour Connection window, Minimum Weight is 20. A larger minimum weight scale than in Figures 7 and 8 to emphasise the stronger connections that indicate higher potentiality of cascading events.

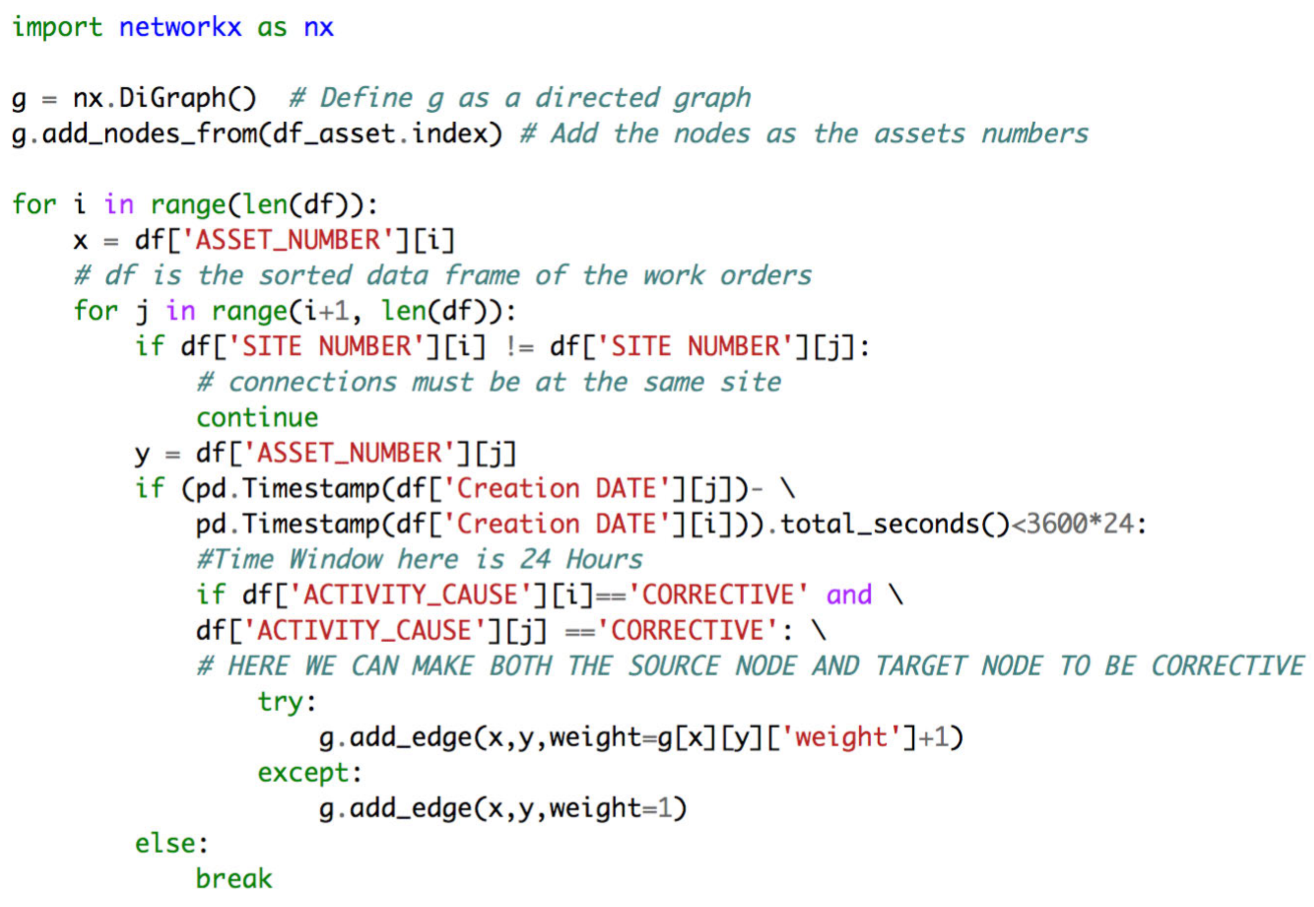
This means that these assets have co–occurrence of work with themselves. An example of a self–loop is the node S1-050-CFP004 in Figures 8 and 9. There are three likely possibilities as to why these self–loops occur. The first is that the asset has many parts that need repairing by different teams of people, hence multiple work orders were created at the same time for the one asset. This causes the network created to contain self–loops. The second possibility is that duplicate work orders are being created. This is not a desirable thing from a management point of view as it makes it hard to view accurate statistics from maintenance data. The third possibility is that the work that is completed on these assets does not solve the problem, and as such a secondary corrective work order is created after the initial one is completed. If this is occurring then a quality issue in the performance of the maintenance is likely present. This aspect is furtherly discussed in Section V.

D. CENTRALITY MEASURES

There are many other measures in network analysis, to rank the nodes by their importance. These ranking methods, often referred to as centrality measures, do not always have a clear relationship with physical properties. These measures provide useful insights into the important nodes of the network. Although there are several centrality measures, in our application to investigate failure cascading events in a complex network, we emphasise static network properties and use eigenvector and betweenness centrality measures.

1) EIGENVECTOR CENTRALITY

Eigenvector centrality ranks the nodes based on the number of important nodes they are connected to [11]. This is done by computing the eigenvector that corresponds to the largest eigenvalue of the adjacency matrix of the network. The eigenvector for this largest eigenvalue is then computed. Each element of this eigenvector corresponds to a node in the network, based on how the adjacency matrix was defined.



```
import networkx as nx

g = nx.DiGraph()  # Define g as a directed graph
g.add_nodes_from(df_asset.index) # Add the nodes as the assets numbers

for i in range(len(df)):
    x = df['ASSET_NUMBER'][i]
    # df is the sorted data frame of the work orders
    for j in range(i+1, len(df)):
        if df['SITE NUMBER'][i] != df['SITE NUMBER'][j]:
            # connections must be at the same site
            continue
        y = df['ASSET_NUMBER'][j]
        if (pd.Timestamp(df['Creation DATE'][j])- \
            pd.Timestamp(df['Creation DATE'][i])).total_seconds()<3600*24:
            #Time Window here is 24 Hours
            if df['ACTIVITY_CAUSE'][i]=='CORRECTIVE' and \
            df['ACTIVITY_CAUSE'][j] =='CORRECTIVE': \
            # HERE WE CAN MAKE BOTH THE SOURCE NODE AND TARGET NODE TO BE CORRECTIVE
                try:
                    g.add_edge(x,y,weight=g[x][y]['weight']+1)
                except:
                    g.add_edge(x,y,weight=1)
        else:
            break
```

FIGURE 10. Python code for constructing the network from the work orders. Representing steps 4, 5, 6, 7, and 8.

These elements of the specific eigenvector are the values for the eigenvector centrality of each node. Mathematically, the eigenvector centrality is expressed by:

$$x(\alpha_i) = \frac{1}{\lambda} \sum_k A_{ki} x(\alpha_k) \tag{9}$$

where $x(\alpha_i)$ is the eigenvector centrality of asset α_i and λ is the largest eigenvalue of the adjacency matrix A. This can be rearranged and expressed in matrix form as the eigenvector equation:

$$A\mathbf{x} = \lambda\mathbf{x} \tag{10}$$

where $\mathbf{x} = (x(\alpha_1), x(\alpha_2), \ldots, x(\alpha_N))^T$.

Hence, highly ranked nodes are connected to other highly ranked nodes within this network. Nodes ranked as important may exhibit cascading events to other nodes that have a high chance of continuing the cascade. A list of the most important assets in the data based on the eigenvector centrality can be seen in Table 2

2) BETWEENNESS CENTRALITY

Betweenness centrality ranks nodes as important based on the sum of the number of times that node is on the shortest path between any two other nodes in the network [33]. To do this, the minimum distance between any pair of nodes in the network must be calculated. Then, for each node in the network, the number of times this node appears on the shortest path between two other nodes must be counted. The algorithm detailed in [34] is used to do this. The algorithm completes this task with a minimised complexity, reducing the overall time of the calculation. Note that the unweighted betweenness centrality is being measured as the weighted version considers the weights as delays, whereas weights in these networks are representative of connection strength.

Betweenness centrality is given by:

$$g(\alpha_i) = \sum_{\{j,k\}\neq i} = \frac{\sigma_{jk}(\alpha_i)}{\sigma_{jk}} \tag{11}$$

where $g(\alpha_i)$ is the betweenness centrality of node α_i, σ_{jk} is the number of shortest paths between nodes α_j and α_k, and $\sigma_{jk}(\alpha_i)$ is the number of shortest paths between nodes α_j and α_k that contain node α_i.

In the network described in this paper, a node with a high betweenness centrality would likely be in the middle of a cascading event chain. By preventing the cascade from reaching this node, it is more likely that the chain is prevented from continuing. This may not always be the most desirable action to take however, as betweenness centrality does not give an indication as to how far through the cascading chain the failures are. A high betweenness node could be the second last asset in a long cascading chain of failures.

It is often seen that there are multiple parallel failures occurring, each of which has the potential to continue the cascading events to other assets. The benefit of using betweenness centrality to target preventative work is that the nodes with high betweenness centrality have the least parallel failures, thus, by interrupting the cascade on these assets, a large number of failures can be prevented with minimal work.

In Table 2, a list of the most 5 important assets in the data based on the betweenness centrality are outlined for different time windows (24 hours or 7 days) using 3 months as well as the entire data set.

V. DISCUSSION

The proposed complex network discovery–approach has been implemented to infer hidden relations among assets by using maintenance work order data. The constructed complex networks are interpreted in two aspects: 1) As a visualisation tool that demonstrated results on communities within the system as well as strong connections for variant scales. 2) Quantitative analysis that allowed ranking of the assets based on their importance in different perspectives. The results have been checked by the company providing the data, as they confirmed that they are aware of some of the detected correlations, while they were not attentive to some other inferences which were reasonable for them and worth further investigations. It can be observed how the results summarised in Table 2 are relatively consistent between the different ranking metrics. For example, in the 3 month–24h connection window network, the asset S1–045P–FFP001 was detected as the most important asset by all of the four measures. Similarly, asset S2–025–MMP031 was the most important by all of the measures for both all data networks of 24h and 7 day connection windows. For each network, most of the other assets were detected by all of the measures but with different ranking levels, and only a few assets were detected by some ranking measures but not the others; for instance, see asset S–S3–110 for the 3 month network and asset S2–025–MPP72 for the all data network. These consistency and variation are expected as the metrics rank the nodes based on different importance perspectives. On the other hand, the results show significant variation when only 3–month data is used in comparison to using all of the data, moreover, less significant variation is observed when using different connection window thresholds even though all of the data has been used (see columns 2 and 3 in Table 2). We suggest this is due to the amount of information included or lost, and the level of randomness involved according to the size of the data. For instance, 3 months of the data includes less information especially as operational strategy can be adjusted and changed during time. Although considering a longer connection window (i.e. 7 days) adds more information, it increases the randomness level as more unrelated connections can appear, as there is always a chance that some connections are created between unrelated assets due to random chance. If there is no true connection between two assets, the network still has a chance to connect them. This connection typically has a low weight, however, in some circumstances, random chance can cause this weight to be above the threshold weight chosen. If this occurs for two assets, the recommendations made to investigate this may not produce any meaningful results and may cost the company money. These high weighted, false connections should be rare but care must be taken, as this method does not guarantee a relationship between corrective work events when a high weighted connection exists in the network. Such limitations do not undervalue the performance and the significant results of the proposed method, they have been discussed to open the doors for future investigations.

As Eq. 3 states, the model does not strictly utilise failures to connect between assets, instead the frequency of co-occurrence of work is used to produce a complex network. High weighted chains of connected assets does not guarantee cascading work is present, only that co-occurrence of work between two connected pairs is frequent, however, such identification could be of high value to the company as they can investigate and confirm cascading event or not. It is entirely possible in a high weighted chain of three assets, the first and the third asset will have a high frequency of co-occurrence of work with asset two, but work never occurs on asset one and three at the same time in the data. Although there appears to be a connection between assets one and three via their connections to asset two, there are no events that cascade between the assets.

Another interesting observation is the present of self–loops in the network as there are many self–loops exhibit in all of the graphs produced. They are particularly obvious in Figures 7, 8 and 9. These correspond to where multiple corrective work orders are created for the same asset within the time window set for that network. This could be another area for future research as further analysis should be done to identify the nature of these self–loops. Questions should be asked such as:

- Are these self–loops an inspection followed by a repair?
- Are these self–loops a repair followed by a repair?
- Are these self–loops repeat work orders?

VI. CONCLUSION

In a case study, complex network analysis identified assets in which cascading work was frequent. This is done by discovering a complex network from maintenance work orders, with connections representing the frequency of co-occurrence of work. By performing different network analysis techniques in both visual and quantitative perspectives, insights into the nature of the assets are obtained. Communities of assets that commonly had co-occurrence of work together are identified.

Assets (experiencing corrective maintenance events) with a large number of connections with other assets (also experiencing corrective maintenance events) were found using the normalised degrees (in–degree and out–degree) of each asset. These two measures are compared to give insight into which communities had the most co-occurrence of work. The network is then used to show where cascading corrective work is most frequently occurring, by looking at the highest weighted edges in the network. This method highlights assets that commonly exhibited cascading events, hidden failures, and repeated corrective work. Finally different centrality measures of the network are used to rank the most important assets in the network. This method is not limited to the context analysed and can instead be applied to many different contexts. For example, corrective maintenance is not the only type of maintenance to exhibit cascading work. A similar approach to the one done in this project should be performed using corrective and preventative maintenance. Particularly looking at where preventative work is followed by corrective work. This would indicate that the preventative work was unsuccessful in preventing the failure of the asset or even caused the failure of the asset.

A continuation of this project is to consider a variable connection widow to overcome the harsh threshold limitation, and to do a temporal analysis of the data and see how it evolves over time. The mining company confirmed that some of the connections identified had already been found and corrective action was taken, however by looking at a long period of time, these connections were still present. Analysing how the network changes with time will allow the reliability engineers to see if their actions are reducing the occurrence of cascading events.

APPENDIX

PROCESS WORKFLOW

The procedure is as follows:

1) Import the work order data to Python.
2) Remove work orders with missing data in the fields of interest for the network.
3) Sort the remaining work orders in time by their creation date.
4) Create an empty NetworkX graph.
5) Add a node to the network for each asset and label the node with the asset number.
6) Select the first work order in the list and find all other work orders that were created within a set time interval of the original work order.
7) Create edges in the NetworkX graph to connect all of the nodes corresponding to the assets in the identified work orders to the node representing the asset in the original work order. If an edge already exists between two assets, increase the edges weight by one.
8) Repeat steps 7 and 8 for all work orders to identify all connections.
9) Use NetworkX functions to acquire the relevant network measures needed to produce different graphs, such as community structure and degree centrality.
10) Export the network as a GEXF file for Gephi to produce a graph.
11) Open the file and select a layout algorithm to plot the network.

The Python code for constructing the network from the associated work orders data is presented in Figure 10.

REFERENCES

[1] *Dependability Management Application Guide–Reliability Centred Maintenance*, Standard IEC 60300.3.11:2011, International Electrotechnical Commission, 2011.

[2] K. A. H. Kobbacy and D. P. Murthy, *Complex System Maintenance Handbook*. London, U.K.: Springer, 2008.

[3] M. P. Brundage, T. Sexton, M. Hodkiewicz, K. Morris, J. Arinez, F. Ameri, J. Ni, and G. Xiao, "Where do we start? guidance for technology implementation in maintenance management for manufacturing," *J. Manuf. Sci. Eng.*, vol. 141, no. 9, Sep. 2019, Art. no. 091005.

[4] K. Feldman, P. Sandborn, and T. Jazouli, "The analysis of return on investment for PHM applied to electronic systems," in *Proc. Int. Conf. Prognostics Health Manage.*, Oct. 2008, pp. 1–9, doi: 10.1109/PHM.2008.4711415.

[5] C. Drummond and C. Yang, "Reverse engineering costs: How much will a prognostic algorithm save," in *Proc. Int. Conf. Prognostics Health Manage.*, Denver, CO, USA, 2008, pp. 1–4.

[6] C. Yang and S. Letourneau, "Model evaluation for prognostics: Estimating cost saving for the end users," in *Proc. 6th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2007, pp. 304–309.

[7] M. Hodkiewicz, S. Lukens, M. P. Brundage, and T. Sexton, "Rethinking maintenance terminology for an industry 4.0 future," *Int. J. Prognostics Health Manage.*, vol. 12, no. 1, pp. 1–14, Mar. 2021.

[8] K. Furuta, "Resilience engineering," in *Reflections on the Fukushima Daiichi Nuclear Accident: Toward Social-Scientific Literacy and Engineering Resilience*. Springer, 2015. pp. 435–454.

[9] Y.-P. Fang, N. Pedroni, and E. Zio, "Resilience-based component importance measures for critical infrastructure network systems," *IEEE Trans. Rel.*, vol. 65, no. 2, pp. 502–512, Jun. 2016.

[10] E. Zio and G. Sansavini, "Component criticality in failure cascade processes of network systems," *Risk Anal.*, vol. 31, no. 8, pp. 1196–1210, 2011.

[11] M. Newman, *Networks*. London, U.K. Oxford Univ. Press, 2018.

[12] L. da F. Costa, "What is a complex network (CDT-2)," Tech. Rep., Apr. 2018, doi: 10.13140/RG.2.2.10450.04804/2.

[13] R. J. Trudeau, *Introduction to Graph Theory*. North Chelmsford, MA, USA: Courier Corporation, 2013.

[14] X. Wang, Y. Koç, S. Derrible, S. N. Ahmad, W. J. A. Pino, and R. E. Kooij, "Multi-criteria robustness analysis of metro networks," *Phys. A, Statist. Mech. Appl.*, vol. 474, pp. 19–31, May 2017.

[15] A. N. Langville and C. D. Meyer, *Google's PageRank Beyond: The Science Search Engine Rankings*. Princeton, NJ, USA: Princeton Univ. Press, 2011.

[16] R. Lambiotte, V. D. Blondel, C. de Kerchove, E. Huens, C. Prieur, Z. Smoreda, and P. V. Dooren, "Geographical dispersal of mobile communication networks," *Phys. A, Stat. Mech. Appl.*, vol. 387, no. 21, pp. 5317–5325, 2008.

[17] H. Zhang, J. Zhang, C. Zhou, M. Small, and B. Wang, "Hub nodes inhibit the outbreak of epidemic under voluntary vaccination," *New J. Phys.*, vol. 12, no. 2, Feb. 2010, Art. no. 023015.

[18] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[19] A. Ballantyne, N. Lawrance, M. Small, M. Hodkiewicz, and D. Burton, "Fault prediction and modelling in transport networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–5, doi: 10.1109/ISCAS.2018.8351658.

[20] Y. Zou, Z. Zhao, D. Yin, M. Fan, M. Small, Z. Liu, C. C. Hilgetag, and J. Kurths, "Brain anomaly networks uncover heterogeneous functional reorganization patterns after stroke," *NeuroImageme Clin.*, vol. 20, pp. 523–530, Jan. 2018.

[21] D. Helbing, D. Brockmann, T. Chadefaux, K. Donnay, U. Blanke, O. Woolley-Meza, M. Moussaid, A. Johanssonx, J. Krause, S. Schutte, and M. Perc, "Saving human lives: What complexity science and information systems can contribute," *J. Stat. Phys.*, vol. 158, no. 3, pp. 735–781, Feb. 2015.

[22] R. Kinney, P. Crucitti, R. Albert, and V. Latora, "Modeling cascading failures in the North American power grid," *Eur. Phys. J. B-Condensed Matter Complex Syst.*, vol. 46, no. 1, pp. 101–107, 2005.

[23] C. Braham and M. Small, "Complex networks untangle competitive advantage in Australian football," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 28, no. 5, May 2018, Art. no. 053105.

[24] B. Schäfer, D. Witthaut, M. Timme, and V. Latora, "Dynamically induced cascading failures in power grids," *Nature Commun.*, vol. 9, no. 1, pp. 1–13, Dec. 2018.

[25] P. Holme, "Congestion and centrality in traffic flow on complex networks," *Adv. Complex Syst.*, vol. 6, no. 2, pp. 163–176, Jun. 2003.

[26] M. P. Brundage, T. Sexton, M. Hodkiewicz, A. Dima, and S. Lukens, "Technical language processing: Unlocking maintenance knowledge," *Manuf. Lett.*, vol. 27, pp. 42–46, Jan. 2021.

[27] A. Hagberg, D. Schult, and P. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python Sci. Conf. (SciPy)*, G. Varoquaux, T. Vaught, and J. Millman, Eds., 2008, pp. 11–15.

[28] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. Int. AAAI Conf. Social Media*, 2009, vol. 3, no. 1, pp. 361–362.

[29] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Softw., Pract. Exper.*, vol. 21, no. 11, pp. 1129–1164, 1991.

[30] M. Jacomy, T. Venturini, S. Heymann, and M. Bastian, "ForceAtlas2, a continuous graph layout algorithm for handy network visualization designed for the Gephi software," *PLoS ONE*, vol. 9, no. 6, Jun. 2014, Art. no. e98679.

[31] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, Oct. 2008, Art. no. P10008.

[32] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir, "A model of internet topology using k-shell decomposition," *Proc. Nat. Acad. Sci. USA*, vol. 104, no. 27, pp. 11150–11154, Jul. 2007.

[33] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 1, pp. 35–41, Mar. 1977.

[34] U. Brandes, "A faster algorithm for betweenness centrality," *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.

JAYMIN MOFFATT received the B.Sc. degree with a double major in engineering science and mathematics and statistics, and the M.P.E. degree in electrical and electronics engineering from The University of Western Australia, in 2018 and 2021, respectively. In 2016, he received the Engineering Pursuit Scholarship and the UWA Excellence Award. In 2020, he began working at Horizon Power part time with the Asset Services Team undertaking tasks of both a Reliability Engineer and a Power Engineer.

AYHAM ZAITOUNY received the bachelor's degree in mathematics and statistics from Damascus University, in 2009, and the master's degree (Hons.) in mathematical and statistical sciences (applied mathematics) and the Ph.D. degree in mathematical and statistical sciences (applied mathematics) from The University of Western Australia (UWA), in 2012 and 2016, respectively. From 2016 to 2019, he worked as a Postdoctoral Fellow in complex engineering systems in a joint position between the Commonwealth Scientific and Industrial Research Organisation (CSIRO) and UWA. Since 2019, he has been a Research Fellow with the Department of Mathematics and Statistics, UWA, and the ARC Training Centre for Transforming Maintenance through Data Science. He is currently working as a member of several multidisciplinary research teams (at UWA, Harry Perkins Institute for Medical Research, the Telethon Kids Institute for Children Health Research, the CSIRO, and the ARC Training Centre for Transforming Maintenance through Data Science). His research varies among applications on ecological, biological, medical, geological, and engineering systems. His research interests include mathematical modeling, nonlinear time series analysis, dynamical systems, complex systems, differential equations, complex networks, data analysis, and filtering.

MELINDA R. HODKIEWICZ (Member, IEEE) received the B.A. degree (Hons.) in metallurgy and science of materials from Oxford University, in 1985, and the Ph.D. degree in mechanical engineering from The University of Western Australia (UWA), in 2004. She has been a BHP Billiton Fellow for Engineering for Remote Operations with UWA, since 2015. Previously, she was with industry in operations and maintenance roles. She currently leads the System Health Laboratory, UWA, where she is involved in the areas of asset health, maintenance, and safety. In 2018, she was a Visiting Fellow at Alan Turing Institute, the U.K's National Centre for Data Science and AI, and a Visiting Fellow at NIST, USA, in 2019. From 2012 to 2015, she was a member of the PC251 ISO55000 Asset Management Standard Committee. In 2015, she was awarded the MESA Medal, a lifetime achievement award for services to the Asset Management Community in Australia. She is a Chartered Engineer (U.K.), a member of the Institute of Materials, Minerals and Mining (IOM3), and the Asset Management Council. In 2020, she was made a fellow of the Australian Academy of Technology and Engineering (ATSE).

MICHAEL SMALL (Senior Member, IEEE) received bachelor's and Ph.D. degrees in pure and applied mathematics from The University of Western Australia (UWA), Perth, WA, Australia. After postdoctoral experience in Stellenbosch and Edinburgh, he worked at the Faculty of the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, from 2001 to 2011. His research interests include complex systems, complex networks, chaos and nonlinear dynamics, nonlinear time series analysis, and computational modeling. In 2011, he was a recipient of the Australian Research Council Future Fellowship and was made a Winthrop Professor of applied mathematics at the School of Mathematics and Statistics, UWA, in 2012. Since 2015, he holds the UWA-CSIRO Chair of complex engineering systems. He is the Deputy Editor-in-Chief of the journal *Chaos* and the Main Editor of *Physica A*.