Efficient Availability Assessment of Reconfigurable Multi-State Systems with Interdependencies

Hindolo George-Williams[a,b], Edoardo Patelli[a,*]

[a] *Institute for Risk and Uncertainty, Chadwick Building, University of Liverpool, Peach Street, Liverpool L69 7ZF, United Kingdom*
[b] *Institute of Nuclear Engineering & Science, National Tsing Hua University, Hsinchu, Taiwan*

Abstract

Realistic engineering systems often possess attributes that complicate their availability assessment. Notable examples being complex topology, multi-state behaviour, component interdependencies, and interactions with external phenomena. For such systems, analytical techniques have limited applicability, and efficient simulation techniques are therefore required. In this paper, a novel load-flow simulation approach is proposed to simplify the availability assessment of realistic engineering systems. The approach is simple and generally applicable to systems, including those with limited maintenance teams, reconfiguration requirements, and multiple commodity flows. A novel metric for assessing maintenance inadequacy and a real-time component ranking procedure are also introduced. In real-time ranking, failed components are assigned maintenance priorities during simulation in accordance with how much their availability improves system performance and how many idle maintenance teams there are. This eliminates the need for component importance ranking algorithms prior to simulation, which for some systems are unnecessary. The applicability of the approach is demonstrated by analysing an offshore plant producing oil, gas, and water. The solution obtained is compared against another Monte Carlo simulation-based solution that requires the enumeration of the plant's cut-sets. The proposed approach is shown to be more intuitive, robust to human-induced errors, and require less human effort.

Keywords:
Multi-State System, Simulation, Interdependencies, Availability, Limited Maintenance, Multi-Commodity

1. Introduction

Engineers and system designers are under immense pressure to build systems robust and adequate enough to meet the ever increasing human demand and expectation. Unavoidably, the resultant systems are complex and highly interconnected, which ironically constitute a threat to their resilience and sustainability. Majority of the systems we interact with on a day-to-day basis exist as multi-state interdependent systems. Two systems are interdependent if at least a pair of nodes (one from each system) are coupled by some phenomena, such that a malfunction of one affects the other. The coupling phenomenon could be proximity in space [1], functional dependence/interdependence [2], or both [3]. A water distribution network, where pumps and other electrical power-driven appliances rely on the reliability and performance of the power grid is a typical example.

The components of a system are normally prone to random failures arising from their intrinsic properties or induced failures stemming from targeted attacks [4], extreme environmental events [5], and erroneous human-system interactions. In interdependent systems, an undesirable glitch in one system could cascade and cause disruptions in coupled systems. The cascade could be fed back into the initiating system and the overall consequences may be catastrophic [1, 6]. This was made clear by the massive blackout that struck Italy in September 2003, affecting the internet network in the process. In the same year, North America was hit by a blackout that lasted 4 days, affecting parts of USA and Canada [7]. To minimize the effects of these failures, some interdependent systems are equipped with reconfiguration provisions. This normally entails transferring operation to another node, rerouteing flow through alternative paths, or shutting down parts of the system. It is, therefore, vi-

*Corresponding author
Email addresses: H.George-Williams@liv.ac.uk (Hindolo George-Williams), epatelli@liverpool.ac.uk (Edoardo Patelli)

tal to analyse the system's performance under the spectrum of possible vulnerability conditions, for adequate planning of defensive and contingency measures [8].

In general, the achievement of maximum overall system performance is desirable. However, in many applications, it is more important to recover the required system performance in the shortest possible time after component failure. This is the case, for instance, in nuclear power plant risk assessment, where the time-dependent recovery probability of offsite power is an important input to the overall safety of the plant [9]. Hence, system recovery time is not only a performance parameter, but a fundamental safety parameter as well. Given the positive correlation between costs and resources (human, financial, and material) required to maintain a system, under economic constraints, there may not be sufficient resources for a speedy recovery. Therefore, an informed and robust decision making process would dictate that the decision support tool used be capable of modelling relevant realistic aspects of the system, including the possibility of limited recovery response.

Various models have been developed to study the effects of interdependencies on systems [8, 10]. However, a good number of these only assess their response to targeted attacks, variation in some coupling factor or the relative importance of system nodes [1, 2, 11]. According to Ouyang [10], these models alone cannot sufficiently analyse the performance of interdependent systems. He intimated that flow based approaches, taking into account material or service flow across the system were required. When faced with the situation of random node failures, a complete reliability and availability analysis should be performed. However, renown analytical multi-state system reliability evaluation techniques like Binary Decision Diagrams (BDD) [12, 13], Sum-of-Disjoint-Products (SDP) [14], and the Universal Generating Function (UGF) [15–17] are of very little use to the evaluation of these systems. Their inapplicability is amplified if, nodes can undergo non-Markovian transitions, their restoration can be delayed, the system is reconfigurable or in the case of BDD and SDP, the system is complex, such that state enumeration is infeasible. In spite of these challenges, there are a few successful attempts at their application to systems with some form of dependencies. Levitin, for instance, in [18] and [15], respectively applied the UGF approach to systems with lateral dependencies and systems prone to common-cause failures. Both instances, however, involved only one system with a single commodity. Stochastic Petri Nets [19, 20] and Bayesian Networks [21] are another set of powerful computational tools for reliability modelling of systems with dependencies. However, they also require state enumeration when applied to multi-state systems, which may be infeasible for some complex system architectures.

Certain realistic aspects of interdependent systems, as previously mentioned, are implementable only by simulation algorithms [22]. However, most multi-state system simulation algorithms rely either on the structure function of the system or enumeration of the system's path or cut sets [22, 23]. Both procedures get cumbersome even for complex systems of moderate size, and with them, the shut down and restart of components (a type of reconfiguration) is non-intuitive [24].

1.1. Proposed Approach

The authors recently presented a load-flow simulation technique for the analysis of multi-state systems [24]. In the technique, each system node is modelled as a semi-Markov stochastic process and the system structure as a directed graph. An event-driven simulation is used to reconstruct the random failure and repair events of system nodes. As nodes go through their cycle of failures and subsequent repairs, their capacities change, and the interior-point algorithm [25] is used to determine the performance of the system. The approach employs an adjacency matrix to define the structure of the system and derives the equations of flow across the entire system in the form of matrices. This particularly makes it suitable and intuitive for any system architecture and easily programmable on a digital computer. In terms of applicability, it outperforms other multi-state system analysis approaches, since it does not require state enumeration or cut set definition. It considers realistic system aspects like flow losses, reconfiguration, forced transitions, and multiple competing demands. The approach, however, is only applicable to homogeneous independent systems and does not consider restrictions on the number of simultaneous maintenance actions that can take place in the system.

In this work, the load-flow simulation approach is extended to support interdependencies, multi-commodity systems, and model limited maintenance team scenarios. Though largely built on the principles proposed in the original work [24], this work makes a series of new contributions as highlighted thus;

- We define a straight-forward procedure for uncoupling interdependencies in systems and propose an intuitive mathematical model for their adequate representation.

- Two recursive algorithms are proposed to accurately account for these interdependencies and compute the performance of the system during simulation.
- To enhance the efficient extraction of system availability and performance indices from the simulation result, easily implementable algorithms have been proposed. Availability, as used here, refers to the ability of a system to function as expected. It, therefore, encompasses the reliability, the output characteristics, and the recovery probability of the system after its deviation from expected performance.
- System reliability and output characteristics are already very common system availability indices. We use the recovery probability and a new metric for assessing the adequacy of the maintenance process as additional system performance indices.
- We also propose a real-time component ranking procedure to help decide the sequence of maintenance response that maximises system performance. In practice, the system operator would use this procedure whenever a scenario dictating preferential maintenance arises.
- Finally, a simple but important modification is also made to the original system flow calculation procedure, resulting in appreciable gains in computation time.

In summary, this work extends the applicability of the load-flow simulation approach and improves its computational efficiency.

1.2. Paper Structure

The remainder of the paper is organised as follows; the next section is dedicated to providing an overview of the relevant modifications to the load-flow approach to include interdependencies. In this section, a generalised procedure for assessing the availability of interdependent multi-state systems is also presented. Details of the simulation procedure and availability assessment algorithms are respectively provided in Sections 3 and 4. Section 5 addresses the availability assessment problem of an offshore multi-commodity plant. The plant is used to illustrate the systematic roll out of the solution strategy developed in Section 2 to a practical problem of industrial relevance. The usefulness of the new metric for maintenance inadequacy and real-time component ranking are also illustrated here. Implications of the results, efficiency of the approach and its limitations climax this section. Finally, the closing remarks; drawing conclusions on the proposed approach make up Section 6.

2. Implementation

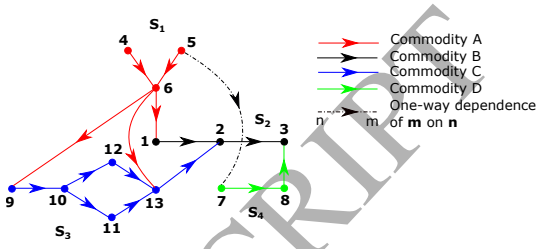


Figure 1: An example of a typical interdependent system

In this section, the relevant principles governing the modelling of the system and its components are described. They are based on the earlier work presented in [24], as a result, premium is placed only on the necessary modifications. For this purpose, we use the arbitrary system shown in Fig. 1, which could be a binary-state system or a multi-state flow network [26]. It consists of 4 subsystems and 13 nodes, transporting 4 commodities. The number of subsystems is normally defined by the number of commodities or more generally by the number of closed-loops. This implies, a system could be composed of multiple subsystems even when only one commodity type is involved. Nodes 1, 2, and 3, transporting commodity-B, respectively require commodity, A, C and D to operate and nodes 9 and 13 in subsystem S_3 rely on flow from subsystem S_1. Also, a certain failure mode of node 5 in subsystem S_1, triggers the partial failure of node 7 in subsystem S_4. This type of interdependence is called one-way dependence, since the failure of node 5 affects node 7, but state change events in node 7 have no effect on node 5. Even for a system this simple, deriving all cut-sets is time-consuming and error-prone. In the remainder of this section, a generally applicable procedure to overcome these complications is presented.

2.1. Decoupling the System

To start the modelling process, all the elements affecting the operation of the system are identified and numbered as illustrated in Fig. 1. This is followed by the identification and definition of all the node dependencies. Constituent systems (hereafter referred to as subsystems), determined by the different commodities

flowing in the system or the number of closed-loops are assigned subsystem IDs. For each subsystem, the associated nodes are identified, the subsystem graph model developed and the relevant flow equation parameters obtained (see [24, 27] for details). In identifying the nodes of a subsystem, only nodes with actual commodity flow are considered. Nodes representing external common-cause initiators, environmental events, or human-system interactions do not belong to a subsystem. Next, the possible states of each node are identified and modelled as outlined in [24].

Consider the system presented in Fig. 1, with focus on load dependencies. Node 2, for instance, uses commodity-C to drive its operation but transmits commodity-B. One would say it exhibits a dual operation mode, operating both as a sink and a transmission node. The sink mode directly influences flow in S_3, while the transmission mode has a direct influence on flow in S_2. It's, therefore, logical to separate the node into its constituent nodes, each representing a mode of operation. The node representing the sink mode is assigned a new ID while the other retains the ID of the original node. A load-source dependency exists between the nodes, since the transmission node is incapacitated if flow into the sink node is inadequate. They, therefore, make a load-source pair, with the transmission node being the load, and the sink node, the local source. This procedure is applied to all load dependency relationships in the system to obtain the following load-source pairs, $\{2, 14\}$, $\{3, 16\}$, $\{1, 18\}$, $\{13, 15\}$, and $\{9, 17\}$, as highlighted in Fig. 2.

Local sources, otherwise known as support nodes in load-source pairs are modelled as binary-state objects. State 1, designated active, and assigned a capacity, l,

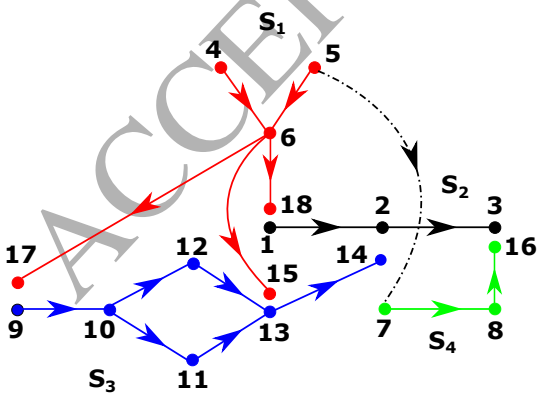


Figure 2: Interdependent system showing load-source pairs

signifies the availability of the dependent node. State 2, with 0 capacity, and designated inactive, depicts otherwise. l is the minimum level of support required to operate the node in the transmission mode, and in practical cases represents the load rating of that component. A $4kW$ rated 3-phase centrifugal pump, for instance, would have $l = 4kW$.

To incorporate these interdependencies in the component model proposed in [24], two additional parameters; $\mathbf{L}$ and $\mathbf{D}$ are introduced. Let i be the index of a node, with $\mathbf{L}_i = \{j, l\}$ defining its load dependency with node j. The dependency defined by $\mathbf{L}_i$ is interpreted as, node i requiring a minimum of l level of flow from node j to operate. When i and j belong to different subsystems, the subsystems are said to be interdependent, since a state change in either node affects flow in both subsystems. If i has load dependency relationships with multiple nodes, $\mathbf{L}_i$ takes the form of a 2-column matrix; each row defining its relationship with another node. Parameter $\mathbf{D}_i = \{d_{j1}, d_{j2}, d_{j3}, d_{j4}\}_{u\times 4} \mid j = 1, 2, ..., u-1, u$ defines the single-way causal-effect relationship between node i and other nodes. This type of coupling specifies induced state changes in other nodes following a state change in i. d_{j1} is the state of i triggering the event, d_{j2}; the affected node, d_{j3}; the state the node has to be in to be affected, and d_{j4}; its target state on occurrence of the event. Each row of $\mathbf{D}_i$, therefore, defines the behaviour of an affected node, and u, the number of relationships. If i and the affected node, d_{j2}, belong to different subsystems, the subsystem the latter belongs to is dependent on the subsystem of the former. In Fig. 2, for instance, S_4 depends on S_1, consequent of the relationship between nodes 5 and 7. Suppose state 3 of node 5 triggers the partial failure (state 2) of node 7. If this happens only when node 7 is in state 1, their dependency is defined by $\mathbf{D}_5$ as,

$$\mathbf{D}_5 = \begin{pmatrix} 3 & 7 & 1 & 2 \end{pmatrix} \tag{1}$$

Equation 1 effectively defines the state change induced in node 7 by a state change in node 5. Using the notation described in the preceding paragraph, the expression states if node 5 makes a transition to state 3 whilst node 7 is in state 1, the latter is forced through a transition to state 2. Similarly, if a state change in node 7 triggered a state change in node 5 or any other node, $\mathbf{D}_7$ would be required to express this mathematically.

The final step entails the derivation of the dependency tree relating the subsystems, and the ranking of these subsystems according to their position on the tree. In the ranking procedure, the independent subsystem is chosen as reference and assigned rank 1. The other subsystems

are ranked in increasing order of their longest distance from this reference. Fig. 3 shows the dependency tree for an arbitrary 4-subsystem system (different from the system in Fig. 1), where the designation $[a, b]$ specifies that the rank of subsystem a is b. If two subsys-

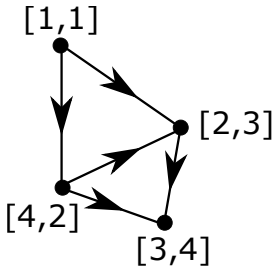


Figure 3: Dependency tree for a 4-subsystem system

tems are interdependent, their mutual link on the tree is discarded, and the ranking done as earlier described. However, if after discarding the mutual links, a node is totally cut off from the rest of the tree, it's assigned the same rank as its dependent pair on the tree. If it's in relationship with multiple nodes, its rank, b, is given by $max\,(\mathbf{R})$, $\mathbf{R}$ being the set of ranks of all subsystems it is associated with. Fig. 4 is an illustration of the rank-

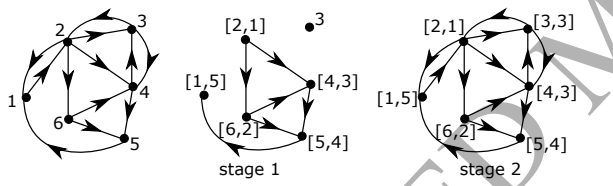


Figure 4: Dependency tree: Subsystem ranking procedure

ing procedure for a system composed of six subsystems with interdependencies. Starting with the tree on the left, all mutual links are discarded, leaving node 2 as the only node without a parent. Hence, it is taken to be the reference, and the nodes ranked to complete stage 1 of the ranking procedure. Discarding the mutual links leaves node 3 completely isolated. However, it is in relationship with nodes 2 and 4, ranked 1 and 3 respectively. Node 3, therefore, is assigned rank 3; the maximum of the ranks of the nodes it is associated with.

Let $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$ and $\mathbf{S}_4$ be the sets of nodes respectively belonging to subsystems S_1-S_4 of the system presented in Fig. 1. From Fig. 2, $\mathbf{S}_1 = \{4, 5, 6, 15, 17, 18\}$, $\mathbf{S}_2 = \{1, 2, 3\}$, $\mathbf{S}_3 = \{9, 10, ..., 14\}$ and $\mathbf{S}_4 = \{7, 8, 16\}$. With numbers 1-4 chronologically assigned to subsystems S_1-S_4, the system's dependency tree is shown in Fig. 5. With all the subsystems ranked, an indicator register, $\mathbf{I}$, of zeros, such that each element corresponds to

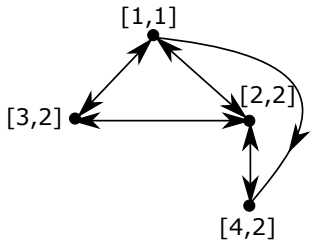


Figure 5: Dependency tree for sample interdependent system

a subsystem is defined. This register indicates (by logic 1 in the relevant position) the subsystem(s) affected by the last node transition.

2.2. Accounting for Dependencies

Let $\boldsymbol{\mu}$ be the vector holding the current performance levels of system nodes. When node i makes a transition that results in a change in its performance level, the current capacity of its load-source pair, j, is modified. If $c_{x^-}^{\{i\}}$ is the node's capacity before transition and $c_x^{\{i\}}$, its current capacity, the capacity of node j changes according to Equation 2. Where $(j, \boldsymbol{\mu})$ denotes the j^{th} element of $\boldsymbol{\mu}$.

$$(j, \boldsymbol{\mu}) = c_x^{\{j\}} = \begin{cases} 0 & \text{If } c_{x^-}^{\{i\}} > 0 \text{ and } c_x^{\{i\}} = 0 \\ l & \text{If } c_{x^-}^{\{i\}} = 0 \text{ and } c_x^{\{i\}} > 0 \end{cases} \tag{2}$$

A recursive algorithm is required to account for the causal-effect relationships between nodes because of the possibility of nested dependencies. If $\mathbf{D}_i$ and x_i are respectively the dependency matrix and current state of node i, the following steps summarise the algorithm;

Step 1 Define a register to hold affected nodes and their target states.

Step 2 Find all nodes affected by the state change (using $\mathbf{D}_i$ and x_i) and update the register defined in step 1.

Step 3 Select the last entry, node y, of the register, set its current state to its target state and delete its records from the register.

Step 4 Using $\mathbf{D}_y$ and x_y obtained in step 3, in place of $\mathbf{D}_i$ and x_i, repeat steps 2 and 3.

Step 5 Repeat steps 2 through 4 until the register defined in step 1 is empty.

On each node transition, $\boldsymbol{\mu}$ is updated, and any load dependencies accounted for as described by Equation 2.

2.3. Node Reconfiguration

Node i is shut down if its flow falls to or below its threshold, Λ_i, or if flow through its load-source pair, j, falls below l. If a node is shut down, its current and next states are saved, its next transition time set to ∞, and its current capacity, to 0. When the condition leading to shut down is resolved, the node is restarted and restored to its previous state. The period, t_{shift}, spent in shut down is accounted for by shifting its next transition time to $t'_{next} + t_{shift}$, where t'_{next} is its transition time before shut down. This time shifting is repeated for the node's next preventive maintenance due time, if its preventive maintenance interval is a function of the time spent in operation.

In practice, maintenance durations are not affected by node shut down events. Therefore, if the next state of a node is superior in performance and reliability to its current, only its current capacity is modified. Modifying its next transition time would mean delaying its restoration, which may negatively affect the simulation outcome. Algorithms for shut down and restart of nodes are presented in [24]. However interdependencies are not considered, therefore the following modifications should be adopted.

Let $\boldsymbol{\delta}$ be the set of all nodes currently in shut down state and $\boldsymbol{\eta}$, the vector of system node flows. Node i is added to $\boldsymbol{\delta}$ if and only if its shut down is due to the condition, $(i, \boldsymbol{\eta}) \leq \Lambda_i$. As a rule-of-thumb, nodes that do not satisfy the threshold flow condition are shut down first. Next, flows through sink nodes that have load-source pairs are assessed. If for a sink node, j, $(j, \boldsymbol{\eta}) < l$, its load-dependent node, i, is shut down. The same order is followed for node restart, where node flows are assessed for satisfaction of the relevant conditions. If the condition $(j, \boldsymbol{\eta}) = l$ is met for a sink node, its load-dependent pair, i, in shut down is restarted.

2.4. Determining system performance at time t

The goal of system analysis is to determine the amount of commodity flow through output nodes. This in turn requires that flow is calculated after every transition that results in a performance level change of a node. Owing to node interdependencies, a state change in one node may give rise to state changes in a series of other nodes. The system may go through a number of performance levels in the process, but the effective performance is the one attained after the last transition. A recursive algorithm is employed for this purpose, as outlined thus;

Step 1 Define $\boldsymbol{\mu}_t$; a temporary variable and set its value to $\boldsymbol{\mu}$ (i.e., $\boldsymbol{\mu}_t = \boldsymbol{\mu}$). Where $\boldsymbol{\mu}$ is the vector of current node capacities.

Step 2 In $\boldsymbol{\mu}_t$, set the capacities of all the nodes in $\boldsymbol{\delta}$ to their values before shut down. This step is required to determine which nodes in shut down can be restarted.

Step 3 Using $\mathbf{I}$, select the highest ranked subsystem which indicator is 1 and calculate its flow using $\boldsymbol{\mu}_t$. The highest ranked subsystem corresponds to the subsystem with the smallest rank. If multiple subsystems meet this requirement, randomly select a candidate. Go to step 8 if there are no non-zero elements in $\mathbf{I}$.

Step 4 Set in $\mathbf{I}$, the indicator for the subsystem in step 3 to 0.

Step 5 Restart and shut down nodes according to the procedure outlined in Section 2.3.

Step 6 Following a node transition, the subsystem hosting the node is identified, and its position in $\mathbf{I}$ set to 1. This is only required if the shut down or restart of the node is a direct effect of a state change in its load-source pair (see Section 2.1).

Step 7 Repeat steps 1 to 6, making sure interdependencies are accounted for, and $\boldsymbol{\mu}$ updated on every transition.

Step 8 Get the flows through the output nodes, save as a function of time and terminate algorithm.

3. The System Simulation Procedure

Simulation normally entails repeated calculation of system output, as nodes undergo their transition cycles. Calling the interior-point algorithm for every transition, as proposed in [24], may impose unprecedented computational burden. This is because, a certain system configuration may be attained more than once, making multiple calculations for the same configuration a possibility. To overcome this problem, it's desirable to determine node flows for all the possible combinations of system node performance levels prior to simulation. Let $\boldsymbol{\beta}$ be the matrix holding these combinations and $\mathbf{C}_u^{\{i\}}$, the set of unique performance levels of node i. $\boldsymbol{\beta}$ is an $M \times \prod_i^M n_i$ matrix; M being the total number of nodes excluding external nodes, and n_i, the number of unique performance levels of node i. For instance, if the capacity of node 1 is defined by $\mathbf{C} = \{10, 20, 0, 0, 10\}$, $\mathbf{C}_u^{\{1\}} = \{0, 10, 20\}$ and $n_1 = 3$. For each combination of performance levels, the corresponding node flows are calculated and recorded in a second matrix, $\mathbf{F}$. During

simulation, $\boldsymbol{\beta}$ is searched for the combination of node performances corresponding to the current system configuration, and its pre-stored node flows in $\mathbf{F}$ simply read off. By this, flow calculation for every configuration is carried out only once. It is worthwhile noting that for large systems, $\boldsymbol{\beta}$ gets prohibitively large, such that the time to search for a configuration exceeds its flow calculation time. The search time, however, can be reduced by smart allocation and search procedures. Therefore, it's advised that the average times to complete both procedures are compared prior to simulation.

For an interdependent system like the one in Fig. 1, the procedure described above is carried out for each subsystem, and in each case, only nodes belonging to that subsystem are considered.

3.1. Node Transition Parameters

Determining the transition time, t_{next}, of nodes is the core of every simulation algorithm. Given the current state, x, of a node, all the possible transitions from x are sampled, and their minimum value, t_{min}, selected. The transition producing t_{min} is the node's next transition, occurring at $t_{next} = t + t_{min}$, t being the current simulation time. If t_{min} is associated with multiple transitions, one of them is randomly selected, as specified by the sampling algorithm in [24].

3.2. Forcing Maintenance

Algorithm 1 Forcing maintenance: Limited dedicated maintenance teams

Require: $m_1, m_1', m_2, m_2', h_1$ and h_2

$k \leftarrow 1$ ▷ initialize indicator
while $k \leq 2$ **do**
  $v \leftarrow m_k - m_k'$ ▷ get idle teams
  while $v > 0$ **and** $h_k \neq \emptyset$ **do**
    select node according to priority
    make maintenance state the current state x
    sample next transition using x
    delete node from h_k
    $v \leftarrow v - 1, \quad m_k' = m_k' + 1$
  end while
  $k \leftarrow k + 1$
end while

With limited maintenance teams, maintenance actions are not instantaneous. Therefore, the transition from a degraded state or to Preventive Maintenance has to be manually executed during simulations. Let m_1 denote the number of teams dedicated to Corrective Maintenance (CM), m_2, the number of Preventive Maintenance (PM) teams, m_1', the number of busy CM teams, and m_2', the number of busy PM teams. Following its transition, a node is added to the set, h_1, of nodes requiring repairs if its new state is directly linked to a CM state, and to h_2, if its PM is due. At time, t, maintenance is forced if there are idle maintenance teams and h_1 or h_2 is not empty. This procedure is described by Algorithm 1, where $k = 1$ and $k = 2$ respectively denote CM and PM.

Algorithm 2 Forcing maintenance: Limited shared maintenance teams

Require: m, m', h

$v \leftarrow m - m'$ ▷ get idle teams
while $v > 0$ **and** $h \neq \emptyset$ **do**
  select node according to priority
  make maintenance state the current state x
  sample next transition using x
  delete node from queue (h and one of h_1 and h_2)
  $v \leftarrow v - 1, \quad m' = m' + 1$
end while

If PM is carried out only when the system is perfect, the algorithm is terminated after the task for $k = 1$ if at least one of $h_1 \neq \emptyset$, $m_1' > 0$, and $m_2' > 0$ is true. Each of these conditions means either there is a failed component waiting to be repaired or maintenance is in progress, any of which suggests the system is not in a perfect state. In most applications, PM of a node is delayed if it is in a degraded state until after CM. If this is the case, a node belonging to both h_1 and h_2 is rejected when selected during the PM task ($k = 2$) of the algorithm. Also, most systems are assumed to operate under a perfect maintenance scenario. This means, nodes are repaired to an as-good-as-new condition, making any pending PM tasks for a repaired node no longer necessary. In such cases, a node's records are deleted from both h_1 and h_2 after CM.

Algorithm 1 is based on the assumption that CM and PM are carried out by different teams (dedicated maintenance). It is, however, adaptable to systems where the same team can carry out both maintenance actions (shared maintenance). Let m be the total number of maintenance teams, m', the number of busy maintenance teams, and $h = (h_1 \cup h_2)$, the set of nodes in the maintenance queue. Algorithm 2 outlines the procedure for forcing maintenance in shared maintenance scenar-

ios.

3.3. Maintenance Priority & Real-time Ranking

In practical applications, system availability is computed based on some predefined maintenance priority ranking of nodes [22]. However, for complex systems, node priority ranking is either impossible, time consuming, error-prone, or requires component importance ranking algorithms [28, 29]. Priority maintenance is required to reduce the *maintenance response inadequacy* for critical nodes, a measure of how long a node waits in the maintenance queue. It depends on the number of CM teams, the failure characteristics of nodes, and the efficiency of the maintenance teams. This implies, a system may have fewer CM teams than nodes and still not require priority maintenance if node failures are rare or maintenance durations are relatively short. For such systems, pre simulation priority ranking is a mere waste of time and computing resources. In view of this, real-time node ranking, where nodes are ranked during simulation is proposed. Prior to forcing maintenance, failed nodes are arranged into groups, based on the number of available CM teams. Let $v_1 = m_1 - m'_1$ be the number of idle CM teams at time t, and n_1, the number of failed nodes. If there are more failed nodes than there are available CM teams (i.e., $v_1 < n_1 \mid v_1 \neq 0$), all the possible combinations of nodes that can be repaired are generated. This produces $\binom{n_1}{v_1}$ groups, each containing v_1 failed nodes. The nodes in the first group are temporarily set to their expected performance levels after CM, whilst those in the other groups remain in their current states. The expected system performance, given the new node states is determined, and the procedure repeated for all the node groups. The maintenance of the nodes in the group with the best system performance is initiated.

Real-time ranking, therefore, does not only take into consideration the current system conditions, but takes place only when necessary. The latter being an important computational efficiency feature, especially for complex systems. The ranking procedure described can be replicated for PM and shared maintenance scenarios. In the case of PM, performance levels of nodes are set to their expected values during PM.

3.4. The Simulation Algorithm

With the relationships between system nodes defined, a Monte Carlo simulation algorithm is required to reconstruct the operation of the system, and derive its availability indices. An efficient event-driven and non system-specific simulation algorithm is proposed for this purpose. It proceeds by going through sampled node transitions, determining the flows through output nodes, and collecting these flows as a function of time. Starting with nodes in their initial states at time, $t = 0$, the system's initial performance is determined, and the next transition times of nodes sampled. The simulation jumps to a new time, $t = t_{min}$, where t_{min} is the minimum of the next transition times of nodes. Nodes with transition times equal to t_{min} are identified, the required state changes effected, their next transition times sampled, the new system performance deduced, and the next simulation jump determined. This continues until the mission time, t_m, is exceeded. The relevant availability and performance indices are derived at the end of the simulation from the saved system performance history. To ease the computational burden, flow is calculated only if at time, t, the current and previous system node capacity vectors ($\boldsymbol{\mu}$ and $\boldsymbol{\mu}_{old}$) are different. The simulation procedure is summarised thus;

Step 1 Initialise the register to store the output node history, and calculate flows across all the subsystems, as described in Section 3. Define the mission time, t_m, number of simulation samples, N, and number of maintenance teams, (m_1, m_2).

Step 2 Define, $\boldsymbol{\mu}$ from the initial states of nodes, and the initial output node performance (from step 1). Set $\boldsymbol{\mu}_{old} = 0$, $\boldsymbol{\delta} = h_1 = h_2 = \emptyset$, $t = m'_1 = m'_2 = 0$, and all the elements of $\mathbf{I}$ to zero.

Step 3 Sample the next transition time for all the nodes, and update $\boldsymbol{\tau}$ (the register holding node transition times). Also determine their PM due times (if applicable), and store in $\boldsymbol{t}_{pm}$.

Step 4 Identify the nodes with transition time equal to t. Update their current states, $\boldsymbol{\mu}$, h_1, and account for any interdependencies.

Step 5 For each node in step 4, determine its subsystem, and set the indicator of the latter in $\mathbf{I}$ to 1. Sample the node's next transition, and update $\boldsymbol{\tau}$. If it is just from PM, determine its next PM due time, and update $\boldsymbol{t}_{pm}$.

Step 6 Determine nodes whose value in $\boldsymbol{t}_{pm}$ equals t, and add them to h_2.

Step 7 Force maintenance, as described in Section 3.2, if there are nodes in the maintenance queue and there are available maintenance teams. That is, $h_1 \neq \emptyset$ & $m_1 - m'_1 > 0$ or $h_2 \neq \emptyset$ & $m_2 - m'_2 > 0$.

Step 8 If $\boldsymbol{\mu}_{old} \neq \boldsymbol{\mu}$, determine the system performance, as outlined in Section 2.4.

Step 9 Set $\boldsymbol{\mu}_{old} = \boldsymbol{\mu}$, and determine the next system transition time, t. This is given by the minimum of all the possible transition times. The next simulation jump, therefore, occurs at time, t, where t is given by, $min(min(\boldsymbol{\tau}), min(\boldsymbol{t}_{pm}))$.

Step 10 Repeat steps 4 through 9 until t exceeds t_m.

Step 11 Repeat steps 2 through 10, N times, and compute the relevant availability indices.

4. Obtaining the Availability and Performance Indices

Details on frequently used availability and performance indices are available in many standard reliability texts [15, 30]. However, the following indices; *reliability*, *recovery probability*, *instantaneous availability*, *steady-state availability*, *instantaneous output*, and *expected output* have been considered, for completeness.

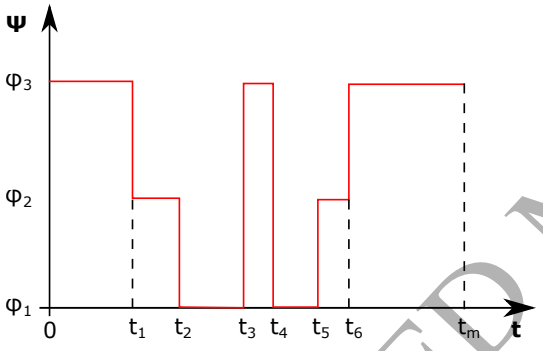


Figure 6: Example of a system performance history for one Monte Carlo realisation

During system simulation, a node transition results in system transition only if it leads to the attainment of a new system performance level. The main task, therefore, is the collection of these performance levels and their corresponding attainment times. For a simulation sample, let system performances be stored in $\boldsymbol{\Psi}$, as they are attained, and the corresponding transition times, in $\boldsymbol{t}$.

$$\boldsymbol{\Psi} = \{\psi_i\}^j \mid \psi_i \in \boldsymbol{\Phi}, \quad \boldsymbol{t} = \{t_i\}^j \mid 0 < t_i \leq t_m \qquad (3)$$

Let these be defined according to Equations 3, where t_i is the i^{th} transition time, ψ_i, the corresponding system performance, and j, the total number of system transitions. $\boldsymbol{\Phi} = \{\phi_1, \phi_2, ..., \phi_k\}$ is the set of possible system performances obtained from the simulation, where $\phi_z \mid z = 1, 2, ..., k$ is the z^{th} system performance level and k, the number of possible performance levels. Shown in Fig. 6 is the performance history of a hypothetical 3-performance level system. A system simulation of N samples contains N such histories, and are used to derive the various reliability and availability indices. The algorithms proposed for this purpose are based on an efficient translation of the system simulation history. Translating the multi-state performance history, $\boldsymbol{\Psi}$, to a binary string effectively reduces the multi-state reliability problem to its simpler binary-state counterpart. This enhances the application of well-known binary-state system reliability algorithms to the calculation of multi-state system reliability and performance indices.

4.1. System Reliability and Recovery Probability

The reliability, $R(t)$, of a repairable system at time, t, is the probability that the system will not fail between times 0 and t, given it was new or repaired to as-good-as-new at time 0. Failure is relative, and depends on the type of system and the success criteria set by the analyst. For multi-state systems, it is normally defined with respect to the system operating below a certain performance level. The likelihood that the system will be restored to this performance level in time, t, after failure, is defined by its recovery probability, $r(t)$.

Consider the system represented by Fig. 6, and let it be considered failed when operating below ϕ_3. Fig. 7

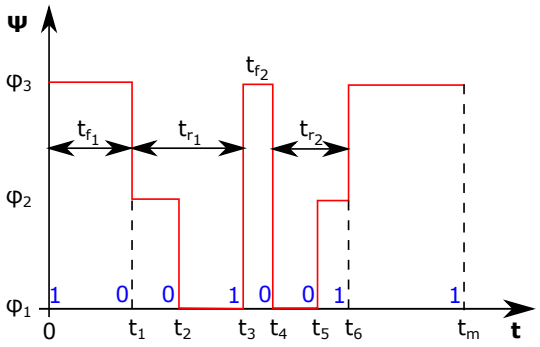


Figure 7: System performance history showing failure and recovery times

shows its performance history for one simulation sample, with annotations portraying the physical meanings of failure and recovery. t_{f_i} is the time the system takes at ϕ_3 before the i^{th} failure, and t_{r_i}, the corresponding recovery duration. If the transitions resulting in system performance of ϕ_3 are replaced with 1, and 0, otherwise, the system performance history is translated into a string of 1's and 0's, as shown in Fig. 7. This string is used in conjunction with $\boldsymbol{t}$ to derive the reliability and other

performance indices of the system. For instance, system failures and recovery are easily identified respectively by the positions of the sub strings '1 − 0' and '0 − 1' in the string. If these positions are defined by vectors $\boldsymbol{\sigma}_f$ and $\boldsymbol{\sigma}_r$, the sets of system failure and corresponding recovery times can be obtained from $\boldsymbol{t}$. These sets respectively define the failure and recovery time density functions, $f(t_f)$ and $f(t_r)$ of the system. Their corresponding cumulative density functions, $F(t_f)$ and $F(t_r)$ are used to deduce $R(t)$ and $r(t)$ from $1 - F(t_f)$ and $1 - F(t_r)$ respectively. For a multi component system assumed initially perfect (which is the case if reliability calculation is of interest), only the first failure times, t_{f_1}, of each simulation sample are used to define $f(t_f)$. This is explained by the fact that, depending on the structure and properties of its components, a system may have one or more components in a degraded state and still attain/maintain the required performance level. It is the case, for example, in a 2-branch, purely parallel system, where, one branch is sufficient to attain nominal system performance. Though both yield the same performance, the system with only one branch available has a higher probability of failure. System failure times yielded in this case, therefore, underestimate the reliability of the system. A system's performance history alone cannot say exactly the states of its nodes at system recovery. It is, therefore, impossible to determine whether higher order failure times, $t_{f_2}, t_{f_3}, ..., t_{f_j}$, were yielded by the perfect system. Though this is possible by collecting the system state vector (vector of node states) at every transition, it is a computationally expensive option. Hence, system reliability, redefined as a function of first failures only, is expressed as, $R(t) = 1 - F(t_{f_1})$.

A system's reliability and recovery probability have been shown to be derived from the cumulative density functions of its failure and recovery times respectively. These density functions are directly obtainable from the system performance history via an approximation technique. With the failure and recovery times collected, the mission time, t_m, can be divided into time-steps, δt, and the average contribution of each failure time, t_{f_i}, and recovery time, t_{r_i}, to each time-step estimated. The accuracy of the estimates is determined by how small δt is, relative t_m. It's, however, worthwhile noting that the discretisation is only required to estimate the instantaneous performance indices, the actual simulation does not require time-steps. Outlined below are the steps for deducing $R(t)$ and $r(t)$ from the system performance history.

Step 1 Define n; the number of time-steps, such that $n = \lceil t_m/\delta t \rceil$ and ϕ_{ref}, the performance of interest. Set $R(t) = r'(t) = \{0\}^n$, $\lambda = 1$ and $count = 0$, where $count$ is the number of recovery times computed, and λ, the index of the current simulation history.

Step 2 Given $\boldsymbol{\Psi}_\lambda$ and $\boldsymbol{t}_\lambda$ are the performance history of the λ^{th} simulation sample, modify their contents to include system performances at $t = 0$ and $t = t_m$. The performance, ψ_j, at the last system transition is taken to be the performance at $t = t_m$.

Step 3 Define a binary string, $\boldsymbol{\Theta} = \{\theta_i\}^j \mid i = 1, 2, ..., j$, such that $\theta_i = 1$ if $\psi_i \geq \phi_{ref}$, and 0, otherwise.

Step 4 Obtain $\boldsymbol{\sigma}_f$ and $\boldsymbol{\sigma}_r$; the locations of failures and recoveries in $\boldsymbol{\Theta}$ with their corresponding times, $\boldsymbol{T}_f$ and $\boldsymbol{T}_r$, such that $\boldsymbol{T}_f = \left(\boldsymbol{\sigma}_f + 1, \boldsymbol{t}\right)$ and $\boldsymbol{T}_r = (\boldsymbol{\sigma}_r + 1, \boldsymbol{t})$.

Step 5 Take the first element of $\boldsymbol{T}_f$, determine ch; the number of time-steps it represents and increment the first ch values of $R(t)$ by 1. That is, $(1 \rightarrow ch, R(t)) = (1 \rightarrow ch, R(t)) + 1$; where, $ch = \lceil \boldsymbol{T}_f(1)/\delta t \rceil$ and $(1 \rightarrow ch, R(t))$ represents elements 1 to ch of $R(t)$.

Step 6 Discard the last element of $\boldsymbol{T}_f$ if it has more elements than $\boldsymbol{T}_r$, and determine the recovery durations, $rtime$, such that, $rtime = \boldsymbol{T}_r - \boldsymbol{T}_f$.

Step 7 Take the first element of $rtime$, determine ch; the number of time-steps it represents and increment the first ch elements of $r'(t)$ by 1, such that, $(1 \rightarrow ch, r'(t)) = (1 \rightarrow ch, r'(t)) + 1$. Also increment the recovery time counter, $count$, by 1, such that, $count = count + 1$.

Step 8 Repeat step 7 until all the elements in $rtime$ have been covered, and increment the simulation index counter, λ, by 1, such that, $\lambda = \lambda + 1$.

Step 9 Repeat steps 2 to 8 until $\lambda = N + 1$, N being the number of simulation samples.

Step 10 Compute the reliability as, $R(t) = R(t)/N$, the non-recovery probability, as $r'(t) = r'(t)/count$, the recovery probability as $1 - r'(t)$, and terminate the algorithm.

4.2. Instantaneous Availability and Expected System Output

Instantaneous availability, $A(t)$, is the probability that system performance at time, t, is greater than or equal to some reference, ϕ_q. The expected system performance at this time defines the instantaneous output, $X(t)$.

Let $\boldsymbol{P}(t) = \{p_i(t)\}^k$ be the vector of instantaneous state probabilities; the probability of the system being in each

of the performance levels, $\mathbf{\Phi} = \{\phi_1, \phi_2, ..., \phi_k\}$ at time, t. Assuming the elements of $\mathbf{\Phi}$ are ordered, such that, $\phi_i < \phi_{i+1} < ... < \phi_k$, $A(t)$ and $X(t)$ are defined as,

$$A(t) = \sum_{i \geq q}^{k} p_i(t), \quad X(t) = \sum_{i=1}^{k} \phi_i p_i(t) \tag{4}$$

The average values of $A(t)$ and $X(t)$ over the mission time, respectively define the steady-state availability and the expected system output. They are obtained from Equation 4 by replacing the instantaneous state probabilities, $p_i(t)$, with their steady-state values, p_i. Where, p_i is the fraction of the mission time spent at performance level i. Therefore, obtaining a system's state probabilities is key to its availability and performance assessment. The following steps describe how this is efficiently achieved.

1. Define n; the number of time-steps, such that $n = \lceil t_m/\delta t \rceil$, k; the number of performance levels, and set $i = 1$. Where, i is the system performance level under consideration.
2. Set $p_i(t) = \{0\}^n$, $\tau_i = 0$, and $\lambda = 1$. Where, τ_i is the total time spent at performance level i.
3. Modify the contents of $\mathbf{\Psi}_\lambda$ and $\boldsymbol{t}_\lambda$ to include system performances at $t = 0$ and $t = t_m$.
4. Define a binary string, $\mathbf{\Theta} = \{\theta_l\}^j \mid l = 1, 2, ..., j$, such that, $\theta_l = 1$ if $\psi_l = \phi_i$, and 0, otherwise.
5. Set the last element of $\mathbf{\Theta}$ to 0. This ensures the period between the last transition and t_m is accounted for, given the transition was to performance level i.
6. Obtain $\boldsymbol{\sigma}$; the locations of the sub string, '1 – 0' in $\mathbf{\Theta}$. Compute $\boldsymbol{T}_1 = (\boldsymbol{\sigma}, \boldsymbol{t})$, $\boldsymbol{T}_2 = (\boldsymbol{\sigma} + 1, \boldsymbol{t})$, and τ_i, such that, $\tau_i = \tau_i + \sum (\boldsymbol{T}_2 - \boldsymbol{T}_1)$.
7. Deduce ch_1 and ch_2; the number of time-steps the first elements of $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ respectively represent. Increment elements ch_1 to ch_2 of $p_i(t)$ by 1, that is, $(ch_1 \rightarrow ch_2, p_i(t)) = (ch_1 \rightarrow ch_2, p_i(t)) + 1$.
8. Repeat step 7 for the remaining elements of $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$, and set $\lambda = \lambda + 1$.
9. Repeat steps 3 to 8 until $\lambda = N + 1$, compute $p_i = \tau_i/Nt_m$, $p_i(t) = p_i(t)/N$, and set $i = i + 1$.
10. Repeat steps 2 to 9 until $i = k + 1$, and terminate the algorithm.

With the system state probabilities known, $A(t)$ is obtained from Equation 4. Often, only the availability with respect to a set of states or a range of performances is required. In both cases, deriving all the system state probabilities is unnecessary and time consuming. However, the algorithm described above remains applicable, but with minimal modifications. The availability with respect to a given condition is obtained by disregarding step 10 and modifying step 4 to reflect the desired condition (s). For instance, the availability with respect to system performance being between 10 and 20 would be implemented as, $\theta_l = 1$ if $10 \leq \psi_i \leq 20$ in step 4.

4.3. Maintenance Response Inadequacy

When there are as many maintenance teams as repairable nodes, the latter spend negligible time in failed or degraded states before maintenance intervention. This is not the case with limited maintenance teams, as failed nodes would have to wait in the maintenance queue until a maintenance team is available. The probability that at time, t, a node is in the maintenance queue defines the system's maintenance response inadequacy relative to that node. It is a measure of how severe the effect of limited maintenance is on the node's availability and contribution to system performance.

With the basis for increasing the maintenance team size established, a system's maintenance response inadequacies can be used to determine which nodes to prioritize, assuming certain node repairs require specific specialist skills. The maintenance response inadequacy of a node is a right continuous increasing function that approaches a steady-state value with time. It is obtained by adding the instantaneous state probabilities of the node's repairable failed and degraded states. At the system level, the maintenance response inadequacies of all its nodes can be combined, and a bound on the probability of at least one of them being in the maintenance queue obtained. These bounds provide a means of comparing the efficiency of two maintenance teams without explicit reference to system performance. They also indicate when, during the mission a maintenance team size scale-up is actually necessary, an attribute useful to

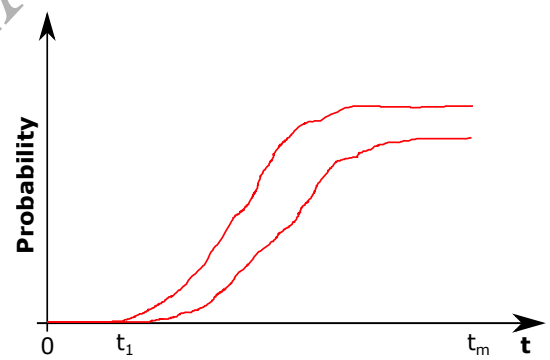


Figure 8: Bounds on maintenance response inadequacy of a sample system

ageing systems. Fig. 8, for instance, suggests maintenance team size scale-up for the sample system is necessary only after $t = t_1$.

Though the maintenance response inadequacy indicates the need to increase the maintenance team size, it does not state its actual effect on system performance. In order to justify this need under cost constraints, its effect on system performance should be established. This entails comparing the performances yielded by the system under zero maintenance response inadequacy, and with the current maintenance team size. Zero maintenance response inadequacy is obtained by setting the number of maintenance teams to infinity. The difference in performance represents the maximum achievable gain from scaling up the maintenance team. Its monetary value can be obtained and compared against the minimum maintenance team scale-up cost, thereby enhancing a robust decision making process.

5. Case Study: An Offshore Oil Installation

Using the system originally presented in [22], we illustrate the application of the proposed approach to interdependent systems and systems prone to limited maintenance teams. Preliminary results were presented at the 26^{th} edition of the *European Safety and Reliability Conference* [31].

5.1. Problem Formulation

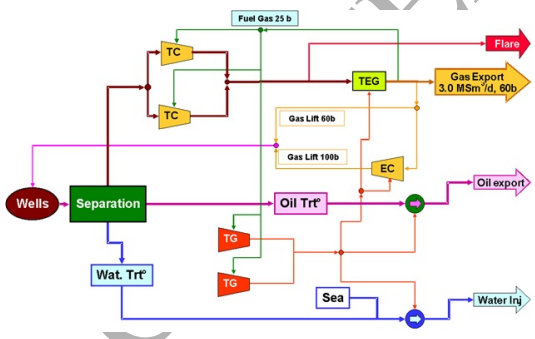
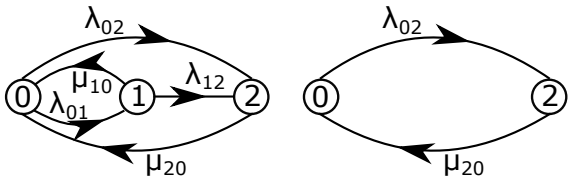


Figure 9: Schematic of offshore installation

Fig. 9 shows the schematic of the offshore installation. Fig. 10 describes the failure and repair transitions of six of its components, the remainder are assumed to be perfectly reliable. The notations in these figures are defined thus, **TG**; Turbo Generator, **TC**; Turbo Compressor, **TEG**; Try-ethylene Glycol Unit, **EC**; Electro-Compressor, λ_{mn}; failure rate from state m to n, and



(a) TCs and TGs (b) TEG and EC

Figure 10: State-space diagrams of components

μ_{mn}; repair rate from state m to n. State 0, in Fig. 10 represents the relevant component in its normal operating mode, and state 1, its partial failure. When partially failed, the component maintains its nominal performance, but with an increased failure probability to state 2, where it is completely failed.

The Well nominally produces 5.0×10^6 units of gas, 26500 units of oil, and 8000 units of water a day. These are separated at the Separation Unit, and transmitted via independent dedicated paths, as shown in Fig. 9. The nominal gas demand is 3×10^6 units at 60bar, and when gas production exceeds demand, for safety reasons, the excess is burnt as flare. Additional details on the offshore plant are available in [22].

Table 1: Component PM Schedule

PM Type	Component	Interval (h)	Mean Duration(h)
1	TC, TG	2160	4
2	EC	2666	113
3	TC, TG	8760	120
4	TC, TG	43800	672

5.1.1. Interdependencies & Reconfiguration

The major components of the plant (TEG, EC, oil pump and water pump) require continuous supply of electricity to function. This reliance creates a functional coupling between the electricity network and the paths transporting the three commodities. A second functional coupling is introduced by the reliance of TCs and TGs on dried compressed gas, for their functioning. Each TG is rated 13MW, the TEG and EC consume 6MW each, while the two pumps consume 7MW each. When only one TG is available, the EC and the water pump are shut down to maintain the production of oil and gas. To ensure nominal production, a fraction of dried compressed gas (1.0×10^6 units) is diverted, compressed by the EC to 100bar, and re-injected into the Well. If the EC is unavailable, the gas is injected

Table 2: Component Repair Priority

Priority	Component	System Condition
1	TEG	-
	TG	other TG unavailable
	TC	other TC unavailable
2	EC	-
	TC	other TC available
3	TG	other TG available

directly into the well at 60bar, resulting in a production at 80% of nominal levels. With no gas injection, production level drops to 60%.

5.1.2. Maintenance Policy

Corrective maintenance (CM) is carried out according to a predefined priority, as expressed in Table 1. Repairs once initiated, remain unaffected by the failure of other components, regardless of their superiority on the priority list. In addition to CM, TCs and TGs undergo three types of preventive maintenance interventions, while the EC undergoes one. To ensure minimal effect on performance, PM is carried out only when the system is perfect. Table 1 outlines the various PM types, their intervals, and mean duration. The latter are assumed to be exponentially distributed, and the former, as the absolute time between successive PM interventions.

5.1.3. Monte Carlo Simulation

In [22], the goal was to determine the production availability of the plant under the maintenance policy described. It was approached by enumerating the plant's production levels, reconstructing the cycle of component failures & maintenance, and monitoring production level occurrences. Identifying the production level corresponding to a given plant configuration during the simulation had required the use of an innovative approach based on cut sets. In practice, each production level is identified by a pair of cut sets defined as minimum and maximum cut sets. Although the solution proposed was very efficient and innovative, it required the manual identification of those cut sets and their corresponding production levels. This, even for a moderately sized system can be quite time-consuming, error-prone, and may become impractical for some systems.

5.2. Solution Procedure

Given the challenges of the Monte Carlo Simulation described in Section 5.1.3, the simulation and modelling approach described in the preceding sections are applied to the plant. The step-by-step modelling procedure is presented here.

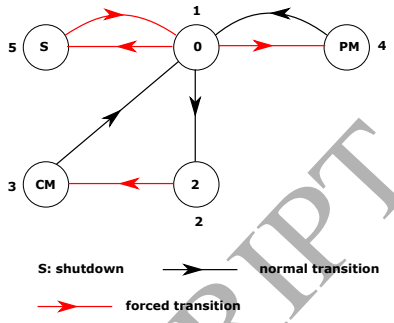


Figure 11: Modified state-space diagram for components EC and TEG

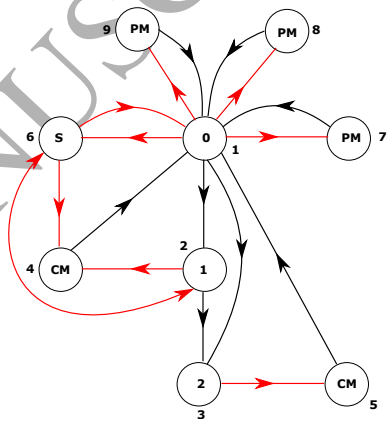


Figure 12: Modified state-space diagram for components TCs and TGs

Figures 11 and 12 are modifications of the state diagrams of the plant's components presented in Fig. 10. The modifications are such that the realistic operation of components, consequent of system dynamics is reflected. The state, *Shutdown* (S), is introduced to account for restart and shut down (reconfiguration of the component). In the plant, PM takes place only when all its components are in their perfect states. This explains why the transition to PM in the modified state diagrams is from state 0. With the exception of state 4 in Fig. 12, a component has 0 capacity when in any of the Shutdown, CM and PM states. State 4 is an exception, since the transition represents a minimal repair, and there is no need to take the component out of operation. Hence, its capacity from state 2 is retained. Transitions to the maintenance states; CM and PM are forced, as they depend on the availability of an idle maintenance team. For instance, a component remains in state 3 indefinitely

until an idle maintenance team commences its repair. Similarly, transitions to *Shutdown* are forced, since they represent an induced unavailability of a component due to the unavailability of another component.

5.2.1. Modelling the Plant

Shown in Fig. 13 is the plant's schematic, with the relevant nodes and their relationships. The Well is separated into three nodes, 1, 2, and 3, each supplying gas, oil, and

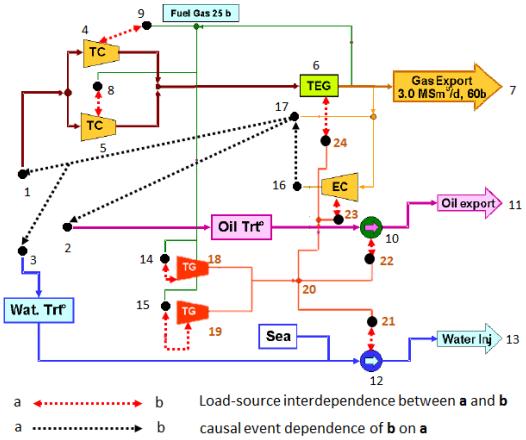


Figure 13: System model showing dependencies

water respectively. Its third production level is unreachable, since there is no gas lift only if the entire system is failed. Therefore, each of the three nodes exists in only two states, 100% (state 1) and 80% (state 2) nominal output. A third state, with capacity 0 is introduced to account for the period when the plant is completely shut down, consequent of either PM or component failure. Transitions between the non-zero output levels are triggered by the EC (node 16), and are, therefore, considered forced transitions. The alternative path for gas lift, node 17, is activated on the unavailability of node 16, and deactivated when available. It, therefore, has a standby relationship with the latter, and exists in two states, *active* (state 1) and *standby* (state 2). Nodes 1, 2, and 3 are affected accordingly on its activation or deactivation, as specified in Equation 5.

$$\mathbf{D}_{16} = \begin{pmatrix} 1 & 17 & 1 & 2 \\ 2 & 17 & 2 & 1 \\ 5 & 17 & 2 & 1 \end{pmatrix}, \quad \mathbf{D}_{17} = \begin{pmatrix} 1 & 1 & 1 & 2 \\ 1 & 2 & 1 & 2 \\ 1 & 3 & 1 & 2 \\ 2 & 1 & 2 & 1 \\ 2 & 2 & 2 & 1 \\ 2 & 3 & 2 & 1 \end{pmatrix} \tag{5}$$

The plant is separated into two subsystems, on the basis of commodity transported. The paths transporting gas, oil, and water are considered a single subsystem (production subsystem), by virtue of their independence. The flare is excess gas, and it's, therefore, discarded, since it has no effect on the gas output. The demands at nodes 7, 11, and 13 are respectively taken as the nominal Well output of gas, oil, and water. The capacities of nodes 8, 9, 14, and 15 are each 0.1×10^6 units of gas, and those of nodes 16 and 17, 1×10^6 units of gas. These nodes, according to the plant's schematic (Fig. 13) appear to be competing with node 7 for the gas output from the TEG. In reality, the quantity of gas required to keep the TGs and TCs in operation and the gas lift are used up first, and any excess exported via node 7. However, this is not considered by the subsystem's network model. Therefore, the gas output (flow through node 7), as deduced from the network model should be corrected. The effective gas is the difference between the gas flow into the TEG, the quantity used for gas lift ($X_{lift} = 1 \times 10^6$), and the gas consumed by any available TCs and TGs.

$$X_{gas} = \mathbf{s}\left(\eta_6 - X_{lift} - \sum_{i \in \mathbf{w}} c_x^{\{i\}}\right) \qquad X_{oil} = \eta_{11}, \qquad X_{water} = \eta_{13} \tag{6}$$

Following flow calculation, the effective outputs, X_{gas}, X_{oil}, and X_{water}, of the three commodities are given by Equation 6. Where, η_i is the flow through node i, $\mathbf{w} = \{8, 9, 14, 15\}$, and $\mathbf{s}$ is an indicator function that takes the value 1 when $\eta_7 > 0$, and 0, otherwise.

Shown in Fig. 14 is the plant's network model, with the maximum flow along each link indicated. Flow along the gas production line is in Mega units, the oil and water lines, in kilo units, and the electricity line, in MW. The electricity network is considered a separate subsystem, as shown in Fig. 14. Nodes 21 to 24 are demand points (local sources) for nodes 12, 10, 16, and 6 from the production subsystem. They, therefore, exist in two states, *active*, when their respective dependent nodes are working, and *inactive*, otherwise. Node 20 is a dummy node, assumed perfectly reliable, and assigned a constant capacity of 26 units, the combined maximum output of the TGs. The minimum threshold flows, Λ_{21} and Λ_{23}, of nodes 21 and 23 are set to 5.99 and 6.99 units respectively, to account for the unavailability of one TG. With only one TG available, the flows through nodes 21 and 23 fall below their threshold, and are shut down, as explained in Section 2.3. This augments the flows through nodes 22 and 24 to their required levels, and

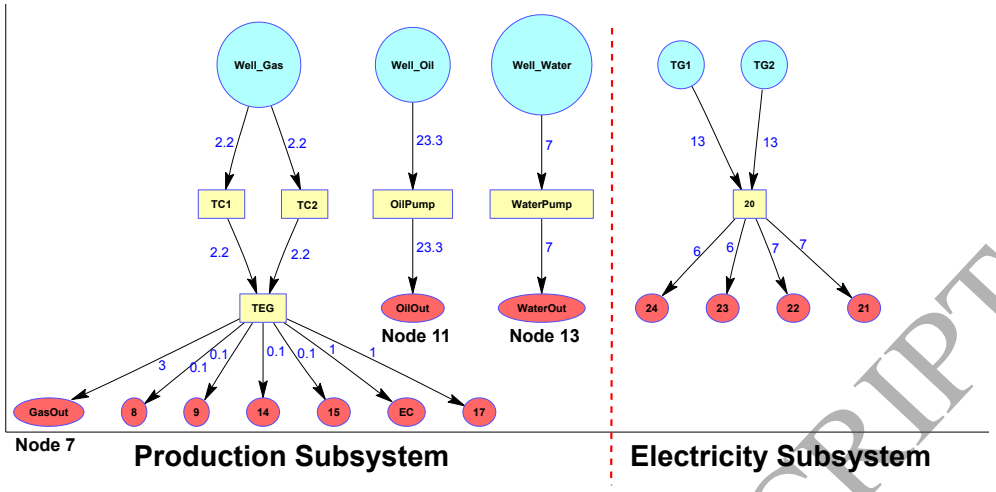


Figure 14: Plant network model

keeps the EC and the oil pump in operation. The demands at the three output nodes are fixed, and the oil and water pumps, as well as node 20, are perfectly reliable. Their reconfiguration (shut down and restart), therefore, is unnecessary. This condition has been implemented by assigning a negative value to their threshold flows. With this manipulation, the shut down requirement due to their effective load is never satisfied, since the actual load flows are non-negative. The remaining nodes are assigned a 0 minimum threshold flow requirement.

5.2.2. *Production Level Determination*

To determine the production availability of the plant, the evolution of X_{gas}, X_{oil}, and X_{water} are recorded as the simulation progresses. At the end of the simulation, the possible performance levels of each commodity are determined from the performance history of its relevant output node. The possible combinations of performance levels of the three commodities are generated, and their occurrences in the simulation history identified, to deduce the possible plant performance levels.

5.3. *Simulation Results*

A Matlab application was developed to model the plant under the following scenarios, CM only by one team (Case 1), CM only by two teams (Case 2), and both CM and PM by two teams; one dedicated to each maintenance type (Case 3). 10^5 Monte Carlo simulations of the plant's performance evolution for a mission time, $t_m = 1000$ hours were used in cases 1 and 2, while 3×10^4 samples and 2×10^5 were used in case 3. A much larger mission time was used in the latter to accommodate several cycles of PM type 4, occurring once every 43800 hours (see Table 1).

Six production levels of gas and three each, of oil and water were identified by the simulation algorithm. These were ordered from lowest to highest, and assigned production level numbers, as shown in Table 3. Their steady-state probabilities are given in Tables 4-6. At the plant level, 7 production levels were identified, as presented in Table 7. The probabilities of the plant residing in any of these levels during a given mission time are presented in Table 8. Fig. 15 shows the instantaneous production of the plant under CM only, for both 1 and 2 maintenance teams. As expected, the plant performs better with two maintenance teams. Overall, its availability at the nominal level improves, albeit slightly (see Table 8). However, both scenarios yield the same performance, within the first 30 to 45 hours of operation. This is explained by the high initial reliability of components, such that there are only a few failures, and can be covered by a single team. As fatigue creeps in, failed components begin to queue and more than one maintenance team is required. It's evident in Table 8 and Tables 4-6 that the overall performance drops with PM. This is attributed to the fact that components exhibit exponential failure characteristics. PM increases their unavailability without improving their reliability [22]. Consequently, the smooth curves in Fig. 15 are replaced by rough curves in Fig. 16, the deep drops in performance being due to PM of critical components (e.g., TEG). A similar trend is portrayed by the plant's instantaneous availability as, shown in Fig. 17.

Table 3: Production levels of individual commodities

Production Level	Commodity		
	Gas ($\times 10^6$)	Oil ($\times 10^3$)	Water ($\times 10^3$)
1	0	0	0
2	0.9	21.2	6.4
3	1	23.3	7
4	2.6		
5	2.7		
6	3		

Table 4: Gas production level probabilities

Production Level	State Probabilities		
	Case 1	Case 2	Case 3
6	9.22×10^{-1}	9.30×10^{-1}	7.78×10^{-1}
5	3.85×10^{-2}	3.60×10^{-2}	8.95×10^{-2}
4	4.80×10^{-3}	4.70×10^{-3}	4.09×10^{-2}
3	2.50×10^{-3}	1.10×10^{-3}	5.90×10^{-3}
2	3.06×10^{-2}	2.76×10^{-2}	8.19×10^{-2}
1	1.90×10^{-3}	7.84×10^{-4}	3.80×10^{-3}

Table 5: Oil production level probabilities

Production Level	State Probabilities		
	Case 1	Case 2	Case 3
3	9.52×10^{-1}	9.57×10^{-1}	8.58×10^{-1}
2	4.62×10^{-2}	4.19×10^{-2}	1.38×10^{-1}
1	1.90×10^{-3}	7.84×10^{-4}	3.84×10^{-3}

Table 6: Water production level probabilities

Production Level	State Probabilities		
	Case 1	Case 2	Case 3
3	9.52×10^{-1}	9.57×10^{-1}	8.58×10^{-1}
2	5.20×10^{-3}	4.80×10^{-3}	4.26×10^{-2}
1	4.29×10^{-2}	3.79×10^{-2}	9.90×10^{-2}

Table 7: Plant production levels identified

Output Type	Production Level						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Gas ($\times 10^6$)	3	0.9	2.7	1	2.6	0.9	0
Oil ($\times 10^3$)	23.3	23.3	21.2	21.2	21.2	21.2	0
Water ($\times 10^3$)	7	7	0	0	6.4	6.4	0

Table 8: Comparison of plant production level probabilities

Production Level	State Probability		
	Case 1	Case 2	Case 3
1	9.22×10^{-1}	9.30×10^{-1}	7.74×10^{-1}
2	2.99×10^{-2}	2.74×10^{-2}	8.03×10^{-2}
3	3.84×10^{-2}	3.60×10^{-2}	8.93×10^{-2}
4	2.50×10^{-3}	1.10×10^{-3}	5.90×10^{-3}
5	4.70×10^{-3}	4.70×10^{-3}	4.09×10^{-2}
6	3.11×10^{-4}	1.43×10^{-4}	1.70×10^{-3}
7	1.90×10^{-3}	7.84×10^{-4}	3.80×10^{-3}

5.3.1. Expected Production

Table 9: Expected annual production

Commodity	Expected Cumulative Output		
	Case 1	Case 2	Case 3
Gas	1.062×10^9	1.069×10^9	1.007×10^9
Oil	8.453×10^6	8.464×10^6	8.366×10^6
Water	2.446×10^6	2.456×10^6	2.292×10^6

A frequently used indicator of performance is the expected cumulative amount of commodity flow through output nodes within a specified period. Using the data in Tables 4 to 6 and the identified production levels of each commodity, their expected annual outputs are as presented in Table 9.

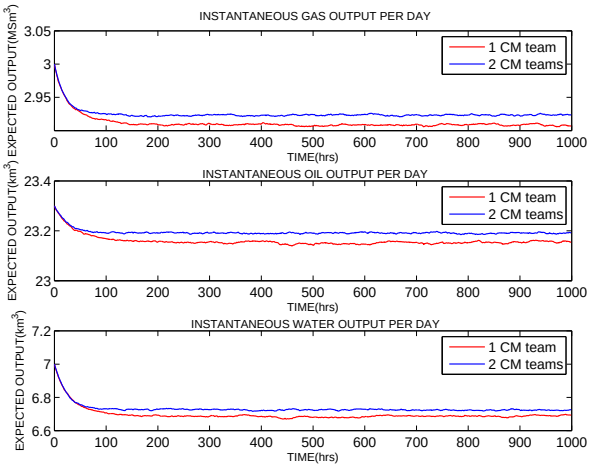


Figure 15: Instantaneous plant performance under CM only

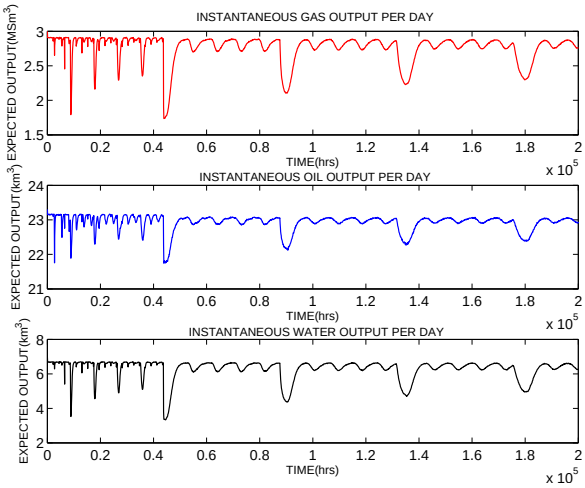
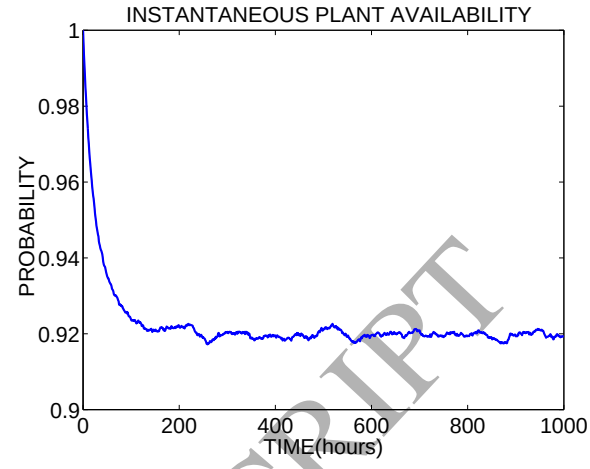
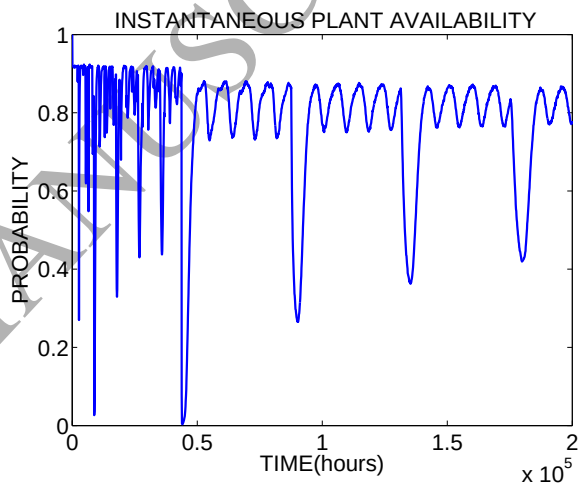


Figure 16: Instantaneous plant performance under CM and PM



(a) CM only by 1 team



(b) CM and PM

Figure 17: Plant availability relative to state 1

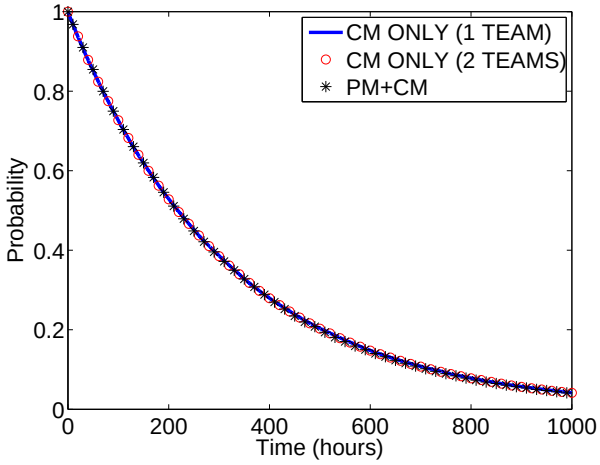
5.3.2. *Reliability and Recovery*

The reliability and recovery of the plant are defined with respect to its nominal production level (state 1). Using the algorithm proposed in Section 4.1, the two quantities were obtained as shown in Fig. 18. The structure and properties of the plant are such that, the unavailability of any component leads to the plant's deviation from nominal performance. Since maintenance comes into play only after component failure, plant reliability is unaffected by the number of maintenance teams. Also, as outlined in Table 1, four types of PM actions are applicable, but the earliest starts after 2160 hours. This implies, plant performance during the first 2160 hours is unaffected by PM. These considerations explain why the same reliability curve was obtained for all the three cases, as Fig. 18a shows.

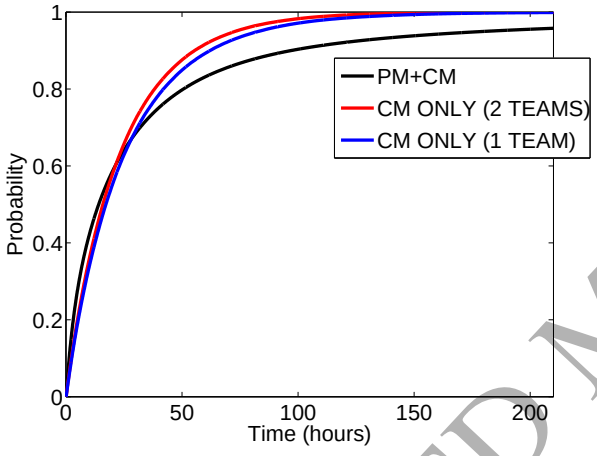
Unsurprisingly, Fig. 18b indicates 2 corrective maintenance teams ensure a higher recovery probability, explained by the increased response to component failures. For recovery within the first 22 hours of deviation from nominal performance, the policy implementing CM and PM gets the upper edge. This is because a significant proportion of these deviations is due to type-1 PM, with a mean duration of only 4 hours. In most instances, however, the PM of some component may be due while another component is under corrective maintenance. Given PM is only carried out when the plant is in its perfect state, the component's PM is deferred until all failed components are repaired. Even though the plant momentarily returns to nominal performance after the last repair, this is not regarded a recovery, as nominal performance is lost instantaneously when the queueing component is shut down for PM. This explains why the recovery probability for case 3 is on average the least.

5.3.3. *Effects of Real-time component ranking*

The plant's production availability was reassessed using real-time component ranking, by maximizing gas production. Though the same outcome yielded by the predefined priority ranking shown in Table 2 was obtained, real-time ranking presented an intuitive alternative with little additional computational burden.



(a) Reliability



(b) Recovery Probability

Figure 18: Plant reliability and recovery probability relative to state 1

5.3.4. *Effects of limited maintenance teams*

The plant's maintenance response inadequacy with respect to each of its six maintainable components was obtained. To enhance this, their state transitions during simulation were collected and saved as a function of time. As shown by their state-space diagram, TCs and TGs can be shut down from state 2 (see Fig. 12), but that does not remove them from the maintenance queue. Since maintenance response inadequacy defines how likely it is to have a component in the maintenance queue, transitions to and from *shutdown* (state 6) were not recorded. Fig. 19 shows the maintenance response inadequacies under one corrective maintenance team.

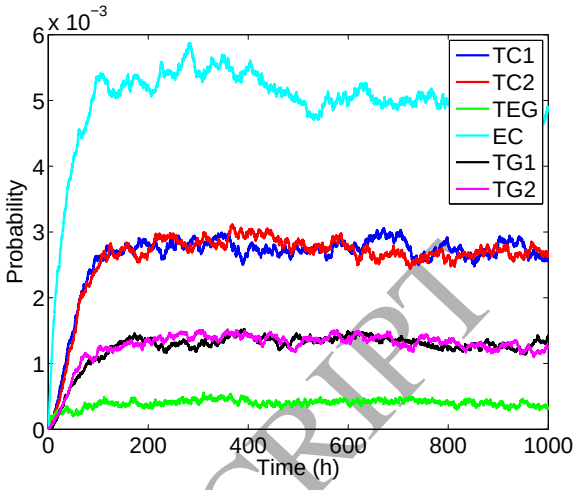


Figure 19: Maintenance response inadequacies for one CM team

Table 10: Maximum gains from maintenance team scale-up

Commodity	Expected Output	Percentage Gain	
		Case 1	Case 2
Gas	1.070×10^9	0.75	0.09
Oil	8.467×10^6	0.17	0.04
Water	2.458×10^6	0.49	0.08

To investigate the effects of limited maintenance on system performance and quantify the possible gains from a maintenance team scale-up, the plant was re-analysed with an unlimited number of maintenance teams. The expected annual production levels were obtained and compared with the values presented in Table 9. The expected output, with unlimited maintenance teams and the possible gains in cases 1 and 2 are given in Table 10. The table reveals, gas production is most effected by limited maintenance, and that case 2 is already close to the optimum number of maintenance teams.

5.4. *Comments and Discussions*

The proposed simulation and modelling approach has allowed us to obtain the same production levels identified via hand calculation by the original authors [22]. In addition, it yielded availability values at the nominal level that are within 5% of the reported values, even though 70% less samples were used in case 3. Using 19 cores on a 1895.257MHz AMD Opteron(tm) 6168 processor, cases 1 and 2 took an average of 10.69 minutes and case 3, a few hours. Cases 1 and 2 required an

additional 2.13 minutes when the plant was re-analysed using real-time component ranking.

To verify the effects of the modifications made to the flow calculation procedure, the case studies presented in [24] were re-analysed on the same computer. Prior calculation of node flows improved the simulation speed by 52.9% in case 1 and 39.73% when applied to an unpublished 14-node system. However, it wasn't a feasible alternative when the 21-node system presented in the second case study was considered. The verification outcome suggests, the smaller the size of matrix $\boldsymbol{\beta}$ (see Section 3), the more advantageous the alternative of calculating and storing node flows prior to simulation. In addition, storage problems may be encountered with large systems, since node flows for all the possible system configurations have to be stored. These constraints make flow calculation during simulation inevitable for large systems, resulting in increased computational burden. The increased burden, however, can be mitigated with access to parallel computing, where the required number of simulation samples is shared across several computers or several workers on a multi-core computer.

The plant under analysis can exist in 437 configurations, considering corrective maintenance alone. Traditional approaches would require matching each of these to a production level [22]. Since this procedure can be time consuming and error prone, Zio et al [22], proposed an innovative approach based on the minimal and maximum cut-sets of each production level. This approach, however, requires considerable human effort and a detailed knowledge of the plant's operational dynamics. It also suffers the set back of not being sufficiently general and intuitive, as a system's cut-sets and performance levels depend on its structure and the properties of its components. Therefore, every system would require a unique approach and a unique degree of difficulty.

Though the approach this paper proposes is computationally more demanding than Zio et al's [22], it does not require the manual identification of production levels and enumeration of system cut sets. All it requires are the definition of inter-component relationships, component properties, and the structure of the system. The rest of the analysis is carried out by efficient algorithms. These attributes, coupled with the fact that it allows system structure to be defined by an adjacency matrix, make it easily applicable to any system structure. Considering the time and human effort involved in the manual identification of production levels, and the possibility of costly errors, the proposed approach is an efficient and credible alternative. Its advantages particularly stand out when applied to complex systems.

6. Conclusions

In this paper, an efficient and powerful simulation tool has been presented for the availability assessment of complex multi-state systems with interdependencies, multi-commodity flows, and limited maintenance teams. Algorithms for quantifying the relevant system availability and performance indices, including a new metric for the inadequacy of maintenance response have also been presented. The proposed simulation approach can implement reconfiguration requirements and derive system performance without reference to the system cut-sets or predefined system performance levels. Traditional approaches, however, would require the manual listing of all the system performance levels and their associated cut-sets, which difficulty increases with system complexity and size. This attribute, therefore, is a key advantage and an illustration of its intuitiveness. Its applicability has been demonstrated by assessing the availability of a multi-commodity offshore plant operated by limited maintenance teams. By only defining the intra and inter component relationships, the approach provided (within an acceptable time frame) an outcome similar to one in literature, without prior knowledge of the plant's production levels or cut sets. This renders it less dependent on human effort, intuitive, robust to human-induced errors, and suitable for any system architecture. It is implemented in the open-source uncertainty quantification tool, OpenCossan [32, 33], and, therefore, readily available to academics and industry.

Acknowledgement

The authors would like to acknowledge the gracious support of this work through the EPSRC and ESRC Centre for Doctoral Training on Quantification and Management of Risk & Uncertainty in Complex Systems & Environments.

References

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (7291) (2010) 1025–1028. URL http://dx.doi.org/10.1038/nature08932

[2] E. Zio, G. Sansavini, Modeling interdependent network systems for identifying cascade-safe operating margins, IEEE Transactions on Reliability 60 (1) (2011) 94–101. doi:10.1109/TR.2010.2104211.

[3] R. Zimmerman, Social implications of infrastructure network interactions, Journal of Urban Technology 8 (3) (2001) 97–119. URL http://dx.doi.org/10.1080/106307301753430764

[4] E. Bompard, C. Gao, R. Napoli, A. Russo, M. Masera, A. Stefanini, Risk assessment of malicious attacks against power systems, IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans 39 (5) (2009) 1074 – 1085. URL http://dx.doi.org/10.1109/TSMCA.2009.2020687

[5] T. Adachi, B. R. Ellingwood, Serviceability of earthquake-damaged water systems: Effects of electrical power availability and power backup systems on system vulnerability, Reliability Engineering & System Safety 93 (1) (2008) 78 – 88. doi:http://dx.doi.org/10.1016/j.ress.2006.10.014.

[6] J. Johansson, H. Hassel, An approach for modelling interdependent infrastructures in the context of vulnerability analysis, Reliability Engineering & System Safety 95 (12) (2010) 1335 – 1344, 19th European Safety and Reliability Conference. doi:http://dx.doi.org/10.1016/j.ress.2010.06.010.

[7] U.S.-Canada Power System Outage Task Force, Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations, Tech. rep. (April, 2004).

[8] E. Zio, Challenges in the vulnerability and risk analysis of critical infrastructures, Reliability Engineering & System Safety 152 (2016) 137 – 150. doi:http://dx.doi.org/10.1016/j.ress.2016.02.009.

[9] H. George-Williams, M. Lee, E. Patelli, A framework for power recovery probability quantification in nuclear power plant station blackout sequences, in: Proceedings of the Probabilistic Safety Assessment and Management Conference, Vol. 13, 2016.

[10] M. Ouyang, Review on modeling and simulation of interdependent critical infrastructure systems, Reliability Engineering & System Safety 121 (2014) 43 – 60. doi:http://dx.doi.org/10.1016/j.ress.2013.06.040.

[11] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. D. Porcellinis, R. Setola, Modelling interdependent infrastructures using interacting dynamical models, International Journal of Critical Infrastructures 4 (1/2) (2008) 63+. doi:10.1504/ijcis.2008.016092. URL http://dx.doi.org/10.1504/ijcis.2008.016092

[12] X. Zang, D. Wang, H. Sun, K. Trivedi, A bdd-based algorithm for analysis of multistate systems with multistate components, Computers, IEEE Transactions on 52 (12) (2003) 1608–1618. doi:10.1109/TC.2003.1252856.

[13] L. Xing, Y. Dai, A new decision-diagram-based method for efficient analysis on multistate systems, Dependable and Secure Computing, IEEE Transactions on 6 (3) (2009) 161–174. doi:10.1109/TDSC.2007.70244.

[14] W.-C. Yeh, An improved sum-of-disjoint-products technique for symbolic multi-state flow network reliability, Reliability, IEEE Transactions on 64 (4) (2015) 1185–1193. doi:10.1109/TR.2015.2452573.

[15] G. Levitin, The Universal Generating Function in Reliability Analysis and Optimization, Springer-Verlag London Limited, 2005.

[16] G. Levitin, A. Lisnianski, Multi-state System Reliability Analysis and Optimization, in: Handbook of Reliability Engineering, Springer, 2003, Ch. 4, pp. 61–90.

[17] A. Lisnianski, I. Frenkel, Y. Ding, Multi-State System Reliability Analysis and Optimization for Engineers and Industrial Managers, Springer-Verlag London Limited, 2010.

[18] G. Levitin, A universal generating function approach for the analysis of multi-state systems with dependent elements, Reliability Engineering & System Safety 84 (3) (2004) 285 – 292. doi:http://dx.doi.org/10.1016/j.ress.2003.12.002.

[19] M. Malhotra, K. S. Trivedi, Dependability modeling using petri-nets, IEEE Transactions on Reliability 44 (3) (1995) 428 – 440. URL http://dx.doi.org/10.1109/24.406578

[20] Z. Liu, Y. Liu, B. Cai, X. Li, X. Tian, Application of petri nets to performance evaluation of subsea blowout preventer system, {ISA} Transactions 54 (2015) 240 – 249. doi:http://dx.doi.org/10.1016/j.isatra.2014.07.003.

[21] H. Langseth, L. Portinale, Bayesian networks in reliability, Reliability Engineering & System Safety 92 (1) (2007) 92 – 108. doi:http://dx.doi.org/10.1016/j.ress.2005.11.037.

[22] E. Zio, P. Baraldi, E. Patelli, Assessment of the availability of an offshore installation by monte carlo simulation, International Journal of Pressure Vessels and Piping 83 (4) (2006) 312 – 320. URL http://dx.doi.org/10.1016/j.ijpvp.2006.02.010

[23] J. E. Ramirez-Marquez, D. W. Coit, A monte-carlo simulation approach for approximating multi-state two-terminal reliability, Reliability Engineering & System Safety 87 (2) (2005) 253 – 264. doi:http://dx.doi.org/10.1016/j.ress.2004.05.002.

[24] H. George-Williams, E. Patelli, A hybrid load flow and event driven simulation approach to multi-state system reliability evaluation, Reliability Engineering & System Safety 152 (2016) 351 – 367. doi:http://dx.doi.org/10.1016/j.ress.2016.04.002.

[25] M. Kojima, S. Mizuno, A. Yoshise, A primal-dual interior point algorithm for linear programming, in: N. Megiddo (Ed.), Progress in Mathematical Programming, Springer New York, 1989, pp. 29–47.

[26] W.-C. Yeh, An improved method for multistate flow network reliability with unreliable nodes and a budget constraint based on path set, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 41 (2) (2011) 350–355.

[27] H. George-Williams, E. Patelli, M. Lee, Reliability & performance analysis of multi-state systems based on analytical load flow considerations, in: Proceedings of the European Safety and Reliability Conference, Vol. 26, 2016.

[28] G. Feng, E. Patelli, M. Beer, F. P. Coolen, Imprecise system reliability and component importance based on survival signature, Reliability Engineering & System Safety 150 (2016) 116 – 125. doi:http://dx.doi.org/10.1016/j.ress.2016.01.019.

[29] W. Kuo, X. Zhu, Some recent advances on importance measures in reliability, IEEE Transactions on Reliability 61 (2) (2012) 344–360. doi:10.1109/TR.2012.2194196.

[30] M. Čepin, Assessment of Power System Reliability: Methods and Applications, Springer, London, 2011, Ch. Distribution and Transmission System Reliability Measures, pp. 215–226.

[31] H. George-Williams, E. Patelli, Efficient availability assessment of reconfigurable complex multi-state systems with interdependencies, in: Proceedings of the European Safety and Reliability Conference, Vol. 26, 2016.

[32] E. Patelli, Handbook of Uncertainty Quantification, Springer International Publishing, 2017, Ch. COSSAN: A Multidisciplinary Software Suite for Uncertainty Quantification and Risk Management, pp. 1–69.

[33] E. Patelli, M. Broggi, M. D. Angelis, M. Beer, Opencossan: An efficient open tool for dealing with epistemic and aleatory uncertainties, in: Vulnerability, Uncertainty, and Risk: Quantification, Mitigation, and Management - Proceedings of the 2nd International Conference on Vulnerability and Risk Analysis and Management, ICVRAM 2014 and the 6th International Symposium on Uncertainty Modeling and Analysis, ISUMA 2014, 2014, pp. 2564 – 2573.
URL http://dx.doi.org/10.1061/9780784413609.258