Algorithm 1 SeedSubTrees(Γ)

where Γ is an ordered set that describes which components can cause which other components to fail

1: **for** *rootC* $\in$ *compSet* **do**
2: **if** (Empty(Γ_{rootC})) **then**
3: continue;
4: **end if**
5: *level* = [ ]; {dynamic array of failed components at subTree's current level}
6: *nFailed* = $(0, 0, \ldots, 0)$; {counts failed components of each type}
7: *BFHist* = $(( ), ( ), \ldots, ( ))$; {an array of linked lists that keeps a breadth-first history of subTrees, array is indexed by component type, linked list for each component type stores parents in breadth-first order}
8: add *rootC* to *level*;
9: *nFailed*[*rootC*] = 1;
10: add @ to *BFHist*[*rootC*]; {signifies one component of type rootC has failed}
11: AddSubTreeLevel(*level*, *nFailed*, *BFHist*, 1, *rootC*);
12: **end for**

Algorithm 2 AddSubTreeLevel(*level*, *nFailed*, *BFHist*, *subTreeRate*, *rootC*)

where *level* describes failed components,
nFailed counts failed components by type,
BFHist is Breadth First History,
subTreeRate is a cumulative probability of comps that failed,
rootC is the root component of the current subtree

1: $nextLevelPossibilities = \bigtimes_{i=1}^{|level|} \mathcal{P}(\Gamma_{level[i]})$;
{Builds set of all possible nodes in next level as Cartesian product of powersets of Γ's}
2: **for** *oneNextLevelPossibility* ∈ *nextLevelPossibilities* **do**
3: *addedChildFlag* = False;
4: **for** *parentC* ∈ *level* **do**
5: **for** *childC* ∈ $\Gamma_{parentC}$ **do**
6: **if** *childC* ∈ *oneNextLevelPossibility* **then**
7: **if** *nFailed*[*childC*] == Redundancy(*childC*) **then**
8: goto line 3; {invalid subtree, requires more comps than available in system}
9: **end if**
10: *addedChildFlag* = True;
11: *nFailed*[*childC*] = *nFailed*[*childC*] + 1;
12: add @ to *BFHist*[*childC*]; {signifies one component of type childC has failed}
13: *subTreeRate* = *subTreeRate* ∗ $\phi_{parentC,\ childC}$;
{update rate with ϕ}
14: **else**
15: add *parentC* to *BFHist*[*childC*]; {signifies one component of type childC has not failed, but was present in $\Gamma_{parentC}$}
16: **end if**
17: **end for**
18: **end for**
19: **if** *addedChildFlag* **then**
20: AddSubTreeLevel(*oneNextLevelPossibility*, *nFailed*, *BFHist*, *subTreeRate*, *rootC*);
{subTree can be grown further}
21: **else**
22: ComputeTreeRates(*nFailed*, *BFHist*, *subTreeRate*, *rootC*);
{current subTree is completed because it cannot be grown further}
23: **end if**
24: **end for**

Algorithm 3 ComputeTreeRates(*nFailed*, *BFHist*, *subTreeRate*, *rootC*)

where *level* describes failed components,
nFailed counts failed components by type,
BFHist is Breadth First History,
subTreeRate is a cumulative probability of comps that failed,
rootC is the root component of the current subtree

1: **for** $x' \in S'$ **do**
2: *prodNotFailedProb* = 1; {cumulative probability of comps that could have failed but did not}
3: **for** $comp \in compSet$ **do**
4: $compsAvailable = \text{Redundancy}(comp) - x[comp]$;
5: **for** $parentC \in BFHist[comp]$ **do**
6: **if** $parentC == @$ **then**
7: $compsAvailable = compsAvailable - 1$;
8: **else if** $compsAvailable > 0$ **then**
9: $prodNotFailedProb = prodNotFailedProb * (1 - \phi_{parentC,\, comp})$;
10: **end if**
11: **end for**
12: **end for**
13: **for** $e \in envSet$ **do**
14: Initialize y as a state with no components failed and environment e;
15: **for** $comp \in compSet$ **do**
16: $y[comp] = x[comp] + nFailed[comp]$;
17: **end for**
18: **if** y is not a valid state **then**
19: continue;
20: **end if**
21: $rootFailureRate = (\text{Redundancy}(rootC) - x[rootC]) * \lambda_{rootC,\, e}$;
22: **end for**
23: $Q(x, y) = Q(x, y) + rootFailureRate * subTreeRate * prodNotFailedProb$;
24: **end for**