Supervised Dynamic Probabilistic Risk Assessment: Review and Comparison of Methods

Renan G. Maidana[a,*], Tarannom Parhizkar[b], Alojz Gomola[a], Ingrid B. Utne[a], Ali Mosleh[b]

[a] *Centre for Autonomous Marine Operations and Systems, Department of Marine Technology, NTNU Trondheim, Norway*
[b] *B. John Garrick Institute for the Risk Sciences, University of California, Los Angeles Los Angeles, United States of America*

Abstract

With the adoption of autonomous systems in higher levels of autonomy, large-scale, complex and dynamic systems are becoming commonplace. Ensuring safe operation of safety-critical autonomous systems is paramount, typically approached through risk assessment. Two challenges associated with using traditional risk assessment methods for complex systems are that these systems are dynamic (i.e., their state changes over time) and interactions between subsystems and components may lead to unpredictable behaviors and impact on the surrounding environment and other systems in the close vicinity. Dynamic probabilistic risk assessment (DPRA) methods are possible solutions to these challenges, where the dynamic and uncertain nature of the systems is considered. The methods, however, usually face combinatorial explosion related to hazards and scenarios, which make their practical application prohibitive; in the DPRA literature, this problem is known as the state explosion problem. In this paper, we present a literature review on methods for DPRA, with focus on the existing solutions to the state explosion problem. Specifically, we analyze and compare these solutions in terms of computational time complexity, traceability and state-space coverage. Finally, we discuss the comparisons and propose potential paths to improved solutions for the state explosion problem based on the knowledge gained in the study.

Keywords: Dynamic probabilistic risk assessment, state explosion, supervised DPRA, literature review.

1. Introduction

Probabilistic Risk Assessment (PRA) can be used to identify possible accident scenarios for complex systems, such as Nuclear Power Plants (NPP) and Maritime Autonomous Surface Ships (MASS), and allow for the quantification of their probability of occurrence. Risk scenarios, in this context, are sequences of hazardous events (e.g., electrical motor failure in MASS) which under some circumstances may lead to system-level undesired consequences (e.g., collision with another ship). When a risk scenario ends in an undesired consequence, it becomes an accident scenario.

In the area of autonomous systems, such as autonomous vehicles for urban mobility [1] and MASS [2], the purpose of risk assessment is to contribute to safe operation. Autonomous systems can be classified by their level of autonomy [3, 4] from one to four, where the last two operate in semi and highly autonomous modes respectively. Semi and highly autonomous systems are capable of reasoning and decision-making - the former can make simple and mission-specific decisions, where a human supervisor is responsible for complex decision-making. Highly autonomous systems are able to plan, re-plan and carry out decisions independently from human operators. In both cases, these systems are dynamic, i.e., their state changes rapidly relative to the system's operational time frame, and often operate in

*Corresponding author
Email addresses: renan.g.maidana@ntnu.no (Renan G. Maidana), tparhizkar@ucla.edu (Tarannom Parhizkar), alojz.gomola@ntnu.no (Alojz Gomola), ingrid.b.utne@ntnu.no (Ingrid B. Utne), mosleh@ucla.edu (Ali Mosleh)

uncertain and uncontrolled environments. The traditional PRAs, such as those performed in the nuclear industry, treat the dynamic aspects of the subject systems and risk scenarios in a highly simplified and often implicit way.

Other risk assessment challenges of complex systems are related to unexpected component interactions and couplings. Typically it is hard to predict how failures between components will interact and how fast their effects propagate to other parts of the system. These unforeseen interactions may lead to emergent behaviors which may result in undesired consequences and accidents.

Dynamic PRA (DPRA) is a class of methods for risk assessment of dynamic systems where a system's behavior evolves in time according to dynamic models, with variations in operating conditions, process variables, operator responses, among others [5]. DPRAs therefore aim to better capture the time-dependent interactions, interconnections, and interdependencies between different constituent parts of a large-scale complex system. DPRA methods have been proposed for the risk assessment in the nuclear power [6, 7, 8, 9], oil and gas [10], aerospace [11], and maritime [12] domains. These methods can aid design, operation, and personnel training - thus, there is a significant research effort in improving DPRA methods.

According to Aldemir [13] and Mosleh [5], we can classify DPRA methods by their time domains: Continuous or discrete-time methods. Examples of continuous-time methods are Continuous Event Trees (CET) [14] and the Continuous Cell-to-Cell Mapping Technique (CCCMT) [15]. Discrete-time methods are the more popular of the two, with examples being methods based on Discrete Dynamic Event Trees (DDET) [6] and Monte Carlo Simulation [7]. One of the challenges in using continuous methods is to define a proper representation for the time evolution of the dynamical system, e.g., with a mathematical model. However, there is a trade-off between the model's accuracy and complexity. If the model is too simplistic, it may not capture dynamic behaviors that lead to relevant risk scenarios. If it is too complex, the number of behaviors explodes, and the model may become unsolvable in feasible time - i.e., finding all unique solutions would take a long computation time.

This challenge is also present in discrete methods, where the number of discrete state transitions to be computed grows exponentially with the number of time steps and complexity of the analyzed system. To illustrate, consider a simple system with three components, each with a binary state (i.e., "working" or "failed"). For this simple system, we have to consider all combinations of component modes for each discrete time step to find all possible accident scenarios - i.e., in a single time step, there are 2^3 possible combinations to be considered. Furthermore, failures are stochastic in time, meaning there is a set of s discrete time steps for each component combination. Thus, for a given discrete time t, there are m^{cst} combinations to be considered, where c is the number of system components and m is the number of operating states for each component. This exponential growth is known in the DPRA literature as the *state explosion problem*, and it makes the use of DPRA methods in practical applications unfeasible.

Over the years, several solutions have been proposed to overcome the state explosion problem. Solutions for discrete-time methods typically use biases to guide the generation of risk scenarios towards the most probable ones [16, 17], to reason backward from top events [18, 19], or to limit the growth of the state-space being explored (i.e., to prune the state-space) [8, 20]. While biasing [21, 22] and pruning [23, 24] are also used on continuous-time methods, other solutions include using reduced-order models [25, 26]. Parhizkar et al. [27] use the term *Supervised DPRA* to refer to DPRA methods with a solution to the state explosion problem. Supervised DPRA thus are methods which address the state explosion problem, for example, through biasing, reduced-order models, and so on.

The computational performance of DPRA methods is typically measured in terms of execution time - i.e., how fast a method generates all the relevant risk scenarios. A caveat to this measurement, however, is that it depends on the computing capabilities of the platform where a DPRA method runs. Comparing the performance of two DPRA methods with execution time is thus difficult, as the computers they run on must have the same capabilities and execution environments for the comparison to hold. Computational complexity analysis yields a metric for measuring an algorithm's performance independent of the hardware used, and is thus useful here for comparing the performance of DPRA methods. Furthermore, it allows us to evaluate if a method will remain computationally efficient when used with large-scale systems with an increasing number of components and interconnections.

Computational complexity refers to an algorithm's performance in terms of the number of operations it must perform given an increasing input - i.e., how the number of operations grows as a function of the input size [28]. To illustrate computational complexity, consider a Tower of Hanoi puzzle [29]. An algorithm can be defined to solve this puzzle generically, whose input is the number of disks N that must be sorted. This algorithm would take $2^N - 1$ moves to complete the puzzle, which equates to an exponential complexity - i.e., the number of operations grows exponentially with the growth of N. Computational complexity analysis, therefore, yields the growth rate for the

number of operations an algorithm must perform given variable input size, not affected by differences in hardware and computing power. The computational complexity for the generic algorithm above, for example, can then be compared to other algorithms. If another algorithm can solve the puzzle in less than $2^N - 1$ moves, then this second algorithm is less computationally complex than the first - i.e., it can solve puzzles with more disks in the same amount of time. Computational complexity also indicates how *scalable* is a given algorithm - i.e., how well it performs as the input size grows. An algorithm with linear complexity is more scalable than one with exponential complexity - i.e., the second algorithm above is more scalable than the first, as it is able to complete more operations in the same time.

The state explosion problem poses a challenge to the feasibility of using DPRA methods to autonomous systems with higher levels of autonomy. Initially, the main objective of the review was to define the state-of-the-art in DPRA applied to such autonomous systems. However, there were few works found in this specific intersection - for example, [12], [19] and [21]. The scope of the review was, therefore, widened to include other application domains. The main objective of this study is thus to review the DPRA literature on current solutions to the state explosion problem and analyze and compare their computational complexity relative to each other. The purpose is to determine which current solutions are most feasible for future applications to semi and highly autonomous systems, and investigate possible ways to solve the state explosion problem. Previously, reviews and surveys of the DPRA literature have been performed, e.g., Aldemir [13] surveyed the DPRA literature and classified the methods by their time domains. There is no study, however, analyzing or comparing the computational complexity of supervised DPRA methods to the best of our knowledge.

In this paper, a systematic mapping of the DPRA literature has been performed to find the supervised DPRA methods. The literature review was performed using the *snowballing* method [30], resulting in 131 papers after three iterations. These papers were reviewed and classified as supervised DPRA (52 papers) and *Classical DPRA* (79 papers). Computational complexity analyses were performed for each supervised DPRA paper, to study the growth rate in the number of operations of each method. The supervised DPRA methods have then been compared in terms of computational time complexity, whether the risk scenarios generated are *traceable* (i.e., if consequences are reachable from an initiating event), and whether the full state-space of risk scenarios is covered. Hence, the contributions of this paper are a systematic review of the DPRA literature, a classification of DPRA methods as either classical or supervised, and a comparison of the computational complexity of the supervised DPRA methods.

This paper is organized as follows: Section 2 presents the snowballing methodology used to review the literature systematically. Section 3 presents and discusses the classical DPRA methods, while Section 4 presents the supervised DPRA methods. The performance comparison of supervised DPRA is then shown in Section 5. Discussions on the review and comparison, and possibilities for new solutions for the state explosion problem based on the knowledge gained from the literature review, are presented in Section 6. Finally, conclusions are presented in Section 7.

2. Literature Review Methodology

A systematic literature review was performed following the snowballing procedure as proposed by Wohlin [30]. The procedure starts with the definition of research questions, from which keywords are obtained, and a search string is constructed. In this case, the research question is: What solutions to DPRA's state explosion currently exist in the literature?

From the research question, keywords are obtained, and a search string is constructed. Using the search string on several academic databases, an initial set of 10 papers was obtained, on which the iterative *forward* and *backward* snowballing process was performed. After three iterations, 121 papers were included in the review - thus, 131 papers in total were considered in the review. As the works are not typically identified as "supervised," data extraction was performed for all papers, which were distinguished according to two categories:

- Classical DPRA: Works on the development or application of DPRA methods, but which are not explicitly concerned with the state explosion;
- Supervised DPRA: Works on DPRA methods that directly address the state explosion problem.

The supervised DPRA works were then further studied to analyze the computational time complexity of their methods and algorithms.

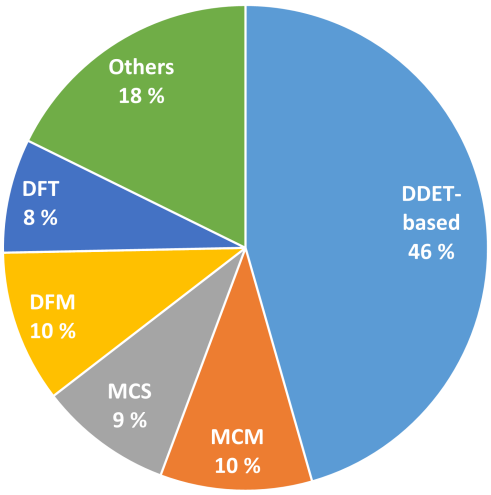

Figure 1: Percentage distribution for classical DPRA approaches. Groups with percentages ≤ 5% (e.g., petri nets, CET) are inserted into the "Others" category for clarity.

3. Classical Dynamic Probabilistic Risk Assessment

DPRA methodologies were primarily developed for providing a more realistic and complete representation of complex systems' responses to critical deviations in operation. Even systems of similar design could be unique, with different operators, site-specific components, and changing environmental conditions in which they are deployed. The case for using DPRA is thus to account for these uncertainties, in order to identify and prevent potential risk or accident scenarios not previously considered.

Generally, DPRA methods consist of model-based simulations that generate system trajectories (i.e., risk scenarios) and their associated probabilities of occurrence [5]. In these methods, rules of deterministic and stochastic behaviors for the complex system and its elements - e.g., hardware, software, human operators, process variables, environmental conditions - are developed and implemented. DPRA methodologies have been proven particularly powerful for systems with control loops and complex interactions between elements. They provide a natural probabilistic environment to include physical models of system behavior and mechanistic models of materials or hardware systems to predict failure and natural hazards.

In this work, a comprehensive review of DPRA methodologies is performed to find the supervised DPRA methods and analyze and compare the methods' computational complexities. Out of 131 studies reviewed, 79 of them were identified as Classical DPRA, which are summarized in Table A.1 in Appendix A. The table is divided according to the primary approach used for obtaining the risk scenarios (e.g., DDET, Monte Carlo simulation). Figure 1 shows the percentage distribution of each approach. Brief descriptions of the approaches are presented here, alongside some randomly chosen example works.

3.1. Dynamic Event Trees

The discrete dynamic event tree approach is the most commonly used DPRA method, as seen in Figure 1, as it is possible to use them in combination with conventional PRA. Most DDET-based methods work with simulation. As seen in Figure 2, a dynamic event tree is generated by simulating the dynamic system's states from an initial condition in discrete time steps while considering possible deviations such as component failures and external disturbances, for example, as well as operator responses. These deviations are predefined to happen at discrete time-steps, and the responses come from the rules and procedures the operators must carry out when the deviations are detected. The deviations and responses have associated probabilities (e.g., component failure probability or probability of successful intervention). The simulation then splits into branches, each with the different outcomes of these probabilities, leading to the tree-like nature of the DDET. The simulation continues in a branch until an undesirable consequence is reached,

e.g., a core damage in NPPs or collision in MASS, resulting in a tree of event sequences where each sequence leads from an initial condition to an undesirable consequence.

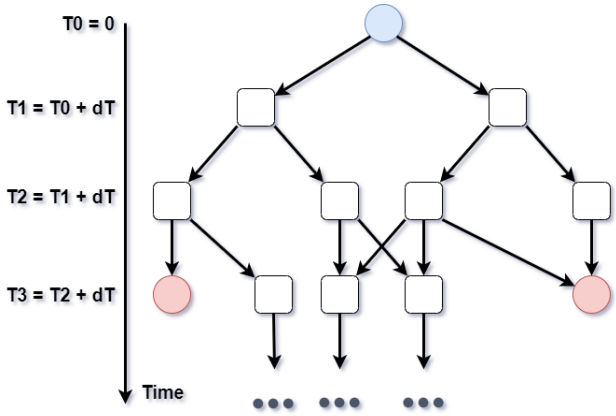

Figure 2: Example of a DDET, where a dynamic system is simulated in discrete time steps (dT) from an initial condition (blue circle). The simulation branches into different paths at branching points (white squares), and each branch is explored until an undesirable consequence (red circles) is reached.

The generated dynamic event tree is closer to a conventional event tree diagram, as the branches occur at predefined discrete times. Therefore, integrating DDETs into conventional PRA methodologies is more straightforward than CETs. Some examples of DDET-based DPRA are [31] and [32]. The first work describes a tool for DDET-based DPRA based on the Dynamic Logical Analytical Methodology (DYLAM), called DYLAM-3. The tool works by simulating the physical system's evolution with a mathematical model, where the system components are modeled in terms of states (e.g., nominal, failed on, failed off, stuck, and so on). DYLAM-3's main program drives the simulation according to a timeline and the logical state transition of the components, and the tool is applied to the PRA of a Boeing 747 executing an approach-to-landing procedure. The second work includes human factors influencing the behavior of the physical system, in this case, an NPP, based on the Dynamic Event Tree Analysis Method (DETAM) [33, 32]. DETAM treats stochastic variations in crew and hardware states, respectively, while plant process variables are treated as deterministic. The deterministic evolution of plant variables and stochastic variations generates the DDET, whose branches are quantified in terms of probability of occurrence.

Continuous Event Trees (CETs) are implicit representations of a dynamic system's state trajectories in time, where the system is subject to stochastic changes due to component failures, for example [34]. More specifically, a set of partial differential equations which describe the system's behavior is augmented with the states of the system's components. A probability density function can be defined for the probability that a component change will cause the system to assume a different behavior - a component's state change causes a transition to another partial differential equation. Stochastic changes in the components' states cause transitions between the possible trajectories, resulting in a continuous event tree.

The problem of DPRA with CETs consists in solving for the probability densities, typically done with grid methods such as the Continuous Cell-to-Cell Mapping Technique (CCCMT), described and exemplified by Tombuyses and Aldemir in [15]. The technique defines regions of the system's state-space evolution, called "cells," and then maps the probability that the system will transition between cells with the system equations and a given time interval. These cell-to-cell transitions constitute the transition matrix of a Markov chain, which allows us to find the probability that the system will be within a cell at a given time [15]. The authors in [15] employ CCCMT and its discrete counterpart to a van der Pol oscillator's case study and compare the execution times for both methods.

3.2. Markov Chain Models

Markov models are comprehensive representations of possible chains of events in a complex system, corresponding to sequences of failures and repairs. Markov models evaluate the probability of being in a state at a given time,

the amount of time the system is expected to spend in a given state, and the expected number of transitions between states.

Aldemir [35] presented a methodology for developing Markov chain models for process control systems - systems with control loops and dynamic variables. The system's feedback equations, describing the interactions between dynamic variables, and the control units' failure and repair rates are used to build a Markov chain model of the system's probabilistic behavior. The methodology is applied and demonstrated in a simple level-control system.

More recently, Cicotti and Coronato developed a DPRA model for "ambient intelligence healthcare systems" [36]. The "ambient intelligence" is a software-based system capable of supporting medical activities and procedures in a highly-regulated and complex healthcare environment. The risk model is based on a Markov decision process (MDP) that considers context-awareness and personalization. The application of the proposed model to a department of nuclear medicine is presented, and results are discussed.

3.3. Monte Carlo Simulation

Monte Carlo Simulation (MCS) methods deal with uncertainty by simulating the system's trajectories while injecting random variations in its states and operating parameters. Uncertainties in a dynamic system may lead to deviations resulting in accident scenarios, and thus the application of MCS to perform DPRA is intuitive. For example, a backward MCS approach has been used to solve the integral equation of a CET and obtain its probability density function [37]. Whereas the simulation starts at an initial condition in forward MCS, backward MCS defines a transition kernel that leads from an end condition to the initial condition. The authors justify this approach by arguing that the domain of possible consequences is smaller than the domain of initial conditions and possible failures. Thus, a backward search is more efficient.

Furthermore, MCS has been compared to DET-based approaches. In [38], the authors look at NASA's Cassini mission PRA. This PRA was performed using a combination of Monte Carlo and event tree approaches, whose results are validated with a DDET-based method. More specifically, a version of the Accident Dynamic Simulator (ADS), called ADS-III, was used. The authors concluded that Monte Carlo and DDET-based approaches led to similar results and were feasible for the PRA of industrial systems, as Cassini's Titan IV was comparable to industrial systems at the time.

3.4. Dynamic Flowgraph Methodology

The Dynamic flowgraph methodology (DFM) presents a system's logic regarding the causal relationships between physical variables of the control systems. The dynamic behavior of complex systems is represented in DFM as a series of discrete state transitions.

DFM has been used for example in PRA of mission-critical software-intensive systems [39], with software modeled with typical PRA methods, integrated with DFM. The methodology was also applied to a mini spacecraft, called the Mini AERCam system, designed and developed by the NASA Johnson Space Center.

3.5. Dynamic Fault Trees

Dynamic fault tree (DFT) utilizes dynamic gates and traditional fault tree gates to model the dynamic behavior and inter-dependency of components of a complex system.

Galileo is a DFT modeling and analysis tool [40, 41]. It uses the *Dynamic Innovative Fault Tree* (DIFtree) methodology, combining static and dynamic fault trees. More specifically, a dynamic fault tree is decomposed into sub-trees which can be static or dynamic, depending on the temporal relationships between basic events. Static sub-trees are solved with binary decision diagrams, while dynamic trees are solved with Markov methods. The sub-trees are then combined for an exact solution to the initial dynamic fault tree.

Another example is by Dugan et al. in 2013, where a patent for DPRA methodology was presented [42]. The methodology presented, called DEFT, allows DFT nodes to be considered as pivot nodes in event tree models. The proposed mathematical method supports all typical functions of PRA and includes modularization, phased mission analysis, sequence dependencies, and imperfect coverage abilities.

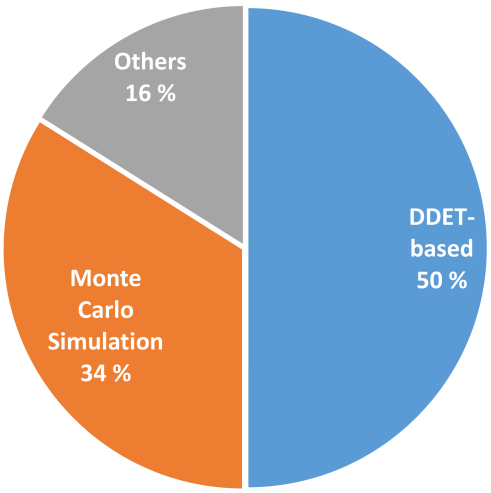

Figure 3: Percentage distribution for supervised DPRA approaches.

4. Supervised Dynamic Probabilistic Risk Assessment

As discussed in Section 1, the state explosion problem prohibits the application of classical DPRA methods to large-scale complex systems. Methods and approaches that address this problem are classified as supervised DPRA, corresponding to 52 of the 131 works reviewed here. The supervised DPRA works are presented in Table A.2 in Appendix A, and the percentile distribution of methods is presented in Figure 3. For brevity, the tables also present the computational complexities of all supervised DPRA methods reviewed and their properties, namely:

- **Traceability:** If the sequences of events or system trajectories can be fully traced from an initial condition to end-states (i.e., accident events, undesirable consequences).
- **Full state-space coverage:** If the solutions to the state explosion problem fully explore the state-space of possibilities in their given application.

The computational complexities are discussed in Section 5. In this Section, we briefly describe the supervised DPRA approaches and provide examples - however, we focus on the solutions to the state explosion problem instead of the full DPRA methods. The underlying DPRA methods are mostly the same as the classical case, namely DDET-based methods and Monte Carlo simulation.

4.1. Guided Exploration, Biasing and Importance Sampling

Guided exploration is an umbrella term used here to group several similar approaches applied to DDET generation, including rule-based guidance, biasing, and heuristic search - i.e., a DDET will branch according to a set of rules, biases, or heuristic functions. The objective is to explore the most relevant risk scenarios first (i.e., the ones with the highest probability or severity). The simulation can then stop earlier, thus alleviating the high execution time brought by the state explosion. An example of this is the ADS framework [6], where branching rules define which scenarios should be explored first. Newer versions of the ADS have included other mechanisms for alleviating the state explosion, such as probability truncation thresholds and simulation time limits - methods of state-space pruning which will be discussed further on.

In Monte Carlo simulation, biasing and importance sampling are used to guide the random walks, which generate the random system trajectories of MCS. The biasing approach consists of nudging the simulation towards a set of desired scenarios - i.e., influencing the random walk of MCS to simulate the most severe or probable scenarios first. Importance sampling is a particular case of biasing. In MCS-based DPRA, importance sampling forces the most critical scenarios to be simulated first by heavily biasing the conditions that lead to these scenarios. These biases are then accounted for when quantifying the scenarios' probabilities of occurrence. An example of biasing and

importance sampling in MCS is in [43], where MCS is used to solve a continuous event tree numerically - i.e., to find the probability densities as discussed in 3.1. The authors' objective is to save computational time while finding rare-event accident scenarios - this is achieved by modifying the probabilistic laws of the simulation to ensure that "very rare events" occur. Then, "statistical weights" are used to compensate for the effects of the modified probabilities, ensuring that "unbiased" rare event trajectories are obtained - the definition of importance sampling discussed above.

4.2. State-space Pruning

State-space pruning consists of reducing the size of the state-space to be searched - for example, pruning is often used in graph search, where edges of the graph are removed according to some criteria [44]. In DPRA, state-space pruning typically consists mainly of truncation using probability thresholds or biases to limit the extent of branch exploration in DDETs [45, 46, 47, 24], or using predefined knowledge of the state-space to avoid re-evaluating similar scenarios [21]. In the case of probability pruning, for example, events whose probabilities of occurring are lower than a certain threshold are not explored further, thus reducing the size of the explored state-space.

The branch-and-bound optimization method is another common approach for state-space pruning. This algorithm relies on the bounding principle: A lower and an upper bound on an optimal value is computed over a given state-space region. DPRA models could use this method to perform supervised branching and generate scenarios, as the branch-and-bound method carries out branching adaptively. Nielsen and Hakobyan [8, 20] apply the branch-and-bound approach to DPRA, where a DDET for a station blackout transient is generated as in ADS. The branch-and-bound method defines which branches of the DDET are explored and pruned according to the upper and lower bounds and an objective function. The bounds are defined by a novel metric called LENDIT[1].

4.3. Reduced-Order Models

In simulation-based approaches to DPRA (e.g., ADS, MCS), an intuitive way of mitigating the state explosion problem is to adjust the level of detail of the physical models being simulated - a less detailed physical model yields faster simulation times, albeit at the cost of accuracy.

These simplified or Reduced-Order Models (ROM) are prevalent in MCS-based DPRA [48], and can be obtained analytically or through linear and non-linear approximators such as Artificial Neural Networks (ANN) [25, 26, 49]. For example, an ANN is trained in [25] with the solutions to a system of differential equations for a holdup tank, similar to a regression or curve fitting approach.

4.4. Hybrid Approaches

As the name suggests, hybrid approaches mix parts of two or more methods to create a new method. In DPRA, the objectives of hybrid approaches are typically to overcome the limitations of single approaches and achieve better performance.

For example, the Monte Carlo Simulation with the Discrete Dynamic Event Tree (MCDET) is a probabilistic simulation method proposed in [7]. MCDET integrates Monte Carlo simulation and DDET generation, where DDET treats discrete variables and continuous variables are treated by MCS. For each set of continuous values treated by the Monte Carlo simulation, MCDET generates a new DDET. In other words, MCDET uses Monte Carlo simulation to account for stochastic uncertainties and continuous variables and DDET for deterministic uncertainties and discrete variables.

5. Computational Complexity of Supervised DPRA Methods

Works addressing the state explosion problem typically evaluate performance, compared to their classical DPRA counterparts, in terms of execution time (also referred to as CPU time). While this metric is helpful as empirical data, it depends significantly on the hardware used, the programming language used to implement the studied method, the operating system, and many other factors. Asymptotic computational complexity analysis [28] is an analytical way of quantifying computational performance without these dependencies. It works by analyzing the underlying algorithm

[1] Length, Energy, Number, Distribution, Information and Time

for a given program and defining functions of the growth rate of the number of computer operations with increasing input sizes. There are three possible approaches when analyzing an algorithm [28]: The best-case, average-case, and worst-case approaches, each defining a different performance bound for the algorithm. The worst-case approach analyzes the algorithm while imagining a use-case where the algorithm would have the worst possible performance. For example, the worst-case analysis for an algorithm searching an array would be if the element to be found was in the last possible position of the array. This approach is the most common one used, as it defines an upper bound limit on algorithmic performance. That is, in the real world, the algorithm will always perform better than its worst-case complexity may indicate.

Asymptotic computational complexity analyses as described above were performed here for the reviewed supervised DPRA methods, considering the worst-case approach, as shown in Table A.2. The objective is to compare the supervised DPRA methods to each other and to the baseline computational complexity of DPRA, which is exponential [50]. Furthermore, we compare the methods in traceability (i.e., if the undesirable consequences are reachable from the initial conditions) and if they cover the full state-space of possibilities (i.e., a complete search of the state-space).

5.1. Exponential Complexity

As seen in Table A.2, the computational complexity for most supervised DPRA methods is exponential in worst-case analysis. The reason is that most solutions to the state explosion problem are not useful for reducing the number of operations when considering the worst-case performance for these algorithms. For example, consider DDET's guided exploration solution, which uses sets of rules and biases to guide the simulation towards the most relevant risk scenarios. The worst possible performance for these algorithms happens when the relevant accident events are in the last possible branches in the DDET, meaning an exponentially large number of branches must be explored before reaching the relevant ones.

A more specific example is [16], where simulation (and therefore a DDET) is guided through hierarchical planning in the SimPRA framework [51]. The level of detail of the physical models is also adjusted when "level control nodes" are reached in the DDET to help reduce the computational time - i.e., the level control nodes adaptively adjust the level of detail in the simulation. The paper describes SimPRA's algorithms for generating the DDET, a combination of the simulator, scheduler, and planner modules. The worst-case performance conditions for this approach are:

1. The end-states are at the last possible level of the DDET, at depth d;
2. The level of detail is as high as possible;
3. Level control nodes are never reached.

We assume the simulator has a constant computational time, meaning it will always take the same amount of time to reach a branching point. According to [16], SimPRA performs a depth-first search in a simulation tree, first looking at the most relevant scenarios as defined by a biasing function (i.e., heuristic) - however, all possible branches of the tree will be reached eventually. As the branches are generated iteratively by the scheduler module and, according to the worst-case condition 1, this algorithm can be classified as an Iterative deepening depth-first search (IDDFS) algorithm[2]. The worst-case time complexity for IDDFS is exponential: $O(b^d)$, where b is the branching factor of the DDET. The space complexity[3] for IDDFS is, however, linear: $O(d)$.

In the average-case performance for [16], end-states will likely be found in levels before the maximum tree depth, biasing will lead to the most relevant scenarios first, and the level of detail will be adjusted from level control nodes, reducing the simulation's computing time. Thus, the average-case time complexity here is likely linear: $T(n \times m \times d)$, where n is the number of simulation rounds and m is the number of simulation runs in each round. However, the simulation tree generated may not be complete in the average case. If level control nodes adaptively reduce the level of detail, some branches may not be reached due to the lower accuracy of simpler models. Thus, the state-space is fully explored in the worst case, but not in the average case. The algorithm is traceable in both cases, as it records the branches in a simulation tree.

[2] https://en.wikipedia.org/wiki/Iterative_deepening_depth-first_search

[3] Typically performed alongside time complexity analysis, space complexity defines the growth rate for an algorithm in terms of memory requirements (e.g., RAM usage) [28].

In state-space pruning, the worst-case performance happens when the truncation parameters or optimal bounds are inefficient - i.e., when the probability thresholds are close to zero or when the lower and upper bounds of the branch-and-bound approach are close to infinity. For example, [45] uses a backtracking algorithm for DDET generation, which equates to a depth-first search approach with probability threshold pruning. The worst-case performance for this algorithm is when the probability threshold is close to zero. The algorithm then has the worst-case exponential time complexity of the depth-first search approach.

Finally, the same phenomenon as in DDET's guided exploration applies in Monte Carlo simulation: The approaches for solving the state-explosion problem in MCS-based DPRA are obsoleted in the worst-case performance scenario. MCS generates a random sequence of events with or without time parametrization. The discrete event sequence generation in Monte Carlo approaches can be biased towards the most risk-significant scenarios. For example, in [52], importance sampling was used to bias the generation of event sequences towards a defined goal, evaluated from the sequences' feasibility. In [53], a heuristic function is used to bias the simulation towards the most significant scenario in terms of risk. The goal, in this case, is to find "low probability-high risk scenarios." The worst-case performance scenario for these methods is when the biases and heuristics do not lead to a desired event or feature. The exponential complexity comes from the number of scenarios that must be simulated before the desired ones are found. The average-case scenario expects the heuristic or search function to converge in a reasonable time, and linear or polynomial computational complexity is achievable.

Generating discrete event sequences over a discrete state-space is another approach for MCS-based supervised DPRA. It discretizes and divides the state-space into bounded cells, aggregating similar regions. This discretization makes the state-space countable and finite. Therefore, it should be possible to search it exhaustively in polynomial or linear time. Examples of this approach with MCS for scenario generation is shown in [54, 55, 56]. Memorizing interesting parts of the event sequence and then using them in the generation of interesting scenarios via MCS is the approach shown in [23, 22]. The worst-case performance scenario is when the discretized state-space is uncountable and infinite - i.e., when the discretized cells are infinitesimal. Although the most accurate results are achieved with the full state-space, the computational complexity to search it exhaustively is exponential.

5.2. Polynomial Complexity

Polynomial complexity is defined by growth functions such as $T(n) = n^2$, being two complexity classes below exponential. Although most supervised DPRA works have exponential worst-case complexity, there are ten instances of polynomial worst-case complexity.

Polynomial computational complexities are primarily achieved using ROMs, discussed in Section 4.3, as mathematical models or data-based approximators (e.g., ANNs). The downside of using ROMs is that full state-space coverage is not achievable, as some accuracy is lost due to the simplified nature of the ROMs. For example, [57] proposes an interpolation method that combines a reduced-order model, a clustering algorithm called dynamic time warping, and a DPRA framework called Risk Analysis Visualization Environment (RAVEN). This approach allows for an approximate estimation of the "response surface" for a system, which relates operation time and response time to the probability of failure. The authors pose that the computational cost to interpolate the reduced-order model over the whole response surface is polynomial: $O(N_1 N_2[(N+2)^3 + (N+2)])$, where N is the size of the sample data used to regress the ROM, and N_1 and N_2 are the sizes of interpolations along the y and x-axis of the response surface - in this case, operating time and response time. There is some accuracy loss in the interpolation compared to the original nonlinear model, which means the full state-space of possibilities is not covered for the response surface - although, according to the paper, the interpolation achieves reasonable accuracy.

Besides using ROMs, novel approaches with polynomial complexities have been proposed to address the state explosion problem. These include, for example, the Repetitive Simulation Trials After Reaching Thresholds (RESTART) method [21]. RESTART guides Monte Carlo simulations using a heuristic function and a precomputed initial representation of the critical scenarios to be explored (e.g., precomputed DDETs). Adaptive sampling approaches also achieve polynomial complexity. An example of adaptive sampling is implemented in RAVEN [58], where a "sampler" module is used to probe the state-space for the most relevant simulation scenarios, based on some input data. Then, only the probed scenarios are simulated.

Finally, event-driven exploration is a novel methodology for supervised DPRA with polynomial complexity, proposed in [59]. It combines Monte Carlo simulation and dynamic fault trees: MCS is sped up by using an event-driven

scheduler and simulator based on DFTs, such that trajectories without relevant events are not simulated. The event-driven approach does not fully explore the state-space, as only the relevant trajectories are simulated.

6. Discussions

This section presents general discussions on the reviewed works, insights from the complexity analyses, and proposals for novel supervised DPRA methods.

6.1. The State Explosion Problem

Since the 1990s, the state explosion problem has been known and discussed in the DPRA literature. For example, Marseguerra and Zio conclude their classical DPRA work [60] by remarking that the complex nature of real-world, large-scale systems may lead to high computation times and that efforts should be made to mitigate this problem. Over the years, these efforts came forth as proposals for solving the state explosion problem. However, as seen in Section 5, these solutions must sacrifice accuracy, traceability, or state-space coverage in order to reduce the computational burden. Traceability is vital to explainable AI. A clear sequence of events or fully described trajectory must be present, from an initial condition to accident events. Accuracy and state-space coverage are also important as some severe accident sequences might be neglected if there is a lack of accuracy or if the full state-space is not covered.

Thus, a balance must be met when using DPRA in practice. Different methods may be more attractive depending on the user's specific needs and the conditions of the system under study. For an autonomous drone, for example, a faster execution time with less accuracy is preferred, while a more accurate but slower model can be used for an autonomous ship. Nevertheless, the exponential time complexity of most supervised DPRA methods makes their application prohibitive to real-world, large-scale dynamic systems, as the execution times grow exponentially with the number of components in a dynamic system. A supervised DPRA solution with a lower time complexity may be desirable as it is more scalable, i.e., it is able to compute a larger number of risk scenarios in a given amount of time.

6.2. Worst-case vs. Average-case Performance

This work aims to find a DPRA method with a computational complexity lower than exponential. Therefore, the computational complexity analyses were performed for the supervised DPRA methods. The methods were evaluated in terms of time complexity, traceability and whether they explore the whole state-space of risk scenarios.

In computer science, an algorithm's complexity is typically analyzed asymptotically for an increasing input size assuming the worst-case performance scenario. This analysis is useful because it defines an upper bound on the algorithm's execution time, meaning it will theoretically not perform worse than its complexity indicates. However, practical applications of the algorithms will likely not face the worst-case scenario. Considering the average-case performance scenario when performing the complexity analyses could yield more information about the methods' real-world performance, as the supervised DPRA's solutions to state explosion are likely to improve the average-case complexity. A caveat is that an average-case input size for each supervised DPRA method must be defined, which may not be trivial.

6.3. Complexity Class of DDET-based DPRA

Defining the complexity class for the DPRA problem gives insight into the motivations behind the current supervised DPRA methods and the challenges to solving the state explosion problem. Complexity classes are more generic than the asymptotic complexities seen before, defined for sets of problems relative to the properties of any algorithm that can solve those problems [28]. A problem's complexity class, in terms of computational time, can be defined by proving that the time complexity for any algorithm that solves that problem lies between the upper (i.e., worst-case) and lower (i.e., best-case) bounds.

Consider a generalization of the DPRA problem based on discrete dynamic event trees. A complete DDET is generated by considering all of the possible component failure modes for each possible time step: From Section 1, m^{cs}. This function defines the branching factor of the DDET, i.e., the horizontal dimension of the DDET. At each simulation branching point, the number of states generated in the next branching point depends on the number of components, component modes and number of time steps to be simulated. The branching factor is unbounded, assuming model uncertainties and the stochastic nature of component failures, as an uncountable amount of time

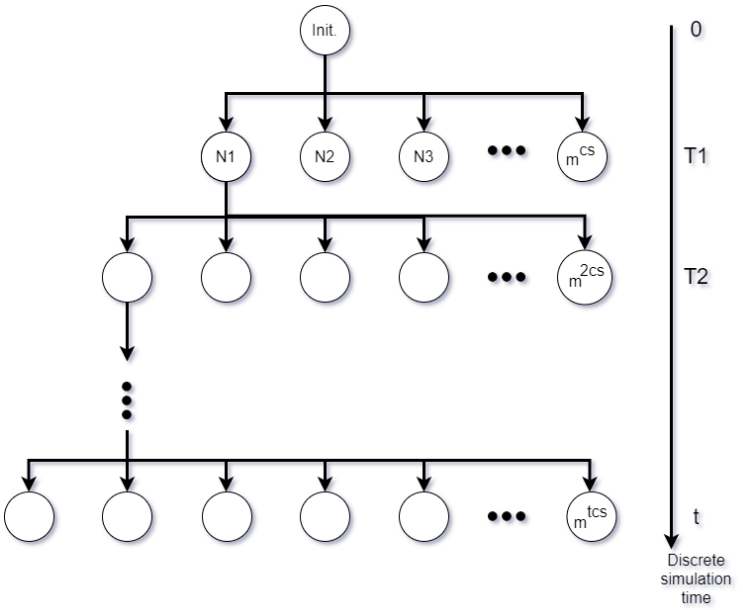

Figure 4: Generic DDET for the complexity class analysis. The simulation tree grows downwards in time until a given discrete time t, and horizontally at each discrete simulation time until all combinations of components and stochastic failure time steps (m^{cs}) are considered.

steps must be simulated to find all possible events. The vertical dimension of the DDET (i.e., the DDET depth) is given by the simulation time, or t in the m^{cst} function from Section 1. This generic dynamic event tree is shown in Figure 4.

The best-case performance for a DDET generation algorithm is if the accident events are present in the first level of the event tree - i.e., if there is only one transition between the initial condition and the undesirable consequences. In Figure 4, this would mean only the first level (at "T1") of the DDET is generated. However, the algorithm would take an exponential amount of time to generate the first level of the tree due to the exponential branching factor, which depends on the unbounded number of time steps. Therefore, the best-case performance for DDET generation is exponential in the generic case described above. As seen in Section 5, the worst-case complexity for DDET-based DPRA is also exponential, and thus DDET generation belongs to the *NP* complexity class: Solvable in exponential time but verifiable in polynomial time or less [28]. As DDET generation is NP-complex, any algorithm that can solve the generic DDET in Figure 4 must have exponential complexity.

This complexity class implies that most likely there is no solution for DDET-based DPRA that can explore a full state space of possibilities - i.e., generate a full DDET as in Figure 4 - in less than exponential time. However, it is possible to reduce the DDET-based DPRA problem to sub-problems with a *P* complexity class, as seen in Table A.2. In order to achieve polynomial time, the supervised DPRA methods search a subset of the full DDET, sacrificing accuracy or traceability. This reduction of DDET-based DPRA problems to polynomial time means DDET-based DPRA is *NP-hard* [28]: It belongs to the NP complexity class, but can be reduced to sub-problems with lower complexities.

6.4. Possible Paths to Improved Supervised DPRA

As discussed in Section 6.1, a supervised DPRA method with low computational complexity and which retains accuracy and traceability is desirable. Methods from automated planning are good candidates, as the algorithms used are generally well optimized. More specifically, the problem of K-Shortest-Paths (KSP) is interesting, defined as the problem of finding the K best paths (e.g., the ten best paths) between two nodes in a graph ranked by some metric.

However, two requirements of KSP are a predefined number of paths K and a predefined representation of the state-space to be explored (e.g., a graph). In other words, we must know the state-space and how many accident scenarios exist beforehand, which is not possible in DPRA. However, the K* (i.e., K-Star) algorithm [61] can solve

the KSP problem without a predefined state-space or number of paths and therefore could theoretically be applied to DPRA. Furthermore, K* has a log-linear worst-case time complexity, making it a good candidate for supervised DPRA.

Another idea for supervised DPRA is to use Temporal Neural Networks (TNN) for pattern identification in DDET generation or system trajectories in MCS, similarly to [62]. TNNs differ from conventional networks as they consider time series data in the prediction - i.e., they consider that past data affects the current prediction [63]. Considering that the risk of a given accident scenario for a dynamic system changes over time due to its dynamic nature, it makes sense to use TNNs to predict the future level of risk, based on the information from the initial condition up to some point in the present.

Another possible solution for supervised DPRA is the *Divide-et-Impera* (DeI) approach [64, 65], which allows for a generic solution to DPRA independent of underlying models and systems. The ideal solution, i.e., with low computational complexity and complete coverage, depends on the size and properties of the modeled system - e.g., an NPP with a limited event set and strict transition constraints or a MASS with a large event set and weak transition constraints. Similar to the Galileo tool for DFTs [40], the DeI approach divides the complete search problem into multiple smaller and more specialized problems, solved by specialized search agents. Each smaller problem focuses on specific features or goals based on examined system properties. These agents can be implemented with one of the presented methods (e.g., K*), each looking for risk scenarios with different parameters - for example, high-consequence, low-probability scenarios, and vice-versa.

6.5. Limitations

There are three main limitations of this work. The first is related to the subjectivity in the computational complexity analysis performed in Section 5. Typically, an algorithm's asymptotic complexity is defined objectively as a function of its input size. For the supervised DPRA methods, this objective analysis was not possible in most cases, as most papers lacked algorithms and implementations to be evaluated. Therefore, the algorithms and inner workings for most methods presented in Table A.2 had to be interpreted from the papers, which makes the complexity analysis presented here intrinsically subjective.

The second limitation is related to the average-case performance, as discussed in Section 6. Most solutions to the state explosion problem are not relevant when considering the worst-case performance. An average-case analysis could yield more information in comparing the supervised DPRA methods. However, defining an average-case performance scenario for the papers is problematic since, once again, most papers do not provide implementation details or algorithms.

Finally, the last limitation relates to the snowballing literature review process. An exhaustive search of the literature was not performed here, as the snowballing process was interrupted after the third iteration. Therefore, relevant works might have been missed, and therefore further exploration of the literature is recommended.

7. Conclusions

In this work, a systematic review of state-of-the-art literature has been performed using the snowballing methodology to find and evaluate solutions to the DPRA state explosion problem. Out of the 131 papers found with snowballing, 79 were classified as classical DPRA, and 52 were classified as supervised DPRA. Classical DPRA works do not address the state explosion problem which leads to challenges with computation complexity and time. Supervised DPRA, however, attempts in different ways to reduce the computation complexity and solve the state explosion problem. Computational complexity analyses were performed on the supervised DPRA methods, where their worst-case time complexities were defined and compared to the baseline exponential complexity defined in [50]. More specifically, the algorithms for each of the 52 supervised DPRA paper were analyzed in terms of the number of operations performed, yielding computational time complexities. The algorithms were analyzed directly when presented in the papers. When no algorithms were presented, the papers were studied and their algorithms deduced, to then be analyzed.

Ten instances of a lower computational complexity (i.e., polynomial) were found, while the other works retained the baseline exponential complexity. In terms of traceability, almost all supervised DPRA works allow for the risk scenarios to be traced, with four exceptions. These works use black-box methods such as ANNs to produce surrogate

or reduced-order models which are faster to compute, thus trading traceability for computational efficiency. The same happens with state space coverage; i.e., methods which avoid exploring the whole state space are more computationally efficient. This trade-off points towards the DPRA problem being NP-hard: Methods which solve DPRA must reduce the problem to achieve complexities lower than exponential.

As future works, the authors work on implementing and evaluating the proposed ideas for supervised DPRA discussed in Section 6, in particular related to K*. The resulting supervised DPRA methods should be applied to real-world case studies of autonomous systems with high levels of autonomy, such as maritime autonomous surface ships.

Acknowledgement

This work is sponsored by the Research Council of Norway through the Centre of Excellence funding scheme, NTNU AMOS, project number 223254, and the UNLOCK project, through the Research Council of Norway FRINATEK scheme, project number 274441. The authors would also like to express their appreciation to the anonymous reviewers for their valuable comments and insights.

References

[1] A. Le, C. Maple, T. Watson, A profile-driven dynamic risk assessment framework for connected and autonomous vehicles, in: Living in the Internet of Things: Cybersecurity of the IoT, 2018, pp. 1–8.

[2] B. J. Vartdal, R. Skjong, A. L. St.Clair, Remote-controlled and autonomous ships in the maritime industry, Tech. rep., DNV GL - Maritime (2018).

[3] M. Ludvigsen, A. J. Sørensen, Towards Integrated Autonomous Underwater Operations for Ocean Mapping and Monitoring, Annual Reviews in Control 42 (2016) 145–157.

[4] I. B. Utne, A. J. Sørensen, I. Schjølberg, Risk mangement of autonomous marine systems and operations, in: Proceedings of the 36th International Conference on Ocean, Offshore and Arctic Engineering, ASME, 2017, pp. 1–10.

[5] A. Mosleh, PRA: A perspective on strengths, current limitations, and possible improvements, Nuclear Engineering and Technology 46 (1) (2014) 1–10.

[6] K.-S. Hsueh, A. Mosleh, The development and application of the accident dynamic simulator for dynamic probabilistic risk assessment of nuclear power plants, Reliability Engineering & System Safety 52 (3) (1996) 297–314.

[7] M. Kloos, J. Peschke, MCDET: A probabilistic dynamics method combining Monte Carlo simulation with the discrete dynamic event tree approach, Nuclear Science and Engineering 153 (2) (2006) 137–156.

[8] J. Nielsen, A. Tokuhiro, R. Hiromoto, J. Khatry, Optimization method to branch-and-bound large sbo state spaces under dynamic probabilistic risk assessment via use of lendit scales and s2r2 sets, Journal of Nuclear Science and Technology 51 (10) (2014) 1212–1230.

[9] M. Kloos, J. Peschke, Monte Carlo and Dynamic Event Tree Simulation for Assessing the Potentials of Tube and Pipe Ruptures, in: Proceedings of the 29th European Safety and Reliability Conference, 2019, pp. 1940–1947.

[10] X. Yang, M. Sam Mannan, The development and application of dynamic operational risk assessment in oil/gas and chemical process industry, Reliability Engineering and System Safety 95 (7) (2010) 806–815.

[11] Y. Hu, T. Parhizkar, A. Mosleh, Guided simulation for dynamic probabilistic risk assessment of complex systems: Concept, method, and application, Reliability Engineering & System Safety 217 (2022) 108047.

[12] S. Shi, T. Wang, M. Diaconeasa, On the Use of the Accident Dynamic Simulator Method in Ship Collision Accident Analysis, in: Proceedings of the ASME 2020 International Mechanical Engineering Congress and Exposition, 2020, pp. 1–7.

[13] T. Aldemir, A survey of dynamic methodologies for probabilistic safety assessment of nuclear power plants, Annals of Nuclear Energy 52 (2013) 113–124.

[14] C. Smidts, Probabilistic dynamics: A comparison between continuous event trees and a discrete event tree model, Reliability Engineering and System Safety 44 (2) (1994) 189–206.

[15] B. Tombuyses, T. Aldemir, Continuous cell-to-cell mapping, Journal of Sound and Vibration 202 (3) (1997) 395–415.

[16] H. S. Nejad, D. Zhu, A. Mosleh, Hierarchical planning and multi-level scheduling for simulation-based probabilistic risk assessment, in: Proceedings of the 2007 Winter Simulation Conference, 2007, pp. 1189–1197.

[17] D. Zhu, A. Mosleh, C. Smidts, A framework to integrate software behavior into dynamic probabilistic risk assessment, Reliability Engineering & System Safety 92 (12) (2007) 1733–1755.

[18] J. Yang, T. Aldemir, An algorithm for the computationally efficient deductive implementation of the markov/cell-to-cell-mapping technique for risk significant scenario identification, Reliability Engineering and System Safety 145 (2016) 1–8.

[19] M. Hejase, A. Kurt, T. Aldemir, U. Ozguner, The backtracking process algorithm: A dynamic probabilistic risk assessment method for autonomous vehicle control systems, in: Proceedings of the 14th International Conference on Probabilistic Safety Assessment and Management, 2018, pp. 1–12.

[20] J. Nielsen, A. Tokuhiro, R. Hiromoto, L. Tu, Branch-and-bound algorithm applied to uncertainty quantification of a boiling water reactor station blackout, Nuclear Engineering and Design 295 (2015) 283–304.

[21] P. Turati, N. Pedroni, E. Zio, Advanced restart method for the estimation of the probability of failure of highly reliable hybrid dynamic systems, Reliability Engineering and System Safety 154 (2016) 117–126.

[22] M. Marseguerra, E. Zio, F. Cadini, Biased monte carlo unavailability analysis for systems with time-dependent failure rates, Reliability Engineering and System Safety 76 (1) (2002) 11–17.
[23] M. Marseguerra, E. Zio, J. Devooght, P. E. Labeau, A concept paper on dynamic reliability via Monte Carlo simulation, Mathematics and Computers in Simulation 47 (2) (1998) 371–382.
[24] P. E. Labeau, Probabilistic dynamics: Estimation of generalized unreliability through efficient monte carlo simulation, Annals of Nuclear Energy 23 (17) (1996) 1355–1369.
[25] M. Marseguerra, M. Nutini, E. Zio, Approximate physical modelling in dynamic psa using artificial neural networks, Reliability Engineering and System Safety 45 (1-2) (1994) 47–56.
[26] M. Marseguerra, M. Ricotti, E. Zio, Approaching system evolution in dynamic psa by neural networks, Reliability Engineering and System Safety 49 (1) (1995) 91–99.
[27] T. Parhizkar, J. E. Vinnem, I. B. Utne, A. Mosleh, Supervised dynamic probabilistic risk assessment of complex systems, part 1: General overview, Reliability Engineering and System Safety 208 (2021).
[28] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009.
[29] C. Gerety, P. Cull, Time complexity of the towers of hanoi problem, ACM SIGACT News 18 (1) (1986) 80–87.
[30] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, 2014, pp. 1–10.
[31] G. Cojazzi, The dylam approach for the dynamic reliability analysis of systems, Reliability Engineering and System Safety 52 (3 SPEC. ISS.) (1996) 279–296.
[32] C. Acosta, N. Siu, Dynamic event trees in accident sequence analysis: application to steam generator tube rupture, Reliability Engineering and System Safety 41 (2) (1993) 135–154.
[33] C. G. Acosta, N. O. Siu, Dynamic event tree analysis method (detam) for accident sequence analysis, Tech. rep., U.S. Office of Nuclear Regulatory Research (1991).
[34] J. Devooght, C. Smidts, Probabilistic reactor dynamics—i: The theory of continuous event trees, Nuclear Science and Engineering 111 (3) (1992) 229–240.
[35] T. Aldemir, Computer-assisted markov failure modeling of process control systems, IEEE Transactions on Reliability R-36 (1) (1987) 133–144.
[36] G. Cicotti, A. Coronato, A preliminary study of a probabilistic risk-based approach for ambient intelligence healthcare systems, in: Proceedings of the 11th International Conference on Intelligent Environments, 2015, pp. 58–69.
[37] B. Tombuyses, P. R. DeLuca, C. Smidts, Backward monte carlo for probabilistic dynamics, Mathematics and Computers in Simulation 47 (2-5) (1998) 493–505.
[38] S. Swaminathan, J. Y. Van-Halle, C. Smidts, A. Mosleh, S. Bell, K. Rudolph, R. J. Mulvihill, B. Bream, The cassini mission probabilistic risk analysis: Comparison of two probabilistic dynamic methodologies, Reliability Engineering and System Safety 58 (1) (1997) 1–14.
[39] S. B. Guarro, Risk-informed safety assurance and probabilistic risk assessment of mission-critical software-intensive systems, Tech. rep., Advanced System Concepts Associates (2010).
[40] K. Sullivan, J. Dugan, D. Coppit, The galileo fault tree analysis tool, in: Proceedings of the 29th Annual International Symposium on Fault-Tolerant Computing, 1999, pp. 232–235.
[41] J. B. Dugan, K. J. Sullivan, D. Coppit, Developing a low-cost high-quality software tool for dynamic fault-tree analysis, IEEE Transactions on Reliability 49 (1) (2000) 49–59.
[42] J. B. Dugan, H. Xu, Method and system for dynamic probabilistic risk assessment, U.S. Patent No. 8,346,694 B2 (2013).
[43] P. E. Labeau, A monte carlo estimation of the marginal distributions in a problem of probabilistic dynamics, Reliability Engineering and System Safety 52 (1) (1996) 65–75.
[44] G. Haggard, Pruning and depth first search, in: Theory and Applications of Graphs, 1978, pp. 216–224.
[45] J. Yang, T. Aldemir, C. Smidts, A deductive method for diagnostic analysis of digital instrumentation and control systems, IEEE Transactions on Reliability 67 (4) (2018) 1442–1458.
[46] A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, B. Rutt, U. Catalyurek, Dynamic generation of accident progression event trees, Nuclear Engineering and Design 238 (12) (2008) 3457–3467.
[47] J. M. Izquierdo, J. Hortal, M. Sanchez Perea, E. Meléndez, C. Queral, J. Rivas-Lewicky, Current Status and Applications of Integrated Safety Assessment and Simulation Code System for ISA, Nuclear Engineering and Technology 49 (2) (2017) 295–305.
[48] P. Turati, N. Pedroni, E. Zio, Simulation-based exploration of high-dimensional system models for identifying unexpected events, Reliability Engineering and System Safety 165 (2017) 317–330.
[49] D. Mandelli, C. Parisi, N. Anderson, Z. Ma, H. Zhang, Dynamic pra methods to evaluate the impact on accident progression of accident tolerant fuels, Nuclear Technology 207 (3) (2021) 389–405.
[50] R. G. Maidana, T. Parhizkar, C. A. Thieme, M. A. Ramos, I. B. Utne, A. Mosleh, Towards risk-based autonomous decision-making with accident dynamic simulation, in: Proceedings of the 31st European Safety and Reliability Conference, 2021, pp. 2391–2398.
[51] Y. Hu, A guided simulation methodology for dynamic probabilistic risk assessment of complex systems, Ph.D. thesis, University of Maryland (2005).
[52] M. Marseguerra, E. Zio, Nonlinear monte carlo reliability analysis with biasing towards top event, Reliability Engineering and System Safety 40 (1) (1993) 31–42.
[53] P. Turati, N. Pedroni, E. Zio, An adaptive simulation framework for the exploration of extreme and unexpected events in dynamic engineered systems, Risk Analysis 37 (1) (2017) 147–159.
[54] P. E. Labeau, E. Zio, The cell-to-boundary method in the frame of memorization-based monte carlo algorithms. a new computational improvement in dynamic reliability, Mathematics and Computers in Simulation 47 (2-5) (1998) 347–360.
[55] M. Marseguerra, E. Zio, The cell-to-boundary method in monte carlo-based dynamic psa, Reliability Engineering and System Safety 48 (3) (1995) 199–204.
[56] M. Marseguerra, E. Zio, Approaching dynamic reliability by monte carlo simulation, in: T. Aldemir, N. O. Siu, A. Mosleh, P. C. Cacciabue,

B. G. Göktepe (Eds.), Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 44–58.
[57] R. Christian, A. U. A. Shah, H. G. Kang, Dynamic pra-based estimation of pwr coping time using a surrogate model for accident tolerant fuel, Nuclear Technology 207 (3) (2021) 376–388.
[58] C. Rabiti, A. Alfonsi, J. Cogliati, D. Mandelli, R. Kinoshita, Raven, a new software for dynamic risk analysis, in: Proceedings of the 12th International Conference on Probabilistic Safety Assessment and Management, 2014, pp. 1–14.
[59] E. Gascard, Z. Simeu-Abazi, Quantitative analysis of dynamic fault trees by means of monte carlo simulations: Event-driven simulation approach, Reliability Engineering and System Safety 180 (2018) 487–504.
[60] M. Marseguerra, E. Zio, Monte carlo approach to psa for dynamic process systems, Reliability Engineering and System Safety 52 (3 SPEC. ISS.) (1996) 227–241.
[61] H. Aljazzar, S. Leue, K∗: A heuristic search algorithm for finding the k shortest paths, Artificial Intelligence 175 (18) (2011) 2129–2154.
[62] B. Zha, A. Vanni, Y. Hassan, T. Aldemir, A. Yilmaz, Deep transformer networks for time series classification: The NPP safety case, in: Proceedings of the International Topical Meeting on Probabilistic Safety Assessment and Analysis, 2021, pp. 1065–1074.
[63] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[64] G. Sharon, R. Stern, M. Goldenberg, A. Felner, The Increasing Cost Tree Search For Optimal Multi-Agent Pathfinding, Artificial Intelligence 195 (2013) 470–495.
[65] D. Patel, R. Zalila-Wenkstern, Scalable monte carlo tree search for cav s action planning in colliding scenarios, in: Proceedings of the 32nd IEEE Intelligent Vehicles Symposium, 2021, pp. 1065–1072.
[66] S. Swaminathan, C. Smidts, The event sequence diagram framework for dynamic probabilistic risk assessment, Reliability Engineering and System Safety 63 (1) (1999) 73–90.
[67] P. C. Cacciabue, A. Carpignano, C. Vivalda, Expanding the scope of dylam methodology to study the dynamic reliability of complex systems: the case of chemical and volume control in nuclear power plants, Reliability Engineering and System Safety 36 (2) (1992) 127–136.
[68] P. C. Cacciabue, G. Cojazzi, A human factors methodology for safety assessment based on the dylam approach, Reliability Engineering and System Safety 45 (1-2) (1994) 127–138.
[69] A. Amendola, Accident sequence dynamic simulation versus event trees, Reliability Engineering and System Safety 22 (1-4) (1988) 3–25.
[70] Z. Nivolianitou, A. Amendola, G. Reina, Reliability analysis of chemical processes by the dylam approach, Reliability Engineering 14 (3) (1986) 163–182.
[71] B. Rutt, U. Catalyurek, A. Hakobyan, K. Metzroth, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, Distributed dynamic event tree generation for reliability and risk assessment, in: Proceedings - Challenges of Large Applications in Distributed Environments, CLADE 2006, 2006, pp. 61–70.
[72] D. Mandelli, C. Smith, T. Riley, J. Nielsen, A. Alfonsi, J. Cogliati, C. Rabiti, J. Schroeder, Bwr station blackout: A rismc analysis using raven and relap5-3d, Nuclear Technology 193 (1) (2016) 161–174.
[73] U. Catalyurek, B. Rutt, K. Metzroth, A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, Development of a code-agnostic computational infrastructure for the dynamic generation of accident progression event trees, Reliability Engineering and System Safety 95 (3) (2010) 278–294.
[74] J. Zhou, G. Reniers, N. Khakzad, Application of event sequence diagram to evaluate emergency response actions during fire-induced domino effects, Reliability Engineering and System Safety 150 (2016) 202–209.
[75] A. Xu, Z. Zhang, M. Zhang, H. Wang, H. Zhang, S. Chen, Research on time-dependent failure modeling method of integrating discrete dynamic event tree with fault tree, Frontiers in Energy Research 7 (2019).
[76] M. J. Rebollo, C. Queral, G. Jimenez, J. Gomez-Magan, E. Meléndez, M. Sanchez-Perea, Evaluation of the offsite dose contribution to the global risk in a steam generator tube rupture scenario, Reliability Engineering and System Safety 147 (2016) 32–48.
[77] J. M. Izquierdo, J. Hortal, M. Sánchez, E. Meléndez, Automatic generation of dynamic event trees: A tool for integrated safety assessment (isa), in: Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 135–150.
[78] D. Mandelli, C. Smith, A. Alfonsi, Integrating classical pra models into dynamic pra, in: Proceedings of the 14th International Conference on Probabilistic Safety Assessment and Management, 2018, pp. 1–11.
[79] A. Amendola, G. Reina, F. Ciceri, Dynamic simulation of man-machine interaction in incident control, in: IFAC Proceedings Series, 1986, pp. 225–230.
[80] D. I. Gertman, L. N. Haney, N. O. Siu, Representing context, cognition, and crew performance in a shutdown risk assessment, Reliability Engineering and System Safety 52 (3 SPEC. ISS.) (1996) 261–278.
[81] D. Mandelli, C. Smith, A. Alfonsi, C. Rabiti, Analysis of the space propulsion system problem using raven, in: Proceedings of the 12th International Conference on Probabilistic Safety Assessment and Management, 2014, pp. 1–14.
[82] D. Mandelli, C. Smith, T. Riley, J. Nielsen, J. Schroeder, C. Rabiti, A. Alfonsi, J. Cogliati, R. Kinoshita, V. Pascucci, B. Wang, D. Maljovec, Overview of new tools to perform safety analysis: Bwr station black out test case, in: Proceedings of the 12th International Conference on Probabilistic Safety Assessment and Management, 2014, pp. 1–12.
[83] C. Rabiti, A. Alfonsi, D. Mandelli, J. Cogliati, R. Martineau, Raven as control logic and probabilistic risk assessment driver for relap-7, in: Transactions of the American Nuclear Society, 2012, pp. 333–335.
[84] D. Mandelli, S. Prescott, C. Smith, A. Alfonsi, C. Rabiti, J. Cogliati, R. Kinoshita, Modeling of a flooding induced station blackout for a pressurized water reactor using the rismc toolkit, in: International Topical Meeting on Probabilistic Safety Assessment and Analysis, PSA 2015, 2015, pp. 454–463.
[85] A. Alfonsi, C. Rabiti, D. Mandelli, J. J. Cogliati, R. A. Kinoshita, A. Naviglio, Dynamic event tree analysis through raven, in: International Topical Meeting on Probabilistic Safety Assessment and Analysis 2013, PSA 2013, 2013, pp. 1697–1709.
[86] Z. K. Jankovsky, M. R. Denman, T. Aldemir, Dynamic event tree analysis with the sas4a/sassys-1 safety analysis code, Annals of Nuclear Energy 115 (2018) 55–72.
[87] D. Mandelli, C. Parisi, A. Alfonsi, D. Maljovec, R. Boring, S. Ewing, S. St Germain, C. Smith, C. Rabiti, M. Rasmussen, Multi-unit dynamic pra, Reliability Engineering and System Safety 185 (2019) 303–317.
[88] D. Mandelli, C. Parisi, A. Alfonsi, D. Maljovec, S. St Germain, R. Boring, S. Ewing, C. Smith, C. Rabiti, Dynamic pra of a multi-unit plant,

in: International Topical Meeting on Probabilistic Safety Assessment and Analysis, PSA 2017, 2017, pp. 1061–1068.
[89] P. C. Cacciabue, A. Amendola, G. Cojazzi, Dynamic logical analytical methodology versus fault tree: The case study of the auxiliary feedwater system of a nuclear power plant., Nuclear Technology 74 (2) (1986) 195–208.
[90] S. Swaminathan, C. Smidts, The mathematical formulation for the event sequence diagram framework, Reliability Engineering & System Safety 65 (2) (1999) 103–118.
[91] R. Boring, R. Shirley, J. Joe, D. Mandelli, C. Smith, Simulation and non-simulation based human reliability analysis approaches, Tech. rep., U.S. Department of Energy (2014).
[92] C. Rabiti, D. Mandelli, A. Alfonsi, J. Cogliati, R. Kinoshita, Mathematical framework for the analysis of dynamic stochastic systems with the raven code, in: Proceedings of the International Conference on Mathematics and Computational Methods Applied to Nuclear Science & Engineering, 2013, pp. 1–12.
[93] A. Alfonsi, D. Mandelli, C. Smith, C. Rabiti, J. Cogliati, R. Kinoshita, New methods and tools to perform safety analysis within rismc, in: Transactions of the American Nuclear Society, 2013, pp. 1–4.
[94] A. P. Macwan, K. S. Hsueh, A. Mosleh, An approach to modelling operator behaviour in integrated dynamic accident sequence analysis, in: Proceedings of the International Symposium on the Use of Probabilistic Safety Assessment for Operational Safety, 1992, pp. 35–46.
[95] J. M. Izquierdo, M. Sanchez-Perea, Dylam-treta. an approach to protection systems software analysis, in: Systems Reliability Assessment, Springer, 1990, pp. 183–203.
[96] D. C. Bley, D. R. Buttemer, J. W. Stetkar, Light water reactor sequence timing: its significance to probabilistic safety assessment modeling, Reliability Engineering and System Safety 22 (1-4) (1988) 27–60.
[97] A. D. Domínguez-García, J. G. Kassakian, J. E. Schindall, J. J. Zinchuk, An integrated methodology for the dynamic performance and reliability evaluation of fault-tolerant systems, Reliability Engineering and System Safety 93 (11) (2008) 1628–1649.
[98] J. Wang, S. Zeng, J. Ma, W. Wu, Research on an integrated methodology of the dynamic performance and reliability evaluation, in: Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety, 2011, pp. 122–128.
[99] I. A. Papazoglou, Markovian reliability analysis of dynamic systems, in: Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 24–43.
[100] B. S. Dhillon, Reliability analysis under fluctuating environment using markov method, in: Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 127–134.
[101] B. K. Walker, Evaluating performance and reliability of automatically reconfigurable aerospace systems using markov modeling techniques, in: Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 101–126.
[102] M. Smotherman, K. Zemoudeh, A non-homogeneous markov model for phased-mission reliability analysis, IEEE Transactions on Reliability 38 (5) (1989) 585–590.
[103] M. Houtermans, G. Apostolakis, A. Brombacher, D. Karydas, The dynamic flowgraph methodology as a safety analysis tool: Programmable electronic system design and verification, Safety Science 40 (9) (2002) 813–833.
[104] C. J. Garrett, S. B. Guarro, G. E. Apostolakis, The dynamic flowgraph methodology for assessing the dependability of embedded software systems, IEEE Transactions on Systems, Man, and Cybernetics 25 (5) (1995) 824–840.
[105] C. J. Garrett, G. E. Apostolakis, Automated hazard analysis of digital control systems, Reliability Engineering and System Safety 77 (1) (2002) 1–17.
[106] M. Yau, S. Guarro, G. Apostolakis, Demonstration of the dynamic flowgraph methodology using the titan ii space launch vehicle digital flight control system, Reliability Engineering and System Safety 49 (3) (1995) 335–353.
[107] C. T. Muthukumar, S. B. Guarro, G. E. Apostolakis, Dependability analysis of embedded software systems, in: Reliability and Safety Assessment of Dynamic Process Systems, 1994, pp. 59–77.
[108] S. Oliva, M. Yau, S. Dixon, S. Guarro, Advanced pra tool benchmark for space system risk using the dynamic flowgraph methodology, in: Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management, 2006, pp. 1–8.
[109] M. Hejase, A. Kurt, T. Aldemir, U. Özgüner, S. B. Guarro, M. K. Yau, M. D. Knudson, Quantitative and risk-based framework for unmanned aircraft control system assurance, Journal of Aerospace Information Systems 15 (2) (2018) 57–71.
[110] J. Devooght, C. Smidts, Probabilistic dynamics as a tool for dynamic psa, Reliability Engineering and System Safety 52 (3 SPEC. ISS.) (1996) 185–196.
[111] Y. M. Lin, X. Pan, Study on risk scenarios of project failure based on monte-carlo simulation, in: Proceedings of the 18th International Conference on Industrial Engineering and Engineering Management, 2011, pp. 1291–1295.
[112] D. L. Deoss, N. O. Siu, A simulation model for dynamic system availability analysis, Tech. rep., Massachusetts Institute of Technology (1989).
[113] C. Smidts, J. Devooght, Probabilistic reactor dynamics—ii: A monte carlo study of a fast reactor transient, Nuclear Science and Engineering 111 (3) (1992) 241–256.
[114] J. B. Dugan, S. J. Bavuso, M. A. Boyd, Dynamic fault-tree models for fault-tolerant computer systems, IEEE Transactions on Reliability 41 (3) (1992) 363–377.
[115] S. Amari, G. Dill, E. Howald, A new approach to solve dynamic fault trees, in: Proceedings of the Annual Reliability and Maintainability Symposium, 2003, pp. 374–379.
[116] R. Gulati, J. Dugan, A modular approach for analyzing static and dynamic fault trees, in: Annual Reliability and Maintainability Symposium, 1997, pp. 57–63.
[117] Y. Dutuit, E. Châtelet, J. P. Signoret, P. Thomas, Dependability modelling and evaluation by using stochastic petri nets: Application to two test cases, Reliability Engineering and System Safety 55 (2) (1997) 117–124.
[118] V. Volovoi, Modeling of system reliability petri nets with aging tokens, Reliability Engineering and System Safety 84 (2) (2004) 149–161.
[119] F. Brissaud, C. Smidts, A. Barros, C. Bérenguer, Dynamic reliability of digital-based transmitters, Reliability Engineering and System Safety 96 (7) (2011) 793–813.
[120] E. Hofer, M. Kloos, B. Krzykacz-Hausmann, J. Peschke, M. Woltereck, An approximate epistemic uncertainty analysis approach in the presence of epistemic and aleatory uncertainties, Reliability Engineering and System Safety 77 (3) (2002) 229–238.

[121] P. Bucci, J. Kirschenbaum, L. A. Mangan, T. Aldemir, C. Smith, T. Wood, Construction of event-tree/fault-tree models from a markov approach to dynamic system reliability, Reliability Engineering and System Safety 93 (11) (2008) 1616–1627.
[122] C. Kermisch, P.-E. Labeau, Implementation of hybrid petri nets — lessons learned from their application to a smr unit, in: Probabilistic Safety Assessment and Management, 2004, pp. 681–686.
[123] J. M. Izquierdo-Rocha, M. Sanchez-Perea, G. Cojazzi, Integrated safety assessment (isa): An approach for the assessment of the software aspects of protection systems, in: Proceedings of the 4th International Topical meeting on Nuclear Thermal Hydraulics, Operation and Safety, 1994, pp. 20–C–1 – 20–C–6.
[124] T. Matsuoka, M. Kobayashi, Go-flow: A new reliability analysis methodology., Nuclear Science and Engineering 98 (1) (1988) 64–78.
[125] R. Sterritt, A. H. Marshall, C. M. Shapcott, S. I. McClean, Exploring dynamic bayesian belief networks for intelligent fault management systems, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2000, pp. 3646–3652.
[126] K. Nakada, K. Miyagi, N. Handa, S. Hattori, A method of state transition analysis under system interactions: An analysis of a shutdown heat removal system, Nuclear Technology 82 (2) (1988) 132–146.
[127] Y. Hu, A. Mosleh, An entropy-based exploration strategy in dynamic pra, in: Proceedings of the Probabilistic Safety Assessment and Management (PSAM) conference, Springer, 2004, pp. 2391–2397.
[128] S. H. Nejad-Hosseinian, Automatic generation of generalized event sequence diagrams for guiding simulation based dynamic probabilistic risk assessment of complex systems, Ph.D. thesis, University of Maryland (2007).
[129] T. Parhizkar, J. E. Vinnem, I. B. Utne, A. Mosleh, Supervised Dynamic Probabilistic Risk Assessment of Complex Systems, Part 1: General Overview, Reliability Engineering & System Safety 208 (2021) 107406.
[130] T. Parhizkar, I. B. Utne, J. E. Vinnem, A. Mosleh, Supervised dynamic probabilistic risk assessment of complex systems, part 2: Application to risk-informed decision making, practice and results, Reliability Engineering & System Safety 208 (2021) 107392.
[131] Y. Hu, H. Nejad, D. Zhu, A. Mosleh, Solution of Phased-Mission Benchmark Problem Using the SimPRA Dynamic PRA Methdology, in: Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management, 2006, pp. 1–9.
[132] D. Zhu, C. Smidts, A. Mosleh, Software modelling in a dynamic pra environment, in: Proceedings of the 8th International Conference on Probabilistic Safety Assessment and Management, 2006, pp. 1–9.
[133] Y. Chen, S. Yang, W. Men, Automatic generation of failure mechanism propagation scenario via guided simulation and intelligent algorithm, IEEE Access 7 (2019) 34762–34775.
[134] L. Ibánez, J. Hortal, C. Queral, J. Gómez-Magán, M. Sánchez-Perea, I. Fernández, E. Meléndez, A. Expósito, J. M. Izquierdo, J. Gil, H. Marrao, E. Villalba-Jabonero, Application of the integrated safety assessment methodology to safety margins. dynamic event trees, damage domains and risk assessment, Reliability Engineering and System Safety 147 (2016) 170–193.
[135] D. Zhu, Integrating software behavior into dynamic probabilistic risk assessment, Ph.D. thesis, University of Maryland (2005).
[136] K. A. Coyne, A predictive model of nuclear power plant crew decisionmaking and performance in a dynamic simulation environment, Ph.D. thesis, University of Maryland (2009).
[137] J. H. Lee, A. Yilmaz, R. Denning, T. Aldemir, An online operator support tool for severe accident management in nuclear power plants using dynamic event trees and deep learning, Annals of Nuclear Energy 146 (2020).
[138] Y. Vorobyev, P. Kudinov, Development and application of a genetic algorithm based dynamic pra methodology to plant vulnerability search, in: International Topical Meeting on Probabilistic Safety Assessment and Analysis 2011, PSA 2011, 2011, pp. 559–573.
[139] J. Li, A. Mosleh, R. Kang, Likelihood ratio gradient estimation for dynamic reliability applications, Reliability Engineering and System Safety 96 (12) (2011) 1667–1679.
[140] J. Li, R. Kang, A. Mosleh, Reliability sensitivity analysis via the likelihood ratio method, in: ICRMS'2011 - Safety First, Reliability Primary: Proceedings of 2011 9th International Conference on Reliability, Maintainability and Safety, 2011, pp. 327–333.
[141] S. Puch, B. Wortelen, M. Fränzle, T. Peikenkamp, Evaluation of drivers interaction with assistant systems using criticality driven guided simulation, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, pp. 108–117.
[142] T. Parhizkar, I. B. Utne, J. E. Vinnem, A. Mosleh, Dynamic probabilistic risk assessment of decision-making in emergencies for complex systems, case study: Dynamic positioning drilling unit, Ocean Engineering 237 (2021).
[143] D. Maljovec, B. Wang, D. Mandelli, P.-T. Bremer, V. Pascucci, M. Pernice, R. Nourgaliev, Exploratory nuclear reactor safety analysis and visualization via integrated topological and geometric techniques, Tech. rep., U.S. Department of Energy (2013).
[144] D. Mandelli, A. Yilmaz, T. Aldemir, K. Metzroth, R. Denning, Scenario clustering and dynamic probabilistic risk assessment, Reliability Engineering & System Safety 115 (2013) 146–160.
[145] J. Kim, A. U. A. Shah, H. G. Kang, Dynamic risk assessment with bayesian network and clustering analysis, Reliability Engineering and System Safety 201 (2020).
[146] S. Swaminathan, C. Smidts, Identification of missing scenarios in esds using probabilistic dynamics, Reliability Engineering and System Safety 66 (3) (1999) 275–279.
[147] D. Zamalieva, A. Yilmaz, T. Aldemir, Online scenario labeling using a hidden markov model for assessment of nuclear plant state, Reliability Engineering and System Safety 110 (2013) 1–13.

Appendix A.

DPRA method	Citations	Number of works
DDET-based	[32], [31], [14], [66], [67, 68], [69], [70] [71], [72], [73], [74], [75], [76], [77], [78] [79], [80], [38], [81, 82], [83], [84], [85] [86], [87], [88], [89], [90], [91], [92] [33], [93], [94], [95], [96]	36
Markov Chain Models (MCM)	[35], [97], [98], [99], [100], [101], [102], [36]	8
Dynamic Flowgraph Methodology (DFM)	[103], [104, 105], [106], [107], [108], [39], [109]	8
Monte Carlo Simulation (MCS)	[110], [37], [111], [38], [60], [112], [113]	7
Dynamic Fault Trees (DFT)	[114], [40], [41], [115], [116], [42]	6
Petri nets	[117], [69], [118], [119]	4
Hybrid approaches	[120], [121], [122]	3
Integrated Safety Assessment (ISA)	[123, 77]	2
Continuous Event Trees (CET)	[15], [14]	2
GO-FLOW	[124]	1
Dynamic Bayesian Networks (DBN)	[125]	1
State Transition Analysis	[126]	1

Table A.1: Works related to Classical DPRA methods, grouped by the methodology used for achieving DPRA. Some works present comparisons between different methodologies, and therefore appear in multiple groups in the table.

DPRA method	State explosion solution	Citation	Worst-case computational complexity	Traceable?	Full state-space coverage?
DDET-based	Guided exploration	[127]	Exponential	Yes	Yes
		[16]	Exponential	Yes	Yes
		[17]	Exponential	Yes	Yes
		[51]	Exponential	Yes	Yes
		[128]	Exponential	Yes	Yes
		[129]	Exponential	Yes	Yes
		[130]	Exponential	Yes	Yes
		[6]	Exponential	Yes	Yes
		[131]	Exponential	Yes	Yes
		[132]	Exponential	Yes	Yes
		[133]	Exponential	Yes	Yes
		[134]	Exponential	Unknown	Yes
		[135]	Exponential	Yes	Yes
		[136]	Exponential	Yes	Yes
		[11]	Exponential	Yes	Yes
	State space pruning	[8]	Exponential	Yes	Yes
		[20]	Exponential	Yes	Yes
		[45]	Exponential	Yes	Yes
		[46]	Exponential	Yes	Yes
		[47]	Exponential	Unknown	Yes
	Reduced-order models	**[57]**	**Polynomial**	**Yes**	**No**
		[137]	**Polynomial**	**No**	**No**
		[62]	**Polynomial**	**No**	**No**
	Hybrid approaches	[19]	Exponential	Yes	Yes
		[18]	Exponential	Yes	Yes
	Genetic algorithms	[138]	Exponential	Yes	Yes
Monte Carlo simulation	Biasing and importance sampling	[43]	Exponential	Yes	No
		[139]	Exponential	Yes	Yes
		[140]	Exponential	Yes	Yes
		[52]	Exponential	Yes	Yes
		[141]	Exponential	Yes	No
		[22]	Exponential	Yes	No
		[56]	Exponential	Yes	Yes
		[142]	Exponential	Yes	No
	Reduced-order models	**[25]**	**Polynomial**	**No**	**Yes**
		[26]	**Polynomial**	**No**	**Yes**
		[49]	**Polynomial**	**Yes**	**No**
	Cell-to-boundary method	[54]	Exponential	Yes	Yes
		[55]	Exponential	Yes	Yes
	Hybrid approaches	[48]	Exponential	No	Yes
		[21]	**Polynomial**	**Yes**	**No**
	State space pruning	**[23]**	**Polynomial**	**Yes**	**No**
		[24]	Exponential	Yes	Yes
	Adaptive simulation	[53]	Exponential	Yes	No
Others	**Adaptive sampling**	**[58]**	**Exponential/ Polynomial**	**Yes**	**Yes**
		[143]	**Polynomial**	**No**	**No**
	Data clustering	[144]	Exponential	Yes	Yes
		[145]	Exponential	Yes	Yes
	ESD guidance	[146]	Exponential	Yes	Yes
	Event-driven exploration	**[59]**	**Polynomial**	**Yes**	**No**
	Guided Markov model	**[147]**	**Polynomial**	**Yes**	**No**
	State space pruning	[7]	Exponential	Yes	No

Table A.2: Supervised DPRA methods, their proposed solutions to the state explosion problem, and their properties. As with Table A.1, the works are grouped by the DPRA method, as well as the approach chosen to solve the state explosion problem. The rows in bold represent the approaches with the best computational complexity. When a property is "Unknown", it means there was not enough information in the paper for defining that property.