Software and Systems Modeling (2022) 21:2213–2242
https://doi.org/10.1007/s10270-021-00956-0

THEME SECTION PAPER

Decision-making under uncertainty: be aware of your priorities

Huma Samin[1] · **Nelly Bencomo**[2] · **Peter Sawyer**[1]

Received: 18 November 2020 / Revised: 27 September 2021 / Accepted: 22 November 2021 / Published online: 25 January 2022

Abstract
Self-adaptive systems (SASs) are increasingly leveraging autonomy in their decision-making to manage uncertainty in their operating environments. A key problem with SASs is ensuring their requirements remain satisfied as they adapt. The trade-off analysis of the non-functional requirements (NFRs) is key to establish balance among them. Further, when performing the trade-offs it is necessary to know the importance of each NFR to be able to resolve conflicts among them. Such trade-off analyses are often built upon optimisation methods, including decision analysis and utility theory. A problem with these techniques is that they use a single-scalar utility value to represent the overall combined priority for all the NFRs. However, this combined scalar priority value may hide information about the impacts of the environmental contexts on the individual NFRs' priorities, which may change over time. Hence, there is a need for support for runtime, autonomous reasoning about the separate priority values for each NFR, while using the knowledge acquired based on evidence collected. In this paper, we propose Pri-AwaRE, a self-adaptive architecture that makes use of Multi-Reward Partially Observable Markov Decision Process (MR-POMDP) to perform decision-making for SASs while offering awareness of NFRs' priorities. MR-POMDP is used as a priority-aware runtime specification model to support runtime reasoning and autonomous tuning of the distinct priority values of NFRs using a vector-valued reward function. We also evaluate the usefulness of our Pri-AwaRE approach by applying it to two substantial example applications from the networking and IoT domains.

Keywords Self-Adaptive systems · Priorities · Non-functional requirements · Decision-making

1 Introduction

Self-adaptive systems (SASs) are systems that take dynamic adaptive decisions under uncertain environmental conditions to achieve their functional and non-functional requirements (NFRs) [26,30,51]. An example of a SAS would be an Internet of Things (IoT) network [2,61], that serves as a basis for the implementation of a cyber-physical system such as a smart home. These systems are continuously exposed to different environmental situations, such as communication interference and dynamic traffic loads, which may have different effects on the satisfaction levels of the NFRs. Such effects might include high energy consumption or poor packet delivery performance [27,35,52]. Hence, a SAS needs to adapt dynamically at runtime. These adaptations may involve trade-offs between the SAS's NFRs as the encountered environmental conditions change, sometimes in ways that deviate from those anticipated at design time.

Specification models exist that describe the decision-making process based on these trade-offs, and include the alternative adaptation actions and the priority of NFRs [20,29]. If these specification models, developed at design time, can also be used and updated at runtime, a SAS can be made requirements-aware [51] by monitoring its requirements' compliance during runtime. The specification model ($\boldsymbol{S}$) is derived by analysts from the requirements and the knowledge domain ($\boldsymbol{K}$). According to Zave and Jackson [63] (see Eq. 1), monitoring compliance with the requirements ($\boldsymbol{R}$) can be done by monitoring compliance with the specification model ($\boldsymbol{S}$).

Communicated by J. Araujo, A. Moreira, G. Mussbacher and P. Sánchez.

✉ Huma Samin
h.samin@aston.ac.uk

Nelly Bencomo
nelly.bencomo@durham.ac.uk

Peter Sawyer
p.sawyer@aston.ac.uk

[1] Aston University, Birmingham, UK

[2] Durham University, Durham, UK

$$S, K \vdash R \tag{1}$$

It is safe to believe that ***S*** will remain a valid implementation of ***R*** if and only if ***K*** does not change from the moment that ***S*** was built until the moment requirements compliance is assessed. However, in a SAS there is uncertainty about ***K*** [10], which cannot be assumed to remain unchanged. On the upside, a SAS may provide opportunities to learn about its environment and so reduce ***K***'s uncertainty.

Several runtime optimisation techniques have been developed to support the decision-making process specified in ***S*** of SASs [1,6,9,16,18,33,59]. These techniques are based on optimisation methods that include decision analysis and utility theory [46,47], that select from a set of alternatives the adaptation that yields the highest utility value.

A problem with these techniques used in SASs is that they typically use single-objective optimisation techniques [44,55], i.e. they use a single-scalar cumulative utility value to represent a combined cardinal priority for all NFRs. However, the adaptive decisions taken by SASs can have different effects, either positive or negative, on the satisfaction levels of individual NFRs [30,55,59]. For example, in an IoT network, the decision to increase transmission power on the links under the situation of high interference will have a positive impact on the packet delivery performance, but it will harm energy consumption [61]. Single-objective techniques using a combined cardinal priority do not give any information about the different impacts of adaptive decisions on the individual NFRs in terms of their satisfaction. Further, these impacts may change according to the evolution of the SAS over time, leading the SAS to evolve its adaptation strategies [13]. Hence, the priorities assigned at design time may no longer be valid at runtime due to unforeseen or emergent contexts that in turn may lead to the violation of a NFR. In a nutshell, the limitations of current optimisation techniques are (i) that they treat NFRs' priorities as a single combined value, and (ii) the assigned NFRs' priorities are considered to be fixed and unchanging.

1.1 Principal ideas

We argue that adaptation decisions need to be informed by the NFRs' priorities; something that cannot be achieved if the priorities are aggregated into a single combined value. However, it may prove that the priorities assigned at design time are not appropriate or even achievable under certain conditions. The priorities may need to be re-evaluated and changed, informed by the knowledge gained by the SAS encountering such conditions.

Let us define priority-awareness.

Definition 1 Priority-awareness is the capability of provision for autonomous changes of priorities to address the required satisfaction levels of NFRs.

The compliance with the requirements (***R***), according to Zave and Jackson [63], can be achieved using a runtime specification model (***S***) that is equipped with the newly found knowledge (***K'***) that has an effect on changing individual NFR priorities.

$$S, K' \vdash R \tag{2}$$

The key challenge here is to have a runtime specification model that has the capability of:

(1) modelling and reasoning with the cardinal priorities of individual NFRs.
(2) supporting the tuning of NFRs' priorities to better match the newly discovered situation and acquired knowledge, while respecting their relative priorities.

Based on [63], next we present our main contributions towards addressing the specified research challenges.

1.2 Contributions

In this paper, we propose Pri-AwaRE, a self-adaptive architecture, that uses an extension of Multi-Reward Partially Observable Markov Decision Process (MR-POMDP) [54,55] called MR-POMDP++, as a runtime specification model (***S***) embedded within the MAPE-K loop. MR-POMDP++ is a multi-objective sequential decision-making technique that uses the concept of rewards to:

a. support priority-aware decision-making by providing the runtime modelling and reasoning of priorities of individual NFRs using a vector-valued reward function. Hence, the decision-making takes into account the knowledge (***K'***) that allows the MR-POMDP++ to re-evaluate the priorities.

b. provide SASs with a principled way to maintain compliance of the *Requirements* (***R***) by autonomously tuning the NFRs' priorities at runtime under uncertain environmental contexts.

We also provide a proof of concept by applying our proposed approach to two different example applications from the networking and IoT domains and comparing it to the existing state-of-the-art techniques. Based on the experiments, we show that priority-aware decisions offered by our Pri-AwaRE architecture support satisfaction of NFRs in terms of more informed choices of NFRs' priorities.

1.3 Organization of the paper

The paper is organized as follows: Sect. 2 explains baseline concepts related to decision-making in SASs. Section 3 presents the Pri-AwaRE architecture to support priority-aware decision-making in SASs. In Sect. 4, the experiments and evaluations are presented followed by threats to validity in Sect. 5. Section 6 presents related work. Finally, the conclusions and future work are presented in Sect. 7.

2 Underlying concepts

In this section, we introduce the key concepts used in the paper. First, we define the decision-making process in SASs driven by NFRs [3,12,20]. We then describe the techniques and architectural concepts that we use in our work to ensure that NFRs are satisfied to an appropriate level that respects their respective priorities. These are: Partially Observable Markov Decision Process (POMDP), Multi-Reward Partially Observable Markov Decision Process (MR-POMDP), Optimistic Linear Support (OLS) algorithm and the MAPE-K architecture.

2.1 Decision-making in SASs

SASs are continuously exposed to different environmental contexts that affect the satisfaction of NFRs. During the decision-making process of a SAS, the system performs different tasks that have different impacts on the satisfaction levels of NFRs. The decision-making process of a SAS involves the following key concepts [4]:

2.1.1 NFRs

The main objective of decision-making in a SAS is to satisfy its non-functional quality requirements, i.e. the NFRs, while fulfilling the functional goals [6]. The NFRs are associated with two important characteristics at runtime [20]:

– *Satisfaction Level* The satisfaction level of a NFR refers to the extent to which that NFR has been satisfied as a consequence of an action performed by the SAS and its level of satisfaction at the previous time step of execution. The satisfaction level can be represented by a conditional probability distribution P(NFRi is satisfied | action a).

– *Priority* The priority value for a NFR is a scalar cardinal value used to represent its importance for satisfaction at runtime. The priority for satisfaction of NFRs may change due to the change in environmental conditions at runtime.

2.1.2 Monitorables

As the environment in which the SAS is operating is continuously changing, a SAS continuously monitors these changes over time by using monitorables that represent information about the state of the environment. For example, the interference and traffic load on the network links can be monitored.

2.1.3 Actions

Actions are defined as the adaptation strategies comprising of discrete set of software configurations, solutions or service components that is selected by a SAS during decision-making [5,49]. To achieve the target functional goals along with satisfying the NFRs, the SAS selects an action [3,38] based on the monitorable values and the satisfaction levels and priorities for the NFRs at a given point of time. The adaptation actions performed by SASs have an impact (positive or negative) on the satisfaction levels of NFRs.

2.2 POMDPs

The significance of POMDPs to SASs is that they are used to solve sequential decision-making problems under uncertain and dynamic environmental contexts [45,56]. They offer an agent-based decision-making approach by considering the agents working in a partially observable environment. This means that the agent cannot directly observe the underlying state. Instead, to choose the optimal action, it must maintain a probability distribution over the set of possible states, known as a belief, based on a set of observations and observation probabilities.

An exact solution to a POMDP yields the optimal action for each possible belief over the possible states. The optimal action maximizes the expected reward of the agent over a possibly infinite horizon. The sequence of optimal actions is known as the optimal policy of the agent for interacting with its environment. The basic elements of a POMDP are shown in Fig. 1.

A POMDP is specified as a tuple $<$S,A,Z,T,O,R,$\gamma >$ where S represents the set of states specifying a description of the state of the environment; A is the set of Actions that the agent can select to perform at a particular time; Z is the set of observations specifying the information received by the agent from the environment using sensors related to the set of states S; T is the transition function $T(s, a, s') = P(s'|s, a)$ specifying the probability of moving to the next state s' given an action a and current state s; O is the observation function $O(s, a, z) = P(z|s, a)$ specifying the probability of observing the observation z given an action a and resulting state s; R is the reward function R(s,a) specifying a scalar real value generated by the environment as a feedback of the action a performed by the agent provided with the state s

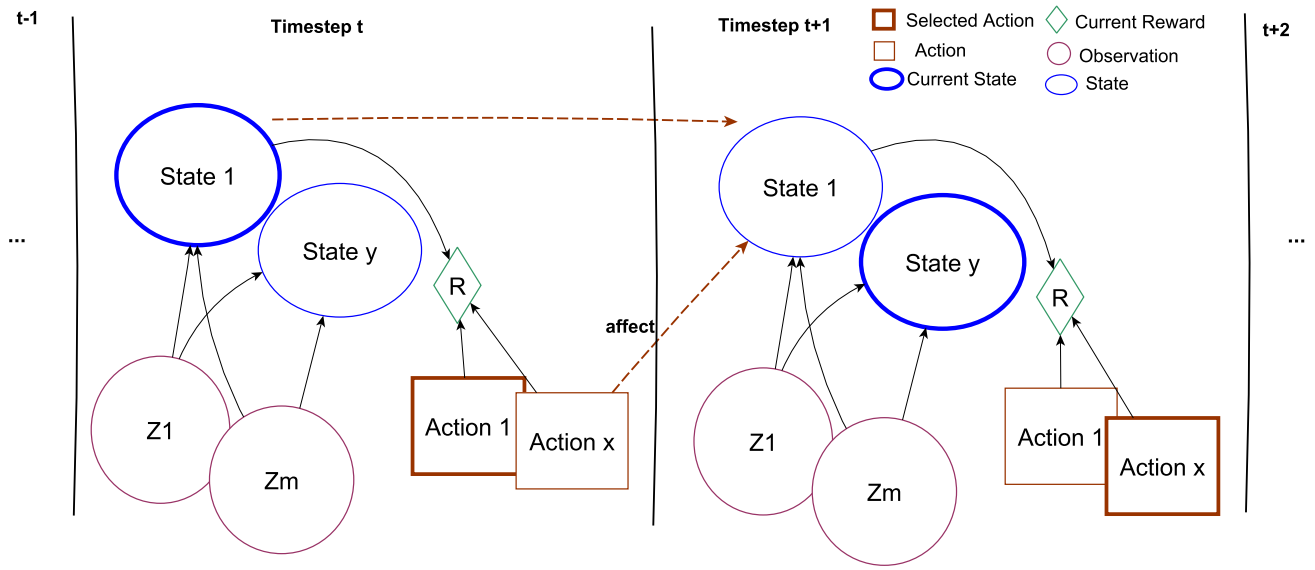


Fig. 1 POMDP

of the environment; γ is the discount factor to support the preference for the immediate reward values over the future rewards. The value of the discount factor lies between 0 and 1. The values for rewards are normally assigned by the domain experts at design time based on the information that is provided to them at design time or from the previous experiences [23,34,36].

The decision-making agent tries to find the policy π, a mapping from the state of the environment to action, that maximizes the value function, i.e. the *expected utility value* of the sum of discounted rewards, as follows:

$$V_\pi = E_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} \ldots | s_t] \tag{3}$$

Hence, based on the current state s_t of the system, the value function V_π is used to compute how much cumulative reward, discounted by γ, we can expect to get in future, if a particular action is performed in a particular state at time t.

Thus, the reward value R(s,a) is used to evaluate the effect of performing an action a during the state s with the help of a value function. Therefore, a *cardinal scale* is assigned to each decision made during a specific state of the system to indicate its priority.

As the states in a POMDP are not fully observable, a belief b over the states of the system is maintained. A POMDP offers the capability to quantify uncertainty in terms of the (partially) observed state of the environment. In reference to point-based planning methods [53,57] for solving POMDPs, that focus on the computation of a policy based on a sampling of points from the belief space, the value function over the belief V_b is represented by a set of α-vectors A. Each α-vector is associated with an action a and has a length of $|S|$ in order to provide a value for each state s. The α-vector is represented as follows:

$$\alpha_a = [V(s_i), V(s_{i+1}), \ldots, V(s_{(n)})] \tag{4}$$

Here, $V(s_i)$ represents the value of the value function for state s_i given a total number of n states.

Thus, given A, the value over the belief is computed as:

$$V_b = \max_{\alpha \in A} b.\alpha \tag{5}$$

Therefore, for each belief b, a set of α vectors A provides a policy π_A for the action that maximizes the value. The application of selected actions by the decision-making agent leads to a change in the state of the system.

2.3 MR-POMDPs

MR-POMDP [44,54,55], similar to POMDPs, is a sequential decision-making technique used to solve multi-objective decision problems. MR-POMDP is actually a POMDP that has more than one reward values represented in the form of a vector-valued reward function **R** as shown in Fig. 2. In MR-POMDPs, each objective (NFRs in our case) is associated with its own separate reward value. Hence, the size of the reward vector is equal to the number of objectives. As a consequence, the value function, given an initial belief $\mathbf{V}_{b_0}$ for the policy π of the MR-POMDP, is also a vector. Thus, each single element in the value function vector represents the *expected utility value* associated with each separate objective. As a result, it evaluates the effect of performing an action on the satisfaction of that objective given a particular state. The values of the elements in the value vector,

associated with each objective, create a relative ranking for the objectives. Hence, these *expected utility values* represent the priority of objectives for their satisfaction, while making the decision. We use this built-in capability of MR-POMDPs to compute *expected utility value* for each individual objective as a base for the *autonomous tuning* of the priorities at runtime.

As **R** is a vector, each element in the α-vector is also a vector as a result creating an α-matrix, $\boldsymbol{A}$. Each row in the α-matrix represents the values for the objectives in a particular state. The multi-objective value of taking an action a associated with alpha matrix $\boldsymbol{A}$ under a belief b is computed as follows:

$$\mathbf{V}_b = b\mathbf{A} \tag{6}$$

As the value function is represented as a vector in MR-POMDP, there may be multiple policies. The value functions of these multiple policies can be considered as optimal on the basis of the different priorities associated with the objectives. In order to select the best optimal policy from these multiple policies, a scalarization function f($\mathbf{V}_b$,W) is used to scalarize the value vectors $\mathbf{V}_b$ with respect to the weights W, and computed by the agent, corresponding to the objectives [43] and computed as follows:

$$f(\mathbf{V}_b, W) = W.V_b = w_i V_{b_i} + w_{i+1} V_{b_{i+1}} \ldots + w_n V_{b_n} \tag{7}$$

where w_i and V_{b_i} refer to the weight and value for the ith objective given n number of total objectives. The size of the weights vector W is also equal to the number of the objectives. In this paper, the weights vector values are computed using the Optimistic Linear Support (OLS) algorithm [44] at runtime.

Hence, for a given belief b, α matrix for each action and weight w, we can compute the policy π_A that takes the maximal value using Eqs. 6 and 7 as:

$$V_b^*(w) = \max_{\mathbf{A} \in A} b\mathbf{A}W \tag{8}$$

Hence, in MR-POMDP the reward vector is used to represent the priorities of objectives (NFRs in our case) by indicating their desirability in terms of their satisfaction given a particular state of the environment. The reward vector has to be initialized with estimated values. These values are assigned by domain experts and should reflect the experts' knowledge and the information they have available to them. The design-time assignment of priorities is a normal requirements practice. However, it is difficult to get right and particularly so if, as is often the case for SASs, the priorities assigned to a set of requirements may not be appropriate for all contexts the system encounters at run-time. Nor is the deployment of expert knowledge always easy, as highlighted in [58] where consensus between experts proved hard to achieve. To mitigate these difficulties, Pri-AwaRE permits requirements' priorities to be revised dynamically at runtime in a principled way. Nevertheless, the NFRs' satisfaction may be compromised if the initial, estimated reward values are poorly chosen. In our work, we have used simulations [21,50] to derive good initial values for the rewards.

2.4 Optimistic Linear Support

The Optimistic Linear Support (OLS) algorithm is based on Cheng's Linear Support [11] approach for solving POMDPs. OLS presented as Algorithm 1 follows an outer loop approach that creates an outer shell around a MR-POMDP solver (Persues[1]) [57] to create a solution set known as the Convex Coverage Set (CCS) X [43] to represent the collection of value vectors $\mathbf{V}^{\pi}$ (specifying the multi-objective values) and their associated policies π such that after performing scalarization a maximizing policy is in the set. In order to select the policy having the maximizing value $V_X^*(w)$, linear scalarization of the value vector is performed using the parameters in the form of weights vector.[2] OLS helps in finding these weights intelligently at runtime. The OLS algorithm considering a two objective problem is presented in Fig 3.

The OLS algorithm starts by taking an empty set of X denoting the CCS of value vectors as shown in line 1 of Algorithm 1. The algorithm repeatedly executes steps 2 to 9 until no improved value vectors are found evaluated by the Maximal Possible Improvement Δ [42,44]. In the first two iterations of the while loop, the algorithm selects the first two corner points as the extrema of the weights simplex, i.e. w_a = 0.0 and w_b =1.0 as represented by the red vertical lines in Fig. 3. Next, the value vectors, for example V_a = [8,1] and V_b =[2,7], for these corner points (w_a and w_b) are computed using a MR-POMDP solver represented by the blue lines in between these corner points. On the basis of these value vectors, a new corner point (w_c) is identified at their intersection. Then, Maximal Possible Improvement Δ [44] is computed for w_c. If Δ is improving then for this new corner weight w_c, a new value vector V_c = [6,5] is calculated by the calling the MR-POMDP solver as shown in Fig. 3. More corner points are generated such as w_d and w_e from the intersecting points of the existing value vectors. Again Maximal Possible Improvement Δ for each w_d and w_e is calculated. The corner weight (out of w_d and w_e) having

[1] Persues is a point-based planning technique for solving POMDPs.

[2] Considering a two-objective problem, the value of one of the weights can be computed as one minus the other weight due to the application of linear scalarization as $w_1 = 1 - w_2$, where w_1 and w_2 refer to the scalarization weights for the values associated with objective 1 and objective 2, respectively.

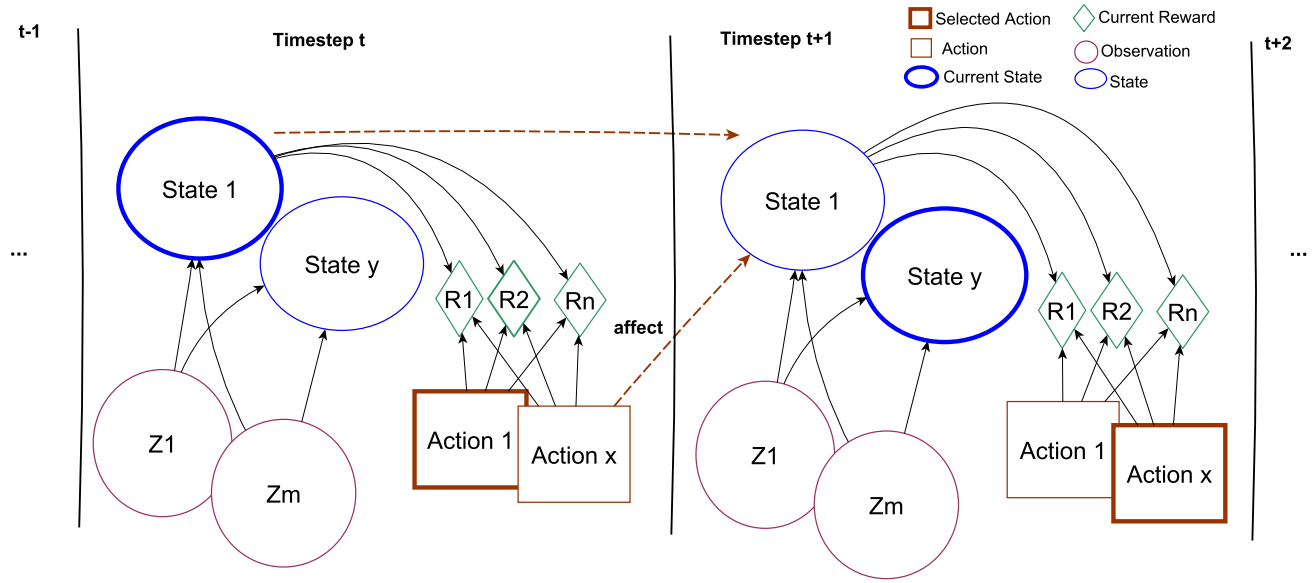


Fig. 2 MR-POMDP

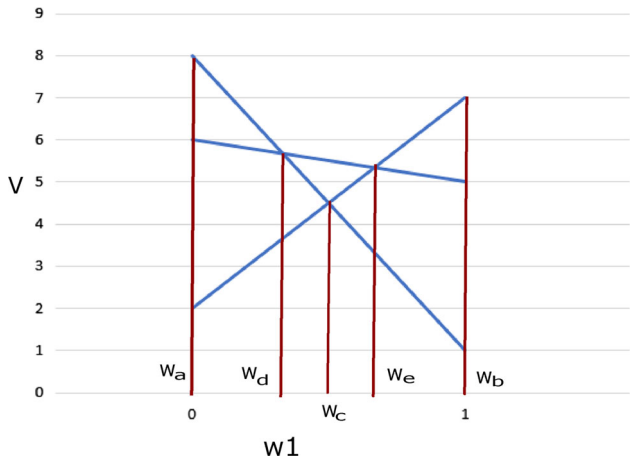


Fig. 3 Optimistic Linear Support

Algorithm 1: Optimistic Linear Support

Result: Convex Coverage Set X
$X \leftarrow \emptyset$;
while $\neg(\Delta.isMaximum())$ **do**
  Select w ; // Select Corner Point w
  // Calculate Value Vector V and π
  $(V, \pi) \leftarrow MRPOMDPSolver(w)$;
  $Calculate\Delta$;
  if $\Delta.isMaximum()$ **then**
    $X \leftarrow X \cup V$;
  end
end

high Δ is selected and MR-POMDP solver is called again to compute the value vector for the selected weight. The process is repeated until none of the remaining corner weights yield an improvement in the form of Δ.

The process returns a set X known as CCS of the value vectors and their associated policies. As we have more than one policy returned, we have to select the best policy on the basis of scalarization function by taking weight values generated as part of the OLS algorithm. The policy having the maximum scalarized value $V_X^*(w)$ is then selected. The flow chart representing the step by step execution of OLS algorithm is shown in Fig. 4. More details on OLS and MR-POMDP solver can be found at [48].

The OLSAR[3] algorithm is an extension of OLS algorithm. It follows the same steps as OLS but it reuses the alpha matrices from previous iterations to compute the approximate Convex Coverage Set (CCS) of value vectors.

2.5 MAPE-K architecture

The MAPE-K control loop is an architectural blueprint for autonomic computing systems and was first introduced by IBM as a vision about Autonomic Computing [25]. As SASs represent a specialized form of autonomic systems, they also employ the MAPE-K architecture loop. This architecture consists of the managing system and the managed system. The managed system corresponds to the application logic while the managing system corresponds with the decision-making agent, which makes use of the feedback loop with the phases Monitor-Analyse-Plan-Execute that run over a knowledge base. It is known as the MAPE-K loop for short. During the Monitor phase, the managing system collects data

[3] More details on the OLSAR and point-based planning techniques are provided in [48].

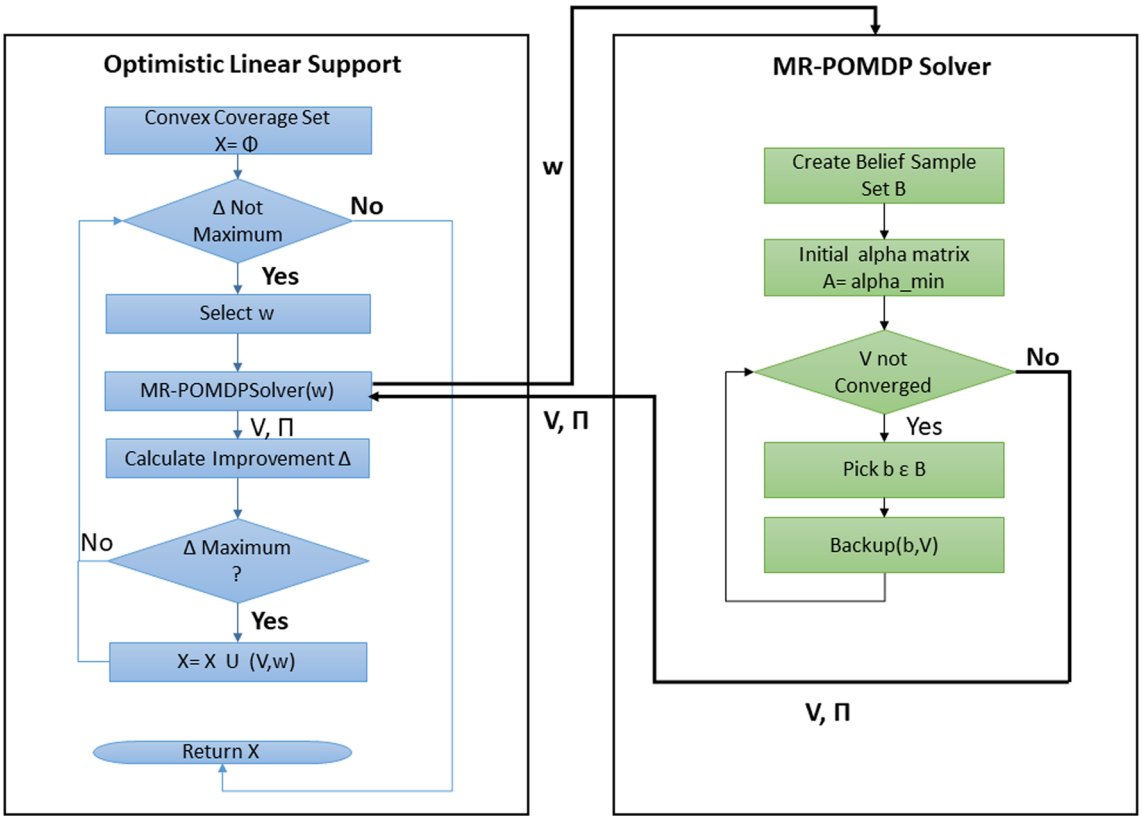


Fig. 4 Step By step execution of OLS

from the managed system supported by sensors connected to the managed system. In the Analyse phase, the managing system analyses the monitored data to check if adaptation actions are required. If an adaptation action is required, the decision-making agent plans for the adaptive actions that will be performed by the managing system. The goal is to achieve operational goals exhibited by the managed system along with satisfaction of NFRs during the Plan phase. Lastly, during the Execute phase, the planned actions are carried out through the actuators over the managed system. The knowledge K of the MAPE-K loop represents the data required by the managing system to execute all the loop. The runtime model based on the MR-POMDP is embedded in the MAPE-K architecture loop to underpin decision-making for SASs.

3 Pri-AwaRE: self-adaptive architecture

This section presents the Pri-AwaRE architecture to support runtime decision-making for SASs. The proposed architecture makes use of MR-POMDP, as a priority-aware runtime model, as part of the phases of the MAPE-K loop.

Next, we present the Priority-Aware MR-POMDP++ and its use as part of the proposed self-adaptive architecture to support decision-making in SASs using an illustrative example of the IoT network.

3.1 Illustrative example

In order to illustrate our proposed approach, we consider the example of a self-adaptive Internet of Things (IoT) [2,61] network. As a concrete application, we consider the simulating environment of DELTA-IoT an exemplar SAS representing an IoT network for a smart campus [21]. The simulator represents a multi-hop IoT network consisting of 15 motes, including RFID sensors, passive infrared sensors and temperature sensors, distributed across the various buildings of KU Leuven campus. These motes, based on LoRa (Long-Range) radio communication, are deployed in each building to provide access to the laboratories, monitor the occupancy status and sense the temperature. The motes communicate with each other to fulfil the functional goal of relaying information to the central gateway deployed at the central monitoring facility of the campus. The IoT networks are required to survive for a long period of time on a single battery along with maintaining communication reliability. Hence, the main goal for an IoT network is to increase the lifetime of the network by satisfying the NFRs of Minimization of Energy Consumption (MEC) and Reduction of Packet Loss (RPL)

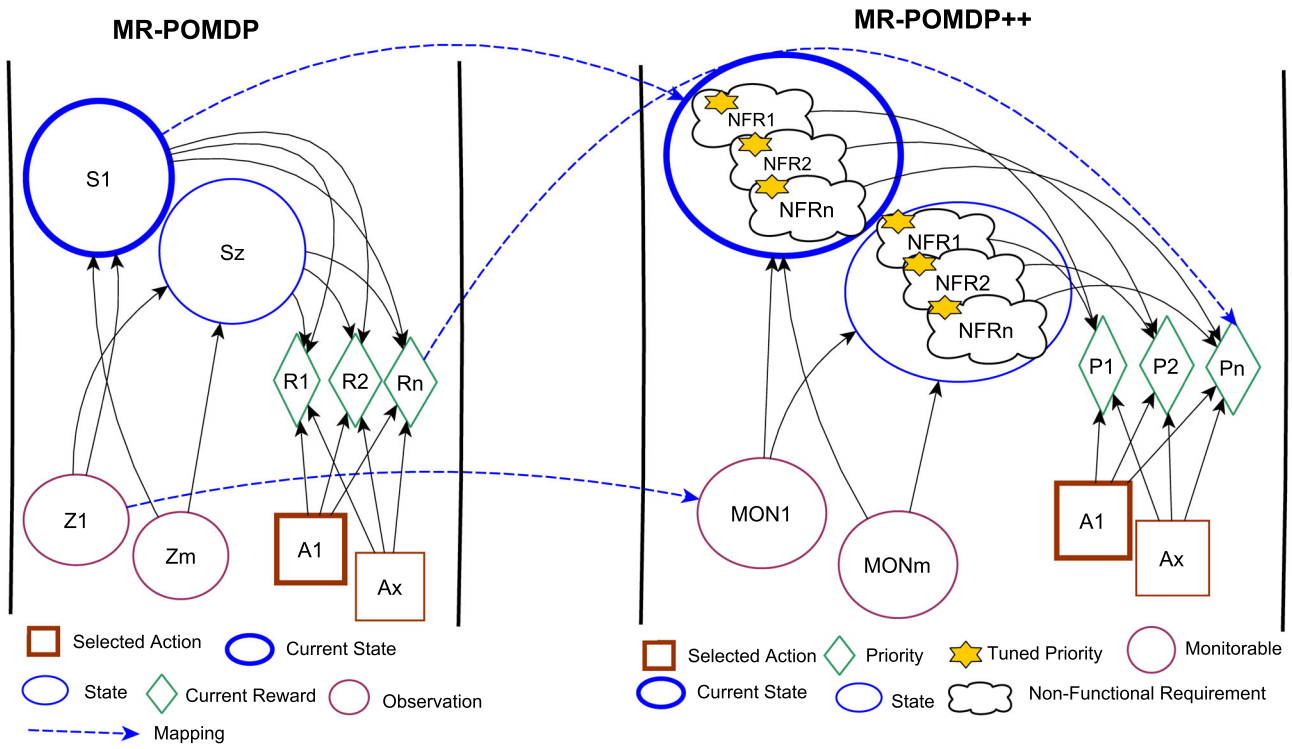


Fig. 5 Mapping of MR-POMDP to priority-aware MR-POMDP++

under uncertain environmental contexts of communication interference and dynamic traffic load. For the purpose of satisfaction of NFRs under uncertain environmental conditions, the network is required to tune network link settings such as the communication range, transmission power and distribution factor for links using different adaptation strategies.

3.2 MR-POMDP++

In this section, we present MR-POMDP++, a priority-aware runtime model, to represent the priorities and satisfaction levels of NFRs to perform decision-making in a SAS as shown in Fig. 5. Next, we present the rules for NFRs representation in the form of MR-POMDPs to support runtime decision-making in SASs.

(1) NFR satisfaction and MR-POMDP states

In order to achieve satisfaction of NFRs, a SAS is required to take adaptation actions that can have different effects (good or bad) on their satisfaction. Therefore, NFRs cannot be labelled as fully satisfied nor fully violated. Due to the lack of crispness in the nature of satisfaction of NFRs, the satisfaction levels of NFRs cannot be represented as an absolute value of True or False [19]. Therefore, the satisfaction levels can be modelled in the form of probability distributions such as P(NFR = True) and a NFR is considered as satisfied if it meets an acceptability threshold constraint defined by the design experts [58]. For example, in IoT case, the satisfaction level of RPL can be specified as P(RPL = True) = 0.9 or P(RPL = True) = 0.4. The RPL can be considered as highly satisfied if the P(RPL = True) > = 0.8 where 0.8 can be considered as the required threshold constraint.

Such specification of satisfaction levels of NFRs can be represented using the states of MR-POMDP. In MR-POMDP++, we consider each state to represent the set of combinations of satisfaction levels of NFRs. As states in MR-POMDP are not directly observable, a belief (i.e. a probability) over each state is maintained. Hence, the satisfaction levels of NFRs can be expressed in the form of marginalized probability distributions $P(NFR_i = True)$ where NFR_i belongs to the set of NFRs [38].

On the basis of this description, we derive a mapping rule as follows:

Rule: 1 *The state $s \in S$ in MR-POMDP++ represents the set of combinations of satisfaction levels of the non-functional requirements $(NFR_1 \ldots NFR_n)$. As the states in the MR-POMDP++ are partially observable, the satisfaction levels of the NFRs can be represented in the form of probability distributions $P(NFR_i = True)$.*

These probabilities can be used to conclude if the satisfaction levels meet the acceptability thresholds.

Table 1 States of the IoT network in terms of NFR

S	NFR_1 = MEC	NFR_2 = RPL
s_1	True	True
s_2	True	False
s_3	False	True
s_4	False	False

Using Rule 1, the total number of states in terms of the satisfaction levels of NFRs in MR-POMDP++ can be computed as: $|S| = |2|^{|NFR|}$ where $|S|$ corresponds to the size of set S, $|NFR|$ corresponds to the number of NFRs, and *2* corresponds to True and False. For example, in an IoT network, if we consider 2 NFRs of Minimization of Energy Consumption (MEC) and Reduction of Packet Loss (RPL), so the number of states for MR-POMDP++ will become $|S| = 2^2 = 4$ as shown in Table 1.

2) NFR priorities and rewards vectors

During the decision-making process in SASs, the adaptation decisions should take into account the satisfaction priorities of individual NFRs. Priorities of NFRs indicate their level of importance for satisfaction. The higher the priority, the more important it is to satisfy that NFR at a particular point of time.

MR-POMDPs facilitate the modelling of these individual NFR priorities using a reward vector. In MR-POMDPs, a vector-valued reward function is used to associate a separate reward value with each objective (a NFR in our case). The reward values are generated as a feedback signal according to the decisions (adaptation actions) taken by MR-POMDPs. The reward value associated with a particular objective indicates the effect, either positive or negative, of performing an action on the satisfaction of that objective. Hence, the reward vector values specify a relative ranking of the objectives (NFRs) in terms of the *cardinal* effect that an action will have on the satisfaction of that objective (NFR) under an uncertain environmental condition. Consequently, a higher reward value for an objective indicates its higher priority (importance level) which is taken into account when an adaptation decision is taken by MR-POMDPs.

For example, in an IoT network, if the communication interference at a particular point of time is high, the decision-making agent might select the adaptation action of increasing transmission power (ITP) in order to support the NFR of RPL. However, increasing the transmission power might have a negative effect on the energy consumption. Therefore, given the current environmental context of high link interference, on the basis of the selected action of ITP, the system will generate an immediate reward for the RPL (e.g. 75) that is higher than the reward for the MEC (that could be, e.g. − 50). The reward value for RPL is higher than the reward value of MEC because it is more important to satisfy RPL at this point of time according to the MR-POMDP, given the current conditions.

Hence, the reward values represented in the form of a reward vector show a relative ranking of the NFRs in terms of the effect that the action will have on their satisfaction as follows:

$$\mathbf{R(s,a)} = [R_{RPL}, R_{MEC}] = [75, -50]$$

The reward vector values are initially assigned by the requirements engineers or domain experts [58]. In MR-POMDP++, we consider these reward values as the initial expert defined priorities for NFRs.

Using these concepts, the priorities representation for NFRs in MR-POMDP++ is explained by the following rule:

Rule: 2 *The values in the reward vector* ***R(s,a)*** *over the execution of an action* $a \in A$ *given state* $s \in S$ *in MR-POMDP++ represent the priorities of non-functional requirements (NFRs).*

$$\mathbf{R(s,a)} = [R_{NFR1}, R_{NFR2}, \ldots R_{NFRm}]$$

where R_{NFR1} represents the reward for NFR1 corresponding to the priority value of NFR_1 and so on.

(3) Expected utility values and autonomous tuning of priorities

In SASs, the priorities of NFRs may vary according to the changes in the environmental context. The ability to autonomously adapt to the context is what MR-POMDP++ offers. For example, if energy conservation is the most highly prioritized NFR, but at some point in time the batteries become fully charged, then rewarding that NFR's satisfaction may no longer be the optimal behaviour for the system. In order to deal with such situations, MR-POMDP++ with help of reward vectors offers an opportunity for *autonomously tuning* the individual NFRs' priorities. This tuning of priorities is done by the computation of the separate *expected utility value* for each NFR, during the operation of MR-POMDPs (using Eq. 3) as follows:

$$V_{NFRi} = E_\pi[Ri_t + \gamma Ri_{t+1} + \gamma^2 Ri_{t+2} \ldots | s_t] \tag{9}$$

where V_{NFRi} and Ri represent the *expected utility value* and reward values for NFRi. As presented in Eq. 9, the rewards, representing the initial expert defined priorities, are used for the computation of the distinct *expected utility value* for each NFR at runtime.

Hence, these *expected utility values* represent the newly *tuned priority values* of the different NFRs. The *expected utility values* consider the individual effect of performing an action on the satisfaction of an individual NFR given an uncertain environmental context, and are considered while making the decisions at runtime. More details about the autonomously tuned priorities are provided in Sect. 4.

Hence, MR-POMDP++ as a runtime specification model (***S***) *takes into account the new knowledge* (**K'**) *about the priorities of the individual NFRs to perform runtime decision-making for SASs for the purpose of conformance to the Requirements* (***R***).

Next, we present the proposed self-adaptive architecture, Pri-AwaRE, that makes use of Priority-Aware MR-POMDP++ to perform decision-making for SASs.

3.3 Architecture for decision-making in SASs

The proposed Pri-AwaRE architecture for SASs is inspired by the feedback architecture of the reinforcement learning (RL) process based on the interactions between the decision-making agent and the environment in which the decision-making agent operates [31]. On the basis of the observations monitored about the current state of the environment, the decision-making agent performs an action in order to achieve the desired goal. This process repeats continuously. Considering the above, we define an architecture for the SASs consisting of two components corresponding to the managed system (i.e. the environment in RL process) and the managing system (i.e. the decision-making agent in RL process), respectively, as shown in Fig. 6.

The Pri-AwaRE architecture structures the *managing system* (based on feedback loop) interacting with the *managed system* using probe and effector interfaces [50] as shown in Fig. 6. Our Pri-AwaRE approach focuses mainly on the aspects of the managing system.

In the next subsection, we present the components of the Pri-AwaRE architecture such as the managed and managing system (the decision-making agent). We also present the use of the MR-POMDP++ to support priority-awareness, by providing support to model and represent the satisfaction levels and priorities for NFRs.

(1) Managed system

The managed system represents the actual environment for which we want to implement self-adaptive capabilities. It is instrumented using probe and effector components that are used to send information to and from the managing system.

For example, an IoT network operating according to its predefined settings can be considered as a managed system. To add self-adaptive capabilities, we have to attach some external managing system that can interact with the network using some probing and effector interfaces [21,50]. More details on the interaction of the managing and managed system are provided in [50]. For our study, we have selected the remote data mirroring (RDM) and IoT network environments represented by the *RDMSim* [50] and *Delta-IoT* [21], respectively, to act as the managed system.

(2) Managing system

The managing system consists of the following components:

(a) Monitoring component Monitoring components of the managing systems use sensors of the managed system in order to get data regarding the monitorable values (e.g. communication interference and traffic load on the links in an IoT network). The monitored values are sent as an input to the MR-POMDP++ model in the form of observations.

(b) Analysis and planning components The analysis and planning components of the managing system make use of the steps of the MR-POMDP++ process as shown in Fig. 6. The MR-POMDP++ model takes observations from the monitoring component and the current belief over the states (maintained by MR-POMDP++) from its runtime ***Knowledge*** as an input. The MR-POMDP++ model analyses these values to perform planning for the adaptation actions for the fulfilment of target operational goals and satisfaction of NFRs, such as Minimization of Energy Consumption and improvement in packet delivery performance in case of the IoT network.

(c) Execution component The execution component takes the prompted action by MR-POMDP++ as an input and performs that action on the managed system using effectors or actuators of the managed system, to therefore meet the operational functional goals and satisfy NFRs to comply to the *requirements* (***R***).

d) Knowledge component: The knowledge component is based on the runtime knowledge maintained by MR-POMDP++ that is taken into account during the decision-making process.

The process is performed continuously during the SAS's execution.

Next, we present the application of Pri-AwaRE, our Priority-Aware self-adaptive architecture, to the example application of DELTA-IoT network.

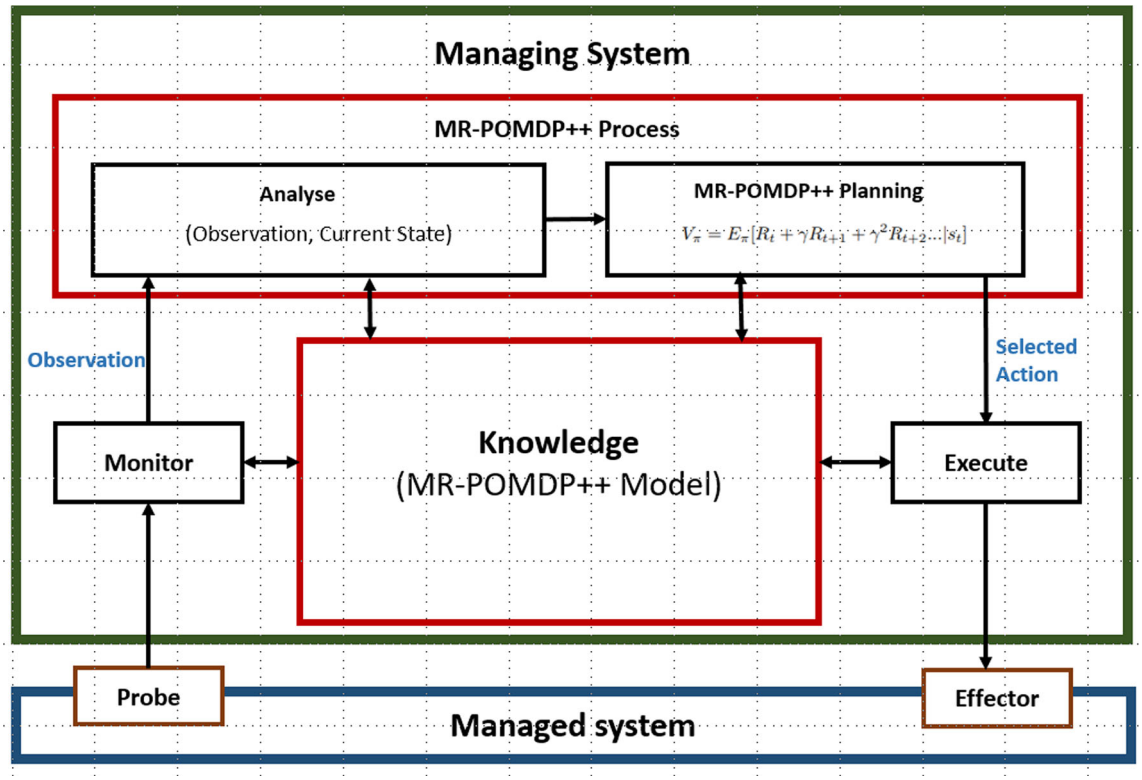


Fig. 6 Pri-AwaRE architecture

3.4 Pri-AwaRE architecture for decision-making in the IoT network

The Pri-AwaRE architecture for the case of IoT networks consists of the following two components:

(1)The managed system represents an IoT network. As a concrete example, we consider an IoT network for smart campus represented by the DELTA-IoT simulator [21].

(2) The managing system based on the constructs of MAPE-K loop and MR-POMDP++ consists of the following components:

(a) Monitoring component

At a given timestep, data from the motes is gathered by the monitoring component using the probe component (i.e. sensor) of the DELTA-IoT simulator. The probe collects information about the link interference in the form of Signal to Noise Ratio (SNR). The monitored SNR value is forwarded to the Analysis and Planning components.

(b) Analysis and planning components

To support priority-awareness, the steps of MR-POMDP++ process are used to support analysis and planning. The monitored SNR value is sent as an input, in the form of observation, to the MR-POMDP++. Then, the MR-POMDP++ analyses the observed SNR value and the current belief over the state maintained by the runtime model of the MR-POMDP++ (i.e. the Knowledge K) and plans for the selection of the next suitable adaptation strategy in the form of an action to be performed by the system. The knowledge component of MAPE-K loop comprises of the runtime knowledge maintained by MR-POMDP++.

The components of the MR-POMDP++ for the considered DELTA-IoT network are explained next.

States: According to Rule 1, for the two NFRs (MEC and RPL), four states are identified and are shown in Table 1.

Actions: represent the adaptation strategies to maintain the satisfaction of the NFRs RPL and MEC. These actions are Increase Transmission Power (ITP) and Decrease Transmission Power (DTP). ITP supports RPL by increasing the communication range [27,28] of the motes along with the adjustment of the distribution factor on the links. As the value of the communication range is directly proportional to the transmission power, increasing the communication range leads to improved packet delivery performance at the cost of high energy consumption [21]. The action DTP supports MEC by decreasing the communication range along with adjusting the distribution factor of the links. Decrease in the communication range leads to decrease in transmis-

sion power and, as a result, leads to lower level of energy consumption.

Rewards: There is a vector-valued reward function to represent the priorities of NFRs from the perspective of the experts. Therefore, the reward vector has a size of 2 represented as $\mathbf{R(s,a)} = [R_{MEC}, R_{RPL}]$. R_{MEC} represents the priority value for MEC and R_{RPL} represents the priority values for RPL at runtime. The reward vector values (provided by the experts) for NFRs in the DELTA-IoT network are shown in Table 2

Transition Function: According to Rule 1, the states are represented as a combination of NFRs in MR-POMDP++. The transition probabilities T(s,a,s′) are factored as marginal conditional probabilities of NFRs $P(MEC' \mid MEC, a)$ and $P(RPL' \mid RPL, a)$ using the property of conditional independence and Bayes rule [38] as follows:

$$T(s,a,s') = P(s'|s,a) = P(MEC' \mid MEC, a) \\ P(RPL' \mid RPL, a)$$

The transition probabilities for going from one state to another as a result of action for the IoT case are shown in Table 3. These transition probabilities are provided by the experts.

Observations As the states of NFRs are not directly observable, we use monitorables to obtain observations required for monitoring the NFRs' satisfaction levels based on the information obtained from the environment. In case of the IoT network, we consider the monitorable SNR to observe link interference [21]. Depending on the possible set of values of SNR, i.e. being less than, greater than or equal to zero, we have three types of observations. The higher the value of SNR, the stronger will be the signal strength of the link Therefore, it indicates less interference and vice versa.

Considering the given set-up, the MR-POMDP++ serves as a priority-aware model to analyse the given observations and plan for the next suitable action.

(c) Execution component

The execution component executes the selected action on the managed system by using the effector component of the DELTA-IoT simulator.

4 Experimental evaluations

We have conducted experiments using two different example applications from the networking domain as a proof of generalization of application of our Pri-AwaRE approach. The example applications are selected on the basis of the type of decisions, i.e. *Global* or *Local* considering a network environment that is composed of nodes. *Global* decisions have an effect on the entire network configuration, whereas *local* decisions affect the link configurations associated with a particular node.

The first example application that is presented is based on the decision-making in a self-adaptive IoT network. In the IoT network, *local decisions* are taken for each sensor based on *local* contextual information retrieved from the links associated with them. On the other hand, our second example application is related to a self-adaptive remote data mirroring (RDM) network. In the RDM system, *global decisions* for topology change are taken based on the analysis of the *global* contextual information (such as total number of active network links) to satisfy the NFRs. We have also compared our Pri-AwaRE approach with the existing single-objective approaches of RE-STORM [38] (implemented using the DESPOT and Perseus solvers) and RE-STORM-ARROW [39]. The experiments were performed on a Lenovo Thinkpad with intel Core i7, 8th Gen processor and 16 GB RAM. A complete account of the results associated with the RDM case study is reported in [48].

4.1 MR-POMDP solver

As solving a MR-POMDP is a computationally intractable problem, we have used Optimistic Linear Support with Alpha Re-use (OLSAR) algorithm based on a point-based MR-POMDP solver, to generate approximate solutions by performing approximate backups while computing $\alpha-$vectors only for a set of sampled belief values. Hence, it has proven to scale well during the experiments performed. The reuse of the alpha matrix in the algorithm also makes it efficient.

4.2 Experimental hypotheses

In this section, we define our hypotheses for the experiments. Let us revisit the concept of priority-awareness as defined in Definition 1.

Based on the above concept of priority-awareness, the null H_0 *and alternative* H_a *hypotheses are described as follows:*

$\mathbf{H_0}$: *Pri-AwaRE does not improve decision-making under uncertainty in comparison with single-objective optimization techniques, which do not offer priority-awareness.*

$\mathbf{H_a}$: *Pri-AwaRE improves decision-making under uncertainty in comparison with single-objective optimization techniques, which do not offer priority-awareness.*

Next, we provide a description of the example applications and experimental evaluations to test the hypotheses using these example applications.

Table 2 Reward values for the NFRs in the DELTA-IoT network

S	Action (A)	Reward Vector Values	
		$R_{NFR1} = R_{MEC}$	$R_{NFR2} = R_{RPL}$
s_1	DTP	89	80
s_2	DTP	75.1	20
s_3	DTP	75	30
s_4	DTP	10	5
s_1	ITP	80	89.67
s_2	ITP	40	75
s_3	ITP	30	70
s_4	ITP	5	10

Table 3 Transition probabilities for NFRs

NFR_1 = Minimization of Energy Consumption(MEC)				
Action (A)	MEC_t	RPL_t	$P(MEC_{t+1} = True)$	$P(MEC_{t+1} = False)$
DTP	True	True	0.89	0.11
DTP	True	False	0.92	0.08
DTP	False	True	0.84	0.16
DTP	False	False	0.87	0.13
ITP	True	True	0.85	0.15
ITP	True	False	0.88	0.12
ITP	False	True	0.73	0.27
ITP	False	False	0.76	0.24
NFR_2 = Reduce packet loss (RPL)				
Action (A)	MEC_t	RPL_t	$P(RPL_{t+1} = True)$	$P(RPL_{t+1} = False)$
DTP	True	True	0.92	0.08
DTP	True	False	0.89	0.11
DTP	False	True	0.96	0.04
DTP	False	False	0.91	0.01
ITP	True	True	0.98	0.02
ITP	True	False	0.96	0.04
ITP	False	True	0.99	0.01
ITP	False	False	0.97	0.03

4.3 Example application 1: Local decision-making for IoT network

As our first example application, we have used the simulating environment of DELTA-IoT; an exemplar self-adaptive system representing an IoT network for a smart campus [21]. Next, we present the initial set-up for the experiments using the DELTA-IoT exemplar.

4.3.1 Initial set-up

In the DELTA-IoT network, each timestep corresponds to a 15 min of network activity [21]. For the current set of experiments, at each timestep, we execute the MR-POMDP++ model for each mote (having mote ids from 2 to 15) individually to make the *local decisions* of performing the action ITP or DTP according to their monitored link interference values. For experiments with DELTA-IoT network, we focus on the NFRs of MEC and RPL which are representation of the NFRs concerned with the quality and performance [19] of the network. For the initialization of the MR-POMDP++ model, the states, rewards vector values and transition function probabilities are presented in Tables 1, 2 and 3, respectively. Further, to evaluate the approach, we have compared the results in terms of the real values from the DELTA-IoT simulator for the satisfaction of 2 NFRs MEC and RPL.

4.3.2 Requirements specification

Defined by the experts [21], following set of Requirements (***R***) regarding the satisfaction levels of NFRs for DELTA-IoT network are considered:

R1: *Total energy consumption in the network should be less than or equal to 20 coulombs, i.e.* $SAT_{MEC} <= 20$

R2: *Total packet loss in the network should be less than or equal to 20%, i.e.* $SAT_{RPL} <= 0.20$

4.3.3 Experiments

The experiments have been performed to demonstrate the following two cases for evaluation of the hypotheses:

Case 1: Priority-aware decisions and autonomous tuning of NFRs' priorities

We have executed the MR-POMDP++ model for each mote individually in the network (having mote ids from 2 to 15) at each timestep. The model monitors the link interference on the outgoing links for a particular mote and takes the decision of increasing or decreasing the transmission power on the links by increasing or decreasing the communication range. So the model takes a *local decision* related to each mote at a particular timestep in order to configure its corresponding links as shown in Table 4. For example, first at timestep t1, all the links for mote 2 are configured as a result of the action of DTP. As a consequence, the satisfaction level for MEC becomes *33.961559* and a satisfaction level for RPL becomes *0.041667*, respectively. The model is then executed for mote 3, mote 4 and so on.

The initial NFRs' priorities represented in the form of rewards are taken into account while taking the adaptive decisions of action selection. Let us study how the priorities of NFRs represented in the form of rewards have an impact on the decision of action selection for the purpose of link configurations for a particular mote in DELTA-IoT network as shown in Table 4. Considering the case of timestep t1, the *expected utility values* for NFRs are considered during the decision-making process at runtime. For example, for mote 2 the *expected utility value* for MEC has a higher value of *764.171898* than that for RPL having a value of *605.188297*. As the *expected utility value* supports the MEC, so the action selected for mote 2 is DTP that supports MEC. In contrast, for the configuration of the links of mote 7, the action of ITP is selected on the basis of the higher *expected utility value* of *788.205554* for RPL than the *expected utility value* of *650.420788* for MEC. This decision of *ITP* increases the satisfaction level of RPL from 0.083333 to 0.0% representing no packet loss as shown in Table 4. Hence, it shows that *expected utility values* that represent the *autonomously tuned NFRs' priorities* has an impact on the decision of action selection leading to *better-informed priority-aware* decision-making. Given the above, it provides evidence that Pri-AwaRE supports priority-aware decision-making.

As a result of all the tuning configurations done for all the motes at the end of timestep t1, the satisfaction of MEC and RPL becomes *33.083955* and *0.009091*, respectively. The same procedure is repeated at each timestep leading to achieve higher levels of satisfaction for both MEC and RPL as shown in Fig. 7. Hence, our approach shows promising results in terms of satisfying both MEC and RPL.

Case 2: Impact of priority-aware decisions on satisfaction levels of NFRs

We have also studied the impact that the Pri-AwaRE approach, performing *priority-aware* decisions, has on the satisfaction levels of NFRs of the DELTA-IoT network as compared to the network performing without any adaptive approach. Furthermore, we also compare our results with the approach of RE-STORM [38] implemented using Perseus [57], a single-objective POMDP solver.

During the execution of the simulator, without adaptation, for the DELTA-IoT network the simulator focuses on the satisfaction of RPL, at the cost of high value of energy consumption, by keeping the overall packet loss less than the specified threshold of 20% as shown in Fig. 7. In this case, the satisfaction for MEC ranges between *25.0* and *44.0* coulombs which is quite higher than the satisfaction threshold. On the other hand, by applying the adaptation mechanism offered by our Pri-AwaRE approach, DELTA-IoT network showed an improvement in terms of compliance to the *requirements* of $SAT_{MEC} <= 20.0$ and $SAT_{RPL} <= 0.20$.

Moreover, our approach also shows better satisfaction levels of NFRs of MEC and RPL in comparison with the network working under the adaptive decision-making offered by RE-STORM. Let us observe Fig. 7, Pri-AwaRE shows promising results in terms of satisfaction RPL as compared to RE-STORM. RE-STORM shows higher levels of packet loss than Pri-AwaRE by having a packet loss above the satisfaction threshold of 0.20% more frequently as shown in Fig. 7. On the other hand, Pri-AwaRE shows comparable results to RE-STORM in terms of satisfaction of MEC. In case of Pri-AwaRE, the satisfaction level of MEC remains below or closer to the satisfaction threshold at almost all of the timesteps. The satisfaction of MEC starts with quite high value of *33.083955* coulombs at the first timestep. But later on, it shows an improvement in the satisfaction of MEC by achieving a satisfaction level of *17.226979* coulombs at second timestep and thereby complying to the *requirements specification* of $SAT_{MEC} <= 20$ as shown in Fig. 7.

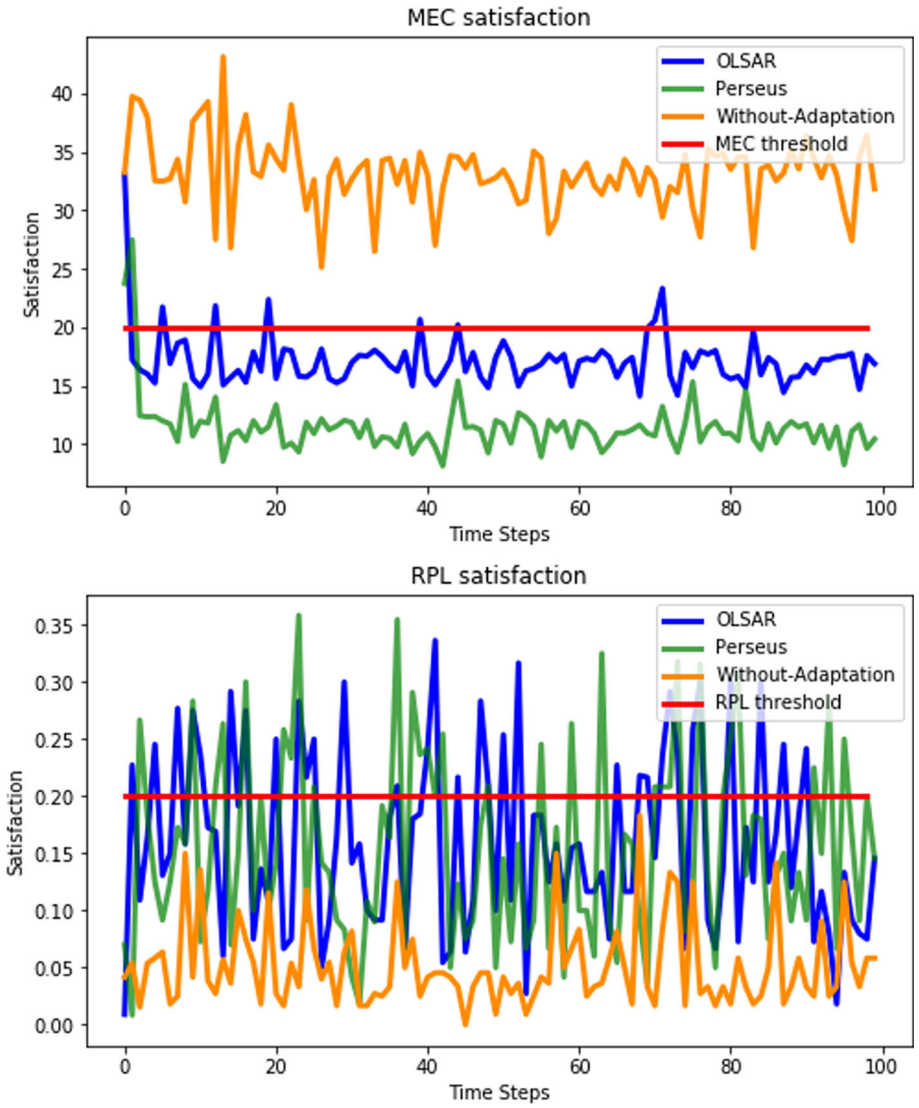


Fig. 7 Satisfaction of NFRs over time without adaptation, by applying Pri-AwaRE (using OLSAR) and RE-STORM (using Perseus)

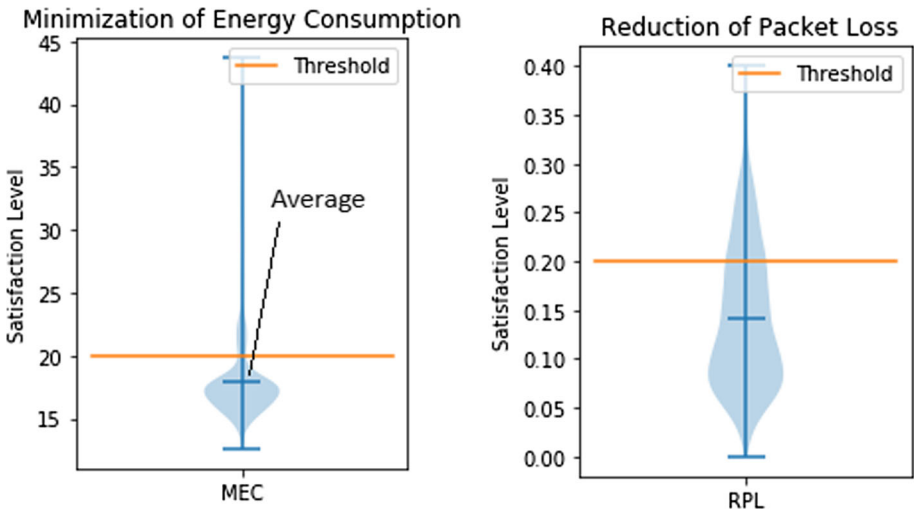


Fig. 8 Average satisfaction of NFRs

Table 4 Experiment Results for timestep 1

Mote Id	Action	Val_{MEC}	Val_{RPL}	Sat_{MEC}	Sat_{RPL}
2	**DTP**	**764.171898**	**605.188297**	**33.961559**	**0.041667**
3	ITP	650.060976	788.044468	38.485869	0.008333
4	DTP	788.335155	642.632839	36.958176	0.015385
5	ITP	650.420667	788.205516	37.720203	0.030769
6	DTP	788.338781	642.644331	29.817813	0.141667
7	**ITP**	**650.420788**	**788.205554**	**31.967988**	**0.083333**
8	**DTP**	**788.338787**	**642.644351**	**33.204893**	**0.0**
9	ITP	650.420789	788.205554	37.024339	0.030769
10	DTP	788.338787	642.644352	30.900698	0.01
11	ITP	650.420789	788.2055545	36.969265	0.146154
12	DTP	788.338787	642.644351	32.772624	0.058333
13	ITP	650.420789	788.205554	37.853066	0.046154
14	DTP	788.338787	642.644351	43.682137	0.014286
15	ITP	650.420789	788.205554	33.083955	0.009091

Val_{NFR} represents the expected utility value of NFRi i.e. Val_{MEC} and Val_{RPL}
Sat_{NFR} represents satisfaction level of NFRi i.e. Sat_{MEC} and Sat_{RPL}

Table 5 Confidence intervals for average satisfaction levels of NFRs

Environmental context	NFR	Sat_{AVG}	Confidence interval	Standard error
Link interference	MEC	17.860959	17.6989–18.0229	0.0826
	RPL	0.141865	0.1381–0.1457	0.0019

Sat_{AVG} represents the average satisfaction level of NFR

Summary of findings in experiments

In summary, our Pri-AwaRE approach shows compliance with the requirements specifications as evident from the average satisfaction levels of MEC and RPL presented in Fig. 8. We have performed a confidence interval test to test our hypotheses. The reported results show a confidence level of 95% that the average satisfaction level of MEC lies between the confidence interval of 17.6989 and 18.0229 with a standard error of 0.0826. Similarly for average satisfaction level of RPL, the confidence interval lies between 0.1381 and 0.1457 showing a confidence level of 95% with a standard error of 0.0019 as shown in Table 5. As a consequence, it verifies the conformance to the requirements specification of $SAT_{MEC} <= 20$ and $SAT_{RPL} <= 0.20$. We can conclude that our priority-aware approach offers statistically sound results in terms of meeting the satisfaction levels of NFRs. Hence, this evidence rejects our hypothesis H_0 and satisfies our hypothesis H_a.

Discussion

From the results, it can be deduced that Pri-AwaRE shows a significant improvement in the satisfaction of NFRs as compared to both the network operating without an adaptive mechanism and the network having the decision-making support of RE-STORM approach. The usage of MR-POMDP++ helps in maintaining compliance against the *requirements* (***R***) for the DELTA-IoT network. The average satisfaction levels for the NFRs MEC and RPL, generated by Pri-AwaRE, are *17.860959* and *0.141865*, respectively, as shown in Fig. 8. Hence, the DELTA-IoT network is conforming to the *requirements specification* of $SAT_{MEC} <= 20$ and $SAT_{RPL} <= 0.20$. Hence, our approach shows comparable and sometimes even better satisfaction levels for NFRs than RE-STORM, which is representative of a single-objective approaches.

To further evaluate our results for the DELTA-IoT network, we have computed the extent of satisfaction (*ExS*) of the NFRs using the quantification tool of DeSiRE [14] as shown in Fig. 9. According to the DESiRE tool, the value of zero is considered as a satisfaction boundary between positive and negative degree of satisfaction represented by *ExS* values. In case of Pri-AwaRE, the *ExS* value for MEC remains positive at almost all of the timesteps representing positive degrees of satisfaction for MEC. With an exception of a few timesteps, the *ExS* goes below zero. Yet, this drop in the *ExS* is very minor and is quite close to the satisfaction boundary of zero. Moreover, Pri-AwaRE also shows promising results in terms of satisfaction of RPL by having *ExS* value above zero,

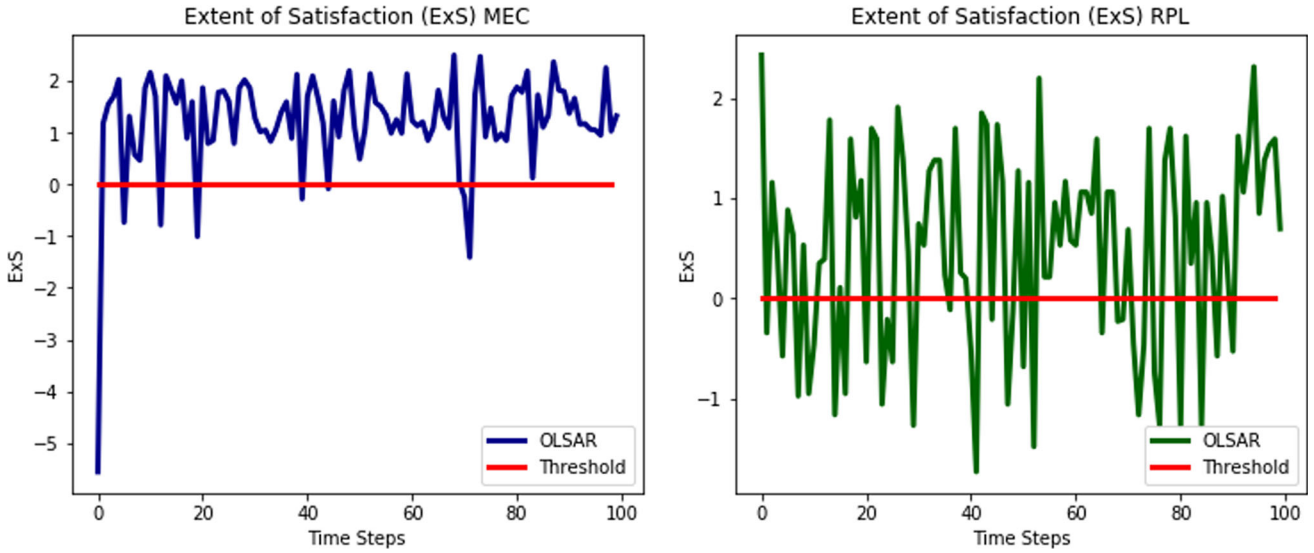


Fig. 9 Extent of satisfaction (*ExS*) of NFRs over time by applying Pri-AwaRE (using OLSAR)

which show positive degree of satisfaction for RPL. However, the *ExS* value for RPL goes below the zero value, indicating its negative degree of satisfaction. However, this deviation goes below to a maximum value ranging between -1.5 to -2.0 which is not that far from the satisfaction boundary of *ExS*. The analysis of *ExS* values creates an opportunity for RELAXation of a particular NFR [60] to temporarily benefit the satisfaction of others. For example, if MEC has a high *ExS* value at a particular timestep and RPL is slightly violated having a negative value for *ExS* that is closer to zero. Therefore, in such a situation, MEC can be RELAXed at that timestep to achieve better satisfaction level for RPL. This autonomous RELAXation of priorities made by the system is part of our future work.

Nevertheless, with no autonomous RELAXation at hand, and considering the analysis of the effects of the *autonomous priority tuning* on the decisions provided above, the Pri-AwaRE approach (based on MR-POMDP++) creates opportunities for the experts to consider the refinement of the initially defined priorities. The refinement would be done by taking into account the new *autonomously tuned priorities* provided by Pri-AwaRE to, therefore, improve the current specification of the requirements. This would take into account the newly found environmental contexts with the new corresponding sets of priorities, which were provided by the MR-POMDP++ model. This consideration can lead to a better assignment of initial expert defined priorities and thereby offer a significant improvement in the decision-making process of SASs in terms of satisfaction of NFRs. We are currently working in such specific cases.

4.4 Example application 2: global decision-making for RDM network

The RDM system [22] is a disaster recovery system for tolerating failures by maintaining multiple replicas (i.e. copies) of data at remotely located mirrors (i.e. servers). Access to the data can continue even if one of the copies of data is lost. Each network link in the network has an associated operational cost[4] and a measurable throughput, latency and loss rate used to determine the reliability, performance and cost of the RDM system. The goal here is to achieve the satisfaction of the NFRs of Minimization of Costs (MC), Maximization of Performance (MP)[5] and Maximization of Reliability (MR) under uncertain environmental contexts of link failures and varying ranges of bandwidth consumption [22]. Hence, the network is required to continuously take *global adaptive decisions* of switching between the topologies of Minimum Spanning Tree (MST) and Redundant Topology (RT) to maintain better levels of satisfaction of NFRs. Both the topological configurations have a different impact on the levels of satisfaction of NFRs. RT provides a higher level of reliability than MST topology but it has a negative impact on the satisfaction of the MC and MP as the cost of maintaining non-stop RT topology will be high and the performance can be reduced because of data redundancy. On the other hand, MST topology provides better levels of satisfaction for MC and MP by maintaining a minimum spanning tree for the network.

[4] In RDM system, operational cost is measured in terms of inter-site network traffic.

[5] In the case of RDM network, we are measuring performance in terms of total time to write the data, i.e. the sum of the time to write each copy of data on each remote site.

Table 6 States of priority-aware MR-POMDP++ for RDM network

S	NFR_1=MC	NFR_2 = MR	NFR_2 = MP
s_1	True	True	True
s_2	True	True	False
s_3	True	False	True
s_4	True	False	False
s_5	False	True	True
s_6	False	True	False
s_7	False	False	True
s_8	False	False	False

Table 7 Reward values for the NFRs in the RDM network

S	Action (A)	Reward vector values		
		$R_{NFR1} = R_{MC}$	$R_{NFR2} = R_{MR}$	$R_{NFR3} = R_{MP}$
s_1	MST	39.17	39.0	40.0
s_2	MST	41.0	40.0	39.0
s_3	MST	39.0	38.0	38.5
s_4	MST	17.0	16.0	15.0
s_5	MST	44.0	43.0	43.5
s_6	MST	29.0	28.0	27.0
s_7	MST	14.0	13.0	13.5
s_8	MST	2.0	1.0	1.0
s_1	RT	41.0	43.0	41.0
s_2	RT	32.0	33.0	31.0
s_3	RT	28.0	29.0	27.0
s_4	RT	26.0	27.0	25.0
s_5	RT	28.0	29.0	27.0
s_6	RT	16.0	17.0	15.0
s_7	RT	23.0	24.0	22.0
s_8	RT	11.0	12.0	10.0

4.4.1 Initial set-up

For experimental purposes, we consider the RDM network [50] that is based on the operational model presented in [22,24]. The RDM network under consideration consists of 25 RDM Mirrors with a total number of 300 physical links used to transfer data between the mirrors [50]. Considering this set-up of the network, the maximum number of concurrent active network links is 120 that does not affect the assigned budget for the network [17]. For the current set of experiments, we focus on the NFRs related to the quality and performance attributes [19] of the RDM network such as MC, MR and MP.

Next, we discuss the initial set-up of the components pf MR-POMDP++ model for the considered RDM network.

Components of MR-POMDP++ for RDM network

The components of MR-POMDP++ model for the considered RDM network are explained as follows:

States We represent states as combinations of satisfaction levels of NFRs. Therefore, for the three NFRs (MC, MR and MP), eight states are identified and are shown in Table 6.

Actions Actions represent the adaptation strategies to support the satisfaction of the NFRs MC, MR and MP. For the case of RDM network, we consider two adaptive actions in the form of the two topological configurations of Minimum Spanning Tree (MST) and Redundant Topology (RT).

Rewards As we are dealing with 3 NFRs of MC, MR and MP for the RDM network, the reward vector has a size of 3 represented as:

$$\mathbf{R(s,a)} = [R_{MC}, R_{MR}, R_{MP}].$$

Table 8 Transition probabilities for NFR MC

NFR_1 = Minimization of Cost (MC)					
Action (A)	MC_t	MR_t	MP_t	$P(MC_{t+1} = T)$	$P(MC_{t+1} = F)$
MST	True	True	True	0.9	0.1
MST	True	True	False	0.88	0.12
MST	True	False	True	0.92	0.08
MST	True	False	False	0.9	0.1
MST	False	True	True	0.85	0.15
MST	False	True	False	0.83	0.17
MST	False	False	True	0.87	0.13
MST	False	False	False	0.85	0.15
RT	True	True	True	0.86	0.14
RT	True	True	False	0.84	0.16
RT	True	False	True	0.88	0.12
RT	True	False	False	0.86	0.14
RT	False	True	True	0.73	0.27
RT	False	True	False	0.71	0.29
RT	False	False	True	0.75	0.25
RT	False	False	False	0.73	0.27

Table 9 Transition probabilities for NFR MR

NFR_2 = Maximization of Reliability (MR)					
Action (A)	MC_t	MR_t	MP_t	$P(MR_{t+1} = T)$	$P(MR_{t+1} = F)$
MST	True	True	True	0.91	0.09
MST	True	True	False	0.93	0.07
MST	True	False	True	0.89	0.11
MST	True	False	False	0.91	0.09
MST	False	True	True	0.93	0.07
MST	False	True	False	0.95	0.05
MST	False	False	True	0.91	0.09
MST	False	False	False	0.93	0.07
RT	True	True	True	0.95	0.05
RT	True	True	False	0.97	0.03
RT	True	False	True	0.93	0.07
RT	True	False	False	0.95	0.05
RT	False	True	True	0.97	0.03
RT	False	True	False	0.99	0.01
RT	False	False	True	0.95	0.05
RT	False	False	False	0.97	0.03

where R_{MC}, R_{MR} and R_{MP} represent the priority values for MC, MR and MP, respectively. The reward vector values (provided by the domain experts) for NFRs in the RDM network are shown in Table 7.

Transition Function According to Rule 1, the states are represented as a combination of NFRs in MR-POMDP++. Considering Rule 1 and using the property of conditional independence and Bayes rule [38], we factor the transition probabilities T(s,a,s′) as marginal conditional probabilities of NFRs P(MC′ | MC, a), P(MR′ | RPL, a) and P(MP′ | MP, a) [38] as follows:

$$\begin{aligned} T(s,a,s') &= P(s'|s,a) \\ &= P(MC' \mid MC, a) \\ &\quad P(MR' \mid MR, a) P(MP' \mid MP, a) \end{aligned}$$

Table 10 Transition probabilities for NFR MP

$\mathbf{NFR}_1$ = Maximization of Performance (MP)					
Action(A)	$\mathbf{MC}_t$	$\mathbf{MR}_t$	$\mathbf{MP}_t$	$\mathbf{P}(\mathbf{MP}_{t+1} = T)$	$\mathbf{P}(\mathbf{MP}_{t+1} = F)$
MST	True	True	True	0.9	0.1
MST	True	True	False	0.85	0.15
MST	True	False	True	0.92	0.08
MST	True	False	False	0.87	0.13
MST	False	True	True	0.88	0.12
MST	False	True	False	0.83	0.17
MST	False	False	True	0.9	0.1
MST	False	False	False	0.85	0.15
RT	True	True	True	0.82	0.18
RT	True	True	False	0.75	0.25
RT	True	False	True	0.84	0.16
RT	True	False	False	0.77	0.23
RT	False	True	True	0.8	0.2
RT	False	True	False	0.73	0.27
RT	False	False	True	0.82	0.18
RT	False	False	False	0.75	0.25

The transition probabilities for the RDM network are shown in the Tables 8, 9 and 10. These transition probabilities are provided by the domain experts [38].

Observations In the RDM system, three monitorable variables Ranges of Bandwidth Consumption (RBC), Active Network Links (ANL) and Total Time for Writing (TTW) related to the NFRs MC, MR and MP, respectively, are specified. Higher the value for the ANL, higher will be satisfaction of MR. In contrast, lower values of RBC and TTW lead to higher levels of satisfaction for MC and MP. In the RDM System, all of these monitorable variables have range boundaries that are defined by the domain experts as shown in Table 11.

Like the transition model, we also factor the observation model into the product of conditional probabilities [38] as follows:

$$O(s', a, z) = P(z|s', a) = P(Mon_1, ..Mon_n|s', a)$$

Hence,

$$P(z|s', a) = P(RBC, ANL, TTW|s', a)$$
$$= P(RBC|s', a)P(ANL|s', a)P(TTW|s', a)$$

The conditional probabilities for the observation model are shown in Table 11.

4.4.2 Requirements specification

The initial set-up of the experiments also considers the *Requirements* (***R***) for the RDM network that reflect the NFRs' satisfaction levels required. These *requirements* are defined by the domain experts based on [17,22]. These requirements indicate the satisfaction thresholds representing the suitable zone of satisfaction of NFRs. For the experiments, we consider the following set of *requirements*:

R1: *The probability of satisfaction of Minimization of Cost shall be greater than or equal to 0.70, i.e.* ***P(MC=True)>=0.70***.

R2: *The probability of satisfaction of Maximization of Reliability shall be greater than or equal to 0.85, i.e.* ***P(MR=True)>=0.85***.

R3: *The probability of satisfaction of Maximization of Performance shall be greater than or equal to 0.75, i.e.* ***P(MP=True)>=0.75***.

The NFRs are considered in their poor zone of satisfaction if they have satisfaction levels below these thresholds.

4.4.3 Experimental scenarios

We have designed the experiments to study the unanticipated impacts of actions on the NFRs' satisfaction levels. The unmatching NFRs' priorities due to uncertain environmental contexts may require the MR-POMDP++ to tune these priorities. The idea is to study how priority-aware decision-making offered by MR-POMDP++ helps to support informed choices of priorities concerning individual NFRs.

To study the impact of actions on the satisfaction levels of the NFRs MC, MR and MP, we simulate dynamic situations by introducing random changes in the environment of the RDM network. These random changes are used to simulate

Table 11 Observation probabilities for RDM network

Mon_1 = Ranges of Bandwidth Consumption(RBC)				
Action (A)	**MEC_t**	**P($RBC_{t+1} < x$)**	**P($MC_{t+1} in [x, y]$)**	**P($MC_{t+1} >= y$)**
MST	True	0.8	0.15	0.05
MST	False	0.72	0.18	0.1
RT	True	0.78	0.16	0.06
RT	False	0.68	0.2	0.12
Mon_2 = Active Network Links (ANLs)				
Action (A)	**MR_t**	**P($ANLs_{t+1} < r$)**	**P($ANLs_{t+1} in [r, s]$)**	**P($ANLs_{t+1} >= s$)**
MST	True	0.06	0.16	0.78
MST	False	0.12	0.2	0.68
RT	True	0.05	0.15	0.8
RT	False	0.1	0.18	0.72
Mon_3 = Time to Write(TTW)				
Action (A)	**MR_t**	**P($TTW_{t+1} < f$)**	**P($TTW_{t+1} in [f, g]$)**	**P($TTW_{t+1} >= g$)**
MST	True	0.83	0.13	0.04
MST	False	0.67	0.23	0.1
RT	True	0.8	0.15	0.05
RT	False	0.63	0.25	0.12

failures in the network links during execution of the self-adaptive RDM network. Such network link failures may be due to problems in devices such as switches or routers. For this purpose, deviations from the initially defined transition probabilities (i.e. $P(NFR_{t+1} = True|NFR_t, A_t)$) for the topologies (MST and RT) are introduced randomly at runtime.

We consider the following detrimental contexts for the purpose of evaluation of our results:

Detrimental Context 1 (DC1): *The deviation levels are introduced to simulate unanticipated packet loss during the execution of the RDM network. This increase in the packet loss during the topological configuration of RT would lead to an unusual rate of data forwarding resulting in an increase in bandwidth consumption and a decrease in the performance. As a consequence, a decrease in the satisfaction levels of* MC *i.e. P(MC = True) and* MP *i.e. P(MP = True) would be expected.*

Detrimental Context 2 (DC2): *The deviation levels are introduced in the RDM network during the execution of MST topology to simulate an unexpected data packet loss resulting in decrease in the reliability of the network. Data packet loss may represent network link failures in the RDM system, which may be caused due to problems with the equipment. As a result, the satisfaction level of MR, i.e. P(MR = True) would be expected to be reduced.*

In order to simulate small realistic changes, we have introduced a maximum deviation of 12% from the current transition probabilities for a randomly selected duration between 5 and 15 timesteps for a specific deviation level. Considering the above detrimental contexts, next we present the two cases for experiments:

Similar to the IoT example application, we evaluate the Hypotheses using the following experimental cases:

Case 1: Priority-aware decisions and autonomous tuning of NFRs' priorities

Here, we demonstrate priority-aware decision-making offered by our Pri-AwaRE approach and how it supports compliance with the requirements specification. To perform priority-aware decision-making, our proposed approach uses MR-POMDP++ to represent distinct priorities of NFRs in the form of rewards. We study how the priorities of NFRs have an impact on the action selection for the satisfaction of NFRs as shown in Table 12. For example, using the initial set-up, at *timestep 45*, MR-POMDP++ provides the best possible trade-off by selecting the MST as a preferred topology over RT. The reason behind this decision is that the expected utility values for MP and MC are *395.32709* and *392.13139*, respectively, which are higher than the expected utility value for MR which is *388.21367* as shown in Table 12. This shows that the application of MST topology on the network has a more positive impact on the satisfaction of MP and MC than on MR. In comparison with MST, at timestep 45, the *expected utility values* in case of RT topology were *386.71867* for MC, *392.08584* for MR and *386.20288* for MP. Due to higher impacts offered by MST, the system selects MST as the preferred topology,

Table 12 Experiment results for timesteps 45–51

Time	Action	Val_{MC}	Val_{MR}	Val_{MP}	Sat_{MC}	Sat_{MR}	Sat_{MP}
45	**MST**	**392.13139**	**388.21367**	**395.32709**	**0.90299**	**0.92395**	**0.91017**
46	MST	391.98060	388.04530	395.15471	0.90445	0.89257	0.91361
47	MST	391.97769	388.04279	395.15358	0.9052	0.89170	0.91431
48	MST	391.71502	387.74353	394.83453	0.90532	0.83722	0.91445
49	**RT**	**386.76743**	**392.13918**	**386.25769**	**0.90618**	**0.92089**	**0.91524**
50	MST	392.30172	388.33157	395.31275	0.86462	0.95651	0.84732
51	MST	392.13863	388.21970	395.33160	0.90089	0.92493	0.90972

Val_{NFR} represents the expected utility value of NFRi i.e. Val_{MC}, Val_{MR} and Val_{MP}
Sat_{NFR} represents satisfaction level of NFRi i.e. Sat_{MC}, Sat_{MR} and Sat_{MP}

to support the reduction of inter-site network links cost and improve the performance of the network. According to this decision, MST topology is set for the network. Hence, to offer priority-awareness, the decisions offered by MR-POMDP++ make the system aware of the explicit impacts of the decisions on the satisfaction of NFRs as presented in Table 12.

On the other hand, at *timestep 49*, the system decides to switch the topology from MST to RT. The reason behind it is that the expected utility value of MR being *392.13918* is higher than the expected utility values of MC and MP, i.e. *386.76743* and *386.25769*, respectively, as shown in Table 12. As a consequence, the adaptation to RT topology shows an improvement in the satisfaction of MR from *P(MR = True) = 0.83722* to *P(MR = True) = 0.92089* by meeting compliance to the required satisfaction threshold, i.e. *P(MR = True)>=0.85*. Hence, the decision-making offered by MR-POMDP++ takes into account these *expected utility values* representing the new values for the tuned priorities of the NFRs. This is one of the *contributions* of our Pri-AwaRE approach, unlike other approaches, to offer SASs with a priority-aware decision-making process.

During the set-up of experiments, the initial priorities for NFRs of the RDM network were defined by the experts considering the different anticipated runtime contexts. According to the rules defined in Sect. 4, these initial priorities were defined in the form of rewards for the MR-POMDP++ model (as shown in Table 7). During the decision-making process, these pre-defined priorities were tuned autonomously by MR-POMDP++, according to the runtime situations, using the computation of the *expected utility value* for each NFR individually (using Eq. 9). This tuning of individual priorities by MR-POMDP++ helps in supporting the SASs to comply to the *requirements* (***R***) by achieving higher levels of satisfaction for NFRs. The newly tuned priorities correspond with the *expected utility values* represented in Table 12. The goal of this *autonomous tuning* with help of *expected utility values* is to meet the *requirements specification* for the NFRs.

Case 2: Impacts of priority-aware decisions on satisfaction levels of NFRs

We have also studied the impact of priority-aware decisions by Pri-AwaRE on NFR satisfaction levels under different detrimental contexts (*DC1* and *DC2*), and have compared its results with the existing techniques of RE-STORM [38], implemented using DESPOT (a single objective POMDP Solver), and RE-STORM-ARROW [39]. The results reported in Table 12 show the implications of setting the selected topology for the network on the satisfaction levels on NFRs at a particular timestep. This decision of topology selection takes into account the *expected utility values* for individual NFRs.

Let us observe Figs. 10, 11 and 12 which show the results of Pri-AwaRE and RE-STORM under (i) the initial set of pre-defined rewards, transition and observation probabilities (stable conditions scenario), and (ii) the detrimental contexts *DC1* and *DC2* where the deviation levels are introduced. Under the stable scenario and *DC1*, both Pri-AwaRE and RE-STORM show comparable results by maintaining the NFRs of MC, MR and MP in the suitable zone of satisfaction, i.e. above the threshold values of P(MC = True) >= 0.70, P(MR = True) >= 0.85 and P(MP = True) >= 0.75. Both techniques show preference for the MST topology, with an increase in the use of MST topology by Pri-AwaRE under *DC1* to support the satisfaction of both MC and MP as shown in Figs. 13 and 14. Under stable conditions, the percentage usage of MST topology by Pri-AwaRE and RE-STORM is 95.2 and 92.9%, respectively, for the simulation duration of the experiments. However, under *DC1*, the percentage usage of MST is increased to 99.8% by Pri-AwaRE as shown in Fig. 14. MST offers lower operational cost and improved performance along with supporting minimal level of required reliability. Therefore, application of MST to the network is the most suitable decision to be taken both under stable conditions and *DC1* [37]. In contrast, under *DC2*, where deviations are introduced to effect the system's reliability, Pri-AwaRE shows better level of satisfaction for MR as compared to RE-

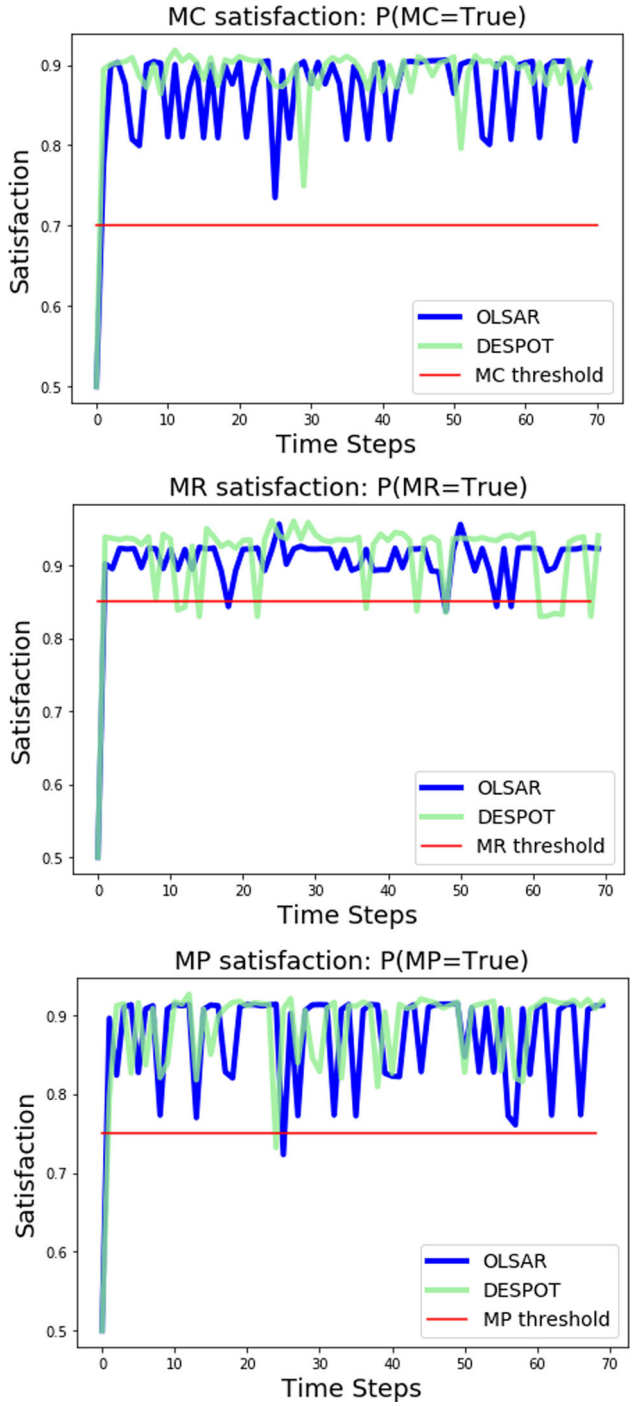


Fig. 10 Satisfaction of NFRs over time under stable conditions by applying Pri-AwaRE (using OLSAR) and RE-STORM (using DESPOT)

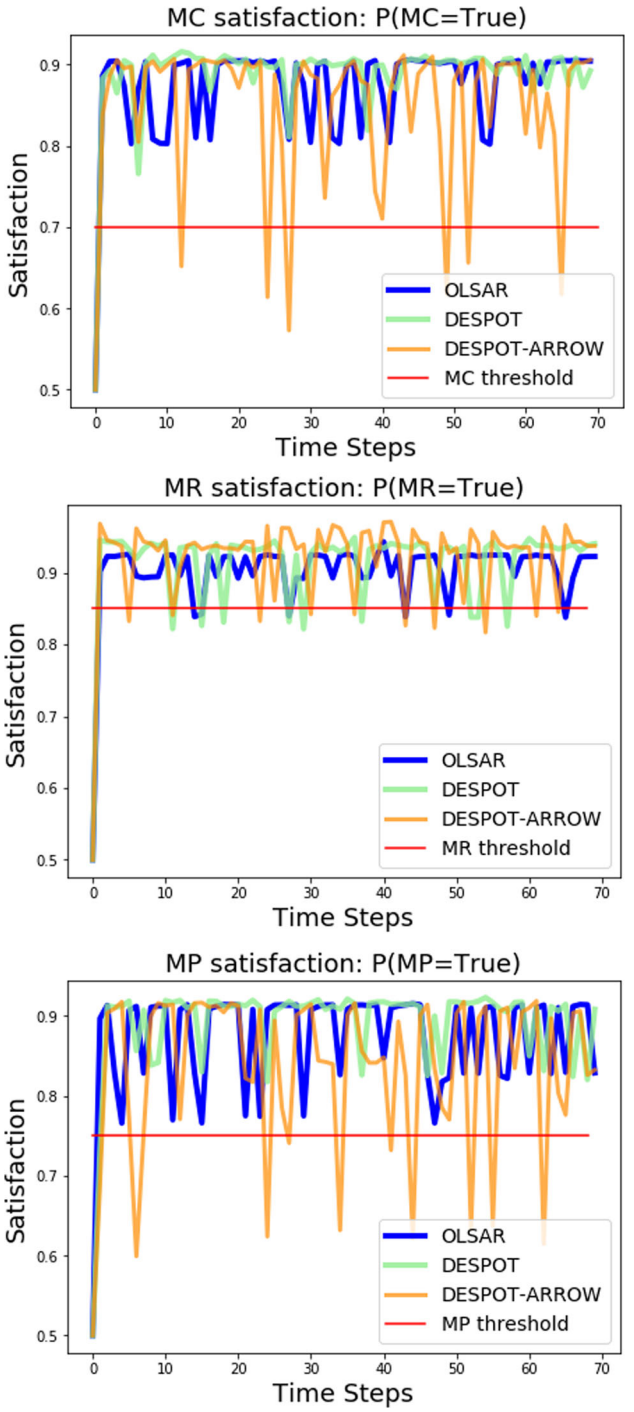


Fig. 11 Satisfaction of NFRs over time under detrimental context 1 (DC1) by applying Pri-AwaRE (using OLSAR), RE-STORM (using DESPOT) and RE-STORM-ARROW (using DESPOT-ARROW)

STORM as shown in Fig. 12. The satisfaction level of MR goes below the satisfaction threshold at a number of timesteps for DESPOT under *DC2*. However, in *DC2* both Pri-AwaRE and RE-STORM show comparable results in terms of satisfaction of MC and MP by keeping them above the satisfaction threshold at most of the timesteps as shown in Fig.12. To support MR, both Pri-AwaRE and RE-STORM show an increase in the use of RT topology to 35.8 and 21.4%, respectively, as shown in Figs. 13 and 15, respectively. This is the required expected behaviour by the self-adaptive RDM network as defined by the experts [17,37].

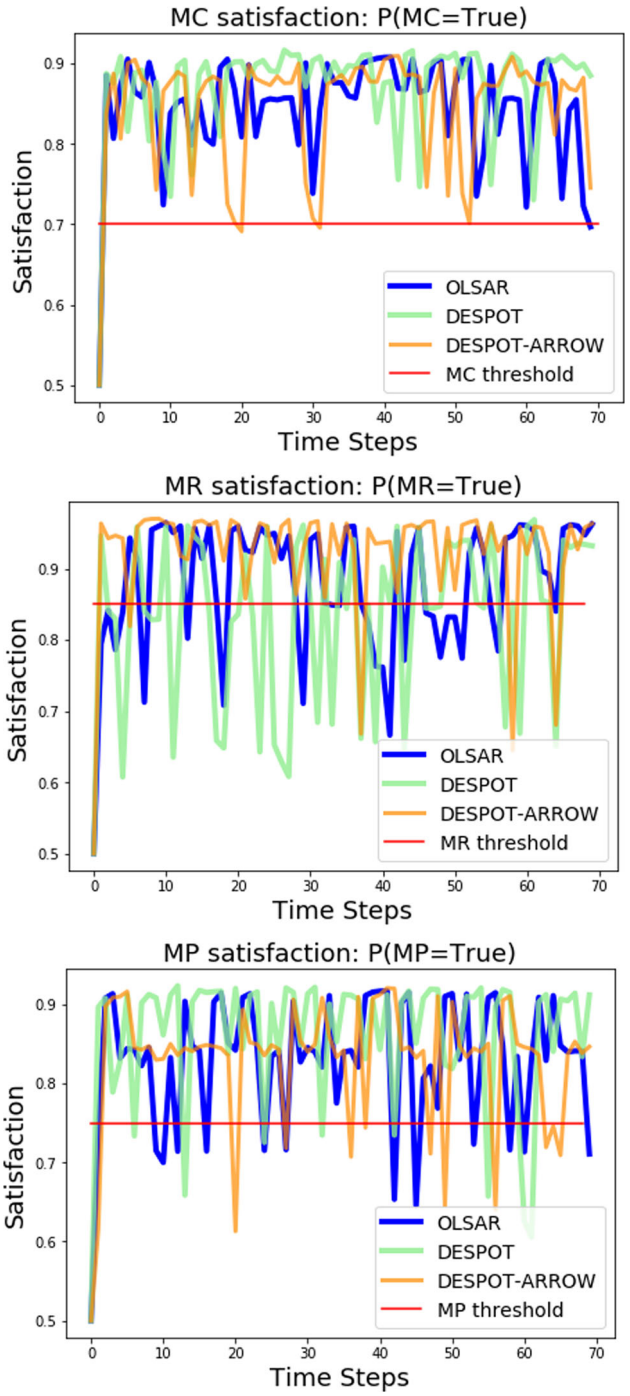


Fig. 12 Satisfaction of NFRs over time under detrimental context 2 (DC2) by applying Pri-AwaRE (using OLSAR), RE-STORM (using DESPOT) and RE-STORM-ARROW (using DESPOT-ARROW)

Furthermore, we have also compared our results with the technique of RE-STORM-ARROW(based on P-CNP) [39], which offers the support of updating initially defined rewards for RE-STORM under both *DC1* and *DC2* as shown in Figs. 11 and 12, respectively. However, even if the technique RE-STORM-ARROW supports the satisfaction of MR under

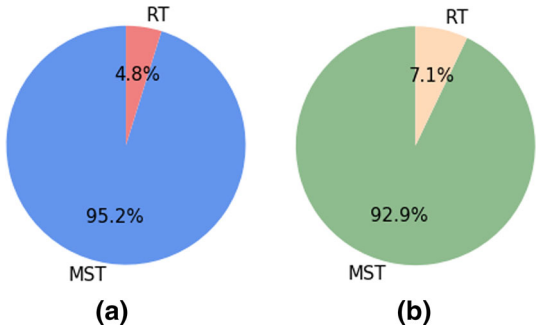


Fig. 13 Topologies selection under stable environmental context **a** Pri-AwaRE and **b** RE-STORM

the *DC1*, it brings the levels of satisfaction of MC and MP to poor zones of satisfaction during several timesteps as shown in Fig. 11. Furthermore, under *DC2*, Pri-AwaRE shows comparable results to RE-STORM-ARROW in terms of satisfaction of MC and MP but RE-STORM-ARROW shows better satisfaction levels for MR as compared to Pri-AwaRE. This is due to the fact that RE-STORM-ARROW with the help of P-CNP approach provides the update of the initially defined rewards for RE-STORM approach. This is not the case in our approach where we are providing autonomous tuning of priorities that uses the rewards (see Eq. 9) but we are not updating the initially defined rewards. Moreover, RE-STORM-ARROW, even with the update of reward values does show poor levels of satisfaction for MR and MP at several timesteps as shown in Fig. 12. In contrast to RE-STORM-ARROW, our approach offers higher levels of NFR satisfaction. Further, another shortcoming of RE-STORM-ARROW is that the update offered by the POMDP is not autonomously executed as is the case of MR-POMDP++. Instead, RE-STORM-ARROW needs external support from the approach of P-CNP which creates efficiency problems. Hence, from the results, we can deduce that the priority-aware decision-making process by Pri-AwaRE offers higher levels of satisfaction of NFRs even under the detrimental contexts when compared to the existing single-objective techniques.

Summary of findings in experiments

In summary, our experimental results show that all the NFRs' levels of satisfaction MC, MR and MP are compliant with the requirements specification as evident from the average satisfaction levels presented in Fig. 16. The reported results show a confidence level of 95%. The confidence intervals and standard error for average satisfaction levels of NFRs under different experimental scenarios are presented in Table 13. Consider first the case of average satisfaction levels of NFRs under DC1. The average satisfaction level of MC lies between the confidence interval of 0.8649 and 0.8863 with a standard error of 0.0054. Similarly, for average satisfaction level of MR, the confidence interval lies between 0.8948

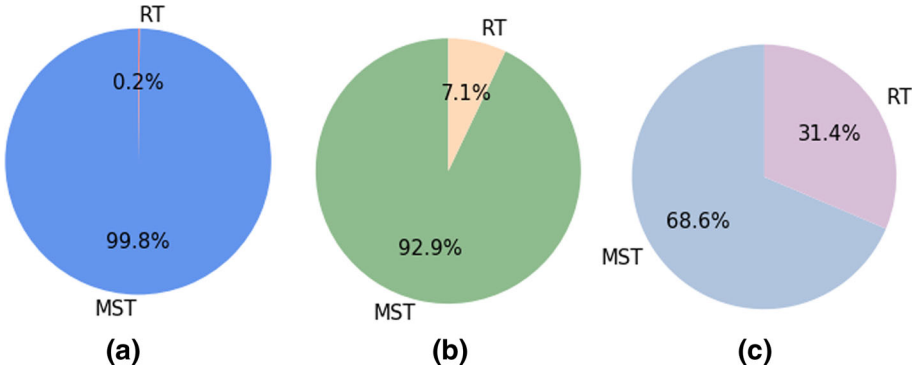


Fig. 14 Topologies selection under detrimental context 1 (DC1) **a** Pri-AwaRE, **b** RE-STORM and **c** RE-STORM-ARROW

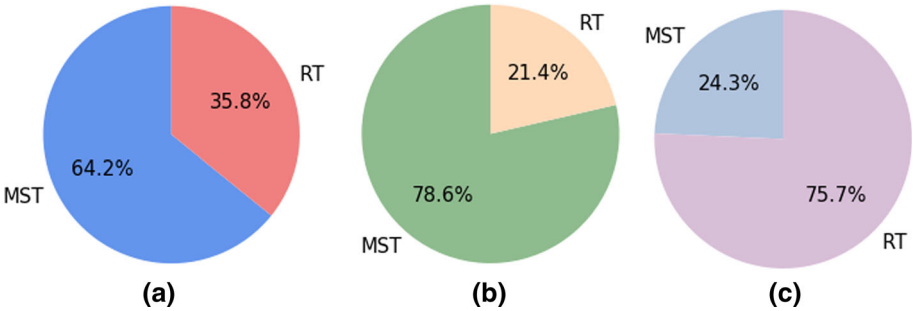


Fig. 15 Topologies selection under detrimental context 2 (DC2) **a** Pri-AwaRE, **b** RE-STORM and **c** RE-STORM-ARROW

and 0.9137 showing a confidence level of 95% with a standard error of 0.0048. Moreover, in correspondence to MC and MR, the average satisfaction level of MP lies between the confidence interval of 0.8617 and 0.8878 with a standard error of 0.0066. These confidence interval values show that the levels of satisfaction achieved for the NFRs meet the requirements specifications $P(MC = True) >= 0.70$, $P(MR = True) >= 0.85$ and $P(MC = True) >= 0.75$. Furthermore, the Pri-AwaRE approach has also shown comparable results under other scenarios (including stable scenario and DC2) as shown in Table 13. Hence, from the results we can conclude that Pri-AwaRE offers statistically sound results in terms of fulfilment of the requirements. Hence, it rejects hypothesis H_0 and supports hypothesis H_a.

Discussion

From the results, it is evident that our Pri-AwaRE approach, based on MR-POMDP++, comply with the *requirements* (***R***) for the RDM network both under the stable and detrimental environmental conditions. The average satisfaction levels for all the NFRs of MC, MR and MP, generated by Pri-AwaRE, under initial stable conditions are *0.8732*, *0.9069* and *0.8747*, respectively. For the detrimental context scenarios, the average satisfaction levels are $P(MC = True) = 0.8756$, $P(MR = True) = 0.9042$ and $P(MP = True) = 0.8811$ under *DC1* and $P(MC = True) = 0.8431$, $P(MR = True) = 0.8906$ and $P(MP = True) = 0.8315$ under *DC2*, respectively. Hence, the system is conforming to the *requirements specification* of $P(MC = True) >= 0.70$, $P(MR = True) >= 0.85$ and $P(MC = True) >= 0.75$ as shown in Fig. 16.

Under all the scenarios, our approach shows comparable and sometimes even better satisfaction levels for NFRs than single-objective techniques of RE-STORM and RE-STORM-ARROW. Hence, based on the results, we can deduce that our approach using the reward vector provides statistically better results as compared to single-objective approaches. Moreover, our Pri-AwaRE approach also provides more *awareness* to the decisions by using individual NFRs' priorities during the decision-making process. To further evaluate our approach, we have also executed experiments using OLS algorithm for the RDM case. The results for OLS algorithm are reported in [48].

Furthermore, we have also compared the overall performance of the approaches. We have performed experiments on a Lenovo Thinkpad with intel Core i7, 8th Gen processor and 16 GB RAM. Using this hardware set-up, Pri-AwaRE, based on MR-POMDP++, takes 1500 ms to come up with a decision and RE-STORM takes 1000 ms for decisions. This is due to the fact that usage of OLSAR with the multi-reward Perseus solver adds this additional overhead as described in Section 2.4. Considering the case studies that we have selected to test our approach, this performance can be considered as adequate. The adaptation process typically depends on the frequency with which the stakeholders want to monitor the managed system. For example, in case of Internet of Things network, one simulation timestep represents a network activity of 15 min [21]. So the adaptation decisions are taken after a gap of this time interval. Whereas in case of Remote Data Mirroring network, we consider one timestep representing an adaptation decision taken after a network activity of 1 h.

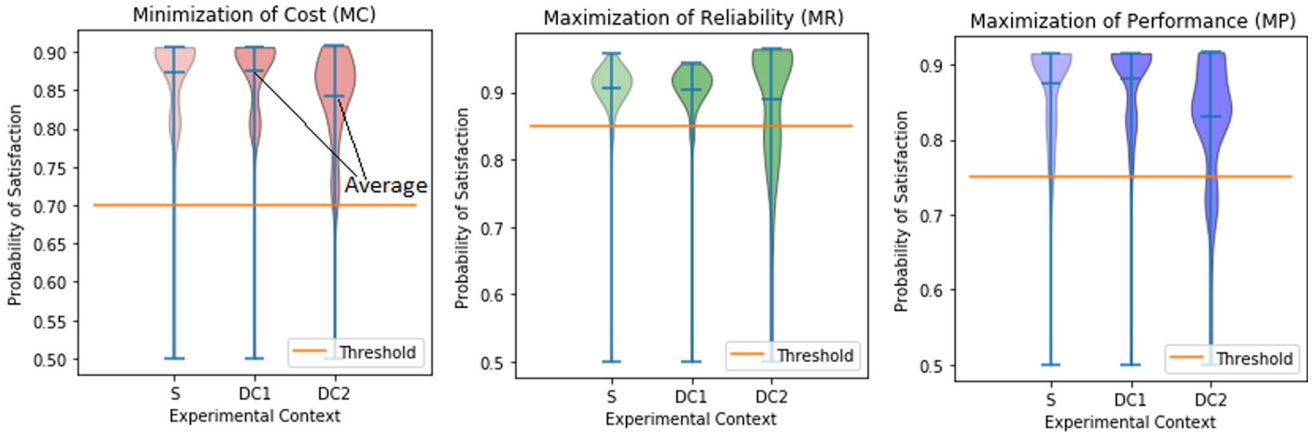


Fig. 16 Average satisfaction of NFRs under different environmental contexts–stable (S), detrimental context 1 (DC1) and detrimental context 2 (DC2)

Table 13 Confidence intervals for average satisfaction levels of NFRs

Scenario	NFR	Sat_{AVG}	Confidence interval	Standard error
Stable	MC	0.8732	0.8622–0.8842	0.0055
	MR	0.9069	0.8975–0.9163	0.8975
	MP	0.8747	0.8617–0.8878	0.0066
Detrimental Context 1 (DC1)	**MC**	**0.8756**	**0.8649–0.8863**	**0.0054**
	MR	**0.9043**	**0.8948–0.9137**	**0.0048**
	MP	**0.8811**	**0.8691–0.8931**	**0.006**
Detrimental Context 2 (DC2)	MC	0.8431	0.8300–0.8561	0.0066
	MR	0.8906	0.8736–0.9076	0.0086
	MP	0.8315	0.8163–0.8467	0.0076

*Sat_{AVG} represents the average satisfaction level of NFR

Hence, the focus of our Pri-AwaRE approach is on runtime decision-making in SASs which is not hard real time [7].

5 Threats to validity

The threats to validity are based on the classification provided in [15]. External validity is related to the generalization of the outcomes outside the scope of our study. Internal validity focuses on ensuring that the treatment used in the experiments is analogous to the actual outcome that is observed. Finally, construct validity is based on the relation between the theory behind the experiments and the observations.

External validity A key threat to validity of our proposed approach lies in the computational cost of MR-POMDPs. Solving an MR-POMDP is a computationally intractable problem in its worst case, even using the OLSAR algorithm [44] that our implementation uses and which overcomes scalability issues related to the "curse of history". In the Pri-AwaRE architecture, the states are defined in terms of combinations of satisfaction levels of NFRs. If the number of NFRs is 2, we would have 4 states for MR-POMDP, for 3 NFRs, it would be 8 and so on. Therefore, in practice, it will work with small number of NFRs. It hints that the design experts of SASs, while using our approach, must limit their reasoning to critical NFRs that drive self-adaptation. Hence, our approach belongs to the multi-objective sequential decision-making techniques that focus both theoretically and practically on a few objectives, which according to [43] includes a considerable number of applications.

We have executed experiments using example applications [21,50] that focus on a centralized setting. The approach has not been tested in a decentralized set-up yet. More experiments would be required to test the application feasibility of our Pri-AwaRE approach in both centralized and decentralized domains.

Internal validity The internal threat to validity concerns the extent to which our approach performs in an actual environmental set-up. In this paper, we have employed a case study approach based on a simulator. Our experimental results are based on the environmental factors presented by a simulating

environment not an actual physical network. The example applications [21,50] that we have selected are well-known and provide simulations that are closer to the real settings. Both RDM [17,50] and IoT [21] are well accepted applications in the research community and are already in use by the other teams.

Construct validity The construct validity concerns the mirroring relationship between MR-POMDP++ and the managed system. However, we have established reflections for the current state of the managed system by MR-POMDP++ presented in [50] but more work is required. As part of our future work, we plan to make more investigations towards studying the mirroring aspect.

6 Related work

In this section, we discuss the different existing techniques that deal with prioritization of NFRs in SASs. There are three criteria that any such technique must satisfy: It must endow the SASs with self-awareness [25], i.e. the ability to reason about how well it is satisfying the required NFRs in terms of their priorities; it must quantify uncertainty about the degree of satisfaction of NFRs that is acceptable in any given runtime context; and it must be sufficiently efficient to enable adaptation decisions to be computed at runtime. We have classified these techniques in two categories as follows:

6.1 Design time techniques

The techniques based on Multi-Criteria Decision-Making (MCDM) approaches such as Analytic Hierarchy Process [30] and Primitive Cognitive Network Process [62] to support ranking and explicit modelling of the NFRs have been presented but they are more of design time techniques. However, the approach of ARROW [39], based on P-CNP, provides support to RE-STORM, which is based on a POMDP model [18], to deal with the prioritization of NFRs. With the help of P-CNP, ARROW supports automatic update for initially defined priorities of NFRs at runtime but the update is not autonomous and does not work from within the POMDP model. Our approach tackles this limitation by improving the runtime representation of priorities of individual NFRs underpinned by the multi-rewards support offered by the MR-POMDP model. Hence, different from [18,39], and supported by the results of the experiments performed, our MR-POMDP-based approach takes it further to support the explicit modelling of the individual NFRs' priorities, allowing improved runtime reasoning, tuning and awareness during the decision-making process of a SAS at runtime.

In order to support optimization of NFRs, there are also approaches that are based on search based techniques such as [8,41]. These techniques perform optimization of priorities at design time, while these initially optimized priorities are further used at runtime in an off-line fashion. Therefore, it is considered that the approaches provide an explicit representation of priorities of NFRs but they are design time techniques. In contrast to these techniques, our approach supports the *runtime modelling* and *autonomous tuning* of *distinct priorities* of NFRs to offer *priority-aware decision-making* and the usage of Optimistic Linear Support algorithm [44] to solve MR-POMDP makes it more *efficient*.

6.2 Runtime techniques

In order to support decision-making in SASs, a number of runtime modelling techniques have been in use to support priority-awareness. Such techniques include the usage of probabilistic models like Dynamic Decision Networks (DDNs) [4]. The DDNs are used to represent goal model as a runtime model to support decision-making under uncertainty. The DDNs with the help of Bayesian Theory of Surprise [6] provides quantification of uncertainty. The approach represents the priorities of NFRs as a single scalar utility value that represents a combined priority value for all NFRs. This scalar utility value is then used to determine the effect (positive or negative) of the decision of an action selection on the satisfaction of all of NFRs as a whole at runtime. Moreover, the technique based on DDNs deals with the problem of scalability over time (i.e. the *curse of history*, the graph to represent the history of observations and actions for the DDN planning grows exponentially with the planning horizon). On the other hand, the techniques presented in [1,9,16,33,38,59] make use of Markov-based approaches such as Markov Decision Process (MDPs), Partially Observable Markov Decision Processes (POMDPs) and Discrete Time Markov Chains (DTMCs) along with probabilistic model checking to support runtime assurance of NFRs during decision-making in SASs. As these techniques are *Markov based* (similar to our approach), they support the *quantification of uncertainty* by maintaining probabilities over the state of the environment. An important limitation of these techniques is that they lack explicit modelling of the distinct priorities of NFRs at runtime. Furthermore, the approaches specifically based on MDPs and POMDPs (with no multiple-reward support) model the ranking of the NFRs as a scalar-reward value to indicate a cumulative priority of all the NFRs hindering priority awareness. Our MR-POMDP-based approach instead tackles these limitations using a vector-valued reward function to support *modelling* and reasoning about individual priorities of NFRs. Furthermore, our approach, based on the reward vector, also offers and exploits the built-in capability of *autonomous tuning* of priorities of NFRs. Hence, our Pri-

AwaRE approach provides the SAS with a higher degree of awareness of priorities during decision-making.

Furthermore, control theory-based approaches such as [32,40] have also been used to support explicit runtime configuration and tuning of NFRs. However, the technique in [32] lacks the autonomous prioritization of NFRs, while the approach in [40] lacks the capability of dealing with the NFRs having the same priority rank. Therefore, in such a situation it fails to perform trade-offs of the individual NFRs having the same priority rank as another NFR. Moreover, the technique in [40] also does not consider uncertainty as a quantifiable measure.

7 Conclusion and future work

In this paper, we have presented Pri-AwaRE, a self-adaptive architecture that uses MR-POMDP++ as a runtime specification model (***S***) within the MAPE-K loop. Based on Zave and Jackson's principle [63], usage of MR-POMDP++ supports priority-aware decision-making in SASs by (i) providing the runtime modelling and reasoning of priorities of individual NFRs by taking into account the knowledge (***K***) about their distinct priorities, (ii) maintaining compliance with the *requirements* (***R***) by autonomously tuning the NFRs priorities at runtime under uncertain environmental contexts. Existing techniques typically represent the priorities of NFRs in the form of a single scalar utility value to show a combined priority for all the NFRs. By contrast, we have shown how Pri-AwaRE, using a reward vector, works towards non-scalar nature of priorities offering therefore better-informed trade-offs of NFRs at runtime. The usage of MR-POMDP++ supports the *autonomous tuning* of priorities of NFRs by the computation of separate *expected utility value* for each NFR under changing contexts at runtime.

For evaluation purposes, we have applied Pri-AwaRE to two different example applications; a remote data monitoring network in which *global* adaptation decisions are taken for the whole system, and an IoT network where *local* adaptation decisions are taken for each individual sensor. These reveal that our decision-making approach shows compliance to the requirements (***R***) by achieving higher levels of satisfaction of NFRs even when applied to two different domains.

We have also compared Pri-AwaRE with existing single-objective techniques for both the example applications. Pri-AwaRE offered informed adaptation decisions, taking into account the individual priorities of NFRs, which led to the achievement of higher levels of satisfactions for NFRs.

For future work, we plan to use this technique as a tool for *apriori*-elicitation of priorities for NFRs. The idea is to perform simulations to learn about the environment to therefore, uncover contexts that otherwise would not be anticipated. The newly discovered knowledge ***K'*** would be made explicit in a new specification ***S'*** that would be used in the implementation of new releases of the SAS. Further, as the approach helps enriching the decision-making mechanism in SASs with newly discovered knowledge ***K'***, we are exploring how it can also be used to provide explanations for unclear adaptation decisions [37,51].

Finally, more research efforts are needed to explore the evolution of the model, including priorities and their impact on NFRs' satisfaction levels, and new alternative actions. This is a whole new important research line in the area of decision-making under uncertainty.

Acknowledgements We thank Luis H. Garcia Paucar for support on the use and experiments' design of the RDM case study. This work has been partially sponsored by The Lerverhulme Trust Fellowship "QuantUn: quantification of uncertainty using Bayesian surprises" (Grant No. RF-2019-548/9) and the EPSRC Research Project Twenty20Insight (Grant No. EP/T017627/1).

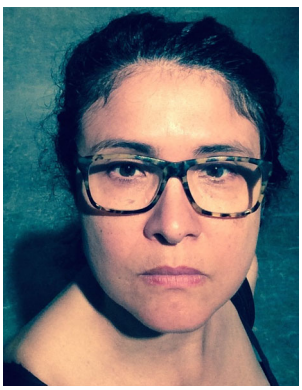
References

1. Abundo, M., Cardellini, V., Presti, F.L.: An MDP-based admission control for a QOS-aware service-oriented system. In: 2011 IEEE Nineteenth IEEE International Workshop on Quality of Service (2011), pp. 1–3. IEEE
2. Atzori, L., Iera, A., Morabito, G.: The internet of things: a survey. Comput. Netw. **54**(15), 2787–2805 (2010)
3. Bencomo, N., Belaggoun, A.: Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, pp. 221–236. Springer (2013)
4. Bencomo, N., Belaggoun, A. Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks. In: Proceedings of the 19th International Conference on Requirements Engineering: Foundation for Software Quality (Berlin, Heidelberg, 2013), REFSQ'13, pp. 221–236. Springer, Essen, Germany
5. Bencomo, N., Belaggoun, A., Issarny, V.: Bayesian artificial intelligence for tackling uncertainty in self-adaptive systems: the case of dynamic decision networks. In: 2013 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), pp. 7–13 (2013)
6. Bencomo, N., Belaggoun, A., Issarny, V.: Dynamic decision networks for decision-making in self-adaptive systems: a case study. In: 2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) (San Francisco, CA, USA, May 2013), pp. 113–122. IEEE

7. Bernat, G., Burns, A., Liamosi, A.: Weakly hard real-time systems. IEEE Trans. Comput. **50**(4), 308–321 (2001)
8. Bowers, K.M., Fredericks, E.M., Cheng, B.H.: Automated optimization of weighted non-functional objectives in self-adaptive systems. In: International Symposium on Search Based Software Engineering, pp. 182–197. Springer (2018)
9. Cámara, J., Lopes, A., Garlan, D., Schmerl, B.: Adaptation impact and environment models for architecture-based self-adaptive systems. Sci. Comput. Program. **127**, 50–75 (2016)
10. Chen, B., Peng, X., Yu, Y., Nuseibeh, B., Zhao, W.: Self-adaptation through incremental generative model transformations at runtime. In: Proceedings of the 36th International Conference on Software Engineering—ICSE 2014 (Hyderabad, India), pp. 676–687. ACM Press (2014)
11. Cheng, H.-T.: Algorithms for partially observable Markov decision processes. Ph.D. thesis, University of British Columbia (1988)
12. Chung, L., do Prado Leite, J.C.S.: On non-functional requirements in software engineering. In: Conceptual Modeling: Foundations and Applications. Springer, pp. 363–379 (2009)
13. De Lemos, R., Garlan, D., Ghezzi, C., Giese, H., Andersson, J., Litoiu, M., Schmerl, B., Weyns, D., Baresi, L., Bencomo, N., et al.: Software engineering for self-adaptive systems: research challenges in the provision of assurances. In: Software Engineering for Self-Adaptive Systems III. Assurances, pp. 3–30. Springer (2017)
14. Edwards, R., Bencomo, N.: DeSiRE: further understanding nuances of degrees of satisfaction of non-functional requirements trade-off. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems—SEAMS '18 (Gothenburg, Sweden, 2018), pp. 12–18. ACM Press
15. Feldt, R., Magazinius, A.: Validity threats in empirical software engineering research-an initial survey. In: Seke, pp. 374–379 (2010)
16. Filieri, A., Tamburrelli, G.: Probabilistic verification at runtime for self-adaptive systems. In: Assurances for Self-Adaptive Systems, pp. 30–59. Springer (2013)
17. Fredericks, E.M.: Mitigating Uncertainty at Design Time and Run Time to Address Assurance for Dynamically Adaptive Systems. Michigan State University, Computer Science, East Lansing (2015)
18. Garcia Paucar, L.H., Bencomo, N.: Knowledge base K models to support trade-offs for self-adaptation using Markov processes. In: 2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO), pp. 11–16. ISSN: 1949-3681, 1949–3673 (2019)
19. Glinz, M.: On non-functional requirements. In: 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 21–26. IEEE (2007)
20. Goldsby, H.J., Sawyer, P., Bencomo, N., Cheng, B.H., Hughes, D.: Goal-based modeling of dynamically adaptive system requirements. In: 15Th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2008), pp. 36–45. IEEE (2008)
21. Iftikhar, M.U., Ramachandran, G.S., Bollansee, P., Weyns, D., Hughes, D.: DeltaIoT: a self-adaptive internet of things exemplar. In: 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) (Buenos Aires, Argentina, May 2017), pp. 76–82. IEEE
22. Ji, M., Veitch, A.C., Wilkes, J.: Seneca: remote mirroring done write. In: USENIX Annual Technical Conference, General Track (2003)
23. Johnson, S.R., Tomlinson, G.A., Hawker, G.A., Granton, J.T., Grosbein, H.A., Feldman, B.M.A.: A valid and reliable belief elicitation method for Bayesian priors. J. Clin. Epidemiol. **63**, 370–383 (2010)
24. Keeton, K., Santos, C., Beyer, D., Chase J., Wilkes, J.: Designing for disasters. In: USENIX Conference on File and Storage Technologies, Berkeley (2004)
25. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. Computer **36**(1), 41–50 (2003)
26. Krupitzer, C., Roth, F.M., VanSyckel, S., Schiele, G., Becker, C.: A survey on engineering approaches for self-adaptive systems. Pervasive Mob. Comput. **17**, 184–206 (2015)
27. Le, T., Moh, S.: An energy-efficient topology control algorithm based on reinforcement learning for wireless sensor networks. Int. J. Control Autom. **10**(5), 233–244 (2017)
28. Le, T.T., Moh, S.: Reinforcement-learning-based topology control for wireless sensor networks. Proc. Grid Distrib. Comput. **2016**, 22–7 (2016)
29. Lemos, R.d.: Ed. Software engineering for self-adaptive systems II: international seminar, Dagstuhl Castle, Germany, October 24-29, 2010: revised selected and invited papers. No. 7475 in Lecture notes in computer science. Springer, Berlin, New York, 2013. OCLC: ocn839358754
30. Liaskos, S., McIlraith, S.A., Sohrabi, S., Mylopoulos, J.: Representing and reasoning about preferences in requirements engineering. Requir. Eng. **16**(3), 227–249 (2011)
31. Liu, C., Xu, X., Hu, D.: Multiobjective reinforcement learning: a comprehensive overview. IEEE Trans. Syst. Man Cybern. Syst. **45**(3), 385–398 (2015)
32. Maggio, M., Papadopoulos, A.V., Filieri, A., Hoffmann, H.: Automated control of multiple software goals using multiple actuators. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 373–384 (2017)
33. Moreno, G.A., Cámara, J., Garlan, D., Schmerl, B.: Flexible and efficient decision-making for proactive latency-aware self-adaptation. ACM Trans. Auton. Adapt. Syst. (TAAS) **13**(1), 1–36 (2018)
34. Morris, D.E., Oakley, J.E., Crowe, J.A.: A web-based tool for eliciting probability distributions from experts. Environ. Model. Softw. **52**, 1–4 (2014)
35. Nguyen, T.N., Ho, C.V., Le, T.T.: A topology control algorithm in wireless sensor networks for IOT-based applications. In: 2019 International Symposium on Electrical and Electronics Engineering (ISEE) (2019), pp. 141–145. IEEE
36. O'Hagan, A.: Probabilistic uncertainty specification: overview, elaboration techniques and their application to a mechanistic model of carbon flux. Environ. Model. Softw. **36**, 35–48 (2012)
37. Parra-Ullauri, J.M., García-Domínguez, A., García-Paucar, L.H., Bencomo, N.: Temporal models for history-aware explainability. In: Proceedings of the 12th System Analysis and Modelling Conference (New York, NY, USA, 2020), SAM '20, Association for Computing Machinery, pp. 155–164
38. Paucar, L.H.G., Bencomo, N.: RE-STORM: mapping the decision-making problem and non-functional requirements trade-off to partially observable Markov decision processes. In: Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems—SEAMS '18 (Gothenburg, Sweden, 2018), pp. 19–25. ACM Press
39. Paucar, L.H.G., Bencomo, N., Yuen, K.K.F.: ARRoW: automatic runtime reappraisal of weights for self-adaptation. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing—SAC '19 (Limassol, Cyprus, 2019), pp. 1584–1591. ACM Press
40. Peng, X., Chen, B., Yu, Y., Zhao, W.: Self-tuning of software systems through goal-based feedback loop control. In: 2010 18th IEEE International Requirements Engineering Conference (Sydney, Australia, Sept. 2010), pp. 104–107. IEEE
41. Ramirez, A.J., Cheng, B.H.C.: Automatic derivation of utility functions for monitoring software requirements. In: J. Whittle, T. Clark, and T. Kühne (Eds.) Model Driven Engineering Languages and

Systems (2011), Lecture Notes in Computer Science, pp. 501–516. Springer, Berlin
42. Roijers, D.: Multi-objective decision-theoretic planning. OCLC: 6893481195 (2016)
43. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. **48**, 67–113 (2013)
44. Roijers, D.M., Whiteson, S., Oliehoek, F.A.: Point-based planning for multi-objective POMDPs. In: Twenty-Fourth International Joint Conference on Artificial Intelligence (2015)
45. Ross, S., Pineau, J., Paquet, S., Chaib-draa, B.: Online planning algorithms for POMDPs. J. Artif. Intell. Res. **32**, 663–704 (2008)
46. Roush, F.W.: Multicriteria decision-aid. : Philippe Vincke, Chichester: John Wiley, 1992, 154 pages. Math. Soc. Sci. **25**(2), 204 (1993)
47. Roy, B.: Multicriteria Methodology for Decision Aiding. Kluwer Academic, Dordrecht (1996)
48. Samin, H.: Remote data mirroring—experiments. https://gitlab.com/re_research/rdmexperiments/. Tech. Rep. v0.9 (2020)
49. Samin, H., Garcia Paucar, L., Nelly, B., Sawyer, P. Towards priority-awareness in autonomous intelligent systems. In: 36th ACM/SIGAPP Symposium On Applied Computing (SAC). ACM (2021)
50. Samin, H., Paucar, L.H.G., Bencomo, N., Hurtado, C.M.C., Fredericks, E.M.: RDMSim: an exemplar for evaluation and comparison of decision-making techniques for self-adaptation. In: 16th international symposium on software engineering for adaptive and self-managing systems (SEAMS), p. 7 (2021)
51. Sawyer, P., Bencomo, N., Whittle, J., Letier, E., Finkelstein, A.: Requirements-aware systems: a research agenda for RE for self-adaptive systems. In: 2010 18th IEEE International Requirements Engineering Conference, pp. 95–103 (2010)
52. Shanavas, J., Simi, S.: An energy efficient topology control scheme with connectivity learning in wireless networks. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1770–1774. IEEE (2014)
53. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based pomdp solvers. Auton. Agents Multi-Agent Syst. **27**(1), 1–51 (2013)
54. Soh, H., Demiris, Y.: Evolving policies for multi-reward partially observable Markov decision processes (MR-POMDPs). In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, pp. 713–720 (2011)
55. Soh, H., Demiris, Y., Soh, H., Demiris, Y.: Multi-reward policies for medical applications: anthrax attacks and smart wheelchairs. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation (2011), pp. 471–478
56. Spaan, M.T.J.: Partially observable markov decision processes. In: Reinforcement Learning, vol. 27 (2012)
57. Spaan, M.T.J., Vlassis, N.: Perseus: randomized point-based value iteration for POMDPs. J. Artif. Intell. Res. **24**, 195–220 (2005). arXiv: 1109.2145
58. Sutcliffe, A., Sawyer, P., Stringer, G., Couth, S., Brown, L.J., Gledson, A., Bull, C., Rayson, P., Keane, J., Zeng, X.-J., et al.: Known and unknown requirements in healthcare. Requir. Eng. **25**(1), 1–20 (2020)
59. Weyns, D., Iftikhar, U.: Model-based simulation at runtime for self-adaptive systems. Proc. Models Runtime Würzburg **2016**, 1–9 (2016)
60. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.M.: RELAX: a language to address uncertainty in self-adaptive systems requirement. Requir. Eng. **15**, 177–196 (2010)
61. Xia, F., Yang, L.T., Wang, L., Vinel, A.: Internet of things. Int. J. Commun. Syst. **25**, 97–114 (2012)
62. Yuen, K.K.F.: The primitive cognitive network process in healthcare and medical decision making: comparisons with the analytic hierarchy process. Appl. Soft Comput. **14**, 109–119 (2014)
63. Zave, P., Jackson, M.: Four dark corners of requirements engineering. ACM Trans. Softw. Eng. Methodol. (TOSEM) **6**(1), 1–30 (1997)

Huma Samin is pursuing her Ph.D. degree in Computer Science from Aston University, Birmingham, UK. She worked as a Lecturer at the Edwardes College, Peshawar and the Institute of Management Sciences, Peshawar, Pakistan. Huma's research interests are in self-adaptive systems and autonomous systems and applying AI techniques to achieve self-adaptive capabilities at runtime.



Nelly Bencomo is an Associate Professor at Durham University in the UK. Her research interests in Software Engineering include Decision-making under Uncertainty, Model-Driven Engineering, Autonomous Systems, Artificial Intelligence and Requirementsl Engineering. She was awarded EU MC and UK Leverhulme Individual Fellowships. She is the PI of the EPSRC Project Twenty20 Insight. Contact her at nelly@acml.org or visit https://www.nellybencomo.me/.

Pete Sawyer is a Professor in CS at Aston University. His main research interests are in Software and Systems engineering, and particularly Requirements Engineering. In recent years, he has developed an interest in Digital Health, particularly applied to Brain health and how diagnostic software for neurodegenerative diseases. He also has a long-standing interest in text mining, with applications to software engineering.