Dynamic multi-objective optimisation using deep reinforcement learning: benchmark, algorithm and an application to identify vulnerable zones based on water quality

Md Mahmudul Hasan[a*], Khin Lwin[b], Maryam Imani[c], Antesar Shabut[d], Luiz Fernando Bittencourt[e], M A Hossain[b]

[a]*Anglia Ruskin IT Research Institute, Anglia Ruskin University, Chelmsford, CM11SQ, United Kingdom*
[b]*School of Computing and Digital Technologies, Teesside University, Middlesbrough, TS1 3BX, UK*
[c]*Water Systems Engineering and the Built Environment, Anglia Ruskin University, Chelmsford, CM11SQ, United Kingdom*
[d]*School of Arts and Communication, Leeds Trinity University, Leeds, LS18 5HD, UK*
[e]*Universidade Estadual de Campinas, Instituto de Computação, Computer Networks Laboratory, 13083-852, Campinas, SP, Brazil*
Email: md.hasan62@pgr.anglia.ac.uk[a], k.lwin@tees.ac.uk[b], maryam.imani@anglia.ac.uk[c], a.shabut@leedstrinity.ac.uk[d], bit@ic.unicamp.br[e], a.hossain@tees.ac.uk[b]*

Abstract

Dynamic multi-objective optimisation problem (DMOP) has brought a great challenge to the reinforcement learning (RL) research area due to its dynamic nature such as objective functions, constraints and problem parameters that may change over time. This study aims to identify the lacking in the existing benchmarks for multi-objective optimisation for the dynamic environment in the RL settings. Hence, a dynamic multi-objective testbed has been created which is a modified version of the conventional deep-sea treasure (DST) hunt testbed. This modified testbed fulfils the changing aspects of the dynamic environment in terms of the characteristics where the changes occur based on time. To the authors' knowledge, this is the first dynamic multi-objective testbed for RL research, especially for deep reinforcement learning. In addition to that, a generic algorithm is proposed to solve the multi-objective optimisation problem in a dynamic constrained environment that maintains equilibrium by mapping different objectives simultaneously to provide the most compromised solution that closed to the true Pareto front (PF). As a proof of concept, the developed algorithm has been implemented to build an expert system for a real-world scenario using Markov decision process to identify the vulnerable zones based on water quality resilience in São Paulo, Brazil. The outcome of the implementation reveals that the proposed parity-Q deep Q network (PQDQN) algorithm is an efficient way to optimise the decision in a dynamic environment. Moreover, the result shows PQDQN algorithm performs better compared to the other state-of-the-art solutions both in the simulated and the real-world scenario.

Keywords:

Dynamic environment; reinforcement learning; deep Q network; water quality resilience; meta-policy selection; artificial intelligence

1. Introduction

Human life consists of various problems which are dynamic in nature. Each of them requires different steps to be followed to make a final decision and it needs optimisation if more than one alternative is available. Therefore, multi-objective optimisation, a process to find an optimum solution for a problem, has become popular in recent days. Many problems involve incessantly changing properties and need optimisation of available solutions, which is more challenging. For instance, booking a flight or hotel, arranging class routines to adapt to constant changes because of staff absence and room unavailability, deploying a military unit in a war and so on. These scenarios require dynamic optimisation since the decisions need to be changed very frequently depending on the situation. Another example can be the use of medicines for cancer patients where the target is not only curing within less time but also to minimise the side effects of the drugs (Preissner et al., 2012). The problem also entails the risks of any new conditions that may arise during medications. The objectives and constraints of the problems vary dynamically from each other and are always evolving. In the above-mentioned scenarios, Evolutionary Algorithms (EA) are widely used to deal with optimisation. However, due to the dynamic nature of the problems, dynamic multi-objective optimisation problems (DMOPs) are more challenging to be solved and EAs often face difficulties to solve them (Jiang et al., 2018). On the other hand, there has been a growing interest (Arulkumaran et al., 2017) to solve the multi-objective optimisation in sequential decision making using reinforcement learning especially deep reinforcement learning after the success of DeepMind in 2015 (Mnih et al., 2015). Our study is also motivated by that success and intends to add values in a deep RL perspective to solve the problem of dynamic multi-objective optimisation.

Besides, many pieces of literature in multi-objective optimisation show a radically different perspective in solving the problems using multi-objective Markov decision process (MOMDP) especially by using reinforcement learning techniques. One of the goals of this technique is to reach the set of solutions known as Pareto-optimal solutions (POS) which determines the closeness to the true Pareto-optimal front (POF) as quick as possible. These techniques not only find the shape of the Pareto front but also help to investigate and decode interesting facts that the solutions might have (Gopakumar et al., 2018). In addition, recently Multi-objective Markov decision process (MOMDP) has received considerable attention not only for its applicability but also for solving practical multi-objective problems (Lizotte and Laber, 2016). However, according to the reward hypothesis (Sutton, 2008; Sutton & Barto, 2012), the goal and purposes can be formalised with the maximisation of the expected value of the cumulative sum of a received scalar signal (i.e. reward). In other words, the resulting MOMDPs can always be converted into single objective MDPs with additive returns. Nevertheless, Roijers et al. (2013) rejected Sutton's view questioning its application in real scenarios. They presented three static scenarios (i.e. known weights, unknown weights and decision support scenario) where authors showed one or both conversions are impossible, infeasible or undesirable. In addition, as far as dynamic multi-objective optimisation is concerned, very few studies have been conducted in this area due to the lack of testbeds (Azzouz et al., 2017a). In this research, we targeted to fill up this gap by proposing a dynamic multi-objective testbed (i.e. dynamic deep-sea treasure hunt) which may lead the researchers to do further investigation in this area. To our knowledge, this is the first work in the context of dynamic multi-objective optimisation using deep reinforcement learning. In our study, we

have argued for the necessity of dynamic multi-objective optimisation benchmark for RL settings. In addition to that, we have developed an algorithm which is primarily responsible to handle more than one objective in the defined dynamic environment. After that, we have implemented our algorithm to identify the vulnerable zones based on water quality resilience in 22 zones in São Paulo, Brazil, which ensures the applicability and efficiency of the algorithm. This implementation breaks the boundary of theoretical knowledge and helps to solve a practical problem.

The Government of the State of São Paulo is working for a universal sanitation in municipals of the state, where the increase in the percentage of the population served by the collection and sewage services is deteriorating quality of water and impeding the sustainable development of São Paulo ("Governo do Estado de São Paulo | Eleições," 2018). The presence of sewage in the waters of rivers, reservoirs, estuaries and coastal regions reduces water quality and restricts its multiple usages while increasing the occurrence of waterborne diseases caused by the primary contact or by ingestion of contaminated water (Nogueira et al., 2018). To identify the vulnerable areas and take proper actions in those areas require massive human efforts and expenses. These actions involve of integrated management of actions involving various sectors and organisations associated with the management of the use of industrial and agricultural effluents, the complexity of the HR management, fixed asset and reactive or planned maintenance (Barbosa et al., 2016). Therefore, it is important to automate the process to detect the vulnerable areas as quick as possible. To do so, an AI-based expert system can reduce the cost of managing such huge tasks and can have a socio-economic impact which may help for sustainable development. In this work, we proposed a novel algorithm called parity Q deep Q network (PQDQN) which is able to identify the vulnerable zones based on water quality resilience in a dynamic multi-objective environment. The agent in the partially observable Markov decision process detects the changes in the dynamic setting and finds the vulnerable zones. This implementation may also lead to solving some of the other dynamic real-world problems. The result outperforms the existing state of the art algorithms and our developed agent can learn the environment with reasonable elapsed time.

Research gap: We criticised as well as found that Sutton's hypothesis is not only inappropriate in the static environment but also in a dynamic MOMDP scenario, which fails to handle a multi-objective optimisation problem. We can resolve this shortcoming by offering vector reward instead of scalar reward in a changing environment. We also found the gap that the existing testbed for multi-objective optimisation is incompetent at handling a dynamic environment. We investigated and pointed out that current RL studies focus on the static environment and does not provide the facilities to anticipate the dynamic constrained environments.

Contribution: The contributions of this research work are as follows:

1. To the best of authors' knowledge, this is the first work that provides a testbed for dynamic multi-objective optimisation for RL settings.
2. We proposed a generic framework to address multi-objective optimisation using deep reinforcement learning.
3. Objective relation mapping (ORM) is used for the first time to form a meta-policy (e.g. govern policy) among different objectives to find out the compromising solutions.
4. An expert system has been developed to address the real-world applicability of the proposed algorithm which identifies vulnerable zones based on water quality resilience in São Paulo, Brazil.

Organisation: Related works with relevant literature are described in section 2. This section also gives an overall idea of Markov decision process, understanding the real-world problem and machine learning practices in water quality management. Section 3 deals with the definition of the problem along with mathematical representation. An experimental setup including mathematical expressions for the dynamic DST has been described in section 4. The high-level architecture of the proposed model and algorithm are explained in section 5. The following section deals with the results and the critical evaluation of this study. Finally, concluding remarks and future directions are illustrated in section 7.

2. Literature review

This is a threefold study. The first segment comprises identifying the gap in the dynamic multi-objective optimisation in the context of reinforcement learning. After that, a benchmark has been created to address that gap and finally, an algorithm has been proposed that can effectively optimise multi-objectives in a dynamic environment both in the simulated and the real-world scenario. In this section, we will be discussing the necessary components of this study and the recent works in this domain.

2.1 Background

Optimisation problems can be categorised into three broad areas, such as single objective, multi-objective, and many objectives problems[1]. The other classifications of the optimisation problem are whether the problem is static or dynamic. In both cases, there is a good number of research that has already been conducted using evolutionary approaches. For examples, 14 test problems for dynamic multi-objective optimisation called DF1 to DF14 were proposed (Jiang et al., 2018) based on PF/PS geometrics, irregular PF shapes, disconnectivity[2]. Multi-objective test problems with BIAS commonly known as BT1-BT9 (H. Li et al., 2017) and Large-scale multi-objective test Problems (i.e. LSMOP1–LSMOP9) are proposed by (Cheng, Jin, Olhofer, &

[1] multi-objective is usually referred to two objectives while many objectives are referred to three or more objectives

[2] existing DMOPs are a direct modification of popular static test suits e.g. ZDT and DTLZ

sendhoff, 2017). F1–F10 for IM -MOEA by (Cheng et al., 2015), C1_DTLZ1, C2_DTLL2, C3_DTLZ4, IDTLZ1, IDTLZ2 (Deb and Jain, 2014) are proposed in 2014. In addition, a comprehensive list of the benchmarks for evolutionary optimisation can be obtained in (Tian et al., 2017).

However, in the domain of multi-objective reinforcement learning, Vamplew et al. (2011) proposed 4 testbeds include a deep sea treasure hunt, MO puddle world, resource gathering and mountain car problem. Natarajan and Tadepalli (2005) used a modified version of Buridan's ass problem for MOP. Tajmajer (2017) demonstrated a cleaning robot for multi-objective adaptation in RL. The classical multi-objective physical travelling salesman problem (MO-PTSP) is utilised to explore the search space by (Perez et al., 2015). A wide-ranging standard can be found in "MORL-Glue" that represents a set of multi-objective reinforcement learning framework by (Vamplew et al., 2017b).

However, in the reinforcement learning settings, to the best of our knowledge, there is no such work on dynamic multi-objective optimisation. We have conducted intensive research and found that there is a lack of a benchmark that can satisfy the dynamic behaviours of the environment in RL settings. This paper addresses this limitation and proposes a dynamic multi-objective testbed in RL settings. The following discussions provide essential knowledge to establish the concept and formalise the problem settings for this study.

2.2 Multi-objective reinforcement learning

Multi-objective reinforcement learning (MORL) was proposed by (Zoltán Gábor et al., 1998). MORL deals with the decision-making problems in uncertain situations where the agent learns by interacting and taking feedback within the environment (Sutton and Barto, 2012). It is a promising way to solve real-world computational problems. This branch of machine learning (i.e. reinforcement learning) is selected as one of the top 10 breakthrough technologies by (MIT Technology Review, 2017). It is inspired by the trial and error process of learning, which is often followed by animals in many situations. Fig.1 shows the basic structure of reinforcement learning settings.

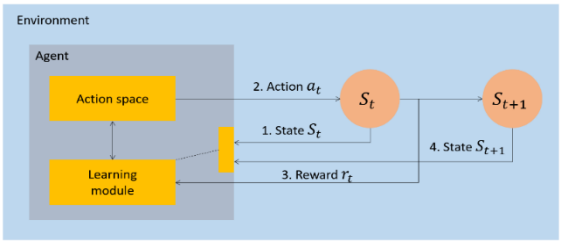


Fig. 1. A typical setup for reinforcement learning

In this study, MORL algorithm is formulated to solve multi-objective Markov decision process (MOMDP), which was pioneered by (Roijers et al., 2013). The major difference between single objective and multi-objective reinforcement learning depends on the use of reward mechanism; such as scalar reward and vector reward which are used for single and multi-objective problem respectively (Chunming L., Xin X., & Dewen H., 2015). In the MORL, the agent needs to learn either all Pareto optimal policies or identify a single policy that best matches a pre-specified trade-off between the objectives. Vamplew et al. (2017) proposed that Multi-objective reinforcement algorithms which can be classified into two broad categories. The first one is the multi-policy approach where the agent learns multiple policies to approximate the Pareto front or a subset of the Pareto front. While, in the second one, the agent learns a single policy which best satisfies the compromising solution between objectives. In addition, a MORL agent differentiates multiple policies in three broad types namely deterministic stationary, stochastic policy and non-stationary policy (Vamplew et al., 2017a). A constrained MORL method was used to optimise the average transmission delay in a cognitive radio network by (Zheng et al., 2012). In the area of decision support system, Lizotte et al. (2010) used MORL for the application of medical informatics. Parisi et al. (2016) used MORL for local utopia policy selection. Tozer, Mazzuchi, & Sarkani (2017) investigated solutions to find the optimal path in the multi-objective stochastic environment by using MORL. However, in this study, MORL is used with deep learning structure to solve DMOP. Furthermore, RL based multi-objective optimisation has been adopted for environment-economic dispatch (Bora et al., 2019) and scheduling in the workflow management where RL approach (Kintsakis et al., 2019) has been utilised. In this case, an obvious question might be − why do we need deep reinforcement learning to solve the dynamic multi-objective optimisation problem? To answer this question, let's have a close look at the deep reinforcement learning architecture in the section below where the agent learns itself like a human to achieve successful strategies. This leads to the highest long-term rewards that are constructed and learnt by interacting with the environment.

2.3 Deep reinforcement learning

Deep learning or deep neural network has been predominant in the reinforcement learning area in the last several years. We witnessed breakthroughs, like deep Q network (Mnih et al., 2015), AlphaGo (Silver et al., 2016), unsupervised reinforcement and auxiliary learning (Jaderberg et al., 2017; Mirowski et al., 2017). Deep RL has significantly reduced the reliance of the domain knowledge and enables highly efficient feature engineering which is usually time-consuming, over-specified or incomplete (Li, 2017). Fig. 2 shows a timeline for deep reinforcement learning.

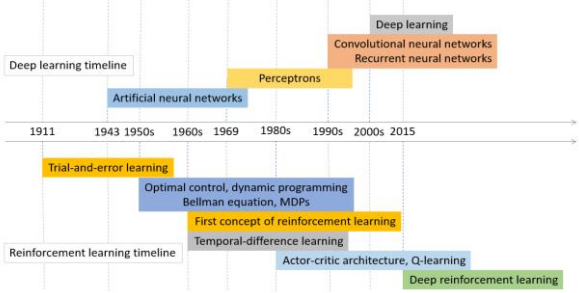


Fig. 2. Timeline for the evolution of the deep reinforcement learning

The successful implementation for deep reinforcement learning has appeared in different areas such as:

- Power System: DRL based AI decreases Google data centre bill by 40% (DeepMind, 2016)
- Robotics: Black-Box data-efficient policy search for robotics (Chatzilygeroudis et al., 2017)
- Games: Human-level Control through deep reinforcement learning (Mnih et al., 2015)
- Spoken dialogue system (Su et al., 2016)
- Information extraction (Narasimhan et al., 2016)

The following Fig. 3 shows a multi-objective deep reinforcement learning model where an agent takes an optimal action (i.e. policy) for a state in an environment and earns reward points (e.g. vector rewards for multi-objective cases). This model is formulated by the Q network, target network, emulator and experience replay.

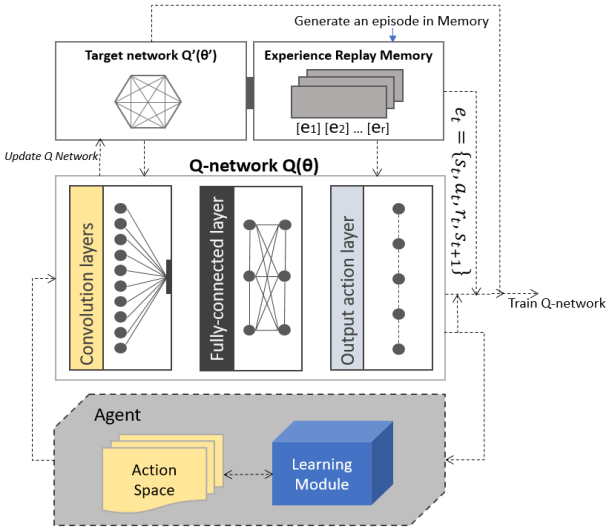


Fig. 3. The system architecture of the deep reinforcement learning

From the above figure, it is observable that the target network $Q'(\theta')$ is iterating until it gets the highest expected reward over time that can easily fit within the dynamic environment according to our requirements.

2.4 Markov decision process (MDP)

Markov decision process (MDP) is the widely studied model to analyse dynamic and stochastic system (Bertsekas and Tsitsiklis, 1996) and used to solve RL problems. This framework provides a standard model to design the system with probabilistic, nondeterministic and controlled behaviour which helps to analyse the states and necessary actions for each state. The reason for using MDP in this study is to clarify each state in the simulated environment and find the optimal policy with Q point which is often denoted by Bellman's equation (Understanding RL by Greaves, J., 2017). MDP is used in a wide range of real-world tasks such as robot control (Kober et al., 2013), game playing (Szita, 2012), clinical management of patients (Peek, 1999), military planning (Aberdeen et al., 2004), control of elevators (Crites & Barto, 1996), power systems (Glavic et al., 2017), and water supplies (B. Bhattacharya et al., 2002) and so forth. MDPs are solved by planning a model inside the MDP (e.g., dynamic programming methods, Bellman, 1958) or by learning through interaction with an unknown MDP (e.g., via temporal-difference methods, Sutton & Barto, 2012).

2.4.1 Single objective Markov decision process

In a single objective RL setting, an MDP is mostly defined with a tuple of state, action, reward, transition, probability distribution and discount factor. It is often defined as follows(Feinberg and Shwartz, 2014):

$$< S, A, T, R, \mu, \gamma >$$

Where S and A represents a finite set of states and a set of actions respectively. T represents transition function, R defines the reward function, μ is the probability distribution over initial states, γ represents a discount factor to signify the importance of short and long-term rewards.

In this single objective setting, the objective of the agent is to maximise the expected return received from the reward function R_t by a suitable policy π. Besides, for the policies which determine an agent's action, there will be a single policy or multiple policies with the same return. In the single objective MDP, an optimal policy $\pi^*$is responsible to find out the maximum expected values for all the states.

2.4.2 Multi-objective Markov decision process (MOMDP)

An MOMDP is an extension of MDP where the reward function describes a vector of n rewards that represents each objective (Roijers et al., 2013). The solution of the MOMDP is a set of policies that contains at least one optimal policy for each possible preference. In an MOMDP, a policy can outperform with one objective, however, the performance can be worst or inferior compared to other objectives. This can be summarised with Pareto dominance relationship such as a policy dominates on the other objectives if it is superior at least in one objective or equal or not worse than any other or all objectives (Lwin et al., 2014). Preferably, a dominated policy is more acceptable than the non-dominated policy (Algoul et al., 2011).

In MOMDP a state value function V^{π} specifies the expected discounted return when following a policy π from initial state S_0. This can be formulated as in Equation 1 (Ruiz-Montiel et al., 2017):

$$V^{\pi} = E_{\pi}\{\vec{R}_t\} = E_{\pi}\sum_{k}^{\infty}\{\gamma^k \vec{r}_{t+k+1}\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Here, the policy basically relies on the definition of Pareto optimality or dominance between vectors. For example, a policy π_1 is dominated over π_2, if the value of $V^{\pi_1} > V^{\pi_2}$. Similarly, π_1 is dominated or equal to π_2, if the value of $V^{\pi_1} \geq V^{\pi_2}$. There may be many other non-dominated policies which are optimal depends on the desired trade-off between policies.

For stationary policies, the state value function is defined in Equation 2 (Ruiz-Montiel et al., 2017),

$$V^{\pi} = E_{\pi}\{\vec{R}_t\} = E_{\pi}\sum_{k}^{\infty}\{\gamma^k \vec{r}_{t+k+1} \mid S_t = s\} \ldots\ldots\ldots\ldots\ldots (2)$$

It is to be noted that, a policy is Pareto optimal or non-dominated if the policy is not dominated by any other policy. The following sub-section describes the applied part of the research.

2.5 Case Study: understanding the problem in terms of water quality in São Paulo, Brazil

As discussed in the literature, many studies are focused on the academic problems where the objective functions are assumed, and constraints are pre-populated and investigated. However, it is not clear whether these problems are still related to real-world problems or not. In order to make our problem relevant and connected to the real-world scenario, we used the dataset ("Publicações e Relatórios | Águas Interiores," 2017) of water quality in São Paulo, Brazil to identify the vulnerable zones based on water quality resilience. The factors of water quality resilience are changing throughout the year due to various reasons and formed an appropriate real-world test case in a dynamic multi-objective environment.

Water quality has a strong impact on the natural environment and human life; therefore, it is a primary concern in the case of surface water and reservoir management. In recent years, dry areas have been facing severe water shortage, which is an alarming issue for the environment (J. Li et al., 2017). Recent studies reveal that about 5.5 billion people will face water crisis in 10 years (Amitrano et al., 2014). Besides, the disposal of sewage in rivers and lakes are adding more to this threat to arid areas (Lindberg et al., 2014; Zhou et al., 2014). The volume and quality of waters (Lima et al., 2018a) in reservoirs (e.g. rivers, lakes etc.) are very important not only for the environment but also for societal and economic development (Pawara, 2017). Hence, ensuring water quality is one of the crucial conditions of the overall development. An AI-based solution can enhance the whole process in an easier and cost-effective way. Reservoirs not only provide pure water for human consumption but also water for various other purposes, like agriculture, industry and habitats for aquatic lives (Hoverman and Johnson, 2012). Various properties of water in reservoirs, especially its quality, must be assessed. Assessing the quality of water critically enables managers to develop optimal water resources management plans.

In this study, our targeted area is in São Paulo that is located on the southeast of Brazil (23° 33′ S and 46° 38′ W) and its total municipal area is of 1,521.11 km² (see Fig. 4 for the targeted zones). Companhia Ambiental do Estado de São Paulo ("CETESB, Brazil," 2000) has been monitoring the quality of surface water in the state of São Paulo since 1974. The main purposes of monitoring are:- i) to make a diagnosis of the quality of the state's surface waters, assessing their compliance with environmental legislation; ii) to assess the temporal evolution of the quality of the state's surface waters; and iii) to identify priority areas for the control of water pollution, such as stretches of rivers and estuaries. The assessment of freshwater quality is complemented by temporal and spatial analysis. The presentation of water quality index (IQA), quality of water for public supply purposes (IAP), protection of Marine Life (IVA), trophic state (IET), bathing (IB) and biological communities (e.g., Phyto and zooplankton and Benthic organisms represented by ICF, ICZ and ICB, respectively), are also part of the evaluation.

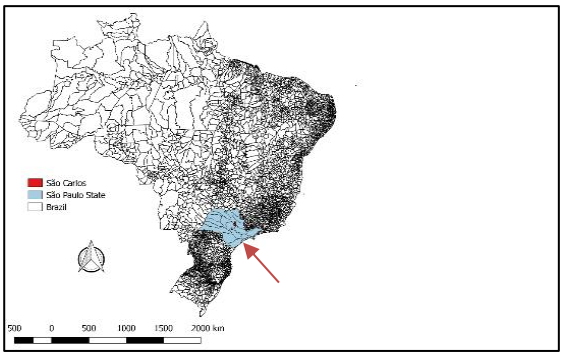


Fig. 4. Targeted area (São Paulo) to implement the proposed framework

Moreover, the frequency of data collection is different based on the different data collection points. Monitoring the quality of surface water in freshwater

bodies such as rivers and reservoirs consists of three manual sampling networks and automatic network, described in Table 1, aiming a diagnosis of multiple uses of water resources.

Table 1

Freshwater monitoring networks in São Paulo State - 2017

Monitored Sources	Goals	Start of Operation	Points	Frequency	Variables
Basic network	Provide a general diagnosis of water resources in the State of São Paulo.	1974	461	Bimonthly	Physical Chemical Biological
Sediment Network	Complement the diagnosis of the water column.	2002	26	Yearly	Physical Chemical Biological
Bathing Rivers and reservoirs	Inform water conditions for primary contact recreation/bath to the population.	1994	35	Weekly / Monthly	Biological
Monitoring Automatic	Control of domestic and industrial pollution sources and control the quality of water for public supply.	1998	12	Hourly	Physical Chemical

In this study, we have considered only the basic network which has 461 data collection points in 2017. The flow measurements in water bodies are carried out by CETESB in partnership with DAEE - Department of Water and Energy of the State of São Paulo. The results are obtained by directly measuring the flow in water bodies by reading scale to sampling the water. In 2017, the core network generated approximately 118,000 (e.g. physical, chemical, biological, bioanalytical and ecotoxicological) volume data.

The identified problems in this test-case are given below:

1. It is a dynamic problem considering the water quality data changes over time due to various factors (mentioned in section 4.2).
2. Collection of these data is expensive and required huge human resources.
3. Identification of the vulnerable zones is difficult because of manual checking and calculation.
4. Investment optimisation for different zones is complicated.
5. Prioritise the zones to enhance the water quality is time-consuming.

2.6 Machine learning for water quality evaluation

Machine Learning (ML) refers to acquire knowledge or skills through data or experiences. Machine learning has harnessed its applications in various sectors such as decision making and optimisation, medical informatics, fraud and anomaly detection, email filtering and so on. Our objective is to utilise the concept of ML and its usefulness in the above-mentioned scenario. A wide range of water quality indexes in relation to different water bodies has been used in many conventional studies using ML techniques. For example, Chou, Ho, & Hoang (2018) used four ML techniques such as ANN, support vector machines (SVM), classification and regression trees, and linear regression to analyse the quality of water. Couto et al. (2012) combined ANNs and decision tree (DT) to forecast the water quality (WQ) in a reservoir. A hybrid ANN model is developed and combined with optimisation algorithms (Chen et al., 2015) for forecasting river flow. Liu & Lu (2014) compared support vector machine (SVM) models with ANNs to predict WQ in a river. Moreover, there are several studies based on multiple linear regression methods combined with AI to develop WQ models (Slaughter et al., 2017; Tomas et al., 2017). An AI system was developed (Ji et al., 2017) by combining multiple models based on SVM, ANNs, LR to prove the supremacy of SVM in predicting dissolved oxygen (DO) concentration in Wen-Rui Tang River, China. Similarly, to predict the level of DO, a general regression neural network (GRNN) was proposed by (Antanasijević et al., 2014) for the Danube River. ML algorithms and remote sensing spectral indices have been used to evaluate the water quality by (X. Wang et al., 2017). A predictive model is developed using long and short-term memory neural network (LSTM NN) by (Y. Wang et al., 2017). A deep learning method is also used for mapping surface water (Isikdogan et al., 2017). In addition, MORL approach has also been utilised using Pareto frontier approximation by (Castelletti et al., 2013).

However, there are hardly any studies that have measured water quality “resilience” using artificial intelligence. In this study, we have used deep reinforcement learning technique to build an expert system to serve the purpose of identifying the vulnerable zones based on water quality resilience which is described in section 5.

2.7 Related work

In this section, a brief description of related works and applications have been mentioned. According to the literature, there are several approaches to deal with dynamic multi-objective optimisation. Among them, evolutionary approaches have gained more attention in the literature to solve the multi-objective optimisation problem. The following list shows several approaches in brief that have been explored in the literature:

a. Diversity-based approaches: The dynamic non-dominated sorting algorithm II (D-NSGA-II) (Deb et al., 2007), the dynamic constrained NSGA-II (DC-NSGA-II) (Azzouz et al., 2015).
b. Change prediction based approaches: This approach of solving DMOPs are useful when to exploit the past information and anticipate the location of the new optimal solutions such as dynamic multi-objective evolutionary gradient search (D- MO-EGS) (Koo et al., 2010), dynamic multi-objective EA with ADLM Model (DMOEA/ADLM) (Li et al., 2014), the Kalman Filter Assisted MOEA/D-DE algorithm (MOEA/D-KF) (Muruganantham et al., 2016).
c. Memory-based approaches: This sort of approach uses extra memory to store useful information form the past generations to guide the future search. This technique can be utilised when the environment changes slightly such as the multi-strategy ensemble MOEA (MS-MOEA) (Wang and Li, 2010), the adaptive population management-based dynamic NSGA-II (A-Dy-NSGA-II) (Azzouz et al., 2017b).

Furthermore, there are several real-world dynamic multi-objective optimisation problem mentioned in the literature. Helbig & Engelbrecht (2014) grouped and classified these applications as follows:

Control Problems: The regulation of a lake-river system (Hämäläinen and Mäntysaari, 2001), the optimisation of indoor heating (Hämäläinen and Mäntysaari, 2002), the control of greenhouse system for crops (Ursem et al., 2002).

Scheduling problems: Hydro-thermal power scheduling problem (Deb et al., 2007) and the job-shop scheduling problem (Shen and Yao, 2015).

Mechanical design problems: Design optimisation of wind turbine (Maalawi, 2011).

Apart from these, there are various real-world applications in the domain of dynamic multi-objective optimisation such as in resource allocation (Hutzschenreuter et al., 2009) and routing problems (Meisel et al., 2015) and so on. In addition, Amato & Farina (2005) have proposed an artificial-inspired EA for DMOP in the situation of unpredictable parameter changes. Azzouz, Bechikh, & Ben Said (2014) presents a multiple reference point-based MOEA (MRP-MOEA) that deals with undetectable changes.

Contrarily, as mentioned earlier, a rising approach of solving MOP is using RL technique, particularly DRL. This area gets a major concentration after the successful implementation by DeepMind to solve the Atari Games 2600 as a superhuman level with only raw pixels and scores as input (Mnih et al., 2015). To achieve human-level expertise on 49 classic Atari games, they used a single architecture and showed how the agent can successfully learn control policies in a range of different environments with only very minimal prior knowledge. It uses the same algorithm, network architecture and hyperparameters on each game. However, we have not found any literature that can satisfy the dynamics of type I, II, III and IV as mentioned in (Farina et al., 2004) in a MORL setting.

2.8 Justification of the study

Literature shows that no significant works have been done for the dynamic multi-objective environment in deep reinforcement learning settings. As a result, for the best of our knowledge, there is no real-world application in the dynamic (Farina et al., 2004) multi-objective DRL context. On the other hand, in the context of water quality resilience, there has been no use of state-of-the-art machine learning techniques. Though a few studies have been conducted in hydrodynamics (French et al., 2017) and other fields such as ocean engineering (Sarkar et al., 2015), there is hardly any research done in resilience-based approaches for water quality evaluation. As discussed earlier, to see the applicability of the well-known deep learning algorithm, researchers cannot ignore the dynamic behaviour in the multi-objective environment in the RL settings. Therefore, in this study, we incorporated this behaviour and discovered the way to fit the dynamics in the multi-objective environment. Moreover, we applied the concept of our proposed benchmark to address a real-world problem. In addition, we also developed an algorithm that can handle dynamics in the multi-objective environment. To prove the above-mentioned concept with an empirical result, our algorithm is designed to tackle the dynamics and its complexity in the simulated environment. In these environments, the proposed algorithm is integrated to examine whether it can satisfy the goal. Another aspect of this algorithm is to identify the vulnerable zones based on water quality resilience to observe whether the deep RL domain can contribute in the domain of hydrodynamics, decision support systems and to direct researchers for further investigation.

In a nutshell, a successful implementation of an AI-enabled system may have a significant impact on the following two scenarios:

1. In the simulated environment:
    a. Understanding the dynamics while objectives are conflicting to each other,
    b. Applying the existing knowledge to deal with constraints and problem parameters that change over time,
2. In the real-world problem:
    a. To categorise the impacts of water contamination on public supply in terms of resilience,
    b. Minimising the manual efforts to collect the data from different zones and
    c. Identifying the vulnerable zones that need prioritisation in the decision-making process for necessary interventions to make the diagnosis process faster while preserves the accuracy.

3. Defining the problem

3.1 Defining dynamic multi-objective optimisation problem (DMOP)

A dynamic multi-objective optimisation problem (DMOP) can be defined as a vector of decision variables x (t), that satisfies the objectives and constraints that change over time. In this section, a general approach of defining DMOP has been articulated. Generally, a minimisation or maximisation problem can be formally defined by Equation 3:

$$Min./Max.\, F(x,t) = \{f_1(x,t), f_2(x,t), \ldots\ldots, f_M(x,t)\} \setminus x \in X^n \ldots\ldots\ldots \text{(3)}$$
$$s.t.\ g(x,t) > 0\ ;\ h(x,t) = 0$$

Where, x, f, g and h represent decision variables, a set of objective functions that need to be minimised or maximised, inequality and equality constraints respectively.

In the context of decision space, the $x^*(i,\ t)$ dominates over $x\ (j,t)$. In other words, the dominating one is better in at least in one objective and not worst in any other objectives. This can be rewritten as in Equation 4:

$$F(j,t) < f\ (i,t)^* \setminus f(j,t)\ \in F^M \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(4)}$$

In this study, the dynamics of the environment is defined based on two parameters such as dynamic optimal Pareto front (DOPF) and dynamic Pareto optimal set (DPOS). DOPF can be represented as in Equation 5 with respect to the objective space:

$$PF(t)^* = \{f(i,t)^* \mid \nexists\, f(j,t)\} < f(i,t)^*, f(j,t) \in F^M\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(5)}$$

Dynamic Pareto optimal set with respect to the decision space at time t is denoted by $PS(t)^*$in Equation 6.

$$PS(t)^* = \{x_i^* \mid \nexists\, f(x_j,t) < f(x_i^*,t)^*, f(x_j,t)\ \in F^M\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(6)}$$

Moreover, the following Δf signifies how frequent the environment changes (Azzouz et al., 2017a) based on the actual PF, T (PF) and obtained PF, O(PF) as in Equation 7.

$$\Delta f = |T(PF) - O(PF)^*| \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(7)}$$

3.2 Defining the dynamics of the DMOP

In order to solve the real-world DMOPs more effectively, it is necessary to take the following characteristics and problem types into account when designing new methodologies. There are several types of dynamics in the literature of multi-objective optimisation. Frequency, severity and predictability are the prominent categories among them (Azzouz et al., 2017a). Farina et al. (2004) described four types of DMOPs according to the changes affecting the optimal Pareto Front (PF) and Pareto Set (PS). These are mentioned in Table 2:

Table 2

Dynamic MOP environment types

PF(t)*	PS(t)*	
	No Change	Change
No change	Type IV	Type I
Change	Type III	Type II

In this context, the dynamics in the environment which is controlled by a time-specific parameter τ (i.e. a changing step when the problem changes) can be represented generally as in Equation 8:

$$\begin{cases} D_t(t(\tau),\tau) = t\ (\tau) + 1 & when\ a\ change\ occurs\ in\ the\ environment \\ D_t(t(\tau),\tau) = t\ (\tau) & otherwise \end{cases} \ldots \text{(8)}$$

Therefore, the dynamic multi-objective optimisation problem in the finite period $[1, \tau^{end}]$ can be defined as in Equation 9 (Raquel and Yao, 2013):

$$Optimise\ \left\{\sum^{\tau^{end}} \mathbb{F}_{\gamma\left(t_\tau, X_{\mathbb{F}}^{G[1,\tau]}\right)}(x_t)\right\} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(9)}$$

Where, $\mathbb{F}$ represents a number of objective functions, $X_{\mathbb{F}}^{G[1,\tau]}$ is the set of solutions achieved by the applying algorithm G to solve $\mathbb{F}$ during $[1,\tau]$.

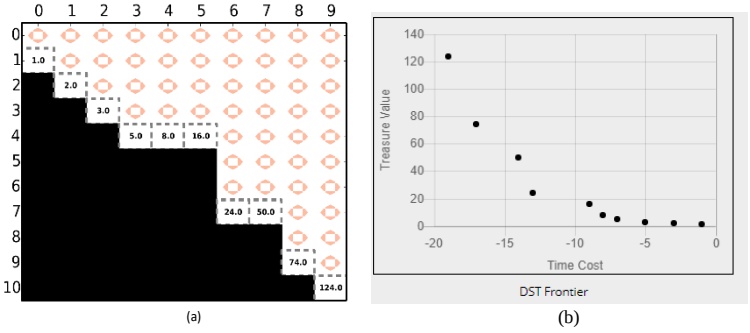
It is worth mentioning that to study the multi-objective optimisation, there are two major approaches in the literature. Researchers tried to play either synthetic or real-world problem. In this study, a simulated environment and a real-world scenario have been taken into consideration. The following section highlights the mathematical representation of the defined problem for the simulated environment.

4. Problem settings/experimental setup

4.1 Deep sea treasure (DST) hunt problem

DST is a standard multi-objective problem introduced by (Vamplew et al., 2011). This environment consists of 10 rows and 9 columns with three different types of cells such as water cells where the vessel can traverse, sea ground cells that cannot be traversed as the edges of the grid and treasure cells that provide different rewards and by reaching these cells, an episode ends. The agent controls a submarine that searches for treasures under the sea. The objectives of the agent are to find out the highest valued treasure within minimum time (i.e. conflicting way). It has got deterministic transitions with non-convex frontier. The submarine starts from the top left corner of the grid and can move up, down, right and left. Unlike the single objective environment, the agent receives vector rewards comprise of punishment of -1 for every move and the value of achieved treasure which is 0 except the agent reaches the location of the treasure. The optimal Pareto front has 10 non-dominated solutions, one per each treasure in the gird. The front is globally concave with local concavities at the treasure value of 74, 24 and 8.

An agent's objective is to find out all the state vectors for a problem where these defines a non-convex Pareto frontier in reward space. The first DST environment demonstrates a traditional one (shown in Fig. 5) with the true Pareto front and reward distributions.



(a) (b)

0	1	2	3	4	5	6	7	8	9
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-100.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-100.0	-100.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-100.0	-100.0	-100.0	5.0	8.0	16.0	0.0	0.0	0.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	0.0	0.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	0.0	0.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	24.0	50.0	0.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	0.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	74.0	0.0
-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	-100.0	124.0

(c)

Fig. 5. Traditional Deep-Sea Treasure problem (a) environment (b) frontier (c) reward distribution over the grid

4.1.1 Proposed benchmark for a dynamic multi-objective environment

To extend this environment and produce a dynamic DST, we have considered three different scenarios. The following environment (shown in Fig. 6) consists of different treasure values that change randomly over time. See Appendix A for the treasure values.

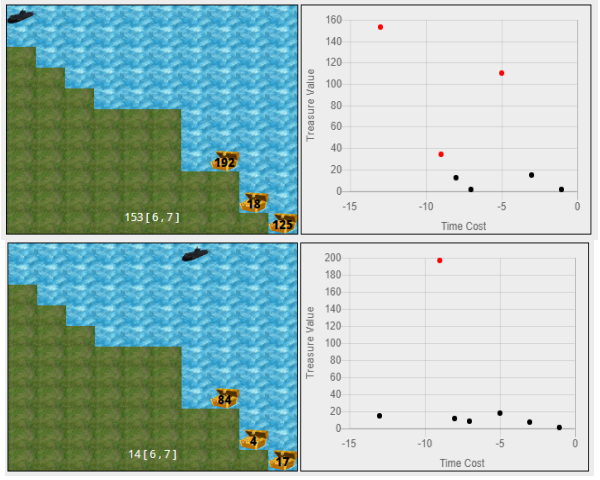
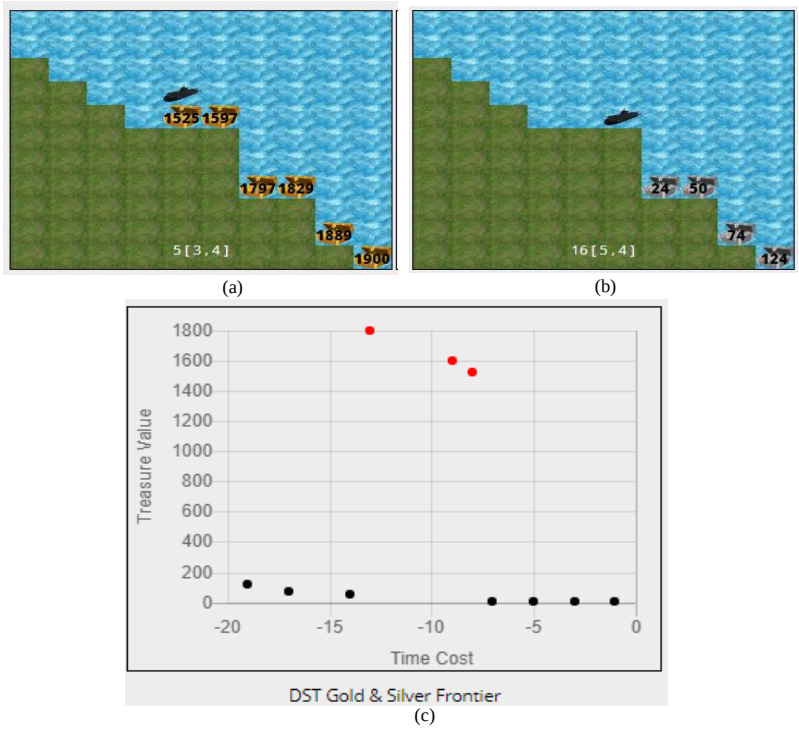


Fig. 6. Two instances of dynamic deep sea treasure (random treasure values-Type II)

As a result, this environment can be categorised in type II where both PS* and PF* changes. However, due to this randomness (i.e. ML incompatibility without having any pattern) of the environment, this environment has been excluded for further implementation and analysis.
The third environment is similar to the second one. However, the treasure values follow a fixed amount of numbers which is categorised as silver (i.e. lower value) and gold (i.e. higher value). The agent dynamically collects from the grid world. This dynamic environment can be considered as Type III where PF* changes and PS* remains invariant. The following Fig. 7 represents the third environment.

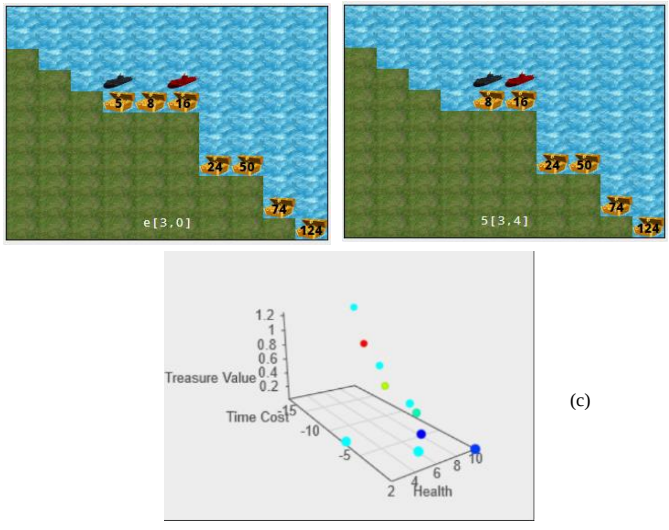


(a) (b)

(c)

Fig. 7. Dynamic Deep Sea Treasure problem- Silver and Gold (Type III)
(a) gold environment (b) silver environment (c) frontier

In the final environment as shown in Fig. 8, we have introduced an enemy submarine (which moves from the start of the gameplay) that attacks the agent randomly and in every clash between these two submarines can damage the health of the agent by -2. This health parameter creates another objective that needs to be satisfied over time to survive. This environment satisfies the dynamics in the category of type IV where both PF* and PS* remains unchanged. As the agent's vessel approaches the enemy location or intends to hit the agent, the priority becomes to save its life by avoiding clash rather than achieving the treasure (watch the attached video to get more relevant information).



(a) (b)

(c)

Fig. 8. Dynamic Deep-Sea Treasure problem- DST Attack by Enemy (Type IV)
(a and b) environment (c) frontier

For instance, to give a comprehensive depiction on how it works, the following Fig. 9 (generated by python library- *matplotlib.pyplot*) represents the movement of the two submarines in the DST (attack by enemy) environment. The arrow indicates the points of clashes between the two submarines by the agent while traversing the environment.

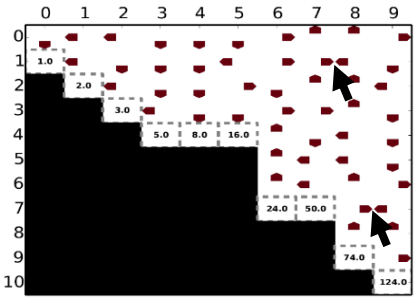


Fig. 9. The agent's traversing (clashes) in the dynamic DST (attack by enemy) in different timestamps

Now, we will be examining the Pareto frontier based on the changing time that is observed by the agent. For the visualisation purposes, the time-changing property is shown with time-unit as shown in Fig. 10 where it reflects the next time and the changing Pareto frontier based on an episode. In the first instance as shown in Fig. 10, when the agent traverse at timestamp 1, the agent has received (-1, 10, 0.01), where, -1 is the time cost, the health meter is pre-defined as 10, and the treasure value is 1. In this case for the consistency of the graph and the visualisation, the reward is divided by 100 and that's why it becomes 0.01. In another case, when the agent moves from the timestamp 2 to 3. The health condition is still 10 at this timestamp which means the agent still did not hit by the enemy submarine. Consequently, the agent achieves the treasure values which is 2 with the cost of the time penalty of -3.

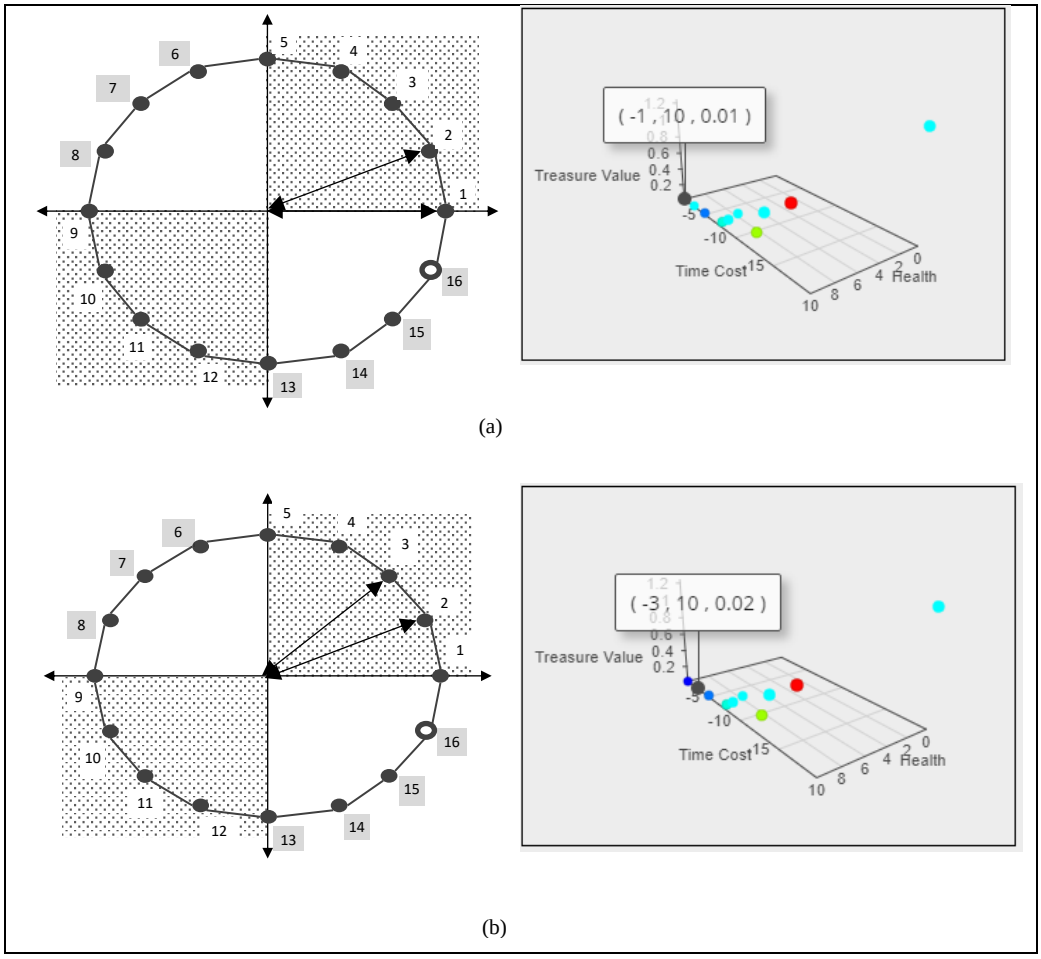


Fig. 10. Changing Pareto frontier (a) at timestamp 1 to 2 (b) at timestamp 2 to 3

As like as this environment, the other environments were also observed and traced based on the timestamp to detect changing treasure.

4.1.2 Mathematical model for the benchmark

To illustrate the problem mathematically, the components of the environment are given below:

- A submarine (i.e. black coloured) that works as an agent,
- the treasure values,
- time,
- rewards and
- health meter for the dynamic DST environment (attack by enemy)

Considering the objectives, the agent needs to hunt the highest treasures in minimum time. However, for every step, the agent is penalised by -1 point. So these two aims consist the conflicting objectives such as maximisation and minimisation for all the considered environments. In the third environment, an extra constraint has been introduced that is the health meter of the agent which is decreased by -2 by the hit of the enemy submarine. The followings are the list of the decision variables and constraints to formalise the objective functions.

Variable 1: Treasure values are divided into four categories as a tuple based on the defined benchmarks

i. $T_{silver} = [1, 2, 3, 5, 8, 16, 24, 50, 74, 124]$
ii. $T_{gold} = [100, 925, 1231, 1442, 1525, 1597, 1797, 1829, 1889, 1900]$
iii. $T_{attack\ by\ enemy} = [1, 2, 3, 5, 8, 16, 24, 50, 74, 124]$
iv. $T_{random} = rand\ [1\ to\ \infty]$, where we only considered integer values

In general, the treasure values can be expressed by the following Equation 10:

$$\text{Treasure} = \left(T_{i=0}^{i=\infty}, t\right)_{episode} \text{; based on the environments ….} \quad (10)$$

$Variable\ 2$: $For\ each\ episode\ E, the\ spent\ time\ by\ the\ agent\ can\ be\ expressed\ by\ Equation\ 11.$

$$Elapsed\ time = X^{E}_{cost(duration\ of\ traversing)} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (11)$$

$Variable\ 3$: $The\ health\ meter\ is\ decreased\ by\ j = -2\ from\ the\ pre-defined\ value\ n = 10\ by\ every\ hit\ of\ the\ enemy\ submarine\ as\ shown\ in\ Equation\ 12.$

$$health\ meter = \sum_{j=-2}^{n=j+(-2)} (H, t)_{episode} \geq 0 \dots\dots\dots\dots\dots\dots (12)$$

$Variable\ 4$: $The\ reward\ function\ for\ the\ agent$

The reward function is defined by the vector $\vec{R} = S \times A \times S'$ of n rewards in an MOMDP. This can be defined as following Equation 13:

$$\vec{R} = \sum_{k}^{\infty} \gamma^{k}\, \vec{r}_{t+k+1} \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots (13)$$

Where, the agent gets a penalty (-1) for every step during the traverse in the grid-world and $\gamma \in [0,1]$ represents the discounted factor where the reward is received after *k* steps.

Hence the two objective functions are defined as the following Equations 14 and 15:

$$\boldsymbol{Objective\ 1} \rightarrow\ Minimise\ (f_1): \sum_{i=1}^{n} (X^{E}_{cost(duration\ of\ traversing)_i}, t)\ ;\ \forall_{episode} \dots\dots\dots\dots\dots (14)$$

$$\boldsymbol{Objective\ 2} \rightarrow\ Maximise(f_2):\ (f_{treasure_{values}}, t)\ ;\ \forall_{episode} \dots\dots\dots\dots (15)$$

Subject to,

(i) $$f(traverse) = \begin{pmatrix} 0 \times 0 & \cdots & 0 \times 9 \\ \vdots & \ddots & \vdots \\ 10 \times 0 & \cdots & 10 \times 9 \end{pmatrix}; except\ the\ blocked\ areas$$

The blocked areas in the grid-world are given below:

$$(2 \times 0);\quad (3 \times 0 \cdots 3 \times 3); (4 \times 0 \cdots 4 \times 4); (5 \times 0 \cdots 5 \times 5);$$
$$(6 \times 0 \cdots 6 \times 6); (7 \times 0 \cdots 7 \times 5); (8 \times 0 \cdots 8 \times 7);$$
$$(9 \times 0 \cdots 9 \times 7); (10 \times 0 \cdots 10 \times 8)$$

The time for changing the parameters are based on current time (pc clock) and the movement of the agent must not take place in the future (pc clock).

(ii) $iteration\ for\ converging \leq maximum\ limit\ of\ the\ epoch$

4.2 Predicting water quality resilience in São Paulo, Brazil

In this test-case, a real-world scenario has been considered to validate the applicability and competency of our proposed framework. This study establishes a machine learning approach for predicting water quality resilience in São Paulo, Brazil and thus, the agent identifies the vulnerable zones. High resilience in a particular zone indicates a safe zone.

Data collected in last 17 years (2000–2017) from the stations at 22 reservoirs in São Paulo, Brazil which was preprocessed as the input for the modelling system (Governo Do Estado De São Paulo, 2018). As like as the DST testbed, this test case also provides the dynamics where the values are not static over time. These data are changing which depends on natural factors such as weather, temperature, drought, precipitation and man-made changes such as usages and contamination caused by human (Lima et al., 2018b). As the selecting criteria, we have chosen IQA, IET and IVA (described below).

IQA - Índice de Qualidade das Águas (English: Presentation of water quality index)

IQA indicates the introduction of sanitary effluents into the water body, providing an overview of the quality of surface water.

IVA - Aquá Índice de Vida Aquática (English: Protection of marine life)

IVA is used to assess the quality of water for the protection of aquatic life.

IET - Índice do Estado Trófico (English: Index of trophic state)

The Index of the Trophic State classifies the bodies of water in different degrees of trophia. It evaluates water quality for nutrient enrichment and its effect related to the growth of Algae and cyanobacteria. The IET is calculated on a priority basis for the protection of aquatic life. The following Table 3 and 4 show the variables and scales to measure the resilience respectively.

Table 3

Quality indices for basic network

Monitoring Network	**Quality Index**	**main purpose**	**Network Points**	**Variables that make up the indexes**
Basic Network	**IQA**	effluent dilution (mainly domestic)	All	Temperature, pH, dissolved oxygen, biochemical oxygen demand, escherichia coli, total nitrogen, total phosphorus, total solids and turbidity.
	IET	eutrophication	All except the rivers classified in Class 4 (CONAMA 357/05) presenting bad quality	Chlorophyll and total Phosphorus.
	IVA	Protection of aquatic life		Dissolved oxygen, pH, Ceriodaphnia dubia, copper, zinc, lead, chromium, mercury, nickel, cadmium, surfactants, Chlorophyll and total Phosphorus.

Table 4: The threshold level to determine the resilience of IQA, IVA and IET

Index	Categories					
IQA	High quality	Good quality	Average quality		Poor quality	Very poor quality
	79<IQA≤100	51<IQA≤79	36<IQA≤51		19<IQA≤36	IQA≤19
IVA	IVA≤2.5	2.6<IVA≤3.3	3.4<IVA≤4.5		4.6<IVA≤6.7	6.8≤IVA
IET	High oligotrophic	Oligotrophic	Mesotrophic	Eutrophic	Highly eutrophic	Very highly eutrophic
	IET≤47	47<IET≤52	52<IET≤59	59<IET≤63	63<IET≤67	67<IET

4.2.1 Calculating resilience

For resilience evaluation in this study, the lower bound of average quality (i.e. IQA=36, IVA=3.4 and IET = 52) for each water quality index has been considered as the threshold in resilience evaluation. This could be justified by the fact that surface water quality, within the average quality range still has maintained essential quality factors.

Resilience is calculated based on Equation 16.

$$R_i = 1 - \int_{t_1}^{t_2} \big(F_{max} - F_i(t)\big)(t_i - t_1).dt \ldots\ldots\ldots\ldots\ldots\ldots (16)$$

Where, R_i is the resilience at the time T_i, F_{max} is the maximum water quality index, F_i is the water quality index at the time T_i and t_i is the time elapsed and t_1 is the failure start time. In this study, resilience is scaled between 0 and 1, hence, different water quality variables could be compared. One of the most used scaling methods i.e. Min-Max, as shown in Equation 17, is used for this purpose.

$$R_i^n = {(R_i - R_{min})}\Big/{(R_{max} - R_{min})} \ldots\ldots\ldots\ldots\ldots\ldots\ldots (17)$$

Where, R_i^n is the normalised value, R_{min} and R_{max} are the minimum and maximum values for each water quality index, respectively. It should be noted that the worst and best water quality indexes for IQA and IVA are interpreted in a different way than IET. The higher IQA indicates a better quality of surface waters while this is opposite regards to IET and IVA (i.e. the lower is better). Therefore, the lowest value (R_{min}) is set to 0 and the highest value (R_{max}) is set to the maximum resilience value over the period of resilience evaluation.

4.2.2 Forming MOMDP to predict vulnerable zones based on water quality resilience

We have formulated the testbed as a multi-objective Markov decision process (MOMDP). This environment is based on 22 zones which have 461 stations. Each data-point is expanded over the 12 months' period where the values are changing depending on the various factors (e.g. weather, climate and so on) over time. The objectives are similar to our DST environment where the agent gets -0.04 point for every step until it gets the vulnerable zone where it gets +1 as a reward and -1 for the resilient zone. See Appendix B for the relevant procedures and the settings for the hyperparameters for this test case. A detailed description of the technique can be found in (Hasan et al., 2019- which has been submitted for the MethodsX paper).

The agent's task is to find out the vulnerable zones based on water quality resilience and rank them to identify the most vulnerable points in the targeted area in minimum time. To make it clearer, the agent's responsibility is to traverse the environment and find out the lowest IET and IVA values compared to the highest IQA. Finally, the agent makes the list according to the compromising objectives and from $\arg(min)$ to find the most vulnerable zones. Fig. 11 shows a schematic diagram of the environment.

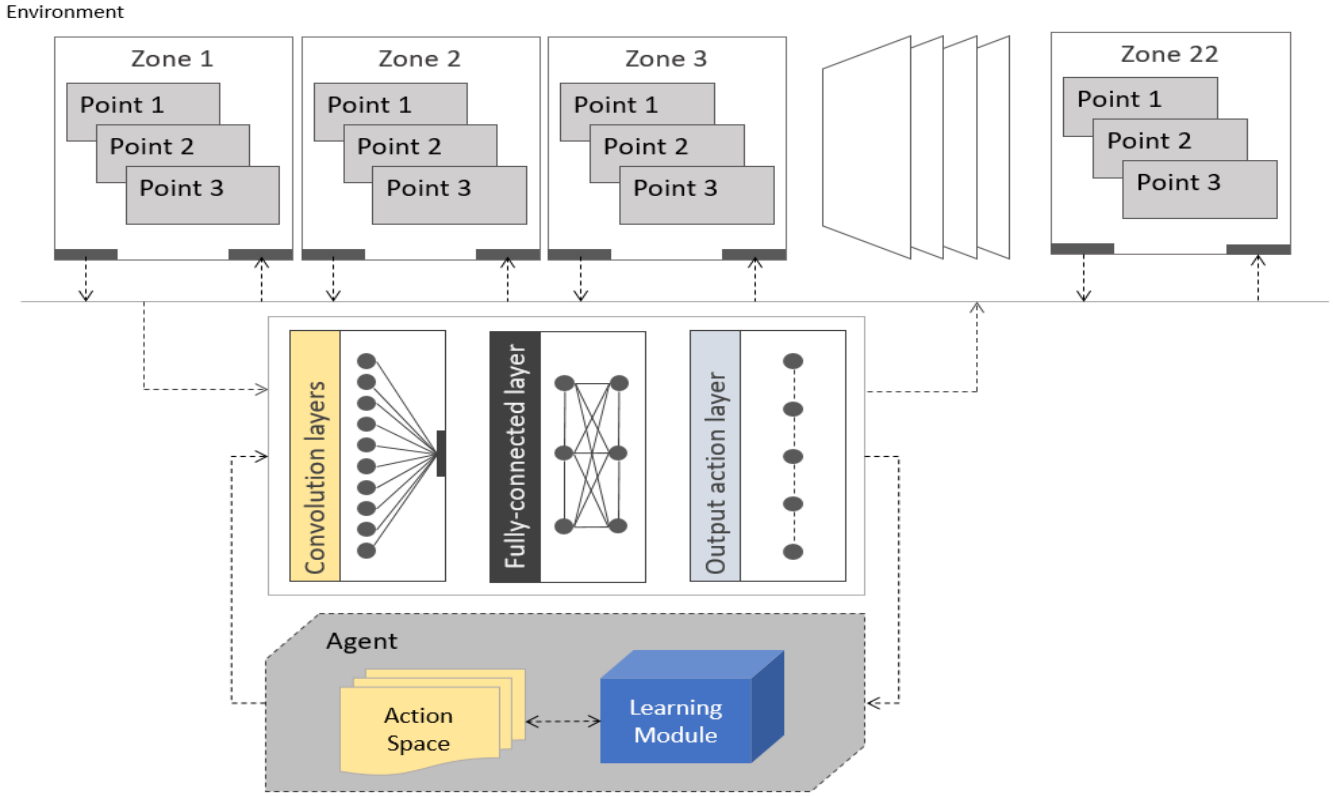


Fig. 11. The agent's interaction in the defined MDP is to identify the vulnerable zones based on the water quality resilience

Here, the procedure of formalising the multi-objective environment in the RL settings for this test case has been explained. For a comprehensive evaluation of RL, readers are referred to (Sutton and Barto, 2012). To solve the RL in this context, the Markov Decision Process (MDP) has been defined as the collection of the following components:

- States: S
- Actions: $A(s), A$
- Transition model: $T(s, a, s') \sim P(s'|s, a)$
- Rewards: $R(s), R(s, a), R(s, a, s')$
- Policy: $\pi(s) \rightarrow \alpha\pi^*$ is the optimal policy

In this case of an MDP, the environment is partially observable because of the dynamics of the environment. Thus, the agent needs a memory (i.e. experience replay) to store the past observations to make the best possible decisions. The problem settings will be clearer to understand by analysing the Markov property as below.

In the Markov property, the information of the near future (i.e. at time $t + 1$) depends on the present information at time t. A sequence of the zones $[z_1, z_2, \dots\dots, z_{22}]$ in the selected state follows the first order of Markov property as expressed by Equation 18 (Silver, 2015):

$$P(z_t|z_{t-1}, z_{t-2}, \dots., z_1) = P(z_t|z_{t-1}) \dots\dots\dots\dots\dots\dots\dots\dots (18)$$

where z_t depends only on z_{t-1}. Therefore, z_{t+1} will depend only on z_t.

However, in this scenario, the Equation can be expressed in Equation 19.

$$P(z_t|z_{t-1}, z_{t-2}, \dots., z_1) = P(z_t|z_{t-1}, z_{t-2}) \dots\dots\dots\dots\dots\dots (19)$$

where, z_t depends on z_{t-1} and z_{t-2}.

Thus, the stations can be converted to a Markov property if the probability of the new state depends on the next state and claim that z_{t+1}, depends on the current state, z_t, such that the current state captures and remembers the property and knowledge from the past. Therefore, as per the Markov property, the grid-world (i.e. the environment) is considered to be stationary (Feinberg and Shwartz, 2014). However, we have type III dynamics in this environment as mentioned in (Farina et al., 2004).

We considered a finite block of 22 zones with their corresponding data stations that are changing based on the monthly data. For each station, we formulated the velocity of the data in a continuous manner where the data changes at the next time step $t + 1$ that is only determined by the current system state. It is independent of the previous states due to the changing values of the monthly data such as weather (e.g. drought, precipitation) and man-made changes such as usages and contamination. Therefore, the stations of each zone can be treated as an MDP (i.e. see a sample in Fig. 12).

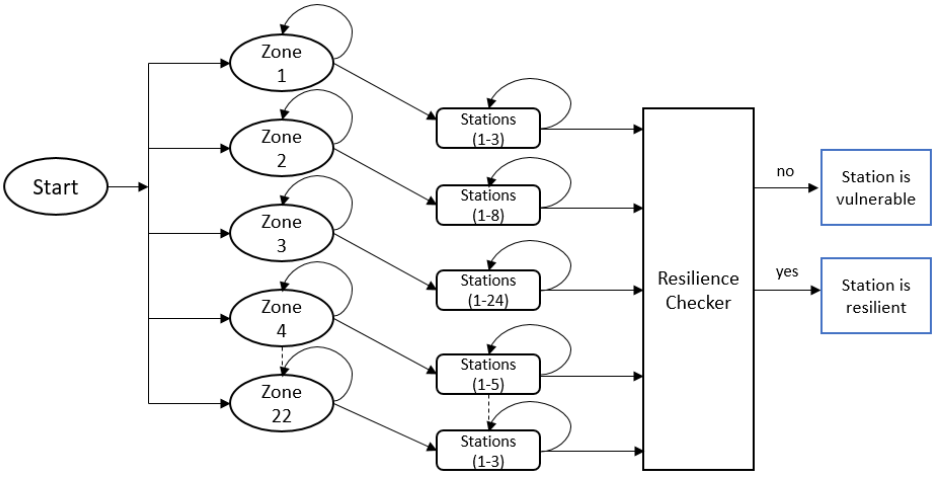


Fig. 12. MDP for the resilient area selection

Here, the key concepts to facilitate the DRL-based WQR prediction have been formulated. The terminal of each zone is related to the dataset which is chosen from the discrete levels, denoted as $Z = \{Z^1, Z^2, \dots., Z^n\}$. Therefore, the entire action space is denoted as: $A = \{A^1, A^2, \dots., A^n\}$. On the other hand, the action space is determined based on the observation of the multi-objective optimisation and the selection on the actions based on the satisfaction of the objectives (maximising IQA and minimising IET, IVA). In this work, we considered the current (physical) time and the values of IQA (i.e. the higher the better) and IET and IVA (i.e. the lower the better) to determine the optimal control action.

In particular, incorporating current time information in the state enables the DRL algorithm to adapt the time-related activities, such as time-varying data for months that changes across the year. This is important because the weather pattern varies significantly throughout the year and thus, the recorded dataset reflects the changes on a monthly basis. Hence, it is required to facilitate the system and give the flexibility for the MOMDP to adapt to time-variant variables.

Considering a reward function, the agent does the job by taking a sequence of actions a_{t-1} at state s_{t-1}, the block will evolve into a new state S_t and the

DRL algorithm will receive an immediate reward r_t in the MOMDP, the reward function $\vec{R} = S \times A \times S'$ is a vector of n rewards rather than a scalar with an element for each objective. Similarly, the reward function is also the vector value such as $\overrightarrow{R} = (s, a, s')$. This can be defined by Equations 1 and 2 as mentioned in the first test case.

The rewarding factor includes a penalty of (-1) for detecting the wrong station and (+1) for detecting the right one (i.e. vulnerable). During the traverse in the block, we want to maximise the accumulative reward R, where $\gamma = [0,1]$ is a decay factor. Agent's target is to find out the optimal policy (π^*), which maximises the projected reward. The optimal policy, in this case, is responsible to maximise the amount of reward received or expected to receive over a lifetime. Thus, in this method, the policy is nothing but a guide to direct which action needs to be taken in a given state. Thus, the agent determines the resilient area with the highest IQA values compared to the lowest IET and IVA.

To dig into deeper, let us consider the following environment and the given information. The agent in the MOMDP needs to identify the resilient zones based on the IQA, IET and IVA. The MOMDP represents a grid-world as like as the first test case. This grid-world is formed into states (i.e. zones with stations), actions (i.e. which zone to traverse), transition models (i.e. the selection from one zone to another) and rewards (i.e. achieved by identifying the resilient zone). The solution for this MOMDP is an optimal policy that relies on the vector rewards which determines the critical as well as the resilient zones. These rewards are connected to the objectives to find out the optimal policy. Since, the dynamic nature and partially observable MOMDP, the agent requires an experience replay as to store the past observations to make the best possible decisions which are close to the Pareto Front.

Consequently, continuous input from the agent can play an important role in state formation. Hence, the state spaces can be either discrete or continuous. The agent starts from the start state S_1 and has to reach the resilient zone by calculating the parameters (e.g. IQA, IET and IVA). The states are represented by the coordinates of (x, y) whereas the actions are the execution by the agent in a particular state. In other words, actions are sets of things that an agent is allowed to do in the given environment. Consider this example, 22 discrete states with 4 discrete actions such as up, down, right and left. Therefore, the action, $a \in A$ where, A = *{up, down, right and left}*. Needless to mention, this action can be treated as a function of the state, that is, $a = A(s)$, where depending on the state function, it decides the best possible action in the current state based on the vector rewards. The transition model is defined by the current state (s), action (a), and the new state (s'). This can be represented as $T(s, a, s')$ that defines the rules to traverse in the environment. It gives the probability $P(s'|s, a)$ *for* the new state s'.
Let us consider the following information is pre-defined for this environment:

- Agent traverse between the Zones Z_1 to Z_{22} and their corresponding stations within a grid-world.
- Once the agent reaches the goal state G (critical stations), it receives a reward of +1. However, for every step, it gets -0.04 rewards.
- The agent gets -1 for selecting the wrong station C.
- Thus, these two states (i.e. G and C) are the terminal states, by reaching any of these two, the traverse is over. If the agent encounters the G state, the agent wins, while if they enter the other one, then the agent loses the game.
- Discounted factor $\gamma = 0.9$, the utility at the first-time step is 0, except for the G and C states.
- Transition probability $T(s, a, s')$ is equal to 0.8 if the agent traverses the preferred direction; otherwise, 0.1. For example, if the optimal location is up and the agent does the same then the action's probability is 0.8 otherwise for every movement the probability is 0.1.

In the deep layer for this test case, as shown in Appendix B, stochastic gradient descent (SGD) optimisation technique is used. The output layer is connected with multiple groups of nodes. The number of groups is identical to the number of objectives (e.g., time vs. treasure value or resilient data). Each group comprises several nodes that correspond to the number of possible actions which leads to governing the policy. Technically, the DQN architecture has been created using TensorFlow and Keras and visualization and fine-tuning have been done using TensorBoard.

Given the RL framework using MOMDP discussed above, we need to find out the optimum Q value from different deep Q networks for all actions $a \in A$. These DQNs represent the state-actions portfolio that needs to be selected by an optimal policy as described in (Silver et al., 2016). A convolutional neural net has been used as the Q network estimator to separate state-value and actions. State-action networks learn the Q-values for each action a_1 given a state S_1, at some particular time step t by minimising the temporal difference error.

5. Solution

In general, our target is to develop the algorithm which can ensure convergence and diversity. To implement this, a well-established reinforcement learning can be approached such as dynamic programming, Monte Carlo tree search and temporal difference learning (TD). To solve the defined problem, we need to find out all the non-dominated policies at once as explained in (Barrett and Narayanan, 2008). Considering the multi-policy search simultaneously and updating the current state, we have chosen a temporal difference learning mechanism that is model-free and off-policy such as Q learning algorithm. Moreover, it has been considered because of its capability to determine the optimal action-value function which learns from the reward function in an MDP.

5.1 Q learning

Q learning (Watkins and Dayan, 1989) is a model-free technique in a reinforcement learning settings which is used to learn an optimal Q(s, a) for the agent in an MDP (s, a, r, s′, π). It is also off-policy, optimised value function and most importantly, this fits for our simulated environment. In addition, it learns a unique policy when rewards are scalar values. This scalar-values $Q(s,a)$: $S \times A \rightarrow$ R, that represent the expected accumulated reward when following a given policy after taking an action a in state s. The action a is selected by the policy in each state is given by the expression $argmax_a Q\ (s,a)$. In single-objective MDPs, a deterministic stationary optimal policy is very likely to be found. In this case, the expected return is usually an undiscounted finite sum of the scalar rewards until the agent reached the terminal states.

In Q learning, a decision-making agent interacts with the environment through a sequence of steps. In the nth step, the agent observes its current state S_n, selects and performs an action a_n, observes its current state s' receives an immediate reward r_n and adjusts the value $Q(s_n, a_n)$ using a learning factor $\alpha_n (0 < \alpha_n \leq 1)$. The updating expression of the scalar Q-learning algorithm can be expressed using the Bellman equation (Bellman, 1958).
However, in the context of dynamic multi-objective optimisation, we need to tweak the Q learning so that it can work with vector rewards. Thus, we need to extend the equation that can handle vector operations. Therefore, in the finite horizon or an episodic environment, the reward function $\vec{R} = S \times A \times S$ is a vector of n rewards rather than a scalar with an element for each objective.
In the vector reward space, an action a in state s under any non-dominated policy can be denoted as $\overrightarrow{Q^*}(s,a)$. Here, the fact is the obtained reward is a vector where the agent does not learn a single policy but a set of policies at the same time. The values learned in this scenario can be denoted as $\vec{Q}(s,a)$ of vectors, which are used to estimate the optimal $\overrightarrow{Q^*}(s,a)$ sets.
This can be denoted in Equation 20 which is inspired from the enhanced proof of Bellman equation mentioned in (Gross, 2016):

$$\overrightarrow{Q_n^*}\{(s,a),t\} = \begin{cases} (1-\alpha_n)\vec{Q}_{n-1}(s,a) + \alpha_n[\,\vec{r}_n + \gamma\, V_{n-1}(s'),t]; & if\ s = s_n \wedge\ a\ =\ a_n \\ \vec{Q}_{n-1}\{(s,a),t\}; & otherwise \end{cases} \quad \ldots\ldots (20)$$

$$\text{Where, } V_{n-1}(s) = \max_{a \in A} \vec{Q}_{n-1}\{(s,a),t\}$$

5.2 Deep Q network selection

Mnih et al. (2015) introduced Deep Q-Network and ignited the area of RL. We critically reviewed the deep Q network and we are also aware of the drawbacks of using Deep Q network such as incompatibility in the non-Markov model (i.e. next state always relies on the current state), overestimation for continuous action space, overfitting and poor exploration due to sparse feedback. The common extensions of the DQN architecture are double DQN (Van Hasselt, Guez, Silver, & Deepmind, 2016), prioritised experience replay (Schaul, Quan, Antonoglou, Silver, & Deepmind, 2016), duelling architecture (Wang et al., 2016). In addition, Anschel, Baram, & Shimkin (2017) proposed an average of previous Q-values estimation to reduce inconsistency and instability. He et al., (2016) proposed to accelerate DQN by optimality tightening to propagate reward quicker as well as improve accuracy over DQN. Furthermore, several multi-policy algorithms have been analysed such as multi-Pareto Q learning (Ruiz-Montiel et al., 2017), multi-policy DQN (Nguyen, 2018), multi-objective Monte Carlo tree search (Wang and Sebag, 2012). The following algorithms as shown in Table 5 are considered based on the policy evaluation in DQNs:

Table 5

Comparison of the analysed algorithms

Item	**Algorithms**				
	Deep Q Network (DQN)	Double DQN (DDQN)	Multi-policy DQN (MPDQN)	Multi-objective Monte Carlo Tree Search (MO-MCTS)	Multi-Pareto Q Learning (MPQ)
Support Multi-policy	✕	✕	✓	✓	✓
Vector reward	✕	✕	✓	✓	✓
Pros	It performs faster while training the network using random mini-batches from temporary memory instead of recent transitions	It performs better to reduce observed overestimations compared to DQN learning	Learn parallel multiple policies and adapt quickly when there is a change	It acquires multiple unsupported policies in a single run by constructing a tree-walk using upper confidence bounds (UCB)	Finds all deterministic PF policies for an MORL settings

Cons	Overestimation and the agent follows greedy approaches to fix the Q value, it often may lead to less optimized policy and increase time complexity	Single stream double DQN performs worse than Dueling DQN (DuDQN)	It may be stuck in local optima due to the assumption of optimal policy in advance	The agent does not support bootstrap and has to wait for the outcome to update the predictable reward	Requires high convergence time
Used in a dynamic environment	×	×	×	×	×

In the context of the dynamic environment and handling MOMDP, we only considered the algorithms which support multi-policy (i.e. Multi-policy DQN, MO-MCTS, and MPQ).

5.3 Meta-policy selection mechanism

As far as the dynamic environment is concerned, we have introduced a meta-policy (i.e. governing the policies in the policy life-cycle) that defines which policy needs to be counted and prioritise the objectives. However, it is difficult to define the meta-policy with respect to all the objectives. To overcome this problem, a parity value (i.e. dynamic weight) has been introduced before summing up the Q-values from DQNs that ensures equilibrium between objectives. Moreover, the agent helps to devise the action selection mechanism that can lead to a better decision. The meta-policy is selected by updating the process of selecting each action by choosing $Q\ (s, a)$ as follows:

a. The set of $Q(s, a)$ is updated with vectors from S' at time step t for the first time after performing an action a in state S. In the meantime, actions in S' are sampled at the time $(t + 1)$ that is previously unexplored.
b. As estimations are being done over time, it is likely that the $Q(s, a)$ can be created, updated and deleted at time step $(t + 1)$. As a result, vectors in S' that can be dominated by other vectors (van Moffaert & Nowé, 2014).

However, in the dynamic environment, the location of the optimum moves either deterministically or stochastically. This move can also be linear, non-linear, periodical or random over time during optimisation. The core difference between stationary and dynamic optimisation problems is any component that exists in the environment gets changes over time in the latter case. In the defined problem with time-varying, the desired goal of the algorithm is:

a. to find the global optima to detect the changes and
b. tracking the changing optima over time

To identify the meta-policy in the changing environment the followings need to be done:

a. detecting changes by re-evaluating detectors or changing components
b. detecting changes based on the algorithmic behaviours
c. balancing the objectives using objective relation mapping based on a dynamic weight (parity value)

In an MOMDP, where the dynamics of the problem changes to different trajectories through the metric space $\delta(t)\ \in\ \Delta$, are the parameters of the function t.

$$\delta(t) \rightarrow\ \ \delta(t+1)\ \rightarrow\ \delta(t+2)\ \rightarrow \cdots$$

Where, T is a period index such that $(t) \neq \delta(t+1)$, $\forall T$

The following Fig. 13 shows the decision, objective space and tracking the optima over time in a dynamic environment which is observed by detecting the moving optima (Lam T. Bui, Branke, & Abbass, 2005).

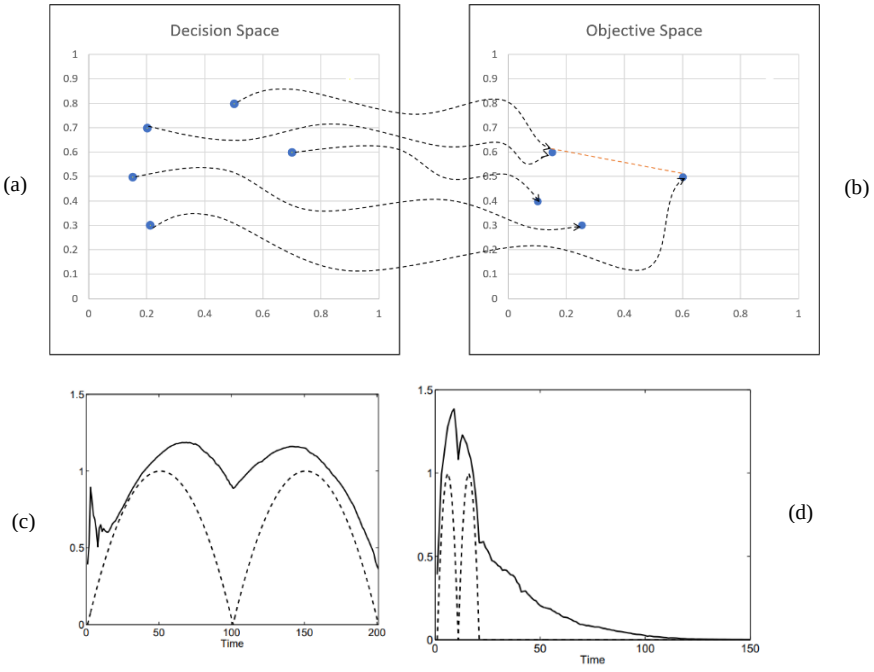


Fig. 13. Objective space and the tracking of global optima: (a) decision space (b) objective space (c) tracking slow-moving optima and (d) tracking fast-moving optima

The following Fig. 14 demonstrates the Q value mappings for the different policies in DQN structures. For the visualisation purpose, we have colour coded the selected policies whereas policy is selected based on the yellow columns of different objectives. Eventually, the MORL agent produces the green one which is the mapped version of the meta-policy.

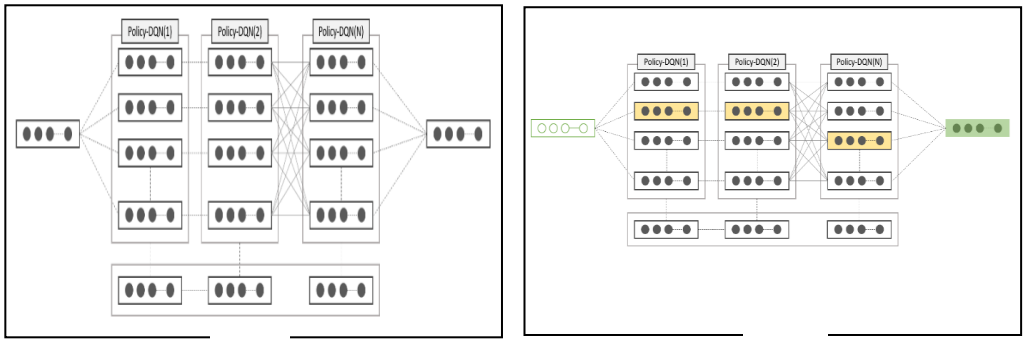


Fig. 14. Q value mapping in DQN architecture for governing policies to detect the changes of the optima (a) without mapped (b) mapping among different objectives

The meta-policy of selecting a policy is defined by the parity value which is obtained by the objective relation mapping (ORM) in a dynamic environment to satisfy or closely satisfy the objectives. From Fig. 14, it is also observable that the policy is selected based on the individual input of the DQNs which are the representatives of the state and action for a particular objective.

The agent selects an action a in the state s to form the vector of $\vec{Q}(s,a)$ that is composed with the reward vectors $\vec{r_t} = [\vec{r}_1, \vec{r}_2, \ldots . \vec{r}_n]$. For each pair of the policy $\Pi(i) = \vec{Q}_i(s,a)$ and the next policy $\Pi(i+1) = \vec{Q}_{i+1}(s',a')$, the agent creates a matrix as in Equation 21:

$$\Pi_{\mathrm{n[i],n[i+1]}}\left(\theta(t)\right), where: \theta(t) = Dynamic\ changes\ (\theta(t-1)) \ldots\ldots\ldots\ldots (21)$$

A transformation matrix is obtained by the following Equation 22:

$$\Pi_{transform} = \Pi_{\mathrm{n[1],n[2]}}\left(\theta(t)\right).\Pi_{\mathrm{n[3],n[4]}}\left(\theta(t)\right) \ldots\ \Pi_{\mathrm{n[p-1],n[p]}}\left(\theta(t)\right) \ldots\ldots\ldots (22)$$

Update the Pareto Front (PF) position by the following Equation 23:

$$X(t+1) = X(t).\Pi_{transform} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (23)$$

Where the changing severity $\Delta f = |T(PF) - O(PF)^*|$is set to different points based on true PF, T (PF) and obtained PF, O (PF).

The following Fig. 15 shows the relation mapping for different objectives based on selected policies where the convolutional layers direct to form a compromised solution. This process is predominantly executed by the neural network of the deep layer and by adjusting the weights and bias for each neuron (Schmidhuber, 2015).

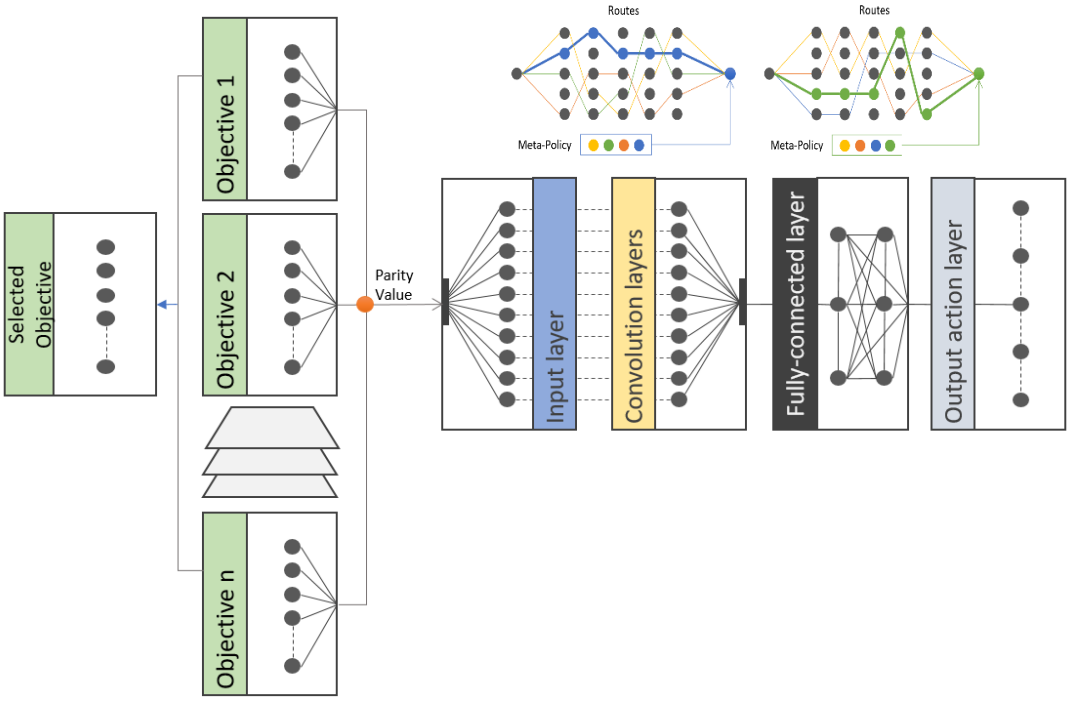
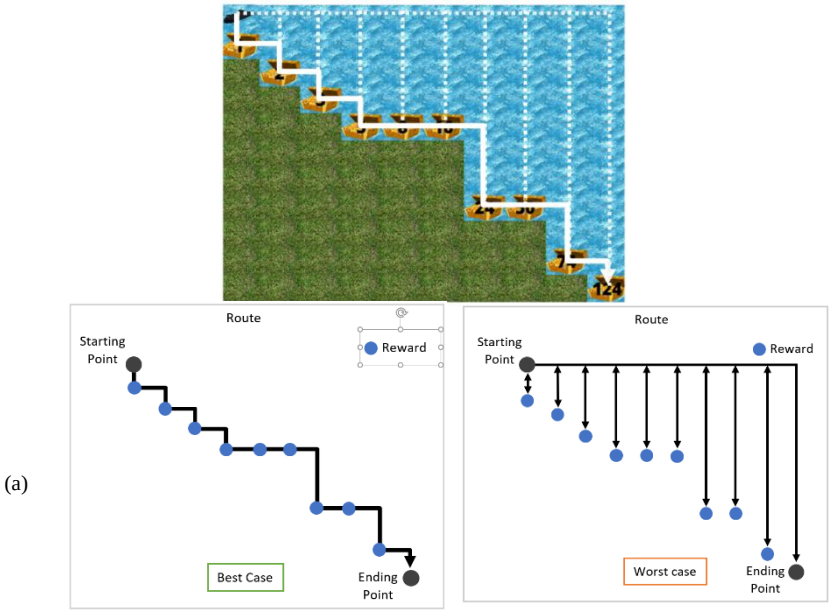


Fig. 15. Object-relation mapping to find the equilibrium between objectives in a PQDQN architecture

From the above Fig. 15, it is observable that the agent determines the compromising solutions based on the balance of the several objectives. These Q values are forwarded by the DQNs which consist of the set of state and action values for a particular episode that is generated by the emulator. In the deep layer, the weights of the neural network are adjusted based on the backpropagation procedure (Baldi and Sadowski, 2018). The obtained Q value by the agent is the average value from the DQNs that characterise all the objectives. Therefore, the selected objectives are the representation of the compound structure of all the objectives. Needless to mention, this value is made by the Q values in a finite horizon. In other words, this can be represented as the most compromising solutions that the agent could achieve in a particular episode.

The following Fig. 16 shows a visualisation of the DST environment where the agent traverse based on different meta-policies.



(a) (b)

Fig. 16. Visualisation of the trajectory based on the meta-policy (a) the best case-scenario and (b) the worst-case scenario

It is worth mentioning that in a special scenario, the changes can be undetected and thus, the moving optima cannot be identified. As a result, the solution cannot be reached.

5.4 High-level architecture of the proposed algorithm

The proposed solution consists of two different blocks which represent a common RL setting such as selecting states and actions. The agent interacts with the environment and selects the actions based on the ORM. The following Fig. 17 shows the components of the proposed algorithm to handle dynamic MOMDP.

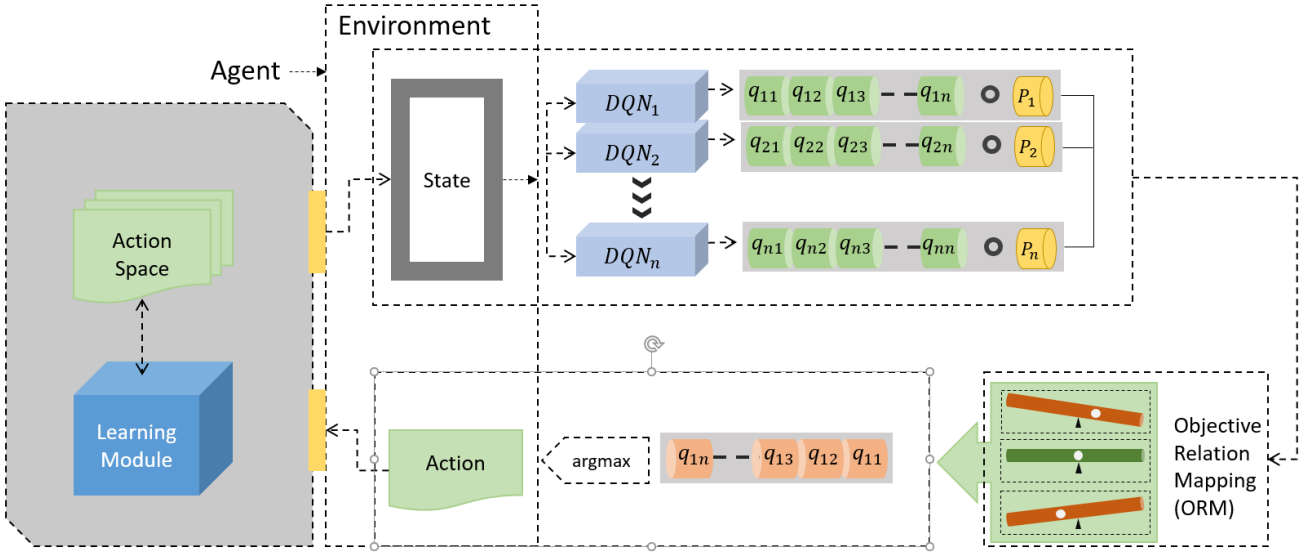


Fig. 17. High-level architecture of the proposed parity-q deep q network

Where,
Agent: The agent or algorithm lives in the simulated environment and helps to make the decision.
State: A state helps to identify the next step which will be determined by the agent. Here, a state S_t is Markov iff $P[S_t + 1 \,|S_t\,] \;=\; P[S_t + 1 \,|S_1, \ldots, S_t\,]$
Action: Agent's possible moves between different states by observing new states and receiving rewards.
Policy: A policy is typically represented by the agent's behaviours of the selection of the action.
Environment: An environment is the external entity of the agent where it interacts with the states. Here, DSTs and WQR are pre-defined environments.
Reward: The agent has a specific task which needs to be performed by the actions to earn points (e.g. negative or positive).

From the above Fig. 17, it is observable that the Q values are selected based on the states and actions. The state is selected based on the DQN networks and this value is sent to a stack where the agent looks for the best Q value. After that, the highest value is mapped with the ORM module where it forwards the best compromising objectives after satisfying the constraints. This module balances the Q values to achieve the possible optimal Q values for the different objectives by averaging them or multiplying with the preference values (e.g., if any). This action is performed based on the balancing of all the objectives. Therefore, the solutions that are provided by the PQDQN must be a compromising solution if there are more than one objectives are available. From this module, the Q value forwards to a buffer where the final action gets selected and the agent has learnt the environment in an episodic style. In the buffer setting, the first Q values pass to the agent since it follows the queue mechanism. It is worth stating that the highest Q value passes because of the *argmax* operator.

In addition, while traversing, the agent ensures that the traversing is completed to visit all the nodes so that there is no unexplored state. Thus, the agent interacts within the environment and learns the optimum values which lead to select the policy as discussed in the earlier section.

The target of our proposed algorithm is to detect the changes and then tracking the changing optima (i.e. local optima or ideally the global optima) over time. From the below algorithm 1, it is noticeable that we need to provide vector rewards for each action and prioritise the objectives (i.e. if needed as like as the DST-enemy attack environment to prioritise health more than treasure). Therefore, unlike the previous one, this is a 3 objectives environment. After that, the agent needs to convey the state-action pair into a deep Q network and get the highest achieved Q value (multiply with the preference value -if any) in the stack for each episode. Furthermore, the agent makes a map among the objectives with parity-Q and scalarise the q value to determine the non-dominated solutions (if any) and distance from each other. Finally, send it to a buffer for argmax and finally select the action and update the target network. The following algorithm 1 shows the proposed PQDQN algorithm.

Parity-Q Deep Q learning Network (PQDQN):

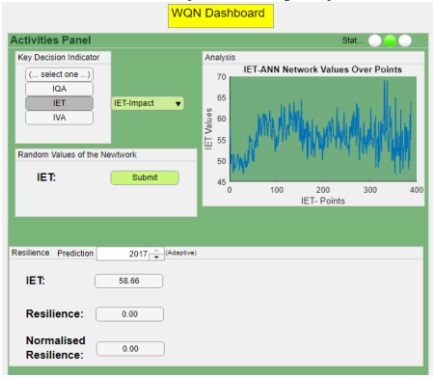
Line 1: Initialise: Temp Memory T // temporary list
Line 2: Initialise: Set action value-function Q with random/arbitrary numbers
Line 3: Initialise: Set two fully connected layers for the n^{th} objective (two/three different objectives) attached to the vector reward $\vec{r}_{t+k+1}$ for multiple policies
Line 4: Initialise: Set the temp functions for each objective
Observe: initial state S
Line 5: for each episode e ∈ {1, 2, . . . ,M } do
Line 6: observe reward r and new state s'
Line 7: if (state!=0){
Line 8: select an action a arbitrarily with possibility of μ
Line 9: else
Line 10: select action $a = Max\, a'\, Q\,(s, a')$ // identified the highest Q value to select the action
Line 11:}
Line 12: store experience in Temp $T\,[s, a, r, s']$
Line 13: Get sample transitions from Temp T
Line 14: Create a list/table R (to store combined Q values)
Line 15: While (Transitions!=0 in store experience){

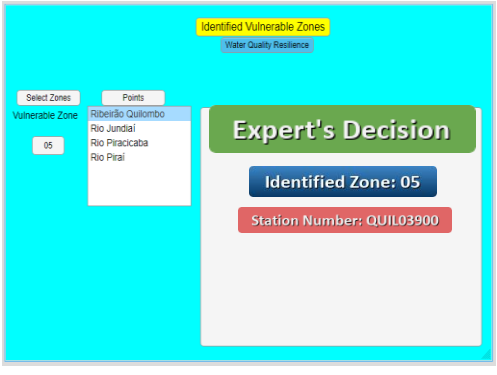
Line 16: If (ss'== terminal state)
Line 17: tt= rr+yMax$_{a'}$ Q(ss', aa')
Line 18: else
Line 19: set tt=rr
Line 20: Train the Q network with pre-defined loss
Line 21: s=s'
Line 22: end
Line 23: Create a stack for objective 1, $\hat{S}_1$ which consists of $[E_1 \ldots\ldots E_n]$ for the value functions
Line 24: while ($\hat{S}_1$ is not empty)
Line 25: Push $in\ stack\ \hat{S}_1 \rightarrow E_1{:}\{Q_1\,(s_1, a_1)\ value\ of\ Objective_{1(\vec{r}_{t+k+1})}\}$
Line 26: If $(E_1 < E_n\ )$
Line 27: Pop the stack $\hat{S}_1$
Line 28: Insert E_n into the stack
Line 29: Otherwise, Insert E_1 into the stack
Line 30: end
Line 31: Create a stack for objective 2, $\hat{S}_2$ which consists of $[E_1 \ldots\ldots E_n]$ for the value functions
Line 32: while ($\hat{S}_2$ is not empty)
Line 33: Push $in\ stack\ \hat{S}_2 \rightarrow E_2{:}\{Q_1\,(s_1, a_1)\ value\ of\ Objective_{2(\vec{r}_{t+k+1})}\}$
Line 34: If $(E_2 < E_n\ )$
Line 35: Pop the stack $\hat{S}_2$
Line 36: Insert E_n into the stack
Line 37: Otherwise, Insert E_2 into the stack
Line 38: end [continue the push/pop block up to n^{th} objectives]
Line 39: Select the action <association map $(E_1 \sim E_2)\ ||\ (E_1 \sim E_n\ )$> based on the dynamic weight to balance the objectives (i.e. parity value)
Line 40: Store the value of Q = $\overrightarrow{Q_n^*}\{(s, a)\}$in a Buffer for enqueue and dequeue at time t (i.e. changing optima).
Line 41: Select the argmax (Q)
Line 42: Update target network $Q^* \rightarrow optimum\ value\ among\ objective_1, objective_2\ and\ objective_n$

5.5 Solving the real-world problem by the developed expert system

AI techniques may involve complex manual implementation that can reduce their practical efficiency for water quality managers. Practitioners need to be trained in programming or coding to apply such complicated models. Hence, an easy user interface (UI) based expert system or application (shown in fig. 18) is developed to serve the purpose. The developed system is created based on MATLAB (2018a) with python version (3.5) with the libraries (i.e. tensorflow and keras). See Appendix C for the implementation of code. We have considered the IQA, IET and IVA values and have calculated the predicted values by the agent for each zone and showed in the normalized forms (resilient or not). Finally, we showed the values for the vulnerable zones based on the water quality resilience that is determined by the meta-policy of our developed PQDQN.

(a)





(b)

Fig. 18. Expert system (a) predicting water quality resilience (b) identifying the vulnerable zones

The Fig. 18(b) shows the identified vulnerable zone is 5 which has 32 stations and among them, the most critical point is Ribeirão Quilombo based on water quality resilience.

6. Empirical analysis and discussions

6.1 Results

In order to determine the performance, we compared the proposed algorithm with the well-known deep Q learning algorithms which can handle multi-policies such as multi-policy DQN (MPDQN), Multi-objective Monte Carlo Tree Search (MO-MCTS), and Multi Pareto Q learning (MPQ).

To make a fair comparison for the different algorithms, we used the following learning hyperparameters (shown in Table 6) for every algorithm.

Table 6

Hyperparameters

Parameter	Value
Learning steps	3 Million
Agent's Evaluation (Interval)	1 Million
Replay memory size	10000
Target network update rate	1000
Learning rate	0.001
Exploration rate	0.4
ε end step	1000
Discount factor	0.97
Optimiser	Adam
Batch size	32
No-op max	40

In this structure as shown in Table 6, an experience replay, and a target network have been utilised to stabilise the network and the learning procedure. To reach all the states including the unvisited nodes, a ε-greedy exploration policy has been implemented with annealing from 1 to 0.05. The discount factor is $\gamma = 0.97$, and the target network is reset in every 150 episodes. To stabilise the learning, mini batches of 32 is used where the number of training chunk for each Adam update is computed. The replay memory size is set to 10K where the Adam are sampled from recent actions. The optimisation algorithm of Adam in Keras implementation with a learning rate of 0.001 is used which is based on RMSProp (Wilson et al., 2017). For the experimental interval, the average steps have been counted after 1000 steps. We have used two convolutional layers of 16×3×3 and 32×3×3 with the rectified linear unit (ReLU) as an activation function. Moreover, a fully connected layer has been established on the top of the two layers. See Appendix D for the whole setup including the decay, loss and the kernel functions architecture with the number of tensors. In addition, in the beginning, the number of "do nothing" actions have been set to 40 at the start of an episode.

The following Table 7 shows the average number of steps in thousands to reach the goal state and total expected return (i.e. reward) of MPQ, MO-MCTS and MPDQN. The ± shows the average higher and lower values to reach the goal state and obtain the rewards. The average calculation has been measured based on 100 agents.

Table 7: Average number of steps and total expected return (in thousands)

Environment	Metric	MPQ	MO-MCTS	MPDQN
Dynamic DST (Silver and Gold)	#Steps	34320.5 (±253.1)	45065.2 (±174.2)	58323.0 (±223.6)
	#Return	6225.2 (±123.6)	7055.0 (±89.4)	5015.4 (±100.6)
Dynamic DST (Attack by enemy)	#Steps	76552.2 (±348.6)	40349.5 (±296.5)	68233.2 (±463.8)
	#Return	2642.5 (±229.0)	3405.6 (±74.2)	2215.7 (±152.2)
Water quality resilience	#Steps	93680.2 (±328.1)	87462.3 (±64.2)	94275.5 (±325.9)
	#Return	22151.2 (±213.5)	30255.3 (±18.6)	14695.7 (±175.8)

Table 8 shows the training steps until convergence over 100 agents.

Table 8: Average number of steps (i.e. elapsed time) and total expected return for the proposed PQDQN

Environment	Parity Q Deep Q Network (PQDQN)		
	Average Steps	Maximum Steps	Average Return
Dynamic DST (Silver and Gold)	25849.01	65882.19	9128.06
Dynamic DST (Attack by enemy)	55894.24	92586.30	4305.21
Water quality resilience	86365.89	256748.53	38130.26

From the above two tables (7 and 8), it is clear that our developed PQDQN earns highest expected rewards than MPDQN, MPQ and MO-MCTS in all the cases. However, the proposed algorithm takes reasonably higher steps compared to MO-MCTS in the Dynamic DST (attack by enemy) environment. The possible reason for this is the randomness generation by the enemy where the enemy submarine hits the agent randomly and damages its health meter.
The following Fig. 19 shows the heatmap of the traverse by the agent in 4 different algorithms. The deep red zone shows a better path to be followed according to the true Pareto front. For the simplification, we have only shown the silver one in the dynamic DST environment.

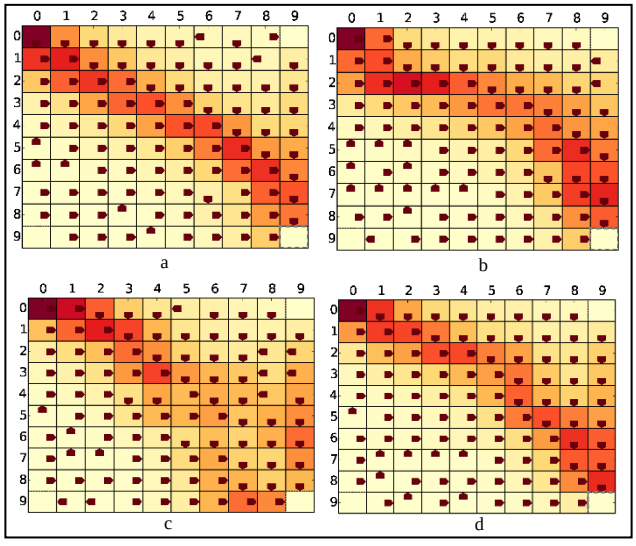


Fig. 19. Heatmap of average visited states over 100 agents for dynamic DST (silver only) (a) Parity-Q DQN (b) MO-MCTS (c) MPQ (d) MPDQN

From the above Fig. 19, it is noticeable that the average traverse of the 100 agents is more accurate for PQDQN compared to the other algorithms.
The following Fig. 20 shows the accuracy based on the known optimal solution for dynamic DST (silver and gold) over 1 million steps where PQDQN performs better than other algorithms. In this environment, MPDQN performs worse than any other algorithms.

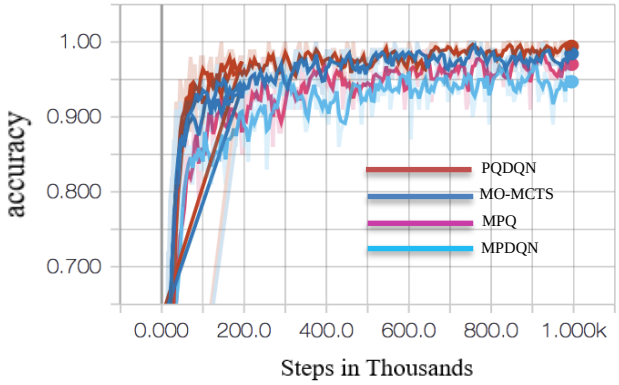


Fig. 20. Learning accuracy over 1M steps for dynamic DST (silver and gold)

The following Fig. 21 shows the mean squared error with respect to the output of the Q values from each DQN compared to the known optimal solution of the dynamic DST (attack by enemy) environment over 1Million steps where our developed algorithm performs reasonably better than MO-MCTS algorithm.

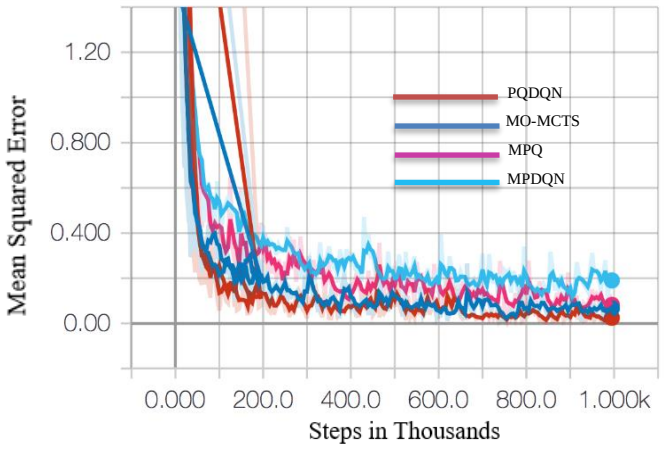


Fig. 21. Mean squared error over 1M steps for dynamic DST (attack by enemy)

Furthermore, the distribution of the weight-bias has been analysed in this research. Considering a typical ANN, each neuron is connected via a weight to another neuron. Besides, the bias unit is an extra neuron to each pre-output layer that is not connected to the previous layers. The following figures show the bias and weights distribution for the different convolutional and connected layers in the given network. The following Fig. 22 shows a bias and weight distributions with the learning rate of 1E-3 for predicting water quality resilience (see Appendix E for more details).

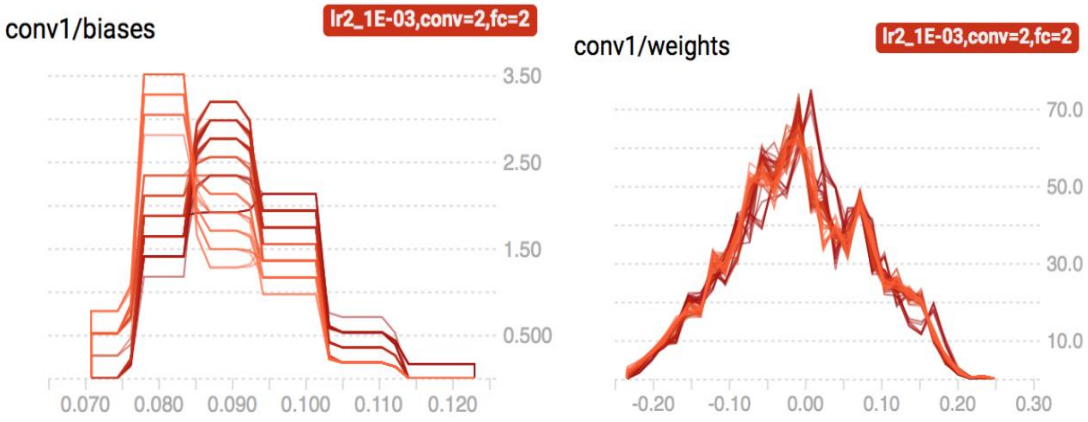
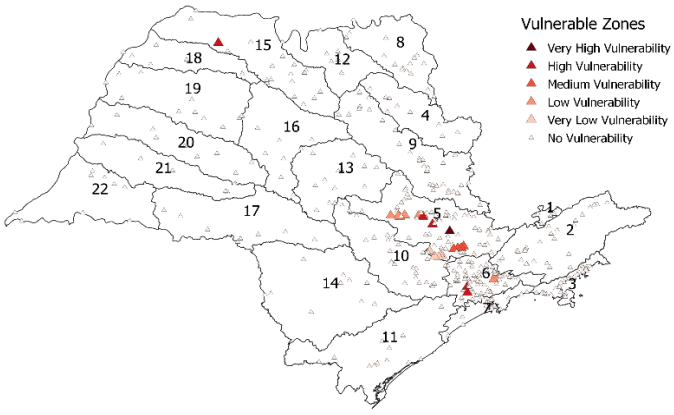


Fig. 22. Bias and weights distributions on the learning rate of 1E-03 for predicting water quality resilience

The following Fig. 23 shows the vulnerable areas identified by the developed algorithm based on IQA, IET and IVA. It is noticeable that the developed algorithm can identify all the zones which are vulnerable (see Appendix F (1, 2, and 3)).



Zones	Stations
5	Ribeirão Quilombo
	Rio Jundiaí
	Rio Piracicaba
	Rio Piraí
6	Reservatório do Guarapiranga
	Reservatório do Rio Grande
	Rio Guaió
15	Ribeirão do Marinheiro

Fig. 23. Vulnerable zones identified by the Parity-Q-Deep Q network based on IQA, IET and IVA

The following Fig. 24 shows the comparison of the efficient Pareto frontier obtained by the PQDQN with the best known Pareto frontier where the red dots show the true Pareto front while the grey dots show the achieved Pareto frontier by the developed algorithm.

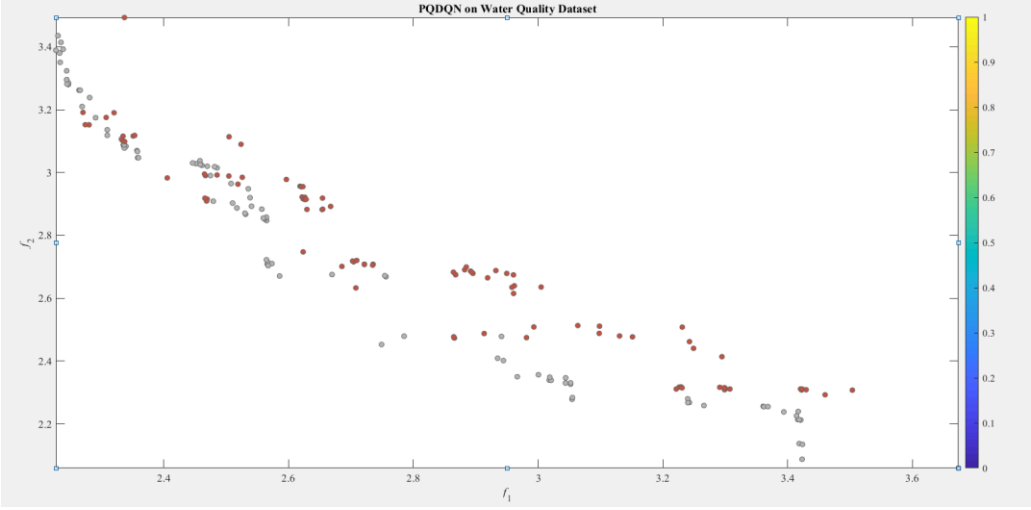


Fig. 24. Efficient frontiers; red dots: best known Pareto front and grey dots: Obtained Pareto frontier by PQDQN

6.2 Evaluation

To evaluate the performance of the proposed algorithm, three evaluation matrices such as generational distance measure (GD), inverted generational distance (IGD) and Hypervolume (HV) are considered. GD (Equation 24) measures the distance between the optimised optimal PF and the true one. Mehnen et al., (2006) have proposed to calculate the GD metric in the decision space since some DMOPs have optimal PSs that dynamically changes over time. To calculate the GD, knowledge of POF is required and a reference set. In this study, we have taken the best known POF for all our defined problems such as DST (silver and gold), DST (attack by enemy) and identifying the vulnerable zones.

$$GD = \frac{\sqrt{\sum_{i=1}^{\eta_{POF^*}} d_i^2}}{\eta_{POF^*}} \quad \text{..... (24)}$$

Where, η_{POF^*} represents the number of solutions in POF* and d_i is the Euclidean distance in the objective space between solution I of POF* and the nearest neighbour of POF*.

IGD (Equation 25) which is introduced by (Sierra & Coello, 2005) to overcome the non-adherence by GD. It is defined mathematically as follows:

$$IGD\ (PF, PF^*) = \frac{\sum_{v \in PF^*} d(v, PF)}{|PF^*|} \ldots\ldots (25)$$

HV is utilised to measure how much of the objective space is dominated by a non-dominated set. HV can be determined by Equation 26:

$$HV = volume\left(U_{i=1}^{S} v_i\right) \ldots\ldots\ldots\ldots (26)$$

Here, $U_{i=1}^{S}$ represents the union of all hypercubes of v_i in accordance with a reference point of POF where $i \in S$ (Cámara et al., 2009).

To clarify more, the distance-based performance has been measured by GD and IGD where the lower values are the better and the dominance is measured by the HV where a higher value of the HV is better.

For the dynamic DST (silver and gold), PQDQN performs better compared to the MO-MCTS, MPQ, and MPDQN with the smallest mean values for the IGD and GD matrices over 100 agents as shown in Fig. 25. In addition, PQDQN achieved the highest mean values for HV that reflects its dominance in this testbed. In this scenario, the second-best algorithm is MO-MCTS that performs better than MPQ and MPDQN in terms of all the cases except for the IGD where MPDQN performs better than MPQ and MO-MCTS. However, MPDQN performs poorly in all the measured criteria, especially compared to the proposed algorithm.

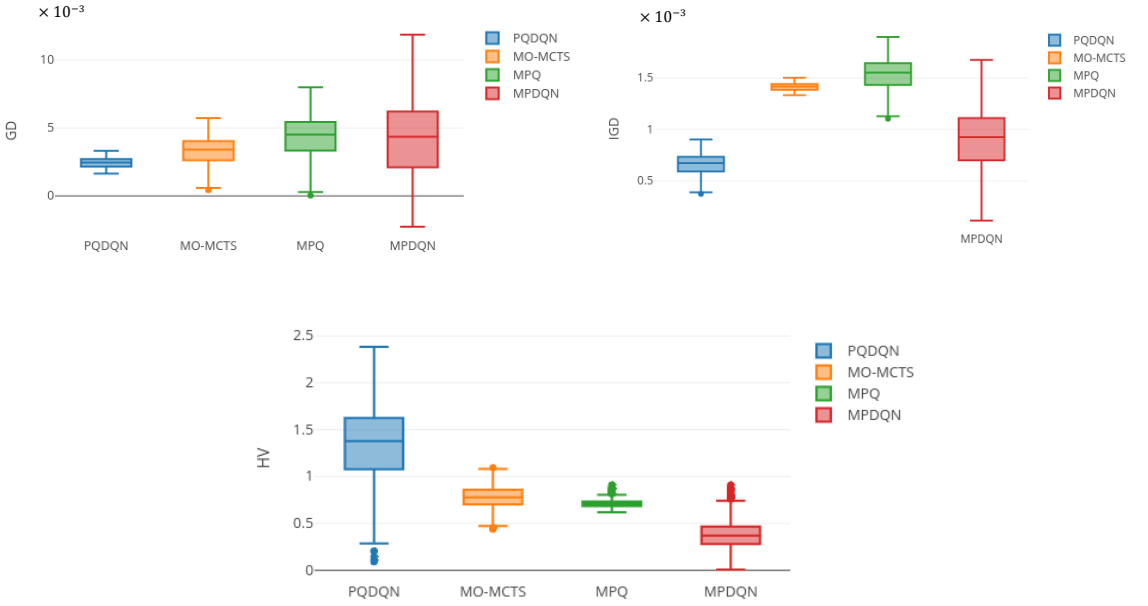


Fig. 25. Performance comparison of the algorithms in dynamic DST (silver and gold) in terms of GD, IGD and HV.

On the other hand, for the dynamic DST environment (attack by enemy), MO-MCTS performs better compared to PQDQN, MPQ and MPDQN for GD as shown in Fig. 26. However, in terms of IGD, PQDQN has achieved better result compared to all the considered algorithms. In this scenario, MPQ performs best in terms of HV and the second-best performer is MO-MCTS.

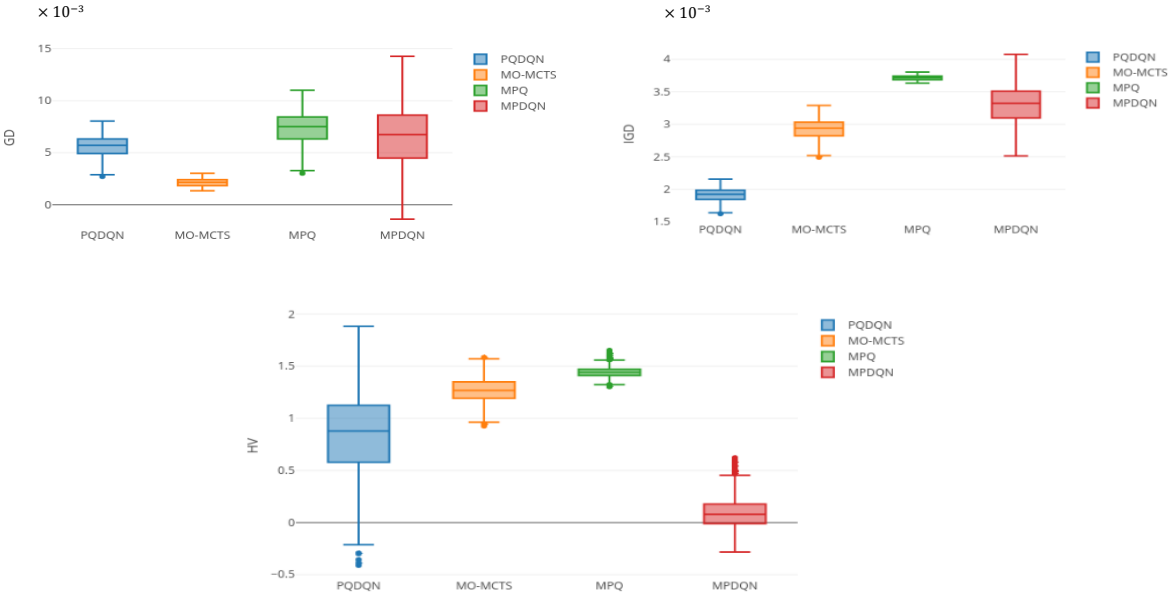


Fig. 26. Performance comparison of the algorithms in dynamic DST (attack by enemy) in terms of GD, IGD and HV.

The following Fig. 27 shows the GD, IGD and HV measure to identify the vulnerable zones based on the water quality resilience environment. In this scenario, our proposed algorithm shows superiority in all the measured criteria (i.e. GD, IGD and HV). MO-MCTS becomes the second best-performer in terms of GD and IGD.

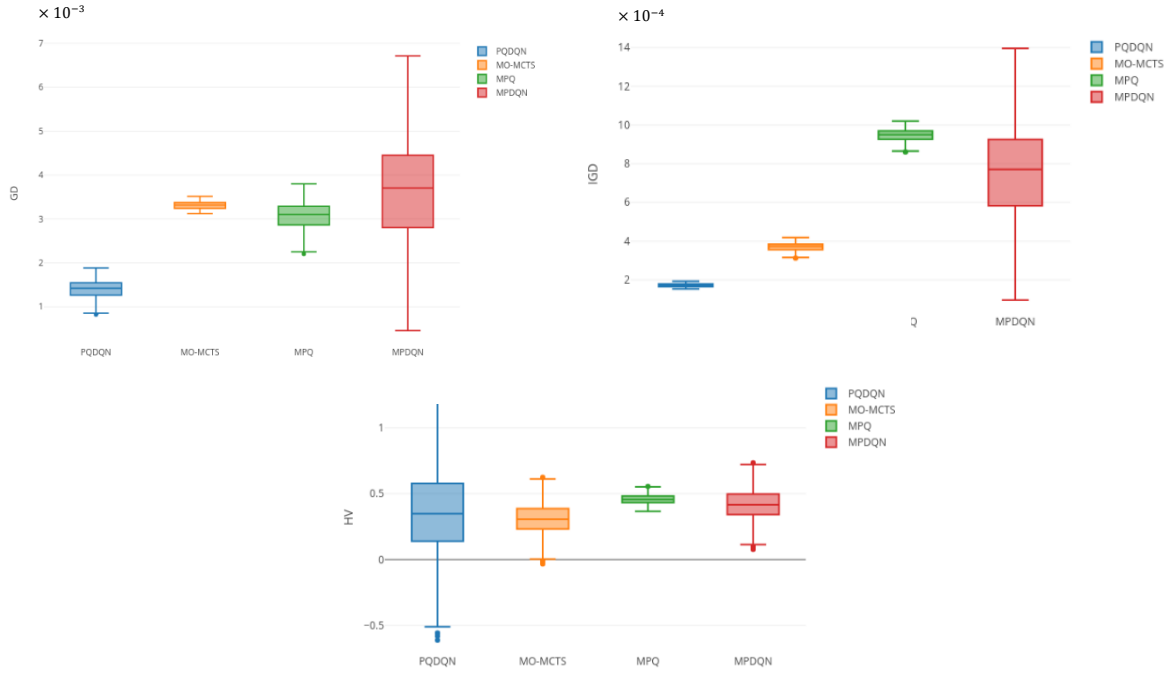


Fig. 27. Performance comparisons of the algorithms for WQR environment in terms of GD, IGD and HV

Overall, we can conclude that PQDQN performs significantly better in all the test cases except the dynamic DST (attack by enemy) environment. It is also observable that our proposed benchmark satisfies the dynamics and supports type II, III and IV. PQDQN outperforms considered algorithms in terms of convergence. The results also show that the 2nd best algorithm in this context is MO-MCTS. In addition, the result also reveals that MPDQN performs the worst in every environment.

To sum up, regarding the benchmark, the following Table 9 shows the changing PF and PS criteria of the existing DST and the proposed dynamic DSTs which eventually give an overall understanding and evidential views for defining the dynamics.

Table 9: Comparisons of the PF and PS between the existing and proposed benchmark

Testbeds	PF	PS	Objective(s)
Classic DST	Invariant	Invariant	Invariant
Dynamic DST (random)	Variant	Variant	Invariant
Dynamic DST (silver and gold)	Variant	Invariant	Invariant
Dynamic DST (attack by enemy)	Invariant	Invariant	Variant

6.3 Limitations and areas for improvement

In the context of the developing of the dynamic multi-objective optimisation benchmark, we designed based on type 2, 3, and 4. There is an opportunity to extend this work for type 1 (i.e. optimal PS changes where the optimal PF remains invariant). Overall, the benchmarks can be further modified by introducing different obstacles that may change over time. We have considered up to 3 objectives (i.e. DST attack by enemy). It is possible to increase the objectives and convert this benchmark into many objectives problem.

In addition, we haven't implemented more than one agent to identify a better policy. Therefore, multi-agent or human-agent teamwork can be integrated to make a better and on-demand resolution in real-time. In terms of integrating real-world scenarios, raw pixels of the environment can be embedded for an interesting enhancement of this work. This can be implicated in video games, stock-exchange prediction, image processing and so on.
In the context of the algorithm development, we have considered predominantly the off-policy algorithms. However, a model-based and on-policy algorithm can be explored further to investigate the applicability and efficacy of the algorithm. The scope of the parity-Q deep Q network can be enhanced in many objectives scenario. Besides, incorporating parallel computing can significantly reduce the elapsed time of being trained for the developed algorithm.

7. Conclusions

In this study, we have focused in three challenges in the domain of dynamic multi-objective optimisation in RL settings in terms of a benchmark and an efficient algorithm for dynamic DMOP and its application in the real-world scenario. We have identified the absence of a benchmark in the area of dynamic multi-objective optimisation in RL settings. We addressed that gap and created the extended version of DST that successfully satisfied the dynamics in terms of changing parameters of a defined environment over time. Our proposed generic algorithm (i.e. PQDQN) enables decomposition of problems into sub-problems and make a relation and map with different objectives to find out compromising solutions that adhere to the POF. It also provides a method for robust manipulation of priorities after the training which may ignore a specific behaviour or objective provided by DQN. It allows new objective to accommodate with the trained agent without any retraining and behaviour tuning of the agent. It also generates meta-policy (i.e. governs the policy) to identify which policy needs to be selected. In addition, we successfully deployed the developed algorithm in a real-world scenario where our agent outperforms other multi-policy algorithms such as MO-MCTS, MPQ and MPDQN. Though the agent requires higher elapsed time to be trained compared to the MO-MCTS agent in the DST (attack by enemy) environment, its accuracy in finding the Pareto optimum solutions is significantly enhanced compared to the MPDQN and MPQ. In addition to that, we were able to identify the vulnerable zones based on the water quality resilience. And, we have found that the most vulnerable zones are 5 (Ribeirão Quilombo), 6 (Reservatório do Guarapiranga) and 15 (Ribeirão do Marinheiro) in terms of water quality resilience in São Paulo, Brazil. This experiment also shows an easy implementation of deep reinforcement learning to solve a practical problem using multi-objective Markov decision process (MOMDP). The result shows that in a stochastic MOMDP setting where there is randomness, the proposed algorithm performs better than MPQ and MPDQN. Our algorithm is being tested for the two objectives scenarios (i.e. DST (silver and gold) and WQR model) and the three objectives (DST attack by enemy) scenario. In future work, we aim to test our developed algorithm in many objectives (i.e. more than 4 objectives) scenario. In addition, using deep learning by nature is CPU intensive, therefore, GPU is going to be used to enhance the matrix operations in the next phase of this research. Furthermore, as far as the dynamic DST testbed is concerned, it will provide the opportunity to the researchers with a new dimension to do their research and enable them to test their algorithms in solving everyday problems of human life that are dynamic in nature.

Acknowledgements

This research project is sponsored by the EU funded Erasmus Mundus Action 2 SmartLink project (Grant Agreement-20140858). This research has utilised the data and findings from WQR_{GIS} project (Frontiers of Engineering-SF1617\1\42, 2016-2017) that was funded by Royal Academy of Engineering in the UK and was led by the Department of Engineering and the Built Environment at Anglia Ruskin University, UK. We would like to thank the Brazilian partners in WQR_{GIS} project for their support and special thanks to helping us to translate some reports from Portuguese to English.

References

Aberdeen, D., Thiébaux, S., Zhang, L., 2004. Decision-Theoretic Military Operations Planning.

Algoul, S., Alam, M.S., Hossain, M.A., Majumder, M.A.A., 2011. Multi-objective optimal chemotherapy control model for cancer treatment. Med. Biol. Eng. Comput. 49, 51–65. https://doi.org/10.1007/s11517-010-0678-y

Amato, P., Farina, M., 2005. An ALife-Inspired Evolutionary Algorithm for Dynamic Multiobjective Optimization Problems, in: Soft Computing: Methodologies and Applications. Springer-Verlag, Berlin/Heidelberg, pp. 113–125. https://doi.org/10.1007/3-540-32400-3_9

Amitrano, D., Martino, G. Di, Iodice, A., Mitidieri, F., Papa, M.N., Riccio, D., Ruello, G., 2014. Sentinel-1 for Monitoring Reservoirs: A Performance Analysis. Remote Sens. 6, 10676–10693. https://doi.org/10.3390/rs61110676

Anschel, O., Baram, N., Shimkin, N., 2017. Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning.

Antanasijević, D., Pocajt, V., Perić-Grujić, A., Ristić, M., 2014. Modelling of dissolved oxygen in the Danube River using artificial neural networks and Monte Carlo Simulation uncertainty analysis. J. Hydrol. 519, 1895–1907. https://doi.org/10.1016/j.jhydrol.2014.10.009

Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A., 2017. Deep Reinforcement Learning: A Brief Survey. IEEE Signal Process. Mag. 34, 26–38. https://doi.org/10.1109/MSP.2017.2743240

Azzouz, R., Bechikh, S., Ben Said, L., 2014. A Multiple Reference Point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes, in: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp. 3168–3175. https://doi.org/10.1109/CEC.2014.6900569

Azzouz, R., Bechikh, S., Ben Said, L., 2015. Multi-objective Optimization with Dynamic Constraints and Objectives, in: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15. ACM Press, New York, New York, USA, pp. 615–622. https://doi.org/10.1145/2739480.2754708

Azzouz, R., Bechikh, S., Ben Said, L., 2017a. Dynamic Multi-objective Optimization Using Evolutionary Algorithms: A Survey. pp. 31–70. https://doi.org/10.1007/978-3-319-42978-6_2

Azzouz, R., Bechikh, S., Said, L. Ben, 2017b. A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population

management strategy. Soft Comput. 21, 885–906. https://doi.org/10.1007/s00500-015-1820-4

B. Bhattacharya, A.H. Lobbrecht, D.P. Solomatine, 2002. Control of water levels of regional water systems using reinforcement learning, in: 5th International Conference on Hydroinformatics, Cardiff, UK. Cardiff, UK.

Baldi, P., Sadowski, P., 2018. Learning in the machine: Recirculation is random backpropagation. Neural Networks 108, 479–494. https://doi.org/10.1016/J.NEUNET.2018.09.006

Barbosa, M.C., Alam, K., Mushtaq, S., 2016. Water policy implementation in the state of São Paulo, Brazil: Key challenges and opportunities. Environ. Sci. Policy 60, 11–18. https://doi.org/10.1016/J.ENVSCI.2016.02.017

Barrett, L., Narayanan, S., 2008. Learning all optimal policies with multiple criteria, in: Proceedings of the 25th International Conference on Machine Learning - ICML '08. ACM Press, New York, New York, USA, pp. 41–47. https://doi.org/10.1145/1390156.1390162

Bellman, R., 1958. Dynamic programming and stochastic control processes. Inf. Control 1, 228–239. https://doi.org/10.1016/S0019-9958(58)80003-0

Bertsekas, D.P., Tsitsiklis, J.N., 1996. Neuro-dynamic programming. Athena Scientific.

Bora, T.C., Mariani, V.C., Coelho, L. dos S., 2019. Multi-objective optimization of the environmental-economic dispatch with reinforcement learning based on non-dominated sorting genetic algorithm. Appl. Therm. Eng. 146, 688–700. https://doi.org/10.1016/J.APPLTHERMALENG.2018.10.020

Cámara, M., Ortega, J., de Toro, F., 2009. Performance Measures for Dynamic Multi-Objective Optimization. Springer, Berlin, Heidelberg, pp. 760–767. https://doi.org/10.1007/978-3-642-02478-8_95

Castelletti, A., Pianosi, F., Restelli, M., 2013. A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run. Water Resour. Res. 49, 3476–3486. https://doi.org/10.1002/wrcr.20295

CETESB, Brazil, 2000. . Filtr. Sep. 37, 39. https://doi.org/10.1016/S0015-1882(00)88903-X

Chatzilygeroudis, K., Rama, R., Kaushik, R., Goepp, D., Vassiliades, V., Mouret, J.-B., 2017. Black-Box Data-efficient Policy Search for Robotics.

Chen, X.Y., Chau, K.W., Busari, A.O., 2015. A comparative study of population-based optimization algorithms for downstream river flow forecasting by a hybrid neural network model. Eng. Appl. Artif. Intell. 46, 258–268. https://doi.org/10.1016/J.ENGAPPAI.2015.09.010

Cheng, R., Jin, Y., Narukawa, K., Sendhoff, B., 2015. A Multiobjective Evolutionary Algorithm Using Gaussian Process-Based Inverse Modeling. IEEE Trans. Evol. Comput. 19, 838–856. https://doi.org/10.1109/TEVC.2015.2395073

Cheng, R., Jin, Y., Olhofer, M., sendhoff, B., 2017. Test Problems for Large-Scale Multiobjective and Many-Objective Optimization. IEEE Trans. Cybern. 47, 4108–4121. https://doi.org/10.1109/TCYB.2016.2600577

Chou, J.-S., Ho, C.-C., Hoang, H.-S., 2018. Determining quality of water in reservoir using machine learning. Ecol. Inform. 44, 57–75. https://doi.org/10.1016/J.ECOINF.2018.01.005

Chunming Liu, Xin Xu, Dewen Hu, 2015. Multiobjective Reinforcement Learning: A Comprehensive Overview. IEEE Trans. Syst. Man, Cybern. Syst. 45, 385–398. https://doi.org/10.1109/TSMC.2014.2358639

Couto, C., Vicente, H., Machado, J., Abelha, A., Neves, & J., 2012. Water quality modeling using artificial intelligence-based tools. Int. J. Des. Nat. Ecodynamics 7, 300–309. https://doi.org/10.2495/DNE-V7-N3-300-309

Crites, R., Crites, R., Barto, A., 1996. Improving Elevator Performance Using Reinforcement Learning. Adv. NEURAL Inf. Process. Syst. 8 8, 1017--1023.

Deb, K., Jain, H., 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. IEEE Trans. Evol. Comput. 18, 577–601. https://doi.org/10.1109/TEVC.2013.2281535

Deb, K., Rao N., U.B., Karthik, S., 2007. Dynamic Multi-objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-thermal Power Scheduling, in: Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 803–817. https://doi.org/10.1007/978-3-540-70928-2_60

DeepMind, 2016. DeepMind AI Reduces Google Data Centre Cooling Bill by 40% | DeepMind [WWW Document]. URL https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/ (accessed 9.8.18).

Farina, M., Deb, K., Amato, P., 2004. Dynamic Multiobjective Optimization Problems: Test Cases, Approximations, and Applications. Trans. Evol. Comput. 8, 425–442. https://doi.org/10.1109/TEVC.2004.831456

Feinberg, E.A., Shwartz, A., 2014. Handbook of Markov Decision Processes - Methods and Applications 93.

French, J., Mawdsley, R., Fujiyama, T., Achuthan, K., 2017. Combining machine learning with computational hydrodynamics for prediction of tidal surge inundation at estuarine ports. Procedia IUTAM 25, 28–35. https://doi.org/10.1016/j.piutam.2017.09.005

Glavic, M., Fonteneau, R., Ernst, D., 2017. Reinforcement Learning for Electric Power System Decision and Control: Past Considerations and Perspectives. IFAC-PapersOnLine 50, 6918–6927. https://doi.org/10.1016/J.IFACOL.2017.08.1217

Gopakumar, A.M., Balachandran, P. V., Xue, D., Gubernatis, J.E., Lookman, T., 2018. Multi-objective Optimization for Materials Discovery via Adaptive Design. Sci. Rep. 8, 3738. https://doi.org/10.1038/s41598-018-21936-3

Governo Do Estado De São Paulo, 2018. CETESB - Companhia Ambiental do Estado de São Paulo [WWW Document]. URL https://cetesb.sp.gov.br/ (accessed 9.16.18).

Governo do Estado de São Paulo | Eleições 2018 [WWW Document], 2018. URL http://www.saopaulo.sp.gov.br/ (accessed 9.16.18).

Gross, E., 2016. On the Bellman's principle of optimality. Phys. A Stat. Mech. its Appl. 462, 217–221. https://doi.org/10.1016/J.PHYSA.2016.06.083

Hämäläinen, R.P., Mäntysaari, J., 2001. A dynamic interval goal programming approach to the regulation of a lake-river system. J. Multi-Criteria Decis. Anal. 10, 75–86. https://doi.org/10.1002/mcda.290

Hämäläinen, R.P., Mäntysaari, J., 2002. Dynamic multi-objective heating optimization. Eur. J. Oper. Res. 142, 1–15. https://doi.org/10.1016/S0377-2217(01)00282-X

Hasan, M.M., Abu-Hassan, K., Khin Lwin, Hossain, M.A., 2016. Reversible decision support system: Minimising cognitive dissonance in multi-criteria based complex system using fuzzy analytic hierarchy process, in: 2016 8th Computer Science and Electronic Engineering (CEEC). IEEE, pp. 210–215. https://doi.org/10.1109/CEEC.2016.7835915

He, F.S., Liu, Y., Schwing, A.G., Peng, J., 2016. Learning to Play in a Day: Faster Deep Reinforcement Learning by Optimality Tightening.

Helbig, M., Engelbrecht, A.P., 2014. Population-based metaheuristics for continuous boundary-constrained dynamic multi-objective optimisation problems. Swarm Evol. Comput. 14, 31–47. https://doi.org/10.1016/J.SWEVO.2013.08.004

Hoverman, J.T., Johnson, P.T.J., 2012. Ponds and Lakes: A Journey Through the Life Aquatic | Learn Science at Scitable. Nat. Educ. Knowl. 3(6):17.

Hutzschenreuter, A.K., Bosman, P.A.N., La Poutré, H., 2009. Evolutionary Multiobjective Optimization for Dynamic Hospital Resource Management. Springer, Berlin, Heidelberg, pp. 320–334. https://doi.org/10.1007/978-3-642-01020-0_27

Isikdogan, F., Bovik, A.C., Passalacqua, P., 2017. Surface water mapping by deep learning. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 10, 4909–4918. https://doi.org/10.1109/JSTARS.2017.2735443

Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K., London, D., 2016. Reinforcement learning with unsupervised auxiliary tasks.

Ji, X., Shang, X., Dahlgren, R.A., Zhang, M., 2017. Prediction of dissolved oxygen concentration in hypoxic river systems using support vector machine: a case study of Wen-Rui Tang River, China. Environ. Sci. Pollut. Res. 24, 16062–16076. https://doi.org/10.1007/s11356-017-9243-7

Jiang, S., Yang, S., Yao, X., Tan, K.C., Kaiser, M., 2018. Benchmark Problems for CEC2018 Competition on Dynamic Multiobjective Optimisation 1–18.

Kintsakis, A.M., Psomopoulos, F.E., Mitkas, P.A., 2019. Reinforcement Learning based scheduling in a workflow management system. Eng. Appl. Artif. Intell. 81, 94–106. https://doi.org/10.1016/J.ENGAPPAI.2019.02.013

Kober, J., Bagnell, J.A., Peters, J., 2013. Reinforcement learning in robotics: A survey. Int. J. Rob. Res. 32, 1238–1274. https://doi.org/10.1177/0278364913495721

Koo, W.T., Goh, C.K., Tan, K.C., 2010. A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment. Memetic Comput. 2, 87–110. https://doi.org/10.1007/s12293-009-0026-7

Li, H., Zhang, Q., Deng, J., 2017. Biased Multiobjective Optimization and Decomposition Algorithm. IEEE Trans. Cybern. 47, 52–66. https://doi.org/10.1109/TCYB.2015.2507366

Li, J., Liu, Z., He, C., Yue, H., Gou, S., 2017. Water shortages raised a legitimate concern over the sustainable development of the drylands of northern China: Evidence from the water stress index. Sci. Total Environ. 590–591, 739–750. https://doi.org/10.1016/j.scitotenv.2017.03.037

Li, Y., 2017. Deep Reinforcement Learning: An Overview 1–70. https://doi.org/10.1007/978-3-319-56991-8_32

Li, Z., Chen, H., Xie, Z., Chen, C., Sallam, A., 2014. Dynamic multiobjective optimization algorithm based on average distance linear prediction model. ScientificWorldJournal. 2014, 389742. https://doi.org/10.1155/2014/389742

Lima, G.N. de, Lombardo, M.A., Magaña Rueda, V.O., 2018. Data on the volumes of water stored in the reservoirs supplying the Metropolitan Area of Sao Paulo (2003–2015). Data Br. 19, 409–412. https://doi.org/10.1016/j.dib.2018.05.049

Lima, G.N. de, Lombardo, M.A., Magaña, V., 2018. Urban water supply and the changes in the precipitation patterns in the metropolitan area of São Paulo – Brazil. Appl. Geogr. 94, 223–229. https://doi.org/10.1016/J.APGEOG.2018.03.010

Lindberg, R.H., Östman, M., Olofsson, U., Grabic, R., Fick, J., 2014. Occurrence and behaviour of 105 active pharmaceutical ingredients in sewage waters of a municipal sewer collection system. Water Res. 58, 221–229. https://doi.org/10.1016/j.watres.2014.03.076

Liu, M., Lu, J., 2014. Support vector machine—an alternative to artificial neuron network for water quality forecasting in an agricultural nonpoint source polluted river? Environ. Sci. Pollut. Res. 21, 11036–11053. https://doi.org/10.1007/s11356-014-3046-x

Lizotte, D., Bowling, M., Murphy, S., 2010. Efficient reinforcement learning with multiple reward functions for randomized controlled trial analysis. Proc. 27th Int. Conf. Mach. Learn.

Lizotte, D.J., Laber, E.B., 2016. Multi-Objective Markov Decision Processes for Data-Driven Decision Support. J. Mach. Learn. Res. 17, 1–28.

Lwin, K., Qu, R., Kendall, G., 2014. A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization. Appl. Soft Comput. https://doi.org/10.1016/j.asoc.2014.08.026

Maalawi, K., 2011. Special Issues on Design Optimization of Wind Turbine Structures.

Mehnen, J., Wagner, T., Rudolph, G., 2006. Evolutionary Optimization of Dynamic Multiobjective Functions.

Meisel, S., Grimme, C., Bossek, J., Wölck, M., Rudolph, G., Trautmann, H., 2015. Evaluation of a Multi-Objective EA on Benchmark Instances for Dynamic Routing of a Vehicle. https://doi.org/10.1145/2739480.2754705

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., Hadsell, R., 2016. Learning to Navigate in Complex Environments.

MIT Technology Review, 2018. 10 Breakthrough Technologies 2017 - MIT Technology Review [WWW Document]. URL https://www.technologyreview.com/lists/technologies/2017/ (accessed 9.8.18).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep

reinforcement learning. Nature 518. https://doi.org/10.1038/nature14236
Moffaert, K. Van, Nowé, A., 2014. Multi-Objective Reinforcement Learning using Sets of Pareto Dominating Policies. J. Mach. Learn. Res. 15, 3663–3692.
Murugananthan, A., Tan, K.C., Vadakkepat, P., 2016. Solving the IEEE CEC 2015 Dynamic Benchmark Problems Using Kalman Filter Based Dynamic Multiobjective Evolutionary Algorithm. Springer, Cham, pp. 239–252. https://doi.org/10.1007/978-3-319-27000-5_20
Narasimhan, K., Yala, A., Barzilay, R., 2016. Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning.
Natarajan, S., Tadepalli, P., 2005. Dynamic Preferences in Multi-Criteria Reinforcement Learning.
Nguyen, T.T., 2018. A Multi-Objective Deep Reinforcement Learning Framework.
Nogueira, T.A.R., Abreu-Junior, C.H., Alleoni, L.R.F., He, Z., Soares, M.R., Santos Vieira, C. dos, Lessa, L.G.F., Capra, G.F., 2018. Background concentrations and quality reference values for some potentially toxic elements in soils of São Paulo State, Brazil. J. Environ. Manage. 221, 10–19. https://doi.org/10.1016/J.JENVMAN.2018.05.048
Parisi, S., Pirotta, M., Restelli, M., 2016. Multi-objective Reinforcement Learning through Continuous Pareto Manifold Approximation, Journal of Artificial Intelligence Research.
Pawara, S., 2017. Remote Monitoring of Waters Quality from Reservoirs 503–506.
Peek, N.B., 1999. Explicit temporal models for decision–theoretic planning of clinical management. Artif. Intell. Med. 15, 135–154. https://doi.org/10.1016/S0933-3657(98)00049-9
Perez, D., Mostaghim, S., Samothrakis, S., Lucas, S.M., 2015. Multiobjective Monte Carlo Tree Search for Real-Time Games. IEEE Trans. Comput. Intell. AI Games 7, 347–360. https://doi.org/10.1109/TCIAIG.2014.2345842
Preissner, S., Dunkel, M., Hoffmann, M.F., Preissner, S.C., Genov, N., Rong, W.W., Preissner, R., Seeger, K., 2012. Drug Cocktail Optimization in Chemotherapy of Cancer. PLoS One 7, e51020. https://doi.org/10.1371/journal.pone.0051020
Publicações e Relatórios | Águas Interiores [WWW Document], 2017. URL https://cetesb.sp.gov.br/aguas-interiores/publicacoes-e-relatorios/ (accessed 9.20.18).
Raquel, C., Yao, X., 2013. Dynamic Multi-objective Optimization: A Survey of the State-of-the-Art. Springer, Berlin, Heidelberg, pp. 85–106. https://doi.org/10.1007/978-3-642-38416-5_4
Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R., 2013. A survey of multi-objective sequential decision-making. J. Artif. Intell. Res. 48, 67–113. https://doi.org/10.1613/jair.3987
Ruiz-Montiel, M., Mandow, L., Pérez-de-la-Cruz, J.L., 2017. A temporal difference method for multi-objective reinforcement learning. Neurocomputing 263, 15–25. https://doi.org/10.1016/j.neucom.2016.10.100
Sarkar, D., Contal, E., Vayatis, N., Dias, F., 2015. A Machine Learning Approach to the Analysis of Wave Energy Converters, in: Volume 9: Ocean Renewable Energy. ASME, p. V009T09A019. https://doi.org/10.1115/OMAE2015-41523
Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2015. Prioritized Experience Replay.
Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003
Shen, X.-N., Yao, X., 2015. Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems. Inf. Sci. (Ny). 298, 198–224. https://doi.org/10.1016/J.INS.2014.11.036
Sierra, M.R., Coello Coello, C.A., 2005. Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ∈-Dominance. Springer, Berlin, Heidelberg, pp. 505–519. https://doi.org/10.1007/978-3-540-31880-4_35
Silver, D., 2015. Lecture 2 : Markov Decision Processes. UCL,Computer Sci. Dep. Reinf. Learn. Lect.
Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D., 2016. Mastering the game of Go with deep neural networks and tree search. https://doi.org/10.1038/nature16961
Slaughter, A.R., Hughes, D.A., Retief, D.C.H., Mantel, S.K., 2017. A management-oriented water quality model for data scarce catchments. Environ. Model. Softw. 97, 93–111. https://doi.org/10.1016/j.envsoft.2017.07.015
Su, P.-H., Gaši´c, G., Mrkši´c, N.M., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., Young, S., 2016. On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems 2431–2441.
Sutton, R., 2008. The reward hypothesis. http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html
Sutton, R.S., Barto, A.G., 2012. Reinforcement learning. Learning 3, 322. https://doi.org/10.1109/MED.2013.6608833
Szita, I., 2012. Reinforcement Learning in Games. Springer, Berlin, Heidelberg, pp. 539–577. https://doi.org/10.1007/978-3-642-27645-3_17
Tajmajer, T., 2017. Modular Multi-Objective Deep Reinforcement Learning with Decision Values.
Tian, Y., Cheng, R., Zhang, X., Jin, Y., 2017. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization.
Tomas, D., Čurlin, M., Marić, A.S., 2017. Assessing the surface water status in Pannonian ecoregion by the water quality index model. Ecol. Indic. 79, 182–190. https://doi.org/10.1016/J.ECOLIND.2017.04.033
Tozer, B., Mazzuchi, T., Sarkani, S., 2017. Many-objective stochastic path finding using reinforcement learning. Expert Syst. Appl. 72, 371–382.

https://doi.org/10.1016/J.ESWA.2016.10.045
Understanding RL: The Bellman Equations [WWW Document], 2017. . joshgreaves.com. URL https://joshgreaves.com/reinforcement-learning/understanding-rl-the-bellman-equations/ (accessed 9.8.18).
Ursem, R.K., Ursem, R.K., Krink, T., Filipic, B., 2002. A Numerical Simulator of a Crop-Producing Greenhouse.
Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., Dekker, E., 2011. Empirical evaluation methods for multiobjective reinforcement learning algorithms. Mach. Learn. 84, 51–80. https://doi.org/10.1007/s10994-010-5232-5
Vamplew, P., Issabekov, R., Dazeley, R., Foale, C., Berry, A., Moore, T., Creighton, D., 2017a. Steering approaches to Pareto-optimal multiobjective reinforcement learning. Neurocomputing 263, 26–38. https://doi.org/10.1016/j.neucom.2016.08.152
Vamplew, P., Webb, D., Zintgraf, L.M., Roijers, D.M., Dazeley, R., Issabekov, R., Dekker, E., 2017b. MORL-Glue: A Benchmark Suite for Multi-Objective Reinforcement Learning.
Van Hasselt, H., Guez, A., Silver, D., Deepmind, G., 2015. Deep Reinforcement Learning with Double Q-learning.
Wang, W., Sebag, M., 2012. Multi-objective Monte-Carlo Tree Search.
Wang, X., Zhang, F., Ding, J., 2017. Evaluation of water quality based on a machine learning algorithm and water quality index for the Ebinur Lake Watershed, China. Sci. Rep. 7, 1–18. https://doi.org/10.1038/s41598-017-12853-y
Wang, Y., Li, B., 2010. Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization. Memetic Comput. 2, 3–24. https://doi.org/10.1007/s12293-009-0012-0
Wang, Y., Zhou, J., Chen, K., Wang, Y., Liu, L., 2017. Water quality prediction method based on LSTM neural network. 2017 12th Int. Conf. Intell. Syst. Knowl. Eng. 1–5. https://doi.org/10.1109/ISKE.2017.8258814
Wang, Z., Schaul, T., Hessel, M., Com, M., Van Hasselt, H., Lanctot, M., 2016. Dueling Network Architectures for Deep Reinforcement Learning.
Watkins, Dayan, 1989. Learning from delayed rewards. PhD thesis, Cambridge Univ.
Wilson, A.C., Roelofs, R., Stern, M., Srebro, N., Recht, B., 2017. The Marginal Value of Adaptive Gradient Methods in Machine Learning.
Zheng, K., Li, H., Qiu, R.C., Gong, S., 2012. Multi-objective reinforcement learning based routing in cognitive radio networks: Walking in a random maze, in: 2012 International Conference on Computing, Networking and Communications (ICNC). IEEE, pp. 359–363. https://doi.org/10.1109/ICCNC.2012.6167444
Zhou, H.-B., Chen, T.-B., Gao, D., Zheng, G.-D., Chen, J., Pan, T.-H., Liu, H.-T., Gu, R.-Y., 2014. Simulation of water removal process and optimization of aeration strategy in sewage sludge composting. Bioresour. Technol. 171, 452–60. https://doi.org/10.1016/j.biortech.2014.07.006
Zoltán G., Zsolt K., Csaba S., 1998. Multi-criteria Reinforcement Learning. Proc. Fifteenth Int. Conf. Mach. Learn. 24–27.

Appendices:
Appendix A:

A: Treasure values and the Pareto frontiers

In this appendix, treasure distributions have been shown in Table A.1. Identified treasure values (i.e. Pareto frontier) for the dynamic DST (silver and gold) have been shown in Table A.2. Two scenarios are described in the dynamic DST (attack by enemy) environment with the identified treasure values along with the time cost.

To get more details, click the following video link: (https://www.dropbox.com/sh/joku75dhwckjhbu/AAAdPEr2lOyZjdMe0Q-fI3eTa?dl=0) which has been created while traversing the agent in different scenarios.

Table A. 1: DST testbed treasure values

		Treasure values
DST (*silver and gold*)	Silver	1,2,3,5,8,16,24,50,74,124
	Gold	100,925,1231,1442,1525,1597,1797,1829,1889,1900
DST (*attack by enemy*)	1,2,3,5,8,16,24,50,74,124	

Table A. 2: Treasure values for the Pareto Frontier (Silver and Gold)

Treasure Types	**Identified Treasures**
Silver	1, 2, 3, 5, 50, 74, 124
Gold	1597, 1797, 1829

Table A. 3: Treasure values for the Pareto Frontier (attack by enemy- scenario 1)

Identified Treasures	**Time Cost**	**Health Meter**
1	1	10
2	3	10
0	4	8
3	5	10
5	7	10
8	8	8
16	9	10
24	13	10
50	14	10
74	17	10
124	19	2

Table A. 4: Treasure values for the Pareto Frontier (attack by enemy- scenario 2)

Identified Treasures	**Time Cost**	**Health Meter**
1	1	10
2	3	10
3	5	10
5	7	10
8	8	10
16	9	10
24	13	10
50	14	8
74	17	10
124	19	8

Appendix B:

The initial phase of the WQR project has been considered of a quantitative nature to build the model. The model uses numerical input data that is gathered from the CETESB dataset for water quality in 22 zones in São Paulo, Brazil. To give an overall understanding of the procedures, the following figure B1 shows a bird's eye view of how to repeat the process where the novelty is to formalise the MOMDP for a historical dataset in RL settings. This has been discussed systematically in the MethodsX paper.

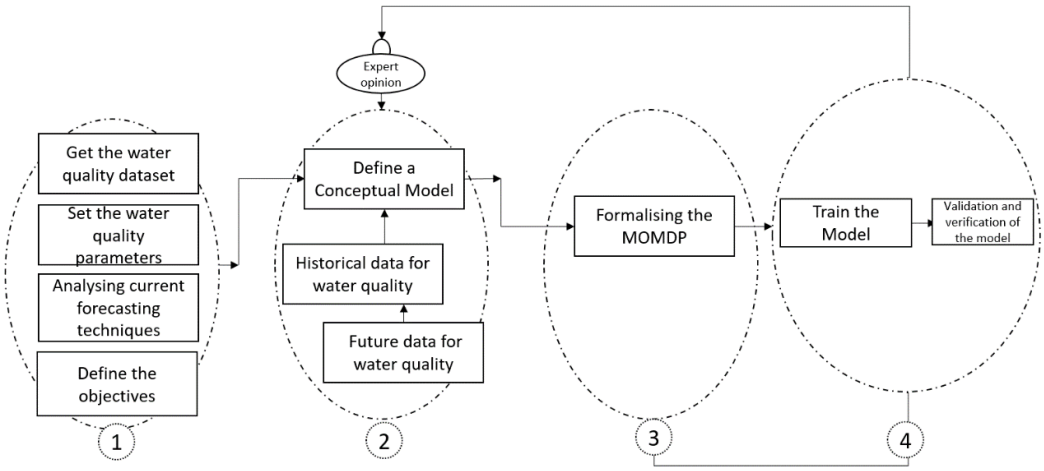


Fig. B1. Process to repeat the work in a different dataset

Similar to the first test case, we have used an identical setting of the hyperparameters for a fair comparison and the experimental setup for all of the considered algorithms. The following Table B1 shows the list of hyperparameters that has been used for the second test case to determine the vulnerable zones based on the water quality resilience in the RL settings.

Table B1: Hyperparameters for the test case 2

Parameter	**Value**
Learning steps	3 Million
Agent's Evaluation (Interval)	1 Million
Replay memory size	10000
Target network update rate	1000
Learning rate (α)	0.001
Exploration rate	0.4
ε end step	1000
Discount factor	0.9
Optimiser	Stochastic Gradient Descent (SGD)
Batch size	32
No-op max	40

Where, the network architecture has 4 convolutional layers. The first convolutional layer is devised by 32 filters of size 8x8 and a stride of 4. Also, the first layer is followed by 2 layers with 64 filters of size 4x4 and a stride of 2. The last convolutional layer comprised of 64 filters of size 3x3 and a stride of 1. These convolutional layers are followed by a 512 units of a fully connected layer.

Appendix C:

Keras Implementation for SGD:

```
sgd = SGD(lr=0.1, decay=1e-6, momentum=0.9, nesterov=True)
```

```
model.compile(loss='mean_squared_error', optimizer=sgd)
```

Keras Implementation for Adam:

```
keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False)
```

```
model.compile(loss='mean_squared_error', optimizer=adam)
```

Appendix D:

In this section, a detailed visualisation of the deep layers has been illustrated which is generated by the TensorBoard.

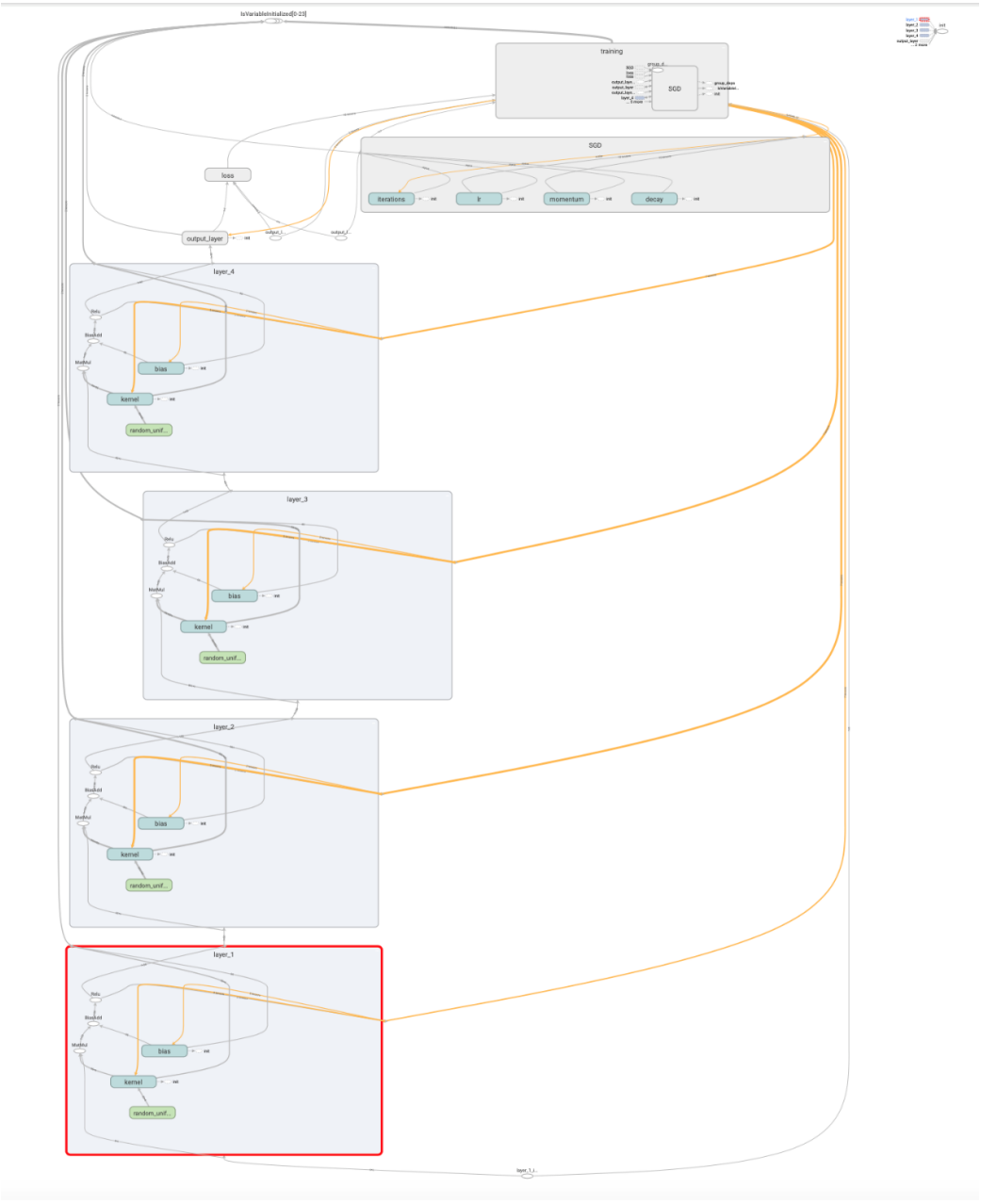


Fig D. 1. Bird's eye view of the deep layer network

Appendix E:

Fig. E. 1: Weight-bias distribution for dynamic DST (silver and gold)

Fig. E. 2: Weight-bias distribution for dynamic DST (attack by enemy)

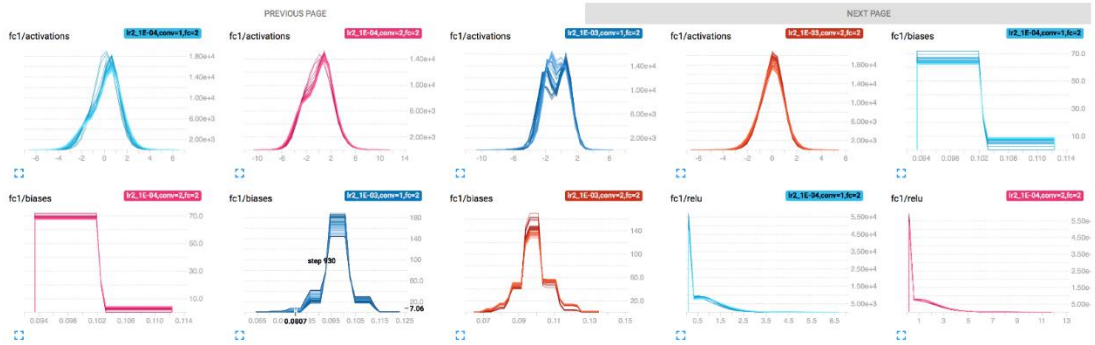


Fig. E. 3: Weight-bias distribution for WQR environment

Appendix F:

Identified vulnerable zones based on IQA, IET and IVA have been shown in Fig. F1, Fig. F.2 and Fig. F.3 respectively.

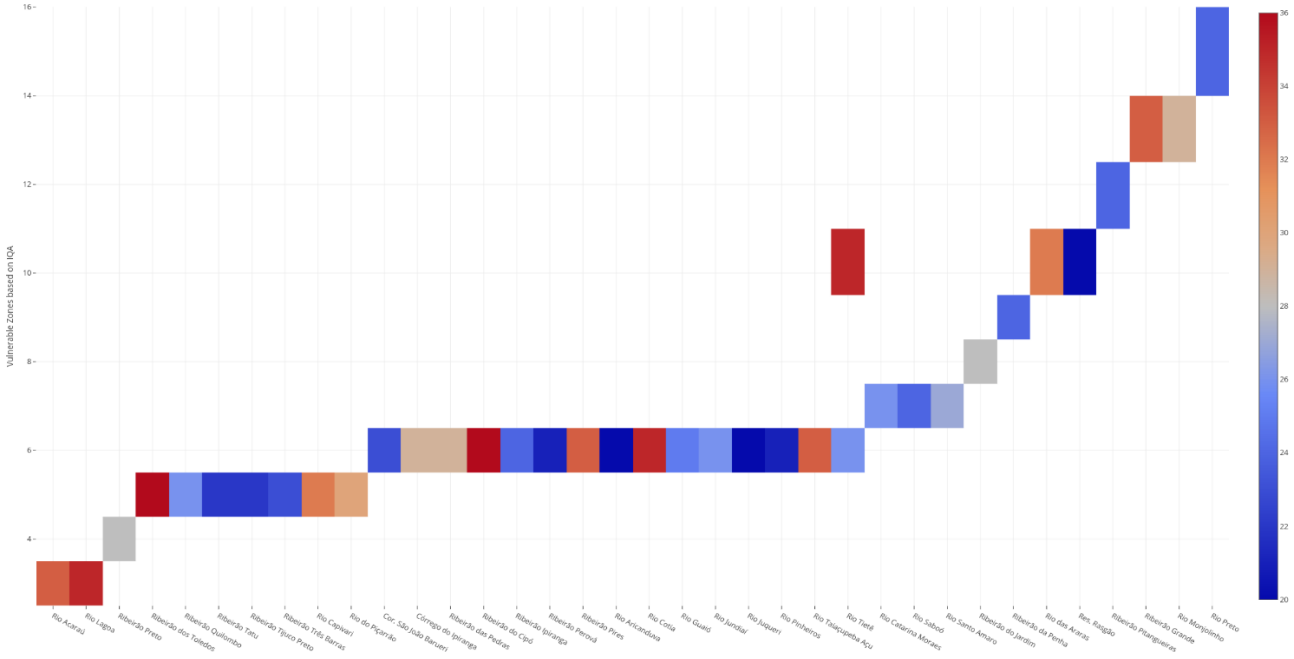


Fig. F1. Heatmap for identified vulnerable zones based on IQA

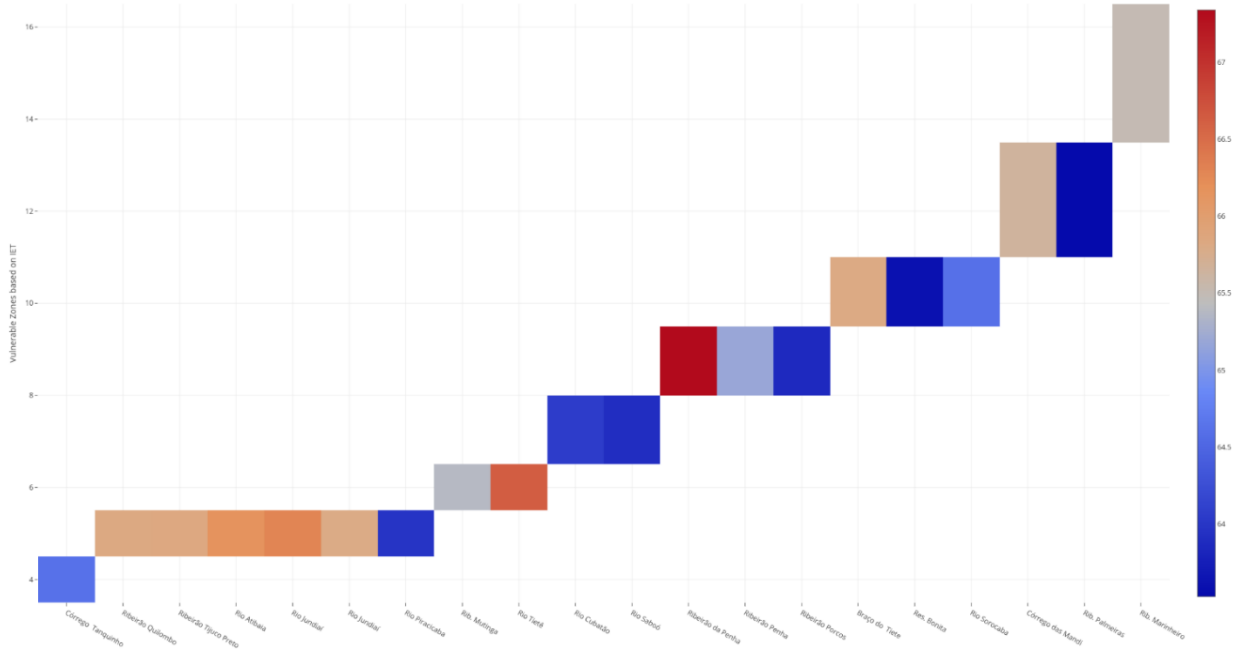


Fig. F2. Heatmap for identified vulnerable zones based on IET

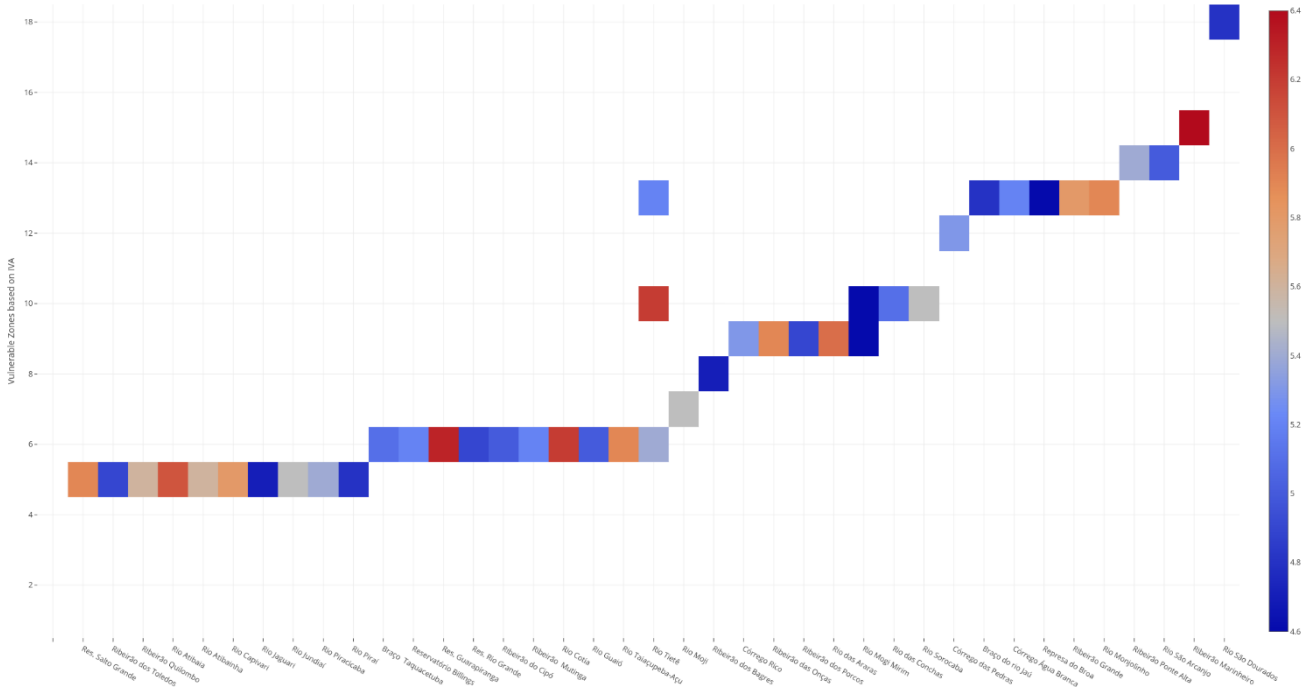


Fig. F3. Heatmap for identified vulnerable zones based on IVA