# Offline and Online Time in Sequential Decision-Making Problems


Aman Soni, Peter R. Lewis and Anikó Ekárt
Aston Lab for Intelligent Collectives Engineering (ALICE),
School of Engineering and Applied Science,
Aston University, Birmingham, B4 7ET, UK
Email: {sonia2, p.lewis, a.ekart}@aston.ac.uk



***Abstract*—A connection has recently been drawn between Dynamic Optimization Problems (DOPs) and Reinforcement Learning Problems (RLPs) where they can be seen as subsets of a broader class of Sequential Decision-Making Problems (SDMPs). SDMPs require new decisions on an ongoing basis. Typically the underlying environment changes between decisions. The SDMP view is useful as it allows the unified space to be explored. Solutions can be designed for characteristics of problem instances using algorithms from either community. Little has been done on comparing algorithm performance across these communities, particularly under real-world resource constraints.**

**In this paper we lay the theoretical foundations for the concept of offline and online time in SDMPs. We implement a method, based on the theoretical formulations, to limit offline time on representative algorithms. We investigate the online performance on a Conceptual Moving Peaks Benchmark (CMPB). Our results show that the performance of an Evolutionary Dynamic Optimisation (EDO) algorithm depends on the offline time constraint while the performance of an EDO-hybrid is noticeably impacted only past a lower bound on the size of the state-action space.**

**Our method evaluates the effects of resource constraints on online algorithm performance and is a promising start to a rigorous method of algorithm selection for real-world problems.**


## I. INTRODUCTION

Decision problems are often encountered in system design and control. If the underlying environment changes the problem is dynamic and requires new decisions over time. There is a need for robust problem-independent search algorithms in cases where knowledge of optimization function can only be gained through sampling, or if there are insufficient resources to construct a problem-dependent algorithm [1]. Current approaches to construct problem-independent algorithms are to frame the problem as a Dynamic Optimisation Problem (DOP) [2] or as a Reinforcement Learning Problem (RLP) [3].

DOPs, often tackled using Evolutionary Dynamic Optimisation (EDO) algorithms, are commonly seen as tracking moving optima problems [2]. On the other hand, Reinforcement Learning (RL) algorithms are used on RLPs, usually defined as Markov Decision Processes (MDPs) [3]. Fu et al. [4] proposed Sequential Decision-Making Problems (SDMPs) as a problem class that includes DOPs and RLPs. This perspective is useful as different algorithms specialise in different types of SDMPs [5]. The unified SDMP view encourages cross-pollination of ideas between well-established communities. Techniques inspired by both research communities include neuro-evolution [6] and learning classifier systems [7].

The first contribution of the paper is theoretical work comparing definitions of RLPs and DOPs in support of the unified problem definition space for SDMPs (Fig. 1). Dimensions in the SDMP space could include the number, range and type of environment state variables; dynamics that relate to the degree [8], rate and pattern of change [2] for the variables; and the affect of time-linkage between states [3]. Environmental dynamics could include cyclical, random or chaotic patterns. More recent benchmarks apply non-linear transformations to break symmetry and introduce irregularity on the fitness landscape [9]. Research is required to understand the overarching dimensions of the SDMP space and how algorithm performance changes under varying resource levels. This would help to select algorithms for resource-constrained problems.

The second contribution of this paper is a method to compare algorithm performance under constrained resources. We present a formalisation of the online and offline time required by algorithms used to solve SDMPs extending work by Fu et al. [4], [5]. This allows us to reason about and evaluate the impact of constrained offline time on the online performance of representative EDO and RL algorithms on different instances of the Conceptual Moving Peaks Benchmark (CMPB). This research has immediate applicability to challenging real world problems such as robot control where the computational resources available for a task is dependent on other available resources (e.g. battery life [10]).

The paper is structured as follows. Section II outlines background knowledge. Section III contains theoretical work to support SDMPs based on current problem definitions. Section IV presents a theoretical formulation for measuring the online and simulation time for algorithms. Section V presents a method for measuring the online performance of algorithms. Section VI details the experimental approach and methodology as well the results, analysis and implications for algorithm selection. Section VII concludes the paper.

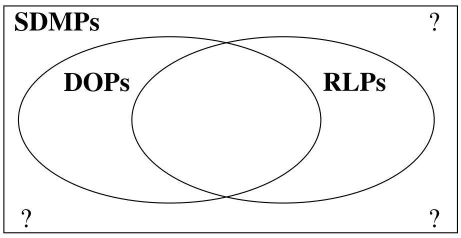


Fig. 1: Unified sequential decision-making problem definition space with gaps, and problem classes that overlap.

## II. BACKGROUND

### A. Reinforcement Learning Problems

RL is a learning approach with roots in optimal control and animal learning theory [3]. RLPs are often framed as MDPs for environments with perfect information [3] or can be defined as Partially Observable Markov Decision Processes (POMDPs) to model uncertainty [11]. While RL algorithms are considered both slow and resource intensive [12], they can learn online under dynamic conditions without a need for an *a priori* model or simulator [3]. This is a major benefit when the model is unknown or too complex to simulate accurately [13]. Additionally, RL algorithms exploit time-linkage between states. This makes convergence to an optima quicker if this information is available and there is a high impact of time-linkage [3].

The *exploration exploitation trade-off* [14] is the need to balance exploration of the environment with exploitation of previous knowledge. While some RL techniques make a distinction between offline exploration and online exploitation [15], there is no clear method to compare online performance with techniques that require offline training, or delegate learning updates. Examples of RL techniques in this class are TD-Gammon [16], a temporal-difference learning method for backgammon that trains itself using games of self-play, or state-of-the-art techniques in playing Go, which use a combination of supervised learning from human expert games, and RL from games of self-play for training, with hard time limits for online analysis and move selection in game play [17].

### B. Dynamic Optimisation Problems

Jin and Branke [18] define DOPs as a special class of optimization problems, that are "solved online over time". EDO is a sub-field of Evolutionary Computing that focusses on DOPs. A major advantage of EDO algorithms in problems with large or continuous action spaces is that the algorithms need only evolve policies that directly map states to actions, [19]. Further, EA policies need only specify an action for each state, which can be simpler to represent than learning methods, such as RL, that specify the value of each state-action pair [19] and policy representations can be evolved rather than needing to be design beforehand [12]. It may be advantageous to use EDO algorithms with uncertain state information [20]. Empirical work with population-based methods show they perform well on some DOPs [1]. This conjecture has been conformed by recent theoretical work [21]. The most common type of DOPs that current academic EDO research considers are unconstrained, non-time-linkage problems [2]. However, real-world problems are often constrained problems and time-linkage problems [22].

### C. Sequential Decision-Making Problems

Fu et al. [4] proposed a unified definition of DOPs, based on the idea of multiple-decision making from RL. They subsequently developed a new algorithm that combines the strength of EDO and RL in a Q-learning Based Evolutionary Algorithm (QBEA) [5] (outlined in Section V-B). Key assumptions are that the environmental state is observable and that there is an available computational model of the reward function. Experiments were run on two instances of the Conceptual Moving Peaks Benchmark (CMBP). The moving peaks benchmark [23] is currently one of the most widely used synthetic problems for DOPs [24]. QBEA employs an Evolutionary Algorithm (EA) to search on the reward function at each time step. The outcome of this search is exploited to speed up convergence to optimal policies. Their results show that EDO and RL algorithms are specialised in different types of DOPs [5]. Additional problems, both benchmarks and application domains, are available from a unified SDMP view. Our work extends their results, considering constraints on the resource of offline time.

### D. Comparing Algorithm Performance

Cruz et al. [24] noted in their review of DOPs that there is no unified criteria regarding what to measure or how to compare the performance of a set of algorithms over DOPs. Measures, such as generations-to-convergence or best-of-generation are used to compare population-based EDO algorithms [19]. These measures do not allow for comparison to non-population-based algorithms [24]. EDO algorithms currently ignore offline time for performance comparisons [4]. Further, it is not clear how the solution is applied to the environment [2][4].

Branke and Schmeck [25] proposed the *offline error* and *offline performance* measures for EDO algorithms. These measures are calculated for each fitness evaluation. The offline error is measured as the average over the error of the best solution found since the last change of the environment. Similar to the offline error measure, the offline performance measure calculates convergence to a previous optimal fitness, in the case where exact values of the global optima are unknown, since the last change. Disadvantages are that these measures require that the time a change occurs is known and that they are not normalised. They do not relate to online performance.

Weicker [26] considered the evaluation of three characteristics in a DOP. These are solution accuracy, stability, which measure whether changes in the environment not affect the optimization accuracy, and reactivity, as the ability of an algorithm to react quickly to changes. There is a need for the development of new measures beyond those that reduce the entire dynamic run to single-values [24] highlighting the need for new methods to compare algorithm performance.

### E. Dealing With Resource Constraints

Resources including sensors, memory, power and time are limited in the real world. There is a range of techniques to analyse algorithm time complexity [1], or build models of an algorithm's run-time [27] that can help gain an understanding of how resource constraints affect solution quality. A different approach is to allow a smaller number of iterations for small, frequent environmental changes by mapping the required iterations to a time parameter based on the frequency and degree of change in the environment rather than resource-constraints [28]. Recent work aims to reduce battery usage by reducing computation [29], or for algorithms to learn to reduce the computational resource and run-times [30].

## III. PROBLEM SETTING

Based on Fu et al. [4], we define a DOP as: *Given an optimization problem F, an optimization algorithm G to solve F, and an optimization period $[t_0, t_e]$, F is a DOP if during $[t_0, t_e]$ the underlying fitness landscape changes and G has to react by providing new solutions.* The function *E* calculates the estimated reward based on $f_t$ at each time step. When the state of the environment at time *t* is a set of variables $s_t$, and the action taken at time *t* is $a_t$, a DOP can be defined[1] as:

$$max \sum_{t=0}^{t_e} E(f_t(s_t, a_t)). \tag{1}$$

Sutton and Barto [3] define the general form of the RLP as a problem where an agent, for a sequence of discrete time steps, *t*, at each *t* evaluates its representation of the environment's state, $s_t \in \mathbb{S}$, where $\mathbb{S}$ is the set of all possible states, and selects an action to perform, $a_t \in \mathbb{A}(s_t)$, where $\mathbb{A}(s_t)$ is the set of available actions for the state $s_t$. As a consequence of the action, $a_t$, the environment moves to a new state, $s_{t+1}$ and provides a numerical reward to the agent for the next time step, $r_{t+1} \in \mathbb{R}$. The agent builds a *policy* $\pi$ for each *t*, where $\pi_t(s, a)$ is the probability that the selected action $a_t = a$ if $s_t = s$. The solution is a policy $\pi$ that maximises[2] the total sum of expected rewards. If $R$ denotes the reward function, an RLP can be defined as:

$$max \sum_{t=0}^{t_e} R(\text{arg max}[\pi_t(s_t, a_t)], s_t). \tag{2}$$

### *A Unified Problem Setting*

In both DOP (1) and RLP (2) the fitness or reward returned is determined by the effects of an action on the current state of the environment. Each problem seeks to maximise the accumulated sum of the returned value. Further, algorithm performance is determined by the values accumulated over the interval $[t_0, t_e]$. Another similarity is that, at at every at time step *t*, EDO algorithms compare individuals in a population set, while RL algorithms select from a policy-value set.

Two significant differences relate to time-linkage between states. The first is that DOP (1) accumulates the maximum reward at every $t$ while RLP (2) maximises the accumulated sum of discounted future rewards. The second difference is that the state representation in DOP (1) is a part of the problem specification while RLP (2) tracks the received rewards using state-action pairs $(s_t, a_t)$ in the learning policy $\pi$. This implies that DOP (1) does not cater for time-linkage between states, while the policy $\pi$ in RLP (2) tracks state transitions.

The similarities between the definitions of DOP (1) and RLP (2) are striking and supports the view that DOPs and RLPs are subsets of a broader class of SDMPs [4]. For some problems the dynamic of state is linked to time [2], [3], [4]. The affect of time-linkage between states could be a useful dimension of the SDMP space to help with algorithm selection.

[1]Notation has been adjusted to aid comparison to RLP (2).

[2]Maximization problems are considered without a loss of generality.

## IV. ONLINE VERSUS OFFLINE TIME

This section presents a theoretical foundation for online and offline time to enable the comparison of online performance for different algorithms. In general, we define $\tau$ as the duration of a single time step from $t_i$ to $t_{i+1}$ where $t_{i+1} = t_i + \tau$. We then define, $\tau_f$, as unit duration for *offline time* that corresponds to the time taken for a single fitness evaluation, an often-used measure in EDO [2], [24]. The environment comprises of everything that is outside of the learning method [3]. *Online time*, $\tau_n$, corresponds to the interaction of a algorithm with the environment. We define the unit duration for *online time*, $\tau_n$ as the time taken to perform an action on the environment. *Online operations* would include sensing to detect the current state or a reward value. We assume that sensing the current state and reward values can occur several times during the duration of a single $\tau_n$. For environments with a finite action set, $a \in \mathbb{A}$, at every state, $s \in \mathbb{S}$, the maximum offline time required allows the algorithm to perform fitness evaluations for each $a$ at any $t$. Then $\tau_{f_{max}} = \tau_f \times |\mathbb{A}|$, where $|\mathbb{A}|$ is the size of the set $\mathbb{A}$. We define the ratio of offline time available to $|\mathbb{A}|$ as the factor $\delta$. We assume a fixed $\delta$ for the duration of the learning episode[3]. Where $A \subseteq \mathbb{A}$ is the subset of actions that can be evaluated at any time step $t$, then $\delta$ is:

$$\delta = \frac{\tau_f \times |A|}{\tau_f \times |\mathbb{A}|} \tag{3}$$

The duration of a time step from $t_i$ to $t_{i+1}$ is defined as:

$$\tau = \tau_n + \delta \times |\mathbb{A}| \times \tau_f \tag{4}$$

If $\delta = 0$ then there is no offline time available for fitness evaluations. If $\delta = 1$ then there is sufficient time for fitness evaluations for all $\mathbb{A}$. If $\delta = 0.5$ then there is only sufficient time to perform fitness evaluations for $0.5 \times |\mathbb{A}|$. An open question would be how to allocate available resources? Fig. 2 provides a visualisation for a single time step.

Current reported experimental values for EDO and EDO-hybrid algorithms such as QBEA have only considered cases where $\delta = 1$ [2], [5]. However, in real-world problems available resources will vary. It is important that methods for algorithm selection include comparisons for algorithm performance under varying resource levels. There is a need for methods to compare how the online performance of algorithms change under varying levels of resource constraints.

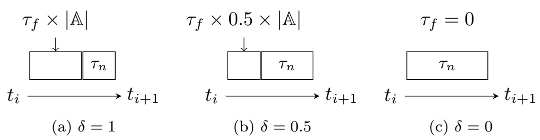


Fig. 2: The effect of $\delta$ on the number of offline fitness evaluations during a single time step from $t_i$ to $t_{i+1}$. In Fig. 2a $\tau_f$ is twice as large as $\tau_f$ in Fig. 2b.

[3]In real world applications $\delta$ may vary over time. We assume algorithms execute actions consecutively with only as single degree of parallelism.

## V. COMPARING ONLINE PERFORMANCE

This section outlines a method to compare the online performance of EDO, RL and hybrid algorithms while varying the available offline time.

### *A. Comparing EDO and RL Algorithms*

Typically, EDO algorithms assume a fitness function [4] while RL algorithms commonly assume some observability of state and a reward function [3], [4]. To compare the performance across EDO and RL algorithms we assume equivalence of the fitness and reward functions. In general, EDO algorithms assume the use an offline environment before the solution is applied to the problem instance [18]. An exception is online EDO where each fitness evaluation is applied online to the problem instance. This occurs when $\tau_f = 0$.

While many RL techniques operate directly on the environment, there are cases where prior knowledge is gained from training [31], or updates to the learning policy are delegated [13]. The resources used for training and policy updates are not clearly outlined in current literature and makes the comparison of online performance across techniques neither systematic nor rigorous.

### *B. Representative Algorithms*

Q-learning [32] (Algorithm 1) is a representative RL algorithm [3]. The internal learning policy is stored in a state-action matrix called the Q-values. An off-policy implementation [3] means that the policy update on line 7 uses the reward for the best known action at new state $s_{t+1}$ rather that the selected action performed on line 5 to improve the tracking of time-linkage between states. While several RL methods [17], [15] require offline time as defined in Section IV, Q-learning (Algorithm 1) interacts directly with the environment in an online manner with no offline time requirements.

**Algorithm 1** Representative off-policy RL algorithm

**Require:** discount factor, learning parameter
1: INITIALIZE *Q-values*
2: **for** $t = 0 \rightarrow t_e$ **do**
3:   OBSERVE state $s_t$
4:   CHOOSE action $a_t$, $a_t \in \mathbb{A}(s_t)$ from *Q-values*
5:   PERFORM action $a_t$;
6:   OBSERVE state $s_{t+1}$ and RECEIVE reward $r_t$
7:   UPDATE *Q-values*
8: **end for**

In general, EDO assumes that solutions are generated offline before being applied to the problem instance [19]. An EDO approach for dynamic environments is hyper-mutation, which increases the diversity of the population of candidate solutions after a change has been detected [2].

A general form of an EA, with state awareness at each time step, is described in the pseudo-code of Algorithm 2, denoted as EDO hereafter. This algorithm would run at each time step of the SDMP. Intuitively this means several offline computational steps are required to make a decision at every time step in comparison to Algorithm 1. We adopt an approach to restart the algorithm at every time step with the fitness function changing to reflect the new state of the environment.

**Algorithm 2** EDO algorithm

**Require:** evaluation function $f(s, a)$
1: **for** $t = 0 \rightarrow t_e$ **do**
2:   OBSERVE state $s_t$
3:   INITIALISE population randomly
4:   **for each** generations **do**
5:     EVALUATE members of population
6:     REPRODUCE members of population
7:   **end for**
8:   CHOOSE action $a_t$, $a_t \in \mathbb{A}(s_t)$ from population
9:   PERFORM action $a_t$
10: **end for**

QBEA [5], described in Algorithm 3, is a RL-EDO hybrid. QBEA uses Q-values similarly to Algorithm 1. In addition, QBEA stores state transition information so that it can estimate the next probable state. The EA on line 4 uses the estimated state to search the reward function at each decision point, updating the Q-values for each action evaluated.

**Algorithm 3** Pseudo code of QBEA

**Require:** discount factor $\lambda$, learning parameter $\alpha$, evaluation function $f(s, a)$
1: INITIALISE *Q-values*, *state-action*
2: **for** $t = 0 \rightarrow t_e$ **do**
3:   OBSERVE state $s_t$
4:   SEARCH reward function $f(s_t, a_t)$
5:   **for** each evaluated $a_i$ on $f(s_t, a_i)$ **do**
6:     UPDATE *Q-values*
7:   **end for**
8:   CHOOSE action $a_t$, $a_t \in \mathbb{A}(s_t)$ from *Q-values*
9:   PERFORM action $a_t$
10:   OBSERVE state $s_{t+1}$ and RECEIVE reward $r_t$
11:   UPDATE *Q-values*
12:   UPDATE *state-action*
13: **end for**

### *C. A Method to Constrain Resources*

The EDO (Algorithm 2) was implemented using both Random Local Search (RLS) and a (1+1)-EA [33]. RLS generates a single generation, with the population size limited to the number of fitness evaluations afforded by the offline time constraint. There are no duplicates in a generation. The (1+1)-EA creates a single individual and evolves the individual for a number of generations, limited by the number of offline fitness evaluations. The EDO algorithm requires additional memory resources to ensure unique candidate solutions.

QBEA (Algorithm 3) uses an EA to search on the reward function. The offline constraint was implemented by limiting the size of the set of actions, $A \in \mathbb{A}$, evaluated for each $t$ on Line 4. The EA implemented for QBEA operates in the same manner as the EA for EDO (Algorithm 2).

## VI. EXPERIMENTAL CASE STUDY

In this section, we use the two instances of the CMPB [5] to study the change in online performance of representative EDO, RL and EDO-RL hybrid algorithms under varying offline time constraints. To explore the impact of constrained resources with different sized state spaces, we define a new cyclical environment dynamic for the two current time-linkage instances of the CMPB. In this paper we consider only environmental dynamics where the environment returns to previous states.

### *A. Benchmark Instances of the CMPB*

For all benchmark instances, the reward function at time step $t$ is:

$$f_t(s_t, a_t) = 30 - 2|a_t - c_t| + b_t, \tag{5}$$

where $s_t$ represents the state at time step $t$: $s_t = (c_t, b_t)$. $a_t$ belongs to the interval $[-10, 10]$ and represents the decision at time step $t$. The dynamic of the bias $b_t$ is as follows ($b_0 = \theta_b$):

$$b_t = \begin{cases} \theta_b & \text{if } a_{t-1} \geq 0, \\ -\theta_b & \text{otherwise}, \end{cases} \tag{6}$$

where $\theta_b$ is a parameter, which controls the influence of $a_{t-1}$ on $b_t$, with larger values being more influential on the effect of time-linkage. In the first benchmark instance, $\theta_b$ is set to 100, and in the second benchmark instance, $\theta_b$ is set to 15.

In the CMPB [5], the dynamic of the environment variable $c_t$ is ($c_0 = 5$):

$$c_t = c_{t-1} \times -1 \tag{7}$$

Therefore $c_t$ oscillates between $c_0$ and $-c_0$ and the environment returns only two states. We denote the benchmark that uses Equation 7 as CMPB-O.

To explore the impact of constrained resources with difference state space sizes, we define a cyclical dynamic for the environment variable $c_t$. We denote the benchmark with this dynamic as CMPB-C. Where $\theta_b$ is set to 100 in the first instance, and $\theta_b$ is set to 15 in the second instance. Where the set $\mathbb{S}$ represents all possible environment states, the dynamic of the environment variable $c_t$ in Equation 7 is replaced as follows ($c_0 = -10$):

$$c_t = c_0 + t \mod |\mathbb{S}| \tag{8}$$

Where $c_t$ now belongs the to interval $[-10, 10]$, there are $|\mathbb{S}|$ states encountered in the CMPB-C. This increases the size of the state-action space, requiring additional exploration, compared to the CMPB-O instances, where the only states encountered are $\{c_0, -c_0\} \subseteq (S)$.

### *B. Representative Algorithms*

Fu et al. [5] use an ideal EDO method and Q-learning for comparative results with QBEA. We compare EDO (Algorithm 2), using both random search and a (1+1)-EA [33], Q-learning (Algorithm 1), QBEA (Algorithm 3) and a random method that performs action selection with no evaluation.

### *C. Experimental Results and Analysis*

The performance of EDO, Q-learning, and QBEA, together with the optimal accumulated rewards and a random method, on the first and second benchmark instances of CMPB-O and CMPB-C with $\delta$ increasing in the range $[0, 1]$ are presented in Figs. 3, 4, 5 and 6 respectively.

For all benchmark instances, experiments were averaged over 100 runs for 1000 time steps. We generated a set of seeds. The same seeds were then used for repeated runs. The standard error is shown by the error bars. Where there is no variance of the standard error, the error bar is shown only once.

*1) General Observations:* The accumulated rewards for neither Q-learning nor the random algorithm are effected by $\delta$ in any benchmark instance. This is expected as there are no offline operations defined for the algorithms.

For $\delta = 1$, the results are in line with Fu et al. [5]. Q-learning achieves better performance than EDO on the first benchmark instances where time-linkage has a higher effect. EDO outperforms Q-learning on the second benchmark instances. QBEA consistently outperforms on all instances.

EDO tracks the performance of the random algorithm for $\delta = 0$. EDO accumulates higher rewards as $\delta \to 1$ and the standard error reduces as $\delta \to 1$. For $\delta = 1$, EDO performs exhaustive search and, as expected, the standard error is 0.

*2) Impact of Constraints on Time:* EDO specialises in the second benchmark instances (Figs. 4 & 6), where the bias reduces the effects of time-linkage between states. EDO outperforms Q-learning on the second benchmark instance only where there is sufficient offline time, which occurs when $\delta \approx 0.23$ for the CMPC-O (Fig. 4) and $\delta \approx 0.11$ for the CMPC-C (Fig. 6).

On the CMPB-O (Figs. 3 & 4) the affect of $\delta$ on the performance of QBEA was unexpected. There is no clear change in accumulated rewards with constrained offline time and increasing values of $\delta$ do not predict higher accumulated rewards. However, the impact of $\delta$ on QBEA can be seen on both instances of the CMPB-C (Figs. 5 & 6) where there is a clear increase in accumulated rewards as $\delta \to 1$. These results show that the performance of QBEA is effected by $\delta$, provided that there is a minimum bound on the required exploration. Further investigation of this is shown in Section VI-C4. Of interest, the QBEA standard error decreases as $\delta \to 1$ however, the standard error does not equal 0 for $\delta = 1$, unlike EDO.

*3) Impact of State Space Size:* The performance of EDO increases slightly in both benchmark instances of CMPB-C (Figs. 5 & 6) as $\delta \to 1$, while the performance of Q-learning falls significantly in comparison to the corresponding CMPC-O instances (Figs. 3 & 4). This suggests that changes in the size of state-space may have a greater impact on RL algorithms than EDO, or that EDO copes better than RL with a wider range of state dynamics.

QBEA clearly outperforms the other techniques when the size of state space is increased. This is particularly marked when compared to Q-learning on the first benchmarks instances (Figs. 3 & 5). The increase in the exploration required in CMPB-C, compared to CMPB-O, before Q-values tend to being fully

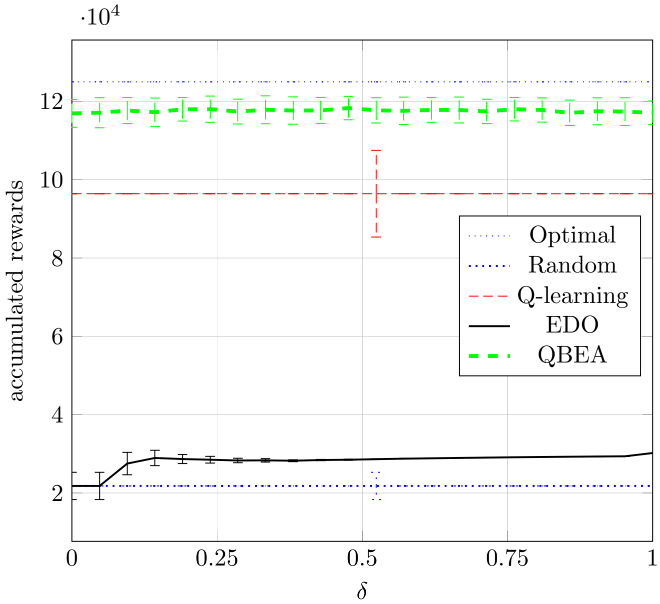


Fig. 3: The averaged accumulated rewards in Equation 5 over 100 runs with $\theta_b = 100$ with $\delta$ in range $[0, 1]$ on the CMPB-O. Environment dynamics oscillate (Equation 7).

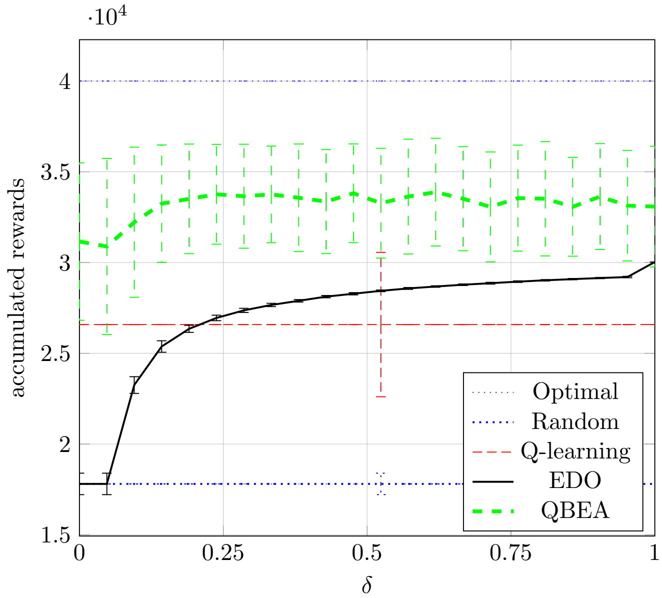


Fig. 4: The averaged accumulated rewards in Equation 5 over 100 runs with $\theta_b = 15$ with $\delta$ in range $[0, 1]$ on the CMPB-O. Environment dynamics oscillate (Equation 7).

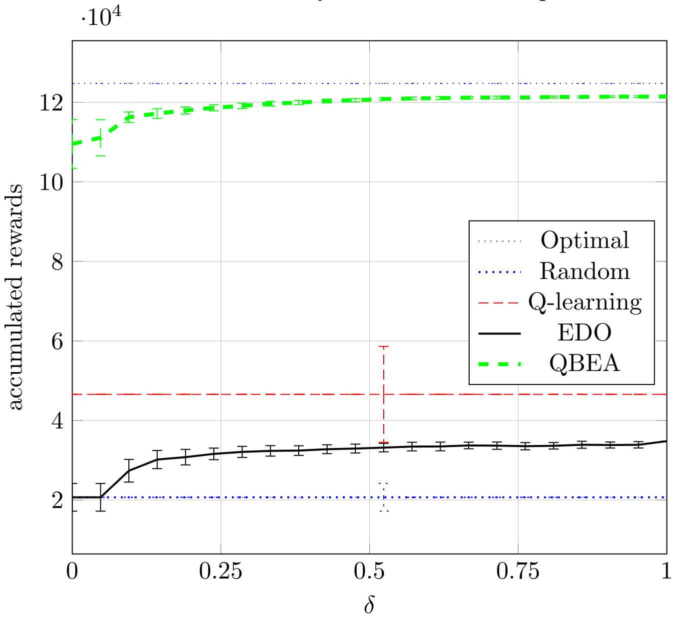


Fig. 5: The averaged accumulated rewards in Equation 5 for 1000 times steps over 100 runs with $\theta_b = 100$ and $\delta$ in range $[0, 1]$ on the CMPB-C. Environment dynamics cycle through $\mathbb{S}$ (Equation 8).

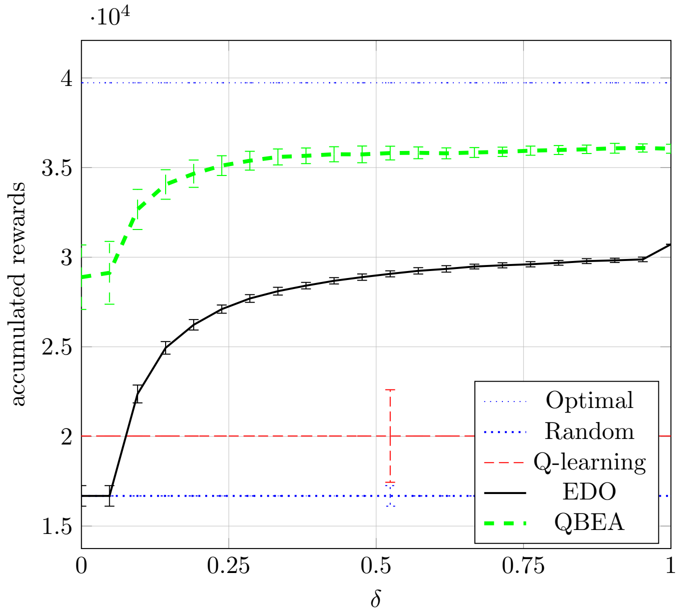


Fig. 6: The averaged accumulated rewards in Equation 5 for 1000 times steps over 100 runs with $\theta_b = 15$ and $\delta$ in range $[0, 1]$ on the CMPB-C. Environment dynamics cycle through $\mathbb{S}$ (Equation 8).

mapped means there is further exploration required before QBEA convergences to optimal policies. Investigation of the algorithm operations shows that dependant on the size of the state space, even with the random exploration strategy used when $\delta = 0$, QBEA converges to an optimal policies quicker than the $\epsilon$-greedy exploration strategy used by Q-learning.

*4) Estimating mean exploration time:* QBEA's Q-values table for the CMPB is a matrix with dimensions that map each state $s \in \mathbb{S}$ to each action $a \in \mathbb{A}$. The required exploration for each state $s$ is a combination of the available actions $\mathbb{A}$. For $\delta = 0$, the action selection is random. To determine the mean number of time steps required explore the environment this can be treat this as the "coupon collector's problem" [1]. This problem asks, *"Given n coupons, how many coupons do you expect you need to draw with replacement before having drawn*

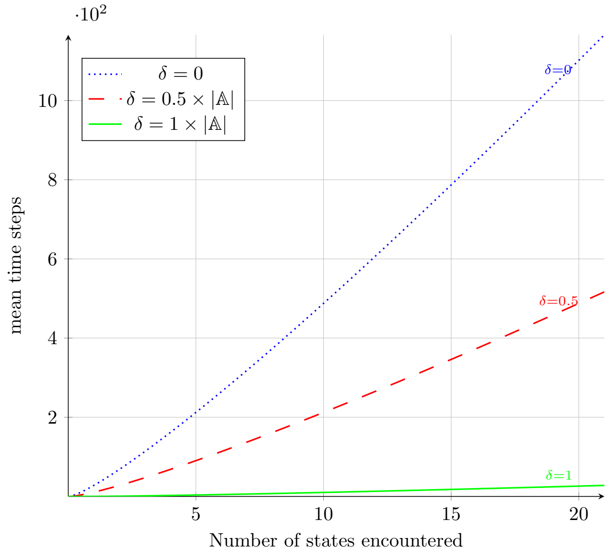


Fig. 7: The mean number of time steps required for exploration of $|\mathbb{A}|$ over the number of states encountered by QBEA as shown in Equation 9.

*each coupon at least once?"*. The mean number of expected time steps $N(t)$ to fully explore the encountered states has been shown to for large $|\mathbb{A}| \times |\mathbb{S}|$ to approach [34]:

$$N(t) = |\mathbb{A}| \times |\mathbb{S}| \log |\mathbb{A}| \times |\mathbb{S}| \tag{9}$$

On the CMPB benchmarks for every $t$ $a \in \mathbb{A}$, and so the set of actions is equal for each state. Fig. 7 plots the mean time steps required for mapping the CMPB against the number of states encountered for three different values of $\delta$. For $\delta = 0$, $|\mathbb{S}| = 2$, the mean time steps required to explore the state action space is $t \approx 250$. For $\delta = 0$, $|\mathbb{S}| = 21$, the mean time steps required to explore the state action space is $t \approx 11.5 \times 10^2$.

### D. Implications for Algorithm Selection

The ratio of offline to online time $\delta$ has been defined in Section IV. Fu et al. [5] show that EDO and RL techniques specialise in different types of SDMPs, where the SDMP type is determined by the influence of time-linkage between time steps. The results from Section VI-C support the findings from Fu et al. [5] where the hybrid QBEA technique outperforms both EDO and RL techniques.

The method from Section V brought an understanding of the impact on algorithm performance for real-world problems with constrained resources. Our results show the impact of a constraint on offline time for EDO and EDO hybrid algorithms.

For SDMPs where there is little impact from time-linkage between states, EDO outperforms Q-learning, provided there is sufficient offline time. With an increase in the state-action space, the relative performance of EDO to Q-learning, for SMPDs where there is a higher impact from time-linkage between states, improves provided there is sufficient offline time.

For $\delta = 0$, the quality of the solutions for EDO match a random method. In this situation, Q-learning may produce better quality solutions than EDO, even for types of SDMPS where EDO previously outperformed. Offline time has an upper bound of $\delta = 1$, which provides sufficient offline time to evaluate all possible actions. To conserve run-time resources, algorithms can limit offline time to $|\mathbb{A}| \times |\mathbb{S}|$.

QBEA outperforms other algorithms for all benchmark instances. Under constrained offline time, random exploration provided an effective, computationally inexpensive exploration for QBEA. This may be an effective strategy for QBEA on new SDMPs with constrained resources. However, further investigation is required to determine when to change exploration strategies or expend additional resource.

For problems that require an always available solution both QBEA and EDO would meet this requirement. However, while QBEA it will initially provide random solutions it can leverage off any available offline time to improve the learning policy between decision points. EDO is particularly suited to SDMPs where solutions found previously are no longer optimal due to rapid or large changes in the environment.

## VII. Conclusions

In real-world sequential decision-making problems, resources are limited. There is a need for methods to model constrained resources to understand how algorithm performance could change under varying resources for real-world implementations. This paper presents a method to model constrained resources based on the proposed theoretical foundations for offline and online time for SDMPs.

Results on a conceptual moving peaks benchmark show that an EDO algorithm provided lower quality solutions with limited offline time. On the other hand, the relative performance of an RL algorithm reduced by a greater degree when the size of the state-action space increased. QBEA, an EDO-RL hybrid, outperforms the other algorithms and copes well with changes in the dynamics of environment state. The relative QBEA solution quality is affected with limited offline time only after the size of the state-action space reaches a minimum bound.

Our results show that the relationship of time-linkage between states, the size of the state-action space and run-time resource constraints affect the online performance of algorithms. When these characteristics are specified for problem instances they can be used to help with algorithm selection. Developing a further understanding of how these constraints affect solution quality could help to design or select more efficient algorithms, particularly when specific resources need to be conserved.

One important consideration is that the paper only considers environmental dynamics where the environment returns to previous states. Q-learning was designed to learn the difference between states. In the presence of random or chaotic dynamics, or in the case of a large or continuous state-space, EDO may be more suitable. Useful extensions to the benchmark would be to include a wider range of varied sized state-action spaces, as well as state dynamics, to test relative algorithm performance under different SDMP dimensions.

There are two key areas of interest for future work. First, problem instances that are exceptions to the assumption of equality between fitness and reward. And second, algorithm performance under uncertainty. Exceptions can be investigated by changing the reward estimation function ($E$ in eq. 1). The problem definitions (eq. 1 and 2) can simulate noise in sensing by adding a Gaussian probability density function to the values of $s_t$.

## ACKNOWLEDGMENT

The authors would like to thank Haobo Fu for his advice on replicating the experimental results from Fu et al. [5] and Luca Rossi for directions to the coupon collectors problem.

## REFERENCES


[1] P. S. Oliveto, J. He, and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 281–293, 2007.

[2] T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[4] H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao, "What are dynamic optimization problems?," in *Evolutionary Computation 2014. CEC14. IEEE Congress on*, pp. 1550–1557, IEEE, 2014.

[5] H. Fu, P. R. Lewis, and X. Yao, "A Q-learning Based Evolutionary Algorithm for Sequential Decision Making Problems," in *Workshop "In search of Synergies between Reinforcement Learning and Evolutionary Computation" at Parallel Problem Solving from Nature (PPSN)*, VUB AI Lab, 2014.

[6] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[7] J. H. Holland, L. B. Booker, M. Colombetti, M. Dorigo, D. E. Goldberg, S. Forrest, R. L. Riolo, R. E. Smith, P. L. Lanzi, W. Stolzmann, *et al.*, "What is a learning classifier system?," in *Learning Classifier Systems*, pp. 3–32, Springer, 2000.

[8] C. J. Uzor, M. Gongora, S. Coupland, and B. N. Passow, "Real-world dynamic optimization using an adaptive-mutation compact genetic algorithm," in *Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014 IEEE Symposium on*, pp. 17–23, IEEE, 2014.

[9] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China, "Benchmark functions for the cec 2013 special session and competition on large-scale global optimization," *gene*, vol. 7, no. 33, p. 8, 2013.

[10] A. Ravankar, A. A. Ravankar, Y. Kobayashi, L. Jixin, T. Emaru, and Y. Hoshino, "A novel vision based adaptive transmission power control algorithm for energy efficiency in wireless sensor networks employing mobile robots," in *2015 Seventh International Conference on Ubiquitous and Future Networks*, pp. 300–305, IEEE, 2015.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1, pp. 99–134, 1998.

[12] M. M. Drugan, "Synergies between evolutionary algorithms and reinforcement learning," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO Companion '15, pp. 723–740, ACM, 2015.

[13] M. Wiering and M. van Otterlo, *Reinforcement Learning: State-of-the-art*, vol. 12. Springer Science & Business Media, 2012.

[14] M. P. Deisenroth, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1, pp. 1–142, 2011.

[15] M. Castronovo, D. Ernst, A. Couëtoux, and R. Fonteneau, "Benchmarking for bayesian reinforcement learning," *PLoS ONE*, vol. 11, pp. 1–37, 2016.

[16] G. Tesauro, "Td-gammon: A self-teaching backgammon program," in *Applications of Neural Networks*, pp. 267–285, Springer, 1995.

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[18] Y. J. Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 303–317, 2005.

[19] A. E. Eiben and M. Schoenauer, "Evolutionary computing," *Information Processing Letters*, vol. 82, no. 1, pp. 1–6, 2002.

[20] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*, vol. 53. Springer.

[21] D.-C. Dang, T. Jansen, and P. K. Lehre, "Populations can be essential in tracking dynamic optima," *arXiv preprint arXiv:1607.03317*, 2016.

[22] T. T. Nguyen, *Continuous dynamic optimisation using evolutionary algorithms*. PhD thesis, University of Birmingham, 2011.

[23] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Evolutionary Computation 1999. CEC99. IEEE Congress on*, pp. 1875–1882, IEEE, 1999.

[24] C. Cruz, J. R. González, and D. A. Pelta, "Optimization in dynamic environments: a survey on problems, methods and measures," *Soft Computing*, vol. 15, no. 7, pp. 1427–1448, 2011.

[25] J. Branke and H. Schmeck, "Designing evolutionary algorithms for dynamic optimization problems," in *Advances in evolutionary computing*, pp. 239–262, Springer, 2003.

[26] K. Weicker, "Performance measures for dynamic environments," in *International Conference on Parallel Problem Solving from Nature*, pp. 64–73, Springer, 2002.

[27] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4197–4201, Jan. 2015.

[28] K. Deb, U. B. Rao N., and S. Karthik, "Dynamic Multi-Objective Optimization and Decision-Making Using Modified NSGA-II: A Case Study on Hydro-Thermal Power Scheduling," *Evolutionary Multi-Criterion Optimization*, pp. 803–817, 2007.

[29] N. Piatkowski, S. Lee, and K. Morik, "Integer undirected graphical models for resource-constrained systems," *Neurocomputing*, vol. 173, pp. 9–23, 2016.

[30] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[31] R. Dearden, N. Friedman, and D. Andre, "Model based bayesian exploration," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 150–159, Morgan Kaufmann Publishers Inc., 1999.

[32] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[33] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+ 1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, no. 1, pp. 51–81, 2002.

[34] G. Blom, L. Holst, and D. Sandell, *Problems and Snapshots from the World of Probability*. Springer Science & Business Media, 2012.