What Are Dynamic Optimization Problems?

Haobo Fu, Peter R. Lewis, Bernhard Sendhoff, Ke Tang, and Xin Yao

***Abstract*— Dynamic Optimization Problems (DOPs) have been widely studied using Evolutionary Algorithms (EAs). Yet, a clear and rigorous definition of DOPs is lacking in the Evolutionary Dynamic Optimization (EDO) community. In this paper, we propose a unified definition of DOPs based on the idea of multiple-decision-making discussed in the Reinforcement Learning (RL) community. We draw a connection between EDO and RL by arguing that both of them are studying DOPs according to our definition of DOPs. We point out that existing EDO or RL research has been mainly focused on some types of DOPs. A conceptualized benchmark problem, which is aimed at the systematic study of various DOPs, is then developed. Some interesting experimental studies on the benchmark reveal that EDO and RL methods are specialized in certain types of DOPs and more importantly new algorithms for DOPs can be developed by combining the strength of both EDO and RL methods.**

I. Introduction

OPTIMIZATION has been long studied using Evolutionary Algorithms (EAs). Generally speaking, optimization problems can be divided into two categories. One is Static Optimization Problems (SOPs), and the other is Dynamic Optimization Problems (DOPs). Without any ambiguity, a SOP can be defined as:

Definition 1.1: Given a fitness function f, which is a mapping from some set $\mathcal{A}$, i.e., a solution space, to the real numbers $\mathbb{R}$: $\mathcal{A} \rightarrow \mathbb{R}$, a SOP is to find a solution[1], i.e., making a decision, $\mathbf{x}^*$ in $\mathcal{A}$ such that for all $\mathbf{x} \in \mathcal{A}$, $f(\mathbf{x}^*) \geq f(\mathbf{x})$.

We can think of a SOP as a one-decision-making problem, where only one solution, i.e., one decision, is determined for a SOP. In contrast, there has not been a consensus about the definition of DOPs studied in the Evolutionary Dynamic Optimization (EDO) community.

Historically, many definitions about DOPs have been proposed in EDO. Applying genetic algorithms to DOPs was initially studied by Goldberg and Smith [9]. In [9], no explicit definition of DOPs was provided, and yet the authors studied the performance (best-of-generation) of a genetic algorithm with dominance and diploidy on a dynamic 0-1 knapsack problem. Similarly, Branke studied the performance (off-line performance) of memory-based EAs on the moving peaks benchmark [4], without explicitly defining DOPs.

In some other works [22][1][17], DOPs were simply defined as a sequence of SOPs over time, in which the goal for each SOP is to find a solution maximizing the fitness function of that SOP. Hence, the performance of algorithms for such DOPs was measured by the average performance on each SOP during a considered time interval.

Another type of definition of DOPs can be found in [3], in which DOPs were considered as problems of maximizing an integration of a Dynamic Fitness Function (DFF) over a time period by determining a solution at each time point. Nguyen [14] also defined DOPs as maximizing such an integral, but with a more flexible definition framework than [3] in terms of quantifying how the DFF changes, how previously determined solutions influence the dynamic, etc.

There also exist some descriptions about what DOPs should look like. Examples are "optimization problems are considered dynamic only if the EA has to cope with the dynamic" by Branke [5], "If any of those uncertainties related to optimization problems are to be taken into account, we call the problem dynamic" by Jin et al [10], "DOPs are optimization problems which must be solved as time goes by" by Bosman [3], and "DOPs are a special class of dynamic problems which are solved on-line by an optimization algorithm as time goes by" by Nguyen et al [15].

The aforementioned four types of definitions of DOPs are limited in different ways. The first type does not define DOPs explicitly but considers DOPs as problems of adapting a solution to a changing fitness landscape using EAs. This largely restricts the possible domain of DOPs, as it unnecessarily assumes that DOPs should be solved by EAs. The second type considers DOPs as a sequence of SOPs over time, and it is assumed that the DOP is solved optimally if and only if each of the SOPs is solved optimally. This formulation is certainly true in some real-world scenarios but not in others. For example, solutions determined for previous SOPs can have an impact on what future SOPs look like, as argued by Bosman [3]. Hence, in such cases, the overall performance can be suboptimal even though an optimal solution is found for each SOP. The third type defines DOPs as maximizing an integration of a DFF over a time period. Yet, it is not straightforward as how the decision maker would provide solutions to such DOPs. With regard to the fourth type, which is all descriptive, one obvious drawback is that those descriptions are not rigorous enough,

H. Fu and X. Yao are with CERCIA, School of Computer Science, University of Birmingham, Birmingham, UK (e-mail: hxf990@cs.bham.ac.uk; x.yao@cs.bham.ac.uk).

P. R. Lewis is with the School of Engineering and Applied Science, Aston University, Birmingham, UK (e-mail: p.lewis@aston.ac.uk).

B. Sendhoff is with Honda Research Institute Europe, 63073 Offenbach, Germany (e-mail: bernhard.sendhoff@honda-ri.de).

K. Tang is with UBRI, School of Computer Science and Technology, University of Science and Technology of China, Hefei, China (email: ketang@ustc.edu.cn).

This work was supported by Honda Research Institute Europe, an EPSRC grant (No. EP/K001523/1), the National Natural Science Foundation of China under Grants 61175065 and 61329302, the Program for New Century Excellent Talents in University under Grant NCET-12-0512, the Science and Technological Fund of Anhui Province for Outstanding Youth under Grant 1108085J16, and the European Union Seventh Framework Programme under Grant 247619.

[1]Maximization problems are considered in this paper without loss of generality. Also, we only consider single objective problems in this paper.

and people may interpret the same description differently.

In this paper, we propose a unified definition of DOPs. Our definition is inspired by the idea of multiple-decision-making in the Reinforcement Learning (RL) [18] community. As a result, most problems studied in RL can also be defined as DOPs according to our definition. The contribution of our definition of DOPs is three-fold. The definition captures the distinctive feature of DOPs compared to SOPs, which is multiple-decision-making. Secondly, according to our definition, most problems studied in EDO and RL are DOPs, and therefore the definition draws a connection between EDO and RL. Thirdly, different types of DOPs studied in EDO and RL can be explicitly categorised within our definition. The second contribution of this paper is a conceptualized DOP benchmark, which is derived from our definition of DOPs and developed for the purpose of systematically studying different types of DOPs. For the third contribution of this paper, one representative method from EDO and one representative method from RL are tested on our conceptualized DOP benchmark. The purpose is to show that EDO methods and RL methods are specialised in certain types of DOPs, and they can benefit each other in solving DOPs.

The rest of this paper is structured as follows. We give our unified definition of DOPs in Section II. In Section III, different types of DOPs studied in EDO and RL are briefly reviewed and categorised. A conceptualized DOP benchmark is developed in Section IV. Some experimental studies on the conceptualized DOP benchmark are conducted in Section V. Finally, conclusions and future work are given in Section VI.

II. A Unified Definition of DOPs

In our opinion, the distinctive feature of DOPs compared to SOPs is that the decision maker has to make multiple decisions over time, and the overall performance depends on all decisions made during an investigated time interval. In contrast, SOPs can be considered as one-decision-making problems. It should be noted that decisions in DOPs are being made sequentially over time. Also, decisions made previously may have an impact on later decision-making in DOPs.

There are various real-world situations where multiple decisions are being made over time, and we identify two main categories from the EDO literature. In the first category, decisions are being made in a fixed frequency, and this is mostly found in control problems. For instance, in the greenhouse control problem [20], a decision maker updates the control parameters every few seconds, so that the performance of the system over time is maximised. One update of the control parameters corresponds to one decision. Other examples in this category can also be found [16][12]. In the other category, decisions are being made over time in an event-triggering manner. In other words, a decision has to be made because something relevant in the environment has changed, and the decision maker has to react to the change by making a new decision. For instance, in the dynamic job shop scheduling problem [2], the decision maker has to assign new incoming jobs in an on-line manner. Also, when a machine breaks down, some jobs have to be reassigned. In this situation, an event corresponds to the arrival of new jobs or the breakdown of machines, and a corresponding decision is about scheduling new jobs or re-scheduling some existing jobs. There are some other examples in the second category [23][6].

The concept of multiple-decision-making has also been discussed in RL, in which a decision is also called an action. We first introduce some key concepts, borrowed from RL, for our definition of DOPs in the following.

State: A state contains all the information, which is relevant to decision-making. Simply put, a state is associated with a system, with which the decision maker is interacting, and can be understood as a set of variables $\boldsymbol{\alpha}$. The system's state is a function of time: the state at time[2] step t is $\boldsymbol{\alpha}_t$, which is assumed to be dependent on previous states and decisions made before time step t.

Action/Decision/Solution: We use action, decision, and solution interchangeably in this paper. The decision maker interacts with a DOP system by making decisions, one decision for one time step, in order to maximize a certain performance. The action taken at time step t is denoted as $\mathbf{x}_t$. $\mathbf{x}_t$ is chosen from an action set/space, $\mathcal{A}_t$, available at time step t, and $\mathcal{A}_t$ usually depends on $\boldsymbol{\alpha}_t$. For instance, assuming the investigated DOP is about setting control parameters to maximize a system's performance, the value of the control parameters at time step t is then the action $\mathbf{x}_t$ taken at time step t. It should be noted that usually some computational time is needed to come up with a decision.

Immediate Reward/Fitness: We use reward and fitness interchangeably in this paper. We assume that the decision maker receives an immediate reward every time after making a decision. The reward is just a signal that indicates the performance of the system at the time step when the decision is made. The reward can be understood as a real number with larger values for better performance. For instance, if the investigated DOP is about maintaining a system at a target state over a time period, the immediate reward after making a decision can be the similarity between the target state and the system's state at that time step. It should be noted that the objective of DOPs is not about maximizing the immediate reward at a time step but the accumulated rewards over a time period.

Bearing in mind these key concepts and the distinctive feature of DOPs compared to SOPs, which is multiple-decision-making over time, we define DOPs as follows:

Definition 2.1: DOPs are problems about how to make an optimal set of decisions over time in order to maximize a certain performance, which is a function of all decisions made over time. More formally, consider a time interval $[0, t_e]$, during which the system's state at time step t, $\boldsymbol{\alpha}_t$, follows a probabilistic distribution $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1})$, which is dependent on previous states and actions. A DOP can be stated as making a sequence of decisions, one at each time step during $[0, t_e]$, so that the decision sequence

[2]We only consider discrete time in this paper.

maximizes the expected accumulated rewards[3] over the entire time interval $[0, t_e]$:

$$\max \sum_{i=0}^{t_e} E(f_i(\boldsymbol{\alpha}_i, \mathbf{x}_i)), \tag{1}$$

where f_i is the reward function, which returns the immediate reward of an action taken in state $\boldsymbol{\alpha}_i$. f_i can be either deterministic or stochastic. $E()$ returns the expected value over the random variable f_i. It should be noted that some computational time is allowed for the decision maker to make a decision at each time step.

In the following, we would like to point out some potential assumptions with regard to DOPs. These assumptions can be used to differentiate different types of DOPs. We will demonstrate in the next section that existing EDO and RL have mainly studied only some types of DOPs via examining the corresponding underlying assumptions.

- **Assumptions about the observability of state**: Basically, the state $\boldsymbol{\alpha}$ can be fully observable, partially observable, or non-observable to the decision maker.
- **Assumptions about the dynamic of state**: These assumptions are solely related to the probability distribution $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1})$. In an extreme case, the full probability distribution is known to the decision maker. If the full probability distribution is not available, a common assumption is that it is Markovian: $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1}, \mathbf{x}_{t-1})$. In some other cases, the dynamic of state is assumed to be independent of decisions: $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1})$. If the dynamic of state is dependent on previous decisions, we say the DOP has a time-linkage property [3].
- **Assumptions about the reward function**: This is about to what extent the decision maker has information about the reward function f. The decision maker can have full information (i.e., a concrete function form of f) or no information (i.e., an action has to be made in order to get its immediate reward at a particular state) about f. Alternatively, the decision maker can have intermediate information about f such that the decision maker can query the immediate reward of any action at a time step without actually implementing the action.

III. DOPs Studied in EDO and RL

We can differentiate different types of DOPs based on our DOP definition by explicitly specifying those assumptions with regard to the observability of state, the dynamic of state, and the reward function. We demonstrate this via briefly reviewing DOPs studied in EDO and RL. We also identify some types of DOPs that may need more research attention in EDO.

[3]In general, we maximize the discounted accumulated rewards: $\sum_{i=0}^{t_e} E(\gamma^i \cdot f_i(\boldsymbol{\alpha}_i, \mathbf{x}_i))$, where γ is the user defined discount factor, $0 \leq \gamma \leq 1$. When t_e goes to infinity, $0 \leq \gamma < 1$. In this paper, we only consider finite t_e with $\gamma = 1$.

A. DOPs Investigated in EDO

Types of DOPs studied in EDO are determined by the benchmark problems and the algorithm performance measures used in EDO.

According to the latest survey on EDO [15], in nearly all benchmark problems in EDO, the state is independent of previous decisions: $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1})$. In addition, most benchmark problems are Markovian as well: $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1})$. Taking the widely used moving peaks benchmark [4] for example, the benchmark is generated by adding a random noise to the current state. Therefore, $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1})$ in the moving peaks benchmark. Other widely used benchmarks [13][25] also have $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1}) = P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_{t-1})$.

One exceptional benchmark, in which previous decisions have an impact on the dynamic of state, was developed by Bosman [3]. However, the benchmark is not general enough as there is not much flexibility in terms of specifying the observability of state, the dynamic of state, and the reward function.

The performance measures also determine what types of DOPs have been studied in EDO. According to [15], there are two major types of performance measures in EDO. The most common type is the optimality-based performance measure, and the other is the behaviour-based performance measure. We will discuss only optimality-based performance measures. The reason is that behaviour-based performance measures usually measure the population diversity in EAs when solving DOPs and therefore are not properties of the underlying DOPs themselves.

One popular optimality-based performance measure is the best-of-generation measure [24], which takes the form:

$$F_1 = \frac{1}{G} \sum_{i=1}^{G} (\frac{1}{N} \sum_{j=1}^{N} F_{BOG_{ij}}), \tag{2}$$

where G is the number of generations. N is the number of algorithm runs, and $F_{BOG_{ij}}$ is the best fitness[4] of individuals at generation i of the jth run. The underlying assumption about the DOPs when using this performance measure is that not every fitness evaluation contributes to the algorithm's performance. In other words, it is assumed that the decision maker has intermediate information about the reward function such that a query of an action's immediate reward, i.e., a fitness evaluation, can be performed without actually implementing the action.

Another widely used performance measure is the off-line performance [5], which takes the form:

$$F_2 = \frac{1}{G} \sum_{i=1}^{G} F_{BS_i}, \tag{3}$$

[4]A solution's fitness is an alias of the immediate reward received by implementing the solution, i.e., making a decision.

where G is the number of generations. F_{BS_i} is the best fitness obtained by the algorithm since the last state change till the ith generation. Similarly, not every fitness evaluation counts in the measure of off-line performance, which indicates the availability of intermediate information about the reward function. Besides, when using the off-line performance, it is assumed that every time a better action, in terms of immediate reward, is found the action is implemented.

A lot of researchers in EDO also use the measure named best-error-before-change [19]:

$$F_3 = \frac{1}{m}\sum_{i=1}^{m} e_i, \tag{4}$$

where m is the number of state changes, and e_i is the best error (the error means the difference between a solution's fitness and the fitness of an optimum) just before the ith change happens. The best-error-before-change takes a similar form to Equation 1, and yet it is still implicitly assumed that the intermediate information about the reward function is available.

We are also aware of the on-line performance measure proposed in [5], which takes the form:

$$F_4 = \frac{1}{N_{fe}}\sum_{i=1}^{N_{fe}} F_i, \tag{5}$$

where F_i is the fitness of the ith fitness evaluation, and N_{fe} is the total number of fitness evaluations. This measure does not assume the information of the reward function. Yet, the on-line performance measure has been rarely used in EDO [15].

B. DOPs Investigated in RL

Problems studied in RL are also about multiple-decision-making, which can be defined as DOPs according to our definition in Equation 1. In the following, we briefly analyse the framework of a general RL method, from which we discuss what assumptions have been made in DOPs studied in RL.

A typical RL algorithm can be generally described as follows. Initially, the agent, i.e., the decision maker, has no prior information about the DOP. The agent first observes the current state of the system and then implements an action. After the action, the system transits into the next state, and an immediate reward is returned to the agent. By interacting with the system this way for a number of time steps, the agent gradually learns a model that summarises all the experiences so far. One experience corresponds to a triplet $(state, action, reward)$, which means the agent received a reward after taking an action in a particular state. The model can be the value function of state or the Q function of the pair $(state, action)$. Through the loop of 'trial-and-error', it is hoped that the agent will gradually approximate the true model, which gives the optimal policy in terms of rewards accumulated in the long run. A policy is basically a mapping from a state to an action. For more information about the value function, the Q function, the policy, and more general information about RL, readers are referred to [18].

TABLE I

ASSUMPTIONS ABOUT DOPS STUDIED IN EDO AND RL. THESE ASSUMPTIONS ARE ABOUT THE OBSERVABILITY OF STATE, THE DYNAMIC OF STATE, AND THE REWARD FUNCTION. 'N' MEANS NO INFORMATION IS ASSUMED. 'I' MEANS INTERMEDIATE INFORMATION IS ASSUMED. 'F' MEANS FULL INFORMATION IS ASSUMED.

research community	observability of state			dynamic of state			reward function		
	'N'	'I'	'F'	'N'	'I'	'F'	'N'	'I'	'F'
EDO	√				√			√	
RL		√		√			√		

There are some underlying assumptions about DOPs in RL. Firstly, the state is at least partially observable to the decision maker, otherwise all the learning about the value function or the Q function would be infeasible. Secondly, most RL algorithms assume that the dynamic of state is Markovian or nearly Markovian. Finally, no information is required a priori about the reward function. In other words, in order to get the immediate reward of an action in a state, the action needs to be implemented.

C. A Comparison View of DOPs in EDO and DOPs in RL

In this subsection, we summarise different types of DOPs studied in EDO and RL based on our previous discussions. We classify DOPs into different groups by checking whether those assumptions discussed in Section II are made. The summary is tabulated into Table I.

In Table I, we divide assumptions into three levels within each category of assumptions. 'N' means no information is assumed. 'I' means intermediate information is assumed. 'F' means full information is assumed. For the assumptions about the observability of state, what existing EDO methods need for solving DOPs is being able to evaluate a solution (i.e., get its fitness) at the current time step. The observability of state is irrelevant to EDO methods. In contrast, the state should be at least partially observable to RL methods. Accordingly, EDO and RL are marked as 'N' and 'I' respectively in this category.

For the assumptions about the dynamic of state, most of the work in EDO deals with DOPs where previous decisions have no impact on later dynamic of state. Besides, EDO methods do not require the full information of state, i.e., the probability distribution $P(\boldsymbol{\alpha}_t|\boldsymbol{\alpha}_0, ..., \boldsymbol{\alpha}_{t-1}, \mathbf{x}_0, ..., \mathbf{x}_{t-1})$. We therefore mark EDO as 'I' in this category. Although traditional RL methods assume the dynamic of state to be Markovian, a lot of RL methods have been proven to be effective in situations where the Markov property does not hold. We therefore mark RL as 'N' in this category.

With regard to the assumptions about the reward function, from the performance measures used in EDO, we can see that most EDO methods assume that a fitness evaluation can be performed without actually taking an action. Therefore, an evaluation model for the reward function is assumed to be available for these EDO methods. Yet, EDO methods do

not require the analytical form of the reward function. In contrast, in order to get the immediate reward of an action in a certain state, that action has to be implemented in RL methods. Therefore, EDO and RL are marked as 'I' and 'N' respectively in this category.

From Table I, we can see that more research attention should be given to DOPs where previous decisions can influence the dynamic of state and DOPs where an evaluation model of the reward function is not available, from the perspective of EDO. It will also be interesting to compare the performances of EDO methods and RL methods on the same DOPs.

IV. A Conceptualized DOP Benchmark

A lot of DOP benchmarks have been developed in EDO [15], in which common features are that decisions made before have no impact on later dynamic of state and the decision maker is able to do a fitness evaluation without implementing an action. There are also many benchmarks developed in RL [8], which can be seen as DOP benchmarks according to our definition of DOPs. Benchmarks in RL have some common features that may favour only RL methods, such as a discrete action space, no evaluation model of the reward function, and full observability of states. In order to systematically studying different types of DOPs, we develop a Conceptualized DOP benchmark in the following. By making DOP assumptions discussed in Section II explicitly, our conceptualized DOP benchmark enables the generation of different types of DOPs, which can be hardly achieved within existing DOP benchmarks. Based on our definition of DOPs, we define the state space, the action space, the reward function, and the dynamic of state respectively in the benchmark.

We generalize the moving peaks benchmark [4] to define our benchmark. The concepts of height, width, and center for the reward/fitness function in the moving peaks benchmark are transferred to our benchmark.

A. The State Space

A state is simply a set of variables in our benchmark. Each state consists of four parts, namely height, width, center, and bias. Each height, width, and center are associated with a peak function, whose meaning will be explained later in the reward function. The bias is just a single scalar. There are two parameters to control the dimension of state. One is called the number of peaks m, and the other is the dimension of peak n. Simply put, a state $\boldsymbol{\alpha}$ is represented using the vector $(h_1, h_2, ..., h_m, w_1, w_2, ..., w_m, c_{11}, c_{12}, ..., c_{mn}, b)$ of length $(n+2)m+1$, where h_i and w_i denote the height and the width of the ith peak function. $\mathbf{c}_i$ ($\mathbf{c}_i = (c_{i1}, c_{i2}, ..., c_{in})$) is the center of the ith peak function, and b is the bias.

B. The Action Space

An action is specified by a set of variables. For a state space that has the dimension of peak function being n, there are n variables to specify an action: $\mathbf{x} = (x_1, x_2, ..., x_n)$. The action/solution space $\mathcal{A}_t$ for the state $\boldsymbol{\alpha}_t$ is basically a subset of the n dimensional real numbers $\mathbb{R}^n$.

C. The Reward Function

The reward function takes a state and an action as inputs and outputs a real number. The reward function in our benchmark consists of the bias b and several peak functions ($h_i - w_i||\mathbf{x} - \mathbf{c}_i||_2$ is the ith peak function):

$$f(\boldsymbol{\alpha}, \mathbf{x}) = \max_{i=1}^{m}\{h_i - w_i||\mathbf{x} - \mathbf{c}_i||_2\} + b, \tag{6}$$

where $\boldsymbol{\alpha}$ represents a state $\boldsymbol{\alpha} = (h_1, h_2, ..., h_m, w_1, w_2, ..., w_m, c_{11}, c_{12}, ..., c_{mn}, b)$, and $\mathbf{x}$ is an action. $\mathbf{c}_i$ denotes a vector $(c_{i1}, c_{i2}, ...c_{in})$, and $||\ ||_2$ is the Euclidean norm. Without loss of generality, we require $w_i > 0$ $(1 \leq i \leq m)$.

D. The Dynamic of State

We divide all state variables in $\boldsymbol{\alpha}$ into two groups. The dynamic of the first group does not depend on previous actions, while the dynamic of the second group depends on previous actions.

The first group consists of all variables except for the bias b in $\boldsymbol{\alpha}$. We employ the 6 different dynamics suggested in CEC09 dynamic optimization competition benchmark generator [11] to update the state variables in the first group. The 6 dynamics are described as follows:

1) *small step*:

$$\Delta\phi = \gamma \cdot ||\phi|| \cdot r \cdot \phi_{severity}, \tag{7}$$

2) *large step*:

$$\Delta\phi = ||\phi|| \cdot (\lambda \cdot sign(r) + (\lambda_{max} - \lambda) \cdot r) \cdot \phi_{severity}, \tag{8}$$

3) *random*:

$$\Delta\phi = \mathcal{N}(0,1) \cdot \phi_{severity}, \tag{9}$$

4) *chaotic*:

$$\phi_{t+1} = \phi_{min} + \beta \cdot (\phi_t - \phi_{min})(1 - (\phi_t - \phi_{min})/||\phi||), \tag{10}$$

5) *recurrent*:

$$\phi_t = \phi_{min} + ||\phi|| \cdot (sin(\frac{2\pi}{P}t + \varphi) + 1)/2, \tag{11}$$

6) *recurrent with noise*:

$$\phi_t = \phi_{min} + ||\phi|| \cdot (sin(\frac{2\pi}{P}t + \varphi) + 1)/2 + \mathcal{N}(0,1) \cdot noise_{severity}, \tag{12}$$

where ϕ represents any single state variable in the first group of $\boldsymbol{\alpha}$. ϕ_t is the value of ϕ at time step t, and $\Delta\phi$ denotes the change in ϕ between two consecutive time steps: $\phi_{t+1} = \phi_t + \Delta\phi$. ϕ_{min}, $||\phi||$, and $\phi_{severity}$ denote the minimum value of ϕ, the range of ϕ, and the change severity of ϕ respectively. λ and λ_{max} are constant parameters. A logistic function is used for the *chaotic* change: β is a positive constant in the interval $(1, 4)$. P is the period for the *recurrent* change and the *recurrent with noise* change, and φ is an initial phase. r is a random number drawn uniformly from the interval $(-1, 1)$, and $sign(r)$ returns 1 when r is positive, -1 when r is negative, and 0 otherwise. $\mathcal{N}(0,1)$ is a random number drawn from the Gaussian distribution with

mean 0 and variance 1. $noise_{severity}$ is the noise severity applied to the *recurrent with noise* change.

The bias b_{t+1} in the state depends on the latest k actions and biases for the latest l time steps:

$$b_{t+1} = g(\mathbf{x}_{t-k+1}, ..., \mathbf{x}_t, b_{t-l+1}, ..., b_t), \tag{13}$$

where $g()$ is a user-defined function returning a real number.

Since we use the concept of peak function in defining the reward function in our benchmark, and our benchmark is aimed at generating different types of DOPs, we name our benchmark Conceptualized Moving Peaks Benchmark (CMPB).

E. Specifications for a Benchmark Instance of CMPB

There are two groups of things need to be specified in order to get a benchmark instance of CMPB.

In the first group, the parameters regarding to the state space and the action space need to be specified. The type of dynamic for the state needs to be specified as well, including the function $g()$ in Equation 13.

In the second group, assumptions discussed in Section II need to be stated explicitly. It should be made clear whether the decision maker is given the full analytical form of the reward function, the intermediate information of the reward function in which the decision maker can pose questions about the immediate reward of any action without implementing such an action, or no information of the reward function where an action has to be implemented in order to get its immediate reward at a particular state. The assumption about the dynamic of state is determined once options in the first group are specified, e.g, whether the dynamic of state is Markovian. Finally, it is necessary to specify to what extent the decision maker has information about the state $\boldsymbol{\alpha}$: $\boldsymbol{\alpha}$ can be fully observable, partially observable in which, e.g., a white noise is added to $\boldsymbol{\alpha}$ before it is passed to the decision maker, or non-observable.

V. Experimental Studies

In this section, we generate two benchmark instances of CMPB. More importantly, we conduct some preliminary experimental studies of the performances of a representative EDO method and a representative RL method on the two benchmark instances. The purpose is to reveal which types of DOPs EDO methods and RL methods are specialised in respectively.

A. Two Benchmark Instances of CMPB

We generate the first benchmark instance as follows. The peak dimension and the number of peaks in the state are both 1, which makes the dimension of the state 4. Accordingly, the action space is one-dimensional. The height and the width in the state are fixed to 30 and 2 respectively. The range of the center in the state is $[-10, 10]$, and the action space is also $[-10, 10]$. The dynamic of the center is recurrent with period being 2 time steps: the center starts at 5 and is multiplied by -1 every time the environmental state changes (i.e., the time step is increased by 1). The function $g()$, which defines the dynamic of the bias in the state, is:

$$b_t = \begin{cases} \theta_b & if \quad \mathbf{x}_{t-1} \geq 0, \\ -\theta_b & otherwise, \end{cases} \tag{14}$$

where $\mathbf{x}_{t-1}$ is the action taken at time step $t-1$, and θ_b is a parameter, which controls the influence of $\mathbf{x}_{t-1}$ on b_t, with larger values being more influential. In the first benchmark instance, θ_b is set to be 100.

The settings of the second benchmark instance is exactly the same as the first one, except the parameter θ_b in Equation 14 is set to be 15. The reason to design such two benchmark instances is that we would like to compare the performances of EDO methods and RL methods in two representative situations. One situation is that actions made previously have a large impact on the dynamic of state, while the other is that actions made previously have little impact on the dynamic of state, i.e., maximizing the reward function at each time step separately tends to be equivalent to maximizing the accumulated rewards. Such experimental design is useful in indicating the advantages of EDO methods and RL methods.

For both benchmark instances, the dynamic of the state and the reward function are assumed to be deterministic. Also, the dynamic of the state is Markovian. We assume that the decision maker has intermediate information about the reward function, i.e., a fitness evaluation can be performed without implementing an action. Besides, the state is assumed to be fully observable to the decision maker.

B. Investigated Algorithms in EDO and RL

The Particle Swarm Optimizer (PSO) with a restart strategy is selected as the representative EDO method, which we denote as 'RPSO' hereafter. The restart strategy means the whole population is randomly reinitialized except that the best solution found in the last time step is copied into the current population whenever the environmental state changes. The corresponding PSO follows the constriction version [7]. The swarm population size is 10. The two constants, which are used to bias a particle's attraction to the local best and the global best, are both set to be 2.05, and hence the constriction factor takes a value of 0.729844. The velocity of particles is constricted within the range $[-V_{MAX}, V_{MAX}]$. The value of V_{MAX} is set to be the range of the search space, which is 20 in our case. 100 fitness evaluations are allowed for 'RPSO' to come up with an action at a time step.

The Q learning algorithm [21] is selected as the representative RL method, which we denote as 'Q-learning' hereafter. The semi-uniform random exploration strategy is used to select an action in 'Q-learning', where at each time step the best action in terms of the currently estimated Q value is selected with some probability $1-\epsilon$, and with probability ϵ, an action is selected at random. In our experiment, ϵ is set to be 0.1. The learning rate η in 'Q-learning' takes the form: $\eta = q_1/(q_2 + t)$, where q_1 and q_2 are two constants, and t is the index of the time step, which starts from 0 and increases by 1 every time the environmental state changes. q_1 and q_2 are set to be 200 and 300 respectively. The discount factor

in 'Q-learning' is set to be 0.7. Also for 'Q-learning', the action space is discretized, which in our experiment is the set $\{-10, -9, -8, ..., 8, 9, 10\}$. At the beginning of the time (i.e., time step $t = 0$), the Q values in 'Q-learning' are all initialised to 0.

The main feature of 'RPSO' is that at each time step it tries to select an action that maximizes the current reward function. In other words, 'RPSO' does not care about what impact the selected action would have on the dynamic of state. This feature is also shared by most other EDO methods [15]. The main feature of 'Q-learning' is that it tries to learn the true Q value of each state and action pair. Based on the true Q values, optimal actions in terms of maximizing the discounted accumulated rewards can be derived for any state. It has been shown in [21] that for Markovian decision problems (i.e., DOPs defined in this paper with the dynamic of state being Markovian), 'Q-learning' converges to the true Q values with probability 1 under mild conditions as the number of time steps goes to infinity. However, in reality, there is always a trade-off between exploration and exploitation in 'Q-learning', and the performance within finite time steps depends on the structure of the Markovian decision problem and the parameters in 'Q-learning'.

C. Experimental Results

1) Results on the First Benchmark Instance: The performances of 'RPSO' and 'Q-learning' on the first benchmark instance, in which an action has a large impact on the bias at the next time step ($\theta_b = 100$), are presented in Figure 1 and Table II. We can see that 'Q-learning' achieves a consistently better performance than 'RPSO'. The reason is that the dependence of the bias in the state on the last action is gradually learned in 'Q-learning', and therefore at each time step 'Q-learning' tends to select an action, which results in the bias being positive at the next time step. In contrast, an action is selected in 'RPSO' solely depending on the current reward function. When the center in the state has a negative value, 'RPSO' will tend to select an action with a negative value. As a result, the bias will oscillate between 100 and -100.

The optimal action at each time step is straightforward to obtain for the first benchmark instance. The optimal action at each time step is that the action equals 5 when the center is positive and the action equals 0 when the center is negative. The optimal accumulated rewards at each time step, denoted as 'Optimal', is also plotted in Figure 1.

2) Results on the Second Benchmark Instance: The performances of 'RPSO' and 'Q-learning' on the second benchmark instance, in which an action has a small impact on the bias at the next time step ($\theta_b = 15$), are presented in Figure 2 and Table II. The optimal action at each time step is the same as that for the first benchmark instance. The optimal accumulated rewards at each time step, denoted as 'Optimal', is also plotted.

From Figure 2, we can see that 'RPSO' achieves a consistently better performance than 'Q-learning'. This is due to the following three aspects. Firstly, the impact of an action

TABLE II

AVERAGE AND STANDARD DEVIATION OVER 30 RUNS OF TOTAL REWARDS ACCUMULATED DURING DIFFERENT PHASES. THE 1ST PHASE IS FROM TIME STEP 1 TO 1000. THE 2ND PHASE IS FROM TIME STEP 1001 TO 2000.

Benchmark Instance	Method	1st Phase Avg.	1st Phase Dev.	2nd Phase Avg.	2nd Phase Dev.
1st instance	RPSO	30097	3.9	29897	3.8
	Q-learning	92801	22575.0	105644	11719.0
2nd instance	RPSO	29927	3.3	29897	2.7
	Q-learning	26035	4743.5	26951	4426.9

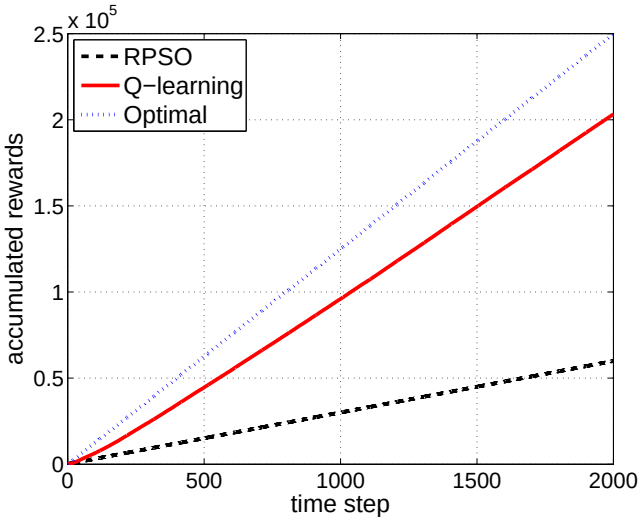

Fig. 1. The averaged accumulated rewards in Equation 1 over 30 runs at each time step, obtained by 'RPSO' and 'Q-learning' respectively on the first benchmark instance, together with the optimal accumulated rewards at each time step.

on the bias at the next time step is small, so maximizing only the current reward function still gives a relatively good performance. Secondly, in order to converge to the true Q values, all actions including suboptimal or even the worst actions in each state need to be implemented many times in 'Q-learning'. This decreases the overall performance. In other words, learning in 'Q-learning' has an overhead of implementing poor actions in each state. Finally, within finite time steps, the convergence rate to the true Q values in 'Q-learning' has a large impact on the performance. The convergence rate is influenced by the initialised Q values, the exploration strategy, the learning rate, and the way to update the Q values after an action is implemented. However, it is not straightforward as how to set the parameters in 'Q-learning' properly.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a unified definition of DOPs based on the idea of multiple-decision-making over time inspired by RL. We draw a connection between EDO and RL, arguing that both EDO and RL are trying to solve DOPs according to our definition of DOPs. We point out that some types of DOPs, where previous decisions can influence the dynamic of state or an action has to be

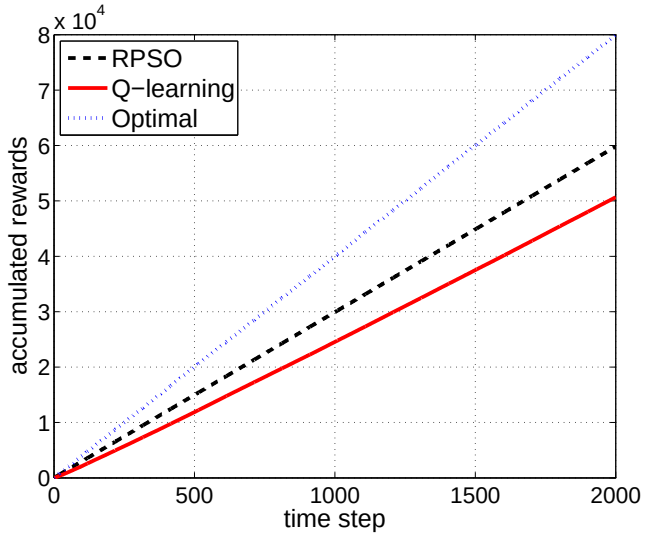

Fig. 2. The averaged accumulated rewards in Equation 1 over 30 runs at each time step, obtained by 'RPSO' and 'Q-learning' respectively on the second benchmark instance, together with the optimal accumulated rewards at each time step.

implemented to get its fitness, need more research attention from the perspective of EDO. Moreover, a conceptualized DOP benchmark, CMPB, is developed for the purpose of systematically studying various types of DOPs within one benchmark. Finally, some interesting experimental results are obtained by testing one representative EDO method and one representative RL method on CMPB.

The experimental studies may indicate that EDO methods, compared to RL methods, may be better at DOPs where actions have a small impact on the dynamic of state. In contrast, as the impact of actions on the dynamic of state increases, the advantage of learning in RL methods may take over EDO methods. Yet, more comprehensive experimental studies are needed for any concrete conclusion.

For the future work, firstly, more research attention in EDO should be given to DOPs in which actions taken previously can influence the later dynamic of state or an evaluation model of the reward function is unavailable. Secondly, since we have established a connection between EDO and RL, arguing that both of them are trying to solve DOPs based on our definition of DOPs, it would then be beneficial to comprehensively compare the state-of-the-art methods in EDO with those in RL on various types of DOPs using CMPB, as a deep understanding can be gained about their own advantages. Finally, it would be interesting to combine the advantages of both EDO and RL methods in solving DOPs.

REFERENCES

[1] V. S. Aragón and S. C. Esquivel. An evolutionary algorithm to track changes of optimum value locations in dynamic environments. *Journal of Computer Science and Technology*, 4(3):127–134, 2004.

[2] G. J. Barlow and S. F. Smith. A memory enhanced evolutionary algorithm for dynamic scheduling problems. In *Applications of Evolutionary Computing*, pages 606–615. Springer, 2008.

[3] P. Bosman. Learning and anticipation in online dynamic optimization. *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 129–152, 2007.

[4] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.

[5] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[6] L. T. Bui, Z. Michalewicz, E. Parkinson, and M. B. Abello. Adaptation in dynamic environments: a case study in mission planning. *Evolutionary Computation, IEEE Transactions on*, 16(2):190–209, 2012.

[7] M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.

[8] A. Dutech, T. Edmunds, J. Kok, M. Lagoudakis, M. Littman, M. Riedmiller, B. Russell, B. Scherrer, R. Sutton, S. Timmer, et al. Reinforcement learning benchmarks and bake-offs ii. In *Workshop at advances in neural information processing systems conference*, 2005.

[9] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithm with dominance and diploidy. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 59–68. L. Erlbaum Associates Inc., 1987.

[10] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3):303–317, 2005.

[11] C. Li and S. Yang. A generalized approach to construct benchmark problems for dynamic optimization. In *Simulated Evolution and Learning*, pages 391–400. Springer, 2008.

[12] Z. Michalewicz, M. Schmidt, Ma. Michalewicz, and C. Chiriac. Adaptive business intelligence: three case studies. In *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 179–196. Springer, 2007.

[13] R. W. Morrison and K. A. De Jong. A test problem generator for non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 2047–2053 Vol. 3. IEEE, 1999.

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

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

[16] H. Pohlheim and A. Heißner. Optimal control of greenhouse climate using real-world weather data and evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1672–1677, 1999.

[17] P. Rohlfshagen and X. Yao. Dynamic combinatorial optimisation problems: an analysis of the subset sum problem. *Soft Computing*, 15(9):1723–1734, 2011.

[18] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[19] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3. IEEE, 1999.

[20] R. K. Ursem, T. Krink, M. T. Jensen, and Z. Michalewicz. Analysis and modeling of control tasks in dynamic systems. *Evolutionary Computation, IEEE Transactions on*, 6(4):378–389, 2002.

[21] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[22] K. Weicker. An analysis of dynamic severity and population size. In *Parallel Problem Solving from Nature–PPSN VI*, pages 159–168. Springer, 2000.

[23] S. Yang, H. Cheng, and F. Wang. Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(1):52–63, 2010.

[24] S. Yang and X. Yao. Dual population-based incremental learning for problem optimization in dynamic environments. In *7th Asia Pacific Symposium on Intelligent and Evolutionary Systems: 49-56, 2003*, 2003.

[25] S. Yang and X. Yao. Population-based incremental learning with associative memory for dynamic environments. *Evolutionary Computation, IEEE Transactions on*, 12(5):542–561, 2008.