Information Sciences 302 (2015) 33–49

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Detecting change and dealing with uncertainty in imperfect evolutionary environments

Hasan Mujtaba [a,*], Graham Kendall [b,c], Abdul Rauf Baig [d], Ender Özcan [e]

[a] Department of Computer Science, National University of Computer and Emerging Sciences, Islamabad, Pakistan
[b] University of Nottingham, UK
[c] University of Nottingham Malaysia Campus, Malaysia
[d] Department of Information Systems, College of Computer & Information Sciences, Al Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, Saudi Arabia
[e] School of Computer Science, University of Nottingham, UK

A R T I C L E   I N F O

Article history:
Received 19 July 2013
Received in revised form 22 November 2014
Accepted 31 December 2014
Available online 13 January 2015

Keywords:
Artificial intelligence
Evolutionary computation
Imperfect Evolutionary Systems
Particle Swarm Optimization
Learning

A B S T R A C T

Imperfection of information is a part of our daily life; however, it is usually ignored in learning based on evolutionary approaches. In this paper we develop an Imperfect Evolutionary System that provides an uncertain and chaotic imperfect environment that presents new challenges to its habitants. We then propose an intelligent methodology which is capable of learning in such environments. Detecting changes and adapting to the new environment is crucial to exploring the search space and exploiting any new opportunities that may arise. To deal with these uncertain and challenging environments, we propose a novel change detection strategy based on a Particle Swarm Optimization system which is hybridized with an Artificial Neural Network. This approach maintains a balance between exploitation and exploration during the search process. A comparison of approaches using different Particle Swarm Optimization algorithms show that the ability of our learning approach to detect changes and adapt as per the new demands of the environment is high.

1. Introduction

An individual as a system shows intelligence with its actions, i.e. an intelligent individual reacts according to its surrounding and needs. The behavior of an intelligent individual is expected to be in coherence with the current situation which the individual is experiencing. The individual should cater for the changes in its environment and objective(s) enabling itself to make the appropriate decisions under perceptual and computational self-limitations [38]. Ideally, an intelligent entity should be capable of detecting new information which becomes available at any time, extracting the relevant knowledge, learning and reacting accordingly without the need of any third party intervention, thereby forming a dynamic relationship with its surroundings. Our work deals with this relationship of an intelligent entity with its surrounding environment. We argue that this relationship has been ignored in the past and our experimentation will show how this ignorance affects the abilities of learning agents.

In this paper we deal with dynamic environments of *imperfect* information, i.e. the set of information changes over time. The information available in the environment is incomplete because new information is constantly added to it or previous information is removed from it. This information can be anything from the availability of new input parameters, changes in

* Corresponding author.

http://dx.doi.org/10.1016/j.ins.2014.12.053
0020-0255/© 2015 Elsevier Inc. All rights reserved.

the objective(s) of the agents or changes in the rules of the environment. The notion of imperfection in evolutionary environments, which will be referred to as Imperfect Evolutionary Systems (IES) from this point forward, was presented by Kendall and Su [25]. IESs are uncertain and chaotic (where the changes can occur abruptly at any time) dynamic environments which provide a variety of challenges to the learning individuals residing in them. In simple words, in a dynamic environment of perfect information, we know what we do not know, whereas in an IES we do not know, what we do not know.

Learning in an IES is limited by the information available, any addition to this information opens up new avenues of learning and acquiring new skills become possible. Such imperfection of information is commonly found in real world problems. A human player may play checkers and then look at a game of chess and decide to learn that game. The same player may focus on other games or may learn a different task altogether. Investors in a stock market may change their strategies based on new information that was not previously available to them. This new information maybe a change in the known indicators that contribute to stock prices or it may add a new indicator previously unknown. The same investor may then leave for a racing track and decide to make bets on a racing horse. These two problems would require analysis of a completely different set of parameters. This ability to deal with imperfection also allows humans to deal with new problems or challenges in our daily life and enables us to detect irrational or unnatural behavior in known scenarios. Humans are able to detect when another player is cheating in a game but evolutionary strategies have had limited success in detecting such behavior.

We argue that the problem facing an intelligent entity is not only to excel in solving a particular problem but also the ability to deal with any problem without any *a priori* information. Due to its uncertain and dynamic nature, an IES presents new challenges to its habitants. These changes in the environment require its habitants to adapt according to the challenges and modify behavior without the need to rely on any human assistance.

Traditional learning approaches fail to embrace the incomplete nature of an IES. We believe, this failure is mostly due to ignorance of the learning strategy to its surroundings. They assume the environment to be perfect and complete; therefore addition of new information is ignored in the evolutionary process. Another weakness of traditional approaches lies in handling detection of changes. These techniques either rely on human intervention to provide information about the changes or change detection is mostly done by reevaluating currently known solutions to see if their fitness values have changed or not. Results of our experiments will show that this approach to detecting change limits the ability of a learning algorithm to perform in noisy and imperfect environments. Furthermore, a simple re-initialization of agents would not be enough to cater for the updated information added to or removed from the situation. Re-initialization of best strategies also means to abandon all learning done till that point and all computational cost till that point is wasted.

In our work we present a learning algorithm that overcomes all these problems. We propose learning to be a continuous process, broken learning into three abstract and simplified sub-processes. Our learning strategy allows individuals to build a dynamic relationship with their surroundings based on the information available to them. Whenever this information changes, these individuals assess their performance and effectiveness of their strategies. If new information is found to be relevant to an improvement in chances of survival then this information is added to knowledge pool otherwise it is discarded.

Our results will show that this learning approach performs well in both perfect and imperfect environments. However, our primary focus in this work has been to create intelligence that is capable of dealing with the incompleteness of any environment. Our learning approach is able to handle abrupt uncertainties which are inherent in an IES. We will show how traditional learning approaches fail to meet the needs of an imperfect world. These approaches either ignore the availability of new information or rely on humans to provide explicit information about the change.

Since learning in such an environment is an ongoing process, it requires a learning approach that is continuous in nature. In this paper, we present such a learning approach that is able to seek its objectives in an imperfect world, detect changes occurring in the environment and adapt accordingly. Our learning approach allows agents to acquire new skills that help them survive the demands of their environment. These skills are acquired and updated by the agents themselves in an automated manner based solely on the information provided by the environment. No other information or guidance is provided to the learning algorithm. This allows agents to learn and adapt as per the requirements of their environment (these requirements change over time without any warning) and thus improve their chances of survival in an automated manner, mimicking the behavior of naturally intelligent individuals. Instead of optimizing a particular skill for a particular environment, we aim to develop an approach that allows agents to learn multiple objectives (or skills) on their own without any human guidance.

In order to test our hypothesis of continuity of learning in uncertain environments we developed an IES to perform our experimentation. This IES is modeled to handle uncertainty and imperfection of information and create new challenges for its inhabitants. It constitutes of different entities both learning and non-learning. We believe that the abstract nature of the environment would allow other researchers to add new entities and challenges to it. Fig. 1 presents a diagrammatic view of this IES at different time stages. E^T is the state of the Imperfect Evolutionary System at time T (represented by the large outer circle). E^T is defined by a set of information blocks. Any change in the environment will bring about a change in the search space with the consequence of no point being the optimal for all the different states of the environment. New information for inhabitants would present exploration opportunities (Fig. 1b), whereas removing previously available information would render evolved strategies to be obsolete (Fig. 1c). Hence we can define E^T in terms of the information available to its inhabitants as in Eq. (1).

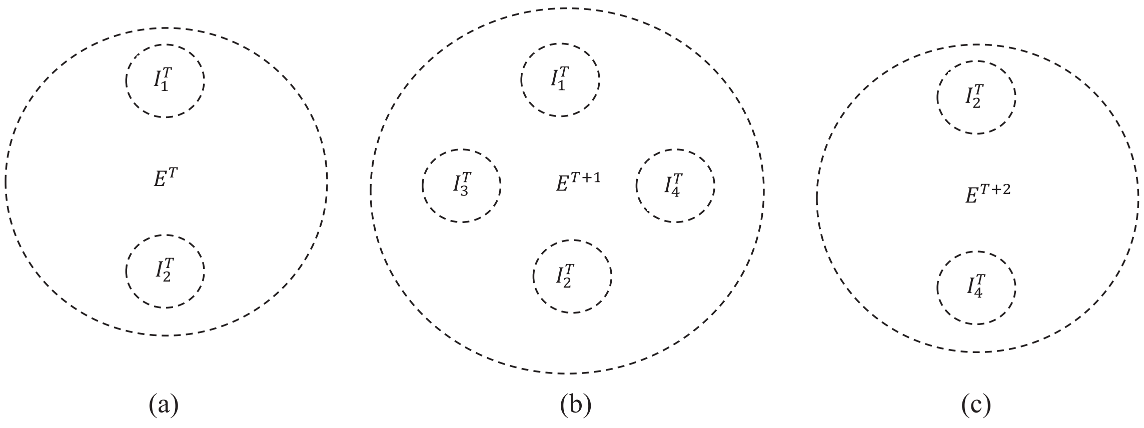


Fig. 1. Imperfect Evolutionary System.

$$E^T = \left\{ I_1^T, I_2^T, I_3^T, \ldots, I_l^T, \ldots, I_n^T \right\} \tag{1}$$

where n is the maximum number of information blocks that are available and I_l^T is the lth information block available for the individual of the IES at time T. We propose that each block of information is itself a set formed by the union of smaller blocks of information about the environment. A block of information, denoted as I_l^T is comprised of (i) available input, x_l^T, (ii) effective rules, r_l^T, (iii) objectives which the individuals must achieve, o_l^T, (iv) novice entities present in the environment, u_l^T, and (v) intelligent entities present in the environment, a_l^T.

$$I_l^T = \sum_{\forall i} x_l^T \cup \sum_{\forall i} r_l^T \cup \sum_{\forall i} o_l^T \cup \sum_{\forall i} u_l^T \cup \sum_{\forall i} a_l^T \tag{2}$$

where $\sum_{\forall i} b^T$, represents the set of all relevant piece of information b visible to all the individuals i residing in all locations of the environment at time T. The partially observable nature of the IES implies that different individuals will be affected by different parameters at any given time. A change maybe local to an area and affect individuals residing in that locality while others may be unaware of that information. Therefore individuals may be exposed to different sets of information. This allows each individual in the environment to build its own perspective about the surroundings based upon its observational space. Eq. (3) presents the perspective of an individual I_A^T and the environmental variable v_n^T, currently present in its observable space.

$$I_A^T = \left\{ v_j^T, \ldots, v_k^T | j, k \in n \right\} \tag{3}$$

Eq. (2) presents the information from the environments perspective, while (3) presents it from the individual's point of view.

Details of this environment are presented in Section 4. It is our hope that this environment will serve as a test bed for further research into IES. In the next section we will discuss related work. Section 3 presents our algorithm while Section 4 presents details of our test-bed IES. Experimental setup is detailed in Section 5 and results are presented in Section 6. Conclusions and future work is discussed in Section 7.

2. Related work

Learning environments can generally be divided into two categories: static (steady) and dynamic (varying). Evolutionary algorithms (EAs) are commonly used for solving computationally hard problems. EAs have been applied to both kinds of environments. In static problems there is no change in the environment while solving a given problem instance. Generally, an EA performs search using a population of individuals where each individual represents a candidate solution for a given problem instance. These individuals interact with each other during the search process via evolutionary operators.

Dynamic environments (DE) are ever-changing environments, which can be observed in many real world problems where a change can occur at any time while the search is in progress. Recently, there are a growing number of studies on dynamic environments, such as, dynamic optimization problems (DOP). However, most of the studies assume that the DE is perfect. By saying that an environment is perfect we mean that the set of information available at time Ti to Ti + n remains unchanged. In a DE, change usually means modifying some environment variables. These environments are not based to model the imperfection of the environment. An example of such an environment is the Moving Peaks Benchmark (MPB). MPB is a well-known problem that provides a dynamic environment of perfect information, where the learning algorithm has to locate a number of peaks present in the environment [5,7]. These peaks can change their spatial location, height and slope. Any good learning strategy would have locate the peaks and then track their movements once the environment changes. However, these environments were not modeled to support imperfection of the environment. For example, any strategy working within MPB would always be aware of the surrounding environment and the challenges it presents. In summary, by DE we mean a dynamic environment of complete and perfect information. While an uncertain, incomplete, imperfect dynamic environment is referred to as an Imperfect Evolutionary System (IES).

Our learning algorithm is based on a Swarm based approach, Particle Swarm Optimization algorithm (PSO). A comprehensive overview of PSO and its parameters is presented in [13]. Numerous studies have investigated the performance of PSO in DE. Hu and Eberhart presented re-randomization PSO (RPSO) to deal with DE [19]. In RPSO, some particles of the population are randomly relocated. Li and Dam [28] showed that restricting the information exchange and preventing convergence, increases the diversity of the population and gave better performance when dealing with DE. Janson and Middendorf proposed a hierarchical PSO [22], which showed improvements over the standard PSO for dynamic environments. Lung and Dumitresc [33] used an approach that relied on two different swarms. One was responsible for maintaining diversity while the other kept track of the optimum. Using the analogy of atomic particles, Blackwell and Bentley, presented their Charged Swarms approach [6]. Their swarm comprised of charged and neutral particles. A cloud of charged particles, that repel each other, orbits around the neutral nucleus. Blackwell et al. further extended this idea and presented a multi-swarm method [4,5,7,8]. Du and Li [10], divided the particles into two parts, Gaussian local search was applied to one part while the others patrol around them to track the optima. This approach was found to be less suitable for environments with more than two local optima.

Li and Yang [26], proposed that the swarm be broken into two types; a parent swarm, which maintains diversity and chooses promising areas for the search process and child swarms which explore these areas for optima [27]. The authors later introduced a clustering particle swarm optimizer, which partitions the main swarm into several sub-swarms, each searching for the optima [26]. Liu et al. proposed a Compound Particle Swarm Optimization approach [30] using composite particles. Later they used a scattering operator to maintain diversity in these swarms [31]. Hashemi and Meybodi presented their Cellular PSO in [17,18]. They partition the search space into cells, a group of particles then searches a cell for the optima using its personal best and best from the neighboring cells. Kamosi [24] applied the idea of hibernation to multi swarm. A parent swarm explores the search space and child swarms exploit promising areas found by the parent swarm. In the proposed model, whenever the search efforts of a child swarm for exploiting an area becomes unproductive, the child swarm hibernates. Results showed that the proposed algorithm outperforms other PSO algorithms, including similar particle swarm algorithms for dynamic environments like mQSO [13] and FMSO [27].

Evolutionary computation [12,15,44] has also been used to study adaptive intelligence. Evolutionary learning has resulted in generating behaviors and solutions which were, in most cases, unexpected or previously unknown. Fogel et al. [14] present advances in developing adaptive intelligence in computer games using EC techniques. Other studies that have investigated uncertain and changing environment in AI research are fuzzy logic [45,46], qualitative reasoning [41] and commonsense reasoning [35,36]. Bayesian networks were used in [37].

In real world environments a wide range of uncertainties have to be considered in the evolutionary process. To model uncertainty in our IES, we use noisy, time variant fitness functions [23]. Noise is an unavoidable part of most real life environments. Therefore in a dynamic environment modeled to represent real life scenarios, noise cannot be ignored. The effects of noise on the performance of evolutionary strategies has been discussed in [2,3]. In multi-objective optimization, two relevant works are [21,40]. In our imperfect environment, noise is modeled via the effects of different environmental factors ζ_i^T (e.g. the spawn location, number of creatures and terminals in the environment, etc.).

Time variant fitness functions are deterministic at any point in time but are dependent on time, t (4).

$$F(X) = f_t(X) \tag{4}$$

This dependence of the fitness function at time t, means that the optimum changes over time. Learning algorithms in such environments must be able to track the movement of the optimum over time t. Our IES is divided into different phases (detailed in Section 3). Each phase introduces or removes certain information into the environment and has its own corresponding fitness function.

For our experimentation we use our algorithm and other approaches to train Artificial Neural Network (ANN) in an IES. ANN based controllers have been used in the NERO game developed by Stanley et al. [39]. ANN controllers were also used by Yannakakis et al. [43] in a simulated world called *Flatland*. They investigate agents controlled by an ANN with a fixed architecture and trainable weights. One of the major contributions of their work is the mechanism of finding the fitness of agents by making them act out their strategies in the environment. We utilize the same technique for finding the fitness of our agents. Yannakakis and Hallam have also evolved ANN controllers for the games of Pac-Man and Dead End [42]. Yet another effort in this field has been made by Lucas for the game of Cellz [32]. Gaxiola et al. [16] have proposed a fuzzy based modification to the back-propogation algorithm for training of weights. A genetic algorithm based weight training approach has been proposed by [34,35].

3. Coping with imperfection

To deal with the uncertainty of the environment our learning algorithm focuses on automating the learning process. Human learning is a complex process involving countless sub-processes. However, for our approach we divide this learning into three abstract sub-processes. We call these the 3'Es of learning (experience, exploration, exploitation). These processes are fairly well-known in the AI community.

i. *Experience:* For every new problem faced, determine its relationship with the acquired knowledge and abilities, and reuse previous learning.
ii. *Exploration:* Formulate a general strategy to deal with the situation based on currently available information.
iii. *Exploitation:* Refine the current skill set based on its effectiveness to deal with the challenges at hand.

Fig. 2 presents a diagrammatic representation of the underlying processes in this continuous learning approach. The continuity of learning is based on these processes for the discovery and exploitation of information (knowledge) in the current environment and modifying behavior according to the surrounding environment. We use these processes to allow imperfect individuals to understand and adapt to their settings, thereby ensuring their survival.

We now present our continuous Particle Swarm Optimization algorithm:

```
1. Initialize all parameters (N, R, c1, c2, r1, r2, w)
2. Create population of Imperfect Individuals i
3. Divide the population into $S$ sub-swarms
4. For each $S$
  a. For each i $in$ $S$
    i. Initialize a random strategy
    Or
    Select strategy from archive
5. For T iterations or while termination criteria not met
  a. Evaluate fitness $f_i^T$
  b. Determine the best solution
  c. Adjust $e_p$
  d. Update agents strategy
  e. After $T_x$ force exploration
    i. Sort $G$ on the basis of $F_i^T$
    ii. Determine $S_w$
    iii. if it's not improving, reinitialize randomly or from pool
    else if $S_{w2}$ is not improving, reinitialize $S_{w2}$
    iv. After $T_e$ iterations
      i. Choose candidate for    experience pool
      ii. If (!full)
      Add $p_c$ to experience pool
    v. Invoke experience pool decay policy.
6. Go to step 3.
```

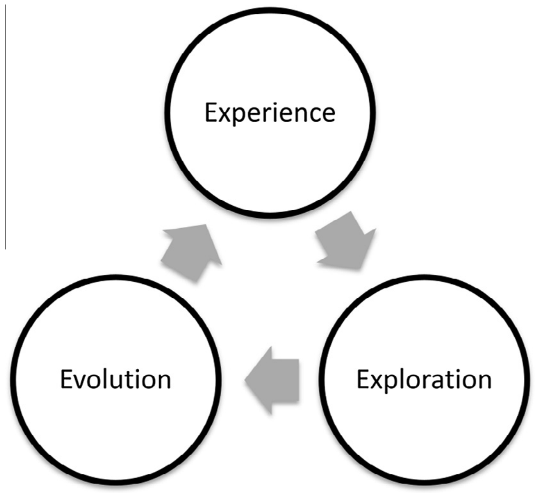


Fig. 2. Continuous learning in an IES.

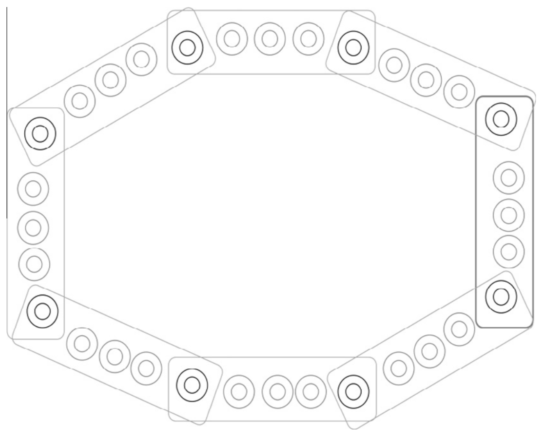


Fig. 3. Architecture used for defining and connecting sub-swarms.

Our algorithm is based on the PSO algorithm and uses a multi-population approach by dividing the main swarm into smaller sub-swarms. These sub-swarms are connected in a fixed-ring topology in which we have several completely connected sub-swarms, isolated from one another except through their corner particles. Each corner particle is completely connected with all the particles of two parent sub-swarms (Fig. 3). These particles act as a source of information transfer. Each sub-swarm is of a fixed size (for our experimentation we used eight sub-swarms, five particles per sub-swarm). Assignment to a sub-swarm is based on the index of a particle. Therefore particles cannot change their sub-swarms. This strategy for sub-swarms allows the particles of a sub-swarm to reside anywhere in the search space instead of in close locality to other members of their sub-swarms. This has two benefits:

i. Each individual shares its perspective of the surroundings with other members of the sub-swarm.
ii. Each sub-swarm acts independently of other sub-swarms; poorly performing sub-swarms do not affect others, while better strategies are disseminated within the population via the connecting particles.

By employing the 3'Es of learning, our learning algorithm maintains diversity within these sub-swarms, and retains the lessons it learns. Change is detected by monitoring the performance of the individuals and the algorithm decides when it has to focus on exploration and when exploitation is required. The entire learning process takes place without relying on any external guidance other than the stimulus provided by the evolutionary environment. The algorithm starts by initializing all parameters necessary for the evolutionary process and an initial population of imperfect individuals is created. The initial population is then divided into smaller sub-swarms. These sub-swarms are of fixed size and are formed on the basis of a particle's index number. The structure of our sub-swarms and their interconnections is shown in Fig. 3.

Individuals within these sub-swarms then develop strategies for dealing with the environment. Strategies are formulated using information provided by the environment, e.g. input sensors, primary and secondary abilities etc. Depending upon the number of inputs (sensors) chosen by the individual, they can formulate unique strategies allowing them to perform different tasks with different skill levels. In other words what the agents can do depends upon what it chooses from what is provided by the environment. At this initial stage, these strategies may be random or chosen from the experience pool.

The appropriateness of an individual's strategy is then determined by a fitness function defined by the environment. Our focus, in this research, is to develop a learning approach which has the ability to automatically detect environment changes, and to adapt by formulating suitable strategies. For this reason, the fitness function is never disclosed to the learning agents. The learning algorithm only receives the fitness value of its current strategy and uses this evaluation to learn what the demands of the unknown fitness function are and if they have changed or not. No other information about the environment is explicitly shared with the individuals or the learning algorithm.

Based on the fitness values, the best strategy is determined. The population is motivated to adapt strategies similar to the best strategies using an exploitation parameter e_p (5). Ideally, an individual doing well should not jump far from its current location in the search space and should explore its current locality for a better solution to the problem. An agent performing badly should jump away from its current location to look elsewhere for a better solution. This parameter stops the population from developing random strategies and encourages them to develop strategies similar to the ones that are currently successful. Setting this parameter too strictly was found to adversely affect the diversity of the population.

$$e_p = \max\left(\left(\beta - \left(\frac{f_s^T}{f_P^T}\right)\right), 1\right) \quad (5)$$

The threshold value β is set to a constant ($\beta = 1.05$), f_s^T is the fitness value of the best particle in the sub-swarm S at time T and f_P^T is the fitness of the best particles from the entire population P at time T. Different sub-swarms may have different e_p, depending on the fitness of their particles.

An essential part of adaptive learning is automated change detection. To detect change in the environment, the learning algorithm monitors its own evolutionary performance. After every T_x iterations it diversifies its population of particles by re-initialization of the worst performing sub-swarms. This selective re-initialization strategy allows the learning algorithm to utilize poorly performing particles for exploring new opportunities and changes in the environment. In order to choose a sub-swarm for re-initialization, we use accumulated fitness value F_s^T (6), to determine the worst sub-swarm.

$$F_s^T = \sum f_p^T \tag{6}$$

F_s^T is calculated by the sum of fitness f of all particles p belonging to the sub-swarm S over a period of T iterations. This allows monitoring of a sub-swarm's performance over a number of iterations. Any sub-swarm that is currently improving itself is allowed to improve. When a sub-swarm has stopped improving and is currently the worst sub-swarm then it becomes an ideal candidate for exploration. A sub-swarm that had high fitness in a previous state of the environment and now relies on strategies that have become obsolete is forced to abandon its outdated methods and explore new opportunities. Only the worst two sub-swarms are used in the exploration process and if both of these sub-swarms are improving then no candidate is selected for further exploration. This limit avoids the algorithm from becoming a random search and to focus on improving current strategies via evolution. Tables 1 and 2 present the parameter values involved in this continuous learning process. Due to the imperfect nature of the environment, inputs and outputs (or any other parameter) can change at any time, without warning. In case a certain input is no longer available in the environment, individuals relying on that input will have to abandon their strategies and formulate new ones. In this manner these individuals build a dynamic bond with their surrounding environment and constantly search for any changes that may have occurred. At each step their actions are evaluated. This evaluation guides the agents as to whether they have selected the right strategy or not.

After every T_e a candidate is chosen to be added to the archive. Every T_e iterations, the best strategy found by the population is added into an experience pool. This experience pool serves as a historical archive for social evolution. An individual can adopt strategies from this experience pool. Since each individual has its own perspective of the environment based on their locality, this pool can be further subdivided into smaller pools where each pool only stores the best strategies from a selected portion of the environment. We do not allow duplication in the pool because it was experimentally found to be less desirable. While adding to the socio-experience pool two different strategies were tested.

i. After T_e iterations, sort individuals on the basis of fitness and pick the best one if it is above a certain threshold.
ii. Add the best solution to the pool after T_e iterations regardless of its fitness values.

The termination criteria for the algorithm could be a predefined number of iterations or a fitness threshold value.

Finally we, increment the decay value. This decay value is used to determine how old a strategy is (the oldest strategy will have the highest decay value). If the pool is full then, for removal of strategies from the socio-experience pool, we reference this decay value.

Table 1
Parameters used for normalization in fitness functions.

mt^T	250
md^T	125
me^T	100

Table 2
Learning algorithm parameters.

P	20
C_1, C_2	1
w	1
r_1, r_2	0–1
$V_{\max}$	0.05–1
T	20
T_e	20
T_x	20
r	5
E	300

4. Wonderland – an imperfect evolutionary environment

We now detail the imperfect De we developed to model imperfection and uncertainty of information. DEs have been classified into different types based on their characteristics and numerous works have presented different categorizations. DE can be categorized on the basis of how often change occurs, the magnitude of change and the movement of optima. Classification systems based on the trajectory of optima and direction of change and have been used by [1,11,19]. For our IES, we use the classification proposed by [9]. It provides four characteristics of change:

i. Predictability of change, i.e. the correlation between changes.
ii. Frequency of occurrence of change.
iii. Magnitude of change.
iv. Repetitiveness of change, whether optima returns to its previous location or close to it.

We designed different challenges for the learning individuals based on these criteria. These challenges are modeled used different phases in our environment. We call our IES, Wonderland and it is a predator–prey environment, similar to the one used by [43] and [32]. It is a 2D world with no boundaries (agents crossing over the edge reappear on the other side) and consists of two types of entities; individuals (learning) and artifacts (non-learning). An individuals' strategy is represented using ANNs. Each strategy is formulated using inputs available in the environment. A change in the environment is represented as an environmental phase.

We now explain different kinds of entities and phases used in our environment.

4.1. Entities in the environment

4.1.1. Humanoids

Humanoids are the basic evolvable individuals within the environment. These humanoids are placed in the environment without any *a priori* information. They must formulate a strategy by choosing a set of inputs. The environment requires humanoids to learn object-avoidance and target-achievement skills without explicit information about which objects are to be avoided and which objects are to be targeted.

Each of the individuals decide to use a set of ANN inputs $p^T_{x_a}$ available from the set of inputs. $p^T_{1_{x_a}}$ is defined as those inputs used by the individual which were available from the set of inputs x allowed for species a, at time T (7).

$$p^T_{1_{x_a}} = \{x^T_{1_a}, x^T_{2_a}, \ldots, x^T_{n_a}\} \tag{7}$$

For our current series of experimentation we restrict the ANN architecture as described in Section 4. This architecture was experimentally derived and is similar to the one used by Yannakakis et al. [43].

4.1.2. Terminals

These are non-evolving stationary points on the terrain. Once a humanoid reaches (i.e. a humanoid reaches the pixel location of the terminal) the terminal, it is considered to be captured and a new terminal point is spawned at a random location.

4.1.3. Creatures

These are non-evolving agents in the environment. Their behavior is scripted. Creatures attack or avoid humanoids based on different phases of the environment. Due to their dual nature they play a unique role in the environment being both prey and predator at different times. Due to their scripted nature they always use the shortest distance between them and the closest agent. The number of pixels visible to a creature is always smaller than the number of pixels visible to the agents. This is done to give both parties a fair chance.

4.2. Phases in the environment

The following environmental phases are used in the experimentation presented in this paper.

4.2.1. Acquisition

In this phase the humanoids learn to capture as many terminal points as they can within a limited time period. The fitness function is based on how many points have been captured (8).

$$F^T_i = \min\left(1, \frac{t^T_i}{mt^T}\right) \tag{8}$$

where F^T_i is the fitness of the ith humanoid and is calculated as the minimum of 1 and the total number of terminal points captured by the humanoid t^T_i divided by the maximum terminals available for capturing, mt^T, at time T.

4.2.2. Survival

In this phase a new entity called *Creatures* is introduced into the environment. These creatures hunt for humanoids. Humanoids cannot fight back and must choose to run away. The addition of creatures requires a drastic change in humanoids' survival strategy and individuals will have to learn to avoid death by a creature (9).

$$F_i^T = \max\left(0, \frac{md^T - d_i^T}{md^T}\right) \tag{9}$$

where md^T is the maximum number of time humanoids are allowed to be killed by the creature and d_i^T is the number of times humanoid i was killed by the creatures.

4.2.3. Hunting

This phase is the opposite of the *Survival* phase. Humanoids are allowed to fight back by learning to capture creatures. Capturing a creature requires teamwork. Creatures will try to avoid humanoids in this phase while two or more humanoids must trap a creature to capture it (i.e. two or more individuals must come within a Euclidian distance of 1 pixel to the creature). Fitness is determined by the number of creatures captured (10).

$$F_i^T = \min\left(1, \frac{e_i^T}{me^T}\right) \tag{10}$$

where me^T is the maximum number of creature captures allowed and e_i^T is the number creatures captured by the humanoid. Whenever two or more individuals cooperate to capture a creature this number is updated for all the humanoids who participated in the capture.

Combinations of these objectives can be used to develop more complicated fitness functions. One example of such a complex learning activity could be (11).

$$F_i^T = \frac{fe_i^T + fc_i^T + ft_i^T}{n} \tag{11}$$

Here, fitness is the summation of total creatures captured fe_i^T, number of collisions made fc_i^T (i.e. number of times an individual bumped into other members of its specie) and number of terminals captured ft_i^T by the humanoid, divided by n (number of factors involved in fitness calculation).

The difficulty of tasks varies between different phases. In order to establish a common ground for comparison all fitness values are normalized, allowing a fair comparison among learning from different phases or tasks. The maximum values used in the normalization of fitness were determined experimentally using individuals whose behavior was scripted to meet the demands of the environment. Since the location of the optima is unknown for these problems, the values from scripted agents were used to compare the quality of results from learning algorithms (Table 1).

There are two levels of change occurring in the environment. The minor changes (movement of creatures, relocation of terminals, and movement of humanoids) are less severe in magnitude but occur very frequently (i.e. they change in every iteration). Major changes are the transition of one phase to another. Successful learning algorithms must be able to generate robust solutions that deal with the chaotic minor changes, yet are adaptive enough to deal with major changes.

5. Experimental setup

In an imperfect environment, the performance of an individual i depends on two factors, its learning ability δ_i^T, and impact of environmental factors ζ_i^T (e.g. the spawn location, number of creatures and terminals in the environment, etc.), at time T. The effects of these external factors cause a noisy fitness evaluation i.e. the same strategy can have different fitness values on subsequent evaluations. The time varying phase changes and the influence of ζ_i^T gives rise to a new problem. How do we determine if the humanoids have learned anything about their environment?

To answer this, we opted for a competitive co-evolutionary approach. The best strategies found by each algorithm compete against each other. For each experiment multiple runs (at least 10) were made to validate the conclusion of the results. Figures show the average values of these runs. Artificial Neural Networks were used to interface the learning individuals (or particles) with the environment. Each learning individual tunes the weights of an ANN which represents the strategy for that individual. Using a common ANN architecture for all algorithms, we test only on their change detection and adaptive learning abilities, allowing for a fair comparison. The ANN architecture used for experimentation comprises 12 input neurons, 1 hidden layer with 5 neurons and 2 output neurons. Inputs comprise of distance and angles of the z ($z = 2$) closest terminals, humanoids and creatures. A sigmoid activation function is used. The two output neurons give the step size and the direction (or angle) in which to move.

To deal with noisy fitness functions, the expected fitness function is often approximated over a number of random samples [23]. In our setup noise is generated by the effects of environmental parameters ζ_i^T. To minimize these, for each of these particles we create N clones. A clone represents the strategy formulated by its parent particle. Each clone takes E steps in the environment, and its movement and interactions are monitored. The fitness of the particle (or the Individual) is then calculated based on these interactions (using the formulae mentioned in Section III B). The values of all the clones are summed to represent the value of the parent particle e.g. the number of terminals reached by a particle during an iteration is the summation of the number of terminals reached by all its clones in E steps of that iteration. This minimizes the effect of random elements of the environment such as; agent spawn locations, proximity to creatures, terminal placement etc., in fitness calculation. The algorithms used in our experimentation are HMSO [24], DMS PSO [29], and an *lbest* version of RPSO [13,20]. The *lbest* RPSO (we will call it RL-PSO) was given an unfair advantage over the others as it was informed of all the phase changes whenever they occurred and told which factors should be ignored and which ones considered. In this way, RL-PSO mimics the behavior of an ideal change detection and adaption strategy.

In all experiments, the same parameter values are used for all algorithms. Each algorithm faces the same phase change at the same iteration. This removes all other factors and focuses on the ability of each algorithm to deal with the change in the IES allowing for a fair comparison.

6. Experimental results and discussion

First, we test the learning ability of the algorithms in a perfect environment and gradually move towards an imperfect one (i.e. increase the complexity of the problem).

6.1. Perfect environment

To test the learning ability of all algorithms in a stable environment, they were allowed to train humanoids in a stable, unchanging environment. We used the environment states along with their fitness function mentioned in Section 3. Tables 4–9 give details of the performance of each algorithm in different scenarios.

Fig. 4 presents the learning curve of each approach for this learning activity showing the terminal acquired (Fig. 4a), number of deaths (Fig. 4b), and successful creature captures (Fig. 4c). The best solutions found competed against random solutions and they outperformed the random ones, proving their ability to learn in a perfect environment. Fig. 5, shows how successful each strategy was compared to others in a perfect environment. It is clear from these results, that the best results were achieved by RL-PSO, because it was informed about all changes. Our CL-PSO came in second, with results that are close to those of RL-PSO. This shows our learning approach can successfully learn in a perfect environment. Results from DMS and HMSO were far below our strategy. These algorithms were confused by the constant chaotic changes and the effects of environmental parameters. Still these algorithms performed better than random search. The jagged learning curve of our CL-PSO is the result of constant exploration of new strategies and reinitializing of the worst sub-swarm. The learning curve of other approaches is smoother because in a perfect environment the best strategies are only improved and do not become obsolete.

6.2. Coping with imperfection

Next we test the learning ability of these algorithms in an imperfect and changing environment. For this test, each algorithm was allowed to learn about the environment for a number of iterations after which a phase change was introduced. We started with phase 1 then transitioned the environment into phase 2 and then phase 3 (each phase change was introduced after every 1000 iterations). Details of these phases are in Section 4.2.

Table 3
Switching between environment states for socio-experience comparison.

Iterations	Environment phase
0–300	Survival phase
300–600	Acquisition phase
600–900	Survival phase
900–1200	Hunting phase
1200–1500	Survival phase
1500–1800	Acquisition phase
1800–2100	Hunting phase

Table 4
Fitness values in perfect environment – acquisition phase.

	DMS	HMSO	CL-PSO	RL-PSO
Max	0.236	0.476	0.615	0.713
Avg.	0.217	0.281	0.553	0.658
Min	0.070	0.136	0.068	0.058

Table 5
Fitness values in perfect environment – survival phase.

	DMS	HMSO	CL-PSO	RL-PSO
Max	0.280	0.440	0.560	0.687
Avg	0.137	0.211	0.396	0.529
Min	0.000	0.000	0.000	0.000

Table 6
Fitness values in perfect environment – hunting phase.

	DMS	HMSO	CL-PSO	RL-PSO
Max	0.090	0.120	0.156	0.175
Avg.	0.076	0.082	0.141	0.163
Min	0.001	0.000	0.025	0.014

Table 7
Performance in acquisition phase.

	DMS	HMSO	CL-PSO	RL-PSO
Max	59.00	104.13	153.75	178.25
Avg.	54.17	62.12	138.16	164.62
Min	17.50	17.00	17.00	14.38

Table 8
Performance in survival phase.

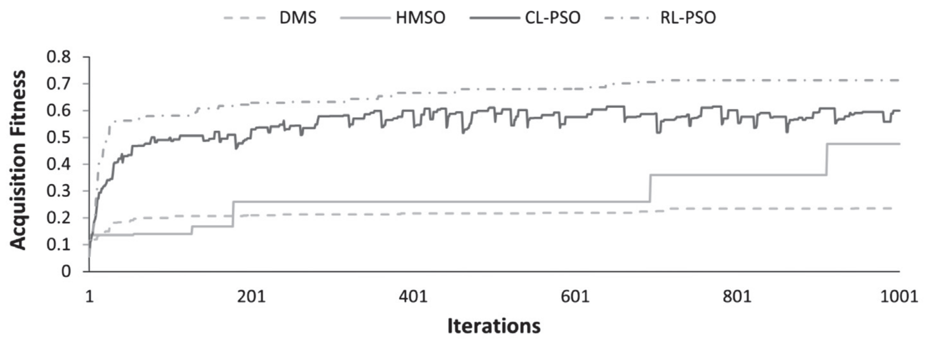
	DMS	HMSO	CL-PSO	RL-PSO
Max	630.88	620.50	988.38	1110.88
Avg.	466.63	323.14	200.62	75.04
Min	90.00	69.25	55.00	39.13

Table 9
Performance in hunting phase.

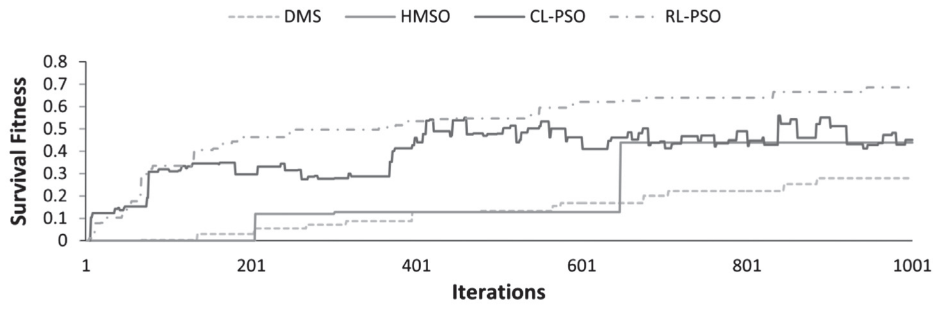
	DMS	HMSO	CL-PSO	RL-PSO
Max	9.00	11.71	15.63	17.50
Avg.	7.64	8.08	14.14	16.27
Min	0.13	0.17	2.50	1.38

The results of this experiment are shown in Fig. 6. Since in the first two phases of the environment the humanoids are not given any information about how to capture the creatures hence for those two stages the number of captures remains at zero. Table 10 details the average values of the best individuals determined by each algorithm on multiple runs. Table shows that all algorithms started to learn terminal acquisition, however, HMSO and DMSPSO were unable to deal with the incomplete and noisy nature of the environment and their resulted in poorer performance than the other two algorithms which had a better change detection scheme. Since human guidance and explicit information about the surrounding environment was available to RL-PSO its performance is better than all algorithms. Therefore the results of RL-PSO should be considered as ideal i.e. something that a human would come close to achieving. Our continuous learning algorithm came in close second, showing its ability to deal with the imperfectness of the environment without any explicit information about the surroundings. Our algorithm detects the current state of the environment based solely on the information provided by its surrounding and Fig. 6(a) shows how the learning for the traditional algorithms halts after the introduction of change and their failure to respond to changes in the environment. In contrast, the learning curve of RL-PSO and our PSO approach show that they respond to the changes in their environment. Whenever the environment changes, learning individuals reflect these changes and adapts by formulating suitable strategies to ensure survival.

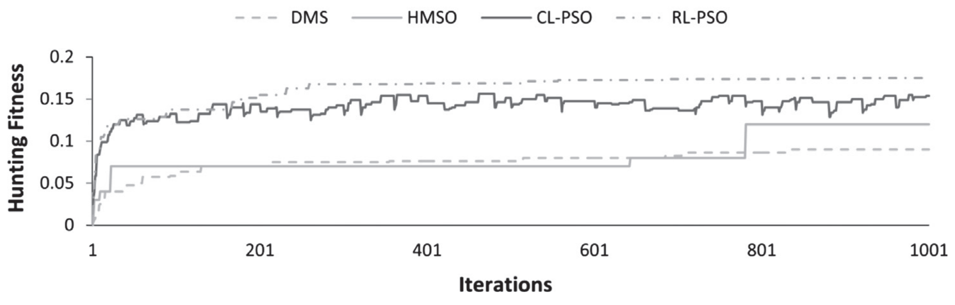
In an imperfect environment, with every phase change, the fitness of the best individuals should decrease because the previously best strategies have now become obsolete. This must be automatically detected by the learning algorithm to match the changes that have occurred in its surrounding environment. If the algorithm successfully detects the changes then it must reevaluate its best strategies to see if their previous fitness values are valid or not. Because RL-PSO was explicitly informed about these changes, its curve perfectly reflects the changes of the environment. Our automated change detecting



(a) Learning curve in Acquisition Phase



(b) Learning curve in Survival Phase



(c) Learning curve in Hunting Phase

Fig. 4. Learning in a perfect environment.

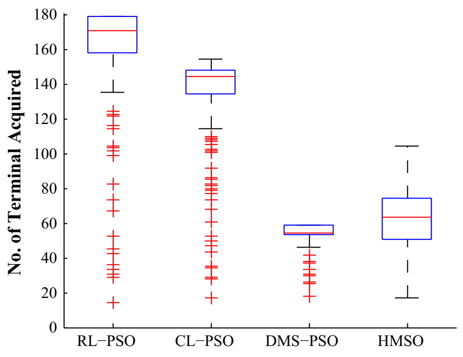


Fig. 5a. Performance in acquisition phase.

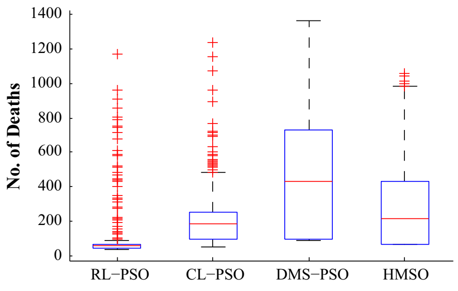


Fig. 5b. Performance in survival phase.

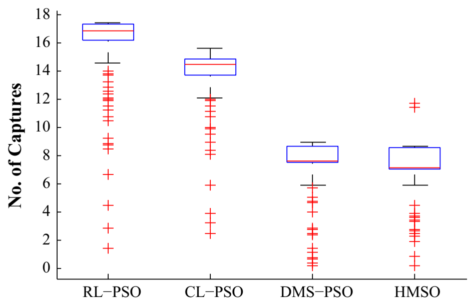


Fig. 5c. Performance in hunting phase.

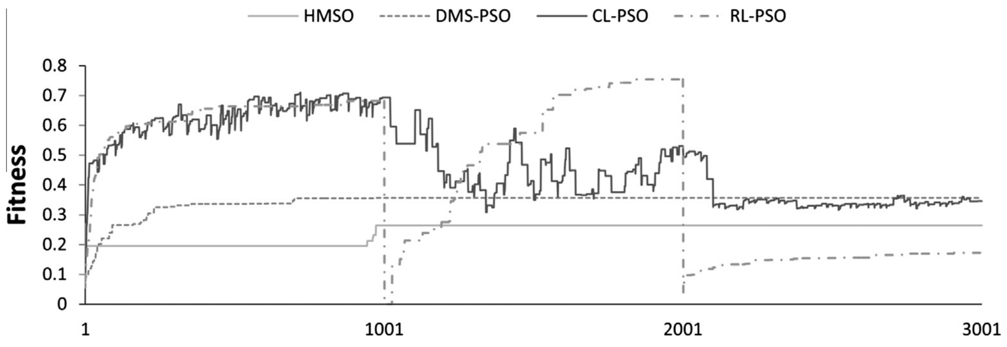


Fig. 6. Learning in an imperfect evolutionary environment. Phase change was introduced every 1000 iterations.

Table 10
Performance in imperfect environment (avg. of best individuals).

		DMS	HMSO	CL-PSO	RL-PSO
Phase 1	Terminals	79.65	61.07	156.40	157.04
	Deaths	1070.55	567.90	809.03	709.57
	Captures	0.00	0.00	0.00	0.00
Phase 2	Terminals	89.25	76.25	40.29	13.14
	Deaths	998.25	523.07	187.82	123.49
	Captures	0.00	0.00	0.00	0.00
Phase 3	Terminals	89.25	47.78	40.47	24.28
	Deaths	998.25	332.76	140.45	22.68
	Captures	0.00	1.80	7.20	15.16

PSO follows closely, whereas the other algorithms do not consider the phase changes and maintain a smooth learning line. The smoothness of their fitness curve shows that the best values, once achieved, are never updated to match the phase changes in the environment.

6.3. Dealing with repetitive chaotic imperfect environment

In this experiment, the phase changes occur even more frequently (Table 3). The environment reverts back to a previous state to test the ability of algorithms to deal with repetitions in a chaotic imperfect environment. The fitness learning curve is shown in Fig. 7. Results show that our CL-PSO responds well to the change in the environment (as evident by the rise and fall in the average fitness value). Other learning approaches fail to respond to the chaos in their surroundings and instead only focus on improving their fitness for the first phase. They never adapt to the changes and their smooth learning curve shows that they fail to detect when their best strategies become obsolete.

6.4. Adding a new dimension to learning

In all the tasks, the humanoids had to learn but they only had to focus on one thing at a time. This fact is highlighted by the statistics shown in Fig. 8. The statistics represent the accumulated values of the most successful strategies found in 3000 simulations. It is clear from Fig. 8 that while a strategy performs well in its own area of specialty, its performance in other dimensions is usually poor. In an imperfect environment, it is possible that new dimensions to the problem may be added (or subtracted) from the current learning ability at any time without any forewarning. Any learning algorithm that deals with imperfection must be able to deal with this situation.

To test this ability we started the learning with an acquisition phase for 1000 iterations, the fitness evaluated by Eq. (5). After the initial 1000 iterations, a new dimension to the problem was introduced as in Eq. (12).

$$F_i^T = \frac{\left(\min\left(1, \frac{t_i^T}{mt^T}\right) + \max\left(0, \frac{md^T - d_i^T}{md^T}\right)\right)}{2} \tag{12}$$

Here F_i^T is the fitness of the ith humanoid which is the average of two entities. t_i^T is the number terminal points captured by the humanoid mt^T the maximum terminals allowed for capturing at time T. md^T is max number of deaths allowed and d_i^T is the number of times humanoid i was died.

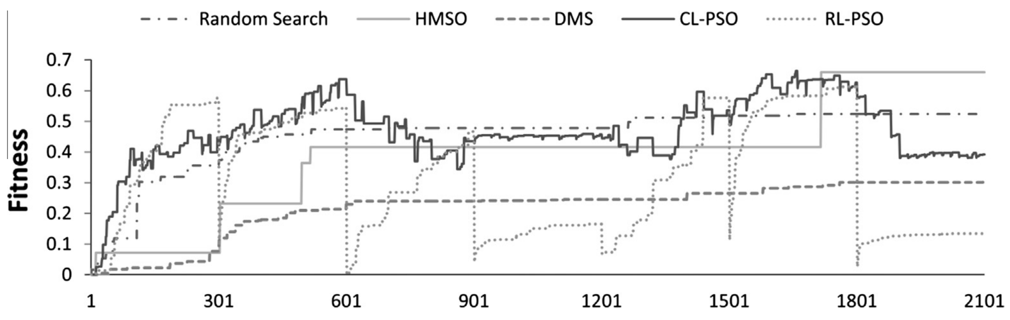


Fig. 7. Learning in a chaotic imperfect evolutionary environment. Phase change was introduced every 300 iterations. Random search has also been plotted to show how a completely random search for best strategies behaves in a chaotic environment.

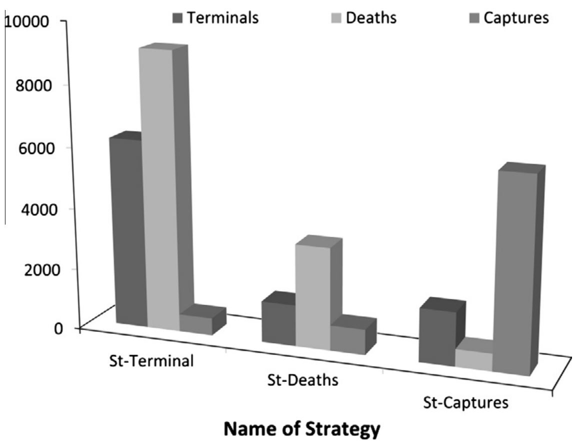


Fig. 8. Comparing performance of different strategies. X-axis shows name of strategy, St-Terminal is the best strategy found for acquisition phase, St-Deaths for survival phase and St-Captures for hunting phase. Y-axis shows the number of terminals acquired, deaths and hunts made by a strategy.

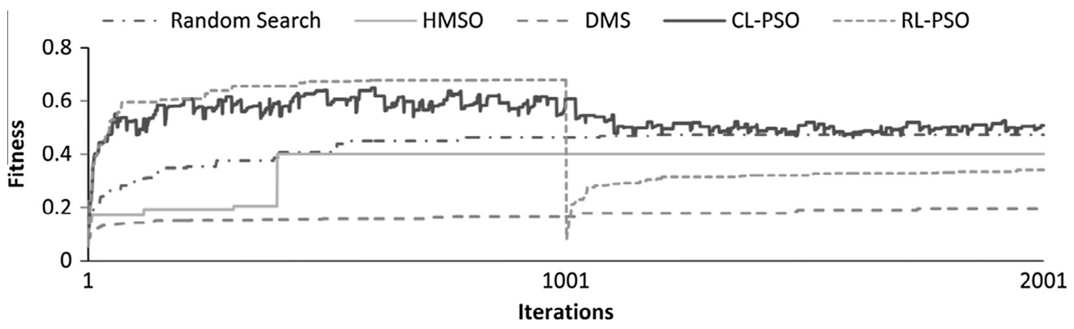


Fig. 9. Adding new dimensions to learning. Fitness function was modified after 1000 iterations.

The results of this experiment are shown in Fig. 9. In the first part of the experiment (0–1000 iterations) humanoids ignore their deaths. In the latter half of the experiments (1000–2000 iterations) they now have to learn to maximize the number of their terminal acquisitions and also minimize their number of deaths. Whereas in the first phase the focus is on achieving a target, ignoring deaths, in the second phase learning algorithms have to balance the two factors.

The new state of the environment is more complex than the previous one. Terminals must be captured while running away from the creatures, doing so will decrease the numbers of terminals captured. RL-PSO and our CL-PSO both respond to this change in the environment and try to balance the two factors, while DMS and HMSO ignore this phase change and instead focus on terminal capturing, which is why their fitness more or less remains the same after the change.

7. Conclusion

An imperfect environment constantly presents new challenges to its inhabitants. Individuals residing in such an environment are called imperfect individuals. They must realize the imperfection of their environment and adapt accordingly. In this paper, we have presented a continuously learning algorithm that enables individuals to learn and survive in an imperfect environment. These individuals form a dynamic relationship with their surroundings. Whenever a change occurs in the environment these individuals detect it and modify their strategies, thereby improving their chances of survival. Their ability to learn does not depend upon any human intervention. These individuals form a diverse and active society where each member contributes to the improvement of the collective. Poorly performing members of the society are utilized in exploration while better members are used in exploitation of the surroundings.

We have developed an environment to test the learning ability of our algorithm. The environment is uncertain and changing in nature. This environment has different states and a change in a state is transmitted via a fitness function. We believe a similar setup can be used as a test-bed for future research in imperfect environments. These environments represent scenarios similar to the ones we face in our everyday lives, where an individual has to deal with different challenges, some of which are not known in advance. Our aim was to develop an environment that can be extended to different problems and domains. Our IES can be used by other researchers to investigate the performance and ability of different learning algorithms when presented with diverse challenges of imperfect information.

Our results show that the self-motivated learning ability of our continuous learning approach is successful in detection and adaptation to the demands of the environment. Our agents not only adapted to survive in their new environment but also exploited any weakness present in their surroundings. We do not claim that the learning ability of these agents is optimal (or that the strategies they evolved are optimal strategies). It was, however, demonstrated experimentally that the agents efficiently learned to deal with changes in the environment within the given time. We conclude that this algorithm demonstrates an adaptive intelligence based behavior, which can be applied to other complex problems.

We believe that IES are the logical extension of current research in dynamic algorithms. These environments can be used to model real life scenarios. These environments are modeled to accommodate incompleteness of information, thereby allowing new information to be added at any stage of evolution. In this regard the environment becomes a part of the evolutionary process. This nature of the environment to support different kinds of information can be scaled to test different problems of various domains, e.g. testing Game AI algorithms, developing AI for exploration of unknown environments, developing testing scenarios for training of employees in various capacities, etc.

Our learning continuous learning approach has shown its ability to handle incompleteness of information. This learning approach is aimed at developing AI that mimics human behavior. Such learning strategies can especially be applied to video games where scripted behaviors become predictable. A learning approach based on our research would allow the game to adapt and behave according to the actions of the players. This work could also be extended to deal with cheaters who exhibit behavior that cannot be predicted in advance using traditional scripting based methodologies. Example uses of such learning agents would improve the replay ability of games by detecting cheaters, or generating behaviors that adapt to the choices made by a human player and coming up with strategies that would challenge and involve him/her.

There are questions that still need to be explored. For example, can we apply the algorithm to problems in which even the structure or representation of the problem is unknown. In such a problem even the particle representation would be dependent upon the individuals of the environment. How would they handle this? It would also be interesting to have continuously learning species compete against one another for survival. We are currently working on the integration of better neuro-evolution strategy to the framework so that it can attempt to optimize its solutions as per the requirements. Better algorithms for the addition and removal from the experience pool are also being investigated. Application of this self-motivated, independent, multi-dimensional continuous learning framework upon other practical problems would be interesting.

Acknowledgement

The authors would like to thank the Higher Education Commission of Pakistan for funding this research work. We would also like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

References

[1] P. Angeline, Tracking extrema in dynamic environments, in: Evolutionary Programming VI, Lecture Notes in Computer Science, vol. 1213, Springer, Berlin, 1997, pp. 335–345.
[2] D. Arnold, Noisy Optimization in Evolution Strategies, Kluwer, Norwell, MA, 2002.
[3] H. Beyer, Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice, Comput. Methods Appl. Mech. Eng. 186 (2000) 239–267.
[4] T. Blackwell, J. Branke, Multi-swarm optimization in dynamic environments, in: Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 3005, Springer, Berlin/Heidelberg, 2004, pp. 489–500.
[5] T. Blackwell, Swarms in dynamic environments, in: Genetic and Evolutionary Computation — GECCO 2003, Lecture Notes in Computer Science, vol. 2723, 2003, pp. 1–12.
[6] T. Blackwell, P. Bentley, Dynamic search with charged swarms, in: Genetic and Evolutionary Computation Conference, New York, NY, USA, 2002, pp. 19–26.
[7] T. Blackwell, J. Branke, Multiswarms, exclusion, and anti-convergence in dynamic environments, IEEE Trans. Evol. Comput. 10 (4) (2006) 459–472.
[8] T. Blackwell, J. Branke, X. Li, Particle swarms for dynamic optimization problems, in: Swarm Intelligence, Natural Computing Series, vol. Part II, 2008, pp. 193–217.
[9] J. Branke, Evolutionary Optimization in Dynamic Environments, Kluwer, 2002.
[10] W. Du, B. Li, Multi-strategy ensemble particle swarm optimization for dynamic optimization, Inf. Sci. 178 (15) (2008) 3096–3109.
[11] R. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 1, 2001, pp. 94–100.
[12] A. Eiben, J. Smith, Introduction to Evolutionary Computation (Natural Computing Series), Springer, New York, 2003.
[13] A.P. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Willey & Sons, 2005.
[14] D. Fogel, A. Blair, R. Miikkulainen (Eds.), Special issue on Evolutionary and Games, IEEE Trans. Evol. Comput., vol. 9(6), December 2005, pp. 537–539.
[15] D. Fogel, Evolutionary Computation: Toward A New Philosophy of Machine Intelligence, second ed., IEEE Press, Piscataway, NJ, 2000.
[16] F. Gaxiola, P. Melin, F. Valdez, O. Castillo, Interval type-2 fuzzy weight adjustment for back propagation neural networks with application in time series prediction, Inf. Sci. 260 (2014) 1–14.
[17] A. Hashemi, M. Meybodi, A multi-role cellular PSO for dynamic environments, in: 14th International CSI Computer Conference, Tehran, Iran, 2009, pp. 412–417.
[18] A. Hashemi, M. Meybodi, Cellular PSO: a PSO for dynamic environments, Adv. Comput. Intell. Lect. Notes Comput. Sci. 5821 (2009) (2009) 422–433.
[19] X. Hu, R. Eberhart, Tracking dynamic systems with PSO: where's the cheese, in: Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN, USA, 2001, pp. 80–83.
[20] X. Hu, R.C. Eberhart, Adaptive particle swarm optimization: detection and response to dynamic systems, in: IEEE Congress on Evolutionary Computation, Honolulu, HI, USA, 2002, pp. 1666–1670.
[21] E. Hughes et al, Evolutionary multiobjective ranking with uncertainty and noise, in: E. Zitzler et al. (Eds.), Evolutionary Multi-Criterion Optimization, LNCS, vol. 1993, Springer-Verlag, Berlin, Germany, 2001, pp. 329–343.
[22] S. Janson, M. Middendorf, A hierarchical particle swarm optimizer for noisy and dynamic environments, Genet. Programm. Evol. Mach. 7 (4) (2006) 329–354.
[23] Y. Jin, J. Blanke, Evolutionary optimization in uncertain environments—a survey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–319.
[24] M. Kamosi, A. Hashemi, M. Meybodi, A hibernating multi-swarm optimization algorithm for dynamic environments, in: Second World Congress on Nature and Biologically Inspired Computing (NaBIC), Tehran, Iran, 2010, pp. 363–369.
[25] G. Kendall, Y. Su, Imperfect evolutionary systems, IEEE Trans. Evol. Comput. 11 (3) (2007) 294–307.
[26] C. Li, S. Yang, A clustering particle swarm optimizer for dynamic optimization, in: IEEE Congress on Evolutionary Computation, 2009, pp. 439–446.
[27] C. Li, S. Yang, Fast multi-swarm optimization for dynamic optimization problems, in: Fourth International Conference on Natural Computation, Jinan, Shandong, China, 2008, pp. 624–628.
[28] X. Li, K. Dam, Comparing particle swarms for tracking extrema in dynamic environments, in: IEEE Congress on Evolutionary Computation, Canberra, Australia, 2003, pp. 1772–1779.
[29] J. Liang, Dynamic multi-swarm particle swarm optimizer, in: IEEE Swarm Intelligence Symposium, 2005, pp. 124–129.
[30] L. Liu, D. Wang, S. Yang, Compound particle swarm optimization in dynamic environments, in: Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 4974, 2008, pp. 616–625.
[31] L. Liu, S. Yang, D. Wang, Particle swarm optimization with composite particles in dynamic environments, IEEE Trans. Syst. Man Cybern., Part B: Cybern. PP (99) (2010) 1–15.
[32] S. Lucas, Cellz: a simple dynamical game for testing evolutionary algorithms, in: IEEE Congress on Evolutionary Computation, 2004, pp. 1007–1017.
[33] R. Lung, D. Dumitrescu, A collaborative model for tracking optima in dynamic environments, in: IEEE Congress on Evolutionary Computation, Singapore, 2007, pp. 564–567.
[34] P. Melin, V. Herrera, D. Romero, F. Valdez, O. Castillo, Genetic optimization of neural networks for person recognition based on the Iris, Telkomnika 10 (2) (2012) 309–320.
[35] M. Minsky, The Society of Mind, Simon & Schuster, New York, 1986.
[36] A. Newell, H. Simon, GPS, a program that simulates human thought, in: E.A. Feigenbaum, J. Feldman (Eds.), Computers and Thought, McGraw-Hill, New York, 1963, pp. 279–293.
[37] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Mateo, CA, 1988.
[38] D. Poole, A. Mackworth, R. Goebel, Computational Intelligence: A Logical Approach, Oxford University Press, New York, 1998.
[39] K. Stanley, B. Bryant, R. Miikkulainen, Real-time neuroevolution in the NERO video game, IEEE Trans. Evol. Comput. 9 (6) (2005) 653–668.

[40] J. Teich, Pareto-front exploration with uncertain objectives, in: E. Zitzler et al. (Eds.), Evolutionary Multi-Criterion Optimization, LNCS, vol. 1993, Springer-Verlag, Berlin, Germany, 2001, pp. 314–328.
[41] D. Weld, J. Kleer, Readings in Qualitative Reasoning About Physical Systems, Morgan Kaufmann, San Mateo, CA, 1988.
[42] G. Yannakakis, J. Hallam, A generic approach for obtaining higher entertainment in predator/prey computer games, J. Game Develop. 1 (3) (2005) 23–50.
[43] G. Yannakakis, J. Levine, J. Hallam, Emerging cooperation with minimal effort: rewarding over mimicking, IEEE Trans. Evol. Comput. 11 (3) (2007) 382–396.
[44] X. Yao, Evolutionary Computation—Theory and Applications, World Scientific, Singapore, 1999.
[45] L. Zadeh, Fuzzy sets as a basis for a theory of possibility, Fuzzy Sets Syst. 1 (1978) 3–28.
[46] L. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.