Analysing Fitness Landscape Changes in Evolutionary Robots *

Renato Tinós
Department of Computing and Mathematics, FFCLRP
University of São Paulo
Av. Bandeirantes, 3900
Ribeirão Preto, SP, 14040-901, Brazil
rtinos@ffclrp.usp.br

ABSTRACT

In this work, the modifications in the fitness landscape in two simple Dynamic Optimization Problems involving Evolutionary Robots are theoretically investigated. In the first dynamic problem, a robot with control laws optimized by a Genetic Algorithm should navigate in a simple environment. The changes in the fitness landscape are caused in this case by sensor faults. The second problem investigated here is when Evolutionary Robots are employed to reproduce the behaviour of rats in a maze. Changes in the rat's condition cause the modification in the fitness landscape. Simulations using the exact model of the Genetic Algorithm in each problem are presented, what allows studying the dynamical behaviour of the population of the GA, helping in the analysis of the performance obtained in practice.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*; G.3 [**Analysis of Algorithms and Problem Complexity**]: Non-numerical Algorithms and Problems—*Computations on discrete structure*

General Terms

Algorithms, Theory

Keywords

Genetic algorithms, dynamic optimization problem, robotics, fitness landscape, dynamical systems approach

*This work was supported by CNPq and FAPESP under grant 2010/09273-1.

GECCO'12 Companion, July 7–11, 2012, Philadelphia, PA, USA.

1. INTRODUCTION

In last decade, a large number of applications of Evolutionary Algorithms (EAs) in the design of robots and their control laws have been described in the scientific literature. One of the main motivations for the use of EAs in Robotics is the need for adaptability in unstructured, flexible and partially unknown environments [6]. Interestingly, in this case, the connection between Biology and Robotics is not in only one direction because robots can also be used for testing cognitive and evolutionary models of biological systems [4].

In several real world problems, where the control laws and navigation strategies are optimized by the EA, changes in the problem occur during the optimization process, what implies in changes in the fitness landscape. Such changes in the optimization problem occur due several reasons, like faults, environmental changes, platform modification, transfer of solutions from simulation to real environments, and cooperation and competition problems. In this way, strategies developed for Evolutionary Robots (ERs), i.e. robots designed using EAs, in dynamic environments have appeared in recent years, like noise addition [5], the fusion with other learning strategies [6], the insertion of random immigrants to maintain the diversity level [3], and the mutation distribution adaptation [13]. In such applications, the performance of the EA is experimentally tested, i.e., the EA is executed and the obtained results are analysed.

According to the author's knowledge, theoretical works analysing EAs applied in robots in dynamic environments were not produced in the literature. In fact, the lack of theoretical investigations on the operation of EAs does not occur only in Robotics, but also in a myriad of other areas. Overall, the number of investigations involving theory does not follow the fast growth of other areas of EAs. The problem is even worse in EAs in dynamic environments, where very few investigated the theory behind the algorithms (e.g., [1, 2, 8, 9, 10]). However, one could argue, the theoretical analysis should be general, independent of the application area, as some tools of theoretical analysis do not take into account the particularities of the search space associated with the problem. However, such tools have had limited success in explaining some particularities of the operation of EAs [7]. On the other hand, tools in which the particular search space must be known, as analysis by Markov Chains or by the Dynamical Systems Theory, have been successfully applied in many cases, despite of various limitations, the most important being the need to treat simple problems due to the size of the mathematical models.

Particularly in the case of modelling Genetic Algorithms (GAs) as dynamical systems, approach developed mainly by Vose [16] and whose model is known as exact model, although demanding a large number of equations to track all possible solutions represented the individuals of the GA, its use is quite interesting because it allows a complete description of the population dynamics [7]. Few works used the dynamical system approach to analyse GAs in dynamic environments. In [9], the standard GA with mutation and selection was investigated on Dynamic Optimization Problems (DOPs) with regular changes In [14], the authors proposed the use of the exact model to study the GA in dynamic environments created by the XOR DOP Generator [17]. Subsequently, this analysis was extended in [15] to the 0-1 knapsack dynamic problem.

In this paper, the modifications in the fitness landscape when a change occurs in two simple DOPs involving ERs are investigated. The choice of simple applications involving mobile robots is justified by the need for a complete description of the fitness landscape of the GA. The study of the fitness landscape changes is very important in understanding the DOP properties and the performance of an EA. It is still useful for the development of new benchmark DOP generators and the analysis of the properties of generators already used in practice, like the XOR DOP Generator,

Here, in the first investigated DOP, presented in Section 2, a robot with control laws optimized by a GA should navigate in a simple environment. Section 3 presents the second problem, where ERs are employed to reproduce the behaviour of rats in a maze. In both problems, the search landscape changes during the optimization process, due to faults in the robot's sensors in the first problem, and due to changes in the rat's condition in the second case. In Section 4, the GAs used to control the robot in the dynamic environments are simulated through their exact models, what allows studying the dynamic behaviour of the population of the GA, helping in the analysis of the performance obtained in practice. The conclusions of the paper are presented in Section 5.

2. PROBLEM 1: NAVIGATION OF A ROBOT IN A SIMPLE ENVIRONMENT

In this application, a mobile robot with one frontal sensor should navigate in a simple square environment. The robot can occupy one of nine different positions (squares) of this environment and its distance sensor generates a signal equal to $s = 1$ if the robot is facing a wall and $s = 0$ otherwise. The robot is controlled by a finite state machine with l bits, where each bit determines the action of the robot for each possible combination of sensor signal s and internal state m. The objective of the optimization process executed by the GA is to find a control law (finite state machine) that allows the robot to navigate in the environment without hitting the walls. Two models are investigated here, one with $l = 4$ bits and other with $l = 8$ bits.

In the first model, the robot can execute two actions: to move forward (0), i.e., the robot moves to next position located in its front, or to rotate clockwise (1), i.e. the robot changes its orientation without changing its position. The internal state signal m (internal memory) indicates the last action performed by the robot, i.e., a rotation ($m = 0$) or not ($m = 1$). In this way, the finite state machine has 4 bits, each one indicating a action for each possible combination of sensor input and memory, i.e. (s, m)= { 0,0 ; 0,1 ; 1,0 ; 1,1 }. For example, if the finite state machine is given by $\mathbf{x} = [0, 0, 0, 1]^{\mathrm{T}}$, the robot rotates only when it moved one step forward in last iteration and finds a wall in front of its current position. Then, the search space is composed by only $n = 16$ possible solutions. In the second model, 4 actions can be executed: to stay in the same position and orientation (00), to rotate clockwise (01), to rotate counterclockwise (10), and to move forward (11). Then, the finite state machine has 8 bits, two for each one of the 4 combinations of sensor input and memory. In this case, the search space is composed by $n = 256$ possible solutions.

In both models, the fitness is given by the number of positions occupied by the robot (i.e., the number of times it moves forward plus 1) until a collision with a wall or, otherwise, until a limit of 10 iterations. As the robot always starts in the same position and orientation (the first position and toward right), the maximum fitness that can be achieved is 8, because it should turn (without leaving its position) at least 3 times. The objective of this study is to investigate the modifications on the fitness landscape when a change occurs in the problem of applying the GA to find the finite state machine that produces the highest fitness. A simple problem of robot navigation is chosen here in order to allow monitoring the fitness of all possible solutions of the search space. In this DOP, changes are introduced in the problem by simulating three types of faults in sensor readings.

In the first fault (fault 1), the sensor readings are always equal zero. This type of fault can occur, for among other reasons, due to malfunction of the sensor or by a disruption of the cables connecting the sensor to the microcontroller controlling the robot. In the second fault (fault 2), the reverse occurs, i.e., the sensor readings are always equal to one, what can occur in case of a short circuit. In the third fault (fault 3), the readings from the sensor are inverted, i.e., when there is an obstacle in front of the sensor its reading is equal to $s = 0$, while $s = 1$ otherwise. This fault can occur due to malfunction of the sensor or the microcontroller.

In order to understand how the changes caused by faults affect the dynamical behaviour of the GA, we should investigate how the fitness landscape is modified in the transition of two consecutive change cycles (a change cycle is the period between two consecutive changes). In case of fault 1, the sensor reading is always $s = 0$, which results that the combinations of sensor input/memory are reduced to (s, m) = {0,0 ; 0,1 }. As a result, the actions given by the third and fourth elements of the vector $\mathbf{x}$ are respectively equal to the actions given by the first and second elements of the same vector. A similar effect occurs in case of fault 2, in which the sensor reading is always $s = 1$. In this case (s, m) = {1,0 ; 1,1 }, and as a result, the actions provided by the first and second elements of the vector $\mathbf{x}$ are respectively equal to the actions given by the third and fourth elements of this vector. In case of fault 3, in which the input signal is inverted, there is a permutation among the elements of the vector $\mathbf{x}$ (the exchanges are between the first and third elements and between the second and fourth elements). Thus, we can write that when a fault occurs in change cycle e, assuming that the robot is in normal operation in cycle $e - 1$, the vector $\mathbf{x}(e)$ is:

$$\mathbf{x}(e) = \mathbf{B}(e)\mathbf{x}(e - 1) \tag{1}$$

where, for model 1 (with $l = 4$), $\mathbf{B}(e)$ is for faults 1, 2 and

3 respectively given by:

$$\mathbf{B}_{f1}(e) = \begin{bmatrix} \mathbf{I}_2 & \mathbf{0}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}, \mathbf{B}_{f2}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix},$$

$$\mathbf{B}_{f3}(e) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{I}_2 & \mathbf{0}_2 \end{bmatrix}.$$

where $\mathbf{0}_2$ is a 2×2 matrix compose by zeros and $\mathbf{I}_2$ is the 2×2 identity matrix. As a consequence of Eq. 1, the fitness vector in change cycle e (faulty robot) can be computed through the fitness vector in change cycle $e-1$ (normal operation). The fitness vector contains the fitness of each possible solution of the search space [7]. In case of fault 1, the fitness of solutions in which the actions (elements of vector $\mathbf{x}$) are equal for $s = 0$ and $s = 1$ in cycle $e-1$ replace the fitness of solutions whose respective actions for $s = 0$ are equal (i.e., solutions with the same first half of the vector $\mathbf{x}$). The same occurs for fault two, but now replacing the solutions whose respective actions for $s = 1$ are equal (i.e., having the same second half of vector $\mathbf{x}$). Finally, for the third fault, there will be a permutation of the fitness vector elements corresponding to solutions that present different actions for different values of s. For the remaining elements of the fitness vector, the values remain unchanged. Thus, we can write:

$$\mathbf{f}(e) = \mathbf{A}(e)\mathbf{f}(e-1), \qquad (2)$$

where $\mathbf{f}(e)$ is the fitness vector and, for the model with $l = 4$, $\mathbf{A}(e)$ is respectively given for faults 1, 2, and 3 by:

$$\mathbf{A}_{f1}(e) = \begin{bmatrix} \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_d & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_e \end{bmatrix},$$

$$\mathbf{A}_{f2}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \\ \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix},$$

$$\mathbf{A}_{f3}(e) = \begin{bmatrix} \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 \\ \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_a & \mathbf{0}_2 & \mathbf{M}_b^{\mathrm{T}} \\ \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{0}_2 & \mathbf{M}_b & \mathbf{0}_2 & \mathbf{M}_c \end{bmatrix}.$$

where: $\mathbf{M}_a = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_b = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \mathbf{M}_c = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix},$

$$\mathbf{M}_d = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}, \mathbf{M}_e = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

In Figure 1, the fitness vector for each environment in the simple navigation problem is plotted. The best solutions for the robot in normal operation are solutions 1 ($\mathbf{x} = [0,0,0,1]^{\mathrm{T}}$) and 3 ($\mathbf{x} = [0,0,1,1]^{\mathrm{T}}$), i.e., when the robot rotates only when it moved forward in the last iteration and found a wall in front of its current position (solution 1), or when it finds a wall in front of the current position regardless of its state (solution 3). In both cases, the fitness is equal to the maximum fitness (8), and the strategy adopted by the robot is navigating in a clockwise direction at positions along the walls. In the solutions that have best fitness in the following (solutions 5 and 7), the robot rotates always after a forward movement. In this case, the fitness is equal to 6, as the robot turns 5 times while for the strategies with maximum fitness, the robot rotates only 3 times. Note, however, that this strategy generates the maximum fitness when the robot has faults 1 or 2 because it does not use the sensor to navigate (one can remember that the sensor readings are always equal to a fixed value for faults 1 and 2). Two other solutions indicating strategies where the sensor readings are not used in the navigation have the maximum fitness too in cases of faults 1 and 2. In case of fault 3, the fitness for the solutions is maintained, only being reordered according to the inversion of the sensor readings.

According to Eq. 2 and the classification of DOPs presented in [15, 14], one can observe that the faults generate linear homogeneous DOPs, i.e, dynamic problems where the fitness landscape in change cycle $e-1$ is modified according to a linear homogeneous transformation:

$$\mathbf{f}(e) = \mathbf{A}(\phi(e))\mathbf{f}(e-1), \qquad (3)$$

where $\mathbf{A}(.) \in \mathbb{R}^{n \times n}$ and $\phi(e)$ is the control parameters vector that define the dynamism of the problem. Furthermore, as $\mathbf{A}_{f3}(e)$ is a permutation matrix, fault 3 generates a DOP with permutation, i.e., where the linear transformation of the fitness landscape is given by a permutation matrix. In this way, the problem has characteristics similar to those found in DOPs produced by the XOR Generator [17], which is widely used by researchers to test EA for dynamic discrete optimization. However, while in DOPs produced by the XOR Generator the elements of the vector $\mathbf{f}$ are reordered according with a uniform permutation, where all elements of the fitness vector are reordered in the same way, fault 3 causes a non-uniform permutation, where some elements of the fitness vector are reordered, while others remain fixed. Anyway, in both cases, the values in the fitness vector remains equal, only being rearranged.

As discussed in [14], in a DOP with permutation with the characteristics of the problems produced by the XOR Generator or by the problem presented here, any metastable state in change cycle e (in case of a robot with fault 3) can be obtained by permutation of the corresponding metastable state in change cycle $e-1$ (normal operation). In addition, the average fitness in the metastable states for both environments are equal. In cases of faults 1 and 2, the linear transformation of the fitness vector causes the disappearance of some values of change cycle $e-1$. Thus, the metastable states are different for each cycle change, as well the corresponding average fitness. These results can be observed in the simulations of the robot presented in Section 4.

3. PROBLEM 2: USE OF ROBOTS FOR THE INVESTIGATION OF THE BEHAVIOUR OF RATS IN A MAZE

In [11], a robot controlled by an Elman Artificial Neural Network (ANN) was used to reproduce the behaviour of rats in an Elevated Plus-Maze (EPM). The EPM is used in Psychobiology for the study of behaviour and to test the effect of different drugs. The maze is composed by four united arms

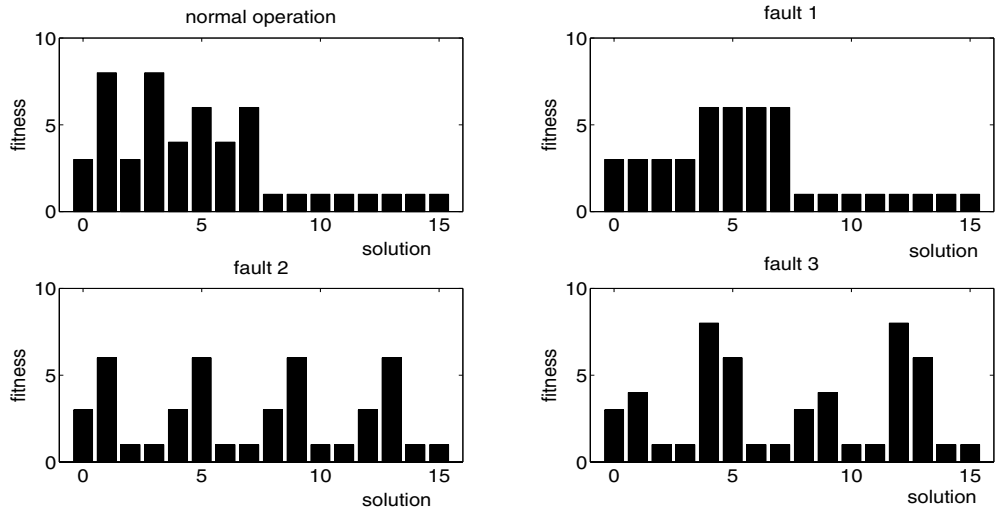

Figure 1: Fitness space (f) for problem 1 with $l = 4$.

elevated from the floor. Two opposite arms of the EPM are surrounded by walls. In [11], the weights of the ANN are optimized by a GA in order that the robot reproduces the behaviour observed in experiments with real rats, what is obtained after some generations.

The problem studied here is a simpler version of the problem investigated in [11]. Here, the same robot described in Section 2, added with a sensor to detect the colour of the floor, is used to reproduce the rat's behaviour. The new sensor is used to detect if the robot is in a closed or open arm as the squares of the closed arms are painted in a dark colour. The EPM here is composed by 9 positions (squares): one central square and 2 in each arm.

Here, the robot is controlled by a finite state machine with 8 bits optimized by the GA. Each bit of the state machine defines an action for the respective combination of frontal sensor reading (s), memory bit (m) and floor sensor reading (r). For example, if the finite state machine is $\mathbf{x} = [0, 0, 0, 0, 0, 0, 0, 1]^{\mathrm{T}}$, the robot rotates only when it finds a wall in front of its current position, moved one step forward in last iteration, and is located in a close arm (i.e., the floor is dark coloured). In this way, the search space has $n = 256$ possible solutions. When a individual (candidate solution) is evaluated, the robot with the respective state machine navigates during 150 iterations in the EPM, starting from the central position and headed toward one of the closed arms. The fitness of the individual $\mathbf{x}$ is computed by:

$$f(\mathbf{x}) = \frac{0.5}{3}\sum_{i=1}^{3}\big(1 - |time_d(i) - time(\mathbf{x}, i)|\big) + \frac{0.5}{2}\sum_{i=1}^{2}\big(1 - |ent_d(i) - ent(\mathbf{x}, i)|\big), \quad (4)$$

where $time(\mathbf{x}, 1)$, $time(\mathbf{x}, 2)$ and $time(\mathbf{x}, 3)$ are respectively the percentage of time spent in the central square, open arms and closed arms by the robot with solution $\mathbf{x}$; $ent(\mathbf{x}, 1)$ and $ent(\mathbf{x}, 2)$ are respectively the percentage of entrances in open and closed arms; and d means a desired value. The desired values were obtained by Silvio Morato and co-workers at the Laboratory for Exploratory Behaviour of the Department of Psychology, FFCLRP, University of São Paulo in experiments with real rats under three conditions [12]: normal and under effect of anxiolytic (chlordiazepoxide, 5 mg/kg) and anxiogenic (pentylenetetrazol, 30 mg/kg) drugs. When compared to the behaviour of the rat under normal condition (called here condition 0), the movement rate and time spent in the open arms increase for the rat under the effect of the anxiolytic drug (condition 2), while they are reduced for the rat under the effect of the anxiogenic drug (condition 1).

Here we investigate how the fitness landscape changes when the optimization problem changes from condition 0 (normal) to conditions 1 or 2. Starting the evolutionary process from the population evolved for condition 0, when compared to restarting the evolutionary process, can save substantial computational effort for the optimization of control laws for conditions 1 and 2. Besides, the analysis of the modifications in the fitness landscape can help to understand how the navigation strategies change when data from the experiments with the rats under the effect of anxiolytic and anxiogenic drugs is used instead of data of the experiments with the rat in normal condition. Figure 2 shows the fitness landscape for the three conditions. When the difference (Δf) between the fitness vectors for condition 1 and condition 0 are analysed, some observations can be made.

First, it is possible to observe that Δf assumes one of three values: 0, $-\Delta f_c$, and $+\Delta f_c$. The first value ($\Delta f = 0$) always occurs when the second bit of the individuals is equal to 1, i.e., when the robot turns (action 1) for $\{s, m, r\} = \{0, 0, 1\}$. In this case, as the robot always starts in the central square of the maze with $m = 0$, the robot will rotate in this position for all iterations. Thus, only the percentage of time spent in the central square is higher than 0. As the desired percentage of time spent in the central square is equal for conditions 0 and 1, the fitness for the solutions where the second bit is equal to 1 does not change when the problem is modified from condition 0 to conditon 1.

From analysing Δf, one can identify three other bits (third, fifth and seventh bits) whose values do not care for the change in the fitness . One can also identify values of bits responsible for $\Delta f = -\Delta f_c$ and $\Delta f = +\Delta f_c$. In order to facilitate the analysis of the bits' influence, schemata, i.e., templates with bits 0, 1 and * (do not care), can be associ-

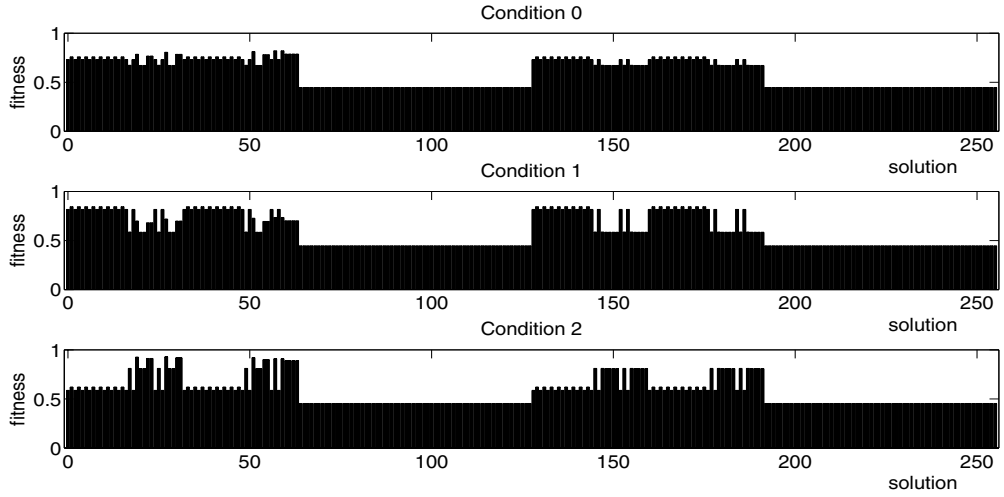

Figure 2: Fitness space (f) for problem 2.

ated for each subset of solutions where the fitness difference is $-\Delta f_c$ or $+\Delta f_c$.

For the subsets of solutions associated with the fitness difference $-\Delta f_c$, the schemata $*0*0****$ and $*0*1*0*0$ can be identified. For the subsets of solutions associated with changes $+\Delta f_c$, we can identify the schemata $*0*1****1$ and $*0*1*1*0$. Those schemata represent subset of solutions related to actions associated with the closed arms, being important for the navigation strategies. When the environment changes from condition 0 to condition 1, the time spent in the closed arms should increase. As a consequence, solutions with actions that makes the robot to spend more time in the closed arms generates an increase ($+\Delta f_c$) in the fitness. However, the fitness should be decreased for those solutions with actions that makes the robot to spend less time in the close arms.

A similar analysis can be made when fitness landscape changes for the problems from condition 0 (normal) to condition 2. The analysis in this case is more complex as Δf assume 10 different values. In this case, actions where the robot spends more time in the open arms should be explored for condition 2, what is not done in conditions 0 and 1, resulting as a consequence in a more complex fitness landscape change pattern. Anyway, in this case, it is possible to identify 18 schemata in order to describe the different subsets of solutions associated with the different values of Δf.

For both DOPs, the changes in the fitness vector from change cycle $e-1$ (condition 0) to change cycle e (condition 1 or 2) are ruled by:

$$\mathbf{f}(e) = \mathbf{f}(e-1) + \mathbf{b}(\Omega(e)), \tag{5}$$

where $\Omega(e)$ is a set of schemata and the i-th element of $\mathbf{b}\big(\Omega(e)\big)$ is given by:

$$b_i(\Omega(e)) = \sum_{j=1}^{|\Omega(e)|} a(\mathbf{x}_i, \mathbf{s}_j(e), e) \tag{6}$$

where $\mathbf{s}_j(e) \in \Omega(e)$ and:

$$a(\mathbf{x}_i, \mathbf{s}_j(e), e)\big) = \begin{cases} \Delta f_j(e), & \mathbf{x}_i \in \mathbf{s}_j(e) \\ 0, & \mathbf{x}_i \notin \mathbf{s}_j(e) \end{cases} \tag{7}$$

By Eq. 5 and according to the classification of DOPs presented in [15], one can observe that the changes in the environments generate linear DOPs.

4. SIMULATION RESULTS

In this section, the exact model [16, 7] of the GA is simulated in the problems described in last two sections. In the exact model, the dynamical behaviour of the GA is described by the trajectories of the the population vector, which defines the proportion of each possible solution in the population at generation t. In a DOP [14], the population vector at generation t is given by:

$$\mathbf{p}(t) = \mathcal{G}\big(\mathbf{p}(t-1), t\big), \tag{8}$$

where $\mathcal{G}(.,t) : \Lambda \times \mathbb{N}^+ \rightarrow \Lambda$ is the generational operator (map) at generation t, $\mathbf{p}(t)$ is the expected population at generation $t \in \mathbb{N}^+$, $\mathbf{p}(0)$ is the initial population vector, and Λ is the simplex where the population vector is contained:

$$\Lambda = \left\{ \mathbf{p} \in \mathbb{R}^n : p_k \geq 0, \text{ for } k = 0, \ldots, n-1 \text{ and } \sum_{k=0}^{n-1} p_k = 1 \right\}, \tag{9}$$

where n is the number of possible solutions in the simplex. In the limit $N \rightarrow \infty$ (infinite population case), where N is the population size, the trajectory of the population in the simplex can be deterministically described. The analysis of the generational operator can provide important insights in understanding the behaviour of the GA. The fixed points of Eq. 8, called metastable states, play an important role in the evolutionary process as they can change the trajectory in the simplex, attract the population vector, and trap finite populations for several generations.

Here, the trajectories of the populations of GA with flip mutation and proportional selection when applied to the problems described before are simulated and analysed. In the simulations presented here, rather than execute the GA, its dynamical system is simulated, i.e., the evolution of the population vector is simulated according to Eq. 8. In the simulations, the initial population is uniformly distributed

and the mutation rate is equal to 0.01. For the problem described in Section 2, the duration of each change cycle is $\tau = 40$, while for the problem described in Section 3, $\tau = 100$.

4.1 Results for Problem 1

Due to space limitations, we present here only the results for the simulations of the second model ($l = 8$) for the problem presented in Section 2. However, similar behaviour of the dynamical system is observed in the simulations of model 1 ($l = 4$). Figure 4 shows the simulations of the dynamical system of the GA with $l = 8$. In the simulations, the robot is in normal operation in the first $\tau = 40$ generations and has one fault in the remaining generations.

In Figure 4, the average fitness of the population and the Euclidean distance between the vector population in the current generation and the three eigenvectors with the highest eigenvalues at the end of the simulations are presented. For the GA with flip mutation and proportional mutation [7], the generational operator is given by:

$$\mathcal{G}(\mathbf{p}) = \frac{UF\,\mathbf{p}}{\mathbf{f}^{\mathrm{T}}\mathbf{p}}. \tag{10}$$

where U is the mutation matrix, $\mathbf{f}$ is the fitness vector, and $F = \mathrm{diag}(\mathbf{f})$ is a diagonal matrix. In the GA with flip mutation and proportional mutation, the eigenvectors of UF define de metastable states (fixed points) of the dynamical system. In Figure 4, the first eigenvector (i.e., the eigenvector with the largest eigenvalue) always corresponds to the main metastable state of the dynamical system, i.e. that one in which the number of individuals in the current global optima is greater than the number of individuals in any other location for the problems presented here. The other eigenvectors correspond to other metastable states that have importance for understanding the population dynamics.

The simulations presented here help to understand the behaviour of the GA when a change occurs in the fitness landscape for the navigation problem. One can observe that in the first change cycle (robot without faults), the population vector quickly converges to the main metastable state, where most of the population is distributed between the two global optima of the fitness landscape, and as a consequence, the average fitness of the population is near the maximum allowed fitness. When a fault occurs, the population is located in the main metastable state of the first change cycle, which is different from the main metastable state of the second change cycle (i.e., when the robot has a fault). Thus, the population vector must migrate to the new metastable state when the fault occurs. In this case, depending on the type of fault, two situations can occur. In the first situation, even with the change of the main metastable state, this one is still the state closer to the current position of the population (i.e., the position of the former main metastable state). As a consequence, the population vector easily reaches the new metastable state after the fault. This is the case for fault 2, in which the metastable states before and after the fault are close due to the fact that one of the solutions that presents maximum fitness before the fault also presents the maximum fitness after the fault. One can observe that for the second fault, the average fitness rapidly converges to a value close to the maximum fitness (value equal to 6).

Note, however, that this does not occur for faults 1 and 3. In these cases, when the fault occurs, there are other metastable states closest to the current position of the population vector (corresponding to the position of the metastable state for change cycle 1, i.e., before the change caused by the fault) than the main metastable state after the fault. One can observe in Figure 4 that, in these cases, the population approaches in a first moment one of these states, spending a time in its neighbourhood, before converging to the main metastable state of change cycle 2. As a result, the population needs more time in order to converge to the new global optima, what can be observed in the graphs of the mean population fitness. In this way, faults 1 and 3 represent changes more severe than fault 2 for this dynamic optimization problem. Another important factor that influence the difficulty of the dynamic optimization problems is the duration of the change cycles. Note that in case of faults 1 and 3, change cycles lasting less (i.e., in which changes occur more frequently, as in the case of fast intermittent faults), the performance can be more affected because the population does not have enough time to reach the neighbourhood of the current main metastable state, staying in the neighbourhood of another metastable state. It is important to observe, however, that a worse performance would depend on other properties of the dynamical system, like the average fitness in each metastable state.

4.2 Results for Problem 2

Figure 4.2 shows the simulations of the Dynamical System of the GA for the experiment with the use of robots for the investigation of the behaviour of rats in the EPM. In the simulations, the fitness of the solutions are computed using the data from the rats in condition 0 for the first $\tau = 100$ generations, and in conditions 1 or 2 for the remaining generations.

One can observe that the population vector reaches the neighbourhood of the main metastable state for the changed landscape after some generations. In the figure, we can observe that the convergence to the new main metastable state, where more individuals are located in the global optima than in any other part of the fitness space, is faster for condition 2 than for condition 1. From Figure 4.2 it is possible to observe that the new main metastable states for both conditions 1 and 2 are locate closer to the old main metastable state (for condition 0) than any other state. However, this distance is smaller for condition 2 (distance equal to 0.2310 against 0.2730 for condition 1), what explains the faster convergence of the population vector for condition 2.

5. CONCLUSIONS

In this paper, the modifications in the fitness landscape for two simple robotic DOPs were studied, as well the behaviour of a simple GA when applied to such dynamic problems. In Section 2, the changes caused by faults in the fitness landscape for the first problem (navigation in a simple environment) were analysed. It was observed that the changes caused by the faults generate linear homogeneous DOPs, where the fitness vector is changed according to a homogeneous linear transformation. Also, in case of fault 3, the problem is also a DOP with permutation. DOPs with permutation are also produced by the XOR DOP Generator, but while this generator produces uniform permutations in the whole fitness landscape, fault 3 produces permutations only in some solutions of the fitness space. Section 3 showed that DOPs with linear transformations are also produced

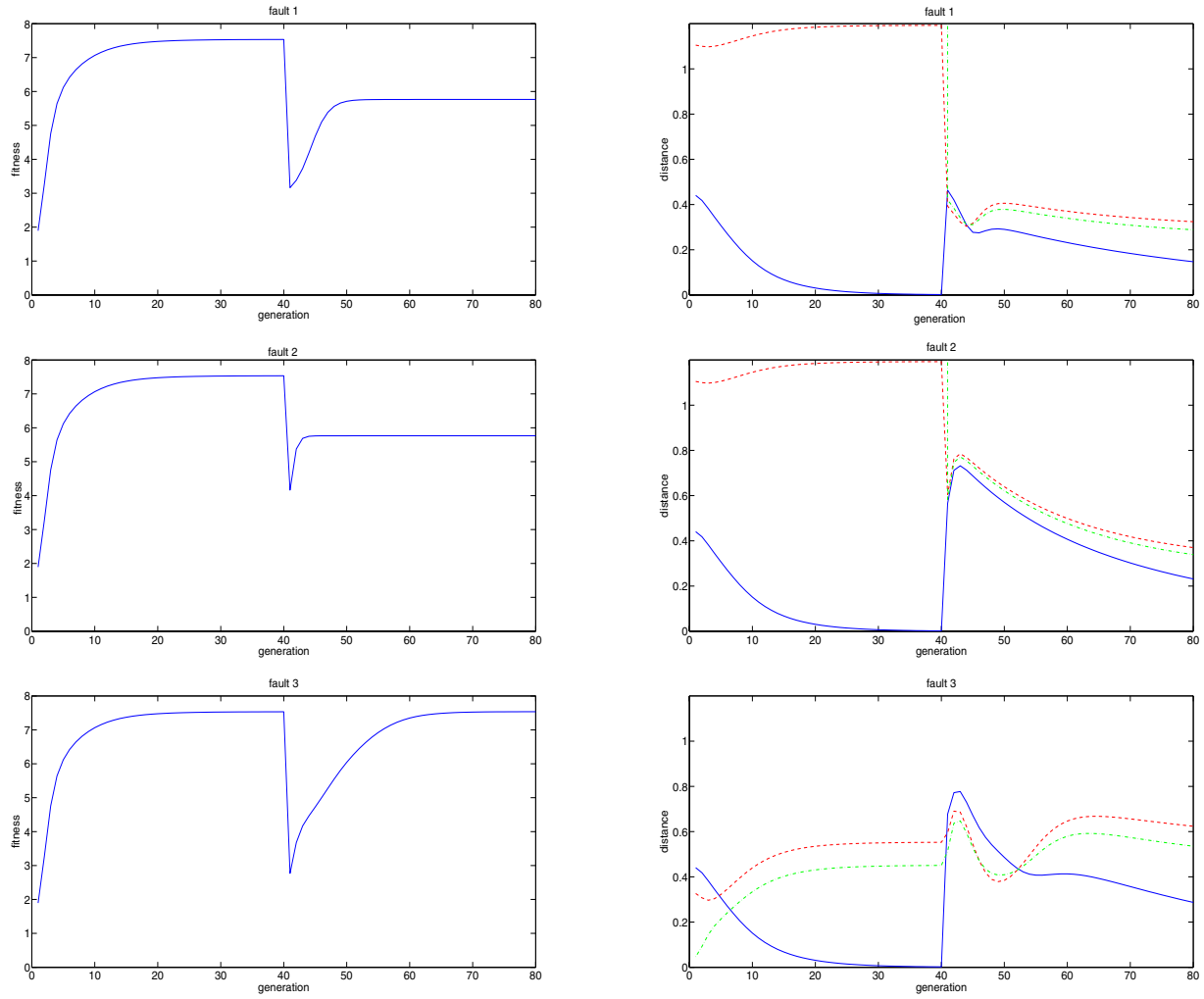

Figure 3: Mean fitness and distances from the population to three metastable states for problem 1 with $l = 8$. The solid line shows the distance to the main metastable state.

in the second problem (use of robots to investigate the behaviour of rats in a maze). However, in this case, the transformations occurs in subset of solutions, what was described using different templates (schemata).

In Section 4, simulations of the dynamical system of the GA in the studied problems were analysed. In the first problem, it was observed that the DOPs generated by faults 1 and 3 are harder than those generated by the second fault. This is due to the fact that, for these faults, metastable states different from the main one after the fault are closer to the main metastable state before the fault. Thus, the population stays for a time in the vicinity of these states prior to converge to the main metastable state (after a fault). As a result, the GA takes longer time to reach the global optima, which causes a deterioration in the performance (in relation to fault 2). In case of the second problem, the distance between the old (before the change) and new (after the change) main metastable states explain the difference in the velocity of convergence in landscapes for conditions 1 and 2. The study of the modifications of the fitness landscape after a change and the use of the dynamical system approach helps the investigation of the behaviour of the GA from a theoretical point of view, which is extremely useful for the study of performance, mostly based on experimental analyses. The main disadvantage of this theoretical approach is the applicability to small problems, requirement to allow the monitoring all possible solutions of the problem. As future works, this approach should be applied to other DOPs. The study of the modifications in the fitness landscape is still important in order to study and propose DOP benchmark problems generators, like the XOR DOP Generator.

6. REFERENCES

[1] D. Arnold and H. Beyer. Random dynamics optimum tracking with evolution strategies. *Parallel Problem Solving from Nature - PPSN VII*, pages 3–12, 2002.

[2] S. Droste. Analysis of the (1+ 1) EA for a dynamically changing onemax-variant. In *Proc. of the IEEE Cong. on Evolutionary Computation*, pages 55–60, 2002.

[3] D. Floreano, S. Nolfi, and F. Mondada. Co-evolution and ontogenetic change in competing robots. In

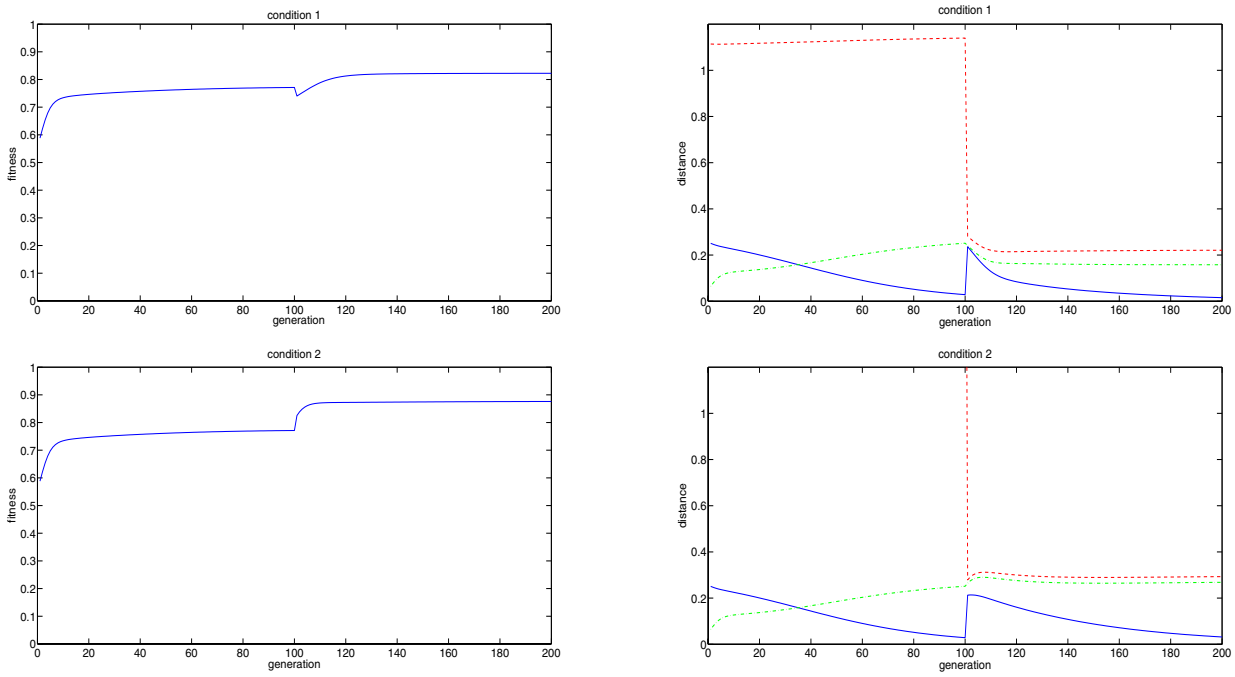

Figure 4: Mean fitness and distance from the population to three metastable states for problem 2.

M. Patel, V. Honavar, and K. Balackishnan, editors, *Advances in the Evolutionary Synthesis of Intelligent Agents*. MIT Press, 2001.

[4] I. Harvey, E. A. DiPaolo, R. Wood, M. Quinn, and E. Tuci. Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, 11(1-2):79–98, 2005.

[5] N. Jakobi. Half-baked, ad-hoc and noisy: minimal simulations for evolutionary robotics. In P. Husbands and I. Harvey, editors, *Proc. of the Fourth European Conference on Artificial Life*, pages 348–357. MIT Press, 1997.

[6] S. Nolfi and D. Floreano. *Evolutionary robotics: the biology, intelligence, and technology of self-organizing machines*. MIT Press/Bradford Books: Cambridge, USA, 2000.

[7] C. R. Reeves and J. E. Rowe. *Genetic algorithms - principles and perspectives: a guide to GA theory*. Kluwer Academic Publishers, 2003.

[8] P. Rohlfshagen, P. K. Lehre, and X. Yao. Dynamic evolutionary optimisation: an analysis of frequency and magnitude of change. In *Proc. of the Genetic and Evolutionary Computation Conf.*, pages 1713–1720, 2009.

[9] C. Ronnewinkel, C. O. Wilke, and T. Martinetz. Genetic algorithms in time-dependent environments. In B. N. L. Kallel and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 263–288. Springer, 2001.

[10] J. E. Rowe. Finding attractors for periodic fitness functions. In *Proc. of the Genetic and Evolutionary Computation Conf.*, pages 557–563, 1999.

[11] H. K. Shimo, A. C. Roque, J. Tejada, S. Morato, and R. Tinós. Use of evolutionary robots as an auxiliary tool for developing behavioral models of rats in an elevated plus-maze. In *Proc. of the 11th Brazilian Symposium on Neural Networks*, pages 217–222, 2010.

[12] J. Tejada, G. G. Bosco, S. Morato, and A. C. R. da Silva. Characterization of the rat exploratory behavior in the elevated plus-maze with markov chains. *Journal of Neuroscience Methods*, 193(2):288–295, 2010.

[13] R. Tinós and A. C. P. L. F. Carvalho. Use of gene dependent mutation probability in evolutionary neural networks for non-stationary problems. *Neurocomputing*, 70(1-3):44–54, 2006.

[14] R. Tinós and S. Yang. An analysis of the xor dynamic problem generator based on the dynamical system. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature - PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 274–283. Springer Berlin / Heidelberg, 2011.

[15] R. Tinós and S. Yang. Analyzing evolutionary algorithms for dynamic optimization problems based on the dynamical system. In S. Yang and X. Yao, editors, *To appear in the book Evolutionary Computation for Dynamic Optimization Problems*, pages 1–25. Springer, 2012.

[16] M. D. Vose. *The simple genetic algorithm: foundations and theory*. The MIT Press, 1999.

[17] S. Yang and X. Yao. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. *Soft Computing*, 9(11):815–834, 2005.