# Dynamic Difficulty Adjustment with Evolutionary Algorithm in Games for Rehabilitation Robotics

Kleber de O. Andrade, Thales B. Pasqual, Glauco A. P. Caurin
Engineering School of São Carlos - EESC
University of São Paulo - USP
São Carlos, SP, Brazil
Email: {pdjkleber,gcaurin}@sc.usp.br
thales.bueno@usp.br

Marcio K. Crocomo
Federal Institute of Education, Science
and Technology of São Paulo
Piracicaba, SP, Brazil
Email: marciokc@ifsp.edu.br

***Abstract*—This article explores game difficulty adjustment for serious game applications in rehabilitation robotics. In this context, a difficulty adjustment system is proposed that takes user performance as input and generates two different responses: a) a change in the distance the user should cover, and b) the velocity provided to the target. User performance is estimated from its ability to achieve the targets (game score) performing movements. The system interference in user displacement value and target speed where chosen to stimulate the user to achieve specific rehabilitation goals. The game difficulty adjustment has received small attention in the context of rehabilitation robotics interfaces. It is important to note that games developed for rehabilitation differ from commercial entertainment games due to severe limitations imposed to patients by pathologies like stroke, cerebral palsy and spinal cord injury. An Evolutionary Algorithm (AE) based optimization strategy was adopted to adjust game's difficulty. A meta-profile for user behavior was also developed allowing to create and simulate different virtual users and game experiences in computer. This user profile includes a reaction time (time delay), motion disturbance and a kinematical motion profile based on a polynomial function. Using the meta-profile, different user motion behavior can be generated for exhaustive test and optimization of the difficulty adjustment system. The approach allows the reduction of development time and also the reduction in the number of experiments with volunteers. The computer simulation test results are presented to demonstrate the capacity of the difficulty adjustment system to adapt the game characteristics to the users' abilities with different skills levels.**

## I. Introduction

Research involving robotic rehabilitation, as other rehabilitation activities, imposes enormous challenges to healthcare professionals, especially regarding patient's motivation to perform the prescribed exercises. It is important to mention that there are currently many initiatives involving the application of commercial games for rehabilitation. Without disregarding the social and human benefits of these initiatives, the clinical effectiveness of these activities are hard to distinguish form the results obtained by conventional treatments [1].

Commercial games offer a very rich and pleasant experience, but at the same time they are designed with difficulty levels that go far beyond the typical capabilities of users with disabilities.

In addition, we postulate there is a need for the development of games specifically tailored for rehabilitation, providing at least the following characteristics:

- To offer health professionals the ability to interfere in the game dynamic;
- To provide tools that allow to record and transfer game data associated with the user performance;
- To allow the combined use with robotic devices and provide interfaces to stimulate perception and register motion of the limbs.

The current interfaces, used by both scientific and commercial rehabilitation robots, are in comply with these ideas. However, these interfaces are still very limited when one considers the experience they offer to user. In this context, as a positive exception, the work developed by [2], [3] shows some concern in the presentation of a pleasant interface to the user.

It is also possible to observe the current implementation lack to offer progressive challenges to the users. This is an important requirement for the game development, especially if we consider the long rehabilitation periods (weeks) that are commonly prescribed to the users. Currently, users have the same interface from the beginning to the end of the treatment.

Previous works developed by [2]–[4] suggest that adaptive procedures provide challenges to the user, trying to transform rehabilitation in a more attractive process.

In this paper, we propose a new difficulty adjustment mechanism based on Evolutionary Algorithm (AE) [5] integrated with a user model. This approach is integrated in a rehabilitation games called "The Catcher" and "Happy Fish" present in Section II-B. We present evidence of the effectiveness of our approach using computer simulations. The approach also reduces the requirement for human volunteers tests.

## II. Rehabilitation Robotic System

This section presents the rehabilitation robotic system, Section II-A explains the robotic device developed; Section II-B explains the serious games developed for rehabilitation robotic.

978-1-5090-2210-6/16/$31.00 ©2016 IEEE

### A. Robotic Device

To serve as an interface between the patient and the game, a robotic device with one degree of freedom was developed. The system illustrated in [3] is backdrivebly (low mechanical impedance) and provided with impedance control [6]. These characteristics are not found in industrial robots and ensure safe contact interaction with the user. This robotic system shows different behavior to meet the therapist and the user demands. The robot can detect if the user is not able to generate movements and provide assistance. Conversely, it can resist the user movement promoting in this way gains in muscle strength. It may also show a completely neutral behavior emulating a conventional joystick.

Our rehabilitation robot device was designed for exercises with a single degree of freedom at a time: flexion/extension; or adduction/abduction; or pronation/supination. The user's wrist angle is shared with the game through TCP/IP protocol. More detailed description of the robot prototype can be found in [3], [7], [8].

### B. Serious Games

Two games were created for rehabilitation purposes. These games are used in combination with a single degree of freedom robot device and where develop using Unity[1] game engine. In both games, difficulty level is encoded in the initial distance to the target (nuts in "The Catcher" (Fig. 1) or space between the seaweed in "Happy Fish" (Fig. 2)), the speed and the size of the target (the latter will not be discussed in this article). If the goal is too far away and the player cannot provide enough speed in his wrist to reach the target, he or she will not earn points.


Fig. 1. Screenshot the games "The Catcher" under study and development for horizontal movement.

The same concept is valid if the goal is reachable, but the target is moving too fast. In principle, it is possible to generalize that most games for single degree of freedom robotic devices will use three variables (or some kinematical equivalent) for difficulty adjustment.

[1]Unity (http://unity3d.com) is a cross-platform game engine with a built-in Integrated Development Environment (IDE). The technology provides a rendering engine with integrated development utilities and workflows that provide the capability to create interactive 3-dimensional (3D) and 2-dimensional (2D) content for visualization, training, and medical simulations.

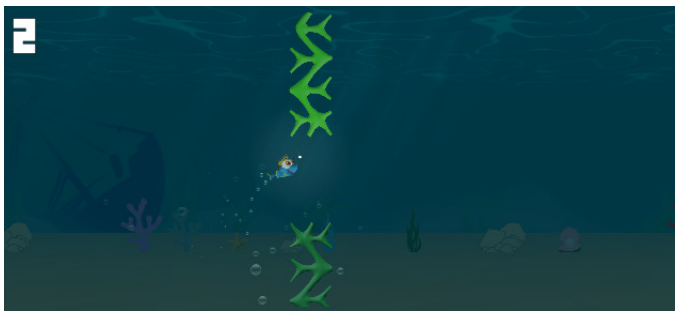


Fig. 2. Screenshot the games "Happy Fish" under study and development for vertical movement.

For clarity reasons some additional definitions are necessary in the context of this paper. A game (robot rehabilitation) session will present the player to a fix number of targets. When the game starts, a single target appears on the screen with is a distance $d$ from the player and a speed $s$. This paramenters define the current difficulty level. When a target is reached or destroyed, a new target is generated. If the player succeed in reaching a target, this single event is called a "hit". If, for any reason, the player do not reach the target this single event is called a "miss". This time interval between two targets,is called a "game round". Each game round has an independent difficulty level. In this work an Algoritm 1 simulates a single player using the game.

**Algorithm 1** Algorithm for game model

**Input:** $goals$ {number of goals}
**Output:** $t_{game}$ and $hits$

$hits \leftarrow 0$
**for** $i = 1$ to $goals$ **do**
  $d_i \leftarrow$ number between $d_{min}$ and $d_{max}$
  $s_i \leftarrow$ number between $s_{min}$ and $s_{max}$
  $rnd \leftarrow$ random number between 0 and 1
  **if** $(rnd \leq \frac{x_{player}}{w})$ **then**
    $direction \leftarrow -1$
  **else**
    $direction \leftarrow 1$
  **end if**
  **if** $(w_{min} < x_{player} + d_i * direction < w_{max})$ **then**
    $direction = direction * -1$
  **end if**
  $x_{goal} \leftarrow x_{player} + d * direction$
  $t_{goal} \leftarrow \frac{h_{max}}{s_i}$
  $x_{player} \leftarrow$ player inputs
  **if** $(|x_{player} - x_{goal}| < size_{player})$ **then**
    $hits \leftarrow hits + 1$
  **end if**
  $t_{game} \leftarrow t_{game} + t_{goal}$
**end for**

## III. Difficulty Adjustment in Games

Game balancing may be defined as the process of ensuring a good game challenge level. The challenges are intended to avoid extremes such as frustrate player where the game difficulty is too high or bore him or her where the game only offers simple challenges [9], [10]. The Fig. 3 shows the concept of flow according to [9].

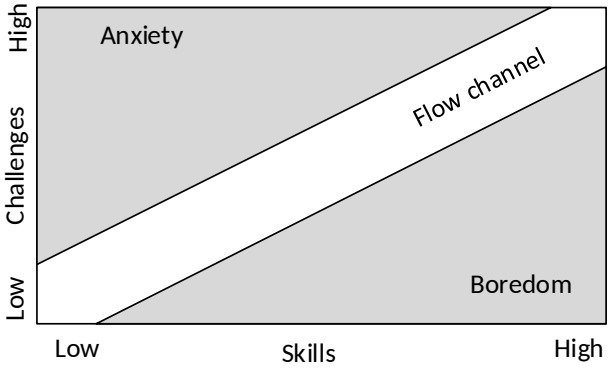


Fig. 3. Mental state in terms of challenge level and skill level, according to [9] flow model.

There are two major classes of game balancing: the static and dynamic:

Static balance method is a well established approach, where predefined difficulty levels (i.e. easy, medium and hard) are presented to the user and he (or she) picks an option. For each difficulty level, techniques associated with this approach define the game parameters that impose specific challenges. Usually the challenges are grouped by phases or stages with increasing difficulty, in order to establish a correspondence with the player's skill evolution. Players access new difficulty levels as their skills improve. [11] considers three problems in the levels selection. *(i)* players have a limited number of difficulty level options (usually three or four levels). *(ii)* players are not able to set appropriate difficulty level for their skills. *(iii)* in general difficulty setting affects only the attributes of the computer-controlled characters, but not their tactics. They suggest a dynamic approach to overcome these drawbacks.

The goal of a Dynamic Difficulty Adjustment (DDA) is to keep the game in constant balance, avoiding situations where the player is in advantage or disadvantage. Most of the DDA techniques found in the literature use an assessment function to define the difficulty level faced by the user along the game or at specific events. This function is called challenge function [12] and for a specific game it should be composed by the most relevant parameters that may affect player performance.

Artificial intelligence (AI) techniques can also be used during the game to contribute for dynamic balancing. These techniques should periodically assess the difficulty level perceived by the player and make the necessary adjustments in the game, ensuring an appropriate challenge level [12]. The use of AI allows even the addition of unpredictability factor, alternating easier and more difficult situations,that increases player interest in the game.

## IV. DDA with Evolutionary Algorithm

This section presents the proposed Evolutionary Algorithm, i) explaining the chromosomes and defining the boundaries for each parameter; ii) explaining how the fitness function is calculated and iii) presenting the adopted selection and mutation operators.

### A. Representation

To define a chromosome for the Evolutionary Algorithm, two important parameters must be introduced which characterize a game round in the The Cather game, but that may be also extended for any other similar game. The first parameter is the distance to the target $d$, which is a real value in $[0.01w\ 0.5w]$, where $w$ is the game screen width in pixels, while the second parameter is the speed of the target $s$, which is a real value in $[0.1h\ 0.5h]$, where $h$ is the game screen height in pixels. According to the defined boundaries, the target may be close to the player character (at $0.01w$) or may be half screen away. The time to reach the target is defined between two to ten seconds. Therefore, $d$ and $s$ are the two values that compose the EA chromosome. A population of size $n$ is represented by a two-dimensional matrix as illustrated in Fig. 4, where each line represents a single chromosome.

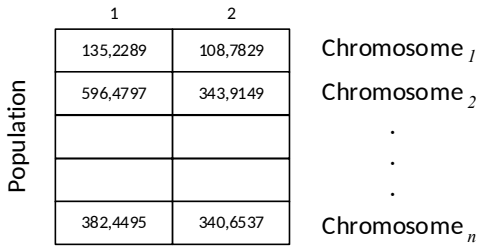


Fig. 4. A Population containing n chromosomes, each represented by a line containing two parameters: $d$ and $s$.

### B. Fitness Evaluation

To evaluate the fitness of a chromosome, current $s$ and $d$ values are used to start a game round. When the round ends, the fitness is calculated using Equation 1.

$$fitness_i = K_d * G_d + K_s * G_s - K_e * \varepsilon + K_c * \delta \quad (1)$$

Where, $K_d$, $K_s$, $K_e$ and $K_c$ are coefficients that set the impact that each of the terms $G_d$, $G_s$, $\varepsilon$ and $\delta$ will have on the fitness value. Terms $G_d$ and $G_s$ reflects the game distance and speed, and are calculated as shown by Equations 2 and 3.

$$g_d = \frac{d - d_{min}}{d_{max} - d_{min}} \quad (2)$$

$$g_s = \frac{s - s_{min}}{s_{max} - s_{min}} \quad (3)$$

Where $d$ is the initial distance from the target to the player and $s$ represents the target speed, both defined by the values of the chromosome being evaluated. Values $d_{max}$ and $d_{min}$ are the maximum and minimal distance that the game allows, varying

respectively between 50% and 1% of the game screen width, as defined in Section IV-A. Similarly, $s_{max}$ and $s_{min}$ are the minimum and maximum allowed speed, respectively 10% and 50% of the game height per second. The first two terms in Equation 1 assume values between 0 and 1 that corresponds to the game round difficulties. While $G_d$ represents how far and $G_s$ how fast the target is, according to the game limitations.

The third term in Equation 1 $\varepsilon$, represents how far from the target horizontal position the player was when the round ended. Equation 4 shows how $\varepsilon$ is calculated, considering the target and the player horizontal position, $x_{target}$ and $x_{player}$, respectively, when the target hits the ground. In other words, this term decreases the fitness value in proportion to how far the player character stands from the target when the round ends. The minimum and maximum values for both $x_{target}$ and $x_{player}$, are 0 (left side of the screen) and 1 (right side of the screen).

$$\varepsilon = \left| x_{target} - x_{player} \right| \tag{4}$$

The last term, $\delta$, represents how the player used the given time to reach the target, and it is calculated as shown in Equation 5.

$$\delta = \begin{cases} \frac{t_{player}}{t_{target}} & \text{if } t_{player} \leq t_{target} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The terms $t_{player}$ and $t_{target}$ in Equation 5 represent, respectively, the time used by the player to reach the target horizontal position and the time it takes for the target to hit the ground. A round is considered good if the player makes good use of the time available. To illustrate this concept, a game round where the player can reach the target horizontal position in 1 second, while it takes 5 seconds for the target to reach the ground can be considered too easy by the player, thus, $\delta$ is given a small value. A more challenging scenario would be one where the player takes 4.5 seconds to reach the target while the target hits the ground in 5 seconds, which would result in a higher value for $\delta$, positively collaborating with the given chromosome $fitness$.

Summarizing, the first two terms in Equation 1 increase the fitness value as the round difficulty increases (higher values of $s$ and $d$), the third therm attempts to increase the fitness value by a measure of how challenging the round was and the last term decreases the fitness for game rounds that are too frustrating for the user.

### C. Selection and mutation

To create each chromosome of the next population, an elitist approach is used [13], where the chromosome with higher fitness is selected. A new population is created by replacing each individual from the previous population by a copy of the selected individual, and applying a bounded mutation operator [14] on all chromosomes, except one. A copy of the best chromosome is maintained. The applied mutation operator adds a uniform random variable in [0.05, 0.05] for $d$ and $s$ within the limits of each parameter. After the introduction of these values, each parameter is adjusted so that they do not exceed the defined boundaries. Fig. 5 illustrates this process.

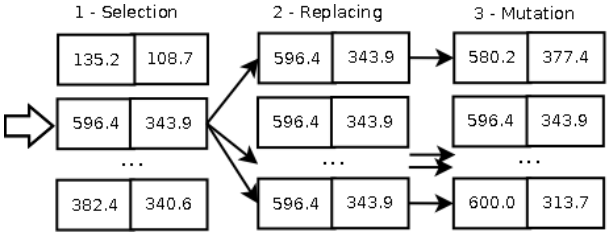


Fig. 5. Applying selection and mutation operators

### D. The proposed Evolutionary Algorithm

Fig. 6 illustrates how the proposed EA works. The starting step is the construction of the first population, when each chromosome is initialized with random values limited by the parameters defined in Section IV-A. The second step is the calculation of the fitness, when one match is played for each set of parameters ($d$, $s$) of the current chromosomes, resulting in a fitness value for each chromosome, as explained in Section IV-B. Step 3 uses selection and mutation operators to create the next population, as explained in Section IV-C, after that, the process returns to the second step (evaluation).

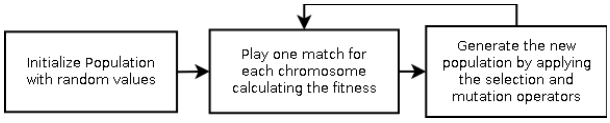


Fig. 6. Applying selection and mutation operators

The cycle represented by steps 2 and 3 in Fig. 6 is repeated until the game ends. The game adaptation is a continuous process, where the game challenge should improve as a function of the player ability. However, for our tests, steps two and three are repeated until a certain number of generations is reached, as explained in Section V.

### E. Simplified User Behavior Model

The proposed model generates velocity profiles that are inspired in the definition of discrete movements [15] performed by healthy subjects. The corresponding velocity profiles parameters may be adjusted to approximate human movements. It is important to note, however, that it is not the purpose of this work to generate high fidelity copies of the single human movements. In our perspective it is more important to establish a software tool that allows the integration of different motion models.

Modularity is one of the important features of this approach. It will allow, in a near future, the use of sub-movements [16] resulting in complex behaviors emulation some forms of motor disability. The generation and the study of more complex movements will be the subject of future work.

For simplicity reasons, a third order polynomial $x(t)$ was adopted to simulate two movement primitives, namely the

movement introduced by the user into the game and an ideal movement that would be necessary to timely reach a task presented in a game round. These polynomials represents the robot handle trajectory as a function of the time from the current initial position $p_i$ to the target position $p_f$. Where the auxiliary variable $d$ represents the displacement.

$$x(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{6}$$

$$d = p_f - p_i \tag{7}$$

The introduction of user motion characteristics defines its corresponding polynomial parameters, i.e. the polynomial coefficient values $a_i$ and its trajectory duration time $T$. User speeds behavior at the beginning $t_0$ and at the end of a movement $t_f$ are set to zero.

$$v(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{8}$$

It is assumed that the user reaches its maximal characteristic speed in the displacement middle time $t_m = \frac{t_f - t_i}{2}$. To provide a more realistic movement, we add to the trajectory function a time delay considering the patient reaction time $\tau$ and a position deviation $e = N(m,s)$. Assuming a normal distribution $N()$, with mean value $m$ a standard deviation $s$.

Then we get analytically the following coefficient values:

$$a_0 = \frac{-27d^2e + 27d^2p_i + 36v^2\tau^2 + 16v^3\tau^3}{27d^2} \tag{9}$$

$$a_1 = -\frac{8(3dv^2\tau + 2v^3\tau^2)}{9d^2} \tag{10}$$

$$a_2 = \frac{4(3dv^2 + 4d^3\tau)}{9d^2} \tag{11}$$

$$a_3 = -\frac{16v^2}{27d^2} \tag{12}$$

As well the movement elapsed time:

$$T = \frac{3d}{2v} \tag{13}$$

Comparing an ideal trajectory with the user trajectory it is possible to generate behaviors where the user reaches the target in advance and waits there for the next target and it is also possible to generate situations when the virtual user do not reach the target. Fig. 7 presents a situation when the user reaches the target in advance, while Fig. 8 shows a experiment when the target is not reached. The Algorithm 2 displays the overall operation of the player

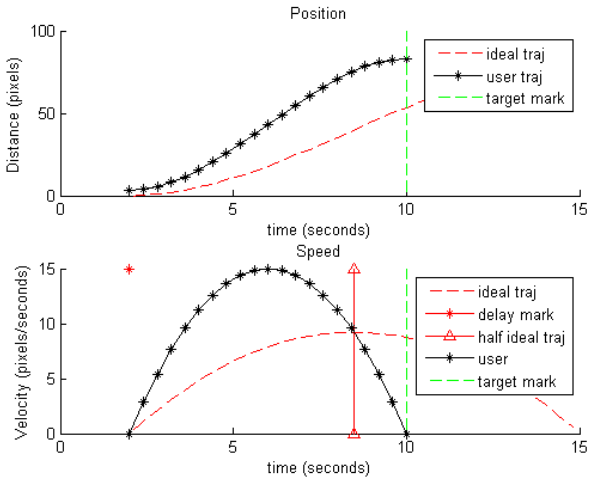


Fig. 7. Example of a Simulated user motion compared with the Ideal Trajectory. User is arriving at target position ahead of time

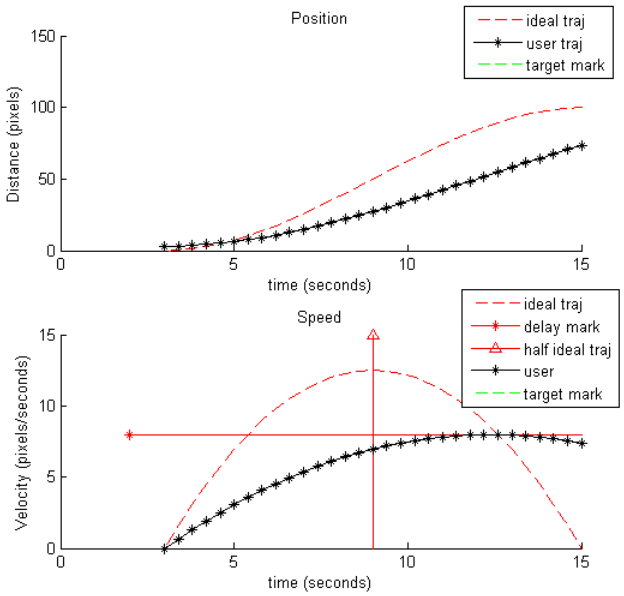


Fig. 8. Example of a Simulated user motion compared with the Ideal Trajectory. User is not able to reach the target position

## V. Experiments and Results

The goal of the first experiment was to verify the ability of the proposed EA in adjusting the difficulty level of the game to match a player skill. For this purpose, we ran a simulation of one thousand game matches for three simulated players, each one with a constant ability level (i.e., the players skills do not improve from a game to another), as explained in Section IV-E. The EA is used to adjust the distance and speed of the target in playing time. In this experiment the coefficients, that were introduced in section IV-B, are set to the same value $K_d = K_s = K_e = K_c = 1$. The behavior of the fitness value over the generations is presented in Fig. 9, and

**Algorithm 2** Algorithm for player model

**Input:** $x_{player}, s_{player}, x_{goal}$ and $t_{goal}$
**Output:** $x_{player}$ and $t_{player}$

$d \leftarrow |x_{goal} - x_{player}|$
$a_0 \leftarrow \frac{-27d^2e+27d^2x_{player}+36v^2\tau^2+16v^3\tau^3}{27d^2}$
$a_1 = -\frac{8(3ds^2_{player}\tau+2v^3\tau^2)}{9d^2}$
$a_2 = \frac{4(3ds^2_{player}+4d^3\tau)}{9d^2}$
$a_3 = -\frac{16s^2_{player}}{27d^2}$
$t_{player} \leftarrow \frac{3d}{s_{player}}$
**if** $(t_{player} > t_{goal})$ **then**
  $t \leftarrow t_{goal}$
**else**
  $t \leftarrow t_{player}$
**end if**
$x_{player} \leftarrow x_{player} + a_0 + a_1 t + a_2 t^2 + a_3 t^3$

game difficulty, defined as the sum of $d$ and $s$ (game challenge increases as the distance to the target or its speed increases) for the chromosome with higher fitness in each generation (200 generations for one thousand matches and an EA with a population size 5).

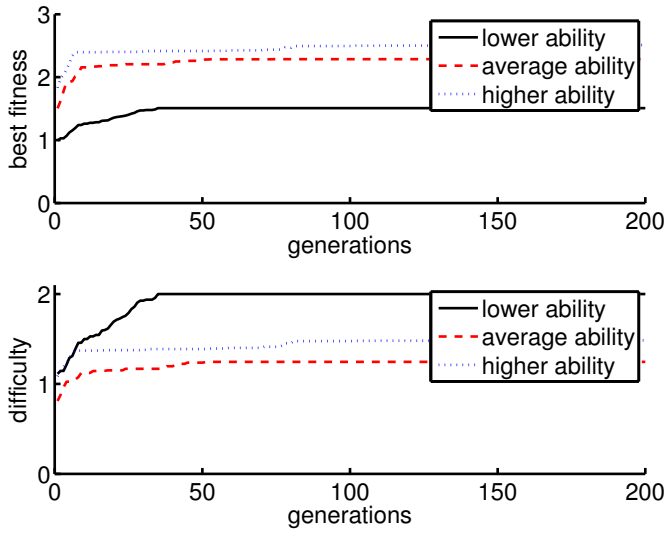


Fig. 9. Fitness value and difficulty for the chromosome with higher fitness in each generation of the EA, obtained in an experiment using $K_d = 1$, $K_s = 1$, $K_e = 1$ and $K_c = 1$.

Examining the graphic in Fig. 9, it is possible to observe that the fitness value increases right from the beginning of the experiment, showing that an adaptation occurs, as expected. We can also observe that difficulty converges to specific values. We expected that the value for which the difficulty converges would reflect the player ability, so the EA would be correctly adjusting the game difficulty. However, the results shows that this is not the case. Tests made with a player that presents less ability converges to a higher difficulty level than the tests realized with more skilled players. In fact, it converges to the highest difficulty value that our game may present ($d = 1$ and $s = 1$). Since the percentage of matches won by the player (definition: a player wins the match when it reaches the target) with lowest ability was very low (28%) when compared to the other simulated players (66% and 67%), we verified that setting the coefficients ($K_d$, $K_s$, $K_e$ and $K_c$ to one is not adequate. This allows chromosomes with higher $d$ and $s$ values to show good scores, even when the player does not win. To verify this problem, we repeated the same experiment 30 times, and calculated the average winning percentage for each simulated player along with the standard deviation. Fig. 10 presents the results, which reinforces our assumption.

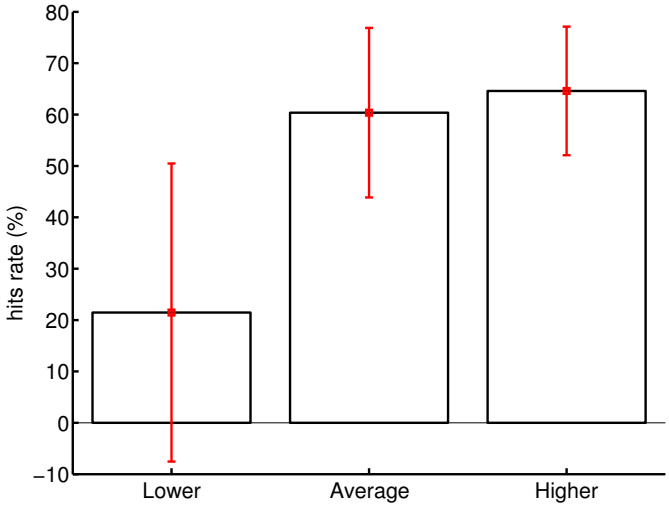


Fig. 10. Hits rate in an experiment using $K_d = 1$, $K_s = 1$, $K_e = 1$ and $K_c = 1$.

To overcame this problem the experiment were repeated with $k_d = k_s = k_c = 1$ and $k_e = 10$. This change increases the importance of the player miss event in the fitness function. The corresponding results are shown in Fig. 11. Here one may observe that the difficulty level was adjusted according to the player ability, as expected. This experiment was also repeated 30 times and the hits rate was calculated. The results are presented in Fig. 12 and they indicate that the hits rate for the player with the lowest ability increased when compared with the results presented in Fig. 10.

The results reinforce the importance of setting the coefficients in the fitness function ( 1), so that the EA can properly adjust the game difficulty. The game properly adjusts the difficulty when it balances the challenge with player skills, so that he or she doesn't get bored by matches that are too easy, nor anxious, when the game is too difficult, as explained in Section III.

To select coefficients that allow the game to achieve the so called "flow state", we ran tests using all possible combinations of values from the set (1, 2, 4, 8, 16) for $k_d$, $k_s$, $k_e$ and $k_c$, totaling 625 different scenarios ($5^4$). We ran the test 30 times for each scenario, using a thousand game rounds for each test, and for each simulated player (with low, average and

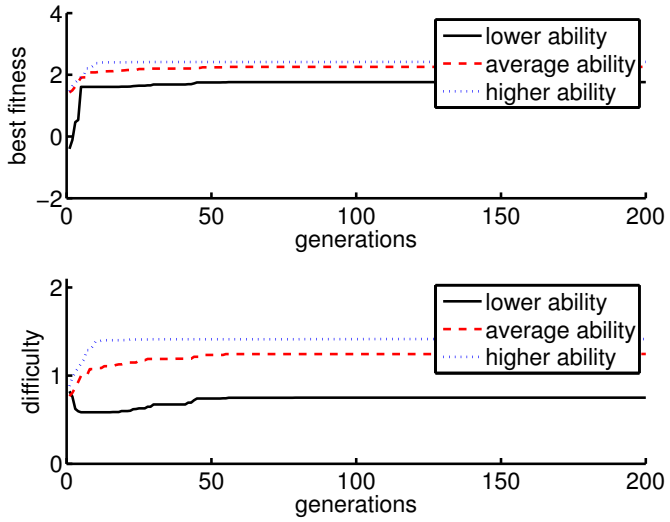


Fig. 11. Fitness value and difficulty for the chromosome with higher fitness in each generation of the EA, obtained in an experiment using $K_d = 1$, $K_s = 1$, $K_e = 10$ and $K_c = 1$.

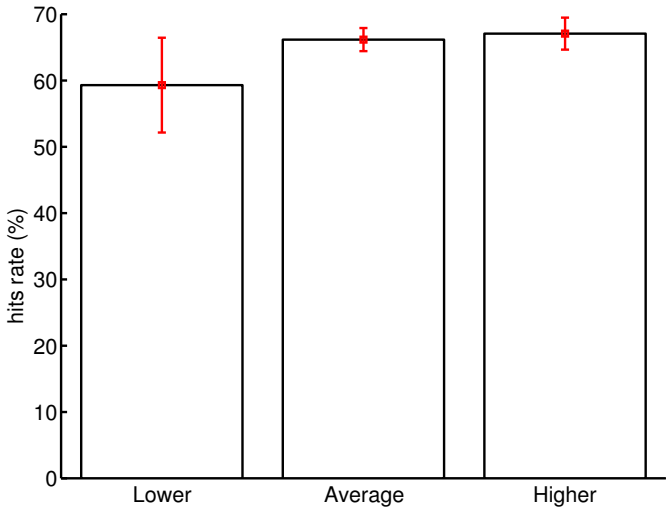


Fig. 12. Hits rate in an experiment using $K_d = 1$, $K_s = 1$, $K_e = 10$ and $K_c = 1$.

high ability levels). Next, we calculated the average hit rate for the last 50 game rounds, which represents the player skill, and the normalized average difficulty level (average of $(\frac{d+s}{2})$ from the last 50 rounds. Then, each scenario was represented by a point on the scatter plot in Fig. 13.

Each point in Fig. 13 represents a tested scenario for a player and a set of fitness coefficient values. As three different simulated players are adopted, a specific set of coefficient values will result in three different points in the scatter plot. The points of interest are the ones that are closer to the diagonal line, therefore, far from the boredom and anxiety areas and inside the flow channel. For this reason, a rank to find the scenarios which results in better choices for each player type was created. This rank was created calculating the distance of each point to the diagonal line. In this way, points

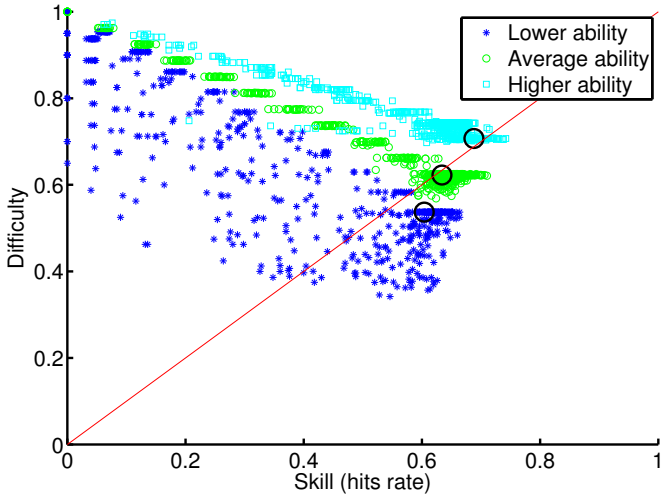


Fig. 13. Scatter plot graphic representing the impact of the coefficient settings in the provided game challenge according to the player skill.

that are closer to the line represent better choices. However, it is important that the fitness coefficient values are appropriate for all different player ability levels.

To find one set of coefficient values that presents good results for all three simulated player, a common set among the top ranked scenarios for each player was adopted. One set of values that was common in the first 81 ranked scenarios of each player was identified, i.e. $K_d = 1$, $K_s = 16$, $K_e = 2$ and $K_c = 16$, represented by the circled points in 13. Then, the first experiment in this section was repeated using this coefficients. As expected, the results presented in Fig. 14 show a coherent difficulty level adjustment with respect to the players skills.

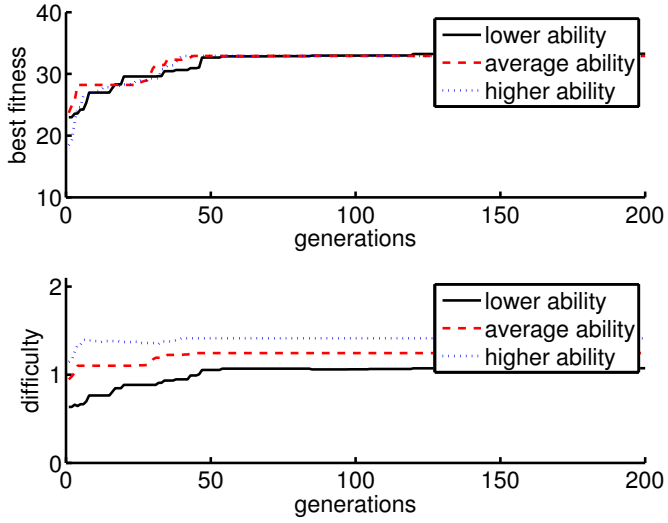


Fig. 14. Fitness value and difficulty for the chromosome with higher fitness in each generation of the EA, obtained in an experiment using $K_d = 1$, $K_s = 16$, $K_e = 2$ and $K_c = 16$.

## VI. Conclusion and Future Work

This paper shows an example of an Evolutionary Algorithm (EA) implementation as a way to adjust the difficulty level in playing time, applied to a robotic rehabilitation system. The Catcher and Happy Fish rehabilitation games were used as implementation examples. To accelerate the development of a difficulty adjustment system, virtual user profiles were created and the objective function (fitness) for difficulty balancing could be exposed to exhaustive tests, verifying the algorithm efficiency.

It is possible to observe in the simulation results that the proposed EA is capable of adjusting the game difficulty level in accordance to the simulated player ability. It is also shown the importance of adjusting the coefficients of the proposed fitness equation, and a corresponding approach, based on [9], was proposed that implements this adjustment.

Future work includes: *i*) the development of an automatic method for adjusting the fitness coefficients; *ii*) the implementation of additional tests with simulated players with dynamic ability level inside a game session. This may also improve the quality of the simulated player representation, allowing also the change skills level due to the exercise playing; and also the simulation of players with impairments *iii*) Perform clinical tests with stroke subjects and evaluate the impact of our proposal on rehabilitation sessions.

## Acknowledgment

The authors acknowledge the financial support from FAPESP, CNPq and CAPES.

## References

[1] G. C. Burdea, "Virtual rehabilitation–benefits and challenges." *Methods of information in medicine*, vol. 42, no. 5, pp. 519–23, 2003. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/14654886

[2] G. A. P. Caurin, A. A. G. Siqueira, K. O. Andrade, R. C. Joaquim, and H. I. Krebs, "Adaptive strategy for multi-user robotic rehabilitation games," *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, no. 1, pp. 1395–1398, 2011.

[3] K. D. O. Andrade, G. Fernandes, G. A. P. Caurin, A. A. G. Siqueira, R. A. F. Romero, and R. D. L. Pereira, "Dynamic player modelling in serious games applied to rehabilitation robotics," *Proceedings - 2nd SBR Brazilian Robotics Symposium, 11th LARS Latin American Robotics Symposium and 6th Robocontrol Workshop on Applied Robotics and Automation, SBR LARS Robocontrol 2014 - Part of the Joint Conference on Robotics and Intelligent Systems,*, pp. 211–216, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7024283

[4] H. I. Krebs, J. J. Palazzolo, L. Dipietro, M. Ferraro, J. Krol, K. Rannekleiv, B. T. Volpe, and N. Hogan, "Rehabilitation robotics: Performance-based progressive robot-assisted therapy," *Autonomous Robots*, vol. 15, no. 1, pp. 7–20, 2003.

[5] K. De Jong, *Evolutionary Computation: A Unified Approach*, 2006. [Online]. Available: http://mitpress.mit.edu/0262041944

[6] N. Hogan, "Impedance control: An approach to manipulation," *American Control Conference, 1984 IS - SN - VO -*, no. March, pp. 304–313, 1985. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=4788393{\textbackslash}npapers2://publication/uuid/B3C531B3-9D3B-4A24-B579-115FB24F446B

[7] K. O. Andrade, G. G. Ito, R. C. Joaquim, B. Jardim, A. a. Siqueira, G. a. Caurin, and M. Becker, "A robotic system for rehabilitation of distal radius fracture using games," *2010 Brazilian Symposium on Games and Digital Entertainment*, pp. 25–32, 2010.

[8] K. De O. Andrade, G. Fernandes, J. Martins, V. C. Roma, R. C. Joaquim, and G. A. P. Caurin, "Rehabilitation robotics and serious games: An initial architecture for simultaneous players," in *ISSNIP Biosignals and Biorobotics Conference, BRC*, 2013, pp. 1–6.

[9] M. Csikszentmihalyi, *Flow - The Psychology of Optimal Experience*, first edit ed. Harper Perennial, 1990, vol. 1. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20{\&}path=ASIN/0060920432

[10] R. Koster, *A Theory of Fun for Game Design*. Paraglyph Press, 2013.

[11] P. Spronck, M. Ponsen, I. Sprinkhuizen-Kuyper, and E. Postma, "Adaptive game ai with dynamic scripting," *Machine Learning*, vol. 63, no. 3, pp. 217–248, 2006.

[12] K. O. Andrade, A. E. A. Silva, and M. K. Crocomo, "Um algoritmo evolutivo para a adaptação de npcs em um jogo de ação," 2004.

[13] a. F. Kuri-Morales and J. Gutiérrez-García, "Penalty function methods for constrained optimization with genetic algorithms: A statistical analysis," *MICAI 2002: Advances in Artificial Intelligence*, pp. 187–200, 2002.

[14] J. Yoon, H.-S. Park, and D. L. Damiano, "A novel walking speed estimation scheme and its application to treadmill control for gait rehabilitation." *Journal of neuroengineering and rehabilitation*, vol. 9, no. 1, p. 62, 2012. [Online]. Available: http://80.apps.webofknowledge.com.dialog.cvut.cz/full{_}record.do?product=WOS{\&}search{_}mode=GeneralSearch{\&}qid=1{\&}SID=N2DjFh3vwWlHUvywyeS{\&}page=7{\&}doc=61

[15] N. Hogan and D. Sternad, "On rhythmic and discrete movements: Reflections, definitions and implications for motor control," *Experimental Brain Research*, vol. 181, no. 1, pp. 13–30, 2007.

[16] B. R. Rohrer, "Evolution of movement smoothness and submovement patterns in persons with stroke by," Ph.D. dissertation, 2002. [Online]. Available: http://hdl.handle.net/1721.1/29256