Chapter 9
An Analysis of the Genetic Evolution of a Ball-Beam Robotic Controller Based on a Three Dimensional Look up Table Chromosome

Mark Beckerleg and John Collins

Abstract This chapter describes how a robotic controller based on a 3-dimensional lookup table was used to control a ball balancing beam system. The evolved motion of the beam and the corresponding chromosome is analysed. The 3 system states of the ball and beam were translated by the lookup table into a motor speed and direction which maintained the ball in balance. The ball-beam states included the ball position, ball speed, and beam position. The reproduction method used 2-point crossover with a mutation rate of 2 percent. The selection method was tournament, and the population size was 100 individuals. Successful evolution was achieved on 4 lookup tables, each containing different maximum motor speeds. Each evolved lookup table was able to maintain the ball in balance for more than 5 minutes.

Keywords Artificial intelligence · Ball balancing beam · Ball-beam · Evolvable robotics · Evolution · Evolving lookup tables · Genetic algorithms · Lookup table-based controllers · Metaheuristic optimization · Robotics

M. Beckerleg (✉) · J. Collins
School of Engineering, AUT University, New Zealand. AUT City Campus, Private Bag 92006, Auckland 1142, NZ
e-mail: mark.beckerleg@aut.ac.nz

J. Collins
e-mail: john.collins@aut.ac.nz

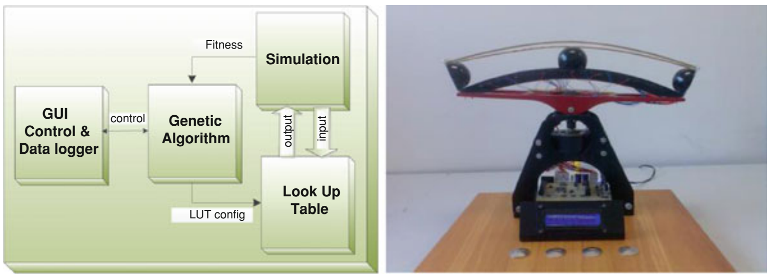


Fig. 9.1 Block representation and connections between the 4 units that were implemented on a computer and the physical beam that the simulation was modeled on

9.1 Introduction

An investigation was undertaken to determine if a genetic algorithm (GA) could be used to generate a ball-beam controller by evolving a population of lookup tables (LUT) employed to control the motion of the beam. The system that was developed (Fig. 9.1) consisted of 4 parts: (i) the graphical user interface (GUI), which displayed the motion of the ball and beam with control and data logging capabilities, (ii) the GA, which generated the final controller by evolving a population of LUTs, (iii) the simulation, which modeled the characteristics of the ball-beam system and (iv) the LUT, which provided the new beam motor speed and direction depending on the current ball-beam state [1].

In this experiment a curved beam was chosen rather than a standard straight beam, as the curved beam provided a more complex control system, and made the ball control more challenging.

The mathematical model and corresponding simulation was based on a physical ball-beam system that was developed at AUT University as part of a student project (Fig. 9.1). The position of the ball was determined using 19 infrared detectors while the motion of the beam was controlled by a stepper. The pulse rate of the stepper motor controlled the angular velocity of the beam. The maximum pulse rate was 125 pulses per second and the angular movement of the beam per pulse was 0.22 degrees. This resulted in a maximum angular velocity of the beam of 27.5 degrees per second.

9.2 Background

The ball-beam has historically been used to demonstrate control systems, owing to its non-linear dynamics and behaviour. It has now become a benchmark for research in this field. Studies using the ball-beam system have been implemented using a range of control systems such as proportional integral differential (PID)

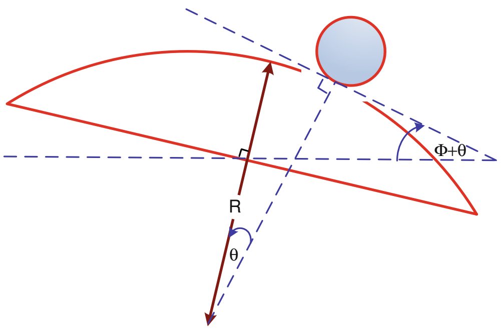


Fig. 9.2 The ball and beam showing the relationships between the angles and motion

control [2, 3], fuzzy logic [4, 5], and neural networks [6, 7]. Ball-beam controllers have been investigated using GAs. Some examples of evolved robotic controllers are the evolution of rules and classes of a fuzzy logic controller [8, 9], the weightings and connectivity of artificial neural networks [10, 11], and the coefficients of a PID controller [12, 13]. However the authors were unable to find any research into the use of a LUT for a ball-beam controller evolved by a GA.

LUTs have been used in evolution in a variety of ways. Robotic simulation has been replaced by lookup tables thus reducing real-time computation requirements [14, 15]. Cellular automata rules have been configured within a LUT and then evolved to create 2- and 3-dimensional shapes [16]. LUT's have been encoded with simulated DNA sequences and evolved to create robotic motion [17]. LUTs contained within a FPGA functional element have been evolved to create a robotic controller [18]. Finally, the authors have used GAs to evolve LUT based robotic controllers for 2 systems including a mobile inverted pendulum [19] and the gait of a hexapod robot [20].

9.3 Mathematical Model

The beam position (Fig. 9.2), is the angle φ from horizontal, while the ball position is the angle θ from the centre of the beam. Eqs. (9.1) and (9.2) outline the final equations for the ball acceleration. The full derivation for this mathematical model has been described by the authors in other literature [21].

$$\ddot{\theta} = A(\theta + \phi) \tag{9.1}$$

$$A = \frac{g}{R\left(1 + \frac{1}{mr^2}\right)} \tag{9.2}$$

Where

g — gravitational acceleration

I — moment of inertia of the ball
R — radius of curvature of the beam
m — mass of the ball
r — radius of the ball
θ — ball position (angle from the centre)
Ø — beam position (angle from horizontal)
x — ball position
v — ball velocity
b — beam position
a — acceleration of the ball

The value for acceleration (a) of the ball on the beam was determined by physical experimentation, as a factor of the ball position (x) and beam position (b) in Eq. (9.3). Using this acceleration the new ball position can be found dependant on it current velocity (v), acceleration and position as shown in Eq. (9.4), and the new speed of the ball dependant on its current speed and acceleration in Eq. (9.5). The simulation was adjusted to a time period of every millisecond in Eqs. (9.6) and (9.7).

$$a = 12x + 2.8b \tag{9.3}$$

$$x_{new} = x + vt + \frac{at^2}{2} \tag{9.4}$$

$$v_{new} = v + at \tag{9.5}$$

$$x_{new} = x + \frac{v}{10^3} + \frac{12x + 2.8b}{2 \times 10^6} \tag{9.6}$$

$$v_{new} = v + \frac{12x + 2.8b}{10^3} \tag{9.7}$$

9.4 Genetic Algorithm

A genetic algorithm is a metaheuristic optimization method that uses natural selection as a search engine. It acts on a population of individuals or chromosomes which are potential candidate solutions to the problem needing to be solved. Chromosomes are comprised of various forms such as bits, numbers or parameter sequences, depending on the problem. The genetic algorithm is iterative and is comprised of 3 main processes including reproduction, fitness evaluation, and selection. Reproduction is the generation of offspring from the surviving population of chromosomes. It uses 2 genetic operators: (a) crossover, where chromosomes are exchanged between parents, and (b) mutation, where parts of the parents' chromosomes are randomly altered. Fitness evaluation determines how

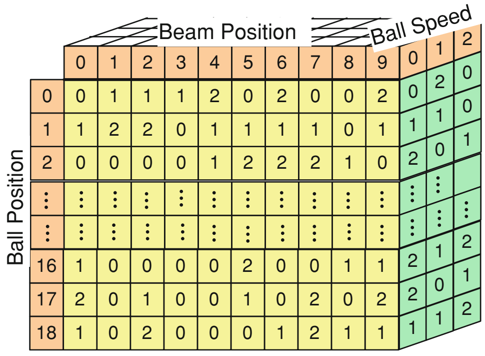


Fig. 9.3 Three dimensional LUT showing 19 ball positions, 10 beam positions, 3 ball speeds

well each chromosome in the population performs as a potential solution to the problem. Selection determines which chromosomes within the population will survive to the next generation based on their fitness.

The steps in a genetic algorithm are: (a) generation of an initial random population of chromosomes, (b) reproduction of offspring, (c) fitness evaluation of each new chromosome, and (d) selection, where the chromosomes with the best fitness are kept. The processes of reproduction, fitness evaluation and selection are repeated until the required fitness is reached or a set number of generations have been completed.

9.4.1 Chromosome

The heart of the controller was a 3-dimensional lookup table (Fig. 9.3). The lookup table contained the motor speed and direction required to drive the motor in such a way as to balance the ball. The 3-dimensions of the lookup table were linked to the ball and beam states. These were ball position (19 inputs), beam position (10 inputs), and ball speed (3 inputs). Several lookup tables were evaluated with a range of motor speeds varying from 2 to 11. The elements of the array were defined as char variables initialized with a randomly generated number quantised into 11 discrete steps ranging from 0 to 250. This enabled each location in the array to describe a motor speed with 5 left speeds, 5 right speeds and 1 stopped. The speed range was reduced when evaluating different speeds by adjusting the threshold so that the motor had limited speeds. For example with a 2 speed range, values below 125 would drive the motor hard left, while values above 125 would drive the motor hard right.

The LUT was used to control the beam's motor depending on the beam states. This was achieved by connecting the simulation's current ball-beam states to the axis of the LUT. The parameter at that location was sent back to the simulation to control the simulation's motor speed and direction.

Table 9.1 Search space within the LUT dependent on the number of motor speeds

Speeds	Search space
2	3.9×10^{171}
3	9.1×10^{271}
5	2.6×10^{398}
11	3.9×10^{593}

The search space that the GA explores was dependent on the total number of combinations that the chromosome can have. This was dependent on the number of locations within the LUT, and the number of speeds that were employed at each location. The experiments were repeated with 4 ranges of motor speeds: 2 (left and right), 3 (left, stopped and right), 5 (2 left, stopped and 2 right) and 11 speeds (5 left, stopped and 5 right). The total search space for each LUT was calculated using Eq. (9.8) and illustrated in Table 9.1. As evident in this table, the search space rapidly increased as the number of speeds increased. The exponent 570 was derived from the size of the LUT ($19 \times 10 \times 3$).

$$\text{Search space} = \text{speeds}^{\text{size of LUT}} = \text{speeds}^{570} \tag{9.8}$$

9.4.2 Reproduction and Selection

The objective of reproduction is to generate new offspring from a population of chromosomes which will have a higher fitness than their parents. The purpose of selection is to decide which offspring and parents to keep, with the goal that the population will move rapidly up the fitness landscape while maintaining enough diversity to bypass local maxima. The 2 parts of reproduction are crossover and mutation. Crossover is a method that is used to split and recombine the chromosomes from 2 or more parents into 1 offspring. Two-point crossover was used in this experiment using the x-axis (ball position) and y-axis (beam position) of the array as the positions within the array to be cut. The first cut points of the crossover were determined by randomly choosing points between 0–18 and 0–9. The end cut points of the crossover were determined by randomly choosing points between the first cut points and the end of the array, 18 or 9. Mutation will randomly alter the parameters within a single chromosome with the purpose to maintain the diversity of the population. It has a low probability of occurring which is typically between 0.1 to 2 percent. In this experiment a mutation rate of 2 percent was chosen, with every individual in the population being mutated after crossover occurred.

The selection pressure is an important parameter in genetic selection. The selection operator has a high selective pressure if it severely reduces the difference between individuals, or a low selective pressure if it allows many different individuals to survive. A low selection pressure will have a slow rate of convergence to the optimum solution and can possibly stagnate, whereas a selection pressure that is too high may get stuck on local maxima due to loss of diversity. The choice of which selection method to use is dependent on what type of problem is to be solved, with each method having its advantages and disadvantages. Tournament selection with a group size of 2 was employed for this experiment. The choice of this group size was primarily to maintain a large diversity within the population with a moderate selection pressure.

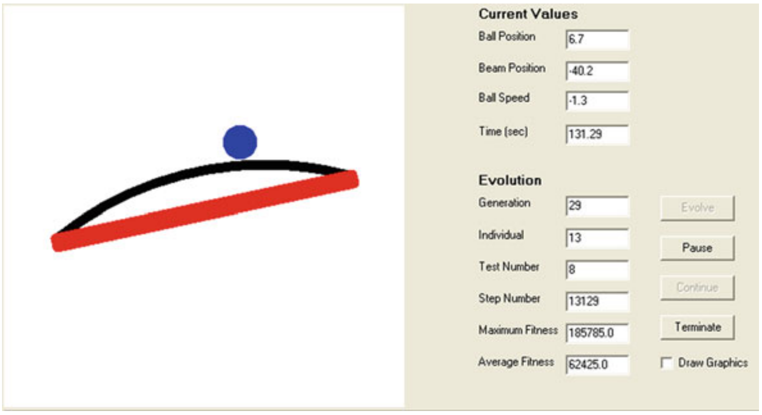
9.4.3 Fitness Criteria

Each chromosome was evaluated by the simulation to see how well it functioned. This fitness evaluation was then used by the genetic selection to determine if the chromosome would be kept. The maximum fitness that a chromosome could have was 420 seconds. This was made up from 7 simulation runs with the beam positioned horizontally and the ball placed at rest on various positions on the beam. Each simulation run would last either until 60 seconds had passed, or until the ball reached a beam end stop.

9.5 Simulation

Using a genetic algorithm to evolve a physical robot controller is difficult due to the potential damage to the robot and its environment. As well, the complexity of the robot's actions has a large search space requiring a large number of generations before a suitable controller can be evolved. This is time consuming if performed in real time on an actual robot. To overcome these problems, evolutionary robotic genetic algorithms are normally performed using a software simulation of the robot. Once a suitable solution is found it can be transferred to the actual robot. However creating a simulation for a robot means modelling the real world which can never be entirely accurate, thus the final solution will carry with it the flaws in the simulation.

In this experiment the new ball position and speed were determined by the simulation every millisecond, based on the new beam position and the previously described Eqs. (9.6) and (9.7). Two maximum beam velocities of 22.7 and 45.4 degrees per second were evaluated. The motor speed and direction was converted by the simulation into a new beam position, and from this, the new ball position and speed were calculated. The motor speed and direction produced by the lookup table, was converted by the simulation into a new beam position. From this, a new ball position and speed were determined by the simulation based on the previously described Eqs. (9.6) and (9.7). The new ball-beam states were then feedback to the lookup table to determine the next motor speed and direction. The simulation recalculated the ball position and speed every millisecond. Two maximum beam velocities of 22.7 and 45.4 degrees per second were evaluated.


Current Values

Ball Position	6.7
Beam Position	-40.2
Ball Speed	-1.3
Time (sec)	131.29

Evolution

Generation	29
Individual	13
Test Number	8
Step Number	13129
Maximum Fitness	185785.0
Average Fitness	62425.0

Fig. 9.4 The graphical user interface

9.6 Graphical User Interface

The GUI (Fig. 9.4) displayed the current ball position, ball speed, beam position, and the current time that the ball had remained balanced. A dynamic visual representation of the ball and beam in motion could be turned on or off, allowing the user to see how the ball and beam were responding at various stages of the evolutionary process. The visual representation was normally turned off, as when it was on, the evolutionary process was slowed to real time. Evolutionary control buttons were used to start, pause, and terminate the evolutionary process. The evolutionary parameters of generation number, current individual under test, average fitness of the population, and maximum fitness that had been reached was provided. A text display showed the number of speed settings, the maximum beam speed, the maximum fitness, average population fitness, generation number and time taken for the evolutionary process. These values were stored for later analysis.

9.7 Results

Two ranges of experiments were performed with 2 maximum motor speeds equating to a maximum beam angular velocity of 22.7 and 45.4 degrees per second. For each experiment, 4 ranges of motor speeds (2, 3, 5 and 11 speeds) were evaluated. Each simulation was repeated 7 times with a different ball start positions lying between ±12 degrees from the top of the beam. The simulation was run until either the ball reached a beam end stop or 60 seconds had passed. This gave a maximum fitness of 420 seconds.

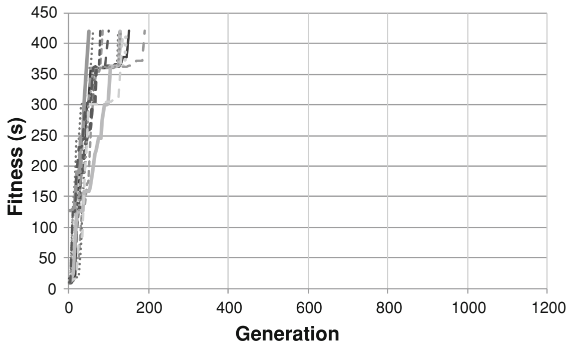


Fig. 9.5 Two motor speeds, maximum angular velocity of 22.7 ° per second

9.7.1 Evolved Motion of the Ball

The relationship between the fitness of number of generations and corresponding fitness of the best individual for 4 different maximum motor speeds is shown (Figs. 9.5, 9.6, 9.7, and 9.8).

The first observation of these results is that the LUT using only 2 motor speeds evolved in a shorter number of generations and time. This was due to 2 factors: (1) the smaller search space of the chromosome, and (2) no requirement for multiple speeds with an associated smoother response of the beam was built into the fitness.

The second observation was that the fitness increased in discrete steps, improving rapidly until it reached a plateau at a fitness of approximately 320 and 360 seconds. On investigation of the ball motion, it was found that it was difficult to capture the ball when it was started at the furthest position from the centre of the beam. To avoid failure at these start positions, the beam was required to move at maximum angular velocity in the opposite direction. This problem was more apparent in a LUT with 5 or 11 motor speeds as there was a greater possibility that the maximum beam angular velocity would not be used.

The motion of the ball-beam during the evolution process could be monitored on the GUI. It could be seen that the ball beam motion evolved through 4 stages. These were: (1) the beam remained stationary and the ball simply fell to an end stop; (2) the beam would react once, reversing the motion of the ball. However the ball would then fall to the opposite end stop; (3) the ball would be captured in 1 position, where the beam would perform an oscillation motion causing the ball to stay within 2 points. This pattern would last for between 5 and 10 seconds, but eventually the ball would break free and reach an end stop; (4) finally the beam was able to trap the ball between 2 points for the full 60 seconds test. This method of balancing by capturing the ball between 2 points by oscillating the beam was observed in all motor speed ranges.

It was observed that the ball tended to be captured near either end of the beam. This seemed unusual as it is a more risky position than a ball captured near the center of the beam. This was due to the design of the beam's sensor location, with

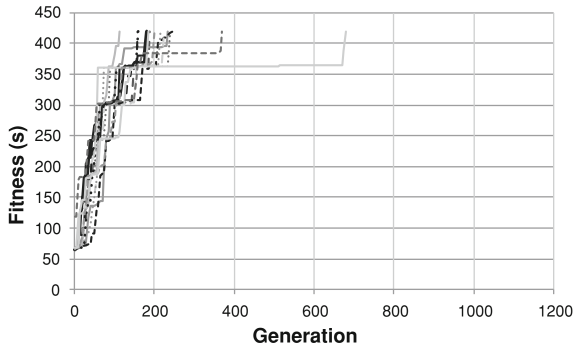


Fig. 9.6 Three motor speeds, maximum angular velocity of 22.7 ° per second

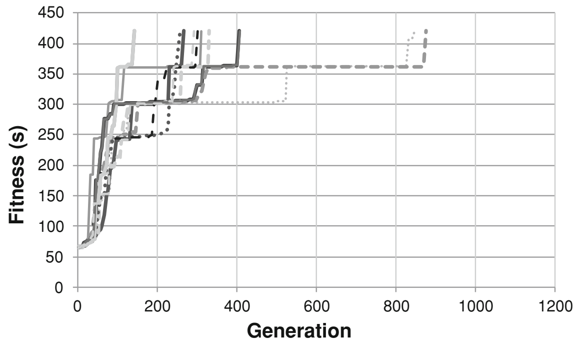


Fig. 9.7 Five motor speeds, maximum angular velocity of 22.7 ° per second

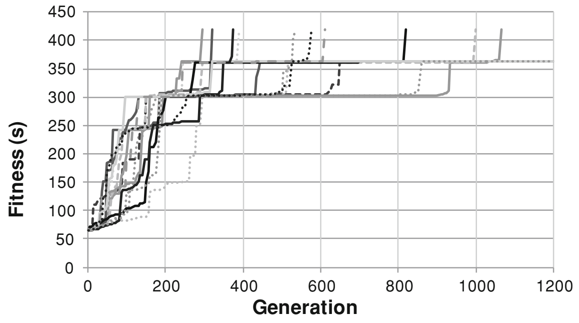


Fig. 9.8 Eleven motor speeds, maximum angular velocity of 22.7 ° per second

more sensors placed near ends of the beam as it was thought that this was a more important position. Unintentionally this gave the simulation a more accurate position of the ball and its speed at this point. Subsequently the evolved controller used the end locations to balance the ball.

A peculiarity of using a robotic simulation was observed when the ball was started at rest in the center of the beam. The evolved chromosome learnt to keep the motor off, thus achieving a perfect score for that run. This behaviour of course would not work well with a real ball-beam system.

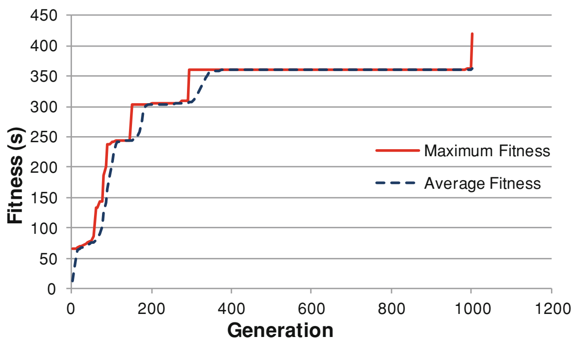


Fig. 9.9 Comparison between the maximum and average fitness, with 11 motor speeds and a maximum angular velocity of 22.7 ° per second

9.7.2 Evolved Chromosome

When the best chromosomes from several successful evolutionary runs were compared, it was found that each chromosome was different, producing a varied pattern for the beam and ball motions. This variation in successful chromosomes was due to the initial random population and the multiple pathways that the ball and beam could interact with the LUT. This variation in chromosomes was compounded by the fact that the evolution stopped once a successful pattern had been found.

Most successful simulation runs did not use a large part of the LUT because the ball would simply be moved to a position on the beam, and be kept in place by beam oscillations.

The maximum and average fitness of a typical run shows that the maximum fitness increased in abrupt steps, and then reached a plateau (Fig. 9.9). The average fitness would then converge with the maximum fitness at each plateau. It was thought that the population had converged at these points allowing only mutation to modify the chromosome and its associated fitness. However an investigation of the individual chromosomes and associated ball-beam motion showed that there was still population diversity, and that the reason for the convergence of fitness was due to the difficulty of evolving a chromosome that could start the ball at the edge of the beam.

9.7.3 Comparison of Two Maximum Motor Speeds

Multiple experiments were run using the 2 maximum beam angular velocities of 22.7 and 45.4 degrees/second with speeds ranging from 2 to 11 settings. A comparison of these results is shown in Table 9.2, which details the average fitness, number of generations and time the evolution was in progress at the end of the evolution. From this table it can be seen that the faster motor and minimum number of motor speeds had the best results in terms of the number of generations

Table 9.2 Comparison of the average fitness, average number of generations and the average time taken to evolve

22.7 ° per second			45.4 ° per second		
Generation	Av fitness	Time(s)	Generation	Av fitness	Time(s)
118	347726	197	42	268456	35
268	364240	592	56	327891	76
398	357240	3624	98	351811	297
861	359427	25794	103	349563	467

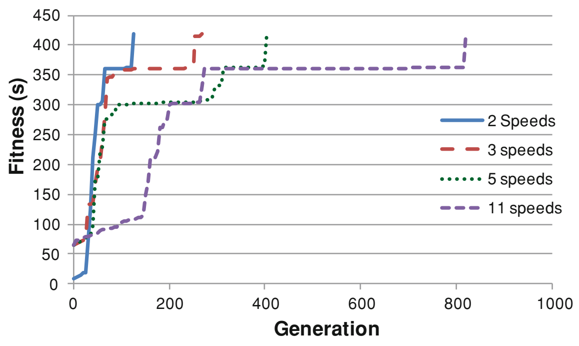


Fig. 9.10 Four motor speeds with maximum beam angular velocity of 22.7 ° per second

and the time taken to come to a successful evolution. It was noted that the time taken for the 5 and 11 motor speeds to successfully evolve was also acceptable despite the much larger search space. This was due to the constrained motion of the beam and the path that the ball took, with only a limited part of the chromosome being used for the beam control.

A comparison of the 4 motor speeds within each maximum motor pulse rate is shown (Figs. 9.10 and 9.11). From these graphs it can be seen that doubling the beam angular velocity had a significant improvement on the ability of the system to evolve, especially at the 5 and 11 speed range. The fitness plateau at 320 and 360 seconds can clearly be seen. All the solutions had difficulty with either one or both of the extreme starting points.

9.8 Conclusion

These experiments have shown that a 3-dimensional lookup table used as a controller for a ball-beam system can be successfully evolved to allow the ball to remain in balance for a total of 5 minutes. The motion of the ball and beam are unique for each successful evolved chromosome. The experiments were performed on 2 maximum beam angular velocities, and a range of 2, 3, 5 and 11 motor speeds. It was found that the higher beam velocity and lower number of motor speeds had the best evolutionary performance.

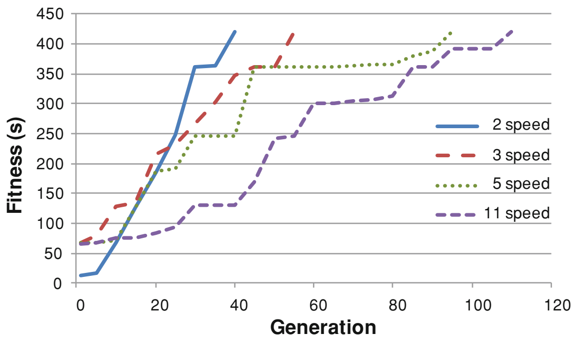


Fig. 9.11 Four motor speeds with maximum beam angular velocity of 45.4 ° per second

Future research will involve modifying the fitness evaluation such that the behaviour of the ball is more accurately controlled, for example to bring the ball to rest at a set point in the shortest amount of time. This could then be compared with other controllers such as proportional, differential and integral control. In addition further research will explore how the evolved chromosome can be moved from simulation to a physical beam.

References

1. Beckerleg, M, Collins J (2011) Evolving a three dimensional lookup table controller for a curved ball and beam system, lecture notes in engineering and computer science: proceedings of The World Congress on Engineering and Computer Science 2011, WCECS 2011, San Francisco, 19–21 Oct 2011, pp 360–366
2. Ka C, Nan L (1987) A ball balancing demonstration of optimal and disturbance-accomodating control. Control Syst Mag IEEE 7(1):54–57
3. Gordillo, F et al (2002) On the ball and beam problem: regulation with guaranteed transient performance and tracking periodic orbits, in proceedings of the international symposium on mathematical theory of networks and systems University of Notre Dame
4. Dadios EP et al (2000) Vision guided ball-beam balancing system using fuzzy logic. In: Industrial Electronics Society, IECON. 26th Annual Confjerence of the IEEE
5. Iqbal J et al (2005) Implementing ball balancing beam using digital image processing and fuzzy logic. In: Electrical and computer engineering, Canadian conference
6. Ng, KC, Trivedi MM (1996) Neural integrated fuzzy controller (NiF-T) and real-time implementation of a ball balancing beam (BBB). In: Proceedings IEEE international conference on Robotics and automation.
7. Eaton PH, Prokhorov DV, Wunsch DC II (2000) Neurocontroller alternatives for fuzzy ball-and-beam systems with nonuniform nonlinear friction. Neural Netw IEEE Trans 11(2):423–435
8. Tettamanzi, AGB (1995) An evolutionary algorithm for fuzzy controller synthesis and optimization. In: IEEE international conference on systems, man and cybernetics, intelligent systems for the 21st century.
9. Hoe Sung (2001) A self-organized fuzzy controller for wheeled mobile robot using an evolutionary algorithm. Ind Electron IEEE Trans 48(2):467–474
10. Bianco R, Nolfi S (2003) Evolving the neural controller for a robotic arm able to grasp objects on the basis of tactile sensors, 2829 edn. Lecture notes in computer science 375–384

11. Kim K-J, Cho S-B (2001) Dynamic selection of evolved neural controllers for higher behaviors of mobile robot. In: Proceedings 2001 IEEE international symposium on computational intelligence in robotics and automation, 2001.
12. Yi Z, Xiuxia Y (2004) Design for beam-balanced system controller based on chaos genetic algorithm. In: Information acquisition: proceedings, international conference
13. Pedersen, GKM, Butz MV (2010) Evolving robust controller parameters using covariance matrix adaptation: proceedings of the 12th annual conference on genetic and evolutionary computation, ACM, Portland, pp 1251–1258
14. Lund HH, Hallam J (1997) Evolving sufficient robot controllers. In: Evolutionary computation, IEEE international conference
15. Lund HH (2001) Co-evolving control and morphology with LEGO robots. In: Proceedings of workshop on morpho-functional machines
16. Chavoya A, Duthen Y (2006) Using a genetic algorithm to evolve cellular automata for 2D/3D computational development. In: Proceedings of the 8th annual conference on genetic and evolutionary computation, ACM, Seattle, pp 231–232
17. Greenfield G (2008) Evolved look-up tables for simulated DNA controlled robots. In: Proceedings of the 7th international conference on simulated evolution and learning, Springer-Verlag, Melbourne, pp 51–60
18. Z, RY et al (2002) Evolving FPGA-based robot controllers using an evolutionary algorithm, in first international conference on artificial immune systems
19. Beckerleg M, Collins J (2007) An analysis of the chromosome generated by a genetic algorithm used to create a controller for a mobile inverted pendulum. Studies in computational intelligence. 76
20. Currie J, Beckerleg M, Collins J (2008) Software evolution of a hexapod robot walking gait. In: 15th international conference on Mechatronics and machine vision in practice, M2VIP 2008.
21. Beckerleg M, Collins J (2008) Evolving electronic circuits for robotic control. In: 15th international conference on mechatronics and machine vision in practice. Auckland, New Zealand