MACHINE LEARNING THROUGH SELF GENERATING PROGRAMS

H. G. LUBBE AND B. J. KOTZE

ABSTRACT

People have tried different ways to make machines intelligent. One option is to use a simulated neural net as a platform for Genetic Algorithms. Neural nets are a combination of neurons in a certain pattern. Neurons in a neural net system are a simulation of neurons in an organism's brain. Genetic Algorithms represent an emulation of evolution in nature. The question arose as to why write a program to simulate neurons if a program can execute the functions a combination of neurons would generate. For this reason a virtual robot indicated in Figure 1 was made "intelligent" by developing a process where the robot creates a program for itself. Although Genetic Algorithms might have been used in the past to generate a program, a new method called Single-Chromosome-Evolution-Algorithms (SCEA) was introduced and compared to Genetic Algorithms operation. Instructions in the program were changed by using either Genetic Algorithms or alternatively with SCEA where only one simulation was needed per generation to be tested by the fitness of the system.

Key words: Genetic Algorithms, intelligent programs

1. INTRODUCTION

For thousands of years, humans have dreamt of intelligent machines [1, p. 2][2, p. 10]. Quite a few artificial intelligent methods have been developed, almost all showing the ability to learn minor things. As soon as the problem increase in difficulty, these processes prove themselves not to be adequate and intelligent enough to resolve the problem at hand [3, p. 9][4][5, p. 1].

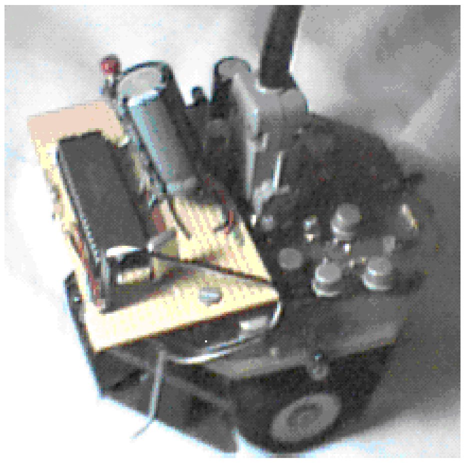


Figure 1 Robot that was build to evaluate the generated programs.

2. GLOBAL LAYOUT OF RESEARCH PROJECT

In Genetic Algorithms as well as SCEA the program was broken up in two parts which interacts with one another [7, pp.1-9]. The first part determines the fitness of a program through simulating the robot movements, its sensors, its controller and its surroundings. The second part generates the new generation of programs to be executed. These two parts were alternated in generating the results. Using a personal computer, the simulation took up to a few minutes to execute while creating a new generation happened without been noticed. To compare Genetic Algorithms with SCEA, only the amount of simulations executed were counted.

3. SINGLE-CHROMOSOME-EVALUATION-ALGORITHMS

To make SCEA possible, two identical generated programs are saved. An instruction in the one program is changed randomly and then this changed program is evaluated by simulating the robot in a maze. The better of the two “new” programs were duplicated so that at the end there are two exact copies in both memory spaces before the process is started again.

Both these methods use a generated variable to indicate how well the evaluated program has performed [6]. The value of this variable is called the fitness of the generated program.

It could be seen as a disadvantage for evaluation purposes that only the distance between the final evaluated position of the robot and the robots destination is given as reference to determine the fitness. The route could be any length, because there are obstacles between the origin and the destination as can be seen in Figure 2. Irrespective of this lack of evaluation the robot had to show the ability to avoid these obstacles.

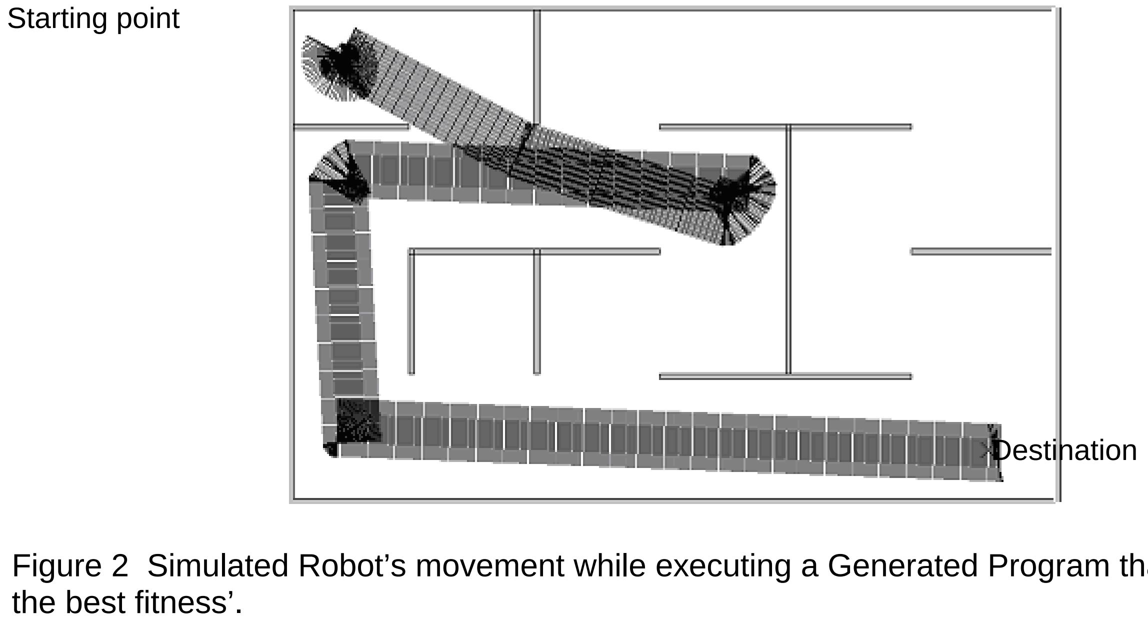


Figure 2 Simulated Robot’s movement while executing a Generated Program that had one of the best fitness’.

4. RESULTS

Because the random functionality is used so much in the generation of programs, the results are also stogastic in nature. For this reason a large amount of simulations were executed for evaluation purposes and the average for some results were calculated. Beacons were set up at some fitness positions and the amount of how many simulations it took to get to these predetermined fitness points. These results and the final fitness' for each situation are shown in Table 1. Two genetic algorithms methods were incorporated. In one instance only mutation is used to generate a new chromosome and in the second instance both mutation and crossover are incorporated.
As can be seen, SCEA took on average 400 simulation before the robot started moving while with the Genetic Algorithms it took an average of 8 300 simulations for a similar response. Over the whole spectrum, with only a few exceptions, the SCEA has proved to be faster than Genetic Algorithms.

It seems that small changes in the situations, for example, introducing noise representing erroneous distance readings to the sensor readings did little to the results. Genetic Algorithms performed better than SCEA when the obstacles in the maze were changed before every simulation.

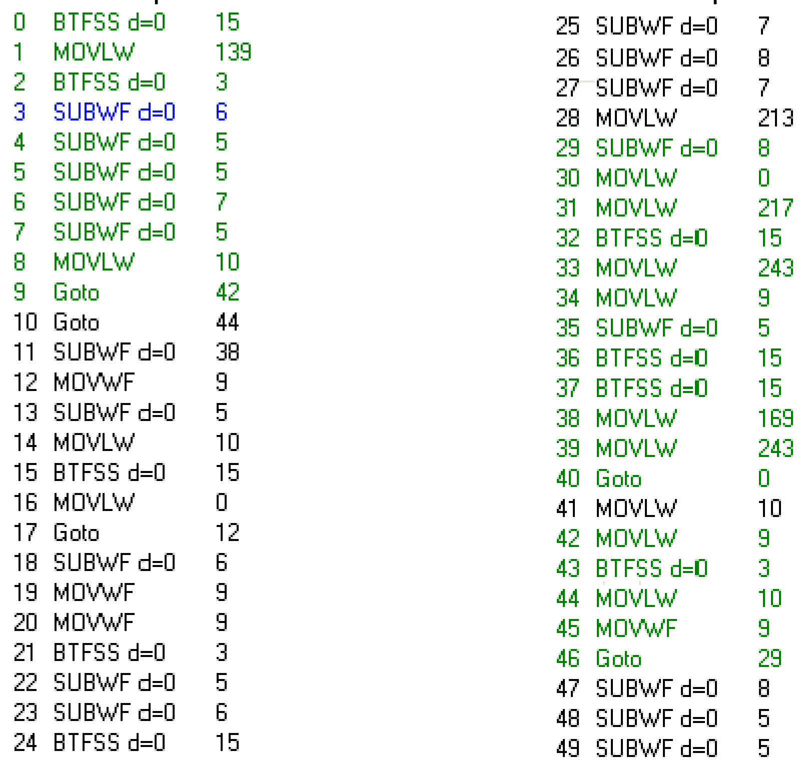
Table 1 The average amount of simulations to get to some fitness point and the final fitness values.

Option	Start	42	96	290	Fin. Fitness.
SCEA	4	9	14	>52	571
Crossover and Mutation	83	96	109	>199	410
Mutation	46	78	93	>202	453

These amounts of simulation values have to be multiplied by a hundred to get the actual results.

In an additional experiment with the option of two destinations, a single bit input to the robot was used to indicate to which one of the two destinations the robot would use as destination. SCEA was unable to adapt to this situation until fitness values was introduced for each destination-input combination.

Figure 3 is an example of a Generated Program for the robot that showed the ability to avoid obstacles. As can be seen the code generated for this virtual controller is similar to the assembler language normally used for the PIC® range of microcontrollers. The instructions shown in black were not executed during the execution of the program. The blue instruction was executed only once. Although the instruction at address 30 was executed, it had no effect on the program as the instruction at address 31 changed the value in the W-register no matter what the circumstance was. Thus many instructions could be changed without influencing the program and later a change to a single instruction could "activate" some of these instructions. The programs that were generated did not seem to have a logical sequence, although with further investigation showed that it will and does work.



```
Address  OPCODE     Operand        Address  OPCODE     Operand
 0       BTFSS d=0  15              25      SUBWF d=0  7
 1       MOVLW      139             26      SUBWF d=0  8
 2       BTFSS d=0  3               27      SUBWF d=0  7
 3       SUBWF d=0  6               28      MOVLW      213
 4       SUBWF d=0  5               29      SUBWF d=0  8
 5       SUBWF d=0  5               30      MOVLW      0
 6       SUBWF d=0  7               31      MOVLW      217
 7       SUBWF d=0  5               32      BTFSS d=0  15
 8       MOVLW      10              33      MOVLW      243
 9       Goto       42              34      MOVLW      9
10       Goto       44              35      SUBWF d=0  5
11       SUBWF d=0  38              36      BTFSS d=0  15
12       MOVWF      9               37      BTFSS d=0  15
13       SUBWF d=0  5               38      MOVLW      169
14       MOVLW      10              39      MOVLW      243
15       BTFSS d=0  15              40      Goto       0
16       MOVLW      0               41      MOVLW      10
17       Goto       12              42      MOVLW      9
18       SUBWF d=0  6               43      BTFSS d=0  3
19       MOVWF      9               44      MOVLW      10
20       MOVWF      9               45      MOVWF      9
21       BTFSS d=0  3               46      Goto       29
22       SUBWF d=0  5               47      SUBWF d=0  8
23       SUBWF d=0  6               48      SUBWF d=0  5
24       BTFSS d=0  15              49      SUBWF d=0  5
```

Figure 3 A generated program created by a program procedure.

5. CONCLUSION

SCEA can adapt to small changes but are unable to adapt to large changes unless a fitness value is given for each situation. With the programs generated that differs from the conventional way a human would write a program, it is still possible that programs could be generated that will function as if it was written conventionally and ideally above a human's ability. It also opens a door to a new way of writing and developing programs by programmers. SCEA is in some cases much faster than Genetic Algorithms and possibly opens a new field in Artificial Intelligence with hopefully better results.

6. REFERENCES

1 McKerrow, P.J. Introduction to Robotics. Singapore. 1991.

2 Scott, P.B. The Robotic Revolution. The Complete Guide. UK. 1984.

3 Wise, E. Applied Robotics. United States of America. 1999.

4 Ross, A. Dynamic Factory Automation. Creating Flexible Systems for Competitive Manufacturing. England. 1992.

5 Andeen G.B. Introduction. Robot Design Handbook.1988.

6 Connell, J.H. & Mahadevan, S. Introduction to Robot Learning. Robot Learning. 1993. pp. 5-6.

7 Man, K.F., Tang, K.S., Kwong, S. and Halang, W.A. Genetic Algorithms for Control and Signal Processing. Great Britain. 1997.