An Evolutionary Approach to Damage Recovery of Robot Motion with Muscles

Siavash Haroun Mahdavi and Peter J. Bentley

Dept. Computer Science, University College London, Gower St. London WC1E 6BT, UK
{mahdavi, p.bentley}@cs.ucl.ac.uk

Abstract. Robots that can recover from damage did not exist outside science fiction. Here we describe a self-adaptive snake robot that uses shape memory alloy as muscles and an evolutionary algorithm as a method of adaptive control. Experiments demonstrate that if some of the robot's muscles are deliberately damaged, evolution is able to find new sequences of muscle activations that compensate, thus enabling the robot to recover its ability to move.

1 Introduction

It has long been the dream of science fiction to have robots capable of self-repair and recovery from damage. While many of the ideas seem plausible (for example, R2D2 repairing other robots in the movie *Starwars*), most rely on conventional methods of replacing damaged mechanisms or switching to redundant systems. Consequently, they require the overheads of spare components or systems and a sufficiently dexterous and knowledgeable repair robot (which in reality is always a human being).

However, these may not be the most efficient approaches of coping with damage. Natural systems recover from damage in very different ways: they either regrow damaged parts of themselves or they adapt their behaviour to compensate for the loss of functionality. A dog may not be designed to walk on three legs, but it can learn to do so very effectively if it must.

In this work we focus on such biologically-inspired ideas. A self-adapting snake (SAS) robot is created, using shape memory alloy as its "muscles". An evolutionary algorithm is then used to evolve a sequence of muscle activations in order to enable the robot to propel itself along. By deliberately damaging some of the muscles of the SAS, we can then assess the capability of evolution to adapt the motion of the robot and use the remaining muscles in a more efficient manner.

2 Background

2.1 Self-Repairing and Shape Memory Alloy Robots

Although evolutionary design [1] is common, little work is evident in the field of self-repairing robotics. Most current research seems to be limited to theory and future

predictions. Bererton and Khosla propose that there will be large self-sufficient robot colonies operating on distant planets. They describe methods where teams of robots could repair each other, and they have done experiments on visual docking systems for this purpose [2]. Michael claims that in the future 'fractal robots' will emerge which can completely change their shape and so if damaged, can reassemble them selves to recover their previous capabilities [10]. Kamimura has built robots that can reconfigure themselves to perform different tasks (as have Dittrich et al [3]), though he has not yet looked into the idea of self-repair [7]. Perhaps the work most similar to that described here was performed by Støy [12], in which a chain robot made of nine modules is robust to signal loss and able to continue effective locomotion. However, unlike the system described here, the controllers are preprogrammed and incapable of learning novel movement strategies after damage to the robot.

The use of smart materials in robotics has already been investigated. For example, Kárník looked at the possible applications of walking robots, which use artificial muscles with smart materials as a drive [8]. Mills has written a book aimed at beginners which teaches them how to make simple eight-legged robots using smart materials as actuators [11]. However, no one has used smart materials and evolutionary algorithms together in a robot before. This work uses nitinol wires as muscles within a robot snake, with the muscle activation sequences evolved using genetic algorithms and finite state machines [5].

2.2 Smart Material

Nitinol, an alloy made of **Ni**ckel and **Ti**tanium, was developed by the **N**aval **O**rdinance **L**aboratory. When current runs through it, thus heating it to its activation temperature, it changes shape to the shape that it has been 'trained' to remember. The wires used in this project simply reduce in length, (conserving their volume and thus getting thicker), by about 5-8 % [6].

Shape Memory Alloys, when cooled from the stronger, high temperature form (Austenite), undergo a phase transformation in their crystal structure to the weaker, low temperature form (Martensite). This phase transformation allows these alloys to be super elastic and have shape memory [6].

The phase transformation occurs over a narrow range of temperatures, although the beginning and end of the transformation actually spread over a much larger range of temperatures. Hysteresis occurs, as the temperature curves do not overlap during heating and cooling [6]. With thick wires, this could bring about problems for the SAS as the NiTi wires would take some time before returning to their original lengths, however, due to the very small diameter of the NiTi wires used (~0.15mm), the hysteresis was almost negligible as the cooling to below the Martensite temperature, (Mf), was almost instantaneous [4].

3 Building the Self Adaptive Snake (SAS)

The main bulk of the robot snake is made of foam. This provides a restoring force great enough to restore the wires to their original lengths after each activation. The

SAS used in the experiments uses twelve NiTi wires (diameter=0.176mm, activation (Austenite) temperature=70^{o}C, recommended current 200mA). The body of the robot snake is split into four segments (this is an extension from the first SAS prototype described in [5] which used only a single segment). These segments are readily detachable from each other and the whole structure can be expanded or segments replaced. Each segment has three NiTi wires running down its length and a central copper wire that runs through the foam and supplies the power, much like a spinal chord carries nerve impulses to muscles through the body see Fig. 1. These four segments are connected together and the ends of the 'spinal chord' are connected together to create a continuous connection along the length of the robot snake see Fig. 2. The total weight of the robot snake is approximately 150g.

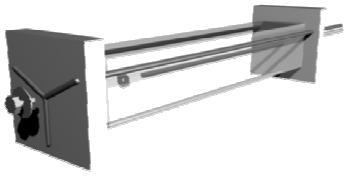

Fig.1. A single segment of the robot snake showing the three NiTi wires and the central copper wire.

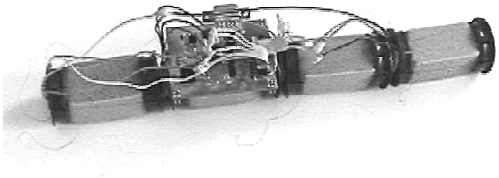

Fig. 2. A photograph of the SAS showing all four segments of the robot snake connected together.

Finite State Machine (FSM)

The NiTi wire activations are determined by a FSM, this is evolved by a special genetic algorithm designed especially for this task. The initial state of the FSM is constructed randomly and an array of FSMs was constructed representing a population of solutions.

Each member of the population is made up of a string of 0s and 1s. Each string consists of two segments, 'sequence' and 'next time', see Fig. 3. The 'sequence' is the part that is sent to the SAS, and determines which wires are to be activated at that particular time. The 'next time' is the part that tells the program to which time slot in the current row it should then jump. For the SAS, the 'sequence' length is twelve bits, and the 'next time' length is six bits, therefore the total length of each string is 2^{6} x (12 + 6) = 1152 bits.

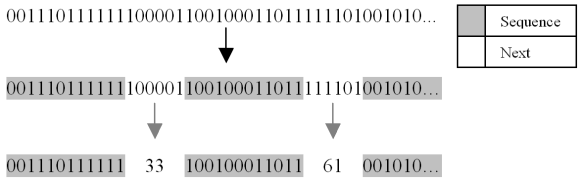

Fig. 3. Binary string is split into 'sequence' and 'next time' segments.

The program is allowed to send 30 sequences to the SAS before stopping, lasting a total of approximately 40 seconds. Finite state machines that act as repeating patterns

are easily created if the jump points point to each other in a loop of some sort. Indeed, very rarely is there a string that did not loop at all.

These patterns are then sent to the SAS for evaluation. The fitness given to each individual in the population is directly proportional to how far the SAS travels after the sequences have been sent.

The computer interfacing hardware, though quite complex, has two simple tasks to perform. The first is to supply enough power to each NiTi wire (~200mA). This is achieved with the use of some Darlington amplifiers. The second task to perform is the ability to activate muscles in parallel. For this, the microcontroller used is the Motorola MC68HC908JK1 [9]. It has 20 pins and with the structure of the circuit board is capable of activating up to 12 pins in parallel (PTBx and PTDx excluding PTB0 and PTB3). [4,5]

Genetic Algorithm

After the program has gone through the whole population, the genetic algorithm is used to evolve the next generation of solutions. This method is described below using a sample string containing 6 'sequence' bits. Two strings are chosen from the population using roulette wheel selection. See Fig. 4

10110 **3**	111000 **4**	101111 **5**	110000 **1**	101011 **7**	001010 **3**	111001 **6**
0101102	100100 **4**	100001 **5**	110111 **3**	001000 **6**	110010 **2**	100101 **3**

Fig. 4. Example of two strings chosen from the population of current solutions.

The repeating pattern that each individual defines is illustrated below see Fig. 5 & 6. These patterns can be extracted and better observed as simple loops. These simple loops are now taken as the chromosomes of the solutions and so only these parts of the complete string are crossed over. As can be observed in Figs 5 & 6, these chromosomes can be of varying lengths. In order to keep the length of the complete string constant, the shorter of the two chromosomes is taken, and a crossover section within its length is randomly cut. A section of the same size as that of the first chromosome is also cut at a random point along the second chromosome. These two sections are then swapped. Note that in order for the loops to remain functional, only the 'sequence' sections of the chromosomes are in fact crossed over leaving the 'next time' sections in unison, see Fig. 7.

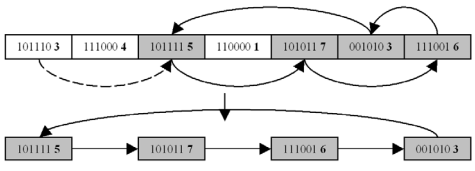

Fig. 5 The first string is converted into the loop that it represents.

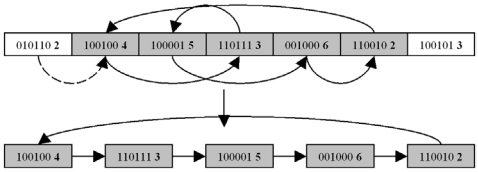

Fig. 6 The second string is converted into the loop that it represents.

Once this is done, mutation is carried out, where each bit of the string has an equal chance of being mutated, Fig. 8. These chromosomes are then placed back into their original slots in the complete string.

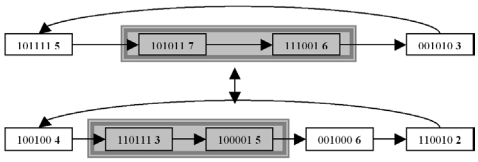

Fig. 7. Sections of the same size are swapped leaving the 'next time' sections intact.

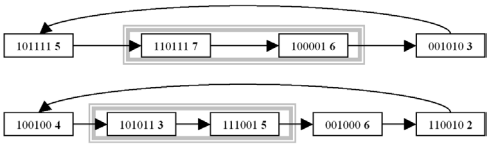

Fig. 8. Mutation is carried out only in the sections crossed over.

Only these chromosome fragments within the entire string are mutated, this ensures that these loops are accessed. If the rest of the sequence was mutated, then there is a good chance that a completely different loop would be accessed, thus the offspring would in no way resemble its parent and its fitness would be completely independent to that of the parent.

Though the sizes of both loops are conserved, mutation has a chance of completely redefining every loop, and so completely new solutions are still given a chance to be discovered which could have very different loop sizes. The genetic algorithm is elitist and so all but two of the solutions in the new population of solutions are constructed in the way described above. The last two solutions in the new population are a direct copy of the best two solutions of the previous population. This ensures that the maximum fitness in each generation never decreases.

4 Experimentation

4.1 Objectives

The objectives of this experiment are to investigate whether the genetic algorithm can find new methods of locomotion to recover from damage caused to the robot snake. In order to test this, a series of experiments are performed to investigate the evolution of movement for the SAS. The SAS's motion is evolved until the maximum fitness in the population of solutions remained unchanged for seven generations (chosen because of feasibility and time constraints). This was taken as a (somewhat local) maximum for the current configuration of the snake and its environment. Two wires that are widely used by the most fit individuals of the previous generations are then damaged. This was done by blocking access to them via software. The genetic algorithm is then allowed to continue evolving motion until another maximum is reached. Again two widely used NiTi wires were damaged and evolution was again allowed to continue.

4.2 Set up

The microcontroller board is attached to the top of the robot snake and all the NiTi wires are connected to the IO ports of the board, see Fig. 3. The only wires leaving the snake are the power supply and RS232 (both designed to minimise weight and resistance to movement). Another important factor that was realised in previous

experiments [4,5] and taken into consideration is that the resistance of the wires remains more or less constant no matter how far the SAS moves.

The distance travelled by the robot snake can be measured to the nearest millimetre. Experimentation was done to observe the true accuracy by which the distance travelled should be stated. The results show that the distance travelled was always within the nearest millimetre, and so the distances travelled by the SAS can be stated to the nearest millimetre without being over accurate [4].

As the program starts, an initial population of randomly generated finite state machines is created. Each individual finite state machine is then sent to the SAS for evaluation. The fitness of any member is determined solely on how far the snake travels (in mm) in the forward direction. The total time taken to complete each generation is roughly twenty minutes (size of population (20) × time taken for each evaluation and reinitialisation (~1 minute)). The total time taken for the whole experiment to complete was 33 hours.

5 Results

Figure 9 shows the distance travelled at each generation. The greatest distance travelled by the SAS before any wires were damaged was 92mm. Immediately after two commonly used wires were damaged, the maximum distance travelled by the SAS in the next generation was 27mm. This is equivalent to a 71% drop in performance.

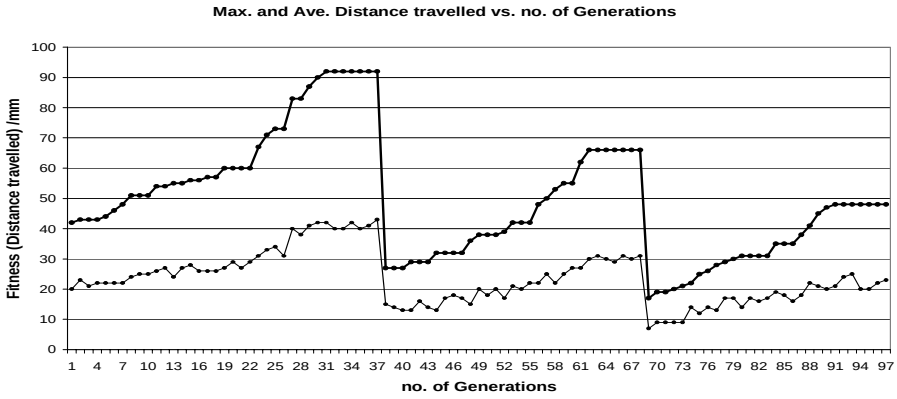

Fig. 9. Maximum fitness (bold) and average fitness plotted at each generation.

The SAS was then allowed to evolve for another 20 generations. This improved the distance travelled to a maximum of 66mm. This is an improvement in performance of 244%, and motion has been recovered to within 72% of the original undamaged SAS. The SAS was then once again damaged, removing another two commonly used wires from action, resulting in a 74% drop in performance. Finally the SAS was then allowed to evolve for a final 18 generations. This improved the distance travelled to a maximum of 48mm. This is an improvement in performance of 282%, and motion has been recovered to within 52% of the original undamaged SAS.

6 Analysis

During the evolution of the SAS, numerous interesting methods of locomotion were carried out. This section seeks to analyse the control strategies of the solutions at the three maximums of the graph above. These methods of locomotion correspond to the best sequence found before any wires were damaged, the best sequence found after the first pair of wires were damaged, and the best sequence found after the second pair of wires were damaged.

Though the sequence lengths could vary from a length of one to a maximum of sixty-four, the sequence lengths that travelled the furthest seemed to average a length of ten. The evolved sequences at these points are very complex. By way of illustration, the two most commonly used wires out of twelve in those sequences are shown below, see Fig. 10, 11 & 12. In each of the sequences, the two wires highlighted were simultaneously activated and deactivated in nearly every step, while a complicated sequence of activations was sent to the rest of the wires. The two wires that are highlighted in Fig. 10 are also the wires that were subsequently damaged, resulting in a new sequence (and thus an entirely new movement plan) that relied on two different wires being activated and deactivated as illustrated in Fig.11. Once again, the two wires highlighted were the most commonly activated in this second solution and so were subsequently damaged resulting in the final solution. This relied on two different wires being activated and deactivated more often than any other wire in order to move effectively (and thus a third entirely new movement plan), as illustrated in Fig. 12.

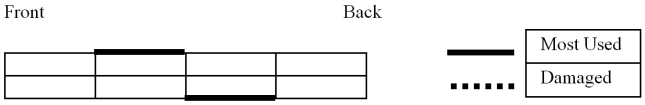

Fig. 10. Plan view of muscle wires in the four segments of the SAS (horizontal lines). The two most commonly used wires when there is no damage are shown in bold.

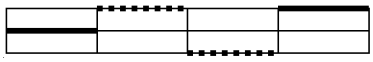

Fig. 11. The two most commonly used wires after two wires are damaged (bold), damaged wires are shown as dotted lines.

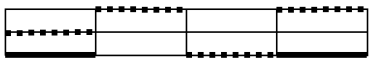

Fig. 12. The two most commonly used wires after four wires are damaged.

In previous experiments performed with fewer wires, the motion exhibited by the SAS could be compared to various types of snake undulation [4,5]. This has not been the case with twelve wires. There are two possible reasons for this. Firstly that the genetic algorithm had not had enough time to evolve elegant snakelike motion. The second reason could be that the genetic algorithm was exploiting asymmetries in the construction of the robot snake, the difference in the tautness of the NiTi wires, and

the individual properties of these wires. This being the case, these more asymmetrical sequences would prove to be more effective for creating motion than if the SAS had explicitly tried to mimic snakelike motion.

7 Conclusion

This work has shown that it is possible to use a combination of a self-adaptive snake robot (using shape memory alloys to provide flexibility) and evolutionary control to produce a robot that is capable of recovering from damage by learning new movement strategies. The experiment demonstrated that a high percentage of locomotion can be recovered more than once, even when muscle wires crucial to a locomotion strategy are disabled. Further work will investigate exactly how the movement of the robot snake is achieved, through analysis of the evolved control strategies. A testbed will also be created, allowing robots and their locomotion to be assessed automatically, enabling faster and longer evolution runs and increased robot capabilities.

Acknowledgements

Thanks to Sara and Saba Haroun Mahdavi for their invaluable assistance in the experiments. This work is being performed in collaboration with BAE Systems.

References

1. Bentley, P. J. (Ed.) (1999) *EvolutionaryDesign by Computers*. Morgan Kaufmann Pub.
2. Bererton, C and Khosla, P. A Team of Robots with Reconfiguration and Repair Capabilities. In proceedings International Conference on Robotics and Automation, 2000.
3. Dittrich, P., Skusa, A., Kantschik, W., and Banzhaf, W. (1999) Dynamical Properties of the Fitness Landscape of a GP Controlled Random Morphology Robot. Proc. of GECCO '99, p. 1002-1008, Morgan Kaufmann, San Francisco, CA, 1999
4. Haroun Mahdavi, S. (2002) Evolving Motion Master's Dissertation, MSc IS, University College London, Dept. Computer Science.
5. Haroun Mahdavi, S and Bentley, P. J. (2002) Evolving Motion of robot with muscles. To appear in Proc. of *EvoROB2003, the 2nd European Workshop on Evolutionary Robotics.*.
6. Hodgson,D.E., Wu,M. H. and Biermann, R. J. (1999) Shape Memory Alloys. Shape Memory Applications, Inc. http://www.sma-inc.com/SMAPaper.html
7. Kamimura, A et al. (2001) Self-Reconfigurable Modular Robot - Experiments on Reconfiguration and Locomotion. Proc. of IEEE Int. Conf. on Intelligent Robots and Systems, 590-597, Hawaii, USA.
8. Kárník, L. (1999) The possible use of artificial muscles in biorobotic mechanism. In ROBTEP'99, Kosice, SF TU Kosice, 1999, pp. 109-114. ISBN 80-7099-453-3.
9. MC68HC908JK1 Microcontroller User's Manual (2002). Motorola Literature Distribution: P.O. Box 5405, Denver, Colorado 80217
10. Michael, J. *Robodyne Cybernetics Ltd*. www.fractal-robots.com
11. Mills, Jonathan W. (1999) Stiquito for beginners. ISBN 0-8186-7514-4, 1999.
12. Støy, K (2002) Using Role Based Control to Produce Locomotion in Chain-Type Self-Reconfigurable Robots. IEEE Transactions on Mechatronics, 7(4), pages 410-417, 2002.