Article

Performance Comparisons of Bio-micro Genetic Algorithms on Robot Locomotion

Francisco A. Chávez-Estrada[1]*, Jacobo Sandoval-Gutierrez [2]*, Juan C. Herrera-Lozada [3], M. Olguín-Carbajal [3], Daniel L. Martínez-Vázquez[2], Miguel Hernández-Bolaños [3] and Israel Rivera-Zárate [3]

1 Ingeniería, Instituto Tecnológico de Tlalnepantla del Tecnológico Nacional de México (ITTLA del TecNM), Estado de México 054070, México; frchaveze1400@alumno.ipn.mx or fachavez@ittla.edu.mx.
2 Procesos Productivos, Universidad Autónoma Metropolitana, Unidad Lerma (UAML), Lerma de Villada, Estado de México CP 52006, México.; j.sandoval@correo.ler.uam.mx, d.martinez@correo.ler.uam.mx
3 Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC-IPN), Ciudad de México 07700, México.; jlozada@ipn.mx, molguinc@ipn.mx, irivera@ipn.mx, mbolanos@ipn.mx
* Correspondence: j.sandoval@correo.ler.uam.mx or fachavez@ittla.edu.mx; Tel.: +527282827002 ext. 3011

Featured Application: The specific application of the work can be found at https://youtu.be/ctJLINSq27Q

Abstract: This paper presents a novel micro-segmented genetic algorithm (μsGA) to identify the best solution for the locomotion of a quadruped robot designed on a rectangular ABS plastic platform. We compare our algorithm with three similar algorithms found in the specialized literature: a standard genetic algorithm (GA), a micro-genetic algorithm (μGA), and a micro artificial immune system (μAIS). The quadruped robot prototype guarantees the same conditions for each test. The platform was developed using 3D printing for the structure and can accommodate the mechanisms, sensors, servomechanisms as actuators. It also has an internal battery and a multicore embedded system (mES) to process and control the robot locomotion. This research proposes a μsGA that segments the individual into specific bytes. μGA techniques are applied to each segment to reduce the processing time; the same benefits as the GA are obtained, while the use of a computer and the high computational resources characteristic of the GA are avoided. This is the reason why some research in robot locomotion is limited to simulation. The results show that the performance of μsGA is better than the three other algorithms (GA, μGA and AIS). The processing time was reduced using a mES architecture that enables parallel processing, meaning that the requirements for resources and memory were reduced. This research solves the problem of continuous locomotion of a quadruped robot, and gives a feasible solution with real performance parameters using a μsGA bio-micro algorithm and a mES architecture.

Keywords: Micro segmented Genetic Algorithm; Multicore Embedded System; Parallel Processing

1. Introduction

The solutions found by genetic algorithm (GA) [1] methods can improve traditional techniques; they are well known for being robust when used for numerical optimization and are applied where there are no standard techniques. In view of the strengths of these algorithms, some researchers have applied GA approaches to solve the problem of locomotion of legged robots. Arranz de la Peña et al. [2] used a GA for the locomotion of a quadruped robot. This approach automatically generated solutions and selected the best option for robot locomotion with 12 degrees of freedom (DOF). Research by Mutka et al. [3] improved stability in robot locomotion with a dynamic system composed of four spring-mass subsystems and a tail-like inertial appendage. A tail with two DOFs was used as a counterweight, and its mass center was able to shift to balance the body of the robot, increasing the number of DOFs but contributing to maintaining autonomy and improving the performance of robot locomotion. One of the main contributions of this study was the use of an ABS plastic platform, as this is a flexible material that contributes to stability during robot locomotion. In the first prototype, the platform was designed using rigid PLA plastic, and the robot locomotion was unstable.

The GAs are the most representative instances of the evolutionary algorithms (EAs). These are population algorithms that are used to generate a set of up to thousands of individuals (i.e. solutions to a problem). The best individuals within this population are then found, with frequent changes being made to the characteristics of the individuals. Changes are controlled by systematically applying the genetic operators of selection, crossover, mutation and replacement, which contribute to maintaining the diversity of the population, as indicated in [4]. Individuals are initially randomly generated, and genetic operators are applied to each, so the improvements generated by these operators have a direct impact on the performance of the algorithm.

Within the population, each individual is differentiated based on its fitness value, which is obtained empirically according to the problem to be solved. The fitness function is the objective function of the optimization problem, as described by Asteroth and Hagg [5] and Burke et al. [6]. Fig. 1 shows the operation of a standard GA for robot locomotion. One of the main features of the GA is its dependence on a randomly generated population of individuals. There is a certain computational cost linked to the size of the population, which arises from memory use, processing time and energy consumption. A hardware implementation of the genetic operators for these algorithms [7] can improve their performance and facilitate the production of solutions in this type of process.

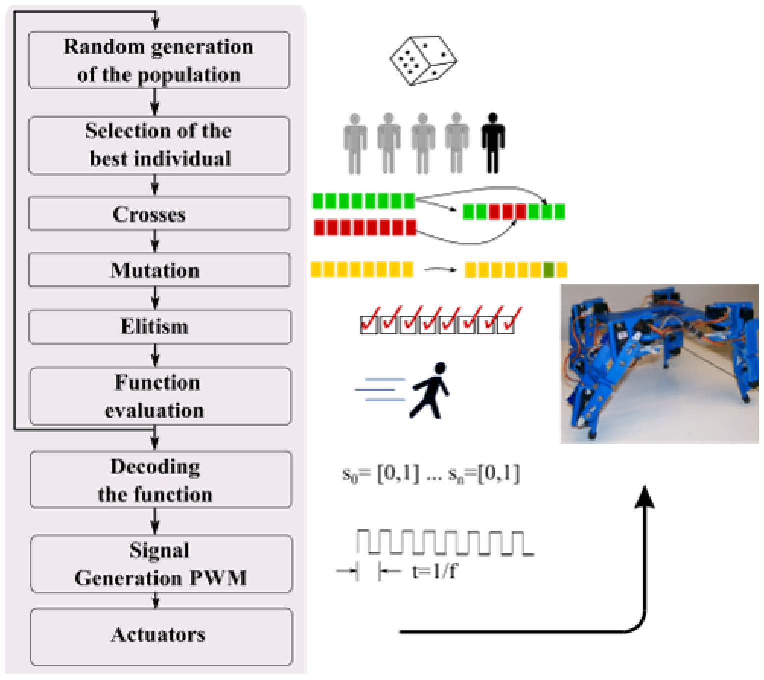

Figure 1. Flowchart for a standard GA for robot locomotion

2. Background

The GA is applied to find solutions to problems of optimization of systems with various DOF and in which traditional methods are not easily adjusted. In the case of robot locomotion, Gumiel applied a GA to a robot called Aibo [8]. In [9], Suzuki performed an optimization of the impulse trajectory of the legs by applying a GA. Parker and Torino [10] applied a cyclic GA to generate a running sequence.

In the research study in [11], optimization, speed and learning on the fly were evaluated, and four GAs were analyzed: a GA, a covariance matrix adaptation evolution strategy (CMA-ES), a swarm optimization of particles (PSO) algorithm and a differential evolution (DE) approach. The results were compared and it was found that none of these algorithms was significantly better than the others. In [12], a central pattern generator (CPG) was combined with a bio-inspired multi- objective evolve algorithm to optimize a gear by reducing vibration and increasing speed, stability and behavior using a simulation platform and Ren et. al [13] a gait generator based on CPGs modulated was studied. Other researchers have focused their efforts [14] on energy optimization by improving the trajectories of a robot when walking using a GA. In [15], a GA was used to produce a CPG and solve the problem of locomotion in robots with legs. Another research study [16] used a mimetic fingerprint matching algorithm and a real chromosome chain, running on a Pentium 2.1 GHz computer, and found that solutions had higher efficiency but took more time. For the implementation of algorithms in hardware to provide locomotion to robots with legs, the use of computers as a primary control loop is very common. In this investigation, integrated multi-core systems (mES) were used, which have a high parallel processing capacity, similar to a computer, allowing to reduce the execution time. The algorithms, which are implemented using an eight-core embedded system (ES), also allow for parallel processing, with the objective of reducing the processing time to less than one second and solving the problem of robot locomotion. This research is a continuation of micro genetic algorithm cited by Chávez et al., based on a novel estimator, for locomotion of a quadruped robot [17]. In [18], a μGA was developed with a sharing function based on the similarity and H-measure of DNA sequences. This algorithm obtained better DNA sequences and improved the computational efficiency. IN relation to the intrinsic execution of bio-inspired algorithms in embedded systems, a micro-artificial immune system (μAIS) was presented in [19] to solve numerical optimization problems; in this micro algorithm, a cloning scheme was proposed with a population of only five individuals (antibodies in the AIS context) with the purpose of hardware implementation.

In this study, we propose an alternative method of using μGA with small populations, and use segmented elements or individuals to reduce the size of the binary chain. This allows for a reduction in the processing time for the search for solutions to less than one second, thus guaranteeing performance similar to that of the GA. We use a binary chromosome chain and an embedded 8 MHz system to solve the problem of robot locomotion using parallel programming. This allows the system to find the solution in less than one second. Several of the works cited above were limited to simulations due to the high computational cost required to find solutions. The new algorithm, called the micro-segmented genetic algorithm (μsGA), segments the individual chain into bytes and applies evolutionary techniques to each. This contributes to reducing the processing time to less than 0.5 s. Our hypothesis is that reducing the number of individuals in a population will decrease the number of evaluations of the objective function, thus increasing the speed of convergence.

The main contributions of this paper are as follows: (i) the proposed μsGA outperforms other GA, μGA and μAIS algorithms; (ii) segmented elements or individuals are used to reduce the size of the binary chain; (iii) the programming of the algorithm was carried out and tested on an embedded system; and (iv) improved performance is seen due the the use of parallel processing.

The remainder of the paper is organized as follows. Section 3 introduces the design and organization of the system. Section 4 presents the novel micro segmented genetic algorithm. Section 5 describes the setup of the experiments. Section 6 presents the experimental results and a discussion. Finally, conclusions are given in Section 7.

3. Design of the robotic platform

The locomotion model of a mammal was used for the design of the quadruped robot prototype (see Figure 2). The structure of the robot prototype designed in this research includes the frame or body of the robot and four legs, each of which has three links joined by servomotors (see Figure 3).

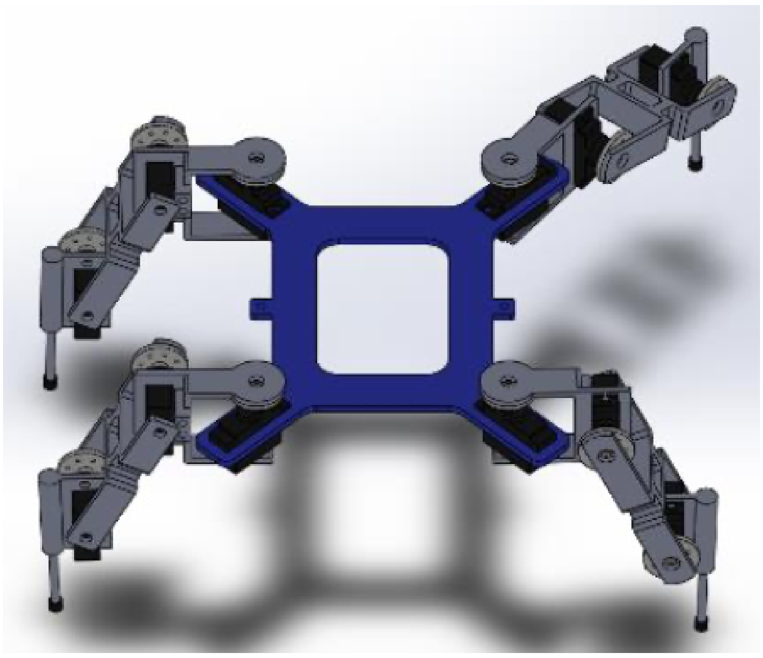

Figure 2. Quadruped robot platform with 12 degrees of freedom

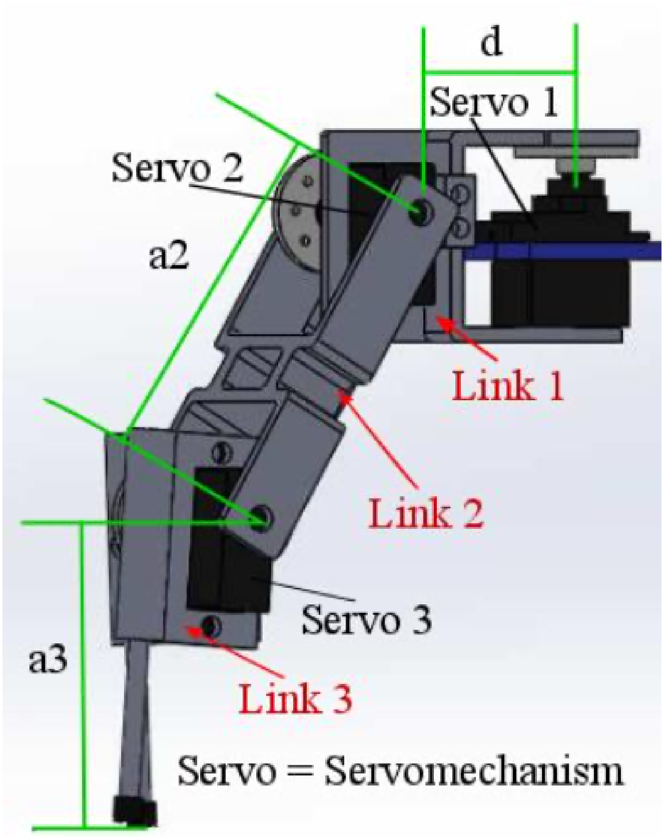

Figure 3. Mechanical design of a robot leg

In the first prototype, the thickness of the pieces of PLA was 0.003 m. With this design, the locomotion of the robot was unstable due to the thinness and rigidity of the pieces causing deformations and stresses in the links. To solve this problem, the finite element method (FEM) was applied and a static analysis of the parts was carried out using the SolidWorks application to detect the stresses in the links (Figure 4).

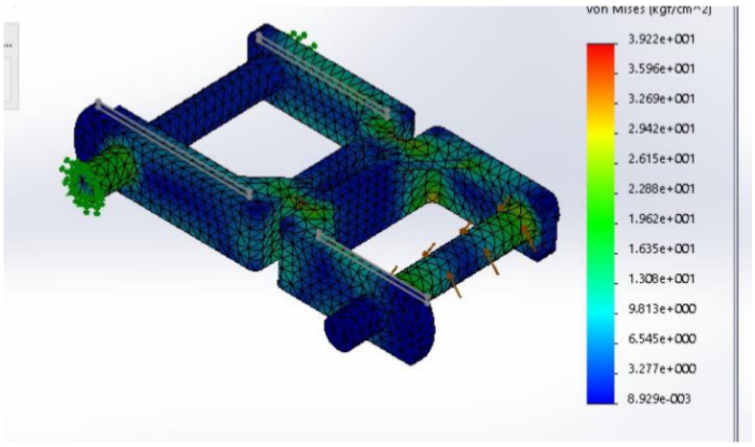

Figure 4. Mechanical design of a link using the finite element method

This analysis indicated the deformations and stress zones in the links and allowed us to determine the new physical characteristics for the second version of the prototype. In this version, the parts were manufactured in ABS plastic with thickness 0.004 m. Pieces that were subject to a greater torque force (TF) or force (F) per volume unit had a thickness of 0.006. The dimensions of the reinforced pieces are shown in Table 1. A rectangular platform was used to allow the installation of four legs, IO electronics and servomotors (devices required for control) and batteries. The legs are arranged symmetrically in each corner to ensure that the center of gravity was within the support polygon, and to guarantee that the torque on each leg was distributed evenly. The four legs need to support 0.75 kg, corresponding to the weight of the robot, and each of the servomechanisms has a 2.0 kg/cm torque, a consumption of 0.16 A, and a 5 V source.

Table 1. Dimensions of the robot parts

Part	Length (m)	Width (m)	Thickness (m)	Centers (m)
Rectangular structure of the robot	0.190	0.160	**0.004**	0.110
Link1 (d)	0.060	0.045	**0.003**	0.035
Link1 (a2)	0.095	0.004	**0.014**	0.070
Link1 (a3)	0.060	0.030	**0.004**	0.045
Servomotor	0.029	0.014	**0.038**	NA

4. Segmented micro genetic algorithm

Fitness is the driving force for both Darwinian natural selection and genetic algorithms [20]. This parameter can be measured in several explicit or implicit ways. In this research, we used an explicit measurement. The individuals of the population are 32-bit binary chains, and their fitness is their equivalent decimal value. Individuals with higher decimal values survive and reproduce to generate a population that evolves towards the solution. The chromosomes or individuals generated in the algorithm are tested to determine their fitness

This measurement controls the ability of the genetic operators to modify the characteristics of the random population. It seeks to maximize the movement of a robot with legs, and the goal is that each link in a robot leg reaches its maximum value. In this case, the best individuals are those with a fitness value high enough to achieve the target of walking.

Thus, the individual solution or fitness is the decimal value represented by $r(i, g)$ for an individual i in a generation g:

$$r(i,g) = \sum_{j=1}^{N} |s(i,j) - c(i,j)| \tag{1}$$

where $s(i,j)$ is the desired value, $c(i,j)$ is the value obtained by the algorithm, and N is the number of events. The design of the μGA begins by identifying and describing the parameters that determine the robot locomotion by trotting.

A GA generates a random population of individuals represented by the values $c(i,j)$, made up of 32-bit binary strings or four bytes. The first three bytes represent the displacement angles in each link; the desired value for each of these is $s(i,j) = 255$, representing the maximum allowed displacement for each link. The fourth byte represents the impulse time in milliseconds (t_I) or foot support time on the ground. This is used for locomotion, and is part of the period of advance (T) i.e. the time it takes for a leg to complete an advance cycle (support-impulse-advance). Each individual or solution contains the values required by the servomechanisms to produce the robot walking pattern (RWP), see Fig. 5.

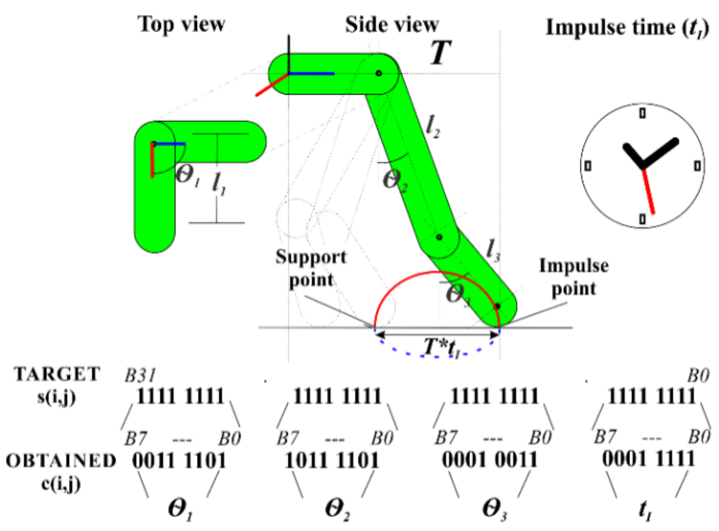

Figure 5. Relationship between the links in a leg and the binary structure of the chromosome or individual

The angle displacements of a leg depend on the movements of the links l_1, l_2 and l_3 and are determined by Θ_1, Θ_2 and Θ_3, respectively. Changes in speed are determined by the time of impulse or time interval in which a robot leg is touching the surface (t_I). These processes determine the first stage of robot locomotion, and there are four parameters for each leg that determine the robot's locomotion (see Figure 5).

The maximum displacement is obtained when the links have an angular value: $\Theta_1 = 90°$ (fixed value), $\Theta_2 = 20°$ and $\Theta_3 = 40°$. In each case, the maximum theoretical value of the angle that each link of the robot can reach is 255°, corresponding to 1111 1111 in binary notation, due to the constraints on the system. In Figure 5, the individual segmentation is represented by four bytes, and the μGA is applied to each segment with the aim of reducing the processing time.

In the design of the algorithm, we use the concepts described by Goldberg [4] regarding the μGA, which is an algorithm based on small populations of individuals. It was determined that a population of 12 individuals ensures fast convergence to feasible solutions. A flowchart for the algorithm using these concepts is given in Figure 6).

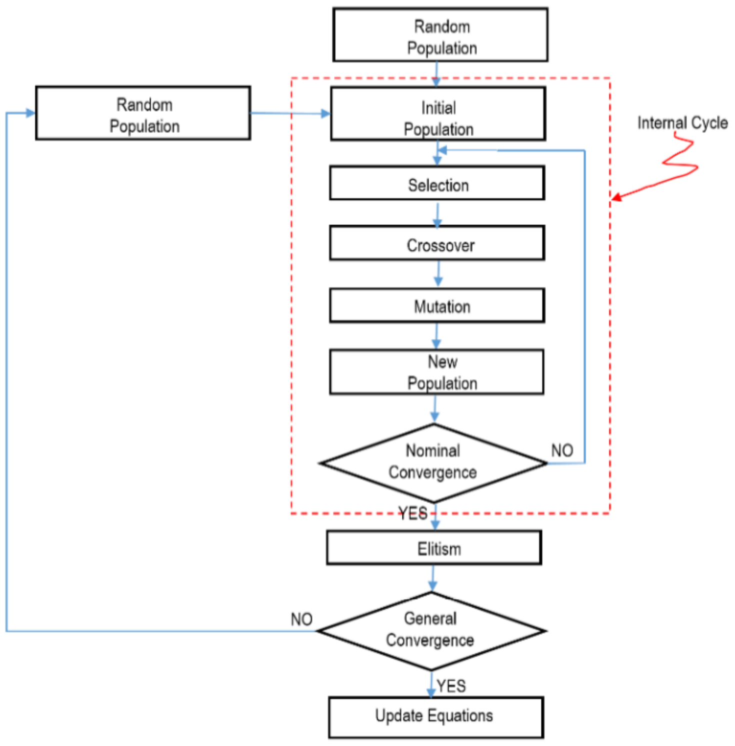

Figure 6. μGA flowchart

In Figure 6, the algorithm starts with a random population of 12 individuals, and the best is copied to the initial population. Several experiments were applied to the genetic operators, such as two mutations per segment, two crossovers per segment, increasing or decreasing the population by generation and others that are not reported in this research. Finally, the parameters were defined as one random crossover and one random mutation of a bit for each segment selected. These conditions allow for faster nominal convergence to the solution (see Figure 7).

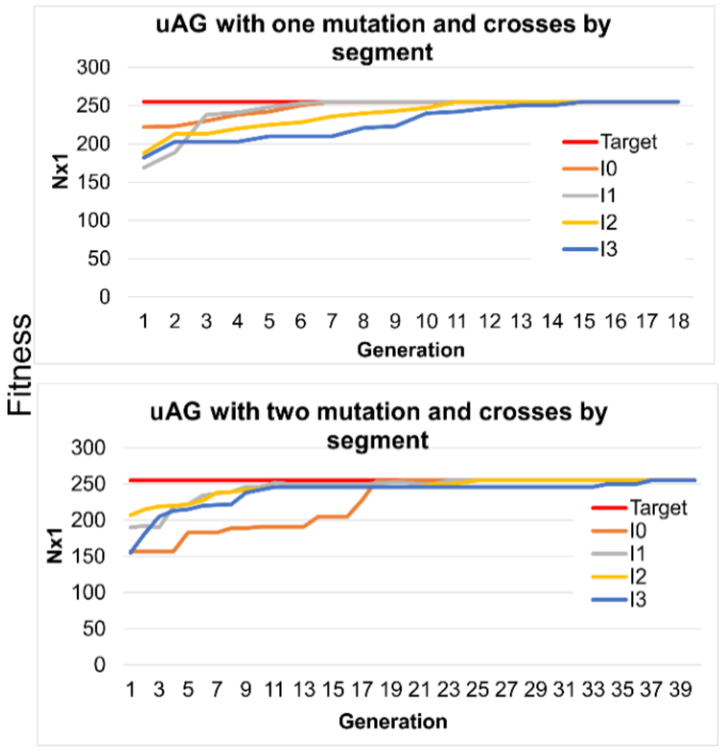

Figure 7. μGA with two mutations and crossovers vs. one mutation and one crossover

The evolution process in the μAG is executed in the nominal convergence. The 12 individuals of the active population remain and are evolved n times to the solution. When nominal convergence is reached, the best individual is copied to the next initial population, and for general convergence, the 11 remaining individuals will be generated at random.

In this way, the internal and external cycles of μGA achieve better results. The internal cycle achieves nominal convergence reaching the solution for the first best string, and the external cycle will terminate when the general convergence reaches the stop criterion, getting the first three best individuals (0, 1 and 2). The characteristics of the μGA such as a small population allow for fast convergence to the solution. The best individual is formed from the bytes with the best fitness (bytes 0, 1 and 2), which will generate the maximum displacement in the robot links and will try to maintain a straight line.

5. Robot locomotion

Locomotion is a continuous re-configuration or action that a system performs to improve the task being executed. The system must be reconfigured until the maximum displacement of the links is reached. The system generates a pattern of movements or CPG, and these are designed using coupled oscillators that generate a sinusoidal function [21] (see Equation (2) and Figure 8).

$$q(j) = temp.individual[i] * sin(\frac{\pi * temp.individual[i]}{2 * 255}) \tag{2}$$

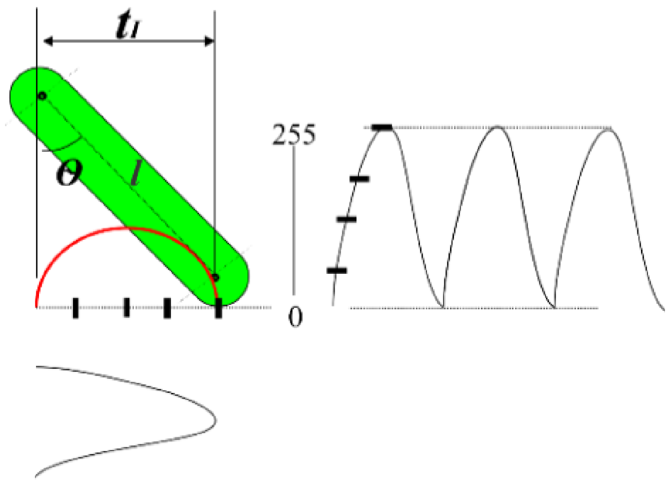

Figure 8. μGA with two mutations and crossovers vs. one mutation and one crossover

$temp.individual[i]$ is the binary value of the best individual resulting of the evolution of the μsGA. The indices i of the bytes that form the first four best individuals are 0, 1, 2 and 3. The best $temp.individual[i]$ is 255 (1111 1111). This is the maximum theoretical amplitude that the CPG can reach for each link. This parameter is also related to the angle displacement. $q(j)$ is the numerical value of the angle that will displace the link resulting from the evolution of the best individuals found. The maximum value that can be obtained is 255° (1111 1111). $j = 1, 2$ and 3. Each value $q(j)$ is multiplied by a constant K_j to obtain the displacement angle (Adj). The constant K_j is obtained by considering the maximum angular value (max_ang_val_j) allowed for each joint. The proportionality relationship is:

$$255 - max\_ang\_val\_j$$

$$q(j) - Adj$$

The displacement angle (Adj) required for the articulation is obtained as shown in Equation (3):

$$Adj = q(j)(max\_ang\_val\_j/255) = q(j)K_j \tag{3}$$

where $K_j = max\_ang\_val\_j/255$ for $j = 1, 2$ and 3.

The first stage of the robot locomotion trotting was completed. The sequences of displacements for the articulations in one leg were applied to the three remaining legs of the four-legged robot, with a phase shift of 90° with respect to each leg based on their distribution in a clockwise direction. The correlation temporal reference generated between the legs when walking gives rise to the phase relationships between each of the legs during locomotion. The period (T) and the support factor (β) are related, and it is shown that when the robot is walking, two legs are on the ground, and two are in the air, as shown in Figure 9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 March 2020 doi:10.20944/preprints202003.0079.v1

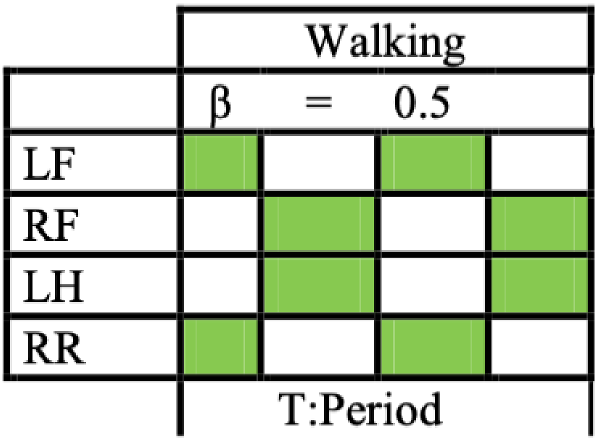

Figure 9. Phase diagram for walking

The robot executes the learning second stage by means of a gyroscope sensor, which allows it to find the optimum impulse time (t_I). This system achieves the best displacement, i.e. locomotion in a straight line, which implies that all the legs have the same displacement. The impulse time t_I is a parameter of the system. This was achieved by comparing the ideal displacement (by simulation of the direct kinematics of the system) versus the real displacement direction using the gyroscope sensor, as shown in Figure 10.

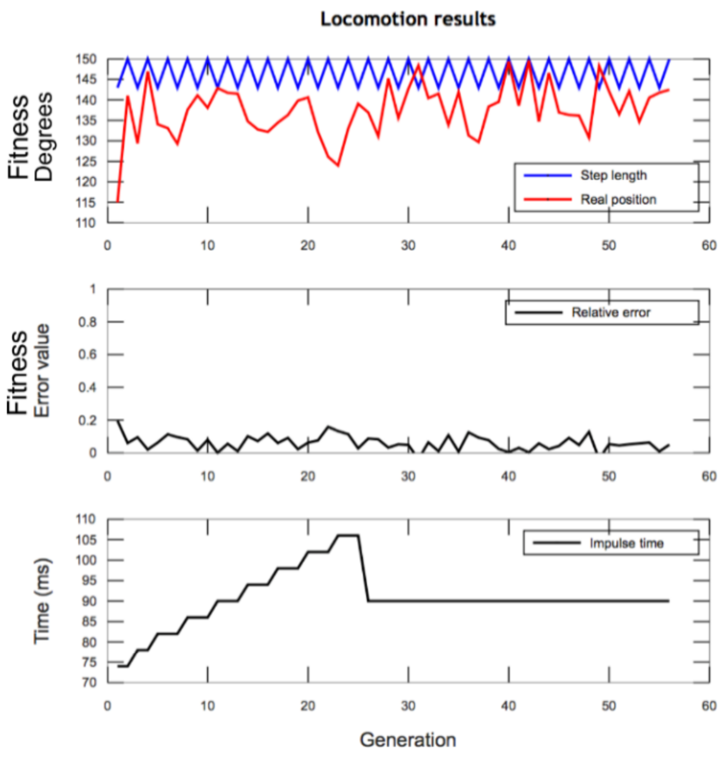

Figure 10. Ideal versus real positions of the walking robot

Figure 10 shows the differences between the real and ideal positions during locomotion of the robot. The lowest error is found between generations 10 to 12, and $t_I = 90$ ms is adopted as the parameter for locomotion. This is maintained up to generation 25. The error decreases and the system tries to follow the ideal locomotion function.

At this point, we use the GA to optimize the configuration for the links in the links robot. The value of the fourth byte represents the impulse time, and is obtained when the robot is displaced in a straight line due to the gyroscope sensor lectures. The μGA evaluates the fitness of each individual, and the best solutions are processed in two ways:

1. The best individual is copied to the next generation.
2. The best individual is decoded to obtain the displacement angles of the servomotors and to generate the locomotion of the robot.

The movements of the servomechanisms are controlled by pulse width modulation signals. These signals are generated by the μGA executed in the embedded system, which creates all the signals required for the servomechanisms of the legs, thus generating the locomotion pattern of the robot.

The second objective of this research is to create a random generation of the strings used to build the individuals and to apply the genetic operators (mutation and crossover) with the aim of reducing the processing time. In this case, we also segment the individuals into four chains of eight bits by applying mutation and indexing the four chains. Only the string of every individual will access by eight changes (Figure 11).

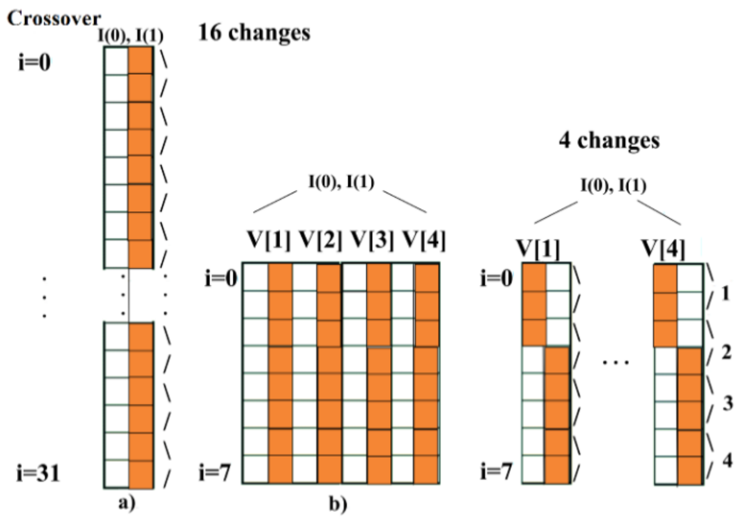

Figure 11. Mutation process

When applying the genetic crossover operator to individuals of 32 bits, this implies 16 changes for every individual, as shown in Figure 12. In this case, we also segment individuals into four 8-bit strings by applying the crossover operator and indexing the four strings; thus, each individual structure will undergo four changes.

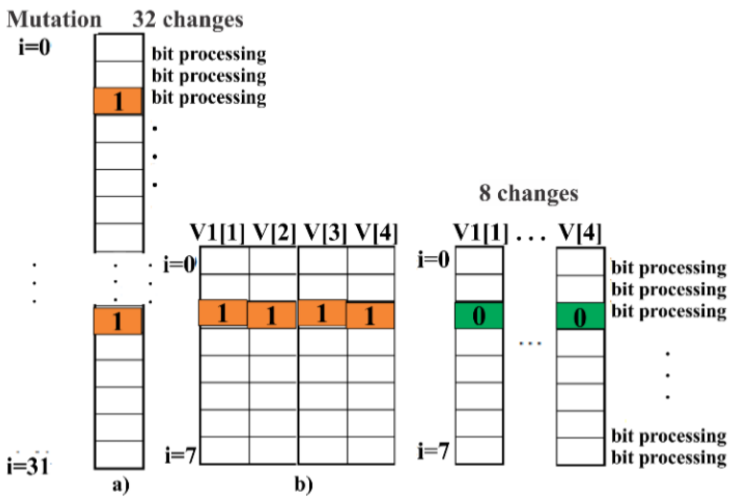

Figure 12. Applying crossover to segmented individuals

When processing segments directly, the decoding process is not required, which saves processing time. Applying the genetic operators of mutation and crossover segmentation reduces the processing time if we index the four segments for each individual. Segmenting individuals allows us to apply the evolutionary technique to each segment formed of one byte to optimize processing, as follows:

1. Generating four 8-bit segments per individual instead of 32-bit individuals allows the processor to generate the population in less time.
2. When processing segments directly, the decoding process is not required, which involves a saving in processing time.
3. Applying the genetic operators of mutation and crossover segments reduces processing time if we index the four segments of each individual.

6. Parallel processing algorithms

A sequential algorithm processes one task at a time, and one task after another. The sequential algorithm used in this study and the flow diagram are shown in Figure 13.

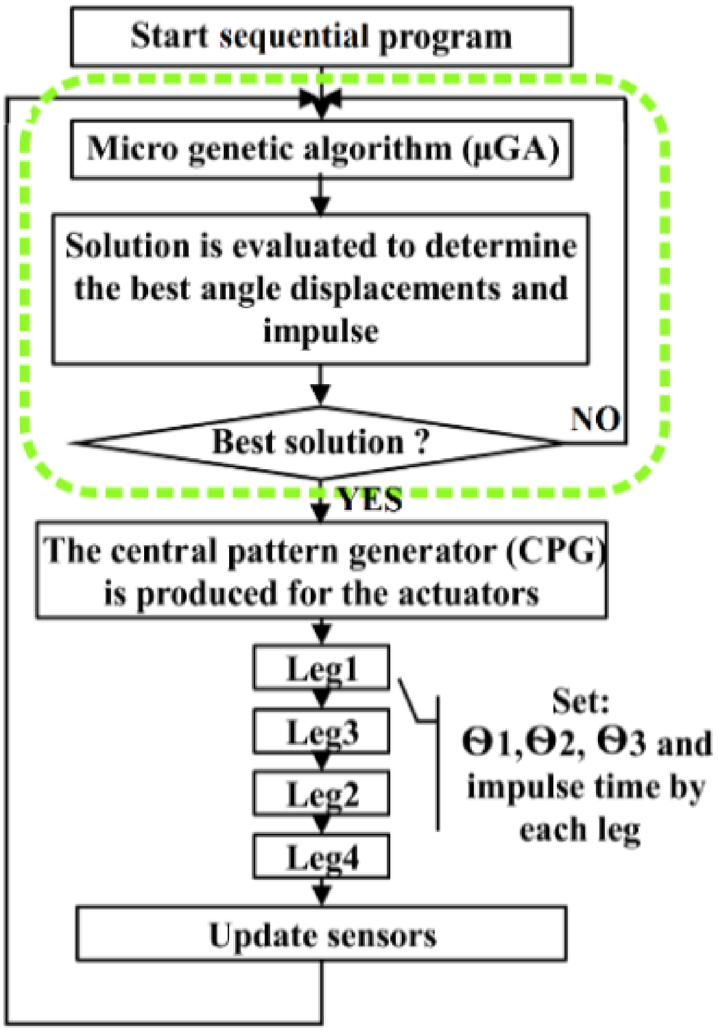
Start sequential program
Micro genetic algorithm (μGA)
Solution is evaluated to determine the best angle displacements and impulse
Best solution ?
NO
YES
The central pattern generator (CPG) is produced for the actuators
Leg1
Leg3
Leg2
Leg4
Set: Θ1,Θ2, Θ3 and impulse time by each leg
Update sensors

Figure 13. Flowchart for the sequential algorithm

The sequential algorithm is executed on a parallax activity embedded system board, and in the first task, the GA is executed to find the best solution. In the second task, the CPG signals are generated at the same time that generating the electrical signals to move the actuators Θ_1, Θ_2, Θ_3 and a pulse t_I. The values are applied sequentially to legs 1, 3, 2 and 4, in this order, which produces the locomotion of the robot in a trotting pattern.

The parallel algorithm is executed on the same embedded multi-core system, which has eight processor cores and can divide a task or execute different tasks at the same time. A flowchart of the process is given in Figure 14.

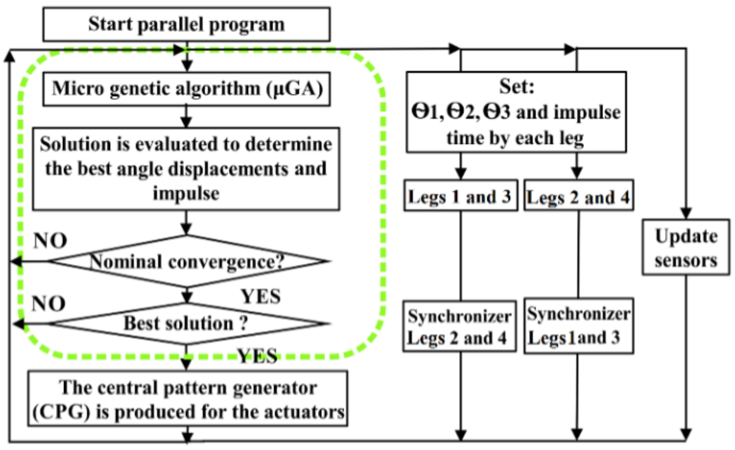
Start parallel program
Micro genetic algorithm (μGA)
Solution is evaluated to determine the best angle displacements and impulse
NO
Nominal convergence?
YES
NO
Best solution ?
YES
The central pattern generator (CPG) is produced for the actuators
Set: Θ1,Θ2,Θ3 and impulse time by each leg
Legs 1 and 3
Legs 2 and 4
Update sensors
Synchronizer Legs 2 and 4
Synchronizer Legs 1 and 3

Figure 14. Flowchart for the parallel algorithm

Four cores were enabled at the same time, and were used to execute the following tasks:

1. The first core executes the μsGA, finds the best solution, and generates the CPG for locomotion.
2. The second core takes the signals from the CPG and produces the electrical signals for the actuators of legs 1 and 3.
3. The third takes the signals from the CPG and produces the electrical signals for the actuators of legs 2 and 4.
4. The fourth core processes the sensor signals.

Each core executes a different task at the same time, in parallel. This reduces the processing time in comparison to sequential processing

7. Experiments

In the first stage, the four bio-algorithms are evaluated by applying them to the locomotion of the robot. The following process was used in the tests of the algorithms, which allowed for a comparison between them in terms of performance:

1. An initial population of 12 individuals is randomly generated.
2. Each subsequent generation has 12 individuals.
3. Each individual is formed of a string of 30 bits.
4. In the μsGA algorithm, each individual is formed of four bytes.
5. Selection is applied based on elitism (the best individual per generation).
6. A crossover is applied to each individual, selected randomly.
7. Mutation is applied to a bit in each byte, selected randomly.
8. The algorithm stops when a solution is found.
9. Each algorithm is run 10 times and the average result was stored. Each algorithm was implemented separately using the same prototype quadruped robot and executed on the mES.
10. The GA, μAIS and μsGA were executed to identify the generation (x axis) in which convergence to the solution is reached, and the corresponding fitness of the best individual (y axis). The fitness of the first three algorithms is calculated as 1073741824 (230) and the fourth is 255 (28). In the case of μAIS, the proposal of Herrera et al. was used [19], which considers five antibodies of 30 bits each and the same cloning and stranding operators.

The second stage of experimentation involved comparing the sequential and parallel processing times for the μsGA algorithm.

1. An impulse time $tI = 70$ ms was set for both processes.
2. The μsGA sequential algorithm shown in Figure 6 was implemented in the mES.

3. The time taken by the robot to move one meter in a straight line over a smooth surface was measured for the sequential algorithm, and then for the parallel algorithm.
4. The test was executed five times, and a displacement-time speed data table was constructed for each algorithm.

8. Discussion and results

Experiments on the four algorithms were carried out to identify the best performing algorithm in terms of the shortest time and the lowest number of generations in which the solution for locomotion is found. Figures 15 to 18 show these parameters.

The GA shows the worst performance of around 7.188 s, with 911 generations needed to obtain the solution for locomotion. In these conditions, learning to walk is slow; it is not continuous, and 7.188 s is a long time for the generation of continuous locomotion (see the information in Figure 15).

The μGA and μAIS schemes have better performance, and find solutions in less than one second. This allows for continuous robot locomotion, meaning that learning to walk is very fast, since each solution was found in 73 generations, in less than one second (Figure 16).

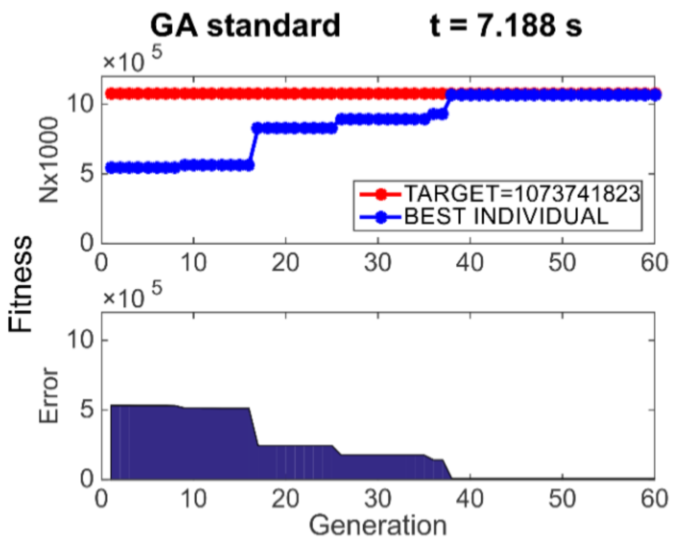

Figure 15. Evolution to a solution with GA, with individuals of 30 bits

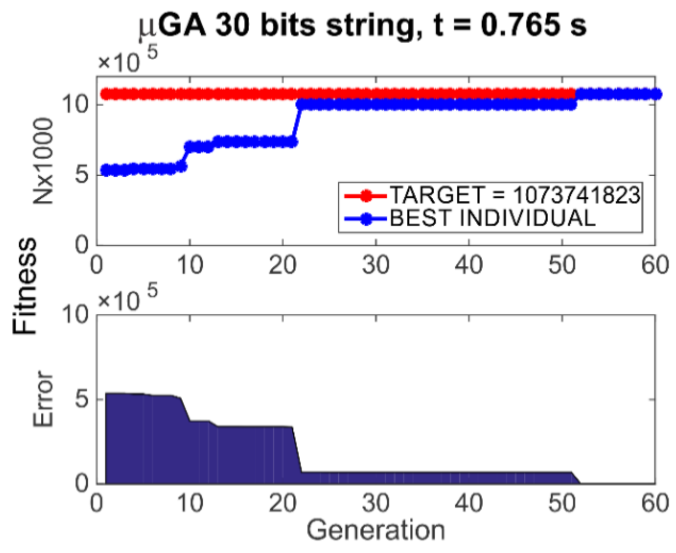

Figure 16. Evolution to a solution with μGA

The AIS scheme found the solution in 0.327s and 55 generations (Figure 17).

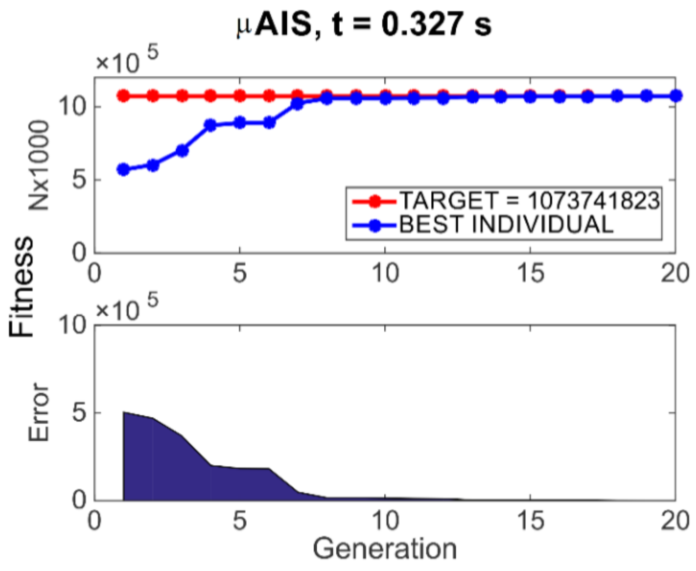

Figure 17. Evolution to a solution with μAIS

Our proposed μsGA scheme found a solution in the shortest time of 0.131 s and 21 generations. The μsGA has several advantages; for example, it has the characteristics of the μGA approach. The nominal convergence accelerates finding process and the general convergence resets the algorithm and explore a new space of solutions. The second and main advantage is that the individual chains were segmented into bytes to find the solution in a shorter time, rather than one individual of 32 bits. The time to find the solution was reduced with the μGA from 0.7687 s to 0.131 s, as can be seen in Figure 18.

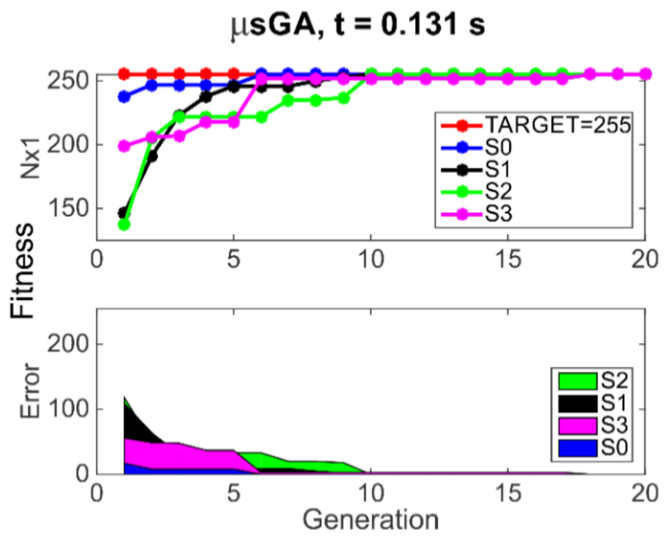

Figure 18. Evolution to a solution with the segmented μGA approach

In the design of the GA, the time required to generate individuals is lower for an 8-bit string than for a 32-bit string. If an individual is a large string, the processing time needed to generate the population is longer; this concept allowed us to segmented the individuals, and required less time (0.131 s) to generate the population, as can be seen in the last experiments with the μsGA.

Another concept is the application of the genetic operators. Processing time can be reduced by eliminating cyclic structures without neglecting the application of the operator to each individual. For example, when applying the mutation, individuals with 32 bits must be accessed 32 times by every other individual, as shown in Figures 11 and 12. The goal is to learn to walk as soon as possible. Table 2 summarizes the results of the experiments and describes the performance of the algorithms in terms of finding the solution, based on the time and numbers of generations.

Table 2. Summary of experimental results

Algorithm	**GA**	**μGA**	**μAIS**	**μsGA**
Generations	911	73	55	21
Time (seconds)	7.188	0.765	0.327	0.131

Two micro-algorithms were evaluated for robot locomotion: a μGA with 30-bit strings and the μsGA. In each experiment, the algorithm was executed ten times and the best individual was selected based on its fitness. The algorithm stops when finds the solution. The program determines the number of generations and the time required to find the solution, as shown in Table 3. In the case of μGA, the individuals are strings of 32 bits. The algorithm is executed 10 times for each configuration, and each generation is made up of 12 individuals

Table 3. Solution times for μGA with 30-bit strings

Result	**Generations**	**Time (s)**
Best	64	0.5800
Worst	163	0.9390
Average	**106.1**	**0.7687**
SD	33.12	0.1182

The solution was found in times ranging between 0.58 s and 0.9390 s, and numbers of generations of between 64 and 144. In the second case, individuals were segmented into four bytes for the μsGA approach, and the results are shown in Table 4.

Table 4. Solution times from μsGA with segmeted strings

Result	Generations	Time (s)
Best	183.00	0.1860
Worst	375.00	0.6690
Average	**280.30**	**0.4298**
SD	56.91	0.1505

Solutions were found within times ranging between 0.186 s and 0.669 s, and numbers of generations between 183 and 375. The time required to find the solution is reduced in μsGA. As shown in Table 4, the solution is obtained in less than one second that allows the robot to execute locomotion. Solution times are in the range 0.186 to 0.669 s, i.e. less than in the first experiment. However, the average numbers of generations used to find the solution for this case increases by a factor of 2.5. This is as expected, since four values need to be found per individual.

The second experiment used sequential versus parallel processing. The speed of movement of the quadruped robot using the sequential algorithm was between 3.1 and 3.3 cm/s with an impulse time of 70 ms. This program has an operating range of t_I =70 to 80 ms.

Table 5. Implementation times for sequential vs parallel versions

	Sequential		Parallel	
Distance (m)	Time (s)	Speed (m/s)	Time (s)	Speed (m/s)
1	30	0.0333	23	0.0434
1	31	0.0322	23	0.0434
1	32	0.0312	21	0.0476
1	30	0.0333	23	0.0434
1	31	0.0322	**20**	**0.0500**

The speed of the quadruped robot with the parallel algorithm was between 4.3 and 5 cm/s with a fixed value of t_I=70 ms and an operating range of 60 to 150 ms. The results for both implementations are presented in Figure 19. Execution times for the sequential and parallel programs are presented for the experiment to measure the movement speed of the robot (Table 5). The parallel program gives a speed that is 38 % higher than the sequential one, and the value of I_t increases significantly to a range of between 60 and 150 ms.

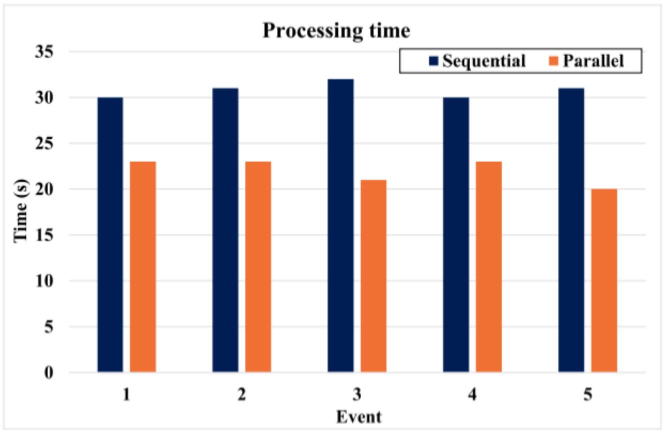

Figure 19. Performance times sequential versus parallel processing.

9. Conclusions

The experimental results obtained from a performance comparison of bio-inspired micro algorithms applied to robot locomotion show that the proposed μsGA scheme obtained solutions within a shorter processing time, and can be applied to individuals made up of binary chains. This avoids the decoding process, and uses fewer cyclic structures when the genetic operators of mutation and crossover are applied.

One contribution of this research in terms of hardware was the use of a minimal system with an eight-core architecture rather than a computer, which enabled parallel processing by the four cores of the embedded system. This enabled us to split the tasks and reduce the processing times. The characteristics of the system improved i.e. the solutions to robot locomotion were found in less time with parallel processing. The system also has a greater range of operation as the range of the impulse time increased. The proposed μsGA scheme is promising, as it can be implemented in new multi-core embedded systems and generates continuous robot locomotion. The main conclusions from this work are as follows:

1) Our proposed μsGA has the best time performance of the four algorithms tested.
2) A minimal eight-core system was used for the algorithm for learning to walk.
3) The robot's locomotion was found in less time using parallel processing.
4) The use of chain segmentation improves performance.

This work will continue in the future, with the robot learning to walk on different kinds of surfaces. The system will save parameter data for walking, identifying the surface and configure itself. We also have the option to focus on improving performance 4 by standardizing the genetic operators, population size and generations ; if this micro algorithm is programmed in assembly language, this will allow a more efficient use of embedded processor memory and thus reduce the execution time.

10. Author Contributions

Author Contributions: F.A. and J. S. conceived the method and wrote the manuscript draft;, J. C., M.O. and D. L. helped to modify it; M.H. and I.R. performed the experiments and analysed the data. All authors have read and approved the final manuscript.

11. Acknowledgments

The authors are grateful to the editor and the reviewers for their valuable comments and suggestions, which have allowed us to significantly improve this paper. The authors would like to thank Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC), Instituto Tecnológico de Tlalnepantla del TecMN, Universidad Autónoma Metropolitana Lerma and CONACYT for financial support for this research.

Abbreviations

The following abbreviations are used in this manuscript:

GA	Genetic Algorithm
μsGA	Micro Segmented Genetic Algorithm
μAIS	Micro Artificial Immune System
ABS	Acrylonitrile Butadiene Styrene
mES	multicore Embedded System
EAs	Evolutionary Algorithms
CPG	Central Pattern Generator
RWP	Robot Walking Pattern

References

1. S.-H. Liu. , M. Mernik and M. Crepinšek, "Exploration and exploitation in evolutionary algorithms: A survey," ACM Computing Surveys (CSUR), vol. 45, no. 3, pp. 1-35, 2013.
2. W. Liu, L. Zhou, H. Qian and Y. Xu, "Turning strategy analysis based on trot gait of a quadruped robot" 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO), Macau, pp. 1306-1311, 2017. doi: 10.1109/ROBIO.2017.8324598.
3. A. Mutka, M. Orsag and Z. Kovacic, "Stabilizing a quadruped robot locomotion using a two degree of freedom tail," IEEE, 21st Mediterranean Conference on Control and Automation, pp. 1336- 1342, 2013.
4. D. E. Golberg, "Genetic algorithms in search, optimization, and machine learning," Addison Wesley, vol. 1989, p. 102, 1989.
5. A. Asteroth and A. Hagg. "How to successfully apply genetic algorithms in practice: Representation and parametrization," IEEE Int. Symp. on Innovations in Intelligent Systems and Applications (INISTA), pp. 390-395, 2015.
6. E. Burke, M. Gendrau and M. Hyde, "Hyper-heuristics: A survey of the state of the art", Journal of the Operational Research Society, vol. 64, no. 12, pp. 1695-1724, 2013.
7. F. Ahmadi, R. Tati, S. Ahmadi and V. Hossaini, "New Hardware Engine for Genetic Algorithms," 2011 Fifth International Conference on Genetic and Evolutionary Computing, Xiamen, pp. 122-126, 2011. doi: 10.1109/ICGEC.2011.37.
8. P. Gumiel-Moreno, "Implementación de técnicas de computación evolutiva a la programación automática de un robot", Universidad Carlos III de Madrid, Madrid España, 2009.
9. H. Suzuki, H. Nishi, A. Aburadani and S. Inoue, "Animal Gait Generation for Quadrupedal Robot," Second International Conference on Innovative Computing, Information and Control, Second International Conference on p. 20, 2007. doi: 10.1109/ICICIC.2007.169.
10. G. B. Parker and W. T. Tarimo, "Using Cyclic Genetic Algorithms to Learn Gaits for an Actual Quadruped," 2011 IEEE International Conference on systems, Man, and Cybernetics, Anchorage, AK, pp. 1938-1943, 2011. doi: 10.1109/ICSMC.2011.6083871.
11. C. Cai and H. Jiang, "Performance Comparisons of Evolutionary Algorithms for Walking Gait Optimization," 2013 International Conference on Information Science and Cloud Computing Companion, Guandzhou, pp. 129-134, 2013. doi: 10.1109/ISCCC.2013.100.
12. M. Oliveira, C Santos and L. Costa, "Sensitivity analysis of multi-objective optimization of CPG parameters for quadruped robot locomotion," AIP Conference Proceedings. 1479. pp. 495-498, 2012. doi: 10.1063/1.4756175.
13. Ren, D.; Shao, J.; Sun, G.; Shao, X. The Complex Dynamic Locomotive Control and Experimental Research of a Quadruped-Robot Based on the Robot Trunk. Appl. Sci. 2019, 9, 3911.
14. P. Gonzalez de Santos, E. Garcia and R. Ponticelli, "Minimizing energy consumption in hexapod robots," Advanced Robotics, vol. 23, no. 6, pp. 681-704, 2009. doi: 10.1163/156855309X431677.

15. L. Teng, X. Wu, W. Chen and J. Wang, "Center of gravity balance approach based on CPG algorithm for locomotion control of a quadruped robot," 2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Wollongong, NSW, pp. 325-329, 2013. doi: 10.1109/AIM.2013.6584112.
16. W. Sheng, G. Howells, M. Fairhurst and F. Deravi, "A Memetic Fingerprint Matching Algorithm," in IEEE Transactions on Information Forensics and Security, vol. 2, no. 3, pp. 402-412, 2007. doi: 10.1109/TIFS.2007.902681.
17. F. A. Chavez-Estrada, J. Sandoval Gutierrez, J. C. Herrera Lozada, J. A. Alvarez Cedillo and M. Olguin Carbajal "The electric current as key parameter in the locomotion of a quadruped robot," DYNA New Technologies España, vol. 3, no. 1, pp. 12-20, 2016. doi: http://dx.doi.org/10.6036/NT8074.
18. X. Peng, X. Zheng, B. Wang, C. Zhou and Q. Zhang, "A micro-genetic algorithm for DNA encoding sequences design", 2nd International Conference on Control Science and Systems Engineering (ICCSSE), Singapore, pp. 10-14, 2016. doi: 10.1109/CCSSE.2016.7784342.
19. J. C. Herrera-Lozada, H. Taud and H. Calvo, "A micro artificial immune system", Polibits Journal, no 43, pp. 107-111, 2011. doi: 10.17562/PB-43-15.
20. J. R. Koza, "Genetic programming: On the programming of computers by means of natural selection", Kluwer Academic Publishers, vol. 1, 1994. doi: 10.1007/BF00175355.
21. S. Rutishauser, A. Sprowitz, L. Righetti and A. J. Ijspeert, "Passive compliant quadruped robot using Central Pattern Generators for locomotion control", 2nd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, Scottsdale, AZ, pp. 710-715, 2008. doi: 10.1109/BIOROB.2008.4762878.