Article

Performance Comparisons of Bio-Micro Genetic Algorithms on Robot Locomotion

Francisco A. Chávez-Estrada ^{1,*} , Jacobo Sandoval-Gutiérrez ^{2,*} , Juan C. Herrera-Lozada ³,
Mauricio 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, Mexico

² Procesos Productivos, Universidad Autónoma Metropolitana, Unidad Lerma (UAML), Lerma de Villada, Estado de México CP 52006, Mexico; d.martinez@correo.ler.uam.mx

³ Instituto Politécnico Nacional, Centro de Innovación y Desarrollo Tecnológico en Cómputo (CIDETEC-IPN), Ciudad de México 07700, Mexico; jlozada@ipn.mx (J.C.H.-L.); molguinc@ipn.mx (M.O.-C.); mbolanos@ipn.mx (M.H.-B.); irivera@ipn.mx (I.R.-Z.)

* Correspondence: frchaveze1400@alumno.ipn.mx or fachavez@ittla.edu.mx (F.A.C.-E.); j.sandoval@correo.ler.uam.mx (J.S.-G.)

Received: 3 May 2020; Accepted: 27 May 2020 ; Published: 2 June 2020

Abstract: This paper presents a comparison of four algorithms and identifies the better one in terms of convergence to the best performance for the locomotion of a quadruped robot designed. Three algorithms found in the literature review: a standard Genetic Algorithm (GA), a micro-Genetic Algorithm (μ GA), and a micro-Artificial Immune System (μ AIS); the fourth algorithm is a novel micro-segmented Genetic Algorithm (μ sGA). This research shows how the computing time affects the performance in different algorithms of the gait on the robot physically; this contribution complements other studies that are limited to simulation. The μ sGA algorithm uses less computing time since the individual is segmented into specific bytes. In contrast, the use of a computer and the high demand in computational resources for the GA are avoided. The results show that the performance of μ sGA is better than the other three algorithms (GA, μ GA and μ AIS). The quadruped robot prototype guarantees the same conditions for each test. The structure of the platform was developed by 3D printing. This structure was used to accommodate the mechanisms, sensors and servomechanisms as actuators. It also has an internal battery and a multicore Embedded System (mES) to process and control the robot locomotion. The computing time was reduced using an mES architecture that enables parallel processing, meaning that the requirements for resources and memory were reduced. For example, in the experiment of a one-second gait cycle, GA uses 700% of computing time, μ GA (76%), μ AIS (32%) and μ sGA (13%). This research solves the problem of quadruped robot's locomotion and gives a feasible solution (Central Pattern Generators, (CPGs)) with real performance parameters using a μ sGA bio-micro algorithm and a mES architecture.

Keywords: Central Pattern Generators; micro-segmented Genetic Algorithm; multicore Embedded System

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. Given the strengths of these algorithms, some researchers have applied GA approaches to solving the problem of locomotion of legged robots. Chae and Park [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).

The GAs are the most representative instances of the Evolutionary Algorithms (EAs). These population algorithms 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. The 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 [3]. Initially, individuals are randomly generated, and genetic operators are applied to each individual, 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 [4] and Burke et al. [5]. Olteanu and Paraschiv mentioned in [6] that the random numbers have the advantage of simplicity and easy implementation in embedded applications, in this sense, it is our case of GA dependence on a randomly generated population of individuals. There is a specific computational cost linked to the size of the population, which increase the memory usage, computing 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 shows the operation of a standard GA for robot locomotion.

Figure 1. Standard genetic algorithm flowchart for the robot's locomotion.

2. Background

The GA is applied to find solutions to problems of optimization at systems with various DOF and in which traditional methods are not easily adjusted. In the case of robot locomotion, Shriyam developed a gait analysis based on the finite state modelling of an alligator-inspired robot [8]. Gumiel applied a GA to a robot called Aibo [9]. In [10], Suzuki performed an optimization of the impulse trajectory of the legs by using a GA. Parker and Torino [11] applied a cyclic GA to generate a running sequence.

The research study in [12], optimization, speed and learning 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 Differential Evolution (DE) approach. The results were compared, and it was found that none of these algorithms was significantly better than the others. In [13], 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. [14] a gait generator based on CPG modulated was studied. Other researchers have focused their efforts [15] on energy optimization by improving the trajectories of a robot when walking using a GA. In [16], a GA was used to produce a CPG and solve the problem of robot's locomotion with legs. Another research study [17] 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 robot's locomotion with legs, the use of computers as a primary control loop is widespread. 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, to reduce the computing time to less than one second and solve 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 [18]. In [19], a μ GA was developed with a sharing function based on the similarity and H-measure of DNA sequences. This algorithm obtained better DNA sequences and improved computational efficiency. Concerning to the intrinsic execution of bio-inspired algorithms in embedded systems, a micro-artificial immune system (μ AIS) was presented in [20] to solve numerical optimization problems; in this micro algorithm, a cloning scheme was proposed with a population of only five individuals (antibodies in the artificial immune system (AIS) context) with the purpose of hardware implementation.

A new proposal in the context of existing GA recently was developed in [21] where an individual with a variable chromosome Genetic Algorithm (vcGA) is simulated, and the authors demonstrated that for a complex situation vcGA is better than GA. In this sense, our applied proposed algorithm with limited hardware resources also is a non-traditional comparison of performance. In this study, we propose an alternative method of using μ GA with small populations and use segmented elements or individuals to reduce the size of the binary chain. This reduces computing time to allow solution search to less than one second. Therefore, we guarantee similar performance with GA. We use a binary chromosome chain and an embedded 8 MHz system to solve the problem of robot locomotion using parallel programming. Thereby, the system finds solutions 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. Another essential idea is shown in [22] where the excessive external non-controlled parameters could cause hardware damage and not only a task failure, these factors are a change of the robot's load, lousy contact with the ground, self-disturbance, among others.

The new algorithm, called the micro-segmented Genetic Algorithm (μ sGA), segments the individual chain into bytes and applies evolutionary techniques to each, so reducing the computing time to less than 0.5 s. We hypothesize that reducing the number of individuals in a population will decrease the number of evaluations of the objective function and consequently, increasing the speed of convergence.

The main contributions of this paper are as follows: (i) the proposed μ sGA outperforms other GA, μ GA and μ AIS 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; (iv) improved performance is seen due to the use of parallel processing; (v) the impulse time values in CPG are automatically changed to achieve to the gait cycle; and (vi) the comparison in a physical robot was made, and the heuristic configuration avoids the risk of a hardware damage.

The remainder of the paper is organized as follows. Section 3 presents the novel micro-segmented genetic algorithm. Section 4 introduces the design and organization of the robotic platform. Section 5 describes the setup of the experiments for trot gait locomotion. Sections 6 and 7 present the experimental results and a discussion. Finally, conclusions are given in Section 8.

3. Micro-Segmented Genetic Algorithm

Darwinian natural selection theory in the computing context means that reproductive success is associated with biological entities in nature on how strong and fit they are to adapt to an environment, simplifying a “survival of the fittest” [23]. This parameter can be measured in four ways: raw, standardized, adjusted and normalized fitness [24]. In this applied research, the natural terminology mentioned is the value of the displacement angle for each robot’s link and impulse time; for this reason, we used a specific measure with raw fitness. The individuals of the population are represented by 32-bit binary chains (2^{32} decimal value) and the higher values survive and reproduce to generate a population that evolves towards the solution (CPG ideal). The chromosomes or individuals generated in the algorithm are tested to determine their fitness toward the CPG ideal.

This measurement controls the ability of 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, trotting or galloping.

Thus, the individual or fitness or reference is the decimal value represented by $r(i, g)$ for an individual i in a generation g in the Equation (1).

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

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 μ GA begins by identifying and describing the parameters that determine the robot locomotion.

A GA generates a random population of individuals represented by the values $c(i, j)$, made up of 32-bit binary strings or 4 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. It 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 contains the values required by the servomechanisms to produce the Robot Trotting Pattern (RTP), see Figure 2.

The leg movements depend on a combination of links l_1 , l_2 and l_3 that Θ_1 , Θ_2 and Θ_3 determine them. Changes in speed are determined by the time of impulse (T_I) or duty factor D_f ; this is a time interval in which a robot leg is touching the surface.

The maximum displacement is obtained when the links have an angular value: $\Theta_1 = 90^\circ$ (fixed value), $\Theta_2 = 20^\circ$ and $\Theta_3 = 40^\circ$. In each case, the maximum theoretical value of the angle that each link of the robot can reach is 255 in decimal value, corresponding to 1111 1111 in binary notation, due to the constraints on the system. In [25] used a Duty factor $= 0.6 \times T$ and $T = 1$ second, with this reference, the maximum theoretical was determined to $T_I = 0.6 \times T$ or 1111 1111 in binary notation. 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 movements of the servomechanisms are controlled by pulse width modulation signals. These signals are generated by the μ GA 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 at the same time in all four legs. Consequently, this combination determines the robot's locomotion (walking, trotting or galloping).

Figure 2. Relationship between the links in a leg and the binary structure of the chromosome or individual.

3.1. Micro-Segmented Genetic Algorithm Description

In the design of the genetic algorithm, we use the concepts described by Goldberg [3] regarding the μ GA. Then the μ GA evaluates the fitness of each individual, and the best individuals are achieved 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 robot's locomotion.

An objective of this research is to apply the genetic operators (mutation and crossover) with the aim of reducing the computing time. In this case, we also segment the individuals into four chains of 8 bits by applying mutation and indexing the four chains. Only the string of every individual will be accessed by eight changes, as shown in Figure 3.

Figure 3. Mutation process comparison using segmented individual.

When applying the genetic crossover operator to individuals of 32 bits, this implies 16 changes for every individual. 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, as shown in Figure 4.

Figure 4. Crossover process comparison using segmented individual.

When processing segments directly, the decoding process is not required, which saves computing time. Applying the genetic operators of mutation and crossover segmentation reduces the computing 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 time.
3. Applying the genetic operators of mutation and crossover segments reduces computing time if we index the four segments of each individual.

In Figures 3 and 4, the individual segmentation is represented by four bytes, and the μ GA is applied to each segment to reduce the computing time. Which is an algorithm based on small populations of individuals. It was determined that a population of 12 individuals ensures fast convergence to the maximum theoretical value.

3.2. Micro-Segmented Genetic Algorithm Process

The μ sGA starts with a random population of 12 individuals, and the best is copied to the initial population after one random crossover and one random mutation of a bit for each segment selected are applied. The 12 individuals of the active population remain and are evolved n times to reach the maximum. When it is reached, the best individual is copied to the next initial population, and for optimal convergence, the 11 remaining individuals will be generated at random. A theoretical maximization on the μ sGA evolution process is executed with the internal and external cycle of μ sGA to achieve better results. The internal cycle achieves the nominal convergence (the best four individuals in five generations), and the external cycle will terminate when the optimal or general convergence reaches the stop criterion (The individual that satisfies the performance desired or the reference value of CPG) which will generate the maximum displacement in the robot links and will try to maintain a straight line. A flowchart for the algorithm using these concepts is given in Figure 5.

Figure 5. micro-segmented Genetic Algorithm (μ sGA) process flowchart with internal and external cycle.

Several other experiments were tested to the genetic operators, such as two mutations per segment, two crossovers per segment, increasing or decreasing the population by generation and so on, but we did not find any suitable conditions that improved the performance. Then, the characteristics of this μ sGA process with a small population, reach the best results for nominal or general convergence when one mutation and crossover are used. Finally, a test is shown in Figure 6, where one mutation and crossover is better than two mutations and crosses in terms of generations or computing time.

Figure 6. Comparison of μ GA with one mutation and crossover vs. two mutations and crossover.

4. Design of the Robotic Platform

In a review of legged robots [26], can be classified into mammal robots and reptile robots based on the mechanical structure. In this research, the mechanical of mammal robot was used for the design of the quadruped robot prototype (see Figure 7).

Figure 7. Quadruped robot platform with 12 degrees of freedom.

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 8).

Figure 8. Mechanical design of a robot leg.

The robot locomotion was unstable due to the thinness and rigidity of the pieces causing deformations and stresses in the links. Thus, the Finite Element Method (FEM) was applied to solve this problem, and static analysis of the parts was carried out using the SolidWorks application to detect the stresses in the links (Figure 9).

Figure 9. Mechanical design of a link using the finite element method.

Based on the previous work [18], the length, width, thickness, materials and settings of building considerations were analyzed. 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 a thickness of 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 m. The dimensions of the reinforced pieces are shown in Table 1.

Table 1. Dimensions of the robot parts with isometric projection.

Part	Length L (m)	Width W (m)	Thickness T (m)	Centers (m)	Isometric Projection
Rectangular base	0.190	0.160	0.004	0.110
Link ₁ (d)	0.060	0.045	0.003	0.035
Link ₂ (a2)	0.095	0.040	0.014	0.070
Link ₃ (a3)	0.060	0.030	0.004	0.045

A quadruped platform was used to allow the installation of I/O electronics components, mES, servomotors (actuators) 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 had a 2.0 kg/cm torque, consumption of 0.16 A, and a 5 V source. The final version is shown in Figure 10. (Short video can be watched at the Supplementary Materials).

Figure 10. Final functional quadruped robot prototype.

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 11.

Figure 11. Flowchart for the sequential algorithm.

The sequential algorithm was executed on a parallax embedded system board, and in the first task, the GA was executed to find the best performance. In the second task, the CPG signals were generated at the same time that generating the electrical signals to move the actuators Θ_1 , Θ_2 , Θ_3 and a pulse t_1 . The values were applied sequentially to legs 1, 3, 2 and 4, in this order, which produced the robot's locomotion in a trotting pattern.

The parallel algorithm was executed on the same embedded multi-core system, which had eight processor cores and could divide a task or execute different tasks at the same time. A flowchart of the process is given in Figure 12.

Figure 12. 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 executed the μ sGA, found the reference value, and generated the CPG for locomotion.
2. The second core took the signals from the CPG and produced the electrical signals for the actuators of legs 1 and 3.
3. The third took the signals from the CPG and produced the electrical signals for the actuators of legs 2 and 4.
4. The fourth core processed the sensor signals.

Each core executed a different task at the same time, in parallel. This reduced the computing time in comparison to sequential computing.

5. Trot Gait Locomotion

The system must be reconfigured until the maximum displacement of the links is reached. The system generates a pattern of movements or CPG (walking, trotting or galloping), and these are designed using coupled oscillators [27] that generate a sinusoidal function (see Equation (2) and Figure 13).

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

Figure 13. Sine oscillator for GCP.

$temp.individual[i]$ is the binary value of the best individual resulting in the evolution of the μ sGA. 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) which is equivalent to the maximum theoretical amplitude that the CPG can reach for each link. Conversely, the worst individual is 0 (0000 0000) on the minimum CPG amplitude. The combination of the angle displacement for each link (AD_j) generates the CPG output as a physical effect, where $j = 1, 2$ and 3 are the links and their angles (shown in Figure 2) are $\Theta_1 = 90^\circ$, $\Theta_2 = 20^\circ$ and $\Theta_3 = 40^\circ$ respectively. Therefore, the angle displacement (AD_j) is obtained in Equation (3) with a normalization operation from Equation (2).

$$AD_j = q(j) * \frac{\Theta_j}{255} \quad (3)$$

with $j = 1, 2$ and 3 .

The trot gait locomotion of the robot considered two stages; in the first stage: 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° concerning each leg based on their distribution in a clockwise direction. The temporal correlation reference is generated between the legs when trotting gives rise to the phase relationships with 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 trotting, two legs are on the ground, and two are in the air, as shown in Figure 14.

Figure 14. Phase diagram for trotting.

In the second stage, the robot executes particular learning using a gyroscope sensor, which allows it to find the optimum impulse time (T_I). The impulse time T_I is a parameter not fixed of the system. This is achieved by comparing the ideal displacement (by simulation of the direct kinematics of the system) versus the real displacement direction using the gyroscope sensor.

When the robot was tested in trotting gait using both synchronized stages to achieve an ideal CPG (reference signal), we found a performance difference between ideal simulations reported and the experiments in real conditions (precision of sensors, quality of actuators, and all perturbations inherent to the machines were seen). In this sense, T_I was the parameter that stabilized the CPG until the robot found an acceptable trotting gait. These experiments were using a Gait cycle $T = 1s$ and a normalized reference of CPG, as shown in Figure 15.

Figure 15 shows the differences between the real and ideal positions during locomotion of the robot. The $T_I = 90$ ms is adopted as the parameter for locomotion. The error decreases, and the system tries to follow the ideal locomotion function.

Figure 15. Central Pattern Generator (CPG) reference vs CPG real of the trotting gait (a) and automatic adjust of T_I in ms (b).

6. Experiments

In the first stage, the four bio-algorithms are evaluated by applying them to the locomotion of the robot. The following same values, conditions and process were 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 μ sGA 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 reference value is found.
9. Each algorithm is executed 10 times, and the average result was stored. Each algorithm is implemented separately using the same prototype quadruped robot and executed on the mES.
10. The GA, μ AIS and μ GA are executed to identify the generation (x -axis) in which convergence to the reference value is reached, and the corresponding fitness of the best individual (y -axis). The fitness of the first three algorithms is calculated as 1,073,741,824 decimal or 30-bit individual length, and the fourth algorithm is calculated with four individuals of 268,435,456 decimal or 28-bit individual length (30-bit/4 = 28-bit). In the case of μ AIS, the proposal of Herrera et al. was used [20], which considers five antibodies of 30-bit length each and the same cloning and stranding operators.

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

1. An impulse time $T_I = 70$ ms was set for both processes.
2. The μ sGA sequential algorithm shown in Figure 5 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.

7. 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 trotting CPG for robot's locomotion was found (Figures 16–19 show these performances).

The GA showed the worst performance of around 7.188 s, with 911 generations needed to obtain the trotting CPG for robot's locomotion. In these conditions, learning to trot was slow; it was not continuous, and 7.188 s was a long time for the generation of continuous locomotion (see the information in Figure 16).

Figure 16. Performance of Genetic Algorithm (GA) standard with a reference of 30-bit length. (a) fitness and (b) absolute error.

The μ GA and μ AIS schemes have better performance than the GA algorithm and find the trotting CPG for robot's locomotion in less than one second since each the trotting CPG for robot's locomotion was found in 73 generations (Figure 17).

Figure 17. Cont.

Figure 17. Performance of μ GA with a reference of 30-bit length. (a) fitness and (b) absolute error.

The μ AIS scheme found the trotting CPG for robot's locomotion in 0.327s and 55 generations (Figure 18).

Figure 18. Performance of μ AIS with a reference of 30-bit length. (a) fitness and (b) absolute error.

Our proposed μ sGA scheme found the trotting CPG for robot's locomotion in the shortest time of 0.131 s and 21 generations. The μ sGA had several advantages; for example, it had the characteristics of the μ GA approach. The nominal convergence accelerated finding process and the general convergence reset the algorithm and explored a new space of values. The second and main advantage was that the

individual chains were segmented into bytes to find the reference values in a shorter time, rather than one individual of 32 bits. The time to find the CPG trot gait was reduced with the μ GA from 0.7687 s to 0.131 s, as can be seen in Figure 19.

In the design of the GA, the time required to generate individuals was lower for the 32-bit individual divided by four 8-bit string than for a 32-bit string. If an individual was a large string, the computing time needed to generate the population was 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 μ sGA.

Another concept is the application of genetic operators. Computing time can be reduced by eliminating cyclic structures without neglecting the use 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 3 and 4. The goal was 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.

Figure 19. Performance of μ sGA with a reference of 30-bit segmented in 4 bytes. (a) fitness and (b) absolute error.

Table 2. Summary of experimental results.

Algorithm	GA	μ GA	μ AIS	μ sGA
Generations	911	73	55	21
Time (s)	7.188	0.765	0.327	0.131

In the paper of Stability Criterion for Dynamic Gaits of Quadruped Robot [25], the trot-gait parameters are shown; in particular, the duration of a gait cycle reported is 1 second. If it is compared with the results of Table 2, we can explain that computing time for GA is not suitable, μ GA uses 76% of a cycle, μ AIS the 32% and the best time is for μ sGA with 13%.

Two micro genetic algorithms were evaluated for robot locomotion: a μ GA with 30-bit strings and the μ sGA (four 8-bit strings). In each experiment, the algorithm was executed 10 times, and the best individual

was selected based on its fitness. The algorithm stopped when it found the reference value. The program determines the number of generations and the time required to find the reference value, as shown in Table 3. In the case of μ GA, the individuals are 32-bit strings. The algorithm was executed 10 times for each configuration, and each generation was made up of 12 individuals.

Table 3. Computing times for μ GA with 32-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 reference value 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 μ sGA approach, and the results are shown in Table 4.

Table 4. Computing times from μ sGA with segmented 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

The reference values 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 reach the reference value was reduced to μ sGA. As shown in Table 4, the reference value was obtained in less than one second that allowed the robot to execute locomotion. Computing times were 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 reference value for this case increased by a factor of 2.5. The factor was 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. 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 20. Execution times for the sequential and parallel programs are shown for the experiment to measure the movement speed of the robot (Table 5).

Table 5. Results for sequential vs parallel processing experiments.

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 parallel program gave a rate that was 38% higher than the sequential one, and the value of T_l increased significantly to a range of between 60 and 150 ms.

Figure 20. Computing times sequential versus parallel processing.

8. Conclusions

The experimental results obtained from a performance comparison of bio-inspired algorithms applied to robot's locomotion show that the proposed μ sGA scheme reached the reference value (fitness) within a shorter computing 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.

The proposed μ sGA scheme is promising, as it can be implemented in new multi-core embedded systems and generates continuous robot's locomotion. Thus, a 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 implementation helped us to split the tasks and reduce the computing times. The characteristics of the system improved, i.e., the reference value in the trot gait to robot's locomotion were found in less time with parallel processing. The system also has a more excellent range of operation as the range of the impulse time increased; in other words, the robot can achieve a fast trot performance.

In many papers reviewed in the literature use the simulation carefully to prove their results, one reason is the cost of hardware damage to the motors, legs and joints when the experiments are executing. However, in this research, after setting the platform with the four bio-inspired algorithms have been reduced to this problem.

The main conclusions from this work are as follows:

- (1) Our proposed μ sGA has the best time performance of the four algorithms tested.
- (2) A minimal eight-core system was used in 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.
- (5) A real trotting gait was successful with fewer resources: compute and energy consumption.
- (6) It was possible to jump from simulation to practice with an autonomous robot operation by bio-inspired solutions; as an alternative of classic techniques reported.

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 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.

Supplementary Materials: A short video of the performance is available online at <http://www.mdpi.com/2076-3417/10/11/3863/s1>.

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

Funding: This research received no external funding.

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 (CIDE TEC), Instituto Tecnológico de Tlalnepantla del TecMN, Universidad Autónoma Metropolitana Lerma and CONACYT for financial support for this research.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GA	Genetic Algorithm
μ sGA	micro-segmented Genetic Algorithm
μ AIS	Micro Artificial Immune System
ABS	Acrylonitrile Butadiene Styrene
mES	multicore Embedded System
EAs	Evolutionary Algorithms
CPG	Central Pattern Generator
RWP	Robot Trotting Pattern

References

1. Liu, S.-H.; Mernik, M.; Crepinšek, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* **2013**, *45*, 35. [\[CrossRef\]](#)
2. Chae, K.G.; Park, J.H. Trajectory optimization with GA and control for quadruped robots. *J. Mech. Sci. Technol.* **2009**, *23*, 114–123. [\[CrossRef\]](#)
3. Golberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1989.
4. Asteroth, A.; Hagg, A. How to successfully apply genetic algorithms in practice: Representation and parametrization. In Proceedings of the IEEE International Symposium on Innovations in Intelligent Systems and Applications, Madrid, Spain, 2–4 September 2015; pp. 1–6.
5. Burke, E.; Gendreau, M.; Hyde, M.; Kendall, G.; Ochoa, G.; Özcan, E.; Qu, R. Hyper-heuristics: A survey of the state of the art. *J. Oper. Res. Soc.* **2013**, *64*, 1695–1724. [\[CrossRef\]](#)
6. Olteanu, M.; Paraschiv, N. The influence of random numbers generators upon Genetic Algorithms. In Proceedings of the IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Albena, Bulgaria, 19–21 June 2013; pp. 1–5.

7. Ahmadi, F.; Tati, R.; Ahmadi S.; Hossaini, V. New Hardware Engine for Genetic Algorithms. In Proceedings of the Fifth International Conference on Genetic and Evolutionary Computing, Xiamen, China, 29 August–1 September 2011; pp. 122–126.
8. Shriyam, S.; Mishra, A.; Nayak, D.; Thakur, A. Design, Fabrication and Gait Planning of Alligator-Inspired Robot. *Int. J. Curr. Eng. Technol.* **2014**, *2*, 567–575. [\[CrossRef\]](#)
9. Gumiel-Moreno, P. IMplementaci3N De TéCnicas De Computaci3N Evolutiva a La Programaci3N AutomáTica De Un Robot. Master's Thesis, Universidad Carlos III de Madrid, Madrid, Spain, 2009.
10. Suzuki, H.; Nishi, H.; Aburadani, A.; Inoue, S. Animal Gait Generation for Quadrupedal Robot. In Proceedings of the Second International Conference on Innovative Computing, Information and Control, Kumamoto, Japan, 5–7 September 2007; pp. 20–24.
11. Parker, G.B.; Tarimo, W.T. Using Cyclic Genetic Algorithms to Learn Gaits for an Actual Quadruped. In Proceedings of the IEEE International Conference on systems, Man, and Cybernetics, Anchorage, AK, USA, 9–12 October 2011; pp. 1938–1943.
12. Cai, C.; Jiang, H. Performance Comparisons of Evolutionary Algorithms for Walking Gait Optimization. In Proceedings of the International Conference on Information Science and Cloud Computing Companion, Guandzhou, China, 7–8 December 2013; pp. 129–134.
13. Oliveira, M.; Santos, C.; Costa, L. Sensitivity analysis of multi-objective optimization of CPG parameters for a quadruped robot locomotion. In Proceedings of the AIP Conference, Kos, Greece, 19–25 September 2012; pp. 495–498.
14. 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. [\[CrossRef\]](#)
15. Gonzalez de Santos, P.; Garcia, E.; Ponticelli, R.; Armada, M. Minimizing energy consumption in hexapod robots. *Adv. Robot.* **2009**, *23*, 681–704. [\[CrossRef\]](#)
16. Teng, L.; Wu, X.; Chen, W.; Wang, J. Center of gravity balance approach based on CPG algorithm for locomotion control of a quadruped robot. In Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Wollongong, Australia, 9–12 July 2013; pp. 325–329.
17. Sheng, W.; Howells, G.; Fairhurst, M.; Deravi, F. A Memetic Fingerprint Matching Algorithm. *IEEE Trans. Inf. Forensics Secur.* **2007**, *2*, 402–412. [\[CrossRef\]](#)
18. Chavez-Estrada, F.A.; Sandoval-Gutierrez, J.; Herrera-Lozada, J.C.; Alvarez-Cedillo, J.A.; Olguin-Carbajal, M. The electric current as key parameter in the locomotion of a quadruped robot. *DYNA New Technol.* **2016**, *3*, 12–20. [\[CrossRef\]](#)
19. Peng, X.; Zheng, X.; Wang, B.; Zhou, C.; Zhang, Q. A micro-genetic algorithm for DNA encoding sequences design. In Proceedings of the 2nd International Conference on Control Science and Systems Engineering, Singapore, 27–29 July 2016; pp. 10–14.
20. Herrera-Lozada, J.C.; Taud, H.; Calvo, H. A micro artificial immune system. *Polibits J.* **2011**, *43*, 107–111. [\[CrossRef\]](#)
21. Park, K.-M.; Shin, S.-H.; Shin, D.; Chi, S.-D. Design of Warship Simulation Using Variable-Chromosome Genetic Algorithm. *Appl. Sci.* **2019**, *9*, 4131. [\[CrossRef\]](#)
22. He, J.; Shao, J.; Sun, G.; Shao, X. Survey of Quadruped Robots Coping Strategies in Complex Situations. *Electronics* **2019**, *8*, 1414. [\[CrossRef\]](#)
23. Tang, R.; Song, Q.; Fong, S.; Wong, R. Improving Metaheuristics by Natural Selection. In Proceedings of the 5th IIAI International Congress on Advanced Applied Informatics, Kumamoto, Japan, 10–14 July 2016; pp. 588–592.
24. Koza, J.R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*; MIT Press: Cambridge, MA, USA, 1992.
25. Jia, Y.; Luo, X.; Han, B.; Liang, G.; Zhao, J.; Zhao, Y. Stability Criterion for Dynamic Gaits of Quadruped Robot. *Appl. Sci.* **2018**, *8*, 2381. [\[CrossRef\]](#)

26. Li, Y.; Li, B.; Ruan, J.; Rong, X. Research of mammal bionic quadruped robots: A review, In Proceedings of the IEEE 5th International Conference on Robotics, Automation and Mechatronics (RAM), Qingdao, China, 17–19 September 2011; pp. 166–171.
27. Rutishauser, S.; Sprowitz, A.; Righetti, L.; Ijspeert, A.J. Passive compliant quadruped robot using Central Pattern Generators for locomotion control. In Proceedings of the 2nd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics, Scottsdale, AZ, USA, 19–22 October 2008; pp. 710–715.

Sample Availability: The design, code, tests and tips of this quadruped robot are available from the authors.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).