Incremental Evolution of Three Degree-of-Freedom Arachnid Gaits

Gary B. Parker, *Senior Member, IEEE*, Manan B. M. Isak, and Jim O'Connor, *Member, IEEE*

Abstract— **In this research, we evolve gaits for an arachnid-inspired robot. The method used is an expansion upon previous research on the incremental evolution of gaits for hexapod robots with two degrees of freedom per leg, which we now apply to a more complex, eight-legged robot with three degrees of freedom per leg. Incremental evolution handles gait generation for legged robots in two discrete increments. The first increment uses a cyclic genetic algorithm to learn the activations (pulse instructions to the servos) required for each leg to perform a single-leg cycle. This learning program takes into account the way each leg is mounted on the body and the range of movement provided by the three servos on each leg to produce a smooth, straight and efficient leg cycle. The second increment uses a genetic algorithm to select the best combination of leg cycles for each leg and to learn the timing to execute each leg cycle to coordinate them all together into a single gait. In this work, we learn the gait incrementally in a simulation and transfer the final gaits to the real robot to confirm the method's viability.**

I. Introduction

The development of legged robots is very important because they have multiple advantages over their wheeled counterparts including better stability and adaptability in harsh terrains. Gait generation plays an important part in the development of legged robots. Manually creating commands for legs with multiple degrees of freedom is difficult and even if successful is unlikely to produce an optimal gait that makes the most of the capabilities of the robot. In this research, we take inspiration from nature to produce efficient gaits for an eight-legged, bio-inspired, arachnid robot with three degrees of freedom per leg. While we do not use a bio-inspired controller, we predict the bio-inspired morphology of the robot will influence the learning algorithm into creating a gait similar to that of a biological spider.

The robot's capabilities need to be taken into consideration when generating a walking cycle to make the best use of its unique specifications. Here we break the gait into two discrete parts to help with gait generation: the cyclic motion of each leg and the coordination of all the legs together into one smooth gait. The leg cycles for each leg are controlled by a microprocessor assigned to that leg, which is similar to biological spiders that have nerve ganglia associated with each leg. Signals from a central processor to these leg processors produces the resultant gait.

The robot in this work uses servos for actuators, which require a pulse that dictates their exact position within the physical range of each servo's sweep. However, each pulse may need to have different durations to account for the actual movement speed of each servo. If two pulses in sequence are too far apart in value they may be instructing the servo to move further than it is physically capable of in just one pulse. Instead, a sequence of smaller pulse increments between two pulses is required to produce a smooth and accurate motion, which can then be looped to produce a movement cycle for one leg.

Using learning methods for gait generation has been a topic of research over the past several years. Recent works have used Deep Reinforcement Learning (DRL) to learn efficient hexapod gaits for legged robots with more than two degrees of freedom per leg [1]; however, the reward functions have a high degree of complexity due to the number of degrees of freedom. Learning and adjusting the reward function thus takes a substantial amount of time and computing power in addition to the time and computing power spent learning the problem itself after the hyperparameters have been refined. However, subsequent work has focused on speeding up the DRL process to mitigate these disadvantages [2]. Other works have used Central Pattern Generation (CPG) which takes inspiration from the way sensory-motor nervous systems of real-life insects handle walking [3]. While this method does make it possible to generate a very natural gait with fewer parameters, the *a priori* domain knowledge required to use this method is larger than other approaches. Further works have combined these approaches. Shafiee et al. combined DRL with CPG to reduce the complexity of the reward function, sensors, and hyperparameters. These two methods combined however still have the disadvantage of needing a lot of prior knowledge [4].

We use an incremental learning approach using a cyclic genetic algorithm because the design of the algorithm, chromosomes and fitness functions are simple, the training time is fast and it requires minimal *a priori* knowledge. The few design choices we made that incorporated prior knowledge were for reducing the complexity to allow the controller to learn faster while still giving it enough learning space to cover all possibilities. A cyclic genetic algorithm (CGA) is a variation of the standard GA in which the chromosome is a series of instructions in a loop which can also have a tail of instructions on either end where pre and post-cycle procedures can be executed. In our version, we only use the loop with no tails. We omit the start section because it was found in prior CGA research that it provided little benefit [5] and we omit the end section because in this research our goal is to produce a sustained forward gait. The final chromosome for each leg contains the cycle of pulses that when executed produce a walking cycle for one leg.

Past work has shown that efficient gaits for hexapod and arachnid robots with two degrees of freedom per leg can be learned using a CGA [6,7]. In these works the pulse widths sent to the servos were not learned, just the general directions of up/down and back/forward. Incremental learning of the more complex controllers involving the need for sequences of

G. B. Parker, M. B. M. Isak, and J. O'Connor are with the Department of Computer Science, Connecticut College, New London, CT 06320 USA (e-mail: parker@conncoll.edu, misak@conncoll.edu, and joconno2@conncoll.edu).

pulse instructions to the servos has been used for a hexapod robot with two degrees of freedom per leg [8]. In this paper, we aim to expand on this gait generation work by using incremental learning and a CGA to learn the walking gait of an eight-legged robot with three degrees of freedom per leg. The CGA is used to learn eight controllers each controlling three servos to produce the leg cycle of one of the eight legs. A GA is then used to learn one controller which controls the timing with which each leg cycle should be executed to organize a coordinated gait. Tests in simulation and on the real robot confirm the viability of this method for producing gaits.

II. Arachnid Gaits

Biological spiders use their eight legs to achieve the ability to climb over obstacles and to continue stable movement even after damaging one or two of their legs. These traits are desirable in legged robots, which is why we will strive to replicate them. We also want to mimic the speed and stability of spider gaits. In research done on spider gaits [9] it was found that the gait pattern in spiders remained constant as velocity increased and decreased. The velocity of the spider is controlled in proportion to the stride frequency whereas stride length has little effect on the final velocity of the spider. The gait used by the spider is the alternating tetrapod gait which consists of two groups of legs being on the ground at any given time. These groups are L1,R2,L3,R4 and R1,L2,R3,L4 as shown in Fig. 1. This gait pattern ensures the center of mass of the spider is within the perimeter of the large quadrilateral made by the legs and effectors on the ground. Fig. 2 shows the gait used by biological spiders, which we assume is optimal, so we predicted that the gait pattern learned for our robot will be the same since this gait is energy efficient, fast and stable.

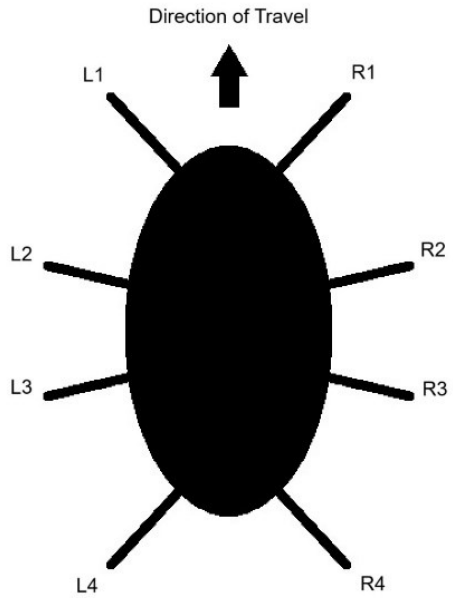

Figure 1. Top-down view diagram of the robot and its eight legs, labeled L (left) or R (right) and then 1-4 from front to back.

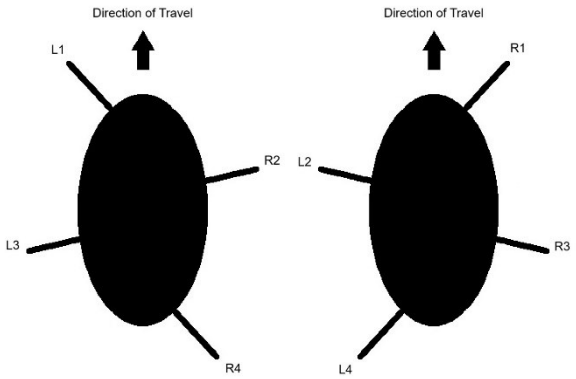

Figure 2. Alternating tetrapod gait used by biological spiders.

III. The Robot

The robot used in this research was developed in the Connecticut College Autonomous Agent Learning Lab by Sarah Dashnaw for experiments with arachnid gaits and is shown in Fig. 3. It is a custom-made eight-legged robot inspired by the ServoBot which was used in previous gait generation research [6]. The ServoBot is a hexapod robot with twelve servos and thus two degrees of freedom per leg which provides thrust and vertical movement. Like the ServoBot, this eight-legged robot design is inexpensive and easy to assemble. It also requires that the vertical servos need the legs at the extremes of their throw (fully up or fully down) to hold the weight of the robot. In addition, the servos are not strong enough to forcefully lift the robot from a fully down position to a fully up position unless more than half the legs are on the ground. This means an energy-efficient and balanced gait is required for the robot to move without collapsing under its weight. This robot differs from the ServoBot in that it has eight legs, 24 servos, and thus three degrees of freedom per leg, and the legs are placed radially on an elliptical body (instead of in-line on a rectangular body) to mimic biological spiders. The developer chose to place the legs radially to emphasize the importance of the third degree of freedom, the extension/contraction degree, in the generation of straight movement on the ground. If this third degree of freedom is not exploited by the cyclic genetic algorithm, the radial nature of the placement of the legs would cause the final gait to rotate the robot resulting in an inefficient gait.

There are certain advantages to eight-legged robots compared to their six-legged and four-legged counterparts. They can maintain better static stability at high speeds due to being able to have four legs on the ground at any given time and they can still maintain static stability even if one or two legs are broken or missing. On the other hand, disadvantages include having more moving parts and thus being harder to build and maintain, and the extra control instructions added by two more legs and increased degrees of freedom add complexity to the learning process.

The robot is controlled by eight BASIC Stamp 2s, one for each leg, and a central BASIC Stamp 2p40 to coordinate the timing of the legs. Each leg's controller can store a sequence of pulses to be executed on its respective leg's three servos. The central controller then sends signals to each of the eight leg controllers instructing them when to start their sequences

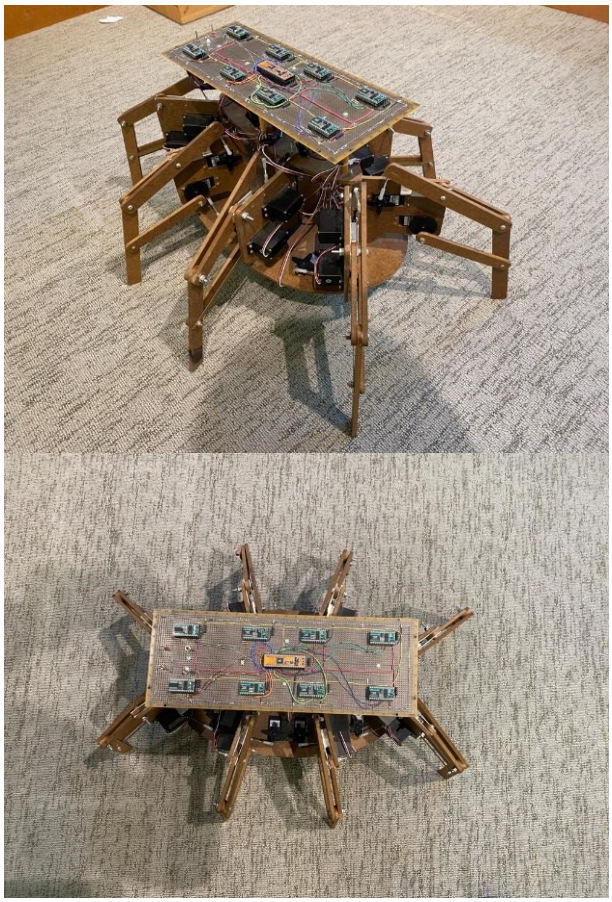

Figure 3. Images showing the robot's construction and circuitry.

and restart their sequences, whether it is after the sequence has ended or if it is early and the sequence gets cut short.

The servos can be set to specific angular positions by sending a 10 to 1,250-microseconds-long control pulse. This pulse needs to be repeated every 25 milliseconds to continually control the servo motors. If the angular position between two consecutive pulses is too different, the servo cannot move the leg fast enough to reach the desired position within one pulse. The speed of the movement can be controlled by incrementally changing the length of control pulses sent to the servos. For example, a set of pulses such as 70, 75, 80, 85, and 90 would make the servo move slower from 70 to 90 than if we'd used control pulses 70, 80, and 90. Each servo is also unique in regards to what pulse corresponds to what position. Some may have a full back position at a pulse length of 200 microseconds while others may be fully back with a pulse length of 20 microseconds. We must account for the peculiarities of each servo and their maximum speed in our learning algorithm and control program.

IV. Cyclic Genetic Algorithms

The Cyclic Genetic Algorithm (CGA) is a variation of the Genetic Algorithm (GA). Instead of using the chromosome's genes as a list of characteristics of the solution, the CGA incorporates time into the chromosome and uses each gene as a task to be executed in order. A loop can then also be incorporated over a portion of the chromosome creating a cycle. This makes a sequential program with a start section, iterative section and stop section. These differences are illustrated in Fig. 4. For our problem, we only made use of the iterative section.

V. First Increment: Evolving Leg Cycles

Our CGA needs to learn eight sequences of pulses that can be transferred to PBASIC programs, uploaded to the eight BASIC Stamp 2s, and then looped on each stamp. We chose to use fixed-length chromosomes as they are more compatible since similar genes are more likely to correspond to similar tasks. We also made some adjustments to the representation of the pulses, as being accurate to 1 microsecond in a 10 to 1,250-microsecond range was unnecessary. Pulses that differed by less than 10 microseconds were virtually indistinguishable from each other, so we chose to have our chromosome represent pulses in 10 microsecond increments. The final range was then determined to be 1-125 for each servo. This can then be represented by a 7-bit number and so a single signal to all three servos can be represented by a 21-bit number. We also found that the total number of pulse signals required to go from the minimum throw to the maximum throw was around 100. To allow for the learning algorithm to also be able to instruct one gene to move the full throw at a slower pace we decided to set the range of repetitions to be 0 to 310 in 10 pulse signal increments which can be represented by a five-bit number. We chose 310 as the new maximum number of repetitions so as to allow the controller to be able to execute up to approximately ⅓ times slower movements to increase the portion of time the leg spends on the ground and to allow the vertical servo, which always moves at maximum speed, to instruct the leg to approach and leave the ground at steeper or shallower angles to mitigate negative thrust.

Smooth movement is required for the horizontal and extension/contraction servos to avoid harsh movements and to maintain a stable and straight gait. A smooth movement can be achieved by spacing the intervals of pulses between two control pulse instructions evenly i.e., instead of having intervals of pulses like 10, 40, 50, 100 we would instead space them evenly like 25, 50, 75, 100. We account for this in our learning algorithm and the final controller by taking the two consecutive control pulses and dividing them up into

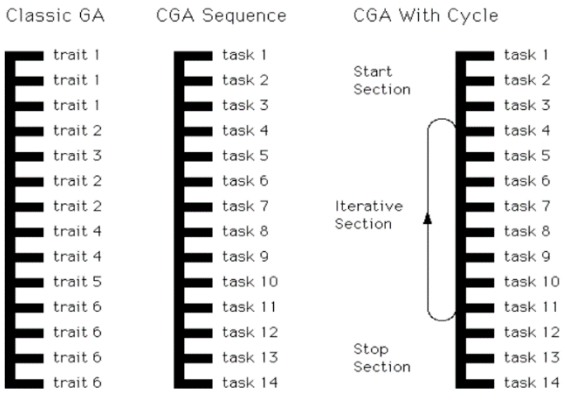

Figure 4. A GA chromosome compared with a CGA chromosome.

several evenly spaced pulses equal to the number of repetitions for that instruction, allowing the system to avoid exceeding the

servos' maximum speed measured in change in pulse length per pulse. To do this we use (1). We do not learn this smoothness in the cyclic genetic algorithm because it is trivial to implement an interpolation function in the simulation and the final controller. The design goal for our chromosome is for it to learn what position the servos should move to next and how quickly that movement should be done, not the smoothness of each movement.

$$\text{pulse increment} = \min[(\text{horizontal pulse} - \text{previous horizontal pulse})/\text{repetitions}, \text{max speed}] \quad (1)$$

We do not need smoothness on the vertical servo since it does not affect the robot's movement and so the vertical servo will always move at its maximum speed. Note also that we use the term "repetitions" in our chromosome to refer to the number of steps one move should take to complete. A more accurate term for the horizontal and extension/contraction movements would be "interpolation steps", but we choose to use repetitions because all the previous papers involving cyclic genetic algorithms used repetitions to refer to the steps and we wish to keep the this terminology consistent.

We based our chromosome, shown in Fig. 5, on the design used in previous work [6]. Each gene is structured the same and is made up of repetitions and pulses. However, we made three key differences. We have added a third pulse, the extension pulse, to each gene in order to control the third extension/contraction degree of freedom per leg. In addition, we have increased the length of the repetitions part to range 0-31 represented by five bits to allow more movement to be achieved in one instruction due to the increased size of the robot and therefore the increased range of its motion. Finally, we chose to use a total of four genes instead of the original eight because it was found that half of the genes would learn to set their repetitions to 0 and be unused. In other words, four genes were sufficient to represent the instructions to the servos to produce a good leg cycle. These decisions on the max number of repetitions, the number of genes, and the pulse signal increments of 10 represent the extent of the *a priori* knowledge we used. Since each gene would individually be longer, we decided to cut down on the number of genes to keep chromosome length roughly the same without compromising on the freedom of the learning algorithm to learn. The final chromosome ends up being 104 bits long.

The final chromosomes resulting from the training can be uploaded directly to their respective leg's stamp and when executed would execute the instructions sent to the three servos. A set of pulses made by an example chromosome is shown in Fig. 6.

[[R1,HP1,EP1,VP1],[R2,HP2,EP2,VP2],[R3,HP3,EP3,VP3],[R4,HP4,EP4,VP4]]

Figure 5. Leg cycle chromosome. Each of the four genes is made of three parts: repetitions (R), horizontal pulse (HP), extension pulse (EP) and vertical pulse (VP).

Genes	Horizontal Pulse	Extension Pulse	Vertical Pulse
[6, 125, 20, 0]	40	55	0
	57	48	0
	74	41	0
	91	34	0
	108	27	0
	125	**20**	**0**
[3, 68, 62, 100]	106	34	100
	87	48	100
	68	**62**	**100**
[3, 23, 62, 42]	53	62	42
	38	62	42
	23	**62**	**42**

Figure 6. A sequence of pulses generated by looping three example genes. In bold are the pulses dictated by the genes with all the other pulses being in-between pulses in even steps. The exception is for the vertical pulses since the vertical servo will always move at maximum speed. Due to the cyclic nature of the chromosome, [40, 55, 0] is not the starting position of the servos but the next step after [23, 62, 42]. The servos could start in any position and eventually converge to this pulse set after enough loops.

A. Leg Model

Each leg's capabilities were measured and the data was stored in a simple model of the leg that held the leg's current position and its previous position along with the capability data. Each position consisted of a horizontal, extension and vertical component. Each horizontal component was defined as 0 when the leg was fully forwards and its maximum throw was fully backwards. The extension component was defined as 0 when the leg was fully inwards and closest to the body and its maximum throw was fully extended. The vertical component was defined as 0 when the leg was fully down and its maximum throw was when it was fully up. The ranges of all the maximum throws were measured and stored in the model.

We also had to account for the fact that a vertical position of 1 was still touching the ground and in fact, a higher vertical position was required before it stopped dragging. This was measured and stored as a "ground level" constant that the vertical component had to exceed before the leg was considered to be off of the ground.

In addition, we had to account for the fact that the movement per pulse of the leg would decrease as it approached the extremes of its throw. Previous research handled this with a lookup table, however, the extra degrees of freedom and the extra two legs made this method impractical as the number of measurements we would have to take would be too time-consuming when there are quicker and comparable methods to use.

Instead, we analyzed the general movement of the servos and noted that they had fairly consistent performance for the middle chunk of their throw. We used this information to focus our measurements on the extremes of the throw and analyzed how the distance moved per pulse sharply decreased. We then took measurements and generalized them using a curve fitted to these measurements which the model can use to determine how much it can move at the extremes of its throw. This provided a sufficient level of accuracy for our model.

B. Training

We used a population of 128 chromosomes for each leg with each chromosome representing a single leg cycle. Our goal was to produce eight different populations each generating a leg cycle for a single leg. We trained each population for 1,000 generations on their respective leg models.

We used a different fitness function from past research to account for some of the changes we made to the chromosomes and to improve the final gait based on the findings found in past research. Our final fitness function took into account the effective thrust of each leg, which depends on the forward movement of the leg, how straight the leg moves and the vertical position of the leg. Forward movement was calculated by taking the distance the horizontal degree moved while it was on the ground, whether it was positive or negative. We did this despite there being a third degree of freedom contributing to the movement because the horizontal degree still provides most of the thrust while the extension degree mostly works to ensure the leg remains straight throughout its throw.

In addition to forward movement, our fitness function included straightness and height. Straightness was calculated by taking the exact position of the leg before and after moving along the ground and taking the absolute value of how parallel the movement was to the direction of travel. Lower straightness values were better, so we subtracted them from the fitness function. Height was just retrieved from the leg model's position as the vertical extension above the ground. We added this because our model now accounts for the fact that the leg is actually touching the ground over a range of values from 0 to our measured "ground level". We can use this information to represent the fact that the lower the leg is extended downwards the more weight is put on that leg and therefore more friction is produced by that leg which results in a more effective horizontal movement. Inversely, negative movement is also less detrimental when the leg is higher up and only barely touching the ground.

To represent this inverse relationship between the grip and the vertical extension while it's below the "ground level" we simply divide the effective thrust by the current up position (plus 1 to avoid division by 0). It is also important to note that no fitness is added while the leg is in the air since forward movement and straightness are both 0 when the leg is not touching the ground. This culminated in the following formula for calculating the effective thrust:

$$\text{thrust} = (\text{forward movement} - \text{straightness})/(\text{up position}+1) \quad (2)$$

The fitness for each chromosome was calculated once per gene as the chromosome was looped through 100 times to ensure a cyclic behavior was learned. The best chromosomes would learn to reset their positions at the end of the cycle so that the next loop would be ready to start again. The fitness was then used to stochastically select chromosomes to produce each new generation. We used uniform crossover at a 100% crossover rate, and we used two types of mutation - inter-gene and intra-gene mutation. Inter-gene mutation resets an entire gene to a random gene while intra-gene mutation only flips a single bit at a time. After the selection, crossover, and mutation were performed, the population would be subjected to a cleanup function which would take any genes with zero repetitions and move them to the end of the chromosome. This was to ensure that any learning that resulted in not using all the genes would not be lost immediately during crossover.

C. Results

Training was done over 200 generations with the fittest individuals saved every 5 generations. The results are shown in Fig. 7. Each solid line represents a leg. All legs learned quickly and settled at an optimal fitness matching their opposite leg i.e. legs L1 and R1 had similar fitnesses and legs L2 and R2 had similar fitnesses. We found the total optimal lengths of the leg cycle to be in the range of 19-22 repetitions. We then repeated the training to prepare for the second increment of learning described in the next section. We added a new component to the fitness function called "desired length". Desired length was calculated as the absolute value of the total repetitions in the chromosome subtracted from an inputted desired number of repetitions. The lower the number the higher the fitness so desired length was subtracted from the total fitness. The length used was 22 repetitions since that was the maximum optimal length found after the initial training. We chose the maximum optimal length because a longer leg cycle would be more useful in further experimentation.

The results of the new training are shown in Fig. 8. Since an optimal cycle length was specified, all the legs learned rapidly at the beginning of the training and most of them settled quickly on a near-optimal solution. We tested the results on the real robot and we observed that at 22 repetitions all legs produced efficient and straight leg cycles with plenty of clearance above ground level. However, at this point, we could not measure how far these legs could walk before we coordinated the movement together into a gait.

Figure 7. Training results for all eight legs learning their own individual leg cycle.

Figure 8. Single leg training with a desired length of 22 repetitions specified. Although the legs do not produce more forward thrust than those shown in Fig. 7, the fitness values are much higher due to the incorporation of the "desired length" into the fitness function

VI. Second Increment: Evolving Gaits from the Leg Cycle

The key to incremental learning is the observation that any uniform gait for a legged robot is just the coordination of each leg's own leg cycles into one combined walking cycle. We use a collection of eight different groups of walking cycles (one per leg) to generate an optimal walking gait for our robot.

A. Evolving Fixed Length Leg Cycles

In the previous section we learned walking cycle chromosomes for all eight legs in 200 generations with a desired length of 22 repetitions. We then saved the populations generated during this training and used them to learn leg cycles in the range of 15 to 30 repetitions inclusive. We achieved this by setting the starting population to be the 22-repetition population instead of a random one and did 200 generations of training on it with a desired length of 21. We then repeated this process with the 21-repetition population to learn a 20-repetition chromosome, then a 19-repetition one and so forth until we had eight chromosomes ranging from 15 to 22 repetitions inclusive. This was then repeated for the 22 to 30 range starting with the 22-repetition population. The 16 optimal chromosomes for each leg were then stored for use in gait training described in section 6C.

B. Robot Model

The training in this section is for the central controller that coordinates the eight leg controllers using timing signals. To achieve this, it needs to decide the total gait cycle length, which leg cycles to use for each leg and when to start and restart each leg cycle. The central stamp ensures that all the pulses are sent simultaneously so that the leg cycles stay synchronized.

Unlike with the leg cycles, the gait coordination was learned using a standard GA because the cyclic behavior had already been learned. All we must learn now is which leg cycle length to use for each leg and at what time should each cycle be started and stopped. The chromosome used had nine genes and is shown in Fig. 9.

The first gene, the gait cycle length (GCL), determined the total length of each leg cycle so that all leg cycles would loop at the same rate. The other eight genes each represented a different leg and were split into two parts - the leg cycle length (LCL) and the start time (START). The LCL would determine which of the 16-leg cycles in the 15-30 repetition range to use for each leg. If the LCL was longer than the GCL it would be truncated and if it was shorter it would be extended so that each leg cycle would match the GCL in length. The start time would determine where within the length of the GCL a particular leg would start and restart its cycle. If the start time was greater than the GCL, the leg would never start moving. The start time was the main factor in coordinating the leg. When the controller program begins, it counts the repetitions until the start time is reached, upon which the corresponding leg cycle would begin. And when the number of repetitions reached the GCL, the count would start again at zero.

In addition to knowing the timing of each leg cycle the model also had to know the position of each leg relative to the center of the robot's body at any given time so as to determine the balance of the robot at each pulse which is discussed further in section 6C. To do this we had to measure how far the hinge for each leg was from the center of the body and incorporate this into the position of each foot by using the leg model data of the actual length of the leg and its current position relative to its hinge.

C. Training

We trained a population of 128 chromosomes over 1000 Generations. Each individual's fitness was calculated with the same fitness function as used in the individual leg training with the desired length removed and with each of the eight legs moving simultaneously in their respective positions. We also added balance and drag into the fitness calculation.

Balance was calculated after all the legs had finished moving by one pulse by taking the area created by all the legs touching the ground. If the legs on the ground form a polygon with a large area that covers the center of gravity of the robot then it can be considered to be statically stable, with bigger areas being more stable.

Dynamic pairs of legs (only two legs on the ground) forming a straight line through the center of gravity would measure as a 0 with this method of calculating balance. However, a dynamically stable pair of legs is better than any unstable group of legs. To represent this fact, we measured

[GCL,[LCLR1,StartR1],[LCLL1,StartL1],…,[LCLR4, StartR4],[LCLL4, StartL4]]

Figure 9. Chromosomes used for learning to coordinate the individual leg cycles into a gait. It is made up of a Gait Cycle Length (GCL) and eight pairs of Leg Cycle Lengths (LCL) and Start times (Start) - one pair for each leg.

the smallest possible area for a stable group of legs that formed a polygon and found it to be around 0.6. We judged that a dynamically stable pair of legs would be slightly worse

than this and so our balance function returns a value of 0.5 for any such pair.

With each pulse, the drag would be calculated per leg by checking if the leg was on the ground but not producing thrust. This was to penalize behaviors which would set the start time for certain legs to be higher than the gait cycle length which would cause them to remain motionless and provide constant stability despite being dragged along the ground by the other legs.

Finally, we would only start calculating fitness after 31 pulses, the maximum value of the gait cycle length, had passed so as not to unfairly favor gaits that had all the legs start sooner in the cycle rather than wait to get a properly timed gait. Using this fitness function individuals were chosen stochastically for crossover with the addition of elitism which would ensure the fittest individual in a population would be cloned into the new population. Crossover and mutation were both done the same as described in section 5B.

D. Results

Five independent tests were run starting with populations of random chromosomes. The best individuals of each generation were stored and the results are shown in Fig. 10. The graph shows the fitness growth of the five different populations. The cause of the discrete steps and flat lines is the use of elitism during training. In addition to this, the starting fitnesses of all the populations are well above 0. This is because all the individual leg cycles used are already near-optimal and so no matter what combination and timing of the legs is used there will always be forward movement. Most populations learn adequate gaits by the 300-generation mark with only small improvements being made after that. In all cases near-optimal alternating tetrapod gaits are generated, similar to the gaits of biological spiders.

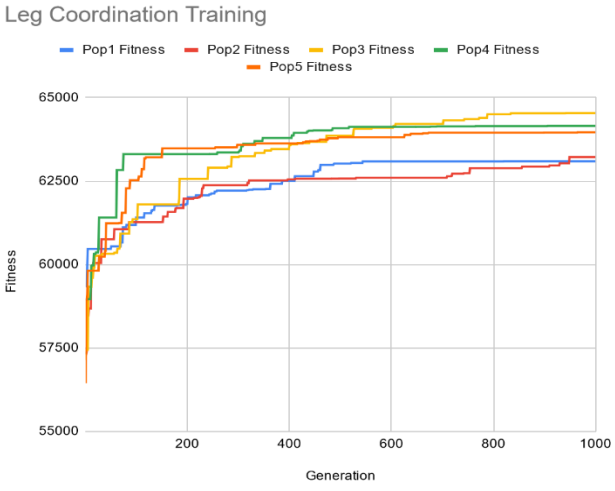

Figure 10. GA gait training coordinating the leg cycles used into a single gait.

Every best chromosome from population 1 was uploaded to the real robot for testing to confirm the viability of the learned gaits. The results of this testing are shown in Fig. 11 along with the training results for population 1 in Fig. 12 for comparison. The gait observed was the alternating tetrapod gait found in biological spiders. The distance moved plot (Fig. 11) also closely matched the slope of the population 1 fitness graph (Fig. 12) which shows that the same rate of learning in simulation translates to an equivalent rate of progress on the real robot. The legs move vertically to their maximum throws to mitigate drag when the legs move upwards and increase grip when the leg is on the ground. The contraction servo moves accurately to ensure the movement of each leg stays straight and the robot doesn't rotate as it walks. The gait is also fast as it switches between each stride step quickly and never lingers. We also uploaded the final chromosomes of the other four populations to the robot. All populations produced similar near-optimal results with the same alternating tetrapod gait pattern.

VII. Conclusions

With moderately simple fitness functions and minimal *a priori* knowledge, incremental evolution in two increments provides a simple yet effective means of evolving gaits for legged robots with three degrees of freedom per leg. CGAs proved reliable in the first increment in generating near-optimal cyclic behavior for individual legs through learning a sequence of pulses for a three-servo leg. It was shown in the simulation that the chromosomes produced by the CGA improved significantly during training and produced

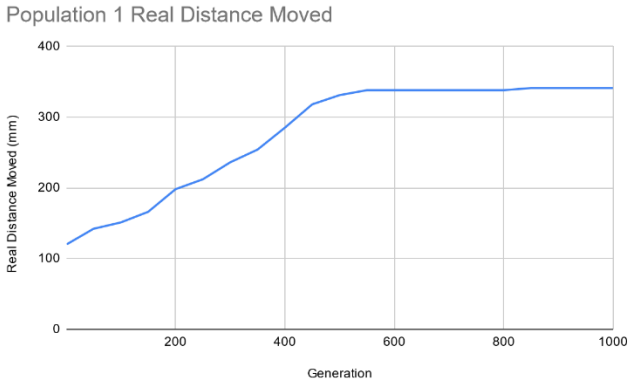

Figure 11. Performance of the real robot using chromosomes sampled over the 1000 generations of training from population 1. Distance moved is measued as how far the robot moves in a straight line for 500 repetitions in millimeters.

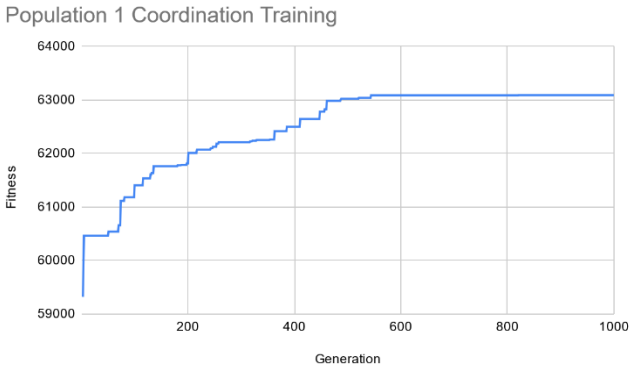

Figure 12. GA gait training (in simulation) coordinating the leg cycles to learn a gait, highlighting population 1.

individual leg cycles which were observed to be viable when uploaded to the real robot. In the second increment, a GA can be used to coordinate the movements of the eight legs together into a near-optimal gait which is also observed in real-life spiders. Tests in the simulation and the actual robot confirm the viability of this method.

REFERENCES

[1] J. Hwangbo *et al.,* "Learning agile and dynamic motor skills for legged robots,=" Sci. Robot. 4, 2019. DOI: 10.1126/scirobotics.aau5872

[2] T. Haarnoja *et al.,* "Learning to Walk via Deep Reinforcement Learning," CoRR, 2018, abs/1812.11103. https://arxiv.org/abs/1812.11103.

[3] M. Ajallooeian, S. Pouya, A. Sproewitz and A. J. Ijspeert, "Central Pattern Generators augmented with virtual model control for quadruped rough terrain locomotion," 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 3321-3328, doi: 10.1109/ICRA.2013.6631040.

[4] M. Shafiee, G. Bellegarda and A. Ijspeert, "Puppeteer and Marionette: Learning Anticipatory Quadrupedal Locomotion Based on Interactions of a Central Pattern Generator and Supraspinal Drive," 2023 IEEE International Conference on Robotics and Automation, 2023, pp. 1112-1119, doi: 10.1109/ICRA48891.2023.10160706.

[5] G. B. Parker, "Evolving Leg Cycles to Produce Hexapod Gaits," Proceedings of the World Automation Congress (WAC2000), Volume 10, Robotic and Manufacturing Systems, June 2000, pp. 250-255.

[6] G. B. Parker, D. W. Braun, and I. Cyliax, "Evolving Hexapod Gaits Using a Cyclic Genetic Algorithm," IASTED International Conference on Artificial Intelligence and Soft Computing, 1997, pp. 141-144.

[7] G. B. Parker, "Generating Arachnid Robot Gaits with Cyclic Genetic Algorithms," Genetic Programming, 1998, pp. 576-583.

[8] G. B. Parker, "The Incremental Evolution of Gaits for Hexapod Robots," Genetic and Evolutionary Computation Conference, 2001, pp. 1114-1121.

[9] Z. Y. Wang, J. T. Wang, A. H. Ji, H. K. Li & Z. D. Dai, "Movement behavior of a spider on a horizontal surface," Chin. Sci. Bull. 56, 2011, pp. 2748–2757. https://doi.org/10.1007/s11434-011-4584-y