Stochastic Training of a Biologically Plausible Spino-neuromuscular System Model

Stanley Gotshall
Department of Computer Science
University of Idaho
Moscow, ID
stan@ecpost.org

Terence Soule
Department of Computer Science
University of Idaho
Moscow, ID
tsoule@cs.uidaho.edu

ABSTRACT

A primary goal of evolutionary robotics is to create systems that are as robust and adaptive as the human body. Moving toward this goal often involves training control systems that process sensory information in a way similar to humans. Artificial neural networks have been an increasingly popular option for this because they consist of processing units that approximate the synaptic activity of biological signal processing units, i.e. neurons. In this paper we train a nonlinear recurrent spino-neuromuscular system (SNMS) model and compare the performance of genetic algorithms (GA)s, particle swarm optimizers (PSO)s, and GA/PSO hybrids. Several key features of the SNMS model have previously been modeled individually but have not been combined into a single model as is done here. The results show that each algorithm produces fit solutions and generates fundamental biological behaviors, such as tonic tension behaviors and triceps activation patterns, that are not explicitly trained.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms

Keywords

Spinal Cord, Neural Networks, Spiking Networks, Genetic Algorithms, Particle Swarm Optimizers, Breeding Swarm Optimizers

1. INTRODUCTION

In biology, neural networks in the spinal cord provide the body with a remarkably precise and adaptive system for control. Motor neurons and muscle fibers exchange electrical signals to produce controlled anatomical movements throughout the body. Thus, designing a biologically plausible neural controller can potentially allow researchers to harness the properties of biological systems that are uniquely suited to control complex mechanical systems.

The nervous system has an unmatched ability to control highly nonlinear systems, i.e. skeletal muscles. Muscles consist of thousands of muscle fibers that respond to electrochemical signals from the spinal cord to move the body [12]. In turn, feedback signals from muscle fibers allow the spinal cord to adapt and activate muscle groups in order to generate smooth and precise motions. By modeling control systems after the spinal cord, we increase our understanding of the role and importance of certain neural pathways in the nervous system. It also opens the door to developing prosthetic devices that interpret brain signals the same way the spinal cord does and to design robots with similar possessing abilities.

An overarching goal of evolutionary robotics is to create autonomous agents that are as robust and precise as the human body. This typically involves training systems to process information received from visual, auditory, and tactile sensors. Research in evolutionary robotics focuses on goals such as training agents to generate controlled motions and finding optimal paths through mazes. However, these methods typically do not use biologically realistic architectures to process sensory signals. Although these methods often use spiking neuron models that mimic the computational power of a single nerve cell, the network architecture does not propagate and process the information in a biological way. Thus, a key step in moving toward biologically realistic information processing is to use a biologically plausible network architecture to control robotic systems.

This paper applies three stochastic/evolutionary algorithms: 1) genetic algorithms (GA)s, 2) particle swarm optimizers (PSO)s, and 3) breeding swarm optimizers (BSO)s, which is a GA/PSO hybrid, to train anatomical motion in a biologically based spino-neuromuscular system (SNMS) model. Neural networks in the spinal cord simultaneously integrate signals from the brain and muscle sensors to determine the amount of stimulation that the muscle fibers require to accomplish a motion. Our model mimics this integration by simulating descending brain signals and muscle feedback pathways which encode the contractile velocity and length of muscle fibers.

Our results show these algorithms, with varying degrees of success, effectively train the 700+ real-value and binary parameters in the model to generate controlled anatomical movements. Furthermore, comparisons with data from human subjects show that the training process produces key biological control mechanisms that are not explicitly trained.

2. BACKGROUND AND PREVIOUS WORK

Artificial neural networks use individual processing units, neurons, to mimic the processing power of the nervous system. Traditional connectionist neural networks consist of neurons that compute static functions of their inputs [6, 16, 24] which, in the biological sense, represents an activity level (i.e. frequency) for that neuron. Networks consisting of these neurons are effective for many applications such as classification, prediction, and control systems. However, as our understanding of the nervous system increases, more biologically realistic neuron models have emerged, i.e. spiking neurons. Spiking neurons more accurately model the temporal dynamics of biological neurons. Thus, networks of spiking neurons can more accurately approximate the signal processing abilities of neural networks in the brain or spinal cord.

Although spiking neural networks are uniquely suited for biological applications, researchers also use them for various types of problems commonly solved with traditional neural networks [9, 18]. Furthermore, spiking neurons are computationally more powerful than traditional neurons [15]. Thus, practitioners can construct small networks of spiking neurons to accomplish the same task as a traditional network of greater size. This is done in the hope that the smaller number of synaptic weights will ease the training process.

However, training spiking neural networks requires innovative techniques. Traditional feed-forward neural networks are trainable via back-propagation or Hebbian learning. Unfortunately, these training techniques cannot accommodate the temporal characteristics of spiking neural networks. Fortunately, researchers are developing a new generation of neural network training techniques. Instead of training numeric input-output patterns, these methods use mathematical methods to train temporal spiking patterns [13, 19]. These techniques are useful in pattern recognition and control system problems, but only when the objective is to train a known input/output spiking pattern. However, in some problems the patterns needed to solve the problem are not known. This is often the case when the task of a neural network is to control a mechanical system because the goal is typically quantified in terms of a desired motion and not a particular network output pattern. These scenarios need a new training approach that does not depend on practitioners knowing the input/output patterns a priori.

Thus, our goal is to use stochastic algorithms to train a spiking neural network, and other parameters in the system, to control a biologically plausible neuromuscular model of an arm. This approach lets the user specify the quality of potential solutions in terms of error from a target motion. This way the practitioner does not need to know the specifics of the network's output patterns required to produce the desired motions.

3. MODELS

Each of the following sub-sections describe key components of the SNMS model. The arm consists of two groups of muscle fibers corresponding to the biceps and triceps. Each muscle fiber is controlled by semi-independent neural subnets. Information about the state of the muscle fibers is fed back into its muscle's subnets to allow for adaptive behaviors in the system. The timestep for all model components is 0.01 seconds.

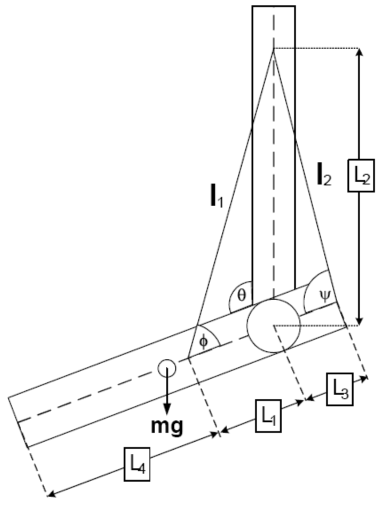

Figure 1: The joint has a movable lower limb with center of mass indicated by mg. The parameters of the joint model are listed in Table 1.

3.1 Joint Model

The joint model used in our experiments (Figure 1) has a single frictionless degree of freedom. The motion of the forearm is dependant on contractions by either muscle and the force of gravity. Furthermore, rotational springs are encountered at π-2.4 and π-0.4 radians to model resistance to over-contraction and hyperextension, respectively, in the human arm. The connections of the muscles are shown as I_1 and I_2 in Figure 1 and the movable section of the limb is modeled as a uniform cylinder. The parameters of the joint are shown in Table 1.

Table 1: Joint model parameters

Movable Limb Radius	0.2 meters
Movable Limb Mass	0.6 kg
Timestep	0.01 seconds
L1	0.08 meters
L2	0.8 meters
L3	0.08 meters
L4	0.72 meters
Spring Constant	20
Spring Damping Constant	10

3.2 Muscle Model

The Hill model [2, 8, 11] is commonly used to simulate skeletal muscle systems and their nonlinearities. We use the second canonical, functionally equivalent, form of the Hill model because this form is easiest to apply to multiple muscle fibers acting in parallel with one another [17]. Our implementation of each muscle has six independently triggerable muscle fibers connected to a common tendon. Each muscle fiber receives a binary signal from its alpha-motoneuron (α-MN) and contracts according to the Hill equations. A muscle fiber contracts during a given timestep if its α-MN is active and relaxes otherwise. The tendon component of the model stretches and shortens depending on the activity of each muscle fiber. The muscles connect to the upper portion of the arm and to the forearm near the joint as indicated in Figure 1.

The Hill model is organized into four mechanical components: the 1) contractile element, 2) damping element, 3) serial elastic components (SEC)s, and the 4) parallel elastic component (PEC). This form of the Hill model depicts the muscle fibers as the "serial" elastic component and the tendon as the "parallel" elastic component. This model also generalizes to represent many motor units where a motor unit is comprised of a single motoneuron and all the muscle fibers it innervates[1]. The SEC models the elastic properties of muscle fibers in a motor unit while the PEC models the elasticity of a single tendon. The contractile element models the contraction and relaxation of muscle fibers and the damping element models the coupling between the rate of muscle contraction and fiber tension.

3.3 Neuron Model: Integrate-and-Fire

Each neuron in the network is an integrate-and-fire (IF) spiking neuron with time constant τ_0=0.05 seconds [4, 10, 21]. Over time an IF neuron receives input signals from presynaptic neurons which increase the neuron's potential. In the absence of input, the potential decays exponentially. However, if a volley of input signals causes the neuron's potential to exceed its threshold, the neuron fires and sends a signal to all its postsynaptic neurons. After a neuron fires it enters a refractory period of a single timestep during which it cannot build up any potential. This is commonly referred to as the absolute refractory period [12]. With this refractory period, each neuron can have a maximum frequency of 50 Hz which is within the range of biological firing rates for regular spiking neurons [22]. After the refractory period, the neuron can build potential and fire again. The IF neuron is modeled with

$$s(t+\Delta t) = \begin{cases} (1-\frac{\Delta t}{\tau_0})s(t) + \frac{\Delta t}{\tau_0}\sum_n \omega_n x_n & , s(t) < 1 \\ 0 & , s(t) \geq 1 \end{cases} \quad (1)$$

and

$$y(t) = \begin{cases} 1 & , s(t) \geq 1 \\ 0 & , s(t) < 1 \end{cases} \quad (2)$$

where the functions $s(t)$ and $y(t)$ are the neuron's electric potential and output, respectively, at time t. The summation in Equation 1 represents the sum of synaptic inputs multiplied by their respective weights.

3.4 Neural Network

Our neural network model is designed to incorporate what are believed to be the key control pathways in the SNMS, including the group Ia and group II afferent pathways, the Ia inhibitory interneurons, and Renshaw cell inhibition (Figure 2). The system's neural network consists of 180 IF neurons with 700+ synaptic connections and is composed of twelve subnets, one for each of the twelve muscle fibers in the biceps and triceps. The subnets have identical topologies and receive independent descending inputs (Figure 2). The individual subnets contain chains of neuron clusters which model individual descending pathways to the spinal neurons. The agonist muscle refers to the muscle fiber innervated by the α-MN and antagonist refers to fibers in the opposing muscle. The input to a given subnet is implemented as three synchronous unipolar inputs.

[1]In this paper, each group consisting of one contractile element, one damping element, and one serial elastic component will be called a motor unit where an entire muscle consists of six motor units. Biologically, a motor unit also includes the innervating α-MN.

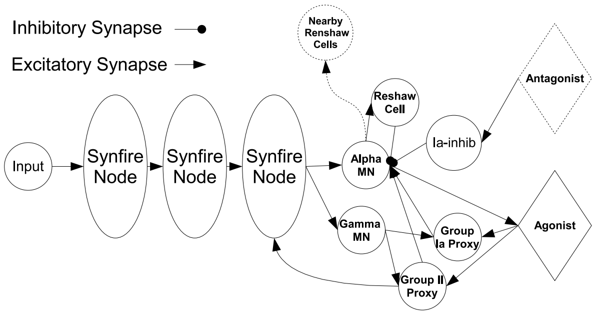

Figure 2: This figure shows a single neural subnet with descending neural pathways to the spinal neurons for one agonist motor unit. The group Ia and group II afferent structures are mathematical proxies for the overall effect of extrafusal and intrafusal fibers on the spindle nerve endings [12]. This subnet is repeated five times for the remaining agonist motor units and six times for the six antagonist motor units.

In biology, it is difficult for a simple chain of neurons to propagate a signal because a relatively high input frequency is needed for each neuron in order to fire. Even with a high input frequency to a simple chain, the firing frequency of each successive neuron decreases significantly unless the synaptic weights are arbitrarily large. It is more likely that these pathways are organized as groups of neurons where information is passed from one group to the next. This architecture is known as a synfire chain. Each descending node in Figure 2 is implemented as a synfire node [1, 23, 25], where each synfire node is a collection of three IF neurons that are fully connected to the preceding node and following node. The use of synfire nodes allows signals to propagate without this attenuation and also allows for complex variations in descending signal patterns.

After neuronal signals propagate from the brain and down the spinal cord, they synapse on the α-MN which activates skeletal muscle fibers. When activated, the α-MN sends a signal to muscle fibers causing them to contract and shorten which moves the associated joint. Furthermore, like many artificial nonlinear control systems, the spinal cord has an inhibitory feedback mechanism, the Renshaw cells. These cells are thought to modulate and control the α-MN's rate of discharge to produce controlled motions [3]. The connections of the α-MN and Renshaw cell are shown in Figure 2. Each Renshaw cell has inhibitory synaptic links to all six α-MNs in its muscle's network.

The spinal cord also has feedback systems to receive state information from muscle fibers. The most prominent types of feedback in skeletal muscles are the primary (group Ia) and secondary (group II) afferents which relay the contractile velocity and length of muscle fibers, respectively, back to the spinal cord. In the model, the Ia proxy afferent neuron receives two real valued inputs which reflect the average change in length per timestep (i.e. velocity) of the six muscle fibers during the current timestep. One velocity input is active when the average muscle fiber length is decreasing and the other is active when it is increasing. These inputs are separated to reflect the high and low frequencies of action potentials associated with muscle fiber contraction and relaxation. The model also includes a group II proxy afferent neuron in each subnet which receives the average muscle fiber length at each timestep. This neuron sends its output only to the local α-MN and the last descending synfire node (Figure 2).

Alpha-MN activation causes direct contraction of normal muscle fibers, also called extrafusal fibers. The gamma-motoneuron (γ-MN), on the other hand, causes contraction of modified muscle fibers called intrafusal fibers. Contraction of these fibers do not contribute to the contractile force of the muscle but increases their sensitivity to changes in length as the entire muscle's length changes. In turn, the intrafusal fibers serve as the input to the afferent feedback neurons mentioned previously. The model also includes the γ-MN which receives descending signals in parallel with the α-MN [12]. This models the phenomena known as $\alpha - \gamma$ (alpha-gamma) coactivation. We simulate the sensitivity of excited intrafusal fibers by adding a synapse from the γ-MN to the proxy afferent neurons (Figure 2). Furthermore, the inhibitory connection from the α-MN to the afferent neuron is a mathematical proxy for the muscle shortening that occurs just after the α-MN fires. After a contraction, the tension on the muscle fiber decreases momentarily lowering the afferent neuron's potential.

Training the network requires adjusting 12 real values used to determine the strength of each contractile element, 12 binary values to determine which biceps motor units receive descending inputs during the upward (6) and downward (6) motion, and 700+ real values for each synaptic connection. We use the GA (Genetic Algorithm), PSO (Particle Swarm Optimizer), and BSO (Breeding Swarm Optimizer) to train the system because adjusting this number of parameters by hand would clearly be prohibitive.

4. EXPERIMENTS

The following algorithms train three distinct parameter types in the model: The 1)synaptic weights, 2)strengths of muscle fibers, and 3)binary selectors for neural subnets. All individuals in each algorithm are randomly initialized in the same manner where synaptic weights are randomly selected uniformly in the interval [0,4] (excitatory) and [-4,0] (inhibitory), fiber strengths in the interval [0,6], and subnet selectors in the uniform binary interval [0,1]. The velocity vectors in the PSO and BSO initialize with the intervals as their respective position vectors both in the positive and negative direction with the exception of the binary values which initialize their velocity vectors in the interval [-6,6].

Each algorithm evaluates fitness of potential solutions with

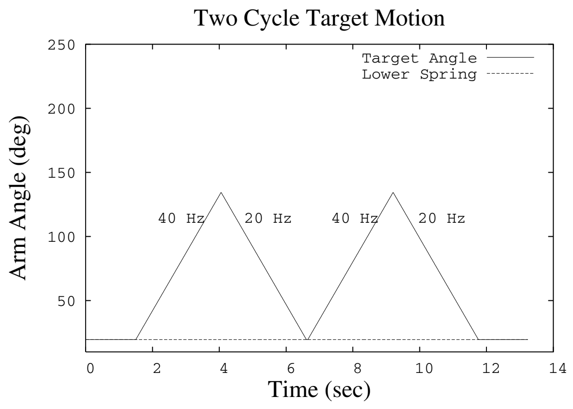

Figure 3: Two cycle target motion. Starting at 0 seconds, no descending pathways receive an input pattern. At 1.5 seconds, an input pattern of 40 Hz is sent to a subset the biceps' subnets, as chosen by the training algorithm, and the signals across each input neuron is phase shifted evenly across a window of $\frac{1}{40Hz}$. At a given frequency, the inputs to the subnets are out of phase and do not overlap during a given timestep unless the frequency is sufficiently high. During the downward motion the biceps network receives a descending input of 20 Hz. The upward and downward target angular velocities are 45 $\frac{deg}{sec}$ in both directions.

the equation

$$F = -\sum_{all\ t} (\Theta(t) - target(t))^2 \quad (3)$$

where $\Theta(t)$ is the angle between the movable forearm and the fixed upper arm at time t, and $target(t)$ is the target angle at time t. It is important to note that the fitness equation does not attempt to minimize overall muscle activity or encourage specific neural behaviors. The fitness function measures the error from the target trace in Figure 3. The following subsections describe each algorithm in more detail.

4.1 Genetic Algorithm Training

These experiments use the two standard models of GAs, steady-state and generational. The generational GA creates a new population every generation and automatically preserves the two most fit individuals from the previous generation (elitism). The steady-state version on the other hand does not generate a new population, but replaces the two least fit individuals with two offspring during each iteration. For the real-value segments of the individuals, recombination is performed via arithmetic crossover [7] added to a small normal random variable as shown in Table 2. Uniform crossover is used for the binary component. In both algorithms, after an individual is generated via crossover, each allele in the offspring then undergoes mutation with probability p, shown in Table 2.

4.2 Particle Swarm Training

The PSO is a relatively new stochastic search and optimization technique based on swarm intelligence developed by Eberhart and Kennedy in 1995 [14]. The following ex-

Table 2: GA, PSO, and BSO parameters

Parameter	GA	PSO	BSO
Trials	35	35	35
Fitness Evaluations	150k	150k	150k
Population Size	30	30	30
Selection Type	Tournament	N/A	Tournament
Tournament Size	2	N/A	2
Crossover Probability	1.0	N/A	N/A
Crossover Type (reals)	Arithmetic + N(0,0.2) [7]	N/A	VPAC [20]
Crossover Type (bits)	Uniform	N/A	Uniform
Mutation Probability	1/dimensions	N/A	1/dimensions
Mutation Type (reals)	N(0,0.2)	N/A	N(0,1→0)
Mutation Type (bits)	Uniform	N/A	Uniform
Mutation Variance (reals)	N/A	N/A	1.0 → 0.0
Social (c_1, c_2)	N/A	2,2	2,2
Inertia	N/A	0.9→ 0.3	0.9→0.3

periments employ the two main PSO types, inertia and constriction. The PSO is modeled with a population of particles where each particle knows its position and velocity in n-dimensional space. Each particle remembers the best location it has seen and also has access to the best location seen by the entire swarm. The maximum and minimum values in the GA are used as constraints for the particles' velocities but not their positions. PSO-style algorithms usually do not need artificial constraints on position values unless the problem itself requires certain constraints. The inertia PSO [20] particle velocity at timestep t is

$$v(t) = \Omega(t-1)v(t-1)+(c_1)(r_1)(x_b(t-1))+(c_2)(r_2)(x_{gb}(t-1)) \tag{4}$$

where c_1 and c_2 are social constants shown in Table 2, the vector x_b is the particle's personal best position, x_{gb} is the global best position, Ω is the population's current inertia value, and r_1 and r_2 are independent uniform random variables in the interval [0,1]. The constriction PSO [5] velocity update equation is

$$v(t) = \tau[v(t-1) + (c_1)(r_1)(x_b(t-1)) + (c_2)(r_2)(x_{gb}(t-1))] \tag{5}$$

where τ is the constriction coefficient derived from the equation

$$\tau = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{6}$$

where

$$\phi = \begin{cases} c_1 + c_2 & , c_1 + c_2 > 4 \\ 4.1 & , \text{otherwise.} \end{cases} \tag{7}$$

In these experiments $c_1 + c_2$=4, thus ϕ=4.1 and $\tau \approx 0.73$. For real numbers, the position vector for both PSO types is then updated as

$$p(t) = p(t-1) + v(t) \tag{8}$$

and for the binary numbers it is updated with

$$p(t) = \begin{cases} 1 & , rand < s(v(t-1)) \\ 0 & , otherwise \end{cases} \tag{9}$$

where $rand$ is a uniform random variable in the interval [0,1] and $s(v)$ is the sigmoid function $s(v) = 1/(1 + e^{-v})$ where v is the velocity vector [14].

4.3 Breeding Swarm Training

The BSO algorithm is identical to the PSO with added steps of crossover and mutation. Crossover is implemented with Settles' VPAC (Velocity Propelled Averaged Crossover) [20]. VPAC is defined as

$$x_{c1} = (x_{p1} + x_{p2})/2 - (\phi_1)(v_{p1}) \tag{10}$$

$$x_{c2} = (x_{p1} + x_{p2})/2 - (\phi_2)(v_{p2}) \tag{11}$$

where ϕ_1 and ϕ_2 are independent uniform random variables on the interval [0,1]. Both offsprings' x_b (personal best) vector is reset, $v_{c1} = v_{p1}$, and $v_{c2} = v_{p2}$ after crossover. Each allele is then mutated with a linearly decaying variance according to Table 2.

To determine the number of particles to undergo crossover, the algorithm uses a breeding ratio r. Typically, a small fixed breeding ratio is chosen and $popsize \cdot r$ particles are generated via VPAC crossover and mutation. The offspring particles then replace the least fit $popsize \cdot r$ particles in the population. VPAC crossover generates offspring in pairs, so after using the breeding ratio to determine the number of offspring we round down to the nearest multiple of 2. In this implementation with a population size of 30 and r=0.1, 2 offspring are created via crossover and mutation and then inserted at each generation.

4.4 Human Subjects

In our experiments, twenty human subjects completed submaximal elbow flexion and extension to produce data for comparison with our SNMS model. Each subject was asked to stand with his upper arm held by his side and perform a biceps curl under 9 different conditions: 3 speeds (45 $\frac{deg}{sec}$, 90 $\frac{deg}{sec}$, 135$\frac{deg}{sec}$) and 3 loads (20%, 50%, and 80% of the subject's maximum repetition weight (MRW)). Five repetitions were completed for the 20% and 50% conditions and three repetitions were completed for the 80% condition. Speed was controlled with a metronome. The order of speeds was randomized and the order of loads within a speed was randomized. Subjects were allowed to practice the speed of the movement in an unloaded condition prior to beginning testing for that speed condition. A two-minute and five-minute rest was permitted between the load and speed conditions, respectively. In this paper we present data corresponding to three human subjects at a speed of 45 $\frac{deg}{sec}$ at 20% and 50% of the subjects' MRW.

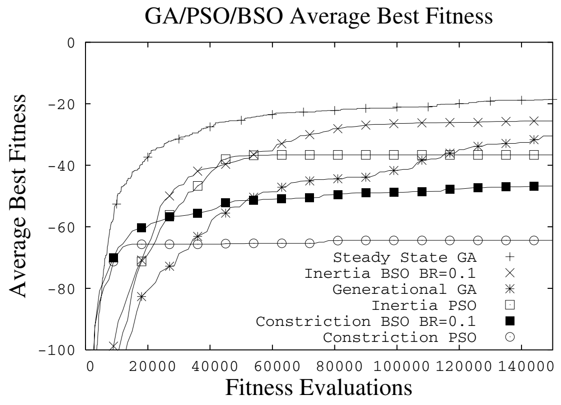

Figure 4: This shows the average best fitness of each algorithm over 35 runs. The steady state GA mean is statistically higher than all other algorithms with $p < 0.01$, however it is higher than the constriction PSO with $p < 0.10$ (Student's t-test). The means of the BSO with inertia and constriction are marginally higher than their PSO counterparts with $p = 0.065$ and $p = 0.013$, respectively (Student's t-test).

5. RESULTS

Now we compare the relative performance of each algorithm and examine the distribution of best fitness values at the end of each algorithm run. We also examine the behavior of individually trained systems and compare this with electromyogram (EMG) data recorded from the human subjects. EMGs reflect the amount of overall muscle activity by measuring the excitation of a muscle.

5.1 Fitness

Figure 4 shows the average best fitness of each algorithm averaged over 35 trials over 150,000 fitness evaluations. On average, the steady state GA yields the highest fit solutions at the end of training followed closely by the inertia BSO. Figure 5 and Table 3 show the distribution of solutions for each algorithm with respect to fitness at the end of each training run.

Table 3: Mean, Standard Deviation of Mean, Best, and Worst Fitness of Each Algorithm at the End of Training

Algorithm	Mean	Std. Dev.	Best	Worst
GA (Steady State)	-18.56	4.94	-10.95	-38.75
GA (Generational)	-30.48	12.91	-15.66	-75.05
PSO (Inertia)	-36.60	28.04	-9.12	-116.13
PSO (Constriction)	-64.40	53.17	-11.70	-210.80
BSO(Inertia)	-25.59	20.46	-5.72	-88.79
BSO (Constriction)	-46.73	43.10	-10.01	-177.16

5.2 Behaviors

These results show behaviors of individually trained SNMS models and the human subjects with EMG readings. In our

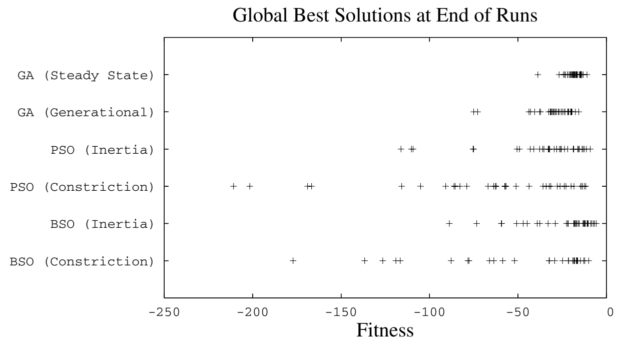

Figure 5: The distribution of the steady state and generational GAs tightly surround their respective means whereas the other algorithms have much wider distributions. The BSO algorithms appear to have slightly tighter distributions around their means than their PSO counterparts.

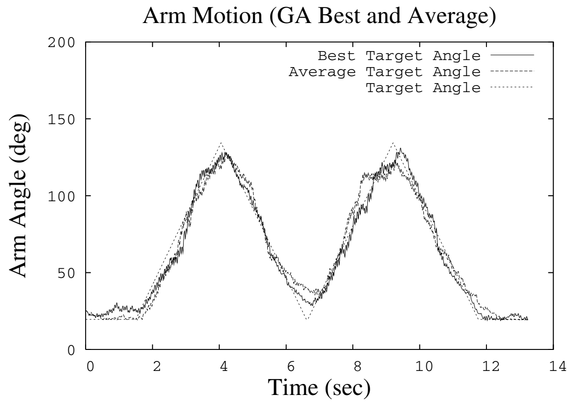

Figure 6: This shows the best fit solution generated by the steady state GA and an average fit solution. The GA's best solution binds more tightly to the target angle throughout the motion.

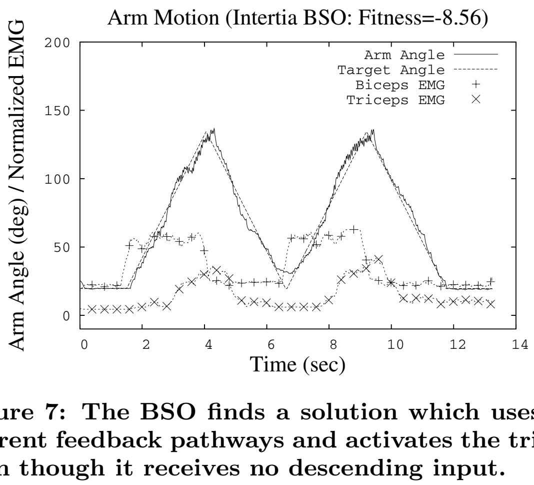

Figure 7: The BSO finds a solution which uses the afferent feedback pathways and activates the triceps even though it receives no descending input.

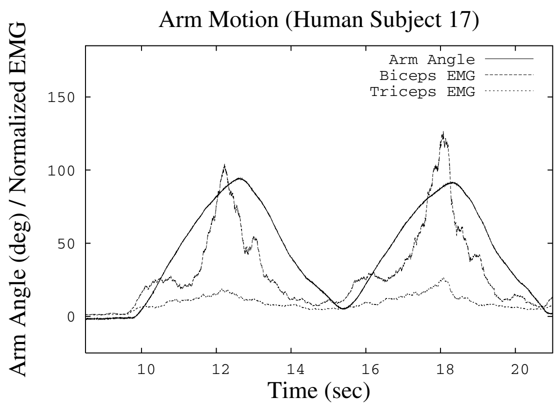

Figure 8: This shows the anatomical movements and surface electrode EMG readings during human subject 17's upward and downward motions at 45$\frac{deg}{sec}$ at 50% of MRW.

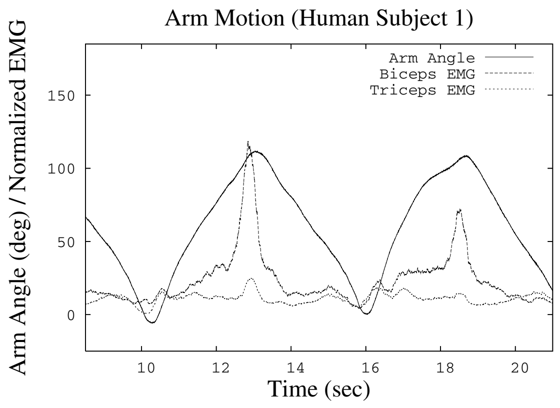

Figure 9: This shows the anatomical movements and surface electrode EMG readings during human subject 1's upward and downward motions at 45$\frac{deg}{sec}$ at 20% of MRW.

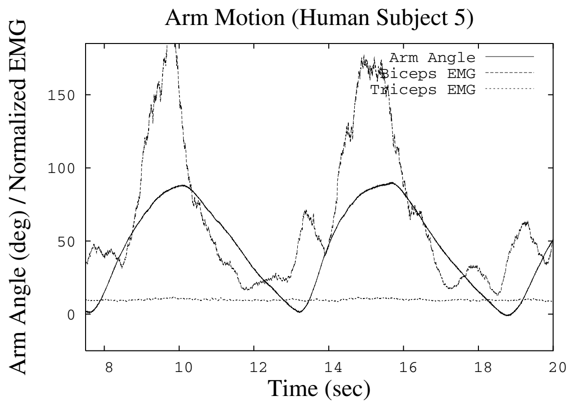

Figure 10: This shows the anatomical movements and surface electrode EMG readings during human subject 5's upward and downward motions at 45$\frac{deg}{sec}$ at 50% of MRW. Note that for this subject increased triceps activity is not needed for stable motion.

experiments we observe that each algorithm generates indistinguishable solutions for a given fitness value. Figures 6 and 7 produce similar anatomical motions and are generated by the steady state GA and the inertia BSO, respectively. Each algorithm consistently yields solutions which activate the triceps during the upward motion[2]. Interestingly, the triceps EMG in two human subjects (Figure 8 and 9) and the PSO solution (Figure 7) steadily increases during the upward motion. However, we hypothesize that human subject 5 has a particularly strong and toned upper body because virtually no extra triceps activity is needed to stabilize the motion (Figure 10). Many solutions also generate small amounts of muscle fiber excitation, i.e. tonic tension, in the absence of descending input also shown in Figure 7 which is also seen in human subject 17 (Figures 8) just before motion begins.

6. CONCLUSIONS

In this paper we show that GAs, PSOs, and BSOs can be used to fill in the unknown details of the SNMS model, e.g. the strengths of the individual muscle fibers and synaptic links, to integrate them all in a single model. We find that these algorithms effectively train the SNMS model to follow the target motion. On average, the steady state GA outperforms all the other algorithms and generates the most consistent results with respect to fitness. We hypothesize that the PSO based algorithms yield less consistent results because the position update equation is a function only of the velocity and not position. The velocity values for the binary component direct the particle to a particular position in binary space, but as the velocity for a given dimension approaches 0, the position alternates between 0 and 1 with equal probability (Equation 9). Since the binary values in the particle represent which subnets receive descending inputs, changing these values during algorithm convergence will likely result in large changes in fitness which may make it difficult for the PSO based algorithms to generate more consistent solutions.

Nevertheless, these algorithms generate fundamental biological behaviors in the SNMS model that are not directly trained. In biology, during contraction of the biceps, the triceps usually becomes active to stabilize motion (Figure 8) which is observed in the model (Figure 7). It is important to note that the triceps motor units do not receiving descending signals in the simulation, and thus could remain inactive. However, the algorithms tend to utilize the triceps via the afferent pathways to generate solutions that satisfy the target motion. This suggests that the afferent feedback components are sufficient to generate the same behaviors they are responsible for in biology.

The algorithms also tend to train the feedback pathways in the model to produce small amounts of muscle fiber excitation, i.e. tonic tension, in the absence of descending input (Figure 7). In biology, the spinal cord's stretch reflex mediated by the afferent pathways maintains this tension which plays a key role in maintaining balance and posture. In the simulation, it facilitates rapid muscle fiber responses at the initiation of the upward motion which is consistent with the function of the stretch reflex and other related reflexes [12].

[2]EMG data from the simulation is taken from a variable in the Hill model which corresponds to the voltage across the neuromuscular endplate. Surface electrodes approximate this voltage on a larger scale.

The successful training of our SNMS model opens the door for modeling increasingly complex neuromuscular system models for a range of anatomical motions such as raising and lowering the arm at varying speeds and by training the system to lift various weights. Researchers can also expand the model architecture by adding neuron types and new types of feedback such as joint or pain receptors. The model can also be changed easily to simulate various motions in different positions such as moving the arm in the horizontal plane or in a supine orientation. These types of improvements in neuromuscular system modeling will increase our understanding of the spinal cord as a nonlinear control system. This knowledge will also allow researchers enhance the information processing abilities of robotic control systems.

7. ACKNOWLEDGEMENTS

We thank Dr. Kathy Browder and her assistant Jessica Sampson for their helpful discussions and for providing the human subject data.

8. REFERENCES

[1] M. Abeles, G. Hayton, and D. Lehmann. Modeling compositionality by dynamic binding of synfire chains. *Journal of Computational Neuroscience*, 17:179–201, 2004.

[2] J. E. Baker and D. D. Thomas. A thermodynamic muscle model and a chemical basis for a.v. hill's muscle equation. *Journal of Muscle Research and Cell Motility*, 21:335–344, 2000.

[3] T. Bui, S. Cushing, D. Dewey, R. Eyffe, and P. Rose. Comparison of the morphological and electronic properties of renshaw cells, Ia inhibitory interneurons, and motoneurons in the cat. *Journal of Neurophysiology*, 90:2900–2918, 2003.

[4] B. Cartling. Control of computational dynamics of coupled integrate-and-fire neurons. *Biological Cybernetics*, 78:383–395, 1997.

[5] M. Clerc. Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation*, pages 601–610, 1999.

[6] N. Durand and J.-M. Alliot. Neural nets trained by genetic algorithms for collision avoidance. *Applied Intelligence*, 13:205–213, 2000.

[7] A. Eiben and J. Smith. *Introduction to Evolutionary Computing.* Natural Computing. Springer, 1998.

[8] G. J. Ettema and K. Meijer. Muscle contraction history: Modified hill versus an exponential decay model. *Biological Cybernetics*, 83:491–500, 2000.

[9] D. Floreano and C. Mattiussi. Evolution of spiking neural controllers for autonomous vision-based robots. In *Proceedings of the International Symposium on Evolutionary Robotics From Intelligent Robotics to Artificial Life*, pages 38–61. Springer-Verlag, 2001.

[10] M. Giuglaiano, M. Bove, and M. Grattarola. Activity driven computational strategies of a dynamically regulated integrate-and-fire model neuron. *Journal of Computational Neuroscience*, 7:247–254, 1999.

[11] A. V. Hill. The heat of shortening and the dynamic constants of muscle. *Proc. Roy. Soc. London*, 126(843):136–195, 1938.

[12] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Priciples of Neural Science.* McGraw-Hill, New York, 4th edition, 2000.

[13] A. Kasinski and F. Ponulak. *Experimantal Demonstration of Learning Properties of a New Supervised Learning Method for the Spiking Neural Networks*, pages 145–152. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005.

[14] J. Kennedy and R. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the Conference on Systems, Man, and Cybernetics*, pages 4104–4109, 1997.

[15] W. Maass and B. Ruf. The computational power of spiking neurons depends on the shape of the postsynaptic potentials. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(25), 1996.

[16] M. Mandischer. Evolving recurrent neural networks with non-binary encoding. In *IEEE International Conference on Evolutionary Computation*, volume 2, pages 584–589, 1995.

[17] T. A. McMachon. *Muscles, Reflexes, and Locomotion.* Princeton University Press, 1984.

[18] N. Pavlidis, O. Tasoulis, V. Plagianakos, G. Nikiforidis, and M. Vrahatis. Spiking neural network training using evolutionary algorithms. In *Proceedings of the 2005 International Joint Conference on Neural Networks*, pages 2190–2194, 2005.

[19] B. Ruf and M. Schmitt. Learning temporally encoded patterns in networks of spiking neurons. *Neural Processing Letters*, 5:9–18, 1997.

[20] M. Settles and T. Soule. Breeding swarms: a ga/pso hybrid. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 161–168. ACM Press, 2005.

[21] M. J. Shelley and L. Tao. Efficient and accurate time-stepping schemes for integrate-and-fire neuronal networks. *Journal of Computational Neuroscience*, 11:111–119, 2001.

[22] G. Shepherd. *Neurobiology.* Oxford University Press, New York, 3rd edition, 1994.

[23] T. Wennekers and G. Palm. Controlling the speed of synfire chains. In *International Conference on Artificial Neural Networks (ICANN)*, pages 451–456, Berlin, 1996. Springer.

[24] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[25] A. Yazdanbakhsh, B. Babadi, S. Rouhani, E. Arabzadeh, and A. Abbassian. New attractor states for synchronous activity in synfire chains with excitatory and inhibitory coupling. *Biological Cybernetics*, 86:367–378, 2002.