# Artificial spider: eight-legged arachnid and autonomous learning of locomotion

Nabil I. Alshurafa and Justin T. Harmon

Department of Computer Science, University of California, Los Angeles

## ABSTRACT

Evolution has produced organisms whose locomotive agility and adaptivity mock the difficulty faced by robotic scientists. The problem of locomotion, which nature has solved so well, is surprisingly complex and difficult. We explore the ability of an artificial eight-legged arachnid, or animat, to autonomously learn a locomotive gait in a three-dimensional environment. We take a physics-based approach at modeling the world and the virtual body of the animat. The arachnid-like animat learns muscular control functions using simulated annealing techniques, which attempts to maximize forward velocity and minimize energy expenditure. We experiment with varying the weight of these parameters and the resulting locomotive gaits. We perform two experiments in which the first is a naive physics model of the body and world which uses point-masses and idealized joints and muscles. The second experiment is a more realistic simulation using rigid body elements with distributed mass, friction, motors, and mechanical joints. By emphasizing physical aspects we wish to minimize, a number of interesting gaits emerge.

**Keywords:** artificial life, robotics, locomotion, animat, spider, simulated annealing, physics modeling

## 1. INTRODUCTION

An organism's actions are heavily influenced, if not dictated, by its physical body. Take away the ability to feel hunger, for example, and a large set of actions dictated by that paradigm will disappear. We postulate that an eight-legged animat with a correctly-modeled physics-based body can learn to control its own muscles to achieve a walking gait. Our approach is to give the animat the ability to respond to feedback from its own body (i.e. determine total muscle-energy expenditure and forward velocity). The animat then uses this information, along with a stochastic minimization algorithm,[1] to make perturbations to its muscle control functions. These perturbations are not random; rather, they attempt to maximize forward velocity while simultaneously minimizing total muscle-energy expenditure. We penalize large changes in the first and second derivatives, as these movements lead to energy-inefficient solutions. The environment is a three-dimensional flat plane, free of obstacles, on which the animat must learn to walk.

Two experiments vary degrees of physical accuracy in modeling. Section 3 presents the first experiment, which assumes a naive physics model of both the world and the animat. The components of the animat body are modeled as point-masses, joints as rotating spheres, and controlled by a simple inverse-kinematics system. Section 4 presents the second experiment which uses a fully-featured physics engine, the Open Dynamics Engine,[2] to more realistically move the animat over a flat plane.

## 2. BACKGROUND

The most well-known work on adaptive behavior in locomotion is Randall Beer's cockroach.[3] Using simulated neurons, Beer trained neural networks[4] to learn locomotion by maximizing forward velocity. Due to computational limits, his approach modeled the roach's legs as two-dimensional line segments with no joints. In the last decade, computer power has advanced to the point where true complex physics models, in three dimensions, have been developed which simulate artificial organisms. Work by Terzopoulos et al. have shown realistic movement of artificial fishes[5] by using models of spring-damper systems for body deformation.

Further author information:
J. Harmon: jharmon@cs.ucla.edu
N.Alshurafa.: nalshurafa@toyon.com

Unmanned Systems Technology VIII, edited by Grant R. Gerhart, Charles M. Shoemaker, Douglas W. Gage,
Proc. of SPIE Vol. 6230, 62301E, (2006) · 0277-786X/06/$15 · doi: 10.1117/12.666491

The problem of robotic locomotion is typically divided into two components: Coordination and Control.[6] The Coordination problem describes how to organize the movements of a complicated figure into coherent, useful motions. In the example of the eight-legged animat with multiple degrees of freedom (DOF), the answer to the Coordination problem tells us which legs to move and how to move them. The Control problem describes how to change the movements over time. Here, we must compute time-dependent attributes for each DOF to achieve the required motion. Michael McKenna and David Zeltzer used physically-based methods with dynamic motor control to generate realistic movements.[7] They also used biologically-based mechanisms to coordinate complex patterns of movement, based on the work of Featherstone.[8,9] In this paper, we combine the Coordination and Control problems and utilize a global optimization algorithm known as simulated annealing[10] to solve locomotion.

Muybridge and Hildebrand have analyzed motion patterns of gaits in different animals, including galloping, cantering, and trotting.[11–13] Bizzi et al. argues that motion is controlled by tuning spring-like properties of muscles during movement.[14] Bizzi's notion of springs as muscles is the impetus for the first of two experiments in our exploration of autonomous gait generation. Our first experiment models the world and body of the eight-legged animat with naive physics, using simulated spring-damper systems to simulate muscles and point-masses to model the pieces of the body. Real-life biological spiders do not have muscles, but instead use a hydraulic system of pumping fluids into and out of the legs to extend and contract them. Nonetheless, our approach was to simulate the body of the animat with muscles attached to rotating joints, greatly simplifying computation. Our second experiment, an extension of the first, utilizes more realistic physics. The body is modeled with rigid body dynamics, the joints with motors, and the system controlled by a fully-featured physics engine, the Open Dynamics Engine.[2]

## 3. EXPERIMENT I: NAIVE PHYSICS

The first experiment uses a naive physics approach to modeling the eight-legged animat and its environment. The location of each point-mass of the animat's body is calculated with a simple physics engine for every clock tick in the world. In our experiments, this tick occurs every 10 milliseconds.

### 3.1. Body

We modeled the body as a cephalothorax and eight legs. Each leg comprises a set of joints, and each joint is rotated via muscles. The joints are modeled as point-masses and the muscles are modeled as spring-damper systems. A point-mass is a virtual object with an absolute three-dimensional position, velocity, orientation, and mass. Rotation of a joint causes the joints below that joint to move through space in an arc, its angle dictated by the angle of rotation of the root joint. This is roughly equivalent to an inverse kinematics engine. The cephalothorax is itself a point-mass and is connected to each leg via the hip joint.

### 3.2. Legs and joints

In this experiment, each leg is composed of three leg segments. A leg segment is defined as a straight (massless) line between two adjacent joints. The legs are actually modeled with point-masses at each joint composing the leg. There are four joints which define the leg: the hip joint, upper leg joint, lower leg joint, and foot joint. In our implementation, the foot joint is an abuse of terminology and usage since the foot joint isn't being used as a joint at all; there are no muscles attached to it. Rather, it is being used as a point-mass to represent the foot. See figure 1.

Each joint is modeled as a point-mass with a radius, defining an outer "shell" on which the muscle is attached. As the muscle contracts or expands, the joint is rotated accordingly. To track rotation, each joint must maintain an orientation vector, which points to the next adjacent joint at all times. There is also a velocity and acceleration associated with each point-mass. The velocity of the cephalothorax is examined to determine efficiency of the walking gait. Further, each joint is rotated by muscles. Each muscle is "anchored" at its ends to the previous joint's center and the outer shell of the next joint. Muscles controlling the hip joint are anchored to the center of the cephalothorax.

The hip joint connects each leg to the cephalothorax and is unique in that it is the only joint which is allowed to rotate both vertically and horizontally. It is responsible for rotating the entire leg in the vertical

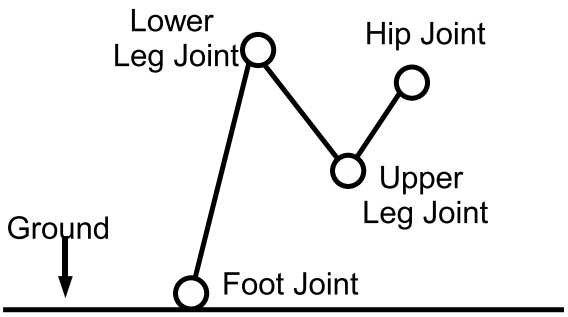



**Figure 1.** Schematic of the leg

and horizontal directions. Two muscles are connected to each hip joint, a *vertical hip muscle* and a *horizontal hip muscle*. The *vertical hip muscle* is responsible for rotations in the vertical while the *horizontal hip muscle* is responsible for rotations in the horizontal. In our design, the hip joint is allowed to rotate a total symmetric arc of 60˚about both axes.

The upper leg joint is connected to the hip joint and the lower leg joint. Both the upper and lower leg joints can only rotate in one axis and, therefore, have only one muscle attached to each joint. Rotations of any joint will change the positions of all joints below it (via inverse kinematics). Therefore, a rotation of the hip joint will cause a change in the position of the upper leg joint, which will cause a change in the position of the lower leg joint, which will cause a change in location of the foot joint. The foot joint is unique in that it is not used as a true joint. Rather, it is used as a point-mass to represent the tip of the foot. Rotating this "joint" would produce no effect. However, the foot is important in determining when rotation of a leg exerts a horizontal force on the entire animat. By definition, movement can occur only when the foot is making contact with the ground.

### 3.3. Muscles

Each joint is rotated via a simulated muscle. In the case of the hip joint, which rotates in two axes, two muscles control overall movement. Other joints rotating in only one axis are controlled by one muscle. Each of the legs is controlled by four muscles: two muscles control the hip joint, and one muscle each for the upper and lower leg joints.

The muscles controlling vertical and horizontal rotation of the hip joint we refer to as the *vertical hip muscle* and *horizontal hip muscle*, respectively. These two muscles are anchored to the point-mass of the cephalothorax and to the outer surface of the sphere comprising the joint. The muscle controlling rotation of the upper leg joint we refer to as the *upper leg muscle*, and controls vertical rotation of the upper leg joint. Note that the upper leg joint cannot rotate horizontally. The final muscle controlling rotation of the lower leg joint we refer to as the *lower leg muscle*. This muscle controls vertical rotation of the lower leg joint, which cannot be rotated horizontally. Contraction and expansion of each muscle rotates its associated joint.

Associated with each muscle is a rest length, a minimum length (of contraction), and a maximum length (of expansion). These values determine the permitted range of rotation, and vary from muscle type to muscle type because of nonuniform distance between joints. When a muscle is displaced from its rest length, energy expenditure can be calculated with the spring energy equation: $\boldsymbol{Energy = \frac{1}{2}kx^2}$, where k is the spring constant and x is the displacement from rest length.

#### 3.3.1. Muscle control functions

Each muscle is controlled by setting its contraction/expansion length from 0% to 100%, 0% corresponding to minimum length and 100% corresponding to maximum length. To expand/contract the muscles over time, a muscle control function is necessary. Our implementation uses 100 values per muscle, successively applied to that muscle every clock tick. Since our clock ticks once every 10 milliseconds, each muscle control function repeats once every second. With four muscles per leg and eight legs, there are thirty-two muscle control functions

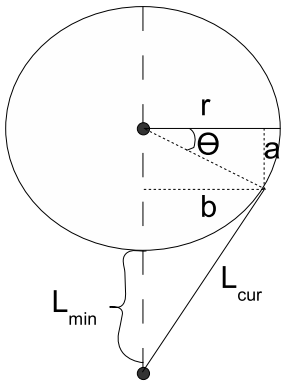



**Figure 2.** Muscle-joint trigonometric setup

and 3200 values. The reinforcement learning will modify these values to converge on an energy-efficient function to control locomotion.

### 3.3.2. Muscle length and joint rotation angle

Because a contraction or expansion of a muscle corresponds to a rotation of the joint to which it is attached, it is important to convert between joint rotation angle and muscle length. The absolute minimum ($\boldsymbol{L_{min}}$) and maximum ($\boldsymbol{L_{max}}$) length of any muscle corresponds to the closest and farthest points on the joint from the anchor point of the muscle, respectively. A muscle is controlled by setting it to a percentage, $\boldsymbol{p}$, of maximum length $\boldsymbol{L_{max}}$. The absolute current length, $\boldsymbol{L_{cur}}$, is calculated via equation 1. Because the muscle travels along the circumference of the joint, simple trigonometry can be used to establish a relation between the joint rotation angle $\boldsymbol{\theta}$, $\boldsymbol{p}$, and $\boldsymbol{L_{cur}}$ via equation 2. Refer to figure 2. Similar equations can be derived if $\boldsymbol{L_{min}}$ and $\boldsymbol{L_{max}}$ aren't defined by the closest and farthest points on the joint.

$$\boldsymbol{L_{cur} = L_{min} - p(L_{max} - L_{min})} \tag{1}$$

$$\boldsymbol{L_{cur} = \sqrt{(L_{min} + r(1 - \sin(\theta)))^2 + (r\cos(\theta))^2}} \tag{2}$$

## 3.4. Physics model, forces, and locomotion

Without adequately defining the physics of each point-mass, the animat is incapable of proper feedback from the environment and, therefore, incapable of properly learning a locomotive gait. When the foot joint is contacting the ground and the hip joint is rotated backward, a forward force is exerted on the cephalothorax of the animat. By summing all such forces, we determine the motion of the cephalothorax and the animat as a whole.

Each muscle is defined as a spring-damper system with a point-mass attached to it. Each point mass $\boldsymbol{i}$ has associated with it a position $\boldsymbol{x_i}$, velocity $\boldsymbol{v_i}$, and acceleration $\boldsymbol{a_i}$. Let $\boldsymbol{k_{ij}}$ denote the spring constant for the muscle connecting point-masses $\boldsymbol{i}$ and $\boldsymbol{j}$ and $\boldsymbol{d_{ij}}$ denote the displacement from the rest position $\boldsymbol{l_{ij}}$ of that spring, where $\boldsymbol{d_{ij} = ||r_{ij}|| - l_{ij}}$ and $\boldsymbol{r_{ij} = x_j - x_i}$. To compute the force exerted on point-mass $\boldsymbol{i}$ from this muscle, the translational spring equation can be used. In equation 3, the component $\frac{\boldsymbol{r_{ij}}}{\boldsymbol{||r_{ij}||}}$ determines the direction of the force applied:

$$\boldsymbol{F_i^j = \frac{k_{ij} d_{ij} r_{ij}}{||r_{ij}||}} \tag{3}$$

An important contributing force is gravity, which is simulated with a uniform field exerting a force of $\boldsymbol{9.8\frac{m}{s^2}}$. For the animat to remain static, forces due to gravity must be equivalent to forces holding the animat standing. Therefore, as one leg is raised from the ground, its effective weight becomes distributed among the remaining legs contacting the ground. To determine the position, velocity, and acceleration of each point-mass per clock tick, all forces on each point-mass are integrated.

### 3.5. Learning to walk

Each muscle maintains a control function consisting of 100 float values between 0 and 1, corresponding to the percent the muscle is extended. The reinforcement learning technique known as simulated annealing[10] works by perturbing a solution vector by small amounts at each energy level. Changes in energy level correspond to large global perturbations in the solution vector.

An objective function must exist to evaluate each proposed solution, $\boldsymbol{S}$ for its "fitness". In our case, we attempt to maximize forward velocity while minimizing total energy expended. Further, we wish to penalize non-fluid motion which corresponds to the first and second derivatives of the muscle control functions. Therefore, our objective function must reward forward velocity and penalize energy expenditure and erratic movements. In our implementation, the objective function is similar to that used by Terzopoulos et al:[5]

$$\boldsymbol{E(S) = v_1 E_m(S) + v_2 E_v(S) + v_3 E_u(S)} \tag{4}$$

Here, coefficients $\boldsymbol{v_1}$, $\boldsymbol{v_2}$, and $\boldsymbol{v_3}$ specify how each energy measure should be rewarded or penalized. $\boldsymbol{E_m}$ is the summation of all energy expended by all muscles for the proposed solution, $\boldsymbol{E_v}$ is the forward velocity of the cephalothorax, and $\boldsymbol{E_u}$ is the summation of the averages of the first and second derivatives of the muscle control functions in the proposed solution. With this formulation, the "fitness" of a solution is determined by how low its objective value evaluates: if $\boldsymbol{E(S') < E(S)}$ then $\boldsymbol{S'}$ is a better solution than $\boldsymbol{S}$.

The process of learning the correct muscle controls follows algorithm 1. An initial solution is generated by randomly populating the muscle control functions. For each new candidate solution, we evaluate its objective value to determine if it is better than our current solution. If its objective value is smaller, we replace our current solution with the new solution. One of the strengths of simulated annealing is its ability to avoid getting trapped at a local minimum. This is done by calculating a probability of accepting a candidate solution even if it is not better than the current solution. The smaller the difference between the current candidate's objective value and the current accepted solution's objective value, the greater the probability of accepting it as a new solution. We use the following equation to calculate this probability:

$$\boldsymbol{P = e^{\frac{-\Delta E}{T}}} \tag{5}$$

Here, $\boldsymbol{P}$ is the probability of accepting the solution even though it is not better than the current solution, $\boldsymbol{\Delta E}$ is $\boldsymbol{E(S') - E(S)}$, and $\boldsymbol{T}$ is the current temperature. Our annealing schedule is to logarithmically decrease the system temperature by 95% at each annealing step. We require each annealing step to complete 10 accepted solutions before reducing the temperature to another annealing step.

**Algorithm 1** Simulated Annealing Algorithm

1: Generate new candidate solution, $\boldsymbol{S'}$
2: Is $\boldsymbol{E(S') < E(S)}$? If yes, go to 3. If not, go to 6.
3: Replace current solution $\boldsymbol{S}$ with $\boldsymbol{S'}$.
4: Adjust system temperature if we have accepted 10 solutions.
5: Terminate? If yes, go to 7. If not, go to 1.
6: Do we accept $\boldsymbol{S'}$ even though it is worse than $\boldsymbol{S}$? If yes, go to 3. If not, go to 5.
7: Done.

### 3.6. Discussion of results

Our initial tests did not attempt to guide the simulated annealing optimization with the $\boldsymbol{E_u}$ term, leading to instantaneous movements which produced overall forward movement, but which were unrealistic and ultimately inefficient. Varying $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ placed different emphasis on the importance of velocity vs. energy efficiency. When $\boldsymbol{v_1}$ was emphasized, overall movement was slight. This is due to the overwhelming energy inefficiency allowed in the solutions. When $\boldsymbol{v_2}$ was emphasized, seemingly random leg movements occurred which had the overall effect of propelling the body forward. These tests seemed to produce functions which would "swing" the spider back and forth with the overall result being a positive forward velocity.

More interesting results occurred when we added a term to account for first and second derivatives of the muscle control functions. With emphasis on the additional term, a more fluid movement resulted which more closely approximated a natural walking gait. We found our best configuration gave $\boldsymbol{E_u}$ 10 times more emphasis than $\boldsymbol{E_v}$, and gave $\boldsymbol{E_v}$ about 10 times more weight than $\boldsymbol{E_m}$. Final results took approximately 700 annealing steps.

This experiment showed that interesting walking gaits can be achieved by using reinforcement learning methods and naive physics. However, to fine tune the resulting gait requires designed guidance of the learning process, in the form of programming constraints.

## 4. EXPERIMENT II: FULL-FEATURED PHYSICS ENGINE

The physics model of the world and the animat in the first experiment are highly idealized and meant to approximately simulate a biological organism. To test whether this technique could be used for autonomous gait generation with a robotic eight-legged animat, we use a full-featured physics engine called the Open Dynamics Engine.[2] Among other things, this engine handles rigid body structures, masses, gravity, collision detection, friction (using the Coulomb Model), inverse kinematics, and joint control via motors.

### 4.1. Body

The body of the animat is modeled similar to that of the first experiment, at least in structure. A cephalothorax is connected to eight legs and each leg is modeled with three leg segments. In this experiment, however, we model the physical leg segments and cephalothorax as rigid bodies. Each rigid body has a position, velocity, angular velocity, orientation, and center of mass. Additionally, each rigid body has an Inertia matrix which describes how the body's mass is distributed around the center of mass. For this experiment, all rigid bodies are modeled with a uniform distribution of mass. The cephalothorax is modeled as a solid sphere, and each leg segment as a solid capped cylinder. Each leg is spaced at 40˚intervals along the circumference of the cephalothorax. The body is placed on a flat, infinite plane with friction and gravity. See figure 3.

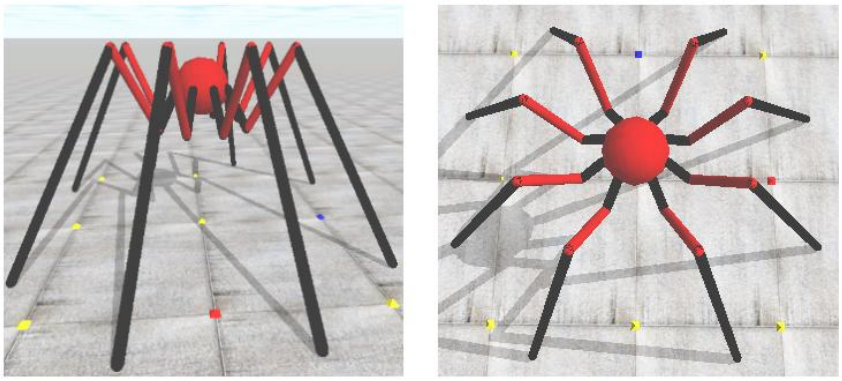


**Figure 3.** Top and side view of model

### 4.2. Legs and joints

In this experiment, we utilize ODE's joint mechanism to connect the rigid sections of the body together. Specifically we use the Universal-type joint mechanism, which keeps two axes of rotation perpendicular throughout all movements of the joint. The Universal joint mechanism constrains joint movement to two degrees of rotational freedom about a pair perpendicular axes, one per axis. In other words, given axis 1 of body 1 perpendicular to axis 2 of body 2, the joint ensures axis 1 and axis 2 are always perpendicular. By restricting the minimum and maximum allowable angles of rotation of each axis, movement via the Universal joint can be constrained. The dimensionality of the joint can also be reduced by specifying equal maximum and minimum rotation angles for

a particular axis, effectively turning the joint into either a hinge (1 degree of freedom) or a fixed and unmovable joint (0 degrees of freedom).

Each leg is composed of three segments, as in experiment 1. Here, however, each segment is a rigid body which is attached to the next segment via a joint. The hip joint is defined similarly to that in experiment 1. It can rotate vertically and horizontally, requiring two degrees of freedom from the joint. Vertical and horizontal rotations are constrained to a symmetric arc of 30°. To further constrain the solution space, we fix the upper and lower leg joints by making them non-rotating. There is no foot joint in this model, since the purpose of such a joint in experiment 1 is to serve as a point-mass. Refer to figure 4.

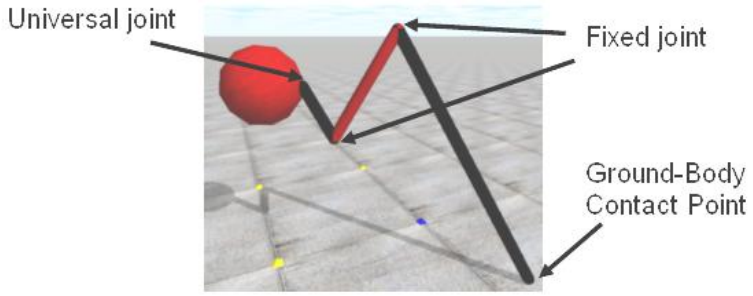



**Figure 4.** Types of joints used in the leg

## 4.3. Motors

Rather than controlling joints with muscles, we use the built-in ODE motor to rotate joints. The motor allows the relative angular velocities of two bodies to be controlled. To rotate a body, an angular velocity must be applied about a rotational axis controlled by the motor. We attach a motor to each rotational axis in each universal joint. Since there are two rotational axes per leg, and eight legs, there are sixteen motors controlling the locomotion of the animat. To rotate a joint, an angular velocity is specified at each time step, and the necessary force is applied by the ODE provided that force does not exceed the maximum torque of the motor.

### 4.3.1. Motor control functions

Motors are controlled via discretized functions, similar to experiment 1, which are repeated once every second. The system clock tick frequency is once every 10 milliseconds, as in experiment 1. Since motors are continuous by nature, we reduced the number of discrete values in each control function to four. This results in a value being applied continuously to the corresponding motor once every 250 milliseconds, and for a duration of 250 milliseconds. The reinforcement learning algorithm modifies these control values to achieve a locomotive gait.

## 4.4. Learning to walk

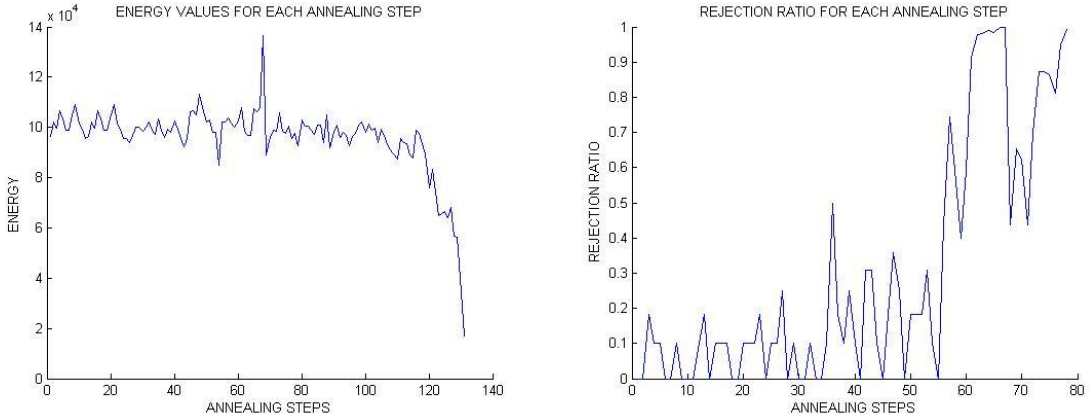
The animat learns the best sequence of repeating control values (angular velocities) for the motors to achieve a walking gait. These velocities produce forces which, in conjunction with the physics environment, yield a locomotive system. As in experiment 1, simulated annealing is used as a reinforcement learning method which perturbs the control functions until an acceptable solution is reached.

Objective functions of the form in equation 4 allow us to analyze the effects of individual component contributions to the resulting walking gait. To avoid sub-optimal solutions, we apply heuristics which aid the algorithm in retreating from local minima, while continuing the search for a better solution. The heuristic incorporates into the algorithm a rejection ratio which defines the percentage of total solutions rejected. The algorithm proceeds as it did in experiment 1, except when the rejection ratio exceeds 99.8%. In this case, we revert to the previous accepted solution and continue our annealing process. Figure 5 shows the performance of the algorithm when backing from local minima. Refer to algorithm 2.

Before computing the objective function for each candidate solution, we apply the candidate solution to the animat over multiple clock cycles (i.e. several seconds) and compute the average velocity over this interval.

**Algorithm 2** Simulated Annealing Algorithm with Heuristic

1: Generate new candidate solution, $\boldsymbol{S'}$
2: Is $\boldsymbol{E(S')} < \boldsymbol{E(S)}$? If yes, go to 3. If not, go to 7.
3: Replace current solution $\boldsymbol{S}$ with $\boldsymbol{S'}$.
4: Adjust system temperature if we have accepted 10 solutions and go to 6.
5: If rejection ratio $> 0.998$, replace $\boldsymbol{S}$ with previous solution.
6: Terminate? If yes, go to 8. If not, go to 1.
7: Do we accept $\boldsymbol{S'}$ even though it is worse than $\boldsymbol{S}$? If yes, go to 3. If not, go to 5.
8: Done.




(a) Energy (b) Rejection Ratio

**Figure 5.** Performance of simulated annealing algorithm

This ensures gaits are stable over longer time periods. In general, the more clock cycles used during a single run of the algorithm, the longer the resulting gait will likely remain stable.

### 4.5. Discussion of results

We are interested in the effect of each component in the objective function (equation 4) on the resulting gait. Use of only the $\boldsymbol{E_v}$ term puts all weight on forward velocity. This seems to result in a gait which can only be described as frog-like. The animat hobbles forward by a movement most closely resembling frog-like jumps. Additionally, because the $\boldsymbol{E_v}$ term does not constrain horizontal movement, the gait tends to veer slightly to the left or right. By incorporating a penalty into the $\boldsymbol{E_v}$ term for horizontal movement, the resulting gaits were straighter and more stable.

Use of the $\boldsymbol{E_m}$ term was coupled with the $\boldsymbol{E_v}$ term to produce the next gaits. Note that the use of the $\boldsymbol{E_m}$ term alone would result in a gait converging on no movement whatsoever. By incorporating the energy term into our objective evaluation function, the resulting gait causes the animat to sink to the ground and crawl. It seems that the annealing function has determined that fighting gravity results in an energy-inefficient solution to locomotion.

As in experiment 1, incorporating the $\boldsymbol{E_u}$ term as the average of the first and second derivatives of the motor control functions results in smoother locomotive gaits, but only slightly so. The computational intensity of including the term requires orders of magnitude more computations. In our results, the benefits of including this term are negligible. We believe this is due to the continuous and smooth nature of the application of force by the motors.

## 5. CONCLUSIONS

Our first experiment attempted to coax a walking gait out of randomness for a biologically-inspired model of an eight-legged animat. Because a true physics engine was not enforcing realistic movements (e.g. non-

instantaneous translations), feedback to the animat for reinforcement learning was limited. Additionally, parameters were required in the objective function which reduced instantaneous and rapid movements. These parameters radically changed the resulting locomotive gaits in the first experiment.

Our second experiment attempted to model a true robotic eight-legged animat with a full-featured physics engine and motors. Because a mature physics engine was in place, non-realistic movements were not allowed. Realistic modeling of the motors guaranteed joint rotation was neither unrealistically rapid nor unrealistically instantaneous. A better collision-detection system also provided realistic feedback to the reinforcement learning algorithm for gaits which push unnecessarily against the ground. In the first experiment these gaits presented a problem when the foot would penetrate the ground.

Interestingly, using a true physics engine allowed us to neglect terms in the objective function for minimizing the first and second derivatives of the control functions. This is because the physics engine enforced realistic constraints on the system which made such movements implausible. The additional computation of a full-featured physics engine were, therefore, offset by the lack of computation required in including first and second derivative components.

Simulated annealing, while computationally expensive, can sometimes produce interesting walking gaits when applied to simulated model locomotive learning. This emerges "for free" in that careful engineering of coordination and locomotion is not required. However, it is computationally intense and may not yield a viable solution.

## 6. FUTURE WORK

We would like to see cognitive abilities added to the animat, as well as abilities to climb and traverse obstacles. A state machine setup would allow us to morph between gaits based on environmental or cognitive states of the animat. Additional work can be done on adaptive gaits which compensate for lost or non-functioning limbs or for occupied limbs (e.g. carrying items).

In addition to gaits, we are interested in physical contortions the animat might learn to perform. For instance, we would like to use reinforcement learning techniques to have the animat learn to wedge itself into a crevice, or learn to right itself after being flipped on its back.

Finally, we are interested in incorporating visual perceptions into the animat to learn from the behavior and movements of those around it. One could envision a community of arachnid-like animats which learn to weave themselves together to span a physical gap, much like ants arrange themselves into a bridge which can be crossed by other ants.

## ACKNOWLEDGMENTS

We wish to thank Toyon Research Corporation for their support on parts of this experiment. We also wish to thank the following individuals: Nicolas Chaumont for his valuable suggestions on the physics engines and editorial comments on this paper, Michael Dyer for helpful early feedback on this research, and Steve Wedig and Joseph Nunn for their valuable comments on early drafts of this paper.

## REFERENCES

1. R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.
2. R. Smith, "Open dynamics engine." [Online]. http://www.ode.org.
3. R. D. Beer, *Intelligence As Adaptive Behavior: An Experiment in Computational Neuroethology*, Academic Press, 1990.
4. R. D. Beer and J. C. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior* **1**(1), 1992.
5. D. Terzopoulos, X. Tu, and R. Grzeszczuk, "Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world," *Artificial Life* **1**(4), pp. 327–351, 1994.

6. P. Kugler, J. Kelso, and M. Turvey, "On the concept of coordinative structures as dissipative structures: I. theoretical lines of convergence," in *Tutorials in Motor Behavior*, G. Stelmach and J. Requin, eds., pp. 3–47, North-Holland Publishing Company, 1980.
7. M. McKenna and D. Zeltzer, "Dynamic simulation of autonomous legged locomotion," in *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pp. 29–38, ACM Press, (New York, NY, USA), 1990.
8. R. Featherstone, "The calculation of robotic dynamics using articulated-body inertias," *The Int. Journal of Robotics Research* **2**(1), pp. 13–30, 1983.
9. R. Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, 1987.
10. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science* **220**(4598), pp. 671–680, 1983.
11. E. Muybridge, *Animals in Motion*, Dover Publishers, New York, 1957.
12. M. Hildebrand, "How animals run," *Scientific American* **May**, pp. 148–157, 1960.
13. M. Hildebrand, "Symmetrical gaits of horses," *Science* **150**, pp. 701–708, 1965.
14. E. Bizzi, N. Accornero, W. Chapple, and N. Hogan, "Central and peripheral mechanisms in motor control," in *New perspectives in cerebral localization*, R. A. Thompson and J. R. Green, eds., pp. 23–34, Raven Press, New York, 1981.