Behavior Chaining: Incremental Behavior Integration for Evolutionary Robotics

Josh Bongard
University of Vermont, Burlington, VT 05405
josh.bongard@uvm.edu

Abstract

One of the open problems in autonomous robotics is how to consistently and scalably integrate new behaviors into a robot with an existing behavioral repertoire. In this work a new technique called behavior chaining is introduced, which allows for gradually expanding the behavioral repertoire of a dynamically behaving robot. The approach relies heavily on scaffolding: gradually restructuring the robot's environment such that selection pressure favors the incorporation of a new behavior. This method teaches a robot a compound behavior not yet reported in the literature: dynamic legged locomotion toward an object followed by grasping, lifting and holding of that object in a physically-realistic three-dimensional environment. The method assumes that success is dependent on the order in which behaviors are learned. This is justified by results which show that if a robot is forced to learn lifting first and then incorporate locomotion, it eventually succeeds at both more often than a robot forced to learn locomotion first and then lifting.

Introduction

Useful autonomous robots must exhibit three properties: they must be able to perform behaviors autonomously; they must be able to adapt an existing behavior on the fly in the face of unexpected situations; and they must be able to exhibit different behaviors in different circumstances. Recent work has reported an autonomous physical robot capable of the former and middle property: maintaining a behavior in the face of unexpected body damage (Bongard et al. (2006)) using automated modeling (Bongard and Lipson (2007)). In this work a virtual robot is introduced that exhibits the former and latter property: it is able to autonomously learn one behavior (lifting) and then integrate a second one (dynamic locomotion) into its repertoire.

Evolutionary robotics (Harvey et al. (1997); Nolfi and Floreano (2000)) is an established technique for generating robot behaviors that are difficult to derive analytically from the robot's mechanics and task environment. In particular, such techniques are useful for realizing dynamic behaviors (eg. Reil and Husbands (2002); Hornby et al. (2005)) in which individual motor commands combine in a nonlinear fashion to produce behavior, thereby making analytical derivations of optimal controllers infeasible. However, evolutionary approaches to dynamic behavior generation have focussed up until now on realizing a single behavior, such as locomotion (Reil and Husbands (2002); Hornby et al. (2005)) or grasping (Fernandez and Walker (1999); Chella et al. (2007)). Alternatively, multiple non-dynamic behaviors have been generated for simpler wheeled robots (Nolfi (1997); Lee et al. (1998)).

The approach described here is a type of robot shaping technique (Singh (1992), Dorigo and Colombetti (1994) and Saksida et al. (1997)) in which the organization of the learning or evolution process is guided manually or automatically. However, in behavior chaining it is assumed that there is an underlying order in which behaviors should be learned, and that this order is dictated more by the agent's morphology, controller, task environment and controller optimization process than it is by the agent's current behavioral competency.

Although it is possible to realize multiple behaviors in a robot by gradually incorporating more modules into its controller (Brooks (1986); Calabretta et al. (2000)), this approach does not scale well. A scalable approach to behavioral flexibility should allow the same dynamic controller to exhibit multiple attractor states, in which individual behaviors correspond to individual attractor states, an idea that is gaining currency in the robotics literature (Inamura et al. (2004); Okada and Nakamura (2004)). One of the main difficulties in this approach however is realizing multistability (Foss et al. (1997)) in the controller: it should settle into different attractor states that correspond to the different desired behaviors in the face of the appropriate sensory stimulation.

The approach to realizing multistable controllers described here relies on scaffolding (Wood et al. (1976)), a concept borrowed from developmental psychology: easing the learning agent's task environment at the outset to allow initial learning, and then gradually removing the constraints to stimulate further learning. The minimal cognition approach (Beer (1996)) has led to agents capable of multiple dynamic behaviors such as legged locomotion and visually-guided orientation (Gallagher and Beer (1999)), but this integration was achieved by awarding the agent for demonstrating both capabilities simultaneously, and without the aid of scaffolding. The work here suggests that it may be easier to evolve a controller that generates one behavior first through scaffolding, and then incorporate additional behaviors by reducing the scaffolding. Further, it is shown here that to realize an agent capable of multiple behaviors some behaviors may have to be learned before others: this suggests that learning multiple behaviors at once may not be scalable, although more investigation into this issue is warranted. Scaffolding has been used with some success in the robotics literature for realizing single behaviors rather than sequences of dynamic behaviors (Pratt et al. (2001); Reil and Husbands (2002); Lungarella et al. (2003); Ziemke et al. (2004)). Alternatively, a teacher may lead a robot through a series of behaviors directly, after which the robot learns to reproduce those behaviors autonomously (Saunders et al. (2007)), but in this approach the exact motions comprising the behaviors must be demonstrated and therefore known *a priori* by the teacher.

In the work presented here we introduce a dynamic scaffolding method that enables a virtual autonomous robot to first learn one dynamic behavior (lifting) and then gradually incorporate a second dynamic behavior (locomotion) using a single monolithic controller. In the next section the method is introduced; the following section reports results demonstrating how this behavioral competency arises; and the final section provides some discussion and concluding remarks.

Methods

In this section the virtual robot is first introduced, followed by its controller. The section concludes with a description of behavior chaining, the dynamic scaffolding method that enables the robot to gradually incorporate new behaviors into its repertoire.

The robot In this work a virtual quadrupedal robot is used (Fig. 1). The robot is comprised of four legs and a front gripper. The legs are comprised of an upper and lower cylinder. The gripper is composed of a small spherical claw base, which connects the main body to the claw pincers. The claw base can be rotated upward relative to the main body, and both the left and right pincers are comprised of a claw arm (proximal to the claw base) and claw tip (distal to the claw base). The robot attempts to grasp and lift a rectangular target object that is placed at varying distances from the front of the robot's body. The physical specifications of the body parts and the joints connecting them together are given in Table 1.

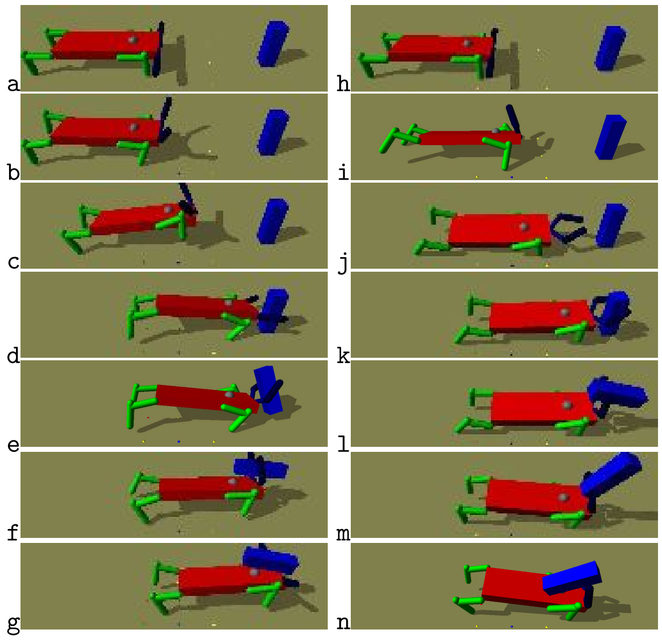


Figure 1: Two evolved behaviors. **a-g**: The robot moves toward the target object placed 2.8 meters ahead (**a-d**), lifts it (**e,f**), and drops it onto its back (**g**). **h-n**: The robot moves toward the target object placed 3.0 meters ahead (**h-k**) and swings it (**l,m**) onto its back (**n**).

Part	length	width	height	mass
Target object[T]	0.4m	0.4	1.4	1kg
Main body[MB]	3.0	1.0	0.3	1
Upper leg[UL]	0.7	0.1*		1
Lower leg[LL]	0.7	0.1*		1
Claw base[CB]	0.1*			0.25
Claw arm[CA]	0.8	0.1*		0.25
Claw tip[CT]	0.8	0.1*		0.25
Joint	min	max	orientation	
[MB][UL]	-20^o	20	sagittal	
[UL][LL]	-20	20	sagittal	
[MB][CB]	-1	120	sagittal	
[CB][CA]	-45	0	frontal	
[CA][CT]	-75	0	frontal	

Table 1: Physical parameters of the robot and environment. *=radius

Eight motors actuate the four upper and lower legs, another motor actuates the claw base, and four motors actuate the base and distal parts of the left and right claw pincers, for a total of 13 motors. A touch sensor and distance sensor reside in both the left and right claw tips, a rotation sensor resides in the claw base, and a distance sensor resides on the robot's back (gray sphere in Fig. 1), for a total of six sensors. The touch sensors return a value of 1 when the corresponding body part touches another object, and zero otherwise. The distance sensors return a value commensurate with the sensor's distance from the target object: they return zero if they are greater than five meters from the object; and a value near one when touching the object. Object occlusion is not simulated here; the object can be considered to be emitting a sound, and the distance sensors respond commensurately to volume.

The controller A continuous time recurrent neural network (Beer (2006)) is used to control the robot. The CTRNN is composed of 11 motor neurons (the two claw arm motors share the same motor neuron, as do the two claw tip motors to ensure the claw closes symmetrically). The remaining 10 motors each receive commands from their own motor neuron. The value of each motor neuron is updated according to

$$\tau_i y_i' = -y_i + \sum_{j=1}^{10} w_{ji}\sigma(y_j - \theta_i) + \sum_{j=1}^{6} n_{ji}s_j \qquad (1)$$

where τ_i is the time constant associated with neuron i, y_i is the value of neuron i, $\sigma(x) = 1/(1+e^{-x})$ is an activation function that brings the value of neuron i back into $[0,1]$, w_{ji} is the weight of the synapse connecting neuron j to neuron i, θ_i is the bias of neuron i, n_{ji} is the weight of the synapse connecting sensor j to neuron i, and s_j is the value of sensor j.

In this formulation, each sensor may have a direct effect on every motor neuron. However this effect may be minimized or eliminated by low values for n, or by behaviors that cause a target motor neuron to reach minimum or maximum values.

The virtual robot with a given CTRNN is evaluated over a set number of simulation steps in a physical simulator[1]. At the outset of each step, the sensor values are retrieved from the physical simulator and the values of the motor neurons are calculated. The resulting values are scaled to the minimum and maximum rotation angles of the corresponding joint (Table 1), forming the desired angle for that joint. Torque is then applied to the joint commensurate with the difference between the joint's current angle and the desired angle. The positions and velocities of the objects in the simulation are then updated using a step size of 0.005; the CTRNN is updated once for each time step.

Behavior Chaining Behavior chaining is a method for dynamically tuning the robot's task environment to facilitate learning which assumes that the order in which behaviors are learned affects the probability of success. The algorithm is outlined in Fig. 2. A random CTRNN is created by choosing all τ from the range $[0.1, 0.5]$, all w from $[-16, 16]$, all θ from $[-1, 1]$, and all n from $[-16, 16]$. This gives a total of $10 + 10*10 + 10 + 6*10 = 180$ evolvable parameters. The robot is then equipped with this controller and allowed to behave in the task environment for 100 time steps, where the target object is placed directly in front of the robot. After evaluation the fitness of the controller is computed as

[1]Open Dynamics Engine, www.opende.com

```
1.  BehaviorChaining()
2.     Create and evaluate random parent $p$
3.     WHILE ~Done()
4.        Create child $c$ from $p$, and evaluate
5.        IF Fitness($c$) $\geq$ Fitness($p$) [see eqns. 2,3]
6.           $p = c$
7.        IF Failure()
8.           EaseEnvironment()
9.           Re-evaluate $p$
10.       WHILE Success($p$)
11.          HardenEnvironment()
12.          Re-evaluate $p$
-----------------------------------------------
13. Done()
14.    18 hours of CPU time have elapsed
-----------------------------------------------
15. Failure()
16.    100 generations since last success
-----------------------------------------------
17. EaseEnvironment()
18.    EvaluationTime $\leftarrow$ EvaluationTime+100
-----------------------------------------------
19. Success($g$)
20.    $\exists k, k \in \{1, \ldots, t\}$ |
21.     $T$(LeftClawTip, $k$)&
22.     $T$(RightClawTip, $k$)&
23.     $D$(SensorNode, $k$) $\geq 0.825$
-----------------------------------------------
24. HardenEnvironment()
25.    TargetDistance $\leftarrow$ TargetDistance+0.01m
```

Figure 2: **Behavior chaining pseudocode.** The algorithm executes a hillclimber [1-14]. If the current genome fails [15,16], the task environment is eased [17,18]; while it is successful [19-23], the task environment is made more difficult [24,25]. $T(x,k)$ returns 1 if body part x is in contact with another object and zero otherwise at time step k. $D(x,k)$ returns the distance of body part x from the target object at time step k.

$$f = \max_{k=1}^{t}(D(\text{LeftClawTip}, k) * D(\text{RightClawTip}, k)) \quad (2)$$

if the touch sensors in the left and right claw tip fail to fire at the same time during any time step of the evaluation period, and

$$f = 1 + \max_{k=1}^{t}(D(\text{SensorNode}, k)) \qquad (3)$$

otherwise, where t is the evaluation time, and $D(x,k)$ indicates the distance of body part x from the target object at time step k. Eqn. 2 rewards controllers for steering the robot toward the target object. Eqn. 3 rewards controllers for also lifting the target object onto the robot's back (where the sensor node is located) after it has touched the target object with both claw tips.

A hill climber (Russell and Norvig (1995)) is used to optimize the initial random CTRNN against this fitness function (Fig. 2[1-12]). Although a more sophisticated optimization process such as a genetic algorithm could be used, hill climbing was found to be sufficient in this case. At each generation a child CTRNN is created from the current best CTRNN and mutated (Fig. 2[4]). Mutation involves considering each τ, w, θ and n value in the child, and replacing it with a random value in its range with a probability of $10/180 = 0.0556$. This ensures that on average, 10 mutations are incorporated into the child according to a normal distribution. It was found that for lower mutation rates runs tended to become mired in local optima. If the fitness of the child CTRNN is equal to or greater than the fitness of the current best CTRNN, it is replaced by the child; otherwise, the child is discarded (Fig. 2[5,6]).

After each possible replacement, the current CTRNN is considered in order to determine whether a failure condition has occurred, or whether it has achieved the success criteria. In the present work the failure condition is defined as 100 generations of the hill climber elapsing before a successful CTRNN is found. A successful CTRNN is defined as one for which, at some time step during the current evaluation (Fig. 2[20]) both claw tips touch the target object (Fig. 2[21,22]) and it is lifted far enough onto the robot's back such that the distance sensor there fires above a certain threshold (Fig. 2[23]).

If the failure condition occurs, the task environment is eased; if the current CTRNN succeeds, the task environment is made more difficult (Fig. 2[7-12]). Easing the task environment involves increasing the current evaluation period by 10 time steps. This has the effect of giving the robot more time to succeed at the current task if it fails. Making the task environment more difficult involves moving the target object 0.01 meters away from the front of the robot. This has the effect of teaching the robot to grasp and lift the target object when it is close, and learning to locomote toward the target object, followed by grasping and lifting it, when it is placed further away. As some CTRNNs that succeeded for a given target object distance also succeed when the object is moved further away, the object is continually moved until the current CTRNN fails, at which time hill climbing recommences (Fig. 2[10-12]). In order to further speed the algorithm an individual evaluation is terminated early if the robot ceases to move before succeeding at the task.

The overall success of a run is indicated by how many times a successful genome was found. That is, how far away the target object has been moved while still preserving a controller that can guide the robot to the object and also enable successful grasping and lifting.

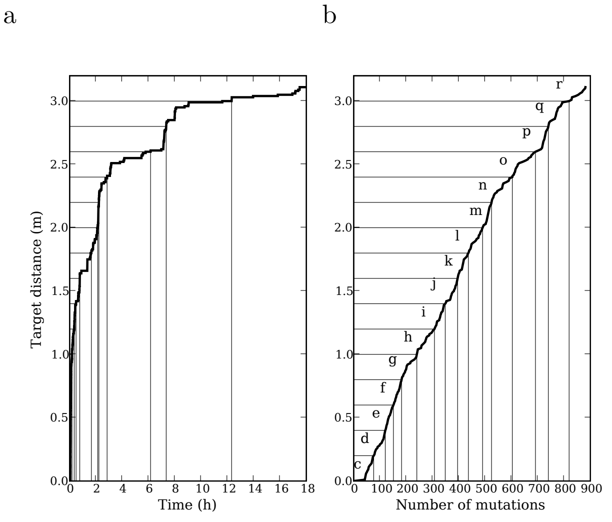
Results

A series of independent **runs** were conducted and are reported on here, where each run is conducted for 18 hours of CPU time. One set of runs were performed using the quadruped robot described above, and are henceforth referred to as **regime I**. Another set of runs were performed in which two additional, middle legs were added to the quadruped, resulting in a hexapod, referred to as **regime II**. The new legs are the same size and have the same orientation as the front legs. The CTRNN for the hexapod requires an additional four motor neurons: two for the middle upper legs and two for the middle lower legs. As all motor neurons are connected to one another, and the sensors are also connected to the additional motor neurons, this gives a total of $14 + 14 + 14 * 14 + 6 * 14 = 308$ evolvable parameters.

Within both regimes, a series of seven **trials** were performed: in the first trial the target object is initially placed directly in front of the robot ($d = 0.0$); in the second trial the object is initially placed 0.5 meters in front of the robot ($d = 0.5$), and so on in increments of 0.5 meters until in the seventh trial the object is initially placed 3.0 meters in front of the robot ($d = 3.0$). For each regime and each trial, 100 independent runs were performed, giving a total of $2*7*100 = 1400$ conducted runs.

Sample run Figure 1 illustrates two compound behaviors that evolved during one of the runs from regime I and trial 1 ($d = 0.0$). Fig. 1a-g illustrates the successful locomotion toward and lifting of the target object after it has been moved 2.8 meters from the robot. Fig. 1h-n illustrates the successful behavior from later in the run when the target object has been moved 3.0 meters from the robot. Both of these controllers are bistable in the sense that when the robot is far from the target object the distance sensors output low values, which push the controllers into a periodic attractor. This periodic attractor moves the robot's limbs in a cyclic pattern leading to locomotion toward the object (i.e. taxis behavior). When the robot nears the object either the high values of the distance sensors, the sudden firing of the touch sensors when the claw tips come in contact with the object, or a combination of both push the controller out of the cyclic attractor and into a point attractor in which the robot's claw lifts the target object onto its back and then the robot stops moving.

Fig. 3 reports the fitness progression of this particular run in more detail. As can be seen in Fig. 3a, a succession of successful CTRNNs allow the target oba b



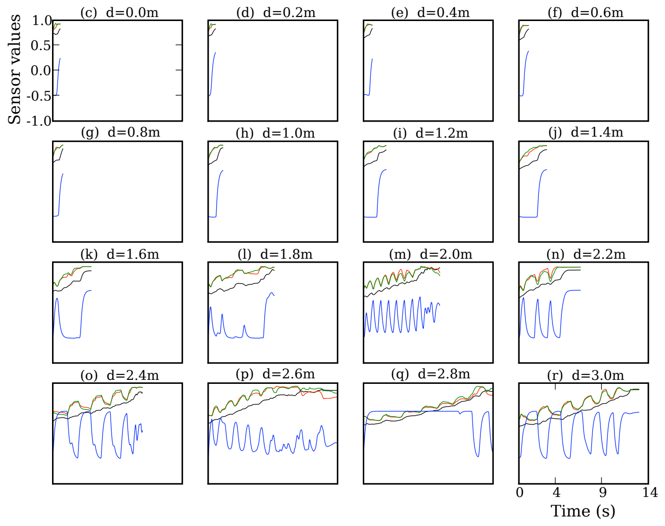


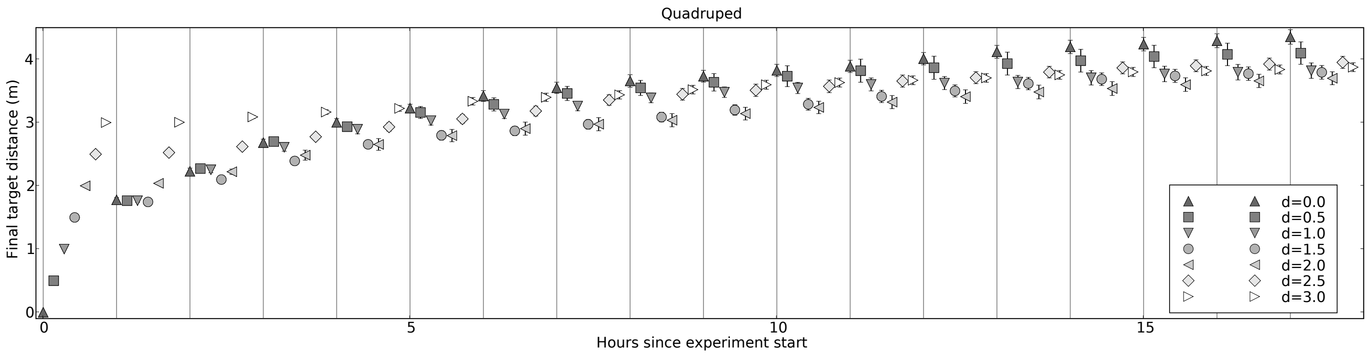
Figure 3: The fitness progression in a sample run. **a**: The fitnesses of the best controllers over the 18 CPU hours of the run (thick line). Thin lines indicate 16 successful controllers discovered when the target object was placed 0.0, 0.2, ... 3.0 meters from the robot. **b**: The same fitness progression is plotted as a function of the number of mutations that separate one successful controller from the next one. The same 16 successful controllers are indicated by the thin lines. **c-r**: Time series produced by the sensors when executing the 16 successful controllers (blue line=claw base rotation sensor; black line=distance sensor on the robot's back; red line=distance sensor on the left claw tip; green line=distance sensor on the right claw tip; the touch sensors are not shown).

ject to be moved just beyond 3.0 meters, at which time the 18 hours elapse. When the target object is much closer (in the first two hours of the run) there is a more rapid succession of successful controllers than when the target object is further out, which is to be expected for three reasons: (1) there are more behaviors that allow for successful lifting when the target object is close at hand than when it is further away; (2) bistability is not required when the target object is close (i.e. a ballistic behavior that blindly lifts the target object may succeed without requiring cyclic behavior beforehand to reach the target object); and (3) a number of failure conditions that occurred between discovery of successful controllers have extended the time period for an evaluation far beyond the initial 100 time steps.

This last factor is removed from consideration in Fig. 3b, in which the same fitness progression is plotted, but each improvement is measured as a function of the number of mutations that occur between the appearance of a fitter controller and its replacement in turn by a superior controller. The thin lines in Figs. 3a,b denote a selection of 16 successful controllers chosen from various stages in the run: when the target object is placed $[0.0, 0.2, \ldots, 3.0]$ meters from the robot. The behaviors in Fig. 1 correspond to the last two such controllers (d = 2.8m and d = 3.0m; Fig. 3q,r). Strikingly, a relatively constant number of mutations separate one successful controller from another: Fig. 3b indicates that for this particular run, an average of between 50 and 100 mutations separate the discovery of a successful controller when the target object is placed i meters away and the discovery of a successful controller when the target object is placed $i + 0.2$ meters away.

Figs. 3c-r report the time series sensor data for these 16 successful controllers. The blue line corresponds to the angle sensor in the claw base motor, and the other lines correspond to the three distance sensors. While the target object is still within reaching distance of the robot there is no evidence of cyclic activity in the controller (Figs. 3c-j), indicating that the dynamics of the controller are driven toward a point attractor that results in the target object being lifted onto the robot's back: the claw is held by the controller in a horizontal position for a short period (indicated by the low horizontal blue lines in Figs. 3e-j) before being rapidly rotated upward (the upward blue curves in Figs. 3c-j). After this point, when the target object is beyond 1.4 meters (Figs. 3k-r), the time series data from the sensors indicates cyclic activity within the controller. This indicates the discovery of controllers that are pushed into cyclic attractors which lead to rhythmic gaits that bring the robot to within reaching distance of the target object. It can be seen that among the bistable controllers, the cyclic attractors exhibit very different patterns: in Fig. 3l the attractor is hardly periodic; in Fig. 3m the frequency of oscillation is much higher than in the other controllers; and in Fig. 3q one part of the controller is saturated (the blue line maintains a constant, maximum value for most of the evaluation) while the other part is periodic.

a



b

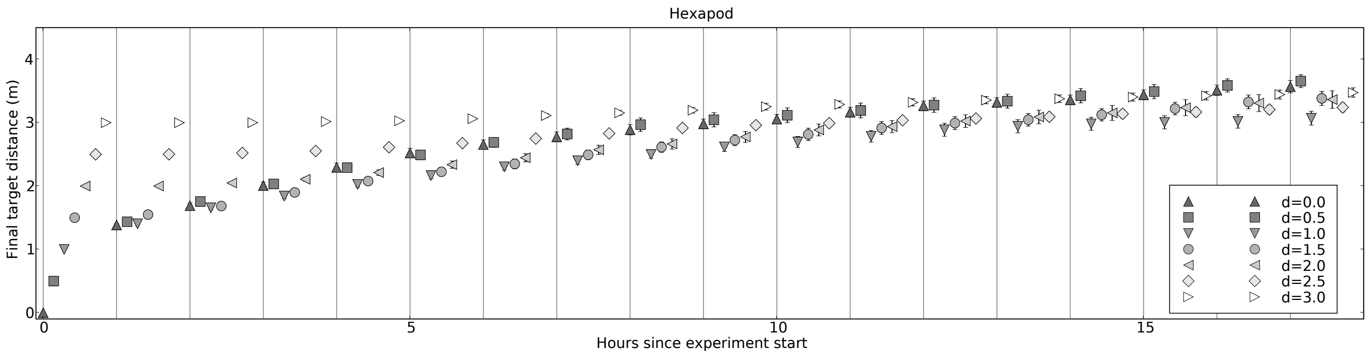


Figure 4: Evidence for behavior trajectory unidirectionality. **a:** The mean performance of the seven trials conducted using the quadruped robot (regime I; trial 1 = upward pointing triangle = initial target object distance is 0.0m from the robot [$d = 0.0$m]; trial 2 = square [$d = 0.5$m]; trial 3 = downward pointing triangle [$d = 1.0$m]; trial 4 = circle [$d = 1.5$m]; trial 5 = rightward pointing triangle [$d = 2.0$m]; trial 6 = diamond [$d = 2.5$m]; trial 7 = rightward pointing triangle [$d = 3.0$m]). **b:** Mean performance of the seven trials using the hexapod robot (regime II). Error bars indicate standard errors of the means ($n = 30$).

This last example behavior allows the robot to keep the claw in a vertical position while walking toward the target object (Figs. 1b,c), but when it nears the object the claw is rotated downward for grasping (Fig. 1d), upward for lifting (Figs. 1e,f) and finally downward to leave the object on its back (Fig. 1g). This example illustrates that under the right conditions the same monolithic controller can be partitioned into different components in which some parts together exhibit similar dynamics (in this case the oscillatory motor neurons involved in locomotion) while other components exhibit different dynamics (in this case the saturated values on the claw base motor keep it raised until it is needed for lifting). This partitioning, however, is evolved and may change over the course of an evolutionary run: when the target object is moved further out to 3.0 meters, behavior shifts such that the claw rotates upward and down in synchrony with the oscillations of the leg motor neurons (Fig. 3r).

Unidirectionality of behavior trajectories Fig. 4 reports the mean performances of the runs when using the quadruped and hexapod. The performance of an individual run is determined as the distance to which the target object was moved beyond the robot at the time of the run's termination: in other words the more successful controllers produced by a run, the further out the target object is moved.

Within both regimes, and within each trial, the 30 runs out of the 100 with the best performances at termination were extracted and the mean performance within that group was calculated at the beginning of the run (the leftmost groupings in Fig. 4), after the first hour (the second leftmost grouping in Fig. 4), and so on up to the mean performance achieved by the group after the 17th hour (rightmost grouping in Fig. 4). As can be seen, for the case of the quadruped the mean performance of the best runs of trial 1 are statistically significantly higher than the same set of runs extracted from trials 3 through 7 (the upward pointing triangle is significantly higher than the third through seventh markers in the rightmost grouping in Fig. 4a).

Also, although not quite significant, the mean performances for the hexapod are higher in trials 1 and 2 after hour 17 compared to the other trials (the heights of the first and second markers are higher than the third through seventh markers in the rightmost grouping of Fig. 4b).

Discussion

Figs. 1 and 3 demonstrate that by using the method introduced here it is possible to teach a robot to learn one dynamic behavior and gradually incorporate a second behavior into the same controller. The optimization process first discovers a controller which settles into a point attractor corresponding to grasping and lifting. This unistable controller then gradually evolves into a bistable controller which can also settle into a periodic attractor that corresponds to locomotion toward the object.

Fig. 3 indicates that during this process a succession of controllers are discovered with marked differences: the shapes and frequencies of the oscillations are quite different. In addition, there are controllers for which only part of the network displays oscillatory behavior, while the other is held at a saturation point until the robot nears the target object. This approach is attractive in that it may be more scalable than approaches that add new controller components for each new behavior (Brooks (1986), Calabretta et al. (2000), and Reil and Husbands (2002)). In these latter approaches the controller size grows linearly with the number of behaviors. In the proposed approach new controller structure may grow sub-linearly with the number of behaviors: new neurons and connections only need be added when the current monolithic controller can no longer incorporate an additional attractor. However, a more rigorous comparison between these approaches is warranted.

Fig. 4 justifies that scaffolding is necessary to achieve successful multistable controllers. In trials 1 and 2, the target object is initially placed close to the robot, forcing it to evolve a controller capable of grasping and lifting first; as the object is moved further out it incorporates locomotion. In trials 3 onward, the target object is initially placed further out, forcing the robot to learn locomotion first, followed by grasping and lifting. In general, among the best 30 runs these latter trials are less successful after 18 hours than the best 30 runs of trial 1: this difference is statistically significant for the quadruped, and marked yet not significant for the hexapod. This shows that there is an inherent unidirectionality in at least some behavioral trajectories: for a given set of behaviors it is easier to learn task i and then task j, compared to learning task j and then task i. Only the best 30 of the 100 runs were compared here, as some runs within all trials and all regimes failed to achieve controllers capable of both behaviors: future work is planned to increase the consistency of this approach.

Behavior chaining is a kind of robot shaping technique (Singh (1992), Dorigo and Colombetti (1994) and Saksida et al. (1997)), but in behavior chaining it is assumed that there is an *a priori* optimal ordering by which behaviors should be incorporated into the controller. Further, it assumes that this order is dictated by the agent, its task environment and the optimization process, and less by the agent's current behavioral competency.

For instance in (Goldenberg et al. (2004)) an agent is initially trained against a subset of environments, after which it is tested in unseen environments: the unseen environment in which the agent performs worst is then incorporated into the training set. It was shown that this can, in some cases, increase an agent's behavioral flexibility. However, the approach introduced here indicates that the order in which the agent is presented with environments affects the probability that an agent will be able to increase its behavioral flexibility. Consider an example: an agent undergoing shaping may perform very poorly in unseen environment i and less poorly on unseen environment j. The shaping schedule as described in Goldenberg et al. (2004) will incorporate environment i into the training set first. However, it may be that the agent should learn to behave successfully in environment j first, and will only then be able to behave successfully in environment i. The hypothetical shaping schedule described above may therefore fail to yield a behaviorally flexible agent. Future investigation will determine whether the optimal sequence in which behaviors should be learned can be predicted before learning begins, or whether it can be determined by the agent's current behavioral competency.

Conclusions

This paper has introduced a method that automatically trains a robot to exhibit a sequence of dynamic behaviors by drawing on evolutionary robotics, developmental psychology, and in particular on advances in embodied artificial intelligence (Pfeifer and Bongard (2006)) that equate specific behaviors with attractor states arising from the interaction of a robot's brain, body and environment, rather than the more subjective labeling of behaviors by an external observer. This method enabled a simulated robot to exhibit a compound behavior not yet reported in the literature: dynamic legged locomotion toward an object followed by grasping, lifting and holding of that object in a physically-realistic three-dimensional environment.

Automated methods such as evolutionary robotics are particularly well suited for domains where it is difficult for a human operator to translate a desired high-level behavior into a detailed sequence of motor commands. This is particularly true when the robot is capable of nonlinear behavior such as dynamic locomotion. However, this advantage has to date seemingly been counterbalanced by a corresponding drawback: scalability. That is, there is no known scalable method for gradually expanding the behavioral repertoire of an autonomous robot. This work suggests scalable behavior generation is possible if both the fitness function and a dynamic scaffolding schedule are carefully chosen. Rather than attempting to create a purely automatic method, this approach takes advantage of the natural ability of a human operator to break down a compound behavior (such as locomotion toward and then manipulation of a distal object) into separate behaviors (such as minimizing the distance to the object, grasping, and then lifting) each of which can then be sequentially mastered using automated optimization methods.

The operator's intuition is formalized by requiring them to determine what constitutes failure or success, and what modifications to the task environment should be made in either case. Future work is planned to determine just what failure and success definitions, and their associated scaffolds, are appropriate to realize robots capable of an increasing number of behaviors such as locomotion, object manipulation, object transport, locomotion over uneven terrain, and teamwork.

Source code The source code, data files and Python scripts for visualizing results are available at `www.cs.uvm.edu/~jbongard`.

References

Beer, R. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural Computation*, 18:3009–3051.

Beer, R. D. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 421–429.

Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Science*, 104(24):9943–9948.

Bongard, J., Zykov, V., and Lipson, H. (2006). Resilient machines through continuous self-modeling. *Science*, 314:1118–1121.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE J of Robotics and Automation*, 2(1):14–23.

Calabretta, R., Nolfi, S., Parisi, D., and Wagner, G. P. (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life*, 6(1):69–84.

Chella, A., Dindo, H., Matraxia, F., and Pirrone, R. (2007). Real-Time Visual Grasp Synthesis Using Genetic Algorithms and Neural Networks. *Lecture Notes in Computer Science*, 4733:567–578.

Dorigo, M. and Colombetti, M. (1994). Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 70(2):321–370.

Fernandez, J. and Walker, I. D. (1999). A Biologically Inspired Fitness Function for Robotic Grasping. *Genetic and Evolutionary Computation Conference*, pages 1517–1522.

Foss, J., Moss, F., and Milton, J. (1997). Noise, multistability, and delayed recurrent loops. *Physical Review E*, 55(4):4536–4543.

Gallagher, J. C. and Beer, R. D. (1999). Evolution and analysis of dynamical neural networks for agents integrating vision, locomotion and short-term memory. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 1273–1280.

Goldenberg, E., Garcowski, J., and Beer, R. D. (2004). May we have your attention: Analysis of a selective attention task. *Proceedings of the Eighth International Conference on Simulation of Adaptive Behavior*, pages 49–56.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the Sussex approach. *Robotics and Autonomous Systems*, 20(2-4):205–224.

Hornby, G., Takamura, S., Yamamoto, T., and Fujita, M. (2005). Autonomous evolution of dynamic gaits with two quadruped robots. *IEEE Transactions on Robotics*, 21(3):402–410.

Inamura, T., Toshima, I., Tanie, H., and Nakamura, Y. (2004). Embodied symbol emergence based on mimesis theory. *International Journal of Robotics Research*, 23(4):363–377.

Lee, W., Hallam, J., and Lund, H. (1998). Learning Complex Robot Behaviours by Evolutionary Computing with Task Decomposition. *Learning Robots: 6th European Workshop*, pages 155–172.

Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G. (2003). Developmental robotics: a survey. *Connection Science*, 15(4):151–190.

Nolfi, S. (1997). Evolving non-trivial behaviors on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22(3):187–198.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics*. MIT Press, Boston, MA.

Okada, M. and Nakamura, Y. (2004). Design of the continuous symbol space for the intelligent robots using the dynamics-based information processing. *Proceedings of the IEEE Intl. Conf. on Robotics and Automation*, pages 3201–3206.

Pfeifer, R. and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Pratt, J., Chew, C., Torres, A., Dilworth, P., and Pratt, G. (2001). Virtual Model Control: An Intuitive Approach for Bipedal Locomotion. *The International Journal of Robotics Research*, 20(2):129–143.

Reil, T. and Husbands, P. (2002). Evolution of central pattern generators for bipedal walking in a real-time physics environment. *IEEE Transactions on Evolutionary Computation*, 6(2):159–168.

Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Upper Saddle River, NJ.

Saksida, L. M., Raymond, S. M., and Touretzky, D. S. (1997). Shaping robot behavior using principles from instrumental conditioning. *Robotics and Autonomous Systems*, 22(3-4):231–249.

Saunders, J., Nehaniv, C. L., Dautenhahn, K., and Alissandrakis, A. (2007). Self-imitation and environmental scaffolding for robot teaching. *International Journal of Advanced Robotic Systems*, 4(1):109–124.

Singh, S. P. (1992). Transfer of learning across sequential tasks. *Machine Learning*, 8:323–339.

Wood, D., Bruner, J., and Ross, G. (1976). The role of tutoring in problem solving. *J Child Psychol Psychiatry*, 17(2):89–100.

Ziemke, T., Bergfeldt, N., Buason, G., Susi, T., and Svensson, H. (2004). Evolving cognitive scaffolding and environment adaptation: a new research direction for evolutionary robotics. *Connection Science*, 16(4):339–350.