Intelligent Computing & Cybernetics
Vol. 00, No. 00, DD Month 200x, 1–12

Influence of Temperature on Swarmbots That Learn

K.S.Patnaik †, J.F.Peters*‡, S. Anwar†

†Dept. of Computer Science and Engineering,
Birla Institute of Technology
Ranchi-835215, India
‡Department of Electrical and Computer Engineering,
University of Manitoba
Winnipeg, Manitoba R3T 5V6 Canada

(*Received 00 Month 200x; in final form 00 Month 200x*)

The problem considered in this paper is how a cybernetic system can learn to control its actions in a hostile environment. This paper focuses on an approach to solving this problem in an environment with varying temperatures. In effect, machines that operate outdoors have higher survivability if actions are chosen during periods when it is cooler (*e.g.*, nighttime or early morning rather than mid- to late afternoon during summer months). The assumption made here is that learning to choose actions that compensate for the influence of temperature has beneficial influence on the functioning of individuals in robot societies (collections of cooperating robots called swarmbots or swarms). In keeping with this idea, a biologically-inspired form of adaptive learning is given in this article. Conventional actor-critic learning provides a framework for the control strategy introduced in this article. It is ethology (study of behaviour of organisms) that provides a basis for monitoring the behaviour of a swarmbot. Individual behaviours together with sensor measurements are recorded in tables called ethograms. Swarm behaviour tends to be episodic. An ethogram is recorded during each episode during the lifespan of a swarm. Each ethogram is a source of measurements that can be used to influence learning during an episode. The contribution of this article is the introduction of a biologically-inspired approach to learning that adapts to changing temperatures.

Keywords: Adaptive learning, behaviour, biology, ethogram, robot, swarmbot, temperature.

1 Introduction

The problem considered in this paper is how a cybernetic system can learn to control its actions in a hostile environment. The particular form of cybernetic system we have in mind is a collection of cooperating robots called a swarmbot. Swarmbots learn by evaluating their actions. The solution to the problem of learning in a hostile environment is a refinement of the approach to guidance of a cybernetic system suggested in (11, 20). That is, adaptively learning to control system behaviour results from an evaluation of returns (cumulative rewards) from actions that are influenced by sensor measurements such as obstacle distances, ambient temperature, humidity, luminescence and electromagnetic field (emf) strength.

In various forms of adaptive learning, the choice of an action is usually based on estimates of the value of a state or the value of an action (see, *e.g.*, (22, 28, 18)). A swarm learns the best action to take in each state by maximizing a reward signal obtained from the environment. Two different forms of actor critic methods are considered in this article as a part of study of adaptive learning in real time by a swarm. First, a conventional actor critic method (22, 28, 29, 18, 15) is considered, where a critic evaluates whether things have gotten better or worse than expected as a result of an action selected in the previous state. A Temporal difference (TD) error term δ is computed by the critic to evaluate an action previously selected. An estimated preference $p(s, a)$ for action a in the current state s is then computed. Using Gibbs softmax method (6), action preference provides a basis for estimating the probability that an action will be chosen in the current state (22).

The approach to adaptive learning suggested in this article has its roots in ethology, the study of organism behaviour based on the pioneering work by Konrad Lorenz and Niko Tinbergen starting during the 1940s

*Corresponding author. Email: jfpeters@ee.umanitoba.ca

Intelligent Computing & Cybernetics
ISSN 0143-1161 print / ISSN 1366-5901 online
http://www.tandf.co.uk/journals
DOI: 10.1080/0143116YYxxxxxxxx

(see, *e.g.*, (10, 23, 24, 25, 26)). Briefly, one observes the behaviour of an organism in a systematic manner to learn the habits, proximate causes for actions, evolution and origin of organism behaviour. Observations are recorded in tables called ethograms. An ethogram is set of descriptions of characteristic behaviour patterns of a species (8, 17). In the study of swarm behavior of multiagent systems such as systems of cooperating robots, it is helpful to consider ethological methods [1,3], where each proximate cause (stimulus) usually has more than one possible response. Swarm actions with lower TD error tend to be favored. The second form of adaptive learning method represents a refinement of the actor-critic method is defined in context of environmental factor, *e.g.*, ambient temperature reading. The contribution of this article is the introduction of a biologically-inspired approach to learning that adapts to changing temperatures.

This article is organized as follows. An ethological prospective on system behavior is given in Sect. 2. A brief introduction to swarmbot testbed for a particular application is given in Sect. 3. The influence of temperature on the lifespan of a robot processor is given in Sect. 4. An adaptive learning algorithm based on the actor-critic method is given in Sect. 5. The results of experiments with two forms of learning are given in Sect. 6.

2 System Behavior: An Ethological Perspective

The origin was an outgrowth of the discovery movement patterns of organisms are homologous (10). Similarly, patterns of behavior can be observed in various forms of cybernetic systems that learn to respond to external stimuli and which evolve. *Cybernetics* has its origins in the study of control and communication in machines and animals based on feedback from the environment (30). In the study of cybernetic system behavior, one might ask why does a system behave the way it does? Tinbergen's four whys are helpful in the discovery of some of the features in the behavior of intelligent systems, namely, *proximate cause* (stimulus), *action response* together with the survival value, *evolution*, and *behavior ontogeny* (origin and development of a behavior) (26). Only proximate cause and action taken in response are considered in this paper. Tinbergen's survival value of a behavior correlates with reward that results from an action made in response to a proximate cause. The assumption made here is that action-preference is influenced by a reference or standard reward.

2.1 *What is a Swarmbot?*

A *swarmbot* (sbots) is a self-organizing robot colony composed of number of smaller robotic devices called bots (2, 13). An sbot performs basic tasks such as autonomous navigation, inspection and mapping of the environment, and grasping objects. In addition to these tasks, sbots communicate with each other and physically connect themselves together in flexible ways.

2.2 *Ethograms for Swarmbot Behavior*

Learning by a swarmbot tends to be episodic, where episodes vary in length depending on the success of an action strategy derived before the beginning of each episode. An action strategy results from an evaluation of bot behaviour during an episode. This evaluation is made possible using an ethological approach, where the measurements associated with each state are recorded in a table called a rough ethogram (17, 19), a tabulation of observed behaviours of an organism.

By way of illustration, consider Table 1 containing a small sample of bot behaviours during an episode. A behaviour can be represented by a tuple

$$(s, a, p(s, a), r, temp, d)$$

where $s, a, p(s, a), r, d$ denote organism functions representing state, action, action preference in a state, reward for an action, current bot processor temperature in ${}^0C$, and a decision about a possible action (d = 1 (accept) and d = 0 (reject)), respectively. A reward r is observed in state s and results from an action

Table 1. Sample ethogram

x_i	s	a	$p(s,a)$	r	$temp$	d
x_0	0	1	0.1	0.75	32	1
x_1	0	2	0.1	0.75	38	0
x_2	1	2	0.05	0.1	40	0
x_3	1	3	0.056	0.1	30	1
x_4	0	1	0.03	0.75	32	1
x_5	0	2	0.02	0.75	45	0
x_6	1	2	0.01	0.9	28	1
x_7	1	3	0.025	0.9	42	0

a performed in the previous state. The preferred action a in state s is calculated using

$$p(s,a) \leftarrow p(s,a) + \beta\delta(r,s),$$

where β is the actor's learning rate and $\delta(r,s)$ is used to evaluate the quality of action a (see (15)). For many applications, the behaviour of a bot that learns will be influenced by the changing temperature of its processor during an episode (this observation provides a basis for a new model for action preference in Alg. 3).

3 Swarmbot Behaviour

This section introduces a testbed for swarmbots designed by Christopher Henry (7). This testbed models as an artificial ecosystem containing swarms of cooperating bots (sbots) in an environment containing sequences of hydro-electric towers which require inspection (see, *e.g.*, Fig. 1). The bots crawl along a tower sky wire strung between the tops of power transmission towers. These bots cooperate to inspect power system equipments (e.g., insulators and towers). Two or more cooperating bots form a swarm-bot. Bots are dependent on sunlight to recharge their solar cells. The ecosystem also models many bot-threatening hazards such as high winds and lightning. Episodes (random in duration in the interval [2s, 4s] with 200 ms increments), have been introduced in the ecosystem to break up learning into intervals, and make it possible for a swarm to "reflect" on what it has learned. Swarm behavior during each episode is recorded in an decision table called an ethograms, which records swarm states, Proximate causes, responses (actions), action preferences, rewards and decisions (actions chosen and actions rejected). The focus of the ecosystem is on swarm activity and swarm learning. At all times, a swarm follows a policy that maps perceived states of the environment to actions. The goal of the learning algorithms is to find an optimal policy in a non-stationary environment.

3.1 *Henry Swarmbot Testbed*

The sbot testbed makes it possible to experiments with on-line learning by bots that must cooperate in a number of ways (e.g, navigate past an obstacle). A prototype for a testbed that automates the production of ethograms that provide a record of observed swarmbot behaviour patterns is briefly introduced in this section. The basic parts of the ecosystem operation in (see Fig. 1), namely, bot (tiny disk), bot charge (number next to bot), swarm (cluster of tiny disks), and average swarm charge (number next to cluster of tiny disks), sensor range (radiating circles surrounding agents in a swarm),adversary (larger solid disk inside circle), power line (line) and power tower (solid square connected to power line). The testbed provides output for an ethogram for each swarm. This output is in the form a tuple $(s, a, p(s,a), r)$ indicating current state s, an action a chosen in the previous state, action preference $p(s,a)$, and reword signal r resulting from action a.

The sample snapshot of the individual and collective behavior of inspect bots in a line-crawling swarmbot testbed is part of a Manitoba Hydro Line-Crawling Robot Research project. The design of a line-crawling bot (called ALiCE II, *i.e.*, Automated Line Crawling Equipment, version II) was completed by Daniel Lockery in 2007 (9). An artist's view of ALiCE II is shown in Fig. 2. This 3D view of ALiCE II was created

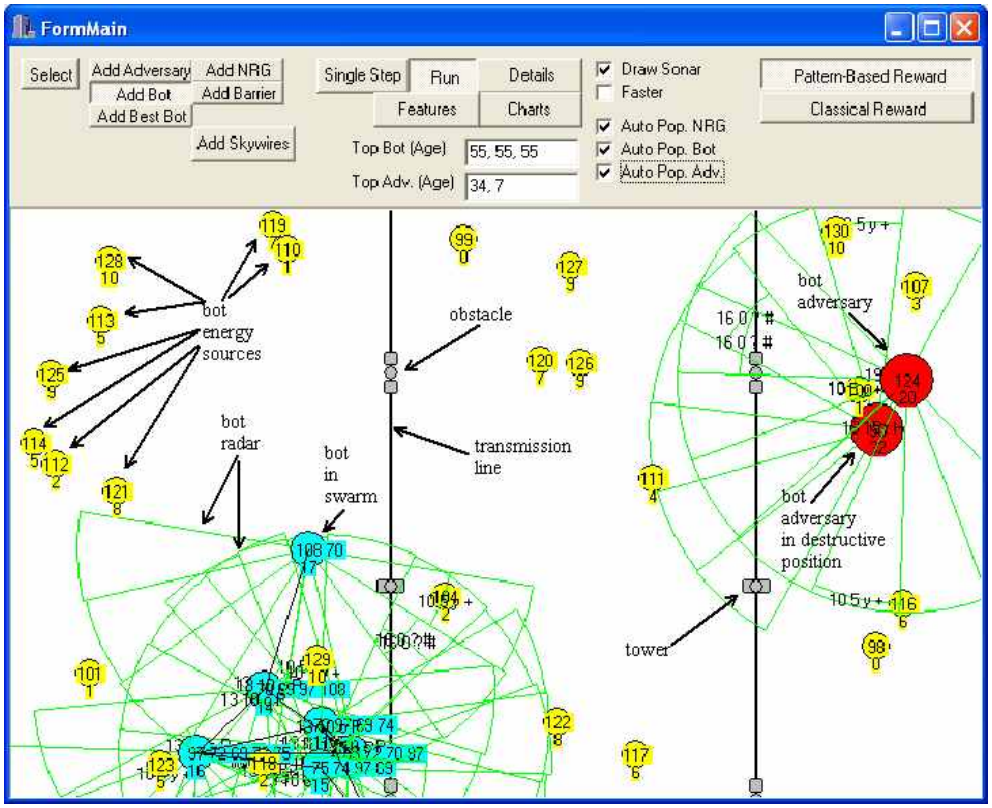


Figure 1. Sample Simulated Swarmbot Behaviour

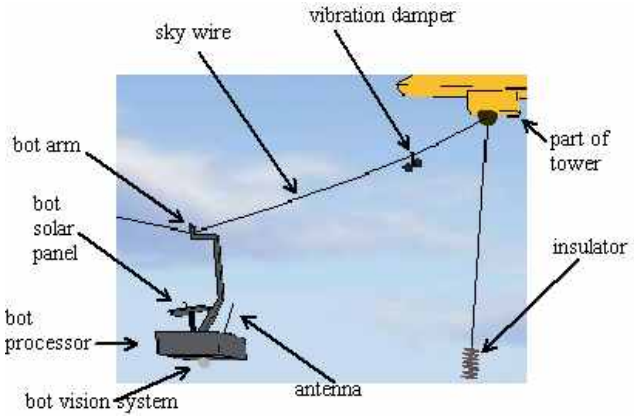


Figure 2. Artist's View of Lockery Line-Crawling Bot

by Maciej Borkowski (3) using OpenGL.A sketch of the methodology implemented by the ecosystem shown in Fig. 1 is given in Alg. 1.

Cooperation between bots is one of the hallmarks of swarmbot behaviour. For example, a swarm will work together to climb a power system tower, crawl around and over obstacles, and rescue stalled bots that require pushing or pulling during navigation over an irregular terrain. Many of the details concerning swarmbots have been already been reported (see, *e.g.*, (13, 14, 16, 17)). A more detailed consideration of the swarmbot testbed is outside the scope of this paper. Instead, this paper focuses the influence of temperature on the behaviour of a bot that learns to cope with its environment.

Algorithm 1: Ecosystem Operation

```
Input  : Bots, Adversaries,Skywires, threshold th.
Output: Ethogram for Swarmbot Behaviour
while (true) do
    start ecosystem;
    select add bot, adversary, NRG battery charger, barrier, skywires ;
    select reward;
    while (true) do
        set environment variables (e.g., lightning, wind, temperature, emf) ;
        take action a ;
        observe state s resulting from action a ;
        compute value of state V(s) ;
        if V(s) < th then
            end episode ;
        else
            adjust bot positions, energy, environment variables ;
            remove dormant, abandoned bots ;
        end
    end
end
```

4 Swarmbot Temperature

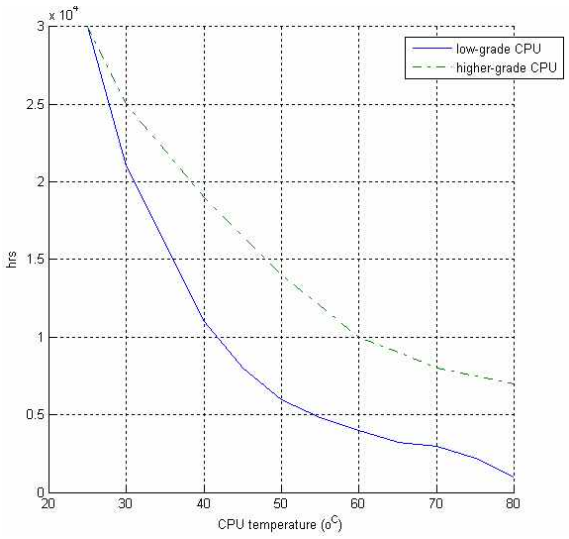


Figure 3. Sample Bot Processor Lifespans

Temperature is the major environmental factor that influences the choice of a particular action by swarm bot. CUP heat has a direct impact on CPU life expectancy. Apart from internal sources of CPU heat (*e.g.*, increased bus speed, increased voltage), there are environmental sources of heat (*e.g.*, solar heat with wind, solar heat without wind). Whenever CPU operating temperatures cross a certain boundary value, a swarmbot's functioning is affected. Swarmbots function well within certain temperature intervals. The influence of heating has an impact on all integrated circuits. Temperature has direct effect on its processor and sensors. Temperature reduces processor life and degrades the performance of sensors due to the thermal effects. From experiments, the estimated life expectancy of two different processors is shown in Fig. 3. These experimental results are similar to those reported by Citarella (5).

Let $temp_{new}, temp_{normal}$ denote the current ambient temperature reading and normal operating temper-

ature for a particular bot processor. We use $temp_{new}/temp_{normal}$ to estimate the life expectancy of a bot CPU (5). A swarmbot's life expectancy is estimated using the average value of this ratio for bots contained in a swarm. Experiments have shown that a bot's processor life and temperature are inversely related, *i.e.*, the higher the temperature, the lower a bot's Life. This holds true for all integrated circuits that are used in bots. For a more detailed study of the influence of ambient temperature on CPU performance, see, *e.g.*, (4).

For example, in cases where the temperature of a bot processor starts at 60^0C and increases to 70^0C, one can expect a corresponding decrease in processor life (see, *e.g.*, Fig. 3). We can use the influence of temperature on swarmbot behaviour as a factor in determining action preferences (*e.g.*, parking, moving to shaded area) in actor-critic learning.

5 Actor-critic methods

Actor-critic (AC) methods are temporal difference (TD) learning methods with a separate memory structure to represent policy independent of the value function used (see Fig. 4). The AC method considered in this section is an extension of reinforcement comparison in (22). Let $S, s, A(s), a, s', r$ denote a set of possible states, current state, set of actions available in state s, bot action, subsequent state after action a, and reward signal from the environment after an action is performed, respectively. Except for the ethogram, the flow diagram in Fig. 4 represents the basic framework of the actor-critic method.

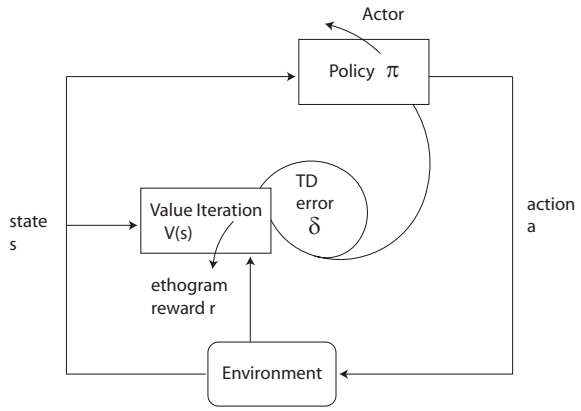


Figure 4. Ethogram-Based Actor-critic Learning

The actor-critic method is represented by Alg. 2. This method begins by initializing $\gamma \in (0, 1]$ (*discount rate*), a number that diminishes the estimated value of the next state. In a sense, γ captures the confidence in the expected value of the next state. Let $C(s)$ denote the number of times the actor has observed state s. As is common (*e.g.*, see (22, 27)), define the estimated value function $V(s)$ to be the average of the rewards received while in state s. This average may be calculated by

$$V(s) = \frac{C(s) - 1}{C(s)} V_{C(s)-1}(s) + \frac{1}{C(s)} \cdot r, \tag{1}$$

where $V_{C(s)-1}(s)$ denotes $V(s)$ for the previous occurrence of state s. After each action selection, the critic evaluates the quality of the selected action using

$$\delta \longleftarrow r + \gamma V(s') - V(s),$$

which is the error (labelled the TD error) between successive estimates of the expected value of a state. If $\delta > 0$, then it can be said that the expected return received from taking action a at time t is larger than

the expected return in state s resulting in an increase in action preference $p(s,a)$. Conversely, if $\delta < 0$, the action a produced a return that is worse than expected and $p(s,a)$ is decreased (29).

The preferred action a in state s is calculated using

$$p(s,a) \leftarrow p(s,a) + \beta\delta,$$

where β is the actor's learning rate. The policy $\pi(s,a)$ is employed by an actor to choose actions stochastically using the Gibbs softmax method (2) (6) (see also (22))

$$\pi(s,a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}. \tag{2}$$

It is assumed that the behaviour represented by Alg. 2 is episodic (with length T_m, an abuse of notation used in (21) for terminal state, the last state in an episode). T_m is important because it provides a stopping mechanism for each episode (see, *e.g.*, (15)). For best results, T_m (episode length) is varied randomly. The while loop in the algorithm is executed continually over the entire learning period, not just for a fixed number of episodes.

Algorithm 2: Actor-critic Method

Input : States $s \in S$, Actions $a \in A$, Initialized γ, β.
Output: Policy $\pi(s,a)$.
for $(all\ \ s \in S, a \in A(s))$ **do**
  $p(s,a) \longleftarrow 0$; $\pi(s,a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$; $C(s) \longleftarrow 0$;
end
while *True* **do**
  Initialize s_t, T_m (randomly);
  for $(t = 0; t < T_m; t = t + 1)$ **do**
    Choose a from $s = s_t$ using $\pi(s,a)$;
    Take action a, observe r, s';
    $C(s) \longleftarrow C(s) + 1$;
    $V(s) \longleftarrow \frac{C(s)-1}{C(s)} V(s) + \frac{1}{C(s)} \cdot r$;
    $\delta = r + \gamma V(s') - V(s)$;
    $p(s,a) \longleftarrow p(s,a) + \beta\delta$;
    $\pi(s,a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
    $s \longleftarrow s'$;
  end
end

5.1 *Temperature-Based Actor-Critic Method*

This section introduces what is known as Temperature Actor-Critic (TAC) method. The preceding section is just an example of Actor-Critic methods[7,8]. The basic approach is to vary the amount of credit assigned to action taken. The assumption made here is that learning by a bot is influenced by temperature change resulting from various sources of heating (sun, lightning strikes on a power transmission line) and cooling (wind chill, winter conditions) in the environment. This idea is reflected in a change in computing action preferences in the actor-critic method. We start by defining a temperature adjustment factor (TAF).

Let $\overline{temp}$ denote the average bot processor (CPU) temperature during an episode. During each episode,

let *temp* vary randomly over a reasonable interval (*e.g.*, *temp* varying over $[60, 75]$) and compute

$$TAF = \frac{\overline{temp}}{temp_{normal}},$$

The running average $\overline{temp}$ can be computed using

$$\overline{temp} \longleftarrow \frac{C(s)-1}{C(s)} \cdot \overline{temp} + \frac{1}{C(s)} \cdot temp,$$

where *temp* is the CPU temperature reading in the current state. Also, during episode 2 and each succeeding episode, compute a new value for $\overline{temp}$ (for the average temperature during an episode). Then use the new value of TAF during the next episode. This leads to a new version of the action preference $p(s, a)$ model, namely,

$$p(s, a) \longleftarrow p(s, a) + TAF \cdot \delta,$$

where TAF takes the place of β in Alg 2. From this, we obtain Alg. 3.

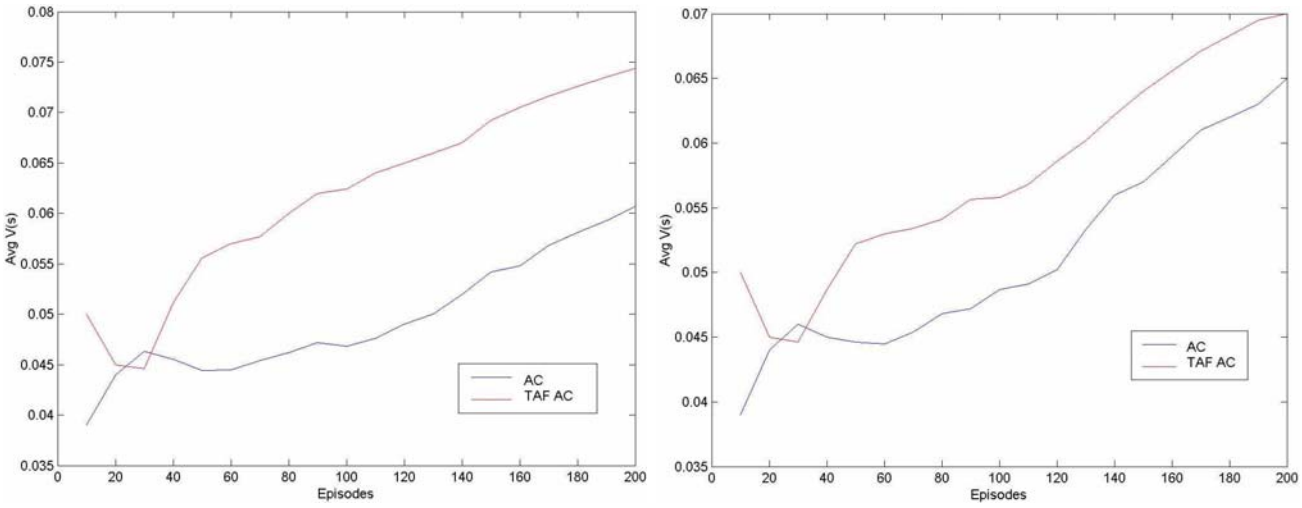
Algorithm 3: Temperature-Based Actor Critic Method

Input : States $s \in S$, Actions $a \in A$, Initialized γ, $temp_{normal}$, ethogram table Tab.
Output: Policy $\pi(s, a)$.
for $(all \;\; s \in S, a \in A(s))$ **do**
  $p(s, a) \longleftarrow 0$; $\pi(s, a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$; $C(s) \longleftarrow 0$;
end
while *True* **do**
  Initialize T_m (randomly), $\overline{temp}$ (from ethogram Tab);
  for $(t = 0; t < T_m; t = t + 1)$ **do**
    Choose a from $s = s_t$ using $\pi(s, a)$;
    Take action a, observe $r, s', temp$;
    $C(s) \longleftarrow C(s) + 1$;
    $\overline{temp} \longleftarrow \frac{C(s)-1}{C(s)} \cdot \overline{temp} + \frac{1}{C(s)} \cdot temp$;
    $TAF = \frac{\overline{temp}}{temp_{normal}}$;
    $V(s) \longleftarrow \frac{C(s)-1}{C(s)} V(s) + \frac{1}{C(s)} \cdot r$;
    $\delta = r + \gamma V(s') - V(s)$;
    $p(s, a) \longleftarrow p(s, a) + TAF \cdot \delta$;
    $\pi(s, a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A(s)|} e^{p(s,b)}}$;
    Store beh $(s, a, p(s, a), r, temp)$ in Tab;
  end
end

Notice that functioning of Alg. 3 depends on the construction of an ethogram during each episode. Each ethogram makes it possible to compute the average temperature $\overline{temp}$ during an episode. This, in turn, will influence episodic action preference and the action policy during.

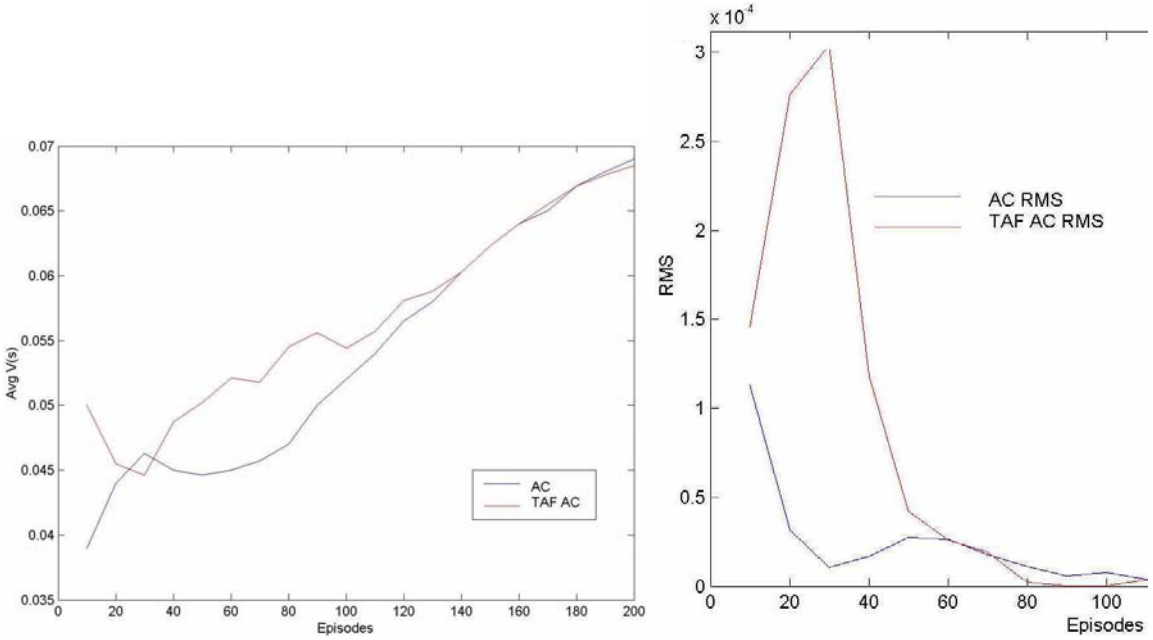
6 Results

A comparison of the sample episodic learning of a swarmbot based on implementations of AC Alg. 2 (actor critic method) and TAC Alg. 3 (learning based on ambient temperature readings). This comparison is viewed in two ways: average value of state values and root mean mean square (RMS) error values.



(a) AC vs. TAC V(s),$\gamma = 0.1$ (b) AC vs. TAC,$\gamma = 0.5$

Figure 5. Episodic Values of State



(a) AC vs. TAC V(s),$\gamma = 0.9$ (b) AC vs. TAC RMS,$\gamma = 0.1$

Figure 6. Episodic V(s) and RMS Values

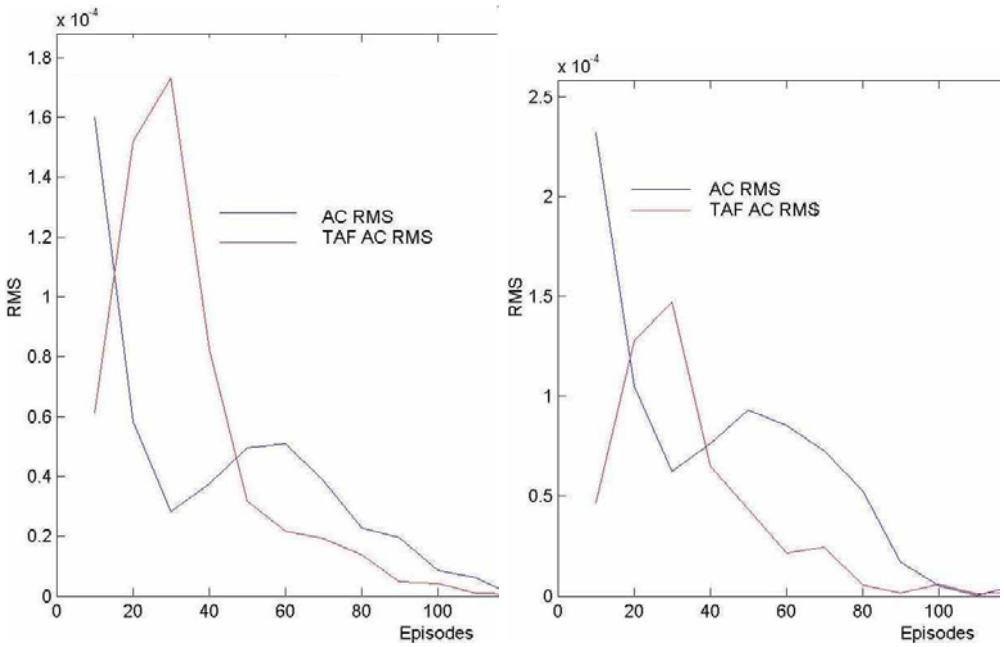
First, for varying values of γ (discount factor on the value of a state $V(s)$), average $V(s)$ values are plotted over a number of episodes. In each of the experiments reported here, average $V(s)$ values were recorded during 200 episodes during which a collection of cooperating bots (swarmbot) carried out its particular mission. In this study, swarmbots were programmed to inspect their environment and avoid sources of threat such as high heat (*e.g.*, lightning strikes). At $\gamma = 0.1$ and $\gamma = 0.5$, the temperature-based TAC Alg. 3 did consistently better than the standard actor critic method (AC Alg. 2) as shown in Fig. 5(a) and Fig. 5(b). With $\gamma = 0.9$, both methods do well (see Fig. 6(a)). This suggests that in the case where there is a more rapid decline in δ values (this is case for lower values of γ), the TAC learning method should be chosen. In more stable environments, both the AC and TAC methods do well.

A record of RMS values during successive episodes in the lifespan of a swarmbot provides a second basis for comparison of the two forms of actor critic learning. Episodic RMS (also known as root mean squared

error (12)) was computed using

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[\overline{V(s)} - V(s_i)\right]^2},$$

where $\overline{V(s)}$ is a running average computed during each episode and $V(s_i)$ is value of the i^{th} state during an episode. From Fig 6(b) and Fig. 7(a), it can be observed that the temperature-based TAC Alg. 3 is more suitable for mid-range values of the discount factor γ, *i.e.*, where the learning algorithm eventually stabilizes in an environment where there is some instability in CPU temperatures. The same phenomenon can be observed in the RMS values plotted in Fig. 7(b) for $\gamma = 0.9$. From what has been observed up to this point, it can be concluded that the temperature-based actor critic algorithm is a better choice as a controller in environments where there is significant temperature variation. From a safety point-of-view in cases where swarmbots must operate in hostile environments such as inspection of power system equipment, the TAC learning method is recommended.



(a) AC vs. TAC RMS,$\gamma = 0.5$ (b) AC vs. TAC RMS,$\gamma = 0.9$

Figure 7. Episodic RMS Values

7 Conclusion

This article considers the influence of temperature on learning by swarmbots that operate in hostile environments. This has led to a variation of the conventional actor-critic method, where action preference is influenced by variations in temperature. A biologically-inspired approach to learning is also considered in this paper. Basically, this means that observed organism behaviors are recorded in tables called ethograms. Each new episode in the lifespan of a swarmbot results in a new ethogram. An ethogram recorded during the current episode provides a basis for influencing learning in the next episode. This article has been limited to showing the influence of temperature on learning. In fact, the TAC algorithm (Alg 3) is part of a family of learning methods that reflect the influence of environmental factors such as temperature, wind, humidity, and radiation. Consideration of other environmental influences on learning are part of our future work.

Acknowledgements

The author gratefully acknowledges the suggestions and insights by Christopher Henry, Dan Lockery and David Gunderson concerning topics in this paper. This research has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) grant 185986.

References

[1] Berenji, H.R. 2003. A convergent actor–critic-based FRL algorithm with application to power management of wireless transmitters, *IEEE Trans. on Fuzzy Systems* 11 (4): 478-485.

[2] Bonabeau, E., Dorigo, M., and Theraulaz, G.. 1999. *Swarm Intelligence. From Natural to Artificial Systems*, UK: Oxford University Press.

[3] Borkowski, M. 2007. *2D to 3D Conversion with Direct Geometric Search and Approximation Spaces*. Ph.D. Thesis, Supervisor: J.F. Peters, Electrical & Computer Engineering, University of Manitoba.

[4] Chen, C.-T., Wu, C.-K., and Hwang, C. 2006. Optimal design and control of CPU heat sink processes. In: *International Control Conference* (ICC2006): 1-6. University of Strathclyde and University of Glasgow.

[5] Citarella, J.: CPU temperature range, `http://www.overclockers.com/tips30/`

[6] Gibbs, J.W. 1960. *Elementary Principles in Statistical Mechanics*. NY: Dover Publications, Inc.

[7] Henry, C. 2006. *Reinforcement Learning in Biologically-Inspired Collective Robotics: A Rough Set Approach*. M.Sc. Thesis, Supervisor: J.F. Peters, Electrical and Computer Engineering, University of Manitoba.

[8] Lehner, P.N. 1996. *Handbook of ethological methods*. UK: Cambridge University Press.

[9] Lockery, D. 2007. *Learning with ALiCE II*. M.Sc. Thesis, Supervisor: J.F. Peters, Electrical and Computer Engineering, University of Manitoba.

[10] Lorenz, K.Z. 1981. *The Foundations of Ethology*. NY, Wien: Springer-Verlag.

[11] Maravall, D., Mazo, M., Palencia, V., Pérez, M.M., and Torres, C. 1990. Guidance of an autonomous vehicle by visual feedback, *Cybernetics and Systems* 21 (2-3): 257-266.

[12] Mood, A.M., Graybill, F.A. 1963. *Introduction to the Theory of Statistics*. NY: McGraw-Hill.

[13] Peters, J.F., and Ramanna, S. 2004. Hierarchical behavioral model of a swarmbot. In *Methods of Artificial Intelligence* (AIMETH04), edited by T. Ruczynski, W. Cholewa, W. Moczulski, 105-106.

[14] Peters, J.F., Borkowski, M., Henry, C., and Lockery, D. 2008. Monocular vision system that learns with approximation spaces. In *Rough Computing: Theories, Technologies, and Applications* edited by A.E. Hassaien, Z. Suraj, D. Ślęzak, and P. Lingras, 186-203. Hershey, NY: Information Science Reference.

[15] J.F. Peters, C. Henry, and D.S. Gunderson. 2006. Biologically-inspired approximate adaptive learning control strategies: A rough set approach. *International Journal of Hybrid Intelligent Systems* 3: 1-14.

[16] Peters, J.F., Borkowski, M., Henry, C., Lockery, D., Gunderson, D., and Ramanna, S. 2006. Line-Crawling Bots That Inspect Electric Power Transmission Line Equipment. In *Proc. 3rd Int. Conf. on Autonomous Robots and Agents* (ICARA 2006), 39-44.

[17] Peters, J.F. 2005. Rough ethology: Toward a Biologically-Inspired Study of Collective behaviour in Intelligent Systems with Approximation Spaces. *Transactions on Rough Sets* III, Springer LNCS 3400, 153-174.

[18] Peters, J.F., and Henry, C. 2006. Reinforcement learning with approximation spaces, *Fundamenta Informaticae* 71 (2-3), 323-349.

[19] Peters, J.F., Henry, C., and Ramanna, S. 2005. Rough Ethograms: Study of Intelligent System behaviour. In *New Trends in Intelligent Information Processing and Web Mining (IIS05)*, edited by M.A. Kłopotek, S. Wierzchoń, K. Trojanowski, 117-126.

[20] Peters, J.F., and Ramanna, S. 1991. Modeling timed behavior in real-time systems with temporal logic. *Cybernetics and Systems* 22 (5), 583-608.

[21] Precup, D., Sutton, R. S., Paduraru, C., Koop, A., and Singh, S. 2006. Off-policy with recognizers. In *Advances in Neural Information Processing Systems*, 1-8.

[22] Sutton, R.S., and Barto, A.G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press.
[23] Tinbergen, N. 1948. Social releasers and the experimental method required for their study. *Wilson Bull* 160, 6-52.
[24] Tinbergen, N. 1951. *Study of Instinct*. UK: Oxford University Press.
[25] Tinbergen, N. 1953. *The Herring Gull's World. A Study of the Social Behavior of Birds*. London: Collins.
[26] Tinbergen, N. 1963. On aims and methods of ethology. *Zeitschrift für Tierpsychologie* 20, 410-433.
[27] Watkins, C.J.C.H. 1989. *Learning from Delayed Rewards*. Ph.D. Thesis, supervisor: Richard Young. King's College, Cambridge University.
[28] Watkins, C.J.C.H., and Dayan, P. 2003. Reinforcement learning. *Encyclopedia of Cognitive Science*. UK: Macmillan.
[29] Wawrzyński, P. 2005. *Intensive Reinforcement Learning*. Ph.D. dissertation, supervisor: Andrzej Pacut, Institute of Control and Computational Engineering, Warsaw University of Technology.
[30] Wiener, N. 1948. *Cybernetics: or Control and Communication in the Animal and the Machine*. Cambridge, MA: The MIT Press.