Time Delays in a HyperNEAT Network to Improve Gait Learning for Legged Robots

Oscar Silva, Pascal Sigel and María-José Escobar*
Department of Electronic Engineering, Universidad Técnica Federico Santa María
Avenida España 1680, 2390123 Valparaíso, Chile.
**Corresponding author: mariajose.escobar@usm.cl*

Abstract—**Gait generation for legged robots is an important task to allow an appropriate displacement in different scenarios. The classical manner to generate gaits involves hand-tuning design generating high computational and time efforts. Neuroevolution algorithms with the ability to learn network topologies, such as, Neuroevolution of Augmenting Topologies (NEAT) and Hypercube-based NEAT (HyperNEAT), have been used in the computational community to learn gaits in legged robots. Recently, a new version of NEAT, called τ-NEAT, has been reported including a time delay in the connectivity between neurons, values that are also learned by the underlying genetic algorithm. Extending this idea, we included time delays in the HyperNEAT implementation (τ-HyperNEAT) making the algorithm capture time-series variations that could be important for gait generation. Using a four-legged robot platform (Quadratot) and a fitness function with two objectives, we compared the performance of HyperNEAT versus τ-HyperNEAT for the learning gait task. The comparative analysis of the results reveals that quantitative performance variables showed no differences between HyperNEAT and τ-HyperNEAT. The difference between the two approaches appears in the non-quantitative observation of the generated gaits: τ-HyperNEAT outperforms HyperNEAT generating more coordinated, realistic and natural gaits.**

1. Introduction

In robotics, to program gaits in legged robots is a challenging task which normally depends on the robot physical morphology, and most of the times, a kinematic model is needed [2], [3], [4], [5].

Neuroevolution algorithms, which use genetic algorithms to train artificial neural networks (ANN), have been proposed as learning rules to set up connection weights in a network with fixed topology [6]. Furthermore, Neuroevolution of augmenting topologies (NEAT) algorithm [1] has been proposed as a solution when not only the connections weights of an ANN need to be found, but also, its topology. This powerful tool has generated several application in automatic feature selection [7], video games [8], [9], mobile *ad-hoc* networks [10], robotics [11], among others. In general, NEAT algorithm encodes the network topology and connection weights in a *genome*, which evolves using genetic algorithm rules to create new generations of ANNs in order to match a fitness function.

Neuroevolutive techniques present here an interesting approach to automatically learn gaits in legged robots without needing to *hard-code* the gait algorithm. Aligned with this objective, NEAT algorithm was extended to HyperNEAT in order to include system regularities or symmetries, that can be used to benefit the learning process [12], [13]. Under this new perspective, HyperNEAT algorithm has been used to automatically learn gaits in legged-robots [14], [15], [16].

Even if methods to automatically learn gaits in legged-robots have been proposed, most of the reported results are not easy to reproduce, or, the gait learned lacks harmony which is always present in gaits observed in nature. Several reasons motivate the search of more naturalistic, harmonic and symmetric gait generation ranging from the understanding of evolutionary rules followed by nature up to create more adaptive systems to interact with humans, for instance in rehabilitation robotics [17].

Using the concepts developed by NEAT and HyperNEAT, a recent work related to the inclusion of time delays between connections in NEAT (τ-NEAT, [18], [19]) and the synaptic connections proposed by [20], we propose a τ-HyperNEAT algorithm for learning gaits with harmonic and coordinated movements in legged-robots. Our τ-HyperNEAT is formed by two networks: the main substrate network where physical regularities are included, and, a secondary NEAT network to compute the connection weights, and most importantly, the time delay between nodes in the the substrate. The performance of each network is evaluated under a multi-objective fitness also proposed in this article. As we presented in our results, τ-HyperNEAT generates more harmonic and symmetric gaits than HyperNEAT exhibiting the relevance of this contribution.

2. Background

Here we briefly introduce the concepts needed to implement a τ-HyperNEAT algorithm for learning gaits in legged robots.

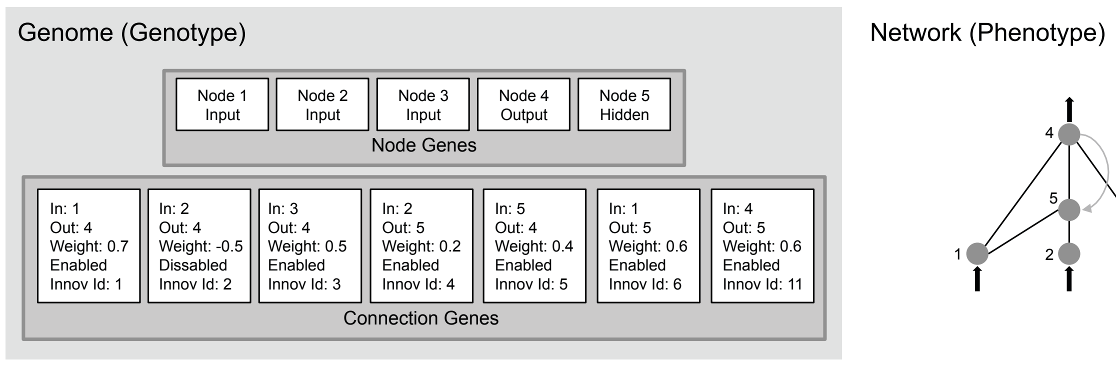

Genome (Genotype)
Network (Phenotype)
Node 1 Input
Node 2 Input
Node 3 Input
Node 4 Output
Node 5 Hidden
Node Genes
In: 1 Out: 4 Weight: 0.7 Enabled Innov Id: 1
In: 2 Out: 4 Weight: -0.5 Dissabled Innov Id: 2
In: 3 Out: 4 Weight: 0.5 Enabled Innov Id: 3
In: 2 Out: 5 Weight: 0.2 Enabled Innov Id: 4
In: 5 Out: 4 Weight: 0.4 Enabled Innov Id: 5
In: 1 Out: 5 Weight: 0.6 Enabled Innov Id: 6
In: 4 Out: 5 Weight: 0.6 Enabled Innov Id: 11
Connection Genes
4
5
1
2
3

Figure 1. Genome example used in NEAT to encode the neural network shown on the right. Image adapted from [1].

2.1. NEAT

NeuroEvolution of Augmenting Topologies (NEAT) [1] uses genetic algorithms to evolve the topology and connection weights of a neural network according to a fitness function. Because its genetic algorithm basis, NEAT evolves following the law that outstanding individuals (large fitness) have a higher probability to create a new generation of individuals, and thus, preserve the specie.

One of the main advantages of NEAT, compared to other types of neural networks, is that the initial network structure (topology) is not needed, but, it is found along the learning process thanks to the neuro-evolution property of NEAT algorithm. Every NEAT network is encoded by a Genome as it is shown un Figure 1, i.e., the network is represented by Node Genes and Connection Genes, representing to nodes and connexions, respectively. Each node or connection has a unique Innovation Id number, which is preserved even if the node or connection is removed (as a result of the evolution) and created again.

The evolution of an individual forming the NEAT networks is driven by its performance for a given fitness function. The fitness function determines the reproduction probability of a certain individual. The child generated by the reproduction mechanism will conserve the parents' nodes and connections, and randomly change the connection weights. If a connection only belongs to one parent, it is inherited instantaneously. A *specie* is obtained when a group of neural networks shares similarities between them, i.e., the distance between the two networks is below a threshold (it depends on the network topology and connection weights). Networks belonging to the same specie, have a higher probability to turn into parents of a new population of neural networks.

In the same manner to genetic algorithms, variability in the reproduction process in NEAT networks is given by mutation. The mutations here allow the creation of new nodes and could also modify the connection weight between them (new, delete, weight modulation).

2.2. HyperNEAT

An extension of NEAT, is the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) generative encoding algorithm [12]. HyperNEAT is formed by two neural netwoks: the main network called substrate, and a secondary NEAT network used to generate the connection weights of the substrate. The topology of the substrate is fixed (it does not evolve), and it is related with the existence of spatial constraints. The substrate contains a fixed number of neurons and layers: one input layer, one output layer and hidden layers; where each node in the network has a spatial position assigned.

The connections between nodes of the substrate are obtained through the secondary network implemented by NEAT. The NEAT network receives as input the spatial positions of the nodes to be connected represented as coordinates of an abstract matrix (see Figure 2). The output of the NEAT network is the connection weight between these two nodes. Following this implementation, HyperNEAT takes advantage of symmetries and regularities in the geometry of the substrate to solve the problem involved with spatial constraints.

Additional information can be also used as input to enhance the geometric characteristic on the implemented network, e.g. euclidean distance between nodes.

The benefit of a certain set of connection weights in the HyperNEAT implementation is evaluated by a fitness function. The fitness value is then passed to the secondary NEAT network in order to evolve the nodes' connection weights. The set of connection weights giving the best fitness value will be then chosen as a solution of the optimization problem.

3. τ-Hyperneat

Using the concepts briefly described in Section 2, and inspired by the recent NEAT variants proposed by [18], [19], [21], we propose a τ-HyperNEAT algorithm to compute the network connection weights and time delays between nodes inside a substrate. The τ-HyperNEAT algorithm uses

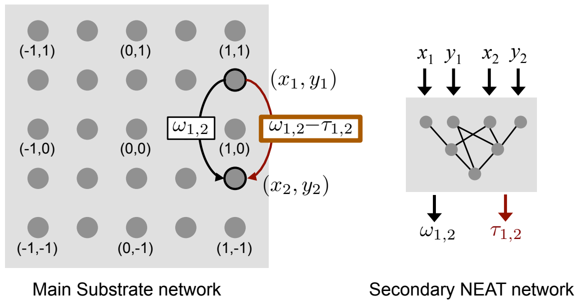


Figure 2. HyperNEAT (black arrows) and τ-HyperNEAT (reddish arrows) algorithms formed by two different neural networks. The main network *substrate* sets the network structure and it can be formed by several layers. The connection weight ($\omega_{1,2}$) between two nodes in the substrate (node 1 and 2) are found using the secondary neural network implemented by NEAT. In the case of τ-HyperNEAT the secondary NEAT network also delivers the time delay associated to that connection ($\tau_{1,2}$). Image adapted from [12].

a secondary NEAT network that not only gives as output the connection weight between two nodes, but also, the time delay between that connection. Similarly than [18], the connection delay between two nodes is implemented through a buffer with a length proportional to the time delay.

Figure 2 (reddish lines) shows a schema of the τ-HyperNEAT proposed in this article. The nodes in the substrate are now connected with a connection weight and a time delay obtained from the secondary NEAT network, adding thus, inertia to the system.

4. Gait learning comparison between HyperNEAT and τ-HyperNEAT

In order to evaluate the advantages of τ-HyperNEAT versus HyperNEAT learning robot gaits, we used the Quadratot robot platform (Figure 3). This robot platform has nine degrees of freedom: two joints per leg (for the ith leg, the inner joint M_{i1} and one outer joint M_{i2}) and a single joint rotating along the robot midline (M_0). The figure 3 shows the initial pose of Quadratot, corresponding to zero central angle for each joint. The angle range of each one of these is: $[-60°, 60°]$ for M_{i1} and M_{i2}, and, $[-23°, 23°]$ for M_0. The physical implementation of Quadratot uses a total of nine Dynamixel AX-12A motors with a torque of 1.5Nm.

Quadratot was imported in the V-REP Robot Simulator[1] mimicking its physical characteristics. The learning and testing process was done under this simulated environment.

4.1. Fitness definition

Each simulation trial gives Quadratot six seconds to perform the gait and then compute the fitness. For this implementation we used a *multi-objective fitness* $\mathcal{F}$ formed

1. V-REP, Virtual Robot Experimentation Platform. ©Coppelia Robotics.

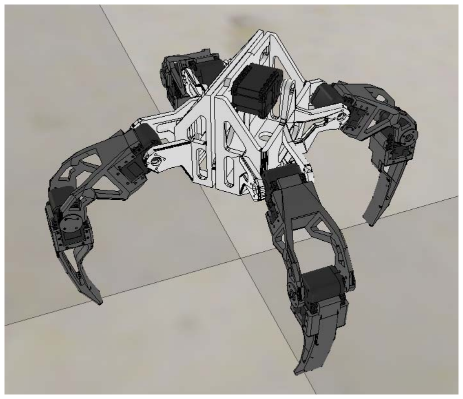


Figure 3. The QuadraTot robot platform used to test gait learning algorithms. Representation of the robot in the simulation environment V-REP.

by two components F_d and F_m. The first component F_d represents the reached distance from the center of the scene to the last position of the robot in the simulation. The second component, F_m, represents the frequency of the motors, and it shows preference for gaits where all the motors are involved in the motion.

Specifically, F_d is defined as

$$F_d = \begin{cases} 10 \ \exp\left(-\frac{(d-1.0)^2}{2(0.4)^2}\right), & \text{if } d \leq 1.0 \\ 10, & \text{if } d > 1.0, \end{cases} \tag{1}$$

where d is the reached distance, 1.0 is the objective distance and 0.4 is the Gaussian variance computed experimentally.

The second component forming the fitness, F_m, is defined as follows

$$F_m = \frac{1}{N_l}\sum_{i=1}^{N_l} F(f_i)$$
$$F(f_i) = \begin{cases} 10 \ \exp\left(-\frac{(\phi_i-1.32)^2}{2(0.7)^2}\right), & \text{if } F(f_i) \leq 6.5 \\ 10, & \text{if } F(f_i) > 6.5 \end{cases} \tag{2}$$

where N_l is the number of legs ($l = 1, 2, ..$) and $F(f_i)$ represents a nonlinear transformation of the average frequency per leg ϕ_i measured in Hz (considering the joint in the robots midline as one leg with one degree of freedom). In equation (2), 0.7 is the variance computed experimentally, and 1.32 is the objective desired frequency in Hz [16]. If the value of $F(f_i)$ is higher than 6.5, it automatically saturates on 10.

In order to simultaneously maximize both F_d and F_m, the global fitness $\mathcal{F}$ is simply defined as follows

$$\mathcal{F} = \min(F_d, F_m). \tag{3}$$

As it will be shown in the results, this simple definition makes both F_d and F_m be maximized along the learning process. This definition guides the fitness establishing some order on the relevance of a certain component. For instance, the saturation law in the F_m case is to allow a certain flexibility in the motor frequencies, and in those cases, improve the fitness according to F_d.

TABLE 1. NEAT PARAMETERS. THE SYMBOLS * AND ** REPRESENT PROBABILITY VALUES FOR SPECIES OF SMALL AND LARGE SIZES, RESPECTIVELY. A SPECIE IS CONSIDER *large* WHEN THE NUMBER OF CONNECTIONS EXCEEDS 10.

Organisms per generation	100
% of childs obtained only by mutation (without reproduction)	25%
Prob. of reproduction between species	0.1%
Prob. to add a new node*	0.08%
Prob. to add a new node**	0.06%
Prob. to add a new connection*	0.08%
Prob. to add a new connection**	0.06%
Number of generations	100
Prob. to change connection weight	30%
Prob. to change node function	10%

4.2. Implementation of τ-HyperNEAT and HyperNEAT

Both, for HyperNEAT and τ-HyperNEAT, the substrate is implemented on the same manner (see Figure 4). The substrate was implemented in three layers with 12 nodes each. *Input layer* only uses 11 nodes, 9 of them associated to each motor leg (4 legs, 2 motors per leg, and one extra motor) and 2 dedicated to a $\sin$ and $\cos$ functions used to provide periodicity to the motors movement. *Hidden layer* uses the 12 nodes to create connections between the input and output layer. Finally, *Output layer* only uses 9 nodes corresponding again to one node per motor. Those outputs are then feedback as inputs to the *Input layer*.

The secondary NEAT network, in both cases, considers 5 inputs: two pairs of spatial coordinates of each node to be connected, plus the euclidean distance between them. The main difference between the secondary network of HyperNEAT versus τ-HyperNEAT is that in this last case the network generates the double number of outputs. For a given pair of nodes, the secondary NEAT network gives two connection weights as output, one for the connection between the input and hidden layer, and a second value for the connection between the hidden and output layer. In the case of τ-HyperNEAT, additionally to these two connection weights two temporal delays are also given as output following the same logic in the connection. The secondary NEAT network is implemented using a custom version of NEAT algorithm[2] according to the algorithm described in [1] with the parameters listed in Table 1. The number of input and output nodes in the secondary NEAT network are constraint by the HyperNEAT design as it is shown in Figure 4.

τ-HyperNEAT network needs to implement time delays between node connections. As we previously mentioned, the time delays are mapped to a buffer size where each position represents a time step. In this case, each time step is $1/160$s and we used buffers with a maximal time delay of 62.5ms.

4.3. Experimental results

As it is specified in Table 1, we ran five times the learning processing for 100 generations each time. In each generation we computed the fitness $\mathcal{F}$ defined in equation (3) for both implementations, HyperNEAT and τ-HyperNEAT obtaining the mean and standard deviation of the five experiments as it is shown in Figure 5(a).

As it shown, in terms of fitness performance which mainly involves the distance walked and the motor frequencies, there is no major difference between the HyperNEAT and τ-HtperNEAT. Both methods reached similar fitness values across generations with a mean close to 5 and a maximal performance close to 10. There is evidence of a small difference in the speed of progress at the start of the training, which seems logical considering that the number of parameters to be optimized in the τ-HyperNEAT case is larger. Furthermore, the magnitude of the fitness dispersion obtained for both methods is maintained.

We also asked whether the values of time delays, given by the buffer size learned by τ-HyperNEAT, are significantly greater than zero showing that adding time delays to the network connectivity has sense. Figure 5(b) shows the distribution of buffer sizes for the *champion*, i.e., the resulting set of connection weights and time delays with the highest performance. Even if 25% of the network time delays are small, the remaining 75% shows a Gaussian distribution of time delay magnitudes throughout the network with a mean value of 37.5ms.

Furthermore, we also analyzed the distribution of connection weights learned by HyperNEAT and τ-HyperNEAT shown in Figure 6. We observe no significant differences in terms of mean values, but the fact that HyperNEAT is biased to inhibitory connections while τ-HyperNEAT has mainly excitatory connections.

Nevertheless, although fitness results does not reflect an improvement between the proposed method and its predecessor, there is an improvement in terms of the gait execution. Figure 7 displays the activation signals delivered to the leg joints by both methods: HyperNEAT (b) and τ-HyperNEAT (c) organized by proximal and distal joints. As it can be seen in Figure 7 (b) all signals given to the leg joints share the same phase, unlike Figure 7(c), which clearly shows phase differences in the signals resulting in an alternated movement of one side of the robot at the time. This phase difference between the leg positions can be also observed in Figure 8 where they are organized by legs. We also observed small movements in the distal joints obtained with τ-HyperNEAT (Figure 8), which also gives as a learning result a smaller rotation of the central joint J9 compared to HyperNEAT. The movement pattern observed in τ-HyperNEAT, where the phase of the driving signals in the leg pairs is shifted, allowed leg joints to move alternately having as a result a more naturalistic gait.

5. Conclusion

After analyzing the results obtained from the experiments it is possible to conclude that, in gait learning task, there were no quantitative differences in the results obtained through the HyperNEAT and τ-HyperNEAT methods, obtained in both cases the same performance according

2. See GitHub mjescobar/NEAT

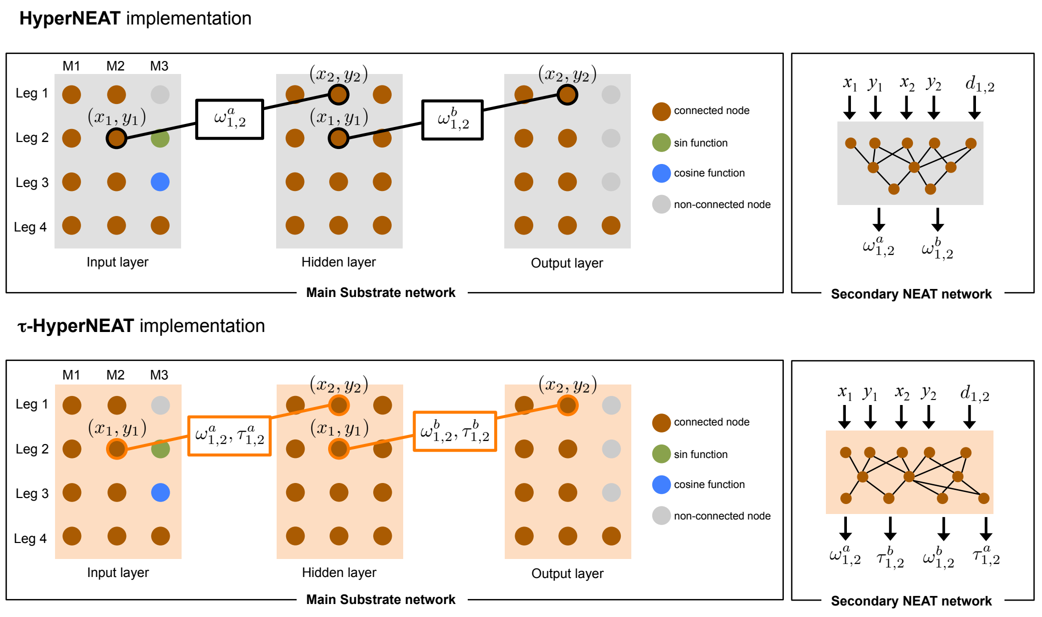
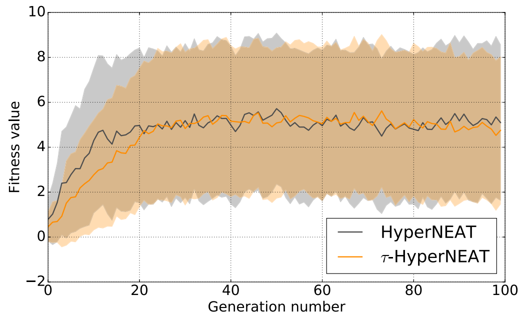
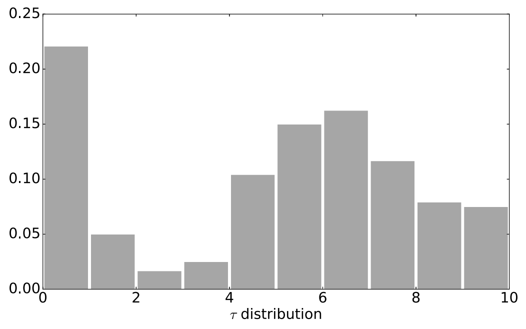


Figure 4. Implementation of HyperNEAT and τ-HyperNEAT used in this article. In both cases the Main Substrate network is formed by three layes of 12 nodes. The input layer only uses 11 nodes where 9 of them are used for each motor (4 legs, 2 motors and one extra motor: M1, M2 and M3), and, the other two for two periodic functions (sin and cos). The hidden layer uses the 12 nodes and the output layer again only uses 9 nodes. Secondary NEAT networks are different in both cases. In the case of τ-Hyperneat it does not only gives as output the connection weights between the three layes, but also, the time delay of those connections.

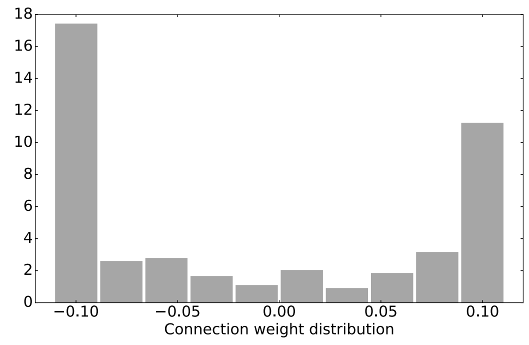


(a)

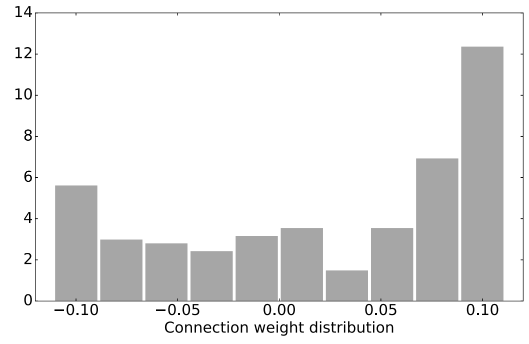


(b)

Figure 5. **(a)** Fitness value between HyperNEAT and τ-HyperNEAT. **(b)** Distribution of the buffer sizes learned in τ-HyperNEAT.

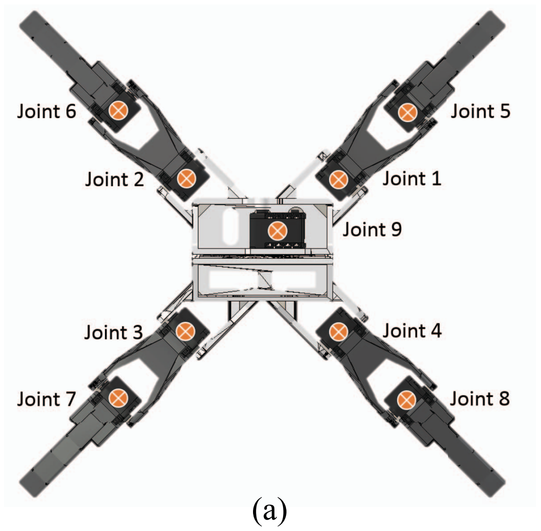


(a)

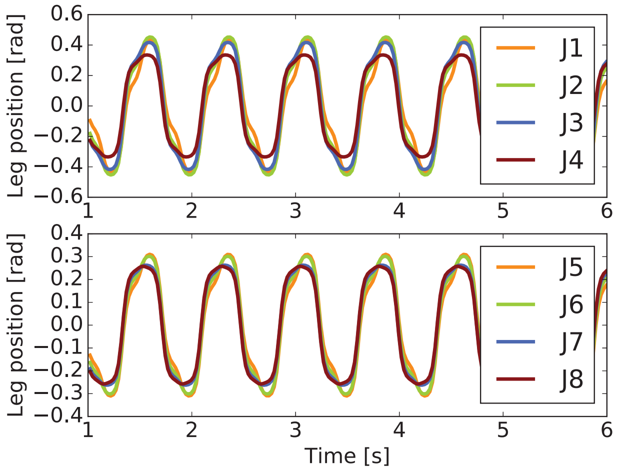


(b)

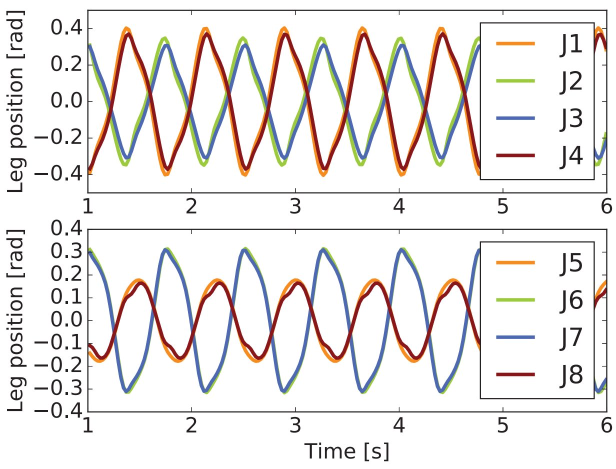
Figure 6. Distribution of the connection weights learned in HyperNEAT **(a)** and τ-HyperNEAT **(b)**.



(a)



(b)



(c)

Figure 7. **(a)** Zenithal view of the robot associating a label with each of their nine motors. **(b)-(c)** Signals delivered to the Quadratot motors, in two sample gaits, indicating the leg position in [rad] using HyperNEAT and τ-HyperNEAT, respectively.

to the fitness declared in this approach. Nevertheless, τ-HyperNEAT developed almost in a 100%, gaits with harmonic and coordinated movements, similar to behaviours seen in nature. This could be observed in the graphs of joint signals generated by τ-HyperNEAT, demonstrating the differences and similarities of phases generated between different legs and joints of a same leg respectively. Quantitative differences could have been observed with the addition of terms measuring the harmony and balance of the resulting gait, but, interestingly the harmonic gait obtained with τ-HyperNEAT emerged as a network property for this learned task.

Since HyperNEAT capabilities to generate gaits with harmonic and coordinated movements were practically null against the capacities shown by the τ-HyperNEAT method, we can confirm that this new method incorporating time delays manages to solve dynamic problems, in particular: gait generation for legged robots.

References

[1] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[2] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, “Planning walking patterns for a biped robot,” *IEEE Transactions on robotics and automation*, vol. 17, no. 3, pp. 280–289, 2001.

[3] S. Ma, T. Tomiyama, and H. Wada, “Omnidirectional static walking of a quadruped robot,” *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 152–161, 2005.

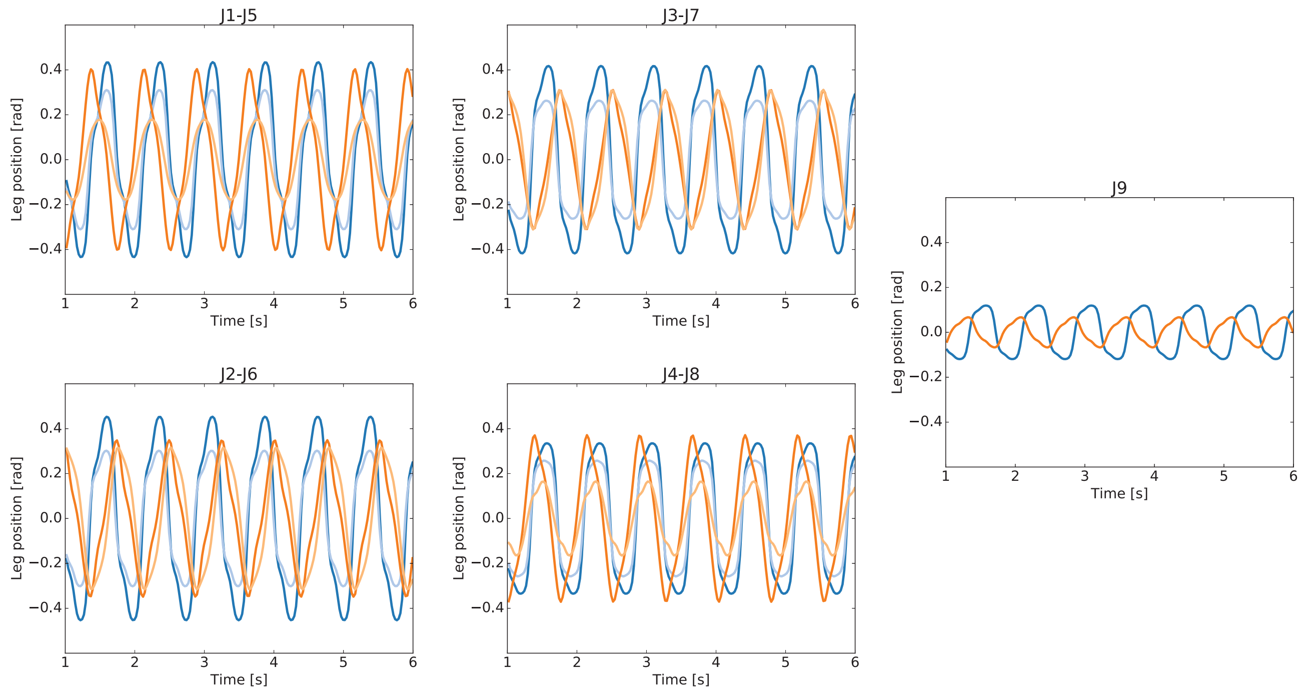


Figure 8. The figure shows the graphs of the signals delivered to the motors of Quadratot, organized by legs, using HyperNEAT (blue lines) and τ-HyperNEAT (orange lines). Dark lines are associated to the inner motors: J1, J2, J3 and J4. By the contrary, light lines are associated to the outer motors: J5, J6, J7 and J8.

[4] S. Hirose, Y. Fukuda, K. Yoneda, A. Nagakubo, H. Tsukagoshi, K. Arikawa, G. Endo, T. Doi, and R. Hodoshima, "Quadruped walking robots at tokyo institute of technology: design, analysis, and gait control methods," *IEEE Robotics and Automation Magazine*, vol. 16, no. 2, pp. 104–114, 2009.

[5] G. Carbone and A. Di Nuovo, *A Hybrid Multi-objective Evolutionary Approach for Optimal Path Planning of a Hexapod Robot*. Cham: Springer International Publishing, 2016, pp. 131–144.

[6] E. Ruppin, "Evolutionary autonomous agents: A neuroscience perspective," *Nature Reviews Neuroscience*, vol. 3, no. 2, pp. 132–141, 2002.

[7] S. Whiteson, P. Stone, K. O. Stanley, R. Miikkulainen, and N. Kohl, "Automatic feature selection in neuroevolution," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1225–1232.

[8] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving neural network agents in the nero video game," *Proceedings of the IEEE*, pp. 182–189, 2005.

[9] ——, "Real-time neuroevolution in the nero video game," *IEEE transactions on evolutionary computation*, vol. 9, no. 6, pp. 653–668, 2005.

[10] D. B. Knoester and P. K. McKinley, "Neuroevolution of controllers for self-organizing mobile ad hoc networks," in *Self-Adaptive and Self-Organizing Systems (SASO), 2011 Fifth IEEE International Conference on*. IEEE, 2011, pp. 188–197.

[11] J. E. Auerbach and J. C. Bongard, "Evolving complete robots with cppn-neat: the utility of recurrent connections," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. ACM, 2011, pp. 1475–1482.

[12] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[13] J. Gauci and K. O. Stanley, "Autonomous evolution of topographic regularities in artificial neural networks," *Neural computation*, vol. 22, no. 7, pp. 1860–1898, 2010.

[14] E. Haasdijk, A. A. Rusu, and A. Eiben, "Hyperneat for locomotion control in modular robots," in *International Conference on Evolvable Systems*. Springer, 2010, pp. 169–180.

[15] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the hyperneat generative encoding," in *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2009, pp. 2764–2771.

[16] J. Yosinski, J. Clune, D. Hidalgo, S. Nguyen, J. Zagal, and H. Lipson, "Evolving robot gaits in hardware: the hyperneat generative encoding vs. parameter optimization," in *Proceedings of the 20th European Conference on Artificial Life*. Citeseer, 2011, pp. 890–897.

[17] A. M. Callegaro, O. Unluhisarcikli, M. Pietrusinski, and C. Mavroidis, *Robotic Systems for Gait Rehabilitation*. Dordrecht: Springer Netherlands, 2014, pp. 265–283.

[18] P. Caamaño, F. Bellas, and R. J. Duro, "τ-neat: Initial experiments in precise temporal processing through neuroevolution," *Neurocomputing*, vol. 150, pp. 43–49, 2015.

[19] P. Caamaño, R. Salgado, F. Bellas, and R. Duro, "Introducing synaptic delays in the neat algorithm to improve modelling in cognitive robotics," *Neural Processing Letters*, vol. 43, no. 2, pp. 479–504, 2016.

[20] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving adaptive neural networks with and without adaptive synapses," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4. IEEE, 2003, pp. 2557–2564.

[21] P. Caamano, F. Bellas, and R. J. Duro, "Augmenting the neat algorithm to improve its temporal processing capabilities," in *2014 international joint conference on neural networks (IJCNN)*. IEEE, 2014, pp. 1467–1473.