Evolutionary Intelligence
https://doi.org/10.1007/s12065-020-00394-9

RESEARCH PAPER

Neuromodulated multiobjective evolutionary neurocontrollers without speciation

Ian Showalter[1] · **Howard M. Schwartz**[1]

Received: 27 September 2019 / Revised: 14 February 2020 / Accepted: 18 March 2020

Abstract
Neuromodulation is a biologically-inspired technique that can adapt the per-connection learning rates of synaptic plasticity. Neuromodulation has been used to facilitate unsupervised learning by adapting neural network weights. Multiobjective evolution of neural network topology and weights has been used to design neurocontrollers for autonomous robots. This paper presents a novel multiobjective evolutionary neurocontroller with unsupervised learning for robot navigation. Multiobjective evolution of network weights and topologies (NEAT-MODS) is augmented with neuromodulated learning. NEAT-MODS is an NSGA-II based multiobjective neurocontroller that uses two conflicting objectives. The first rewards the robot when it moves in a direct manner with minimal turning; the second objective is to reach as many targets as possible. NEAT-MODS uses speciation, a selection process that aims to ensure Pareto-optimal genotypic diversity and elitism. The effectiveness of the design is demonstrated using a series of experiments with a simulated robot traversing a simple maze containing target goals. It is shown that when neuromodulated learning is combined with multiobjective evolution, better-performing neural controllers are synthesized than by evolution alone. Secondly, it is demonstrated that speciation is unnecessary in neuromodulated neuroevolution, as neuromodulation preserves topological innovation. The proposed neuromodulated approach is found to be statistically superior to NEAT-MODS alone when applied to solve a multiobjective navigation problem.

Keywords Artificial neural network · Hebbian learning · Multiobjective · NEAT-MODS · Neuromodulation · Speciation

1 Introduction

Fully autonomous robots are needed to aid humans in many fields. Robots can go places that biological lifeforms can not, and willingly perform tasks that humans find monotonous. When communication between robots and human controllers is difficult due to distance or interference, some degree of autonomy is required. Autonomy requires robots able to adapt to changing environments. Evolution and unsupervised learning are both mechanisms that can provide autonomous adaptation with respect to a changing environment.

Lifeforms face competing problems. For example, plants typically face competing objectives such as finding water and obtaining sunlight. These objectives compete for the same resources, yet the plant cannot survive without the resources provided by both. Similarly, the vehicle designer faces competing objectives such as minimizing energy consumption by reducing mass, and maximizing vehicle range requiring energy storage (which increases mass). Autonomous robots face may different objectives such as completing missions in the minimum amount of time while simultaneously minimizing power consumption. Multiobjective optimization is an area of research allowing several objective functions to be maximized simultaneously without the use of an auxiliary function. Using an auxiliary function requires that separate objectives be weighted and combined into a single function. Auxiliary functions require assumptions about the Pareto front, whereas multiobjective solutions aim to search for the entire Pareto front simultaneously. Multiobjective evolutionary neurocontrollers have been shown to successfully adapt neural network topology and weights

✉ Ian Showalter
ianshowalter@cmail.carleton.ca

Howard M. Schwartz
howard.schwartz@sce.carleton.ca
http://www.sce.carleton.ca/faculty/schwartz/index.html

[1] Department of Systems and Computer Engineering, Carleton University, 4456 Mackenzie, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada

Published online: 06 April 2020

[1], incrementally growing from a basal initial structure, and evolved to a minimal topological solution [21].

Artificial neural networks (ANN) have been successfully used to operate robotic systems over the last few decades. They are an effective tool for robotic control, and promise many advantages over conventional control such as the ability to learn, and adapt unsupervised to changing environments. Determining the smallest size network topology is desirable to minimize computational cost, latency, and power consumption. Many different techniques have been applied to the training and topology of ANNs, including gradient descent methods with grown or pruned topologies [16], evolutionary methods, and biologically plausible methods such as Hebbian learning and neuromodulation [11]. NeuroEvolution of Augmented Topologies (NEAT) was successfully demonstrated for function approximation and the double pole balancing problem in the original publication [21], and subsequently for other problems. NEAT-MODs has adapted NEAT for multiobjective problems, and demonstrated the evolution of robot neurocontrollers [1]. Similarly, NEAT has been adapted to evolve neurocontrollers in a distributed on-line manner in odNEAT [18]. The odNEAT method has been augmented with Hebbian neuromodulation to further reduce convergence times [17].

Combining evolution and learning can provide a powerful synergy between complementary search algorithms. Networks with evolved initial weights can be trained faster, and to a higher degree of accuracy, than networks with random initial weights [15]. According to Hebbian theory, synaptic plasticity is the mechanism by which, when an axon of cell *A* repeatedly excites cell *B*, a change takes place in one or both cells such that *A*'s efficiency in firing *B* is increased [6]. Hebbian learning is therefore an unsupervised method of training where the connection weights (strengths) are updated as a functions of pre- and post-synaptic activity [19]. Neuromodulation is considered to be a major mechanism producing memory and learning in biological nervous systems [19]. Specialized neuromodulatory neurons control the amount of plasticity of other neurons in biological organisms by using neurotransmitters such as dopamine and serotonin [19]. Neuromodulation of the synaptic plasticity augments the Hebbian learning rule by providing gating of the plasticity of a synapse between two other neurons, by updating the synapse after the neuron has fired [7, 11].

It is proposed that adding neuromodulation to neuroevolution will improve neurocontroller performance. Secondly, it is hypothesized that protecting innovation by the use of species as in NEAT [21], and NEAT-MODS [1] is unnecessary when neuromodulation is used to adapt the neurocontrollers during the operation of each generation, between times when the offspring population is generated.

We propose a novel architecture where a neuromodulated multiobjective evolutionary neural controller is trained in real time during each evaluation tournament. Here, a tournament is defined as the time period during which candidate controllers are evaluated (here, by simulation of a robot in the test environment) to determine their individual fitness. Like NEAT-MODS [1], the multiobjective neural controller is assembled based on NEAT [21] and evolved using NSGA-II [3], but unlike NEAT-MODS, speciation is not used. Each candidate neurocontroller's fitness is determined by tournament, where each candidate neurocontroller is judged based on its performance controlling the robot in the test arena. The candidate neurocontroller's weights are modified online during each tournament using neuromodulated Hebbian learning. The proposed architecture overcomes both the evolutionary shortcomings described above by alternatively testing the learning space with a genetic operator, and then attempting to improve upon these results using neuromodulated learning to adapt the network between each time step during operation. This approach also allows exploration of the entire learning space, and fine tuning to find each local error minimum, until a solution with the global minimum error is found. Using NEAT-MODS alone, fine-grain adjustment of the connection weights requires mutation, which only occurs between generations, when offspring are produced. By including neuromodulation the weights can be adjusted continuously during the lifetime of each generation due to synaptic plasticity [17].

To demonstrate the effectiveness and improved performance given by neuromodulation when applied to evolved multiobjective neurocontrollers, simulated robots using neurocontrollers evolved by simple NEAT-MODS, neuromodulated NEAT-MODS, and the proposed neuromodulated multiobjective non-speciated NEAT are applied to a basic autonomous foraging and maze task. Foraging is a task that would be required to be performed by fully autonomous machines, in order to acquire fuel and parts for self-repair. The maze aspect simulates obstacles encountered during operation.

2 Background

Neuroevolution is the design of artificial neural networks using evolutionary methods. Parameters of the neural network are encoded in mathematical models of genes, then optimized in an evolutionary fashion. Evolutionary techniques have been used in artificial neural networks to determine weight values, network architectures such as number of neurons in a layer, or connections between neurons, or deep convolutional neural networks in a block-wise fashion. Evolutionary techniques have also been used to determine activation functions in [5]. Neuroevolution has been applied to many problems, including evolving platooning strategies in intelligent transportation systems [23], and estimating stock

closing indices using evolutionary neural networks [10], and neuro-evolutionary systems for foreign exchange trading [9].

2.1 NeuroEvolution of augmenting topologies (NEAT)

Neuroevolution representations can be divided into three classes: direct, developmental and implicit. Direct representations have been generally used to evolve the parameter values of fixed-sized networks [5]. NeuroEvolution of Augmented Topologies (NEAT) is a direct representational method for genetically encoding and evolving the weights and architecture of ANNs [21]. NEAT uses a unique innovation number associated with each gene to track the history of the genetic markers. This facilitates crossover without suffering from the "competing convention", where computation is wasted when duplicates of the same or virtually identical structure compete against each other. The competing conventions problem arises because the order of the genes is unimportant. For example, a genome containing neurons *A*, *B*, and *C* can be represented by the sequences [*ABC*] and [*ACB*] (and others). Applying the crossover function to these two genomes would lead to missing information, as the offspring would be formed [*ABB*] or [*ACC*]. When a new gene is added, the global innovation number is incremented and assigned to that gene. As a new gene is added to the genome, it receives its own innovation number, so that genes do not get out of order when crossover is performed. When crossover is performed, the genes of the two parent sequences are lined up sequentially by innovation number (see Fig. 2). Thus the sequence [*ACB*] is sorted to become [*ABC*], and the sequence [*ACC*] (with duplicate gene *C*) is prevented from being created by the crossover of [*ABC*] with [*ACB*]. The NEAT genome is shown in Fig. 1.

To produce the offspring generation from the parent generation, NEAT applies crossover and mutation operators. The crossover between two parents of different structures is based on the innovation number. When crossover is performed in NEAT, the genes of both parents are lined up by innovation number. Genes that have the same innovation number are called matching genes. In Fig. 2, the genes with innovation numbers 1 through 5 are matching genes. Genes that do not match are called disjoint if they occur within the range of the other parent's innovation numbers, or excess if they occur outside the other parent's range. In Fig. 2, the genes with innovation numbers 6 through 8 are disjoint genes, those with innovation numbers 9 and 10 are excess genes. Crossover in NEAT is accomplished by randomly selecting matching genes from both parents, and disjoint and excess genes from the parent with better fitness. Figure 2 shows disjoint and excess genes and the basic NEAT

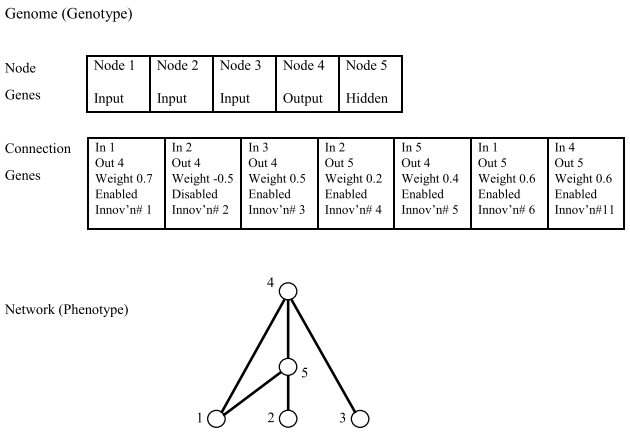
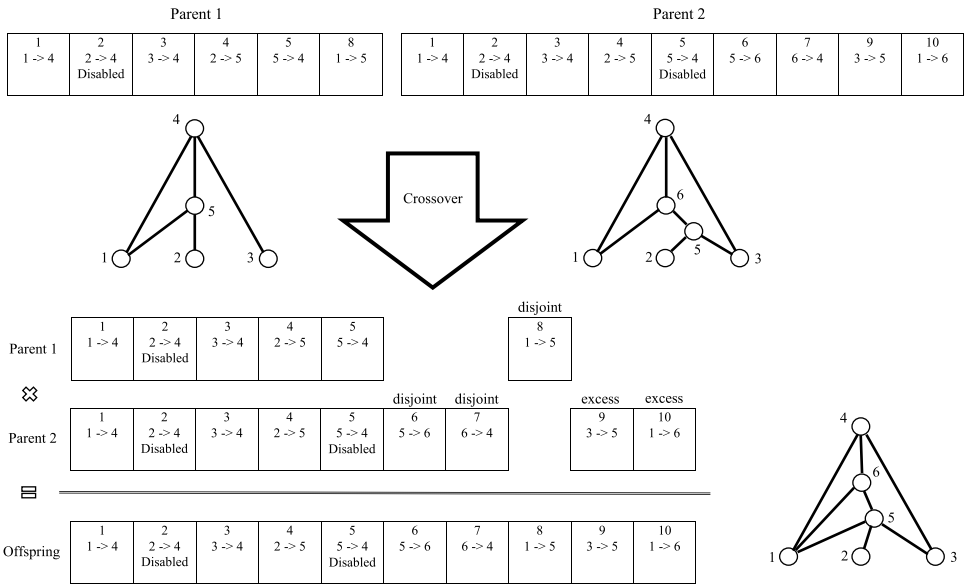


Fig. 1 NEAT genome

Fig. 2 NEAT crossover



crossover operation. Note that this figure does not take into account the fitter parent. Regardless of which parent is fitter, each of the genes with innovation number 1 through 5 would be selected at random from one or the other of the parents. If Parent 1 is the fitter of the two parents, the disjoint gene with innovation number 8 would also be selected to form the complete Offspring sequence. If Parent 2 were the fitter parent, the Offspring would inherit the genes with innovation numbers 6, 7, 9 and 10 in addition to the five genes randomly selected from both parents.

The mutation operation allows for weight perturbation or replacement of one gene of a sequence, or the addition of a node or a connection. NEAT mutation is shown in Figs. 3 and 4. In Fig. 3, the left-hand network is modified by adding a connection between nodes 3 and 5. The result is shown in the right-hand figure, with a seventh gene being added to the sequence describing the connection between nodes 3 and 5. This is also shown in the resulting network.

Figure 4 shows mutation in the form of the addition of a node. Node 6 is added as two new genes to the end of the sequence, with innovation number 8 (indicating a connection between nodes 3 and 6), and innovation number 9 (indicating a connection between nodes 6 and 4). The node sequence is also updated with the new node.

NEAT also uses speciation, in which the total population of individuals is divided into species, to preserve innovation: Sequences with similar genes and structures are considered to be of the same species, and are more likely to breed together (crossover) than with members of other species. Dividing the population into species allows new topologies with non-optimal weights an opportunity to evolve their weights (via the evolutionary weight mutation operation) towards optimal values without immediately being killed off. Inter-species differences between individual i and j are determined using a compatibility distance δ function based on the number of excess genes, disjoint genes, and average weight differences.

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \overline{W} \tag{1}$$

Coefficients c_1, c_2, and c_3 adjust the importance of the number of excess genes E, disjoint genes D, and matching genes $\overline{W}$ respectively. N is the connection gene sequence length, but is set to 1 in [21] if the connection gene sequence length is less than 20.

Species are weighted using a sharing function based on the compatibility distance function, such that organisms in the same species share their fitness. Offspring populations

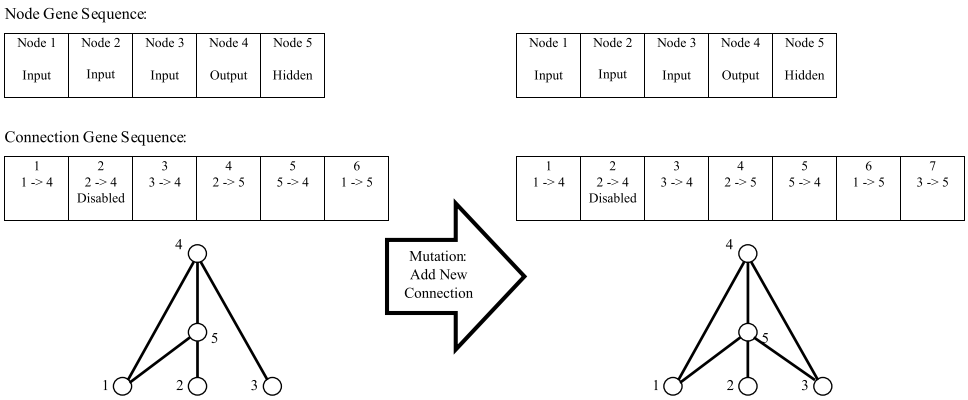


Fig. 3 NEAT mutation: add connection

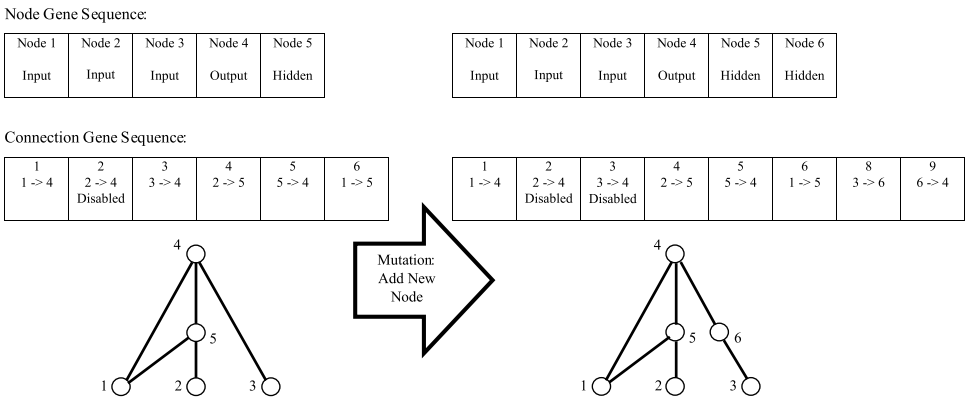


Fig. 4 NEAT mutation: add node

are evaluated using a fitness function f. The result is weighted using the sharing function presented in Eq. 2. This weighted fitness f' is then ranked by nondominated sorting.

$$f'_i = \frac{f_i}{\sum_{j=1}^{n} \mathrm{sh}(\delta(i,j))} \tag{2}$$

where the sharing function sh(δ) is:

$$\mathrm{sh}(\delta(i,j)) = \begin{cases} 0 & \delta(i,j) > \delta_t \\ 1 & \delta(i,j) \leq \delta_t \end{cases} \tag{3}$$

NEAT then follows the method used in NSGA-II [3]. The next offspring population is populated based on the weighted ranking. The new population is then randomly mutated by any of: Perturbation of weights, replacement of weights, addition of a new node, addition of a new connection, disabling a connection, intraspecies crossover, or interspecies crossover. NEAT has been applied to many problems, including the pole balancing problem [21], computer games [14, 20], and robot control [22].

2.2 NEAT-MODS

NEAT-MODS is a NEAT-based multiobjective evolutionary algorithm that aims to maximize two (or more) objectives without the use of an auxiliary function. In [2] it is argued that it is more efficient to approach objectives in a simultaneous manner than sequentially in the search for the Pareto-optimal solution, as multiobjective evolutionary algorithms are more easily parallelizable, and conflicting objectives ensure good diversity in the search space [8]. In NEAT-MODS, the basic genotype, species diversification and steps of NEAT are followed, but with the substitution of the nondominated sorting of NSGA-II being used, allowing Pareto-optimal controllers to be evolved simultaneously for problems with conflicting objectives. For the problem of robot navigation, the conflicting objectives here as in [1] are *achieving goals* while *avoiding obstacles*. NEAT-MODS uses NEAT's speciation. The NEAT-MODS process implemented for the research is presented in Algorithm 1.

Algorithm 1 NEAT-MODS

1. Initialization	A minimal topology network is defined with no hidden layer nodes. One edge connects each input directly to each output. An initial Offspring population of individuals is generated with randomly assigned weights. An empty Parent population is also defined.
while $gens < gens_{max}$	Repeat until the generational count has reached the termination condition.
2. Tournament	The NEAT genes of each offspring individual are used to construct an ANN that is then used to control a simulated robot.
3. Evaluation	The performance of each offspring individual's ANN is calculated for each objective based on their performance.
4. Combine Populations	The Parent and Offspring populations are combined for selection.
5. Ranking	The combined population is ranked using the nondominated sorting algorithm of NSGA-II.
6. Species	The species affiliation of each individual in the combined population is calculated per Equation 1.
7. Sorting	Individuals grouped into their species, and sorted within the species based on the nondominated ranking from Step 5.
8. Sorting of Species	The species are sorted based on their highest nondominated ranking individuals.
9. Selection	From top rank species to lowest ranked, the top ranking individual of each species is selected, followed by the next top ranking individual of each species. The process continues down the ranking of each combined population species until the offspring population is filled.
10. Parent Population	The new Offspring population is saved as the Parent population.
11. Reproduction	As in NEAT, the mutation of ANN weights by uniform perturbation and random replacement, new node addition, new connection addition, connection disabling, crossover and inter-species crossover are performed on the Offspring population in a probabilistic manner.
12. Stopping criteria	Steps 2 through 11 are repeated until the generational count has reached the termination condition.

2.3 Hebbian learning

It has long been known that any two nerve cells that are repeatedly active at the same time become associated in such a manner that activity in one facilitates activity in the other. Hebb's theory proposes the following: "Let us assume then that the persistence or repetition of a reverberatory activity (or 'trace') tends to induce lasting cellular changes that add to its stability. The assumption can be precisely stated as follows: *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*"—[6] Hebb continues to say that "When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell."—[6] The synapse is the junction through which signals flow between two nerve cells. Figure 5 shows a stylized neuron

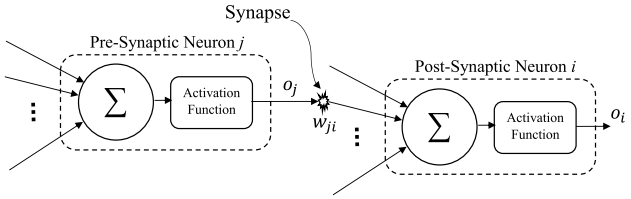


Fig. 5 Neuron and synapses

cell indicating the synapse, and the pre- and post-synaptic neurons.

Hebbian learning is an unsupervised method of training where the connection weights (strengths) are updated as a function of pre- and post-synaptic activity [19]. Synaptic plasticity Δw is the strengthening or weakening of synapse strength over time according to increases or decreases in their activity [6]. As in [11], the updating of synaptic weights is performed as per Eq. 4 where η is the learning rate, o_j is the activation level of the pre-synaptic neuron, o_i the activation level of the post-synaptic neuron, and w the connection weight.

$$\Delta w = \eta o_j o_i \tag{4}$$

Often, a more advanced rule of synaptic plasticity is used, such as that of Eq. 5, where A, B, C, D are the correlation term, pre-synaptic term, post-synaptic term, and constant weight increase or decay rate. These parameters are tuned to adapt the synaptic plasticity.

$$\Delta w = \eta\left(Ao_j o_i + Bo_j + Co_i + D\right) \tag{5}$$

Hebbian-based learning has similarities to backpropagation, but does not include or require an error signal or value. While allowing fully unsupervised learning, the Hebbian learning rule alone is not as effective as the delta rule, at least for the simplified model of neural networks presented above [13].

2.4 Neuromodulation

In the brain, some specialized neurons release chemical transmitters to control the rate of learning of the connections between neurons [12]. This phenomenon is called neuromodulation and is considered to be a major mechanism producing memory and learning in biological nervous systems [19]. The neuromodulatory neurons control the amount of plasticity of other neurons in biological organisms by using neurotransmitters such as dopamine and serotonin [19]. The computational theory on the roles of neuromodulatory systems and how they mediate signals that regulate the learning mechanisms in the brain is presented in [4]. Based on a review of experimental data and theoretical models, a unified theory on the roles of neuromodulators is presented. In this model, dopamine controls the error in reward prediction, serotonin controls the time scale of reward prediction, noradrenaline controls the randomness in action selection, and acetylcholine controls the speed of memory update.

Neuromodulation of the synaptic plasticity augments the classic (Hebbian) learning rule by providing gating of the plasticity of a synapse between two other neurons, by updating the synapse after the neuron has fired [7, 11]. Increased performance in ANNs through simple Hebbian plasticity has previously been demonstrated, but shown to have limited learning and memory capabilities in more complex tasks [11]. Controlling Hebbian synaptic plasticity by neuromodulation has been presented as more powerful and biologically plausible than simple Hebbian plasticity in [7]. In [19], neural networks that employed neuromodulatory neurons were found to have a clear advantage over those with no neuromodulatory neurons based on experimental data.

A simplified version of neuromodulation is assumed in [17], and a similar approach is used in this research. Here the model of the neuromodulation activation for each neuromodulating neuron is calculated using Eq. 6, where w_{ji} is the weight connection of the pre-synaptic neuron j and the post-synaptic neuron i, and o_j is the output of pre-synaptic neuron j.

$$m_i = \sum_j w_{ji} o_j \tag{6}$$

Applying neuromodulation from Eq. 6 to the model of synaptic plasticity described in Eq. 4, the weight between neuron j and neuromodulated neuron i is modified using Eq. 7 (o_i is the output of the post-synaptic neuron i, and o_j is the output of pre-synaptic neuron j).

$$\Delta w_{ji} = \eta \tanh\left(\frac{m_i}{2}\right)\left(Ao_j o_i + Bo_j + Co_i + D\right) \tag{7}$$

where m_i represents the amount of neuromodulator (such as dopamine) received and is the neuromodulation transmitted by the neuromodulating neuron and connections. The values A, B, C, and D can be determined in a variety of manners, including evolutionary methods. Figure 6 shows how neuromodulation is applied by a neuromodulating neuron to neuromodulated neurons. Here, each weight represents a synapse. The value of the weight represents the amount of signal transmitted from the pre-synaptic neuron, through the synapse, to the next neuron, the post-synaptic neuron.

Hebbian learning, and by extension neuromodulated learning, are unsupervised learning methods, as no desired value is necessary. Figure 7 compares two neural networks with the same topology. The neural network that is trained using backpropagation requires a desired output signal that is needed to calculate the output error signal, which is then

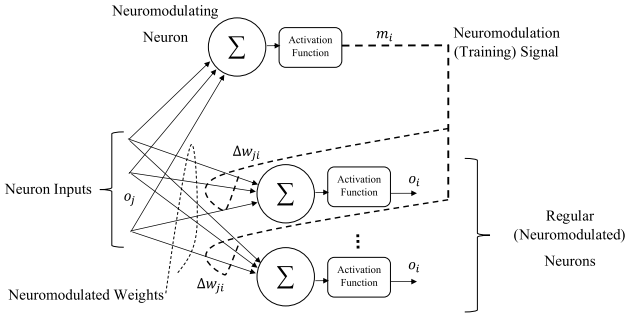


Fig. 6 Neuromodulation

used to adjust (control the rate of plasticity of) the weights of the neural network. The neural network trained using neuromodulation uses its own specialized neuromodulating neurons to control the rate of plasticity. Unlike the backpropagation algorithm, no error feedback is required in neuromodulated Hebbian learning (as Fig. 7 shows), and thus it is fully unsupervised, fulfilling one of the requirements for fully autonomous robots. Evolutionary methods can be used to determine the parameters of the neural networks, including those of neuromodulation, and in these cases, the objectives used in the evolutionary optimization could be considered as a form of supervision.

3 NEAT-MODS with neuromodulation

The proposed architecture overcomes both of the evolutionary shortcomings by alternatively testing the learning space with a genetic operator, and then attempting to improve upon these results using neuromodulated Hebbian learning to adapt the network during operation. This approach allows exploration of the entire learning space, and fine tuning to find each local error minima, until a solution with the global minimum error is found. We propose a multiobjective evolutionary neurocontroller that is assembled based on NEAT-MODS [1], with network weights that are modified during each evaluation tournament using neuromodulated Hebbian learning, as applied to a single objective NEAT-based neurocontroller in [18]. Here, we define a tournament as the time period during which candidate controllers are both learning *and* evaluated to determine their individual fitness. The NEAT node (neuron) gene is augmented to include the synaptic plasticity terms A, B, C, D, and a flag to denote if the node was standard or neuromodulated. The NEAT connection (synapse) gene is similarly augmented to include neuromodulatory neurons. In the experiments presented here, the model of neuromodulation allows neuromodulating neurons to modulate any neurons, including themselves and other neuromodulating neurons. The neuromodulated NEAT node and connection genomes are presented in Fig. 8:

In Fig. 8, node 6 is a neuromodulating node, the connection gene with innovation number 16 is a neuromodulating connection, and node 4 is a neuromodulated node.

3.1 Species and neuromodulation

In NEAT and NEAT-MODS, species are maintained based on the following premise: "Speciating the population allows organisms to compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected in a new niche where they have time to optimize their structure through competition within the niche." [21]. The concern is that when a new topology is evolved, it may have non-optimal weight values, and therefore will not be selected to form part of the offspring generation, and will be discarded as a viable topology. By grouping similar topologies, and protecting new topologies for a few generations, the new topologies are given an opportunity to evolve their weights to more optimal values.

When neuromodulation is used with NEAT, the candidate topologies are able to optimize their weights during the tournament (as defined in Sect. 2.4, and in Algorithm 1), before ranking and selection. It is therefore hypothesized that speciation is unnecessary when neuromodulation is used with NEAT-based neuroevolution, as neuromodulation produces the desired topological innovation. Neuromodulation allows candidates (organisms) to optimize their structure (weights) while operating, during the tournament. Thus at the end of each tournament, candidates (organisms) do not need to be protected within a niche, as they are optimized to compete with the entire population.

Fig. 7 Neuromodulation is unsupervised learning

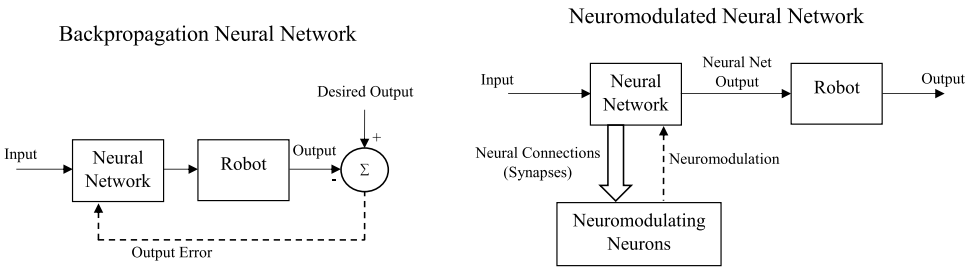


Fig. 8 Neuromodulated NEAT genome

Genome (Genotype)

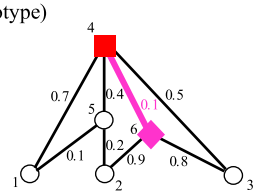
Node Genes

Node 1 Input A 0 B 0 C 0 D 0 LearningRate 0 Neuromodulated F	Node 2 Input A 0 B 0 C 0 D 0 LearningRate 0 Neuromodulated F	Node 3 Input A 0 B 0 C 0 D 0 LearningRate 0 Neuromodulated F	Node 4 Output A 0.2 B 0.6 C 0.4 D 0.3 LearningRate 0.7 Neuromodulated T	Node 5 Hidden A 0 B 0 C 0 D 0 LearningRate 0 Neuromodulated F	Node 6 Neuromodulating A 0 B 0 C 0 D 0 LearningRate 0 Neuromodulated F

Connection Genes

Innov'n# 1 In 1 Out 4 Weight 0.7 Enabled Neuromod'g F	Innov'n# 2 In 2 Out 4 Weight -0.5 Disabled Neuromod'g F	Innov'n# 3 In 3 Out 4 Weight 0.5 Enabled Neuromod'g F	Innov'n# 4 In 2 Out 5 Weight 0.2 Enabled Neuromod'g F	Innov'n# 5 In 5 Out 4 Weight 0.4 Enabled Neuromod'g F	Innov'n# 6 In 1 Out 5 Weight 0.1 Enabled Neuromod'g F	Innov'n#11 In 4 Out 5 Weight 0.6 Disabled Neuromod'g F	Innov'n#15 In 2 Out 6 Weight 0.9 Enabled Neuromod'g F	Innov'n#16 In 6 Out 4 Weight 0.3 Enabled Neuromod'g T	Innov'n#17 In 3 Out 6 Weight 0.8 Enabled Neuromod'g F

Network (Phenotype)



3.2 Neuromodulated multiobjective non-speciated NEAT

The proposed neuromodulated multiobjective non-speciated NEAT (NM-MO-NS-NEAT) method augments NEAT-MODS with neuromodulation, but removes the speciation innovation protection that is no longer considered necessary, as its function is performed by neuromodulation.

3.3 Simulation

In order to demonstrate the effectiveness of the proposed NM-MO-NS-NEAT method, it, NEAT-MODS, and neuromodulated NEAT-MODS each evolve neural networks to control simulated robots that are applied to a basic autonomous foraging and maze task. Natural foraging is a task that animals must perform to obtain food resources, and similarly a task fully autonomous machines would be required to perform in order to acquire fuel and material for self-repair when human operators are unable to provide such things. The maze aspect simulates obstacles encountered during operation.

3.3.1 The simulated robot

A differential wheel robot is simulated to demonstrate the effectiveness of neuromodulation when applied to multiobjective evolutionary neurocontrollers. The robot test platform is similar to the Khepera used in [1]. The simulated robot consists of a 20 cm radius body with 2 motorized wheels. As in [1], the robot has 8 obstacle range sensors configured at [– 167 – 64 – 38.5, – 13, 1338.5, 64, 167] degrees. These are neural network inputs 1–8. The maximum obstacle range that can be sensed is 2 m, and sensor output is on a range of [0, 1], zero being maximum distance. Unlike [1], the robot uses a radar-style rotational range sensor to determine goal location, maximum range is 25 m, the range being given on [0, 1] in the same manner as the obstacle sensors. The goal location radar sensor range and angle are neural network inputs 9 and 10. Sensor noise is not simulated, and no sensors can penetrate walls.

3.3.2 The neurocontrollers

In NEAT, NEAT-MODs and here, the initial network topology is minimal, and there are 10 input neurons directly connected to two output neurons through 20 initially randomly assigned weights as shown in Fig. 9. The population of the candidate neurocontrollers used is 44 as in [1].

The network topology is then augmented in a minimalist fashion, by a maximum of one node gene and one connection per individual per generation. The process outlined in Algorithm 1 (both with and without the speciation steps), is then used to evolve the candidate neurocontrollers, with neuromodulation being performed as described in Sect. 2.4 during the tournament step.

3.3.3 Simulated arena

The test arena is shown in Fig. 10, and is a simple rectangular room with five additional walls that act as obstacles for the robot as it tries to acquire all goals. The initial robot position is marked with an 'x'. The target goal locations are marked with an asterisk '*'. The targets are distributed so that they are not all visible to the robot at any given time,

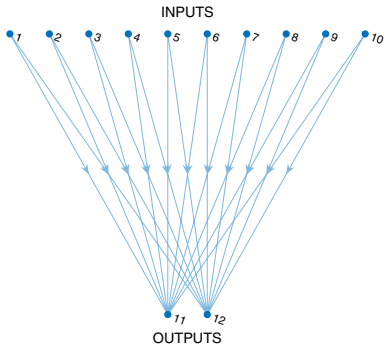


Fig. 9 Initial network topology

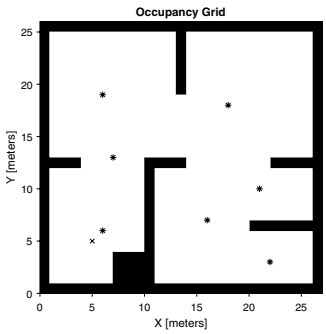


Fig. 10 Test arena

and there are locations where the robot cannot see the next target goal.

3.3.4 Objective functions

For each generation, the individual candidate neurocontrollers are evaluated based on their performance driving the simulated robot through the test arena. As in [1], the robot's starting location is the same for each candidate neurocontroller. The fixed start position prevents bias in the fitness values that would be introduced should there be varying travel times between the starting position and the first goal position. Both objective functions use the number of time steps required to complete the journey to the next goal as divisor. Thus, starting positions further from the first goal would result in lower fitness values for each objective. Different starting positions, with the same distance but different orientation with respect to the starting position, would require the robot to turn, potentially requiring more time steps to follow the same distance. As such, different starting positions for each generation would mean that fitness values could not be compared within or across generations. The maximum number of time-steps without reaching a target is 40. Upon reaching a target, the neurocontroller is given 40 more time-steps to reach the next target. The robot's neurocontroller receives input from the sensors at each time-step. The value of 40 time-steps was determined experimentally by running many simulations of a reduced number of generations (to save simulation time), as opposed to the 150 generations used to generate the results presented here. Simulations with 35 or fewer time-steps did not give the robots enough time to reach each target. Values of 45 and more allowed the robots too much time to reach the targets, wasting computational time. Learning factors of 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 100, and 200 were also tried, with 0.05 being chosen as the best compromise. The neurocontroller's outputs are the robot's speed and heading commands. As in [1], the candidate neurocontrollers are evaluated using two objective functions. Collision with a wall is undesirable. It indicates an unfit individual, and therefore results in termination of the candidate neurocontrollers in the simulation, and a value of zero for both fitness functions, effectively removing it as a possible parent for the next generation. For comparative purposes, the two objective functions F_1 and F_2 are the same as those used in [1]:

$$F_1 = \frac{\sum_{t=1}^{N} \left[V_t \left(1 - \sqrt{\theta} \right)_t (1 - I)_t \right]}{N} \tag{8}$$

$$F_2 = \frac{\sum_{t=1}^{N} f_{2_t}}{N} \tag{9}$$

$$f_{2_t} = \begin{cases} H & robot\ reaches\ target \\ \frac{1}{1+d_t} & otherwise \end{cases} \tag{10}$$

where V is the robot speed, θ is the difference between wheel velocities (the magnitude of the robot's angular heading, $0 \leq \theta \leq 1$), I is the normalized ($0 \leq I \leq 1$) activation value of the obstacle sensor with the highest value (I is zero when no obstacle is sensed); H is the score for reaching a target or goal (a value of 500 was used in [1], here a value of 500 is also used), d is the distance from the robot to the closest target, and N is the total number of time steps that the neurocontroller kept the robot alive. The purpose of F_1 is to promote speed and direct motion while avoiding obstacles, but without any destination. The purpose of F_2 is to reach as many targets as possible, without concern for obstacle avoidance. Objective function F_1 rewards forward speed, but punishes candidate robots that turn or that move close to walls. Both turning and close proximity to walls may be required to maximize objective F_2. Increasing the robot's forward speed increases the distance it travels between scans of the targets. If the distance travelled becomes too large between the radar scans, the robot will miss the target. Thus rewarding forward speed can have an adverse effect on maximizing objective

function F_2. As such F_1 and F_2 are considered contradictory, and a Pareto-optimal set of neurocontrollers should exist [1].

3.4 Results

As evolutionary algorithms are stochastic in nature, repetitive runs (a run being a random seeded completion of Algorithm 1) are generally used to obtain statistically relevant results. As in the infinite monkey theorem, given an infinite amount of time, a monkey hitting keys at random on a keyboard will surely type any given text (for example the complete works of William Shakespeare), given an infinite amount of time, evolutionary algorithms will arrive at the optimal solution. When applied to real world problems, an evolutionary algorithm would be expected to achieve a solution with reasonable fitness(es) within a practical number of generations, or preferably the minimum number of generations. Therefore, a superior evolutionary algorithm is one that arrives at a good enough, best, or better solution in the least amount of (computational) time, or generations of evolution. Thus, the objective of these experiments is not necessarily to produce the most optimal neurocontrollers, but to produce better neurocontrollers in fewer generations. Hence, over a set of repetitive runs, the superior evolutionary algorithm has the greatest mean fitness values, and the smallest standard deviation in the mean, indicating that it is more likely to come up with the best solution in fewer generations. The desired outcome is a generalized improvement. Here, and in both [1] and [17], 30 independent runs of 150 generations were performed to demonstrate the performance improvement of multiobjective neuromodulation.

The simulation and algorithms were coded in Matlab. Parameters for each of the neurocontrollers are presented in Table 1. These values were determined based on those in [1, 17, 21], with the exception of the probability of neuromodulation and learning factor. The probability of neuromodulation (that when a node is added it will be a neuromodulating node as opposed to a standard neuron) was determined experimentally, along with the learning factor and the number of time-steps a robot has to reach a target (as previously discussed in Sect. 3.3.4). Uniform perturbation is the modification of a value by the randomly decided addition or subtraction of a constant value (amount of uniform perturbation).

The proposed architecture is compared to NEAT-MODS, and demonstrated by simulation of a differential wheeled robot applied to an autonomous foraging task in a maze. Evolved neurocontrollers are tasked with acquiring seven target goals within the maze. The simulations exhibit the effectiveness of neuromodulation on the evolved neurocontrollers, and the improved performance given by augmenting NEAT-MODs with neuromodulation. Table 2 exhibits the performance of the NEAT-MODS, Neuromodulated NEAT-MODS (NM-NEAT-MODS), and Neuromodulated Multiobjective Non-Speciated NEAT (NM-MO-NS-NEAT) simulated controllers. The NM-MO-NS-NEAT has the greatest (fittest and therefore best performing) mean F_1 value, has the greatest mean and maximum F_2, in comparison to both NEAT-MODS and the neuromodulated NEAT-MODS neurocontrollers. The NM-MO-NS-NEAT also has a smaller standard deviation than NM-NEAT-MODS in both fitness functions. This is a generalized improvement. As NM-MO-NS-NEAT has improved mean fitness values in comparison to NEAT-MODS, then given any random initial seed, NM-MO-NS-NEAT is more likely to produce fitter neurocontrollers than NEAT-MODS. And NM-MO-NS-NEAT is also more likely to produce a good solution in fewer generations. The conclusion that can be drawn from these results is that neuromodulated learning has allowed the algorithm to improve on the genetic solution by changing the neural network weights to better performing values, producing a fitter neurocontroller, during operation, with no supervision. NM-MO-NS-NEAT also uses on average fewer nodes and connections than NM-NEAT-MODS.

Table 1 Neurocontroller parameters

Parameter	Value
Probability of neuromodulation	0.5
Probability of weight mutation	0.8
Probability of uniform perturbation	0.9
Probability of disabled connection	0.75
Probability of mutation without crossover	0.55
Interspecies mating rate	0.1
Probability of new node	0.23
Probability of new connection	0.7
Amount of uniform perturbation	0.001
Crossover gene replacement probability	0.25
Learning factor	0.05

A two-sided Wilcoxon (Mann-Whitney) U-test was performed on the best fitness values in the final generation using Matlab's ranksum function. When comparing NM-MO-NS-NEAT against NEAT-MODS, the test returned a p-value of 0.001302 for F_1 and 7.043E-07 for F_2. Both these values indicate a rejection of the null hypothesis (that the NEAT-MODS and NM-MO-NS-NEAT best fitnesses are samples from continuous distributions with equal medians), at the 1% significance level (99% confidence interval).

When NM-MO-NS-NEAT (no speciation) is compared with NM-NEAT-MODS (with speciation), the two-sided Wilcoxon test returned a p-value of 0.2226 for F_1. This value indicates a failure to reject the null hypothesis at the 1% significance level. Thus we must assume that the NM-NEAT-MODS and NM-MO-NS-NEAT best F_1 fitnesses are samples from continuous distributions with equal medians).

Table 2 Neurocontroller performance comparison

Algorithm (30 runs)	F_1			F_2			Added nodes (mean)	Added edges (mean)
	Mean	Max	σ	Mean	Max	σ		
NEAT-MODS	0.02017	0.05153	0.008777	18.98	50.28	15.11	20.07	52.10
NM-NEAT-MODS	0.02363	0.05963	0.01005	26.04	50.28	18.86	23.87	74.87
NM-MO-NS-NEAT	0.02626	0.04840	0.009525	42.11	50.31	15.66	21.07	63.67

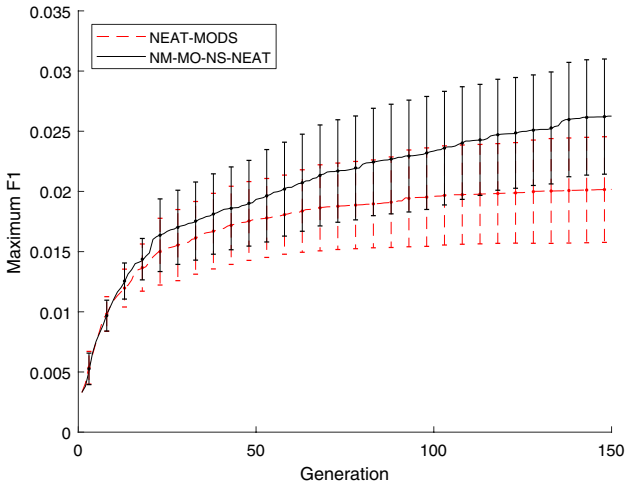


Fig. 11 F_1 performance for NM-MO-NS-NEAT and NEAT-MODS controllers versus generation

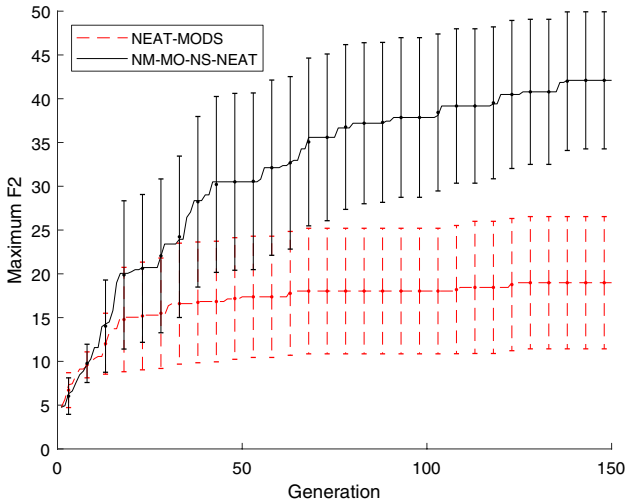


Fig. 12 F_2 performance for NM-MO-NS-NEAT and NEAT-MODS controllers versus generation

Here, we are not claiming that NM-MO-NS-NEAT is superior in performance to NM-NEAT-MODS. Instead, we are demonstrating that speciation is unnecessary when neuromodulation is used with NEAT-based neuroevolution. Equal performance in F_1 objective fitness does not negate this argument. The two-sided Wilcoxon test returned a p-value of 5.265E-05 for F_2. Again, the null hypothesis that the NM-NEAT-MODS and NM-MO-NS-NEAT best F_2 fitnesses are samples from continuous distributions with equal medians can be rejected at the 1% significance level (99% confidence interval).

This demonstrates that not only does neuromodulation produce less complex and fitter neurocontrollers than neurocontrollers without neuromodulation, but that speciation is unnecessary when neuromodulation is used.

The increased performance provided by neuromodulation is exhibited statistically as the standard deviation (error bars), and mean of 30 trials of 150 generations of the NM-MO-NS-NEAT and NEAT-MODS neurocontrollers in Figs. 11 and 12. Figure 11 shows performance objective F_1 of both NM-MO-NS-NEAT (solid line) and NEAT-MODS (dashed line) controllers for each generation. The neuromodulated multiobjective neurocontrollers have better mean F_1 values after approximately the first 20 generations in comparison to those without neuromodulation.

Figure 12 shows performance objective F_2 of both neuromodulated (NM-MO-NS-NEAT, solid line) and non-neuromodulated (NEAT-MODS, dashed line) controllers for each generation. Both types of controllers exhibit similar objective F_2 performance for the first 12 generations. The NM-MO-NS-NEAT controllers display improved objective F_2 performance over the NEAT-MODS after the first 20 generations.

In order to show that speciation is not necessary when using neuromodulated controllers, the performance of NM-MO-NS-NEAT and neuromodulated NEAT-MODS neurocontrollers is compared in Figs. 13 and 14. Figure 13 shows performance objective F_1 of both non-speciated (NM-MO-NS-NEAT, solid line) and speciated neuromodulated NEAT-MODS (NM-NEAT-MODS, dashed line) controllers for each generation. The non-speciated neuromodulated multiobjective controllers have similar or better mean F_1 values over all generations in comparison to those with speciation.

Figure 14 shows performance objective F_2 of both non-speciated (NM-MO-NS-NEAT, solid line) and speciated neuromodulated NEAT-MODS (NM-NEAT-MODS, dashed line) controllers for each generation. The NM-MO-NS-NEAT neurocontrollers display improved objective F_2 performance over the speciated controllers after the first 20

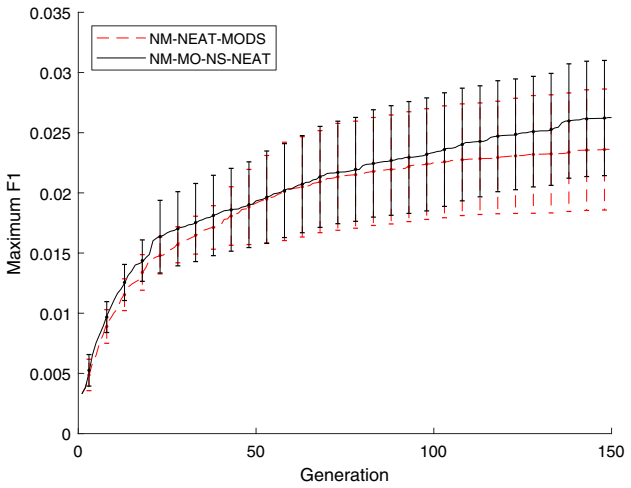


Fig. 13 F_1 performance for NM-MO-NS-NEAT and NM-NEAT-MODS controllers versus generation

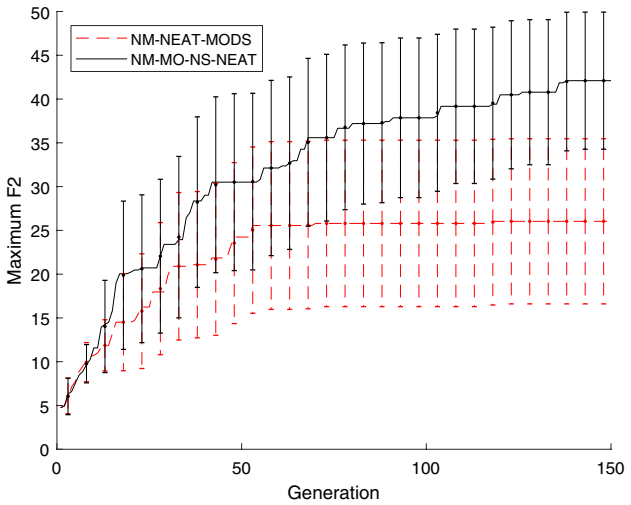


Fig. 14 F_2 performance for NM-MO-NS-NEAT and NM-NEAT-MODS controllers versus generation

generations. Speciation is intended as a system by which candidate topologies are given an opportunity to adapt their weights to a better solution through mutation. When speciation is combined with neuromodulation, poor candidate topologies that have not increased their fitness though neuromodulated weight adjustment are still kept in the offspring population by speciation protection. Thus, unfit topology solutions can occupy positions that could be occupied by candidates that have evolved to fitter topologies, impeding the rate of evolution.

Figures 15 and 16 show the final robot position and robot path for the fittest non-neuromodulated (NEAT-MODS) and neuromodulated (NM-MO-NS-NEAT) individuals for objective functions F_1 and F_2. These figures show the robots

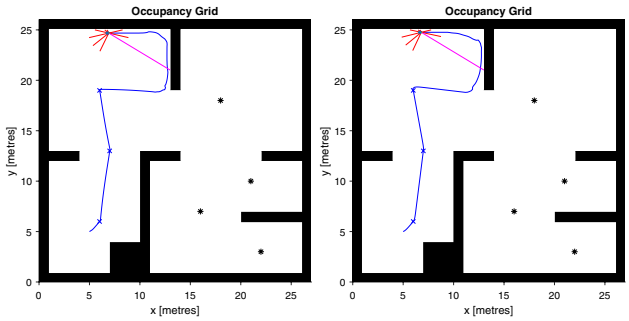


Fig. 15 Robot paths for fittest F_1 performance objective. NEAT-MODS (left), and NM-MO-NS-NEAT (right). The maximum F_1 NEAT-MODS value (0.05153) is slightly greater than the maximum NM-MO-NS-NEAT F_1 value (0.04840), as presented in Table 2. These figures show the fittest robots over all generations in all the runs, not the average. Note also that robot speed is not visible here

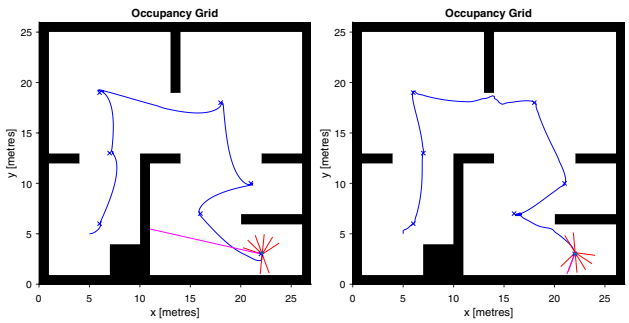


Fig. 16 Robot paths for fittest F_2 performance objective. NEAT-MODS (left), and NM-MO-NS-NEAT (right). Here, the NM-MO-NS-NEAT maximum F_2 value (50.31) is slightly greater than the maximum NEAT-MODS F_2 value (50.28), as presented in Table 2. These figures show the fittest robots over all generations in all the runs, not the average. Note also that robot speed is not visible here

that achieved the maximum fitness over all generations in all the runs, not the average. Note also, that robot speed is part of the F_1 fitness equation, but the effect of speed is not visible in these figures. Both of the fitness functions include division by the total number of possible time-steps, so that a robot that completes the maze in fewer time-steps achieves a greater fitness function. Greater speed can reduce the amount of time-steps required to complete the maze, and thus both F_1 and F_2 fitnesses are a indirectly a function of speed. Hence, two sample plots of apparently similar robot paths may be associated with considerably different fitness values.

The robot orientation is displayed as a stylized robot symbol with its obstacle range sensor traces and 'radar-style' detector. The targets are displayed as asterisks before being acquired, and after acquisition they are 'x' symbols. Figure 15 shows the robot paths for the evolved non-neuromodulated and neuromodulated networks with the best performance objective F_1. The calculation of this objective

function does not include the target goals, and therefore candidate neurocontrollers do not attempt to visit them. The robots have reached the first three targets, but this is a result of being crossbred within a general population that is also evolving to maximize objective F_2. The crossover mechanism of the evolutionary algorithm does not discriminate between F_1 or F_2 fitness functions when choosing candidates for crossbreeding, and thus candidates that are fit in F_1 can crossbreed with candidates that are fit in F_2. These are the respective maximum values of performance objective F_1, and both neurocontrollers display similar performance as in Table 2.

Figure 16 shows the robot path for the non-neuromodulated (NEAT-MODS) and neuromodulated (NM-MO-NS-NEAT) neurocontrollers with the fittest performance objective F_2. Here the best neuromodulated robot follows a more direct path than the best non-neuromodulated robot. Diversions from the direct path of the non-neuromodulated neurocontroller are marginally larger than those of the neuromodulated controller. The direct motion of the robot path plot travelling from each target goal to the next indicates not only that it has maximized objective F_2, but also that it has maximized objective F_1, exhibiting the effectiveness of evolving multiple objectives concurrently within a common population (multiobjective optimization). Here the objective of avoiding walls (F_1) has been maximized, but not at the cost of reaching targets (objective F_2), as shown between the third and fourth targets, where the fittest neuromodulated robot has come close to the top vertical wall.

Figures 17 and 18 show the evolved neural network structure for the non-neuromodulated (NEAT-MODS) and neuromodulated (NM-MO-NS-NEAT) neurocontrollers with the fittest performance objective F_2 over all runs and generations. Non-neuromodulated neurons are circular shaped nodes with regular connections being solid lines. Connections that have been disabled by the NEAT mutation algorithm are shown as a dashed line. Neuromodulating neurons are diamond shaped, and neuromodulating connections are bold. Neuromodulated neurons are square shaped. The values displayed for each edge are the weight values for the synapse associated with their respective edge. The inputs are numbered one through ten, the outputs eleven and twelve.

In Fig. 18 there are two neuromodulating neurons (nodes 14 and 15), and four neuromodulated neurons (nodes 11, 12, 13 and 14). Node 14 is both neuromodulating and neuromodulated. In Fig. 18 it can also be seen that Node 11 has 12 input connections (including the recurrent one), Node 12 has 11 input connections (including the recurrent one), Node 13 has 13 input connections (including the recurrent one), and Node 14 has six non-neuromodulating input connections. Therefore there are a total of 42 connections each with a weight that is adapted by neuromodulation over the four neuromodulated nodes.

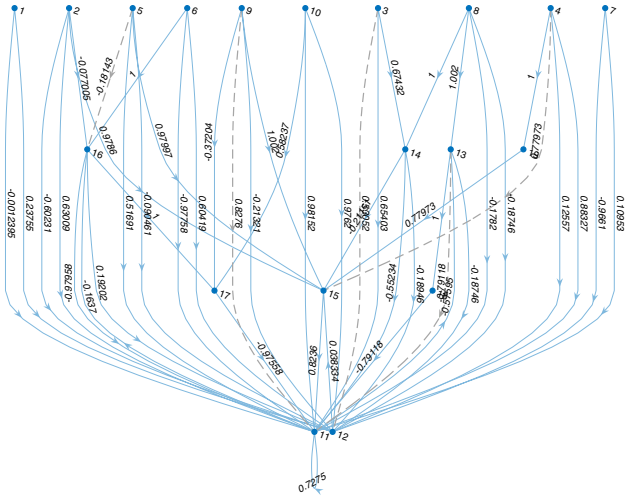


Fig. 17 Evolved NEAT-MODS neural network structure for fittest F_2 performance objective. This figure shows the fittest neurocontroller over all generations in all the runs, and is not representative of the average

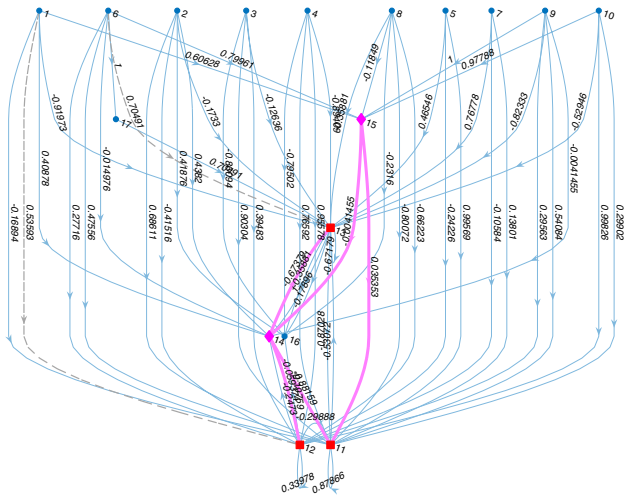


Fig. 18 Evolved NM-MO-NS-NEAT neural network structure for fittest F_2 performance objective. This figure shows the fittest neurocontroller over all generations in all the runs, and is not representative of the average

Table 3 shows the number of added neurons and connections for the non-neuromodulated and neuromodulated controllers with the best F_2 performance objective. In Table 2, the best non-neuromodulated controller has an F_2 fitness of 50.28, which is very close to the value of 50.31 that the best neuromodulated neurocontroller has achieved. The neuromodulated controller has 2 fewer neurons, but 11 more connections.

The following figures are intended to illustrate graphically the effects of neuromodulation on the neuromodulated neurons. The figures happen to show the fittest F_2

Table 3 Neurocontroller topology statistics for best F_2 performance objective

Algorithm (30 runs)	Added neurons	Neuromodulating neurons	Neuromodulated neurons	Added connections	Neuromodulating connections
NEAT-MODS	7	0	0	30	0
NM-MO-NS-NEAT	5	2	4	41	5

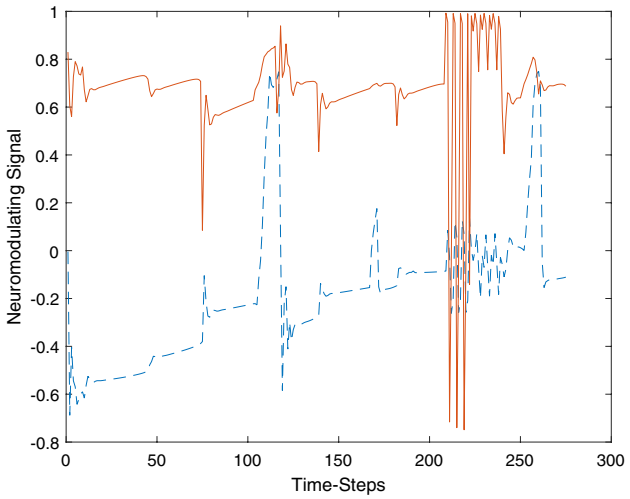


Fig. 19 Neuromodulating neuron's neuromodulation signal

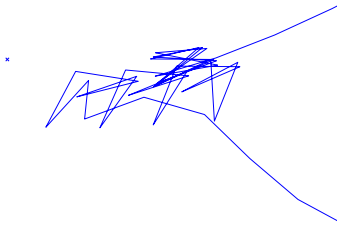


Fig. 20 Detail of final target of robot path for best neuromodulated F_2 performance objective

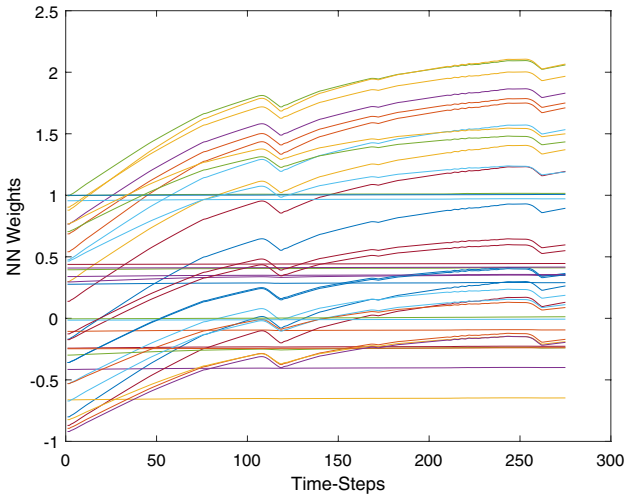


Fig. 21 Neuromodulated neuron's neuromodulated weights. Close inspection of the weights that appear to be unchanging will reveal that they are in fact changing very slowly

neurocontroller, but any neuromodulated neurocontroller with some interesting behaviour would have sufficed.

Figure 19 presents the neuromodulation signal (with respect to time-step) produced by the neuromodulating neurons. In Fig. 19, spikes in the neuromodulating signal can be seen at times when the robot encounters a situation more challenging than moving freely towards the next target. There are spikes at locations when the robot encounters obstacles, and at times the robot has difficulty getting close enough to the targets to acquire them. The neurocontroller has had difficulty reaching the sixth target (as can be seen in the NM-MO-NS-NEAT robot path in Fig. 16, and is detailed in Fig. 20). The neuromodulating neuron is attempting to compensate for this problem, which is exhibited as oscillation around the 220th time-step.

Figure 20 shows an expanded portion of the NM-MO-NS-NEAT robot path exhibited in the right half of Fig. 16. The robot can be seen approaching target 6 from the top right of the figure. Target 6 is the leftmost object in the figure, a small 'x'. As the robot nears target 6, it zig-zags in an attempt to acquire the target. After several attempts, the robot acquires the target, and continues on its mission to acquire the remaining target. It then exits at the bottom right of the figure.

The neuromodulated weight values with respect to the time-steps for each of the 42 plastic connections are shown in Fig. 21. This shows the effect of the neuromodulation signals presented in Fig. 19 on the neuromodulated weights. Here, the neuromodulated weights can be seen to be learning—adapting to better values as the robot moves through the maze, based on the neuromodulating signals presented in Fig. 19. The neuromodulated weights appear to be converging towards a final value, suggesting that the neurocontrollers have learned to operate the robot in the maze to the best of their ability for their neural network topology.

4 Discussion and future work

Multiobjective evolutionary algorithms such as NSGA-II [3] select individuals that dominate each objective (and objectives), and as such maintain a population of fitter individuals of each objective. As multiobjective evolutionary algorithms

maintain a population of the fittest individuals of each objective, (and given that the objectives are sufficiently conflicting) good diversity is ensured [8]. The objectives used so far (as presented in Sect. 3.3.4) are composite objectives in that each is a function of separate atomic objectives, an atomic objective being one that cannot be further separated into component objectives. For example, 'maximize forward velocity' is an atomic objective, as the wheel position/velocity from an encoder cannot be further broken down into components. In theory, simplifying the composite objectives by breaking them up into separate atomic objectives will increase the efficiency of the evolutionary algorithm, as increasing the number of objectives increases diversity, and therefore the probability of finding a good solution in fewer generations.

It is proposed that the NM-MO-NS-NEAT algorithm be augmented to include the evolution of a number of the metaparameters. Encoding the simulation metaparameters in the evolutionary algorithm would allow these to be optimized at the same time as the network parameters. This will not only find the best metaparameters, but may also improve network parameter performance and help reduce the number of generations required to evolve good controllers.

5 Conclusions

A novel architecture for neuromodulated multiobjective topology and weight evolution of artificial neural networks is proposed. Combining neuromodulation with multiobjective neuroevolution provides a powerful tool for exploring the search space. This combination gives the unique ability to test the search space with a genetic operator, and then improve upon these results using neuromodulated learning to adapt the network during operation, during each tournament. At the end of each tournament, candidates do not need to be protected through speciation, as they are optimized to compete with the entire population. This approach allows exploration of the entire search space, and fine tuning to find each local maximum, until a solution with the global (or at least a more global) maximum is found.

The proposed neuromodulated multiobjective non-speciated NEAT (NM-MO-NS-NEAT) architecture is demonstrated by simulation of a differential wheeled robot applied to an autonomous foraging task in a maze (to be consistent with the work presented in [1]). Evolved neurocontrollers are tasked with acquiring seven target goals within the maze. The simulations compare the performance of neuromodulated speciated NEAT-MODS (NM-NEAT-MODS), and NEAT-MODS with NM-MO-NS-NEAT. The simulations exhibit the effectiveness of neuromodulation on the evolved neurocontrollers, and the improved performance given by augmenting NEAT-MODs with neuromodulation. On average NM-MO-NS-NEAT evolves neurocontrollers with greater fitnesses and smaller or similar fitness standard deviations, in fewer generations than either NEAT-MODS or NM-NEAT-MODS. Since the objective of these experiments is not necessarily to produce the most optimal neurocontrollers, but to produce better neurocontrollers in fewer generations, and as NM-MO-NS-NEAT is more likely to come up with the best solution in fewer generations, it is considered to be the superior algorithm here. It is also shown that speciation is unnecessary when neuromodulation is used with NEAT-based neuroevolution, as neuromodulation produces topological innovation.

The results presented show the superiority of multiobjective neuromodulated neuroevolution over multiobjective neuroevolution alone. Neuromodulated robots have been shown to achieve better mean F_1 and F_2 values at each generation. The results indicate that even two neuromodulating neurons can improve controller performance. Using NEAT-MODS alone, fine-grain adjustment of the connection weights requires mutation, which only occurs between generations, when offspring are produced. By including neuromodulation, the weights can be adjusted continuously during the lifetime of each generation using synaptic plasticity. The results demonstrate that the combination of neuromodulation with multiobjective NEAT in NM-MO-NS-NEAT gives an effective and efficient tool for generating neurocontrollers by facilitating learning while the neurocontrollers are evolving.

References

1. Abramovich O, Moshaiov A (2016) Multi-objective topology and weight evolution of neuro-controllers. In: 2016 IEEE congress on evolutionary computation (CEC), pp 670–677. https://doi.org/10.1109/CEC.2016.7743857
2. Deb K (2001) Multi objective optimization using evolutionary algorithms. Wiley, New York
3. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197. https://doi.org/10.1109/4235.996017
4. Doya K (2002) Metalearning and neuromodulation. Neural Netw 15(4–6):495–506. https://doi.org/10.1016/S0893-6080(02)00044-8
5. Floreano D, Dürr P, Mattiussi C (2008) Neuroevolution: from architectures to learning. Evol Intell 1(1):47–62. https://doi.org/10.1007/s12065-007-0002-4
6. Hebb DO (1949) The organization of behavior, a neuropsychological theory. Wiley, New York
7. Katz PS, Harris-Warrick RM (1999) The evolution of neuronal circuits underlying species-specific behavior. Curr Opin Neurobiol 9(5):628–633. https://doi.org/10.1016/S0959-4388(99)00012-4
8. Knowles JD, Watson RA, Corne DW (2001) Reducing local optima in single-objective problems by multi-objectivization. In: International conference on evolutionary multi-criterion optimization. Springer, pp 269–283
9. Mandziuk J, Rajkiewicz P (2016) Neuro-evolutionary system for FOREX trading. In: 2016 IEEE congress on evolutionary

computation, CEC 2016, pp 4654–4661. https://doi.org/10.1109/CEC.2016.7744384
10. Nayak SC, Misra BB (2018) Estimating stock closing indices using a GA-weighted condensed polynomial neural network. Financial Innov 4(1):1–22. https://doi.org/10.1186/s40854-018-0104-2
11. Niv Y, Joel D, Meilijson I, Ruppin E (2002) Evolution of reinforcement learning in uncertain environments: a simple explanation for complex foraging behaviors. Adapt Behav 10(1):5–24. https://doi.org/10.1177/1059-712302-010001-01
12. Norouzzadeh MS, Clune J (2016) Neuromodulation improves the evolution of forward models. In: Proceedings of the 2016 on genetic and evolutionary computation conference—GECCO '16, pp 157–164. https://doi.org/10.1145/2908812.2908837
13. Prados D, Kak S (1989) Neural network capacity using delta rule. Electron Lett 25(3):197–199. https://doi.org/10.1049/el:19890142
14. Reisinger J, Bahceci E, Karpov I, Miikkulainen R (2007) Coevolving strategies for general game playing. In: 2007 IEEE symposium on computational intelligence and games, pp 320–327. https://doi.org/10.1109/CIG.2007.368115
15. Richard K. Belew, McInerney J, Schraudolph NN (1991) Evolving networks: using the genetic algorithm with connectionist learning. In: Proceedings of the second artificial life conference, pp 511—-547
16. Showalter I, Schwartz HM (2004) A growing and pruning method for a history stack neural network based adaptive controller. In: Proceedings of the IEEE conference on decision and control, vol 5, pp 4946–4951. https://doi.org/10.1109/CDC.2004.1429590
17. Silva F, Urbano P, Christensen AL (2014) Online evolution of adaptive robot behaviour. IGI Global, Hershey, pp 59–77
18. Silva F, Urbano P, Oliveira S, Christensen AL (2012) odNEAT: an algorithm for distributed online, onboard evolution of robot behaviours. Artif Life 13:251–258. https://doi.org/10.7551/978-0-262-31050-5-ch034
19. Soltoggio A, Bullinaria JA, Mattiussi C, Dürr P, Floreano D (2008) Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In: Artificial life XI: proceedings of the 11th international conference on simulation and synthesis of living systems (ALIFE 2008), vol 2, pp 569–576. https://doi.org/10.1016/S0269-7491(01)00278-0
20. Stanley KO, Bryant BD, Miikkulainen R (2005) Real-time learning in the NERO video game. IEEE Trans Evol Comput 9(6):653–668
21. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127. https://doi.org/10.1162/106365602320169811
22. Stanley KO, Miikkulainen R (2004) Competitive coevolution through evolutionary complexification. J Artif Intell Res 21:63–100. https://doi.org/10.1613/jair.1338
23. van Willigen W, Haasdijk E, Kester L (2013) A multi-objective approach to evolving platooning strategies in intelligent transportation systems. In: Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference—GECCO '13. ACM Press, New York, NY, USA, pp 1397–1404. https://doi.org/10.1145/2463372.2463534