# Evolving Transferable Artificial Neural Networks for Gameplay Tasks via NEAT with Phased Searching

Will Hardwick-Smith, Yiming Peng, Gang Chen, Yi Mei, Mengjie Zhang

School of Engineering and Computer Science
Victoria University of Wellington
Wellington, New Zealand
{whardwicksmith}@gmail.com
{yiming.peng,gang.chen,yi.mei,mengjie.zhang}@ecs.vuw.ac.nz
http://ecs.victoria.ac.nz/Groups/ECRG/

**Abstract.** NeuroEvolution of Augmenting Topologies (NEAT) has been successfully applied to intelligent gameplay. To further improve its effectiveness, a key technique is to reuse the knowledge learned from source gameplay tasks to boost performance on target gameplay tasks. We consider this as a Transfer Learning (TL) problem. However, Artificial Neural Networks (ANNs) evolved by NEAT are usually unnecessarily complicated, which may affect their transferability. To address this issue, we will investigate in this paper the capability of *Phased Searching* (PS) methods for controlling ANNs' complexity while maintaining their effectiveness. By doing so, we can obtain more transferable ANNs. Furthermore, we will propose a new Power-Law Ranking Probability based PS (PLPS) method to more effectively control the randomness during the simplification phase. Several recent PS methods as well as our PLPS have been evaluated on four carefully-designed TL experiments. Results show clearly that NEAT can evolve more transferable and structurally simple ANNs with the help of PS methods, in particular PLPS.

**Keywords:** Transfer Learning, Intelligent Gameplay, Mario AI, Artificial Neural Networks, NEAT, Phased Searching, Power-Law

## 1 Introduction

With well-defined rules, easily controllable environments and clear success criteria, computer games have been increasingly considered favorable for Artificial Intelligence (AI) research [10, 2, 11]. Among all the different AI technologies, *Artificial Neural Networks* (ANNs) are popular choices for establishing the intelligent game playing strategies since they are highly adaptive on nonlinear problems [10]. Moreover, with the capability of learning the ANN's weights and its topology simultaneously, NeuroEvolution of Augmenting Topologies [14] (NEAT) has shown great potential in successfully playing a wide range of games, such as Checkers [2], Car Racing [1], Othello [11], Ms. Pacman [12], Atari Games [5], etc. However, many of these NEAT-based approaches for gameplay require the agent to

be trained from scratch for every newly encountered gameplay task, potentially affecting the efficiency and effectiveness of these approaches [20].

As a major solution to the above issue, *Transfer Learning* (TL) is capable of obtaining a solution from a *source* task and transferring the solution to improve learning on a *target* task [9, 13]. It is assumed that the source task is previously known and the target task presents related but unseen problems to be encountered in the future. In line with this idea, the usefulness of applying TL to NEAT for intelligent game playing has been fruitfully explored in the literature [17, 20, 18].

Different from existing works, we focus on a new context of TL in this research. Specifically, both the source and target tasks are assumed known to us. However, it may not be wise to directly tackle the target task as it is more challenging than the source task. A better strategy based on TL is to first identify a solution to the source task by using NEAT, which is subsequently transferred to build a new solution that can perform the target task effectively. For example, the source task could be utilized to teach a NEAT agent to avoid enemies in a computer game. Afterwards the agent can continue to learn how to quickly reach its target while accruing more credits on its way in the target task. Clearly if NEAT can reliably learn transferable ANNs, it is highly likely for difficult gameplay tasks to be conquered effectively and efficiently.

One major hurdle of our TL approach lies in the fact that NEAT is designed to increasingly complexify the topology of evolved ANNs. The learned ANN can be highly complicated and strongly tied to irrelevant features of the source task and hence may not be of any real use in the target task. In fact, our empirical study shows that, without controlling their complexity, ANNs evolved by NEAT from the source target may actually hinder effective learning in the target task. To address this technical problem, we will investigate the *Phased Searching* (PS) [3] techniques that can constantly simplify evolved ANNs while maintaining NEAT's performance at a good level. We believe that, by doing so, the resulting ANNs will be more transferable. To the best of our knowledge, few studies have ever considered PS for learning transferable ANNs in NEAT. Rather than following simple PS methods in [3], in order to further enhance the transferability of ANNs, a new power-law driven PS (PLPS) technique will also be proposed in this paper.

***Goals***: Motivated by the importance of enhancing TL in NEAT, we aim to develop a new PS based NEAT transfer learning algorithm (PLPS-NEAT-TL) capable of evolving effective and simple ANNs that are transferable for intelligent gameplay. Armed with the new algorithm, we aim to pursue two main research objectives.

- Design a new PS method for evolving transferable (effective and simple) ANNs on various source tasks through NEAT.
- Examine the effectiveness of ANNs evolved by PLPS-NEAT-TL algorithm on various target gameplay tasks.

## 2 Background

In this section, we firstly review NEAT algorithm. Next, we introduces the application of NEAT for TL to intelligent gameplay. Lastly, we discuss various PS methods for NEAT, which are capable of simplifying evolved ANNs.

### 2.1 NEAT

NEAT and its variations have been studied for their abilities to play various games such as Ms. Pacman [12], Robot Soccer [19], and collections of Atari 2600 games [4] [5]. It is proposed by Stanley and Miikkulainen [15] to discover effective ANNs for solving some given problem. In addition to its capability of tunning the network's weights, NEAT is also armed with a powerful weapon of adjusting the network's topology. To achieve this, it searches possible network topologies by starting from a population of minimal architectures and then incrementally adding and modifying neural components (links and neurons).

### 2.2 NEAT for Transfer Learning

Transfer learning is a popular direction for intelligent gameplay research that has been studied in many works [13, 9, 17, 20]. Although NEAT shows a great success in intelligent gameplay, only a limited number of studies considered NEAT for transfer learning. Two related works are proposed in [17] and in [20]. These works successfully combined NEAT and TL to improve learning effectiveness on a soccer simulation game. However, as discussed in Section 1, they shared a different TL context from our work. We believe that NEAT for TL is an important research topic deserving substantial investigation.

### 2.3 Phased Searching for NEAT

The original NEAT is designed to increasingly complexify the ANNs during evolution process. As reported in a few works [3, 16], this may result in unnecessarily complicated solutions which can be vulnerable to over-fitting to one specific task [16]. Hence, the chance for them to be transfered to another task will be compromised.

To address this issue, a key technique called *Phased Searching* (PS) has been proposed by Green [3] and later extended by Tan et al. [16]. Both works observed that PS can evolve substantially simplified ANNs (measured by the number of connections and neurons) with good performance. This suggests that PS tends to find ANN solutions that are minimal while still preserving structure that is necessary for achieving a high performance that rivals that of regular NEAT. These properties makes phased searching ideal for finding ANN solutions that are transferable.

Another similar work attempts to address the issues is called *Blended Search* (BS) proposed by James and Tucker [6], which can be viewed as a special case of PS methods. BS blends complexifying and simplifying mutations in evolution process which results in finding higher performing ANNs than original NEAT for XOR and tic-tac-toe problems [6].

However, existing PS methods rely on a pure random selection strategy for removing links and hidden neurons of evolved ANNs. Such uncontrollable randomness might affect the transferability of solutions. This question is further investigated experimentally in this research. Motivated by this understanding, we attempt to develop a new PS with easily controllable randomness for evolving transferable ANNs.

# 3 Methodology

In this section, we present a new PS based NEAT algorithm for TL (PLPS-NEAT-TL) to effectively evolve ANNs for intelligent gameplay. Firstly, we give an overall design of the algorithm. Next, we depict the principle of the new PS method based on a power-law ranking probability to effectively control randomness in ANN simplification phase.

## 3.1 Overall Design

PLPS-NEAT-TL consists of two learning stages, namely *source task learning* (STL) and *target task learning* (TTL), which are bridged by TL. Fig. 1 illustrates the blueprint of the learning algorithm. As seen in the figure, for STL, we rely the learning on NEAT with a power-law ranking probability based PS method (PLPS) so as to simplify the increasingly complexified topologies evolved by NEAT. By doing so, we can obtain reasonably simpler ANNs at the end of STL. A similar algorithmic description for STL can be found in [16]. Subsequently, TL is conducted where the best solution obtained from STL is used as transferable knowledge to initialize NEAT (without using any PS methods) for TTL.

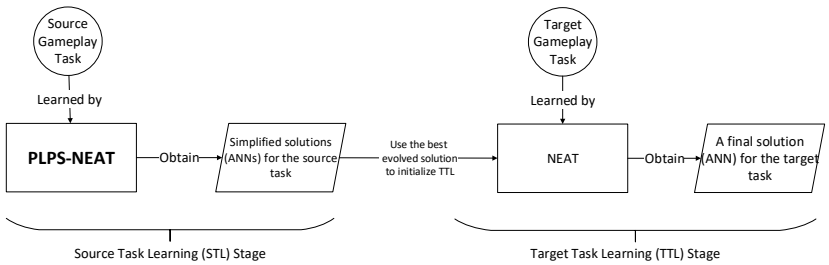



Fig. 1: An overall design of the PLPS-NEAT-TL algorithm.

## 3.2 Power-law Ranking Probability based Phased Searching

In this work, we develop a new PS method for NEAT to evolve structurally simpler and potentially transferable ANNs. Our new PS method is designed to be able to control randomness in ANN simplification phase, which differentiates from existing PS methods whose randomness is purely uncontrollable. For doing so, we define *removal probability* for each gene (including both links and hidden neurons) based on some ranking of the gene.

This ranking is dependent on the type of the gene. For a link gene, we consider the ranking according to the link's absolute weight value. Because a link's weight with a small absolute value (e.g., close to 0) can be considered little contributing to the learning effectiveness, hence such links can be removed without causing much side-effects to the ANN. Additionally, we rank neuron genes based on the number of links connected to each neuron gene. This is straightforward as a neuron of an ANN with more connections indicates that the ANN has more dependencies on the neuron.

Hence, we can determine the remove probability by considering that it follows a power-law relation with respect to the ranking of each gene. The power-law relation can be expressed as,

$$\Phi_g = \begin{cases} \texttt{RANK}(|\omega_g|)^{-1} & , \text{ if } \texttt{IS\_LINK}(g) \\ \texttt{RANK}(\texttt{LINKS}(g))^{-1} & , \text{ if } \texttt{IS\_HIDDEN\_NEURON}(g) \end{cases} \tag{1}$$

where $\Phi_g$ denotes the removal probability for any gene $g$ (NOTE: $\sum_{i=1}^{\texttt{NUMBER\_OF\_GENES}} \Phi_{g_i} = 1$), $\texttt{RANK}(\omega_g)$ extracts the ranking of the link gene $g$ based on its absolute weight value $\omega_g$, and $\texttt{RANK}(\texttt{LINKS}(g))$ returns the ranking of the neuron gene $g$ according to the number of links connected to $g$, i.e., $\texttt{LINKS}(g)$.

We present an algorithmic description of PLPS in Algorithm 1 [1]. Note that, we follow the same functional design for link removal mutation and neuron removal mutation operators presented in [16]. In the simplification phase, these two mutation operations will be conducted sequentially based on the link mutation rate $m_l$ and the neuron mutation rate $m_n$ respectively.

**Algorithm 1** Power-Law Ranking Probability based Phased Searching

**Require:** the population with original individuals $P$, the population size $p$, the link removal mutation rate $m_l$, the neuron removal mutation rate $m_n$
**Ensure:** the population with structurally simplified individuals $P'$

**function** POWER_LAW_PHASED_SEARCH(P)
  **for** k = 1,2,..., p **do**
    $N \leftarrow P[k]$
    $G \leftarrow \texttt{SORT}(\texttt{ALL\_GENES}(N))$
    **for all** $g \in G$ **do**
      **if** $\texttt{IS\_LINK}(g)$ **then**
        $\Phi_g = \texttt{RANK}(g)^{-1}$
      **end if**
      **if** $\texttt{IS\_HIDDEN\_NEURON}(g)$ **then**
        $\Phi_g = \texttt{RANK}(\texttt{LINKS}(g))^{-1}$
      **end if**
    **end for**
    **if** $\texttt{RANDOM}() < m_l$ **then**
      $l \leftarrow \texttt{SELECT\_BY\_PROBABILITY}(g, G, \Phi_g)$
      $N' \leftarrow \texttt{REMOVE\_LINK\_MUTATION}(l, N)$ ▷ Remove the link gene [3]
    **end if**
    **if** $\texttt{RANDOM}() < m_n$ **then**
      $n \leftarrow \texttt{SELECT\_BY\_PROBABILITY}(g, G, \Phi_g)$
      $N' \leftarrow \texttt{REMOVE\_NEURON\_MUTATION}(n, N')$ ▷ Remove the neuron gene [3]
    **end if**
    $P' \leftarrow P' \cup \{N'\}$
  **end for**
  **return** $P'$
**end function**

[1] All source code and detailed parameter settings can be found in *https://github.com/willhs/NEAT-Mario*.

## 4 Experimental Design

In this section, we firstly present a widely-used game benchmark environment, i.e., Mario AI Benchmark, for evaluating the PLPS-NEAT-TL algorithm. Subsequently, we describe a general experiment setup as well as depict four TL experimental designs.

### 4.1 Mario AI Benchmark

Mario AI Benchmark (MIB) is a widely-recognized challenging environment for testing intelligent gameplay algorithms [7]. The various challenges of MIB serves as a good testbed for clearly distinguishing the learning ability of any intelligent gameplay algorithm. Additionally, MIB is highly flexible as most of its game properties (e.g., game level, difficulty, time limit, gaming objectives, etc,) are adjustable. This flexibility makes the implementation of both source and target gameplay tasks straightforward. Hence, we consider MIB ideal for designing our TL experiments.

**Environment Representation** In our experiments, the environment representation (a.k.a, feature) is defined as a sensor grid where each grid cell represents the occupancy of a small region of the screen as proposed in [7]. Originally in [7], the representation uses continuous values for each cell, which may make the learning slow and ineffective [8]. To mitigate this affection, as seen in Fig. 2, we simplify the representation where each cell is represented by one scalar: 0 for nothing, 1 for an environmental feature (like walls), and -1 for an enemy.

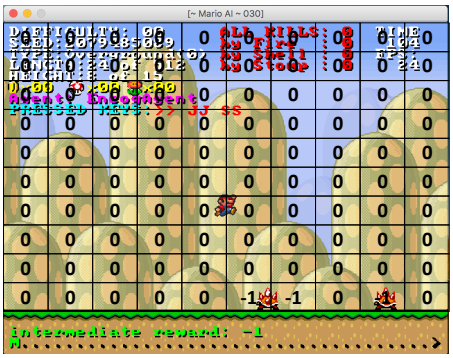



Fig. 2: The environment representation of Mario AI Benchmark.

**Action Representation** In MIB, available actions are defined as: *right* (move Mario to the right), *left*, *down* (duck), *jump*, and *speed* (when held makes movements to the left or right further). These actions can be executed in parallel; for

instance, the right and jump action can be executed together to jump to the right. Additionally, these available actions are represented as *timed-actions*. This means that the five actions can be executed over a period of time measured by a length of frames. In other words, if an action (or two actions in parallel) is chosen at one frame, it (or they) will be continuously carried out for several subsequent frames. The reason of doing so is to cope with situations as Mario needs to jump higher when the jump action must be held for consecutive frames.

### 4.2 Experimental Setup

In the following TL experiments, we compare the transferability and complexity of ANNs evolved by different algorithms. These algorithms include original NEAT (NEAT), NEAT with Blended Searching (BS-NEAT), NEAT with Green's Phased Searching (GPS-NEAT), NEAT with Tan's Phased Searching (TPS-NEAT) and NEAT with Power-Law Ranking Probability based Phased Searching (PLPS-NEAT). They are used independently to learn on the source tasks, then the learned ANNs are transfered to original NEAT for learning on the target tasks. In addition, we also apply original NEAT to learning on the target tasks only for all experiments as a baseline for demonstrate the effectiveness of TL.

To evaluate transferability and complexity of ANNs evolved by each algorithm, we define two measurements respectively. Transferability is measured by the final learning performance (fitness value) of the best ANN obtained by an algorithm on a target task. Besides, we measure complexity by counting the total number of links and neurons of the best ANNs in STL phase. We focus on transferable and simplified ANNs, thus the complexity only needs to be gauged in STL phase.

For the purpose of reliable evaluation, all experiments have been conducted over 30 trials. For each trial, the fitness value for one algorithm is averaged over 10 unique game instances on learning both source and target tasks. Additionally, seeds for source and target tasks are different to ensure that the knowledge transferred cannot be level-specific nuances.

### 4.3 Experimental Design

In this research, we carefully design four TL experiments on MIB in order to assess the transferability as well as complexity of evolved ANNs of each algorithm.

**Experiment 1: Speed-Enemies** For this experiment, the source task is designed as that the player must travel as fast as possible to the right while navigating hilly terrain randomly generated for each task instance. Score is determined by the distance traveled by Mario (measured in game tiles) minus the time taken in seconds. The target task is designed similar to the source task but with the addition of Goomba enemies. If the player touches a Goomba (with the exception of jumping on its head) Mario dies and score is taken from that point. The objective of the target task is that the player must travel fast and avoid Goombas.

**Experiment 2: Speed-Coins** In the source task of this experiment, the player must travel as far to the right as fast as possible in a flat landscape while avoiding enemies (Goombas). Afterwards in the target task a new objective of collecting coins is added. Touching a coin will give the player a boost in score of the equivalent of traveling 5 game tiles. The positions of enemies and coins are randomly generated for each unique task. The optimal results can be achieved when the solutions capable of balancing three conflicting objectives: running to the end of the level as fast as possible, avoiding Goombas, and collecting coins.

**Experiment 3: Dist-Kills** In the source task, the player is scored by the distance traveled through hilly terrain occupied by Goombas. The target task differs by offering additional points for each Goomba killed by the player (by stomping on them). This is another example of a transfer learning task where an additional objective is introduced, making it more difficult to obtain an optimal score.

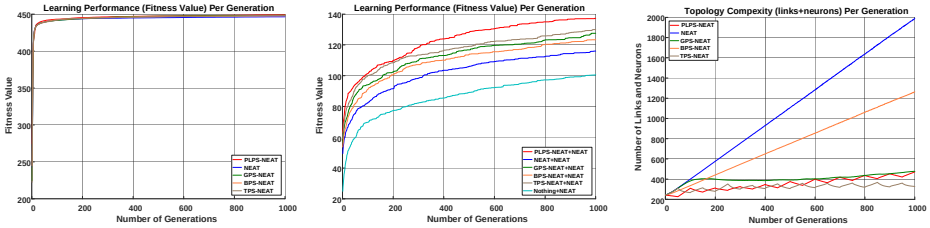
**Experiment 4: Goombas-Winged Goombas** In the experiment, the player is scored by their progression through a level with hilly terrain and Goombas (the same as the source task in 4.3) in the source task. Slightly different form the source task, regular Goombas are replaced with Winged Goombas in the target task. The challenge of the target task is that Winged Goombas can jump as high as Mario and may collide with the player in midair resulting in a death. Thus the objective is to achieve a higher score in a more difficult task.

## 5 Results and Discussion

In this section, the experimental results obtained from the four experiments are presented and analyzed separately. The section ends with a discussion on transferability and complexity of ANNs evolved by different testing algorithms.

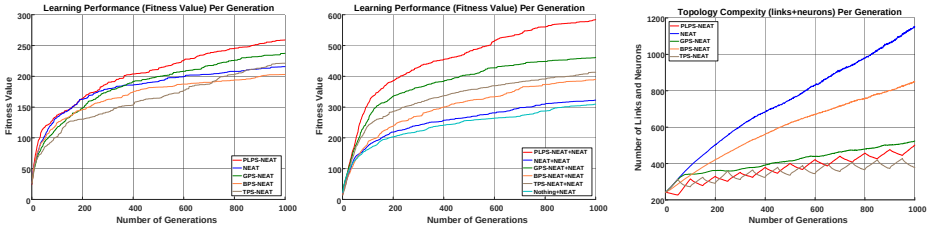
### 5.1 Results on Speed-Enemies Experiment

In the Speed-Enemies experiment, we find that NEAT with PLPS can evolve more effective (higher fitness value shown in Fig. 3(a) and (b)) and less complex ANNs (smaller number of links and neurons illustrated in Fig. 3(c)) than other candidate algorithms. In STL phase, learning performance of NEAT with PS methods are competitive to that of the original NEAT shown in Fig. 3(a). In TTL phase, observably competitive performance of PLPS-NEAT+NEAT against other algorithms can be evidenced in Fig. 3(b). Moreover, PLPS-NEAT+NEAT yields p-values of $1.2214 \times 10^{-3}$, 0.0041 and 0.0342 while the performance of it is compared to that of Nothing+NEAT, NEAT+NEAT and BPS-NEAT+NEAT respectively. Nevertheless, the statistical significance cannot be supported between PLPS-NEAT+NEAT and TPS-NEAT+NEAT(or GPS-NEAT+NEAT) with p-value 0.4447 (or 0.1660). Lastly, Fig. 3(c) clearly illustrates that the complexity of evolved ANNs by NEAT with PS methods are managed to lower levels compared to that of original NEAT.




(a) STL (b) TTL (c) Complexity in STL

Fig. 3: The comparison of transferability and complexity of ANNs (averaged over 30 trails) obtained by different algorithms on the Speed-Enemies experiment: (a) displays the averaged learning performances on STL ($y$-axis in scale [200,450]), (b) displays the averaged learning performances on TTL ($y$-axis in scale [20,140]), and (c) displays the averaged total numbers of links and neurons for the best ANNs obtained from STL.

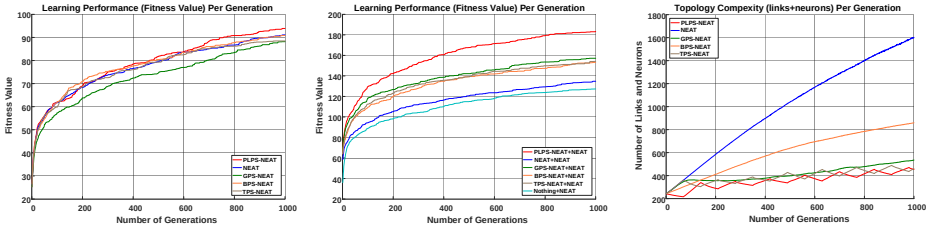



(a) STL (b) TTL (c) Complexity in STL

Fig. 4: The comparison of transferability and complexity of ANNs (averaged over 30 trails) obtained by different algorithms on the Speed-Coins experiment: (a) displays the averaged learning performances on STL ($y$-axis in scale [0,300]), (b) displays the averaged learning performances on TTL ($y$-axis in scale [0,600]), and (c) displays the averaged total numbers of links and neurons for the best ANNs obtained from STL.

### 5.2 Results on Speed-Coins Experiment

While being performed in Speed-Coins experiment, PLPS-NEAT exhibits higher learning performance meanwhile results in structurally simplified ANNs. As seen in Fig. 4(a) and 4(b), in terms of learning performance, PLPS-NEAT not only observably outperforms all other algorithms either in STL or TTL phase, but also is confirmed statistically better. For example, in TTL phase, the p-values are $9.1103 \times 10^{-6}$ (Nothing+NEAT), $1.8500 \times 10^{-5}$ (NEAT+NEAT), 0.0102 (GPS-NEAT+NEAT), 0.0015 (BPS-NEAT+NEAT), and 0.0076 (TPS-NEAT+NEAT). In addition, the complexity of final ANNs obtained by NEAT with PS methods is effectively controlled as shown in Fig. 4(c).




(a) STL (b) TTL (c) TTL

Fig. 5: The comparison of transferability and complexity of ANNs (averaged over 30 trails) obtained by different algorithms on the Dist-Kills experiment: (a) displays the averaged learning performances on STL ($y$-axis in scale [20,100]), (b) displays the averaged learning performances on TTL ($y$-axis in scale [20,200]), and (c) displays the averaged total numbers of links and neurons for the best ANNs obtained from STL.

### 5.3 Results on Dist-Kills Experiment

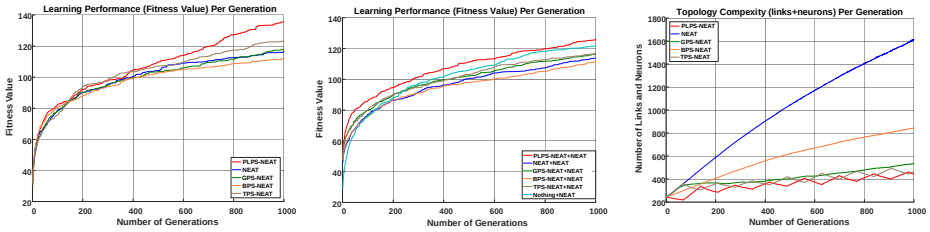
Results on the Dist-Kills experiment also evidently show that PLPS-NEAT is capable of producing simpler ANNs to learn both source and target tasks effectively in the TL context. In Fig. 5 (a) and 5 (b), PLPS-NEAT is observed to be more effective among all candidate algorithms. To verify this observation in TTL phase, we perform pair-wised T-tests in-between PLPS-NEAT against all other algorithms. Statistical results reject the null hypothesis with p-values $1.3132 \times 10^{-7}$ (Nothing+NEAT), $8.1118 \times 10^{-6}$ (NEAT+NEAT), 0.0094 (GPS-NEAT+NEAT), 0.0426 (BPS-NEAT+NEAT) and 0.0322 (TPS-NEAT). Fig. 5(c) also shows that PLPS is able to well control the complexity of its evolved ANNs.

### 5.4 Results on Goombas-Winged Goombas Experiment

Similar to the previous three experiments, results of the Goombas-Winged Goombas experiment give evidences of that PLPS-NEAT effectively learns to search ANNs with simple topology in this TL scenario. Clearly, PLPS-NEAT exhibits a leading performance among all testing algorithms in the STL stage as shown in Fig. 6(a). Moreover, Fig. 6(b) also shows that PLPS-NEAT+NEAT conducts a more effective TTL than other algorithms. Statistical tests provide a confirmation of the significance where the p-values are 0.0381, 0.0181, 0.0475, 0.0032 and 0.0297 corresponding to Nothing+NEAT, NEAT+NEAT, GPS-NEAT+NEAT, BPS-NEAT+NEAT and TPS-NEAT+NEAT respectively. Meanwhile, Fig. 6(c) also indicates that PLPS-NEAT as well as other PS methods based NEAT algorithm manage to suppress the topology bloat of evolved ANNs.

### 5.5 Discussion on Transferability and Complexity

In this research, we have achieved the research goal by successfully developing PLPS-NEAT-TL algorithm which can evolve transferable and simple ANNs on source tasks and later apply to target tasks improve TTL effectiveness.




(a) STL (b) TTL (c) TTL

Fig. 6: The comparison of transferability and complexity of ANNs (averaged over 30 trails) obtained by different algorithms on the Goombas-Winged Goombas experiment: (a) displays the averaged learning performances on STL ($y$-axis in scale [20,140]), (b) displays the averaged learning performances on TTL ($y$-axis in scale [20,140]), and (c) displays the averaged total numbers of links and neurons for the best ANNs obtained from STL.

First of all, PLPS-NEAT can evolve transferable ANNs, which can be verified from two aspects. First, experiments results on source tasks clearly reflect that PLPS-NEAT is effective in terms of the learning performance. Second, while perform PLPS-NEAT+NEAT on target tasks, we can also find that it is competitively effective in comparison to all other testing algorithms.

Secondly, the PLPS-NEAT-TL is capable of controlling the complexity in a reasonably low level. However, for experiment Speed-Enemies and Speed coins, a lower complexity but lower effectiveness has been obtained by TPS-NEAT algorithm compared to PLPS-NEAT-TL algorithm. This can be explained as that, during the simplification process, the pure randomness of TPS-NEAT may delete some important genes resulting in the performance worse than that of PLPS-NEAT-TL. Thus, it is useful to control the randomness during simplification phase for NEAT.

## 6 Conclusions

This paper explored the transferability of ANNs evolved by NEAT for intelligent gameplay, and further investigated the usefulness of PS. Motivated by the belief that simpler and effective ANNs are more transferable, we focused on NEAT with PS for TL. In line with this, we proposed a PLPS method with power-law ranking probability to effectively control the selection randomness during simplification phase. Experimental evidences showed that the PS methods are capable of improving the transferability of ANNs obtained from NEAT. Specifically, our proposed PLPS method outperformed NEAT and other PS based methods across all our TL experiments. Besides, we also found that the complexity of evolved ANNs can be effectively controlled. The size of ANNs consistently stayed at a low level while the performance was continuously improved. This work provides possibilities of applying PS and similar methods to other TL tasks. The advantages of using Power-Law distribution to control the PS methods should also be investigated in more depth.

## References


1. Cardamone, L., Loiacono, D., Lanzi, P.L.: Learning to drive in the open racing car simulator using online neuroevolution. TCIAG 2(3), 176–190 (2010)
2. Gauci, J., Stanley, K.O.: Autonomous Evolution of Topographic Regularities in Artificial Neural Networks. Neural Computation (2010)
3. Green, C.D.: Phased searching with neat: Alternating between complexification and simplification (2004), `http://sharpneat.sourceforge.net/phasedsearch.html`
4. Hausknecht, M., Khandelwal, P., Miikkulainen, R., Stone, P.: Hyperneat-ggp: A hyperneat-based atari general game player. In: GECCO 2012. pp. 217–224. ACM (2012)
5. Hausknecht, M., Lehman, J., Miikkulainen, R., Stone, P.: A neuroevolution approach to general atari game playing. TCIAG 6(4), 355–366 (2014)
6. James, D., Tucker, P.: A comparative analysis of simplification and complexification in the evolution of neural network topologies. In: GECCO 2004 (2004)
7. Karakovskiy, S., Togelius, J.: The mario ai benchmark and competitions. TCIAG 4(1), 55–67 (2012)
8. Kotsiantis, S., Kanellopoulos, D.: Discretization Techniques : A recent survey. TCSE 32(1), 47–58 (2006)
9. Kuhlmann, G., Stone, P.: Graph-Based Domain Mapping for Transfer Learning in General Games. ECML 4701(Chapter 20), 188–200 (2007)
10. Mandziuk, J.: Knowledge-Free and Learning-Based Methods in Intelligent Game Playing. Springer (2010)
11. Moriarty, D.E., Miikkulainen, R.: Discovering Complex Othello Strategies through Evolutionary Neural Networks. Connect. Sci. 7(3), 195–210 (1995)
12. Schrum, J., Miikkulainen, R.: Solving multiple isolated, interleaved, and blended tasks through modular neuroevolution. ECJ (2016)
13. Sharma, M., Holmes, M.P., Santamaría, J.C., Irani, A., Isbell Jr, C.L., Ram, A.: Transfer learning in real-time strategy games using hybrid cbr/rl. In: IJCAI. vol. 7, pp. 1041–1046 (2007)
14. Stanley, K.O.: Efficient reinforcement learning through evolving neural network topologies. In: GECCO 2002. Citeseer (2002)
15. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary Computation 10(2), 99–127 (2002)
16. Tan, M., Pu, J., Zheng, B.: Optimization of network topology in computer-aided detection schemes using phased searching with neat in a time-scaled framework. Cancer informatics 13(Suppl 1), 17 (2014)
17. Taylor, M.E., Stone, P.: Behavior transfer for value-function-based reinforcement learning. AAMAS p. 53 (2005)
18. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. JMLR 10(Jul), 1633–1685 (2009)
19. Taylor, M.E., Whiteson, S., Stone, P.: Temporal difference and policy search methods for reinforcement learning: An empirical comparison. In: NAAI. vol. 22, p. 1675. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007)
20. Taylor, M.E., Whiteson, S., Stone, P.: Transfer via inter-task mappings in policy search reinforcement learning. In: AAMAS. p. 37. ACM (2007)