Transfer of Evolved Pattern-Based Heuristics in Games

Erkin Bahçeci and Risto Miikkulainen
{erkin,risto}@cs.utexas.edu
Dept. of Computer Sciences
The University of Texas at Austin

Abstract— **Learning is key to achieving human-level intelligence. Transferring knowledge that is learned on one task to another one speeds up learning in the target task by exploiting the relevant prior knowledge. As a test case, this study introduces a method to transfer local pattern-based heuristics from a simple board game to a more complex one. The patterns are generated by compositional pattern producing networks (CPPNs), which are evolved with the NEAT neuroevolution method. Results show that transfer improves both final performance and the total learning time, compared to evolving patterns for the target game from scratch. Pattern-based transfer is therefore a promising approach to scaling up game players toward human-level.**

Keywords: Transfer, NEAT, CPPN, Patterns, Game Playing

I. Introduction

Computer game players to date have been mostly developed for a particular game and have incorporated as much human expertise for that game as possible, such as opening and end-game databases. Deep Blue is a prime example of a successful game player that is engineered for a single game [1]. While this approach works with few games, it does not scale to others such as Go, which remains prohibitively difficult for computers compared to human players.

Rather than building a player for a single game, a more useful and more human-like approach would be to develop an agent that is able to learn to perform more than one task. Such generality is the aim of, e.g., the general game playing (GGP) competition [2]. In GGP the emphasis is learning to play previously unseen games without any available human expertise for that game. The ability to transfer past knowledge would be substantially beneficial in learning new tasks in such a setting, since games usually share many similar properties, such as piece movement patterns, reward structures, and neighborhood relations between pieces.

Such an ability to transfer is one of the reasons why we are more capable than computers in learning to play games like Go. Another reason is our capability to reason in terms of spatial and temporal patterns. Many board games involve spatial patterns, but Go has an especially pattern-oriented gameplay. Identifiable patterns arise in many situations in Go, which helps us assess the relative desirability of the current board state and the life-and-death status of a particular group of stones, determine whether we should sacrifice a group, and where we should play next on the board. Being able to learn geometric regularities and relevant patterns would help develop more general and capable agents for spatial tasks.

While many studies have been conducted on transfer learning [3], [4], [5] and on learning patterns for tasks [6], [7], to the authors' knowledge, these two approaches have not been utilized together in the domain of games. This paper aims to combine them, using the populations in evolution as the medium for transfer, where individuals in the population encode patterns to be used as game heuristics. The hypothesis is that, given a related source task, transfer will improve performance and total learning time in a target task, compared to an approach that learns the target task from scratch. This hypothesis is tested on a particular game by encoding game heuristics as compositional pattern producing networks (CPPNs) [8], which produce 2D patterns, and evolving them with the NEAT neuroevolution method [9].

An experiment was carried out in a simple board game to compare the performance and total time required with and without transfer. The results show that transfer improves both performance and total learning time in this game. These results can be seen as the first step towards improving the learning in challenging pattern-based games and other complex tasks that involve patterns.

This paper is organized as follows. Section II reviews related work on transfer and the use of evolution and patterns in learning to play games. A description of the task domain, the patterns in it, the evolutionary method that was employed, and how patterns were evolved are given in Section III. The experiment setup and results are described in Section IV. Section V discusses advantages and shortcomings of the method, and Section VI suggests possible directions for future research, such as extensions to more complex games and to life-long learning.

II. Related Work

Transfer learning is a widely researched topic. It can be defined as using the knowledge learned on a source task to improve learning on a target task. In one successful transfer study Taylor et al. [3] used inter-task mappings for effective transfer with policy search methods, such as NEAT, in keepaway soccer and server job scheduling domains. The study showed that even incomplete mappings are beneficial if complete, hand-coded mappings are not available. Furthermore they introduced a method for learning these mappings from experience, reducing the necessary expertise to perform transfer. However, this approach is effective only if high-level state variables, like "the distance to each player", and high-level actions, such as "pass the ball to player x", are available, which might not be in every task.

Transfer learning was also utilized by some of the players in the GGP competition. For instance, Kuhlmann et al.'s general game player [4] successfully transferred learned knowledge between games that were recognized to have similarities by constructing *rule graphs* from game rules and using graph isomorphism. To determine similarity, their method checked whether a newly encountered target game was isomorphic to a previously played game or a variant of one, where a game variant corresponds to a transformation of the rule graph of that game. Several identified variant classes were considered. The target game had to be tested for graph isomorphism against all variants of all previously played games, which could cause problems in scaling to a larger set of games and variants. If the target game was found to resemble a previous game that was learned earlier, the value function learned for the previous game was transferred to the target game through an appropriate mapping depending on which variant of the earlier game the target game was isomorphic to. This speeded up and improved learning in the target game.

Although approaches like Kuhlmann et al.'s were able to leverage the learning done earlier to some degree through transfer, the GGP competition in general does not allow learning and evolutionary methods to be used effectively, by considerably limiting the time provided for each game during the competition.

While transfer in board games is the focus of this paper as well, the type of transfer is more akin to the way knowledge is carried between different versions of a task in *incremental evolution* (IE) [10]. In IE, transfer is performed implicitly by taking the final population of evolution on a task and starting evolution on a slightly more difficult task with that population instead of a random population. Incremental evolution was utilized by several researchers including Gomez et al. [11] in a prey capture task with ESP/delta-coding, resulting in improved performance and generalization. On the other hand, Christensen et al. [12] were not able to improve performance in a cooperative phototaxis task neither by decoupling behaviors nor by increasing the complexity of the environment incrementally, suggesting that transfer is by no means guaranteed. Similar to these studies, the final population of the evolution on one task is used as the initial population of a more difficult task in this paper, but in a board game domain.

Evolutionary methods have been employed to build board game players in numerous studies. A successful one by Chellapilla and Fogel [13] evolved a checkers player without incorporating any expertise of checkers. The player was an artificial neural network with three hidden layers. The network was manually designed except for the weights, which were evolved through coevolution. Their network had inputs that consisted of an encoding of the checkers board as a set of overlapping, square regions of varying size, and had a single output that represented the value of the input board state. The evolved checkers player, called Blondie24, was able to compete with human experts on an Internet gaming website.

While Chellapilla and Fogel [13] did not optimize the input encoding or the neural network structure they used, Gauci et al. [14] did so in a more recent study, which also evolved a checkers player. They compared three approaches: (1) regular NEAT with 32 inputs, (2) regular NEAT with the same engineered inputs as Blondie24, and (3) HyperNEAT [7] with a network substrate that had inputs arranged in an 8×8 grid and a hidden layer of the same size as the input grid. They found that the networks that were able to learn from geometry learned and generalized better. The HyperNEAT method, which learned geometric regularities itself, generalized the best of the three approaches. Transferring the evolved connectivity patterns between different games or different versions of the same game could be an interesting extension to this HyperNEAT approach, however transfer learning was not the focus of their study.

A different approach to incorporating patterns in games was followed by Epstein et al. [6]. Their method started with a generic and novice player for board games, and shifted its focus to game-specific, spatial heuristics that were learned during gameplay. This study expanded on their earlier work on a multi-tiered game player that learned how to use a set of *advisors*, i.e., domain-specific rationales, by playing against an external expert [15]. Though their study on pattern-based heuristics made good use of patterns in gameplay, they did not transfer them from one game to another, which could be beneficial for certain games bearing enough similarity among the ones they experimented with.

In this paper, the approach of using patterns for gameplay will be combined with the evolution of task solutions. The approach is similar to the work of Gauci et al. [14], but also incorporates transfer to improve performance and total time in a related target task.

III. Method

The method employed in this study can be summarized as the transfer of pattern-based game heuristics that are encoded as CPPNs and evolved through NEAT. This section describes the chosen task, how 2D patterns are used to play the game, how NEAT works, and how patterns are encoded and evolved with NEAT.

A. The Task Domain

The task chosen for investigating transfer is a single-player board game similar to the one used in a reusable neural module evolution study by Reisinger et al. [16]. The initial state of the game is a single black piece placed randomly on a square board. The goal of the game is to place a fixed number of white pieces so that as many of them as possible *check* the black piece, i.e., both are on the same row or on the same column. The motivation for this game is to avoid

TABLE I
PROPERTIES OF THE SOURCE AND TARGET GAMES

	Source game	Target game
Board size	4×4 board	12×12 board
Check	Same row or column	Same row or column with distance ≤ 3
# of moves in a game	6	12
# of game evaluations	16	144

the complexities of two or multi-player games, while still being able to demonstrate the effects of transfer.

Different versions of this game were used as the source and target for transfer as summarized in Table I. The source game has a 4×4 board, with six moves allowed for the player. The target game has a larger board of size 12×12, allows 12 moves, and additionally has a distance limit of three squares for a check. A distance constraint of three squares is applicable in the target game, but not in the source game because of the 4×4 board size, which effectively makes the source game easier to learn with pattern-based heuristics. Figure 1 shows sample board state instances for source and target games; the distance constraint is shown as a shaded region around the black piece.

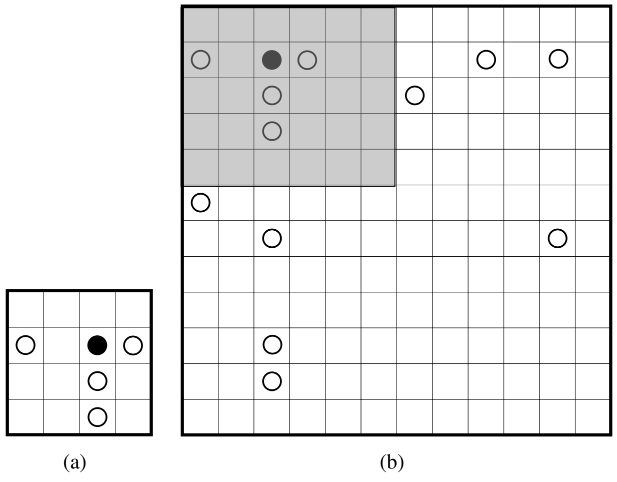

Fig. 1. Example game board states for source (a) and target (b) games. In this single-play game, the objective is to place a white piece on the board to check the black piece, i.e., locate in the same row or column. The shaded region shows the distance constraint for a check in the target game.

When a game ends after a fixed number of moves, the score is calculated as the ratio of the achieved number of checks to the possible number of checks, which is always six for the source game and less than or equal to 12 for the target game (depending on how close to the edge the black piece is). The next section describes the player for this game and how patterns are utilized for gameplay.

B. Using Patterns for Gameplay

At each step during the game, the game player picks a move, i.e., where to place a white piece, by evaluating the possible moves at that step. To evaluate a move, the player uses a pattern consisting of a discrete, 2D array of real values between 0 and 1. This 2D pattern is used as a local board heuristic, i.e., for each empty square on the board, the player aligns the pattern to be centered on that square and computes a heuristic value for that square by taking the pattern value at the point where it coincides with the single black piece. The size of the 2D pattern is determined so that it is large enough to cover every square wherever the pattern is placed on the board. For instance, for a 4×4 board, the pattern has a size of 7×7. After the heuristic value is computed for each empty square on the board, the player picks the square with the highest value, and makes a move by putting a white piece on that square.

For this particular game, patterns that reflect the concept of check as close as possible are likely to perform better than ones that fail to do so, because such patterns will lead to white pieces being placed where they would check the black piece. Hence a plus-shaped pattern will perform well on the source task, whereas on the target task a plus shaped pattern with smaller values near the edges is better. Thus, the hypothesis is that a pattern that performs well on the source game is a good starting point to learn a pattern for the target game and may result in better final performance.

In this study the task of learning 2D patterns for playing the game is accomplished by encoding them as networks called CPPNs, and evolving these networks via NEAT, which is detailed in the next section.

C. Neuro-Evolution of Augmenting Topologies

Creating a neural network to perform a particular task requires determining the network topology and weights of connections between the nodes of the network. Although many simple tasks can be solved with fixed (e.g. fully connected) network topologies, more complex tasks may benefit from evolving the topology as well. To this end, Stanley [9], introduced the Neuro-Evolution of Augmenting Topologies (NEAT) method, which evolves both connection weights and topology for neural networks, given a fitness function that is tailored to the particular problem that one is trying to solve. This fitness function is used to evaluate each population member, which is a neural network, to determine how well it solves the problem.

NEAT has three components that differentiates it from other neuroevolution methods. First, the initial population of networks consists of minimally connected individuals with no hidden nodes. The networks gradually become more complex through mutations that add nodes and connections. Since only the additions to the topology that improve performance are kept, this process results in finding neural networks, i.e., solutions to the problem, that are as small as possible. Starting with minimal topology also speeds up learning since the number of connection weights to be optimized during evolution, i.e., the size of the search space for connection weights, is minimal.

Second, crossover between individuals with different topologies is made possible by keeping an *innovation number* for each gene. Innovation numbers are used to match genes that have similar historical origin. They are an abstraction of homology in biological evolution, which is the mechanism for aligning similar genes during crossover. Keeping these numbers for each gene circumvents the expensive task of matching topologies of networks for crossover.

The third component of NEAT is that innovation in population members is protected by separating the population into species depending on similarity. When a structural mutation considerably alters an individual, the individual may not initially perform as well as other population members. If this happens, that individual will not survive even though this mutation might have led to a better performing individual after some optimization. Speciation protects such individuals by putting networks that are too different from others into separate species, allowing them to be optimized within the species first. To prevent the whole population from being reduced to a single species, *explicit fitness sharing* [17] is used. This principle means the fitness within the species is shared among its members, dividing the fitness of each individual by the size of its species, preventing species from becoming too large.

NEAT has been shown to be successful in several domains, such as double pole balancing [9], vehicle control and collision warning [18], and video games [19]. Recently, it has been used to evolve patterns through CPPNs, i.e., networks that produce patterns [8]. The next section describes how NEAT is used to evolve patterns for transfer.

D. Evolving 2D Patterns

The 2D patterns that are used to play the game are encoded as CPPNs. A CPPN is a functional description of a pattern in the form of a network and hence a compact way of storing patterns. CPPNs can be evolved through NEAT, since they are structurally akin to neural networks [8]. To produce a 2D pattern, a CPPN with one output, two actual inputs, and a bias input can be used, where the two inputs are the x and y coordinates of a point on a plane and where the output is interpreted as the value of the pattern at that point. The x and y coordinates are taken as relative to the center of the pattern. An example CPPN and the 2D pattern it produces can be seen in Figure 2. The range for the pattern coordinates is determined such that the pattern covers the whole board wherever it is placed on the board. Thus r is taken to be $s-1$ for a board of side length s, since such a board can always be covered with a pattern with coordinates in the range $[-(s-1),(s-1)]$.

Each individual in the NEAT population encodes a single CPPN. In this study, each neuron in a CPPN has either a Gaussian or a sigmoid activation function. The number of possible activation functions are limited to these two functions because they are sufficient to obtain patterns that are useful for the particular game in this study. For more complex games it may be beneficial to employ a larger set of activation functions, as is done in HyperNEAT [7].

In this paper, a mutation is added to NEAT to switch activation functions of a node between Gaussian and sigmoid without changing its innovation number. Since the probability

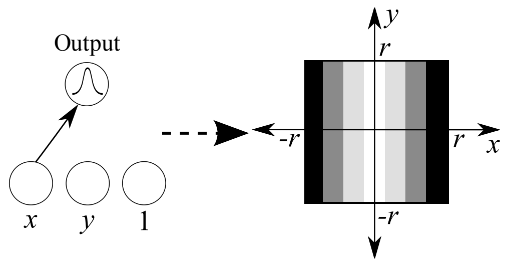

Fig. 2. A CPPN with x, y, and bias inputs. A 2D pattern is generated from a CPPN by running the CPPN for each (x,y) input pair within a given range, $[-r,r]\times[-r,r]$, of discrete values and using the output of the CPPN as the pattern's value at position (x,y), where $r=s-1$ for a board with side length s. Since only the x input is connected to the output node in this CPPN, the 2D pattern is a function of x only. The Gaussian activation function of the output node causes a band-like pattern to be generated.

of this mutation is low, crossover is not modified, where a mutated and a non-mutated version of the gene are crossed over, one of the activation functions is chosen randomly. This mutation allows improving individuals that are close to the solution but have the wrong activation function in one of their nodes.

The individuals in the initial random population of NEAT are minimally connected networks, i.e. networks with only one connection, such as the one in Figure 2. Therefore, initial random population consist of patterns that are Gaussian or sigmoid functions of either x or y, i.e. vertical or horizontal bands and gradients, centered at the axis, as can be seen in Figure 3.

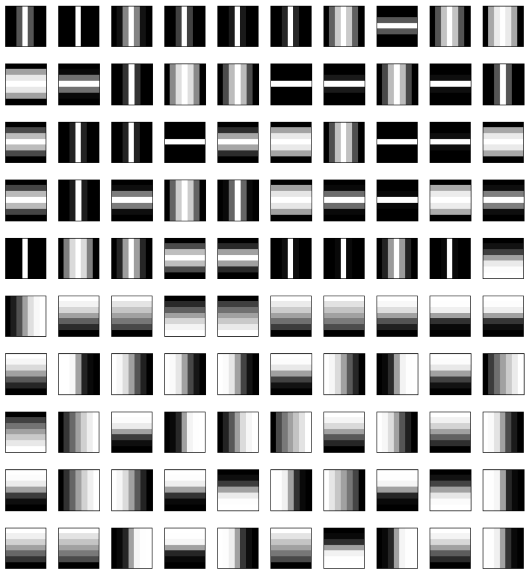

Fig. 3. One hundred sample patterns from the random initial population for the source game. Due to the way NEAT initializes population members, the random initial population consists of patterns produced by Gaussian and sigmoid functions of either x or y inputs, centered at the axis.

The algorithm for evaluating the fitness of a single pattern is given in Figure 4. Each pattern is evaluated over multiple games to reduce bias introduced by similar initial random configurations. Specifically, 20 and 144 games are used for

```
1:  Generate 2D pattern from CPPN
2:  **for** fixed number of games **do**
3:      Initialize board randomly
4:      **for** fixed number of steps **do**
5:          **for each** empty square $s$, i.e., each possible move **do**
6:              Center pattern on $s$
7:              Get pattern value at the square with black piece
8:              **if** that value $>$ value of the best move **then**
9:                  Update the best move as $s$
10:             **end if**
11:         **end for**
12:         Play the best move found
13:     **end for**
14:     $score_i \leftarrow \frac{\text{\# of checks}}{\text{\# of possible checks}}$
15: **end for**
16: $fitness \leftarrow \frac{\sum_i score_i}{\text{number of games}}$
```

Fig. 4. The algorithm for evaluating a CPPN. A fixed number of games with a fixed number of steps are played to evaluate each CPPN. Since a move in this game corresponds to placing a new white piece in an empty square, at each game step every possible move is considered by centering the pattern on each empty square. The best move is picked so that the pattern value at the square with the black piece is maximized over all possible moves for that step. The final score for a CPPN is the ratio of the number of achieved checks to the number of possible checks averaged over all games.

the source and target games, respectively. The average score on these games is taken to be the fitness of the CPPN that generated that particular pattern.

Although both methods employ patterns encoded as CPPNs, there are two main differences between this method and the HyperNEAT approach by Gauci et al. [14]. First, their method discovers geometric regularities automatically, whereas the regularity is built-in to this method through the repetitive procedure of centering and applying a pattern at each empty square. The second difference is that their approach utilizes a hidden layer of 2D grid of nodes of the same size as the input node grid, whereas this method corresponds to using a HyperNEAT substrate without that hidden layer of nodes. These differences reduce the CPPN search space compared to Gauci et al.'s method by using two fewer inputs and one fewer output: the inputs are the x and y position of a square relative to the square on which the pattern is centered, while the HyperNEAT approach has four inputs, i.e., two sets of x and y inputs to compute the connection weights between input and hidden layers, and an additional output to compute the weights between the hidden layer and the output node. Such a simplification is possible because the game is simple and focuses on local patterns.

Using the fitness evaluation method described above, the effect of transfer was evaluated with the experiment detailed in the next section.

IV. Experiment

An experiment was conducted to evaluate the effects of transfer with evolved pattern-based game heuristics. Patterns were evolved to play the source game, and were then transferred to the target game as the starting population for evolution. This approach was compared to evolving patterns for the target game from scratch. Details of the experiment, performance and time results, and evolved patterns are presented next.

A. Experiment Design

To test the hypothesis that the set of patterns learned for the source game can be a good starting point for learning for the target game, CPPNs were first evolved with NEAT on the source game with a 200-CPPN population. The fitness of the pattern generated by each CPPN was evaluated with the procedure described in the previous section. The population for this evolution was initialized with a minimal number of connections and with random connection weights. The evolution terminated at the 45^{th} generation when NEAT produced a population that contained one or more perfect players, i.e., patterns that can score the maximum number of checks in the source game. This population, which is illustrated in Figure 5, was then transferred to the target game, and was used as the initial population for NEAT, instead of a random initial population. This population transfer is similar to how evolved populations are used as initial populations for modified or new tasks in incremental evolution. When transferring CPPNs to the target game, no modification was done on CPPNs other than adjusting r in the range $[-r, r]$ of pattern coordinates for the board size in the target game.

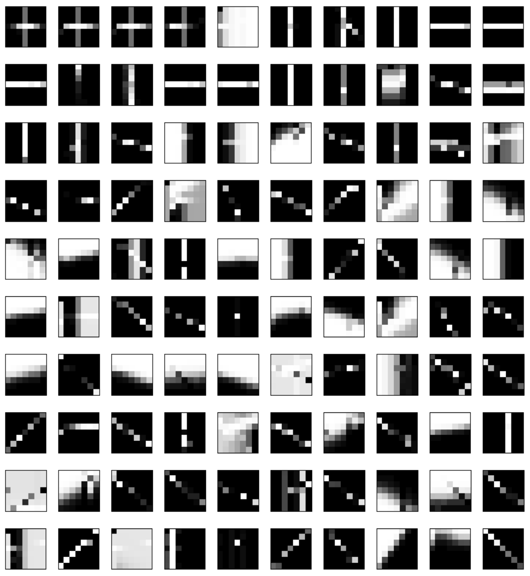

Fig. 5. One hundred sample patterns from the population of the 45^{th} generation for the source game, sorted by fitness. This population contains a perfect solution (the top left pattern), which is a plus-shaped pattern (i.e. all squares of the plus are above threshold). Patterns that have either a vertical or a horizontal narrow band, like a few ones in the top three rows, do not perform as well as plus-shaped patterns, but are still better than many others in the final population. This particular population was transferred to the target game to be used as the initial population for the transfer approach.

The NEAT population consisted of 200 CPPNs, and evolutions were terminated after 1000 generations or when a pattern was found that played all games perfectly, i.e., scored the maximum number of checks. The 1000 generation limit was chosen because there was no significant improvement after that point. A total of 20 evolutions were run for the target game, each initialized with the same population evolved for the source game.

To assess how effective transfer was, another 20 evolutions were run as the control experiment. In these evolutions no transfer was used, but patterns were instead evolved for the target game from scratch. Initial populations were created randomly as was done for the source game. The same NEAT parameters were used for this second set of evolutions. To evaluate the effects of transfer, game playing performance and the total learning time were compared in the two cases.

B. Performance Results

The performance of the two methods are shown in Figure 6. This plot shows the success percentage of the best individual in the population through NEAT generations averaged over 20 evolutions, with 95% confidence intervals. The differences are statistically significant with $p < 10^{-11}$ in Student's t-test. The plot shows that transfer improved the performance by an average of 15% across all generations. At the first generation of evolution, the transferred population already performs better than the random initial population, which means that a pattern learned for the source game performs well, to some degree, on the target task. Furthermore, the performance increase in the first few generations was much faster with transfer. However, the most interesting result is that transfer also led to an eventual performance gain of 15% after 1000 generations. This discrepancy still existed even after the evolution without transfer was allowed to run for 2000 generations. This result confirms the hypothesis in section III-B: the transferred population serves as a better starting point for the target game than a population of random CPPNs, while also leading to performance levels that were not achievable without transfer.

C. Total Learning Time

The performance improvement implies that with transfer it takes less time to reach a given performance level. However, it is more interesting to investigate whether the *total* learning time is reduced as well, i.e., the time needed to learn the target game plus the time spent for the source game. Such a reduction would mean that transfer is a faster way to learn to play the target game, compared to starting learning from scratch.

Figure 7 compares the probability of finding a player that plays the target game perfectly using the same total number of NEAT generations, for the approaches with and without transfer. For the approach that starts evolution from scratch, generations were only used for the target game. On the other hand, the transfer approach also uses generations for the source game before evolving players for the target game. The number of source game generations were 45, as mentioned earlier. The plotted probability at each generation is estimated based on the whether or not a perfect player was found at each generation in the 20 evolution runs done for the two

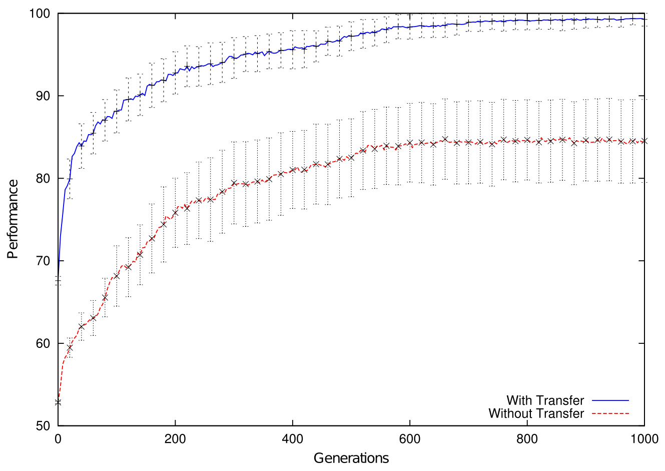

Fig. 6. Target game performance of evolution with transfer and evolution starting from scratch over 1000 generations. Generations spent on the source game for the transfer approach are not shown. The plot shows the game success percentage of the best NEAT population member at each generation. The results are averaged over 20 evolutions. 95% confidence intervals are also indicated. Transfer improves performance by about 15% across all generations, while also speeding up learning in the first few generations. After 1000 generations the eventual performance with transfer is significantly closer to the best possible performance.

approaches. When any horizontal cross-section is considered in the figure, it is clear that the transfer approach requires fewer generations, or equivalently less total time, to achieve the same probability of obtaining a perfect player for the target game. The plots also shows that the probability of finding a perfect player after 1000 generations is significantly improved with transfer.

D. Evolved Patterns

Figure 8 shows hundred sample patterns generated from target game populations at the end of an evolution with transfer. Through mutations and crossovers over the course of many generations, NEAT produces a diverse population of patterns. The final evolved population encompasses multiple species of patterns, which correspond to groups of patterns resembling each other in the figure.

The top left pattern in the figure is a perfect solution for the target game, i.e., a pattern that achieves the maximum possible number of checks in the game. That pattern and a few of the following ones contain a plus shape with values diminishing toward the edges and thus perform better than others in the population. These patterns are descendants of the simpler plus-shaped patterns in the transferred population in Figure 5, which is to be expected since NEAT gradually makes networks in the population more complex over generations by keeping the structural additions that improve performance. The rest of the patterns in Figure 8 that have two narrow bands with one being mostly vertical (like most of the ones on the second and third rows), do not perform as well as the plus-shaped ones, but still fare better than the ones with a single slanted band. Maintaining diversity in the population through speciation, as observed here, protects innovation in the population [9]. Moreover, species that do

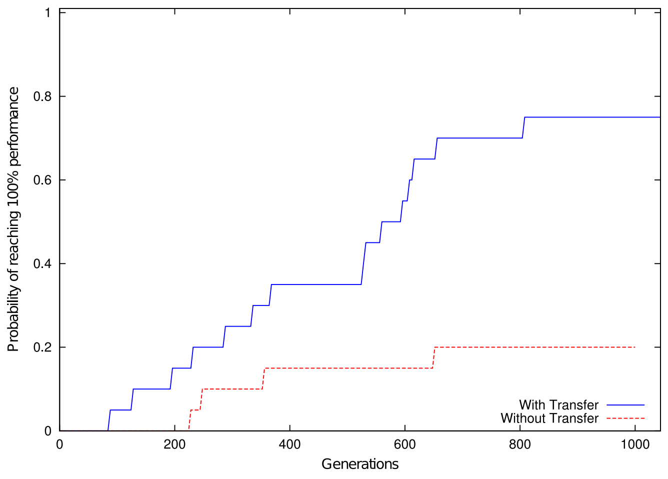

Fig. 7. Probability of reaching 100% performance on the target task versus the total number of NEAT generations used, comparing the transfer approach to starting from scratch. The plotted probability is an estimate based on the 20 evolution runs for the two approaches. For the evolution without transfer, only target game generations are shown, whereas for the evolution with transfer, the total number of generations is shown, which includes generations for the source game (45 generations) and generations for the target game. Even though the transfer approach has to spend generations on the source game before the target game, it requires much fewer generations to obtain the same level of performance. The transfer approach also results in higher odds of obtaining a perfect player at any total number of generations used.

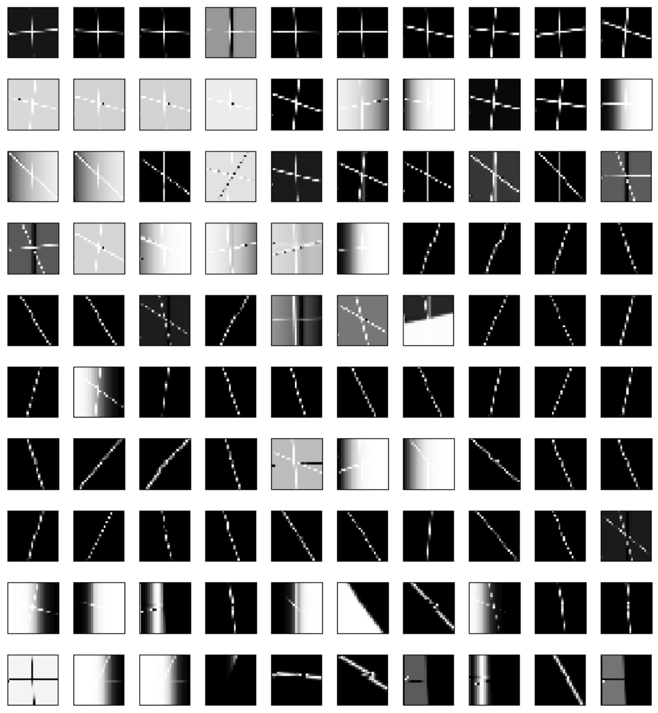

Fig. 8. One hundred sample patterns from a final evolved population for the target game, sorted by fitness. The first few patterns on the top row performed the best in this population because they contain a plus shape with values getting smaller toward the edges. Multiple species can be observed as groups of similar patterns, such as the patterns with a plus shape, the ones with a single narrow band, and the ones with a gradient.

not perform the best in a source task but are still kept in the population by NEAT may prove useful when that population is transferred to a new task, where the differences in that species could be more beneficial to performance. Investigating such effects of population diversity on transfer is part of future work.

V. Discussion

The results show that first evolving patterns to play a simplified version of the target game and transferring the obtained population to the target game as the initial population improves both the total time required to reach the same performance level and the performance eventually achieved after a fixed number of generations. Utilizing transfer is therefore a better learning method for the target game than evolving patterns starting from scratch.

There are certain advantages to spending time on NEAT generations on a simplified version of a board game instead of spending the same number of generations on the more complex version of the game. If the simplified game has a smaller board, fitness evaluations for each CPPN take much less time. Furthermore, a pattern that can play the source game perfectly is learned in fewer generations than a similar pattern for the target game, because a simpler CPPN is needed to generate it.

Another important point in this study is the use of local heuristics to break the board evaluation heuristic down to transferable chunks, as opposed to learning a single neural network to evaluate the whole board. This approach makes it easier to transfer the learned heuristics between games with different board sizes. In games like Go, transfer between varying board sizes should be even more beneficial because the branching factor grows with the board size.

The target game in this study is simple and it is necessary to determine a source suitable game manually. Future work can be carried out to overcome these limitations and to extend the study in several directions.

VI. Future Work

The main direction for future work is to investigate how patterns can be transferred in more complex and more challenging games, such as board games with multiple players and multiple piece types, and possibly in games that are not played on a 2D board. For two or more player games, opponents to learn against may not be available (as is the case in the GGP competition [2]). In that case, coevolution may be necessary to evolve game players [20].

In games with multiple types of pieces, like chess, each piece type can have its own pattern, applied by centering the pattern on that piece. The simplified source game can be obtained through simplified game features, such as a smaller board size or fewer piece types. In chess, end games can perhaps be used as the source game. Further, it may be possible to evolve blueprints to combine multiple local heuristics per piece type, while also evolving the local heuristics. The full HyperNEAT approach [14] could result in more general patterns as well.

A second main direction for future work is to extend or modify NEAT to make it more suitable for evolution of patterns. Activation functions that are better suited to generate patterns for board games might be utilized. A systematic study can be carried out to analyze how useful specific activation functions are for certain groups of board games. Furthermore, a new genetic operator that can merge two patterns in an intuitive way can be introduced. Such a reproduction operator could combine two CPPNs so that the resulting CPPN produces a weighted sum of the two patterns. However, such a new genetic operator should be designed carefully to avoid breaking innovation numbers.

A third direction focuses on choosing and creating an appropriate source task. Currently, the source task must be chosen manually such that it serves well as a stepping stone for the particular target task. Such construction may not be straightforward or possible in every task. Even if it is possible, it would be desirable to automate it. Taylor et al. [5] take a step towards this goal by providing a transfer hierarchy to make selecting source tasks easier. This process however, still does not fully automate the task selection.

Generalizing from source task selection, one can imagine a human-like learning scenario with a life-long learning of many tasks [21], going from easier tasks to more difficult tasks. For each new target task, relevant past tasks would be identified, and experience and patterns would be transferred from those tasks without creating new source tasks.

A fourth direction for future work is to learn abstract spatial patterns. Such a goal might be possible by utilizing a hierarchy while learning patterns, as was done by George et al. [22] and Serre et al. [23] to successfully learn to classify sets of patterns with a considerable variance. In addition to spatial patterns, temporal patterns can be learned as well, like Wang et al. did in learning a finite number of sequences [24]. Temporal and abstract patterns would prove useful in a life-long learning framework, since such generalizations would make patterns more broadly applicable and would allow the transfer of knowledge among a larger set of tasks.

VII. Conclusion

This paper presented an approach to utilizing transfer to improve game playing performance with local, pattern-based game heuristics evolved with NEAT. Given a target game, a simplified version of that game was used as the source task for population transfer. Compared to patterns evolved for the target game from scratch, transfer improved evolution in two ways: (1) Significantly higher performance was achieved after a fixed number of NEAT generations; and (2) the total learning time was reduced, where the total time for the transfer approach includes the time spent on the source task as well as the time for the target task. The results suggest that pattern transfer may allow scaling learning to complex games in the future.

Acknowledgements

This research was supported in part by NSF under grant EIA-0303609.

References

[1] M. Campbell, A. J. H. Jr., and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[2] M. R. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the AAAI competition," *AI Magazine*, vol. 26, no. 2, pp. 62–72, 2005.

[3] M. E. Taylor, S. Whiteson, and P. Stone, "Transfer via inter-task mappings in policy search reinforcement learning," in *Proc. of the 6th Int. Joint Conf. on Autonomous Agents and Multiagent Systems*, May 2007.

[4] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," in *Proc. of the 18th European Conf. on Machine Learning*, September 2007.

[5] M. E. Taylor, G. Kuhlmann, and P. Stone, "Transfer learning and intelligence: an argument and approach," in *Proc. of the 1st Conf. on Artificial General Intelligence*, March 2008.

[6] S. L. Epstein, J. Gelfand, and E. Lock, "Learning game-specific spatially-oriented heuristics," *Constraints*, vol. 3, no. 2-3, pp. 239–253, 1998.

[7] D. B. D'Ambrosio and K. O. Stanley, "A novel generative encoding for exploiting neural network sensor and output geometry," in *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2007)*, 2007, pp. 974–981.

[8] K. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[9] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[10] I. Harvey, P. Husbands, and D. Cliff, "Seeing the light: Artificial evolution, real vision," *From Animals to Animats*, vol. 3, pp. 392–401, 1994.

[11] F. Gomez and R. Mikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, no. 3-4, pp. 317–342, 1997.

[12] A. Christensen and M. Dorigo, "Incremental evolution of robot controllers for a highly integrated task," *LNCS*, vol. 4095, p. 473, 2006.

[13] K. Chellapilla and D. Fogel, "Evolving an expert checkers playing program without using human expertise," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 4, pp. 422–428, 2001.

[14] J. Gauci and K. Stanley, "A case study on the critical role of geometric regularity in machine learning," in *Proc. of the 23rd AAAI Conference on Artificial Intelligence (AAAI-2008)*. AAAI Press, 2008.

[15] S. L. Epstein, "For the right reasons: The FORR architecture for learning in a skill domain," *Cognitive Science*, vol. 18, no. 3, pp. 479–511, 1994.

[16] J. Resinger, K. O. Stanley, and R. Miikkulainen, "Evolving reusable neural modules," in *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2004)*, 2004.

[17] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. of the 2nd Int. Conf. on Genetic Algorithms*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1987, pp. 41–49.

[18] N. Kohl, K. O. Stanley, R. Miikkulainen, M. Samples, and R. Sherony, "Evolving a real-world vehicle warning system," in *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2006)*, 2006.

[19] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving neural network agents in the NERO video game," in *Proc. of the IEEE 2005 Symp. on Computational Intelligence and Games (CIG-2005)*, 2005.

[20] J. Reisinger, E. Bahçeci, I. Karpov, and R. Miikkulainen, "Coevolving strategies for general game playing," in *Proc. of the IEEE Symp. on Computational Intelligence and Games (CIG-2007)*, 2007.

[21] S. Thrun, *Explanation-Based Neural Network Learning: A Lifelong Learning Approach*. Kluwer Academic Publishers, 1996.

[22] D. George and J. Hawkins, "A hierarchical bayesian model of invariant pattern recognition in the visual cortex," in *Proc. of the Int. Joint Conf. on Neural Networks*, vol. 3, 2005, pp. 1812–1817.

[23] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, pp. 411–426, 2007.

[24] D. Wang and B. Yuwono, "Incremental learning of complex temporal patterns," *Neural Networks, IEEE Transactions on*, vol. 7, no. 6, pp. 1465–1481, 1996.