Acquiring Evolvability through Adaptive Representations

Joseph Reisinger and Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 USA
{joeraii,risto}@cs.utexas.edu

ABSTRACT

Adaptive representations allow evolution to explore the space of phenotypes by choosing the most suitable set of genotypic parameters. Although such an approach is believed to be efficient on complex problems, few empirical studies have been conducted in such domains. In this paper, three neural network representations, a direct encoding, a complexifying encoding, and an implicit encoding capable of adapting the genotype-phenotype mapping are compared on Nothello, a complex game playing domain from the AAAI General Game Playing Competition. Implicit encoding makes the search more efficient and uses several times fewer parameters. Random mutation leads to highly structured phenotypic variation that is acquired during the course of evolution rather than built into the representation itself. Thus, adaptive representations learn to become evolvable, and furthermore do so in a way that makes search efficient on difficult coevolutionary problems.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global Optimization; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

experimentation, design, performance

Keywords

genetic algorithms, evolvability, modularity, adaptive representations, indirect encodings, coevolution, neural networks

1. INTRODUCTION

Given a fixed set of variation operators (e.g. mutation and crossover), the genotypic representation determines what kinds of phenotypic variation can occur, and thereby structures the search. In general it is desirable to include as much domain knowledge as possible into the design of the representation. However, in many cases such domain knowledge is unavailable or difficult to encode meaningfully. In such cases it would be desirable to *learn* the optimal representation. Such *adaptive representations* can structure the effects of mutations and therefore result in more efficient search [2, 10, 22].

In this paper, an *implicit encoding* approach for evolving neural networks is developed based on representational principles inherent to genetic regulatory networks. The proposed representation is *generative*, encoding complex phenotypes with compact genotypes, *many-to-one*, allowing many different genotypic representations of the same phenotype, and based on *weak-linkage* between genes, allowing the representation to structure how mutations affect the phenotype.

The implicit encoding approach is evaluated in Nothello, a complex game-playing domain that is part of the AAAI General Game Playing Competition [9]. It is compared to two other neural network representations in this domain: A direct encoding and a complexifying encoding that generates arbitrarily complex network topologies. The implicit representation outperforms both encodings, evolving better solutions in fewer generations. Furthermore, as evolution progresses, the implicit encoding adapts variation significantly, making the representation more evolvable and the mutation effects more canalized (i.e. regular and consistent) in the phenotypes. By focusing on how implicit definition results in such canalization and evolvability, this work takes the first step towards determining what aspects of adaptive representations are necessary for creating powerful indirect encodings.

The paper is divided into five main sections: Section 2 describes the Nothello domain and the coevolutionary experimental setup, section 3 describes the three representations, section 4 presents the results and analyzes the genotypic and phenotypic structures learned, and section 5 draws conclusions on evolvability and points out areas for future work.

2. EXPERIMENTAL SETUP

The domain used to test the representations compared in this paper is Nothello, an Othello variant drawn from the AAAI General Game Playing Competition corpus [9]. Since Nothello is a two-player game, player strategies must be co-evolved in order to provide robust opponents for each population. This section describes the Nothello domain, the coevolutionary process employed for all experiments, and the mechanism for evolving heuristic evaluators.

2.1 Nothello

As in Othello, game play in Nothello proceeds in turns

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

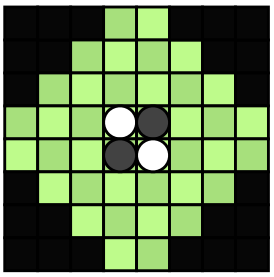

Figure 1: Starting configuration in Nothello. Game play proceeds identically to Othello except the player with the fewest pieces wins. Nothello is well suited for empirical study using coevolution because it is simpler than Othello yet still challenging.

where each player places a piece on the board, flipping one or more of the opponents pieces that are sandwiched between the player's pieces Nothello introduces two major differences: First, the game is played on a diamond shaped board (the original Othello board without the corners; figure 1) and second, the winner is the player with the fewest pieces on the board in the end. These modifications make the game play shorter, but maintain much of the complexity of Othello. The Nothello domain was chosen primarily because it is a difficult learning problem; although simpler than Othello, Nothello has many of the same features and presents a challenge for learning methods, particularly because no standard benchmark opponents exist.

2.2 Coevolution Process

To ensure that the evolved heuristics work well against a range of opponents, the opponent strategies themselves are evolved. In coevolution, an individual is evaluated against some combination of opponents drawn from the evolving populations, rather than against a fixed fitness metric. This approach has several advantages over traditional evolution: (1) Opponent strategies are learned by the algorithm, reducing the amount of information the algorithm designer must provide a priori; (2) coevolution may reduce the total number of evaluations necessary to learn successful behavior, leading to more efficient search [6]; and (3) under certain conditions, coevolution results in an *arms race*, where individuals in both populations try to outdo each other, and end up learning more innovative behaviors [23].

In order to facilitate arms races and make coevolution efficient, the algorithm needs to ensure that monotonic progress is made. Without such a guarantee populations can "forget" past strategies, resulting in cycling behavior [6, 8]. To ensure monotonic progress in Nothello, a simplified variant of MaxSolve [6], a coevolutionary solution concept for maximizing the expected utility of each individual, is used. Such a concept is useful in situations where the performance of a strategy must be measured based on a limited set of experiences. Formally, for a set of candidate solution strategies $\mathbf{C}$, a set of test strategies $\mathbf{T}$ and a game with payoffs $u_{i\in\{C,T\}}$, the MaxSolve solution concept is a set of strategies $\mathbf{S}$ that maximize the expected utility with respect to a test $T \in \mathbf{T}$ is defined as

$$S_T = \{C \in \mathbf{C} | \forall_{C' \in \mathbf{C}} : E(u_C(C,T)) \geq E(u_{C'}(C',T))\}. \quad (1)$$

A candidate C is added to the $\mathbf{T}$ when $C \in \bigcap_{T\in\mathbf{T}} S_T$. This solution concept can then be implemented algorithmically by maximizing the sum of an individual's utilities across all tests. Although this formulation assumes that all tests are weighted equally, it has been shown to perform well in practice [6].

2.3 Evolving Heuristic Evaluators

The coevolutionary process described above is used to evolve neural networks to estimate the heuristic score for each game state in Nothello. The board state is mapped onto a 64 input units. Each square on the board corresponds to a single input, with its value given as 1 if a white piece occupies the square, -1 if a black piece occupies it, or 0 if the square is empty or forbidden (e.g. at the board corners).

The neural networks serve as heuristic board evaluators that estimate the value of game states when explicit goal information is not available. These heuristic evaluators are combined with standard lookahead search using α-β-pruned minimax [12]. In all experiments presented in this paper, lookahead search is restricted to a single ply in order to reduce evaluation time, allowing longer evolutionary runs.

3. COMPARING REPRESENTATIONS

Three different neural network representations are compared using the coevolutionary setup described above: A direct weighted mapping of input features to a heuristic value, a complexifying neural network using NeuroEvolution of Augmenting Topologies (NEAT), and an implicit representation based on properties of genetic regulatory networks. Each representation biases search in a different manner, leading to different performance. This section discusses each separate encoding and details how each of them affects the structure of search.

3.1 Direct Mapping

The first representation maps the 64 state features through weighted connections directly to the heuristic value of the state. Every possible state feature s is assigned a real-valued parameter $[-1, 1]$ in the genome. This parameter determines the weight that the feature adds to the state valuation v_S for state S, i.e.

$$v_S = \sum_{s\in S} w_s. \quad (2)$$

Although such direct encodings are conceptually simple, they often outperform more complex encodings. However, they cannot learn more efficient mutation parameterizations over time. For example, in the encoding specified above, mutations cannot exploit symmetries inherent in the fitness function. That is, if several weights were discovered to be highly correlated, it would make sense to employ mutations that were correlated in the same way. It is not possible to learn such a structure with a direct mapping, which may lead to poor performance in more complex domains.

3.2 Complexifying Neural Network

The second representation uses a topologically complex neural network as the heuristic evaluation function. Inputs are provided in the same manner as with the direct mapping, but instead of a fixed weighting, a multi-layer neural network is employed to map them to the heuristic score. The appropriate network structure is found using NeuroEvolution of Augmenting Topologies (NEAT) [21]. NEAT evolves network topologies automatically to fit the complexity of the problem while simultaneously optimizing network weights.

Each genome in NEAT includes a list of *connection genes*, each of which refers to two *node genes* being connected. Each connection gene specifies the in-node, the out-node, the weight of the connection, whether or not the connection gene is expressed (an enable bit), and an *innovation*

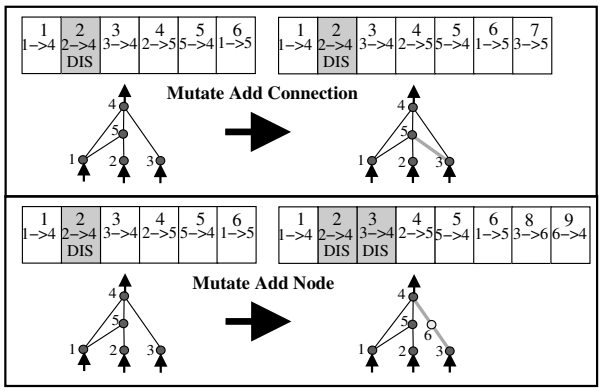

Figure 2: Structural mutation in NEAT. Genetic representations are shown above their corresponding phenotypes. Each gene contains an innovation number (top), input and output node specifications (middle), a possible "disabled" symbol, and a weight (not shown). New connections and nodes are added to the network by adding new connection genes to the genome. In this manner, topological structure is added incrementally to the genome.

number, which allows finding corresponding genes during crossover (figure 2). Innovation numbers are inherited and allow NEAT to perform crossover without the need for expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies [15] is essentially avoided. NEAT populations are seeded with fully-connected networks with no hidden nodes. During evolution, networks with increasingly complex topologies are generated through "add node" and "add link" mutations (figure 2). This approach is highly effective: NEAT outperforms other Reinforcement Learning methods on control tasks like double pole balancing and robotic strategy-learning [21]. These properties make NEAT an attractive method for evolving neural networks in complex tasks.

The NEAT neural network representation was chosen for two reasons. First, its principled complexification works well in competitive coevolution: As the antagonistic populations refine their strategy, complexification allows evolution to generate novel strategies in response, without *forgetting* past strategies [21]. Second, as networks become more complex their genetic representations becomes more canalized and modular. Canalization emerges as redundant nodes are added into the genome (figure 3a). Such nodes increase the number of mutations required to make a significant change to the heuristic function. Modularity emerges as hidden nodes are added; they aggregate the activation from several network subcomponents, organizing them into modules (figure 3b). However, as the network becomes more complex, it becomes harder to form such modules because every subcomponent needs to be connected to the hidden nodes. It is unlikely that such mutations can be realized in concert even with high mutation rates. In other words, modularity develops *uninclusively* in NEAT: Each low-level component must be added explicitly because there are no mutations that would affect larger structures.

Therefore, as networks become more complex in NEAT, it becomes increasingly difficult to restructure fundamental

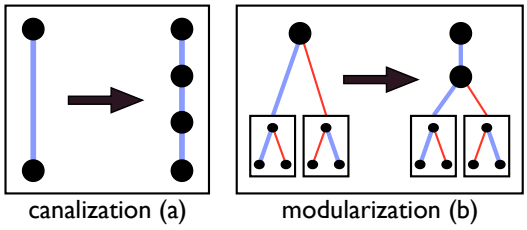

Figure 3: Canalization and modularization in NEAT. Canalization can arise in NEAT through the addition of redundant hidden nodes. Modularization arises through the addition of hidden nodes connected to one or more sub-networks, however the number of mutations required to create new modules increases as the network complexity increases.

design choices, even if those choices turn out to be unsustainable. Furthermore, no *simple* methods exist that correlate the phenotypic effects of mutations, even at the most basic level of weights. Thus, it may take many hundreds of mutations to realize such restructuring through mutation. A powerful extension to this approach, discussed in the next section, is to represent connections implicitly, making it possible to learn and represent correlations not only between individual weights, but between entire subnetworks as well.

3.3 Implicit Encoding

The third representation is inspired by the implicit regulation of protein transcription in genetic regulatory networks (GRNs) and is derived from that in [2]. The *implicit encoding* represents a neural network based on if-then production rules. In prior work, several variants of GRN encodings have been proposed [2, 4, 7]. These models are all *multicellular*, where the GRN evolution over time describes cell division and differentiation. In contrast, the implicit encoding evaluated in this paper utilizes only the representational aspect of GRNs, without temporal or spatial development. An important goal is to test whether such encodings are evolvable, even without adding more complex features.

3.3.1 GRN-Motivated Representation

In the implicit encoding approach, an arbitrarily complex neural network is represented using a sequence of if-then production rules. Each rule contains an antecedent (regulatory) region and a product (transcription) region. Both the antecedent and product regions are composed of one or more real-valued *regulatory factors* (figure 4). The regulatory factors in the antecedent are paired with tolerance values, and the regulatory factors (products) in the product regions represent either hidden or output nodes in a neural network. When the values of products match the values specified in an antecedent region within the given tolerance, connections between nodes are generated. The incoming connections to a given node are determined implicitly by its antecedents. If the matched antecedent is has a negative, then the resulting connection is inhibitory, otherwise it is excitatory. The connection weight is given by

$$w_{ij} = \sum_{\mathbf{g} \in \mathbf{G}|j} \sum_{(a,\tau) \in \mathbf{g.a}} \frac{1}{1 + \exp(-\tau(|i| - |a|)^2)}, \qquad (3)$$

where $i, j \in \Re_+$ are unique regulatory factors, $\mathbf{G}|j$ denotes all genes in genome $\mathbf{G}$ that contain j as a product, $\mathbf{g.a} \in$

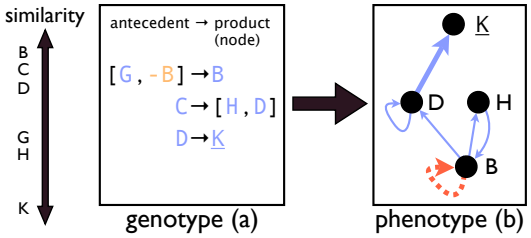

Figure 4: Implicit encoding of a neural network. Regulatory factors in the product region, e.g. B, H, and D of (a), become hidden nodes in the neural network (b) and K is designated as an output node. Connections are created between nodes based on the *similarity* of the product (node) regulatory factor to regulatory factors in another product's (node's) antecedent region; e.g. hidden node B connects to D via similarity to C). Connections are either excitatory (solid line) or inhibitory (dotted line) depending on the sign of the antecedent. The weight is determined by the similarity according to equation 3 (e.g. the connection between D and K (through D) is stronger than between B and H (through C); tolerance values are not shown). In this manner, the implicit encoding does not require explicit representation of every neural network weight, biasing search in an efficient and evolvable way. For a more detailed example, see http://www.cs.utexas.edu/users/joeraii/adaptive.

$(\Re_+ \times \Re_+)^n$ is the set of antecedents in gene $\mathbf{g}$, and τ specifies the tolerance range of the antecedent a. Tolerance determines how similar two regulatory factors must be in order for a connection to form. A simple example of this process is shown in figure 4.

The neural network inputs are connected to the resulting structure using the same implicit tolerance mechanism as for the other nodes. Each unique input is assigned a fixed regulatory factor, and connections to the hidden and output nodes are assigned weights using equation 3.

Initially each genome contains five genes with two antecedents and two products each. Additional regulatory structure is added through three types of mutations: Duplicate gene, add regulatory factor (either to the antecedent or product region) and change a weight. Innovation numbers similar to those in NEAT are used to perform crossover, since the genomes have variable length [21]. Innovation numbers record when new regulatory factors or new genes are introduced to a genome. During crossover, the innovation numbers are lined up so that similar coding regions can be crossed over correctly.

3.3.2 *Evolvability and Canalization Properties*

Implicit encoding has several advantages over direct mapping and NEAT encoding. First, in both the direct encoding and NEAT, mutations can only affect individual weights. In contrast, in the implicit encoding, mutations can have not only small precise effects on individual weights, but also large, even global effects on nodes and groups of nodes, depending on the connectivity structure (i.e. the encoding utilizes weak-linkage between genes [17]). Because the network structure is described implicitly, complex networks can be represented more compactly than with traditional explicit encodings (i.e. the encoding is generative and many-to-one).

Second, GRNs inherently support the "duplication and divergence" process whereby a gene is first duplicated, the copy then begins to develop some new function, and slowly diverges from the original [16]. This phenomenon allows new function to develop while the old regulatory structures are preserved. Duplication and divergence may be one of the sources of modularity in biology, and in turn modularity is important for evolvability [17]. Implicit encoding supports duplication and divergence genetically through the duplicate-gene mutation, and phenotypically through the add-product mutation, which duplicates an entire hidden node in the network.

Third, complex GRN network models are often canalized, i.e. correspond to highly constrained phenotypes [3], which is considered to be one necessary component for adaptive complexity [20]. An implicit encoding can become canalized if many genes control a given connection (thus reducing the effects of a single mutation) but also through the tolerance values, which can lower the number of interactions in which a regulatory factor participates. Thus the implicit encoding exhibits many of the representational features associated with evolvability, unlike the direct or NEAT encodings.

4. RESULTS

This section details the experimental results comparing the three representations on the coevolutionary Nothello domain. It is divided into three parts: Section 4.1 compares the learning curves for the three representations against a 1-ply lookahead opponent with a random heuristic, 4.2 analyzes differences in the genotypes of NEAT and the implicit encoding, and 4.3 analyzes differences in their phenotypes.

4.1 Learning Curves

Each representation was evolved on the coevolutionary Nothello domain for 200 generations in 10 independent runs. Since fitness is a function of both evolving populations, it cannot be used directly to compare the different representations. Instead, separate tests against a 1-ply opponent using a random heuristic evaluator were conducted. Each evolved network played 200 games against this opponent and the winning percentage of the current population champion at each generation was averaged over the 10 runs.

Playing as white, the implicit encoding significantly outperforms the NEAT encoding in all generations, winning 99.1% of matches on average in generation 200, while NEAT wins only 97% ($p < 10^{-4}$, measured using Student's t-test; figure 5a). The implicit encoding also outperforms the direct encoding in all generations, however the difference is only significant in the first 40 generations. In generation 200, the direct encoding wins 98.6% of matches ($p = 0.11$). Furthermore, the implicit encoding reaches the 95% accuracy level on average more than 20 generations faster than the direct encoding, and 50 generations faster than NEAT.

Playing as black, the implicit encoding outperforms both the NEAT encoding and the direct encoding in all generations, however for NEAT the difference is significant only until generation 30. In generation 200, the implicit encoding wins 93.1% of games, NEAT wins 91.6% ($p = 0.06$), and the direct encoding wins 91.5% ($p = 0.01$). During early evolution, the implicit encoding reaches the 90% performance level 36 generations earlier than NEAT and 88 generations earlier than the direct encoding. Thus, the implicit encoding outperforms both encodings and is able to find better solutions in significantly fewer generations.

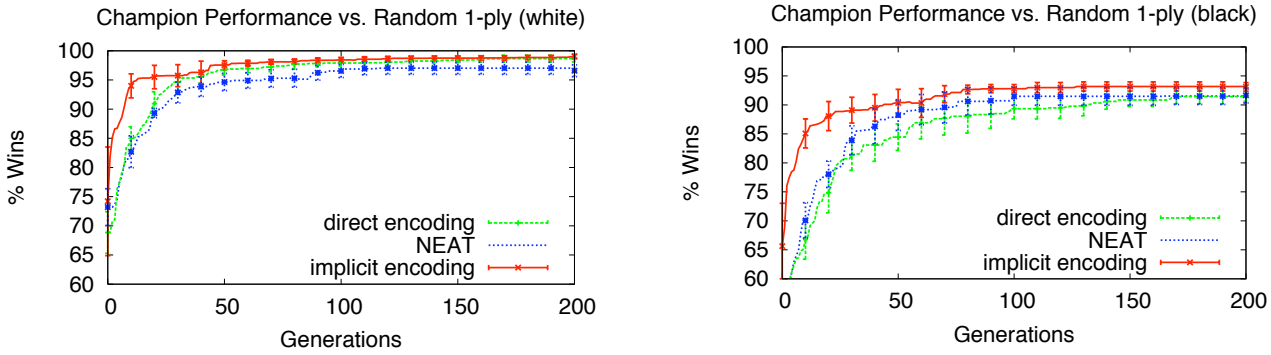

Figure 5: Fitness of evolved heuristic evaluators. Generation champions are compared using 200 trials against a 1-ply lookahead opponent with a random heuristic. The results are averaged over 10 runs and 95% confidence intervals are shown every 10 generations. The implicit encoding significantly outperforms both the direct and NEAT encodings.

4.2 Genotypic Analysis

In order to understand the performance differences between the direct encoding, implicit neural network encoding and the standard NEAT encoding, their genotypes are analyzed in this section, investigating how they differ in terms of genome size, evolvability and canalization.

4.2.1 Genome Size

The number of parameters in each genome determines how many mutations are required to make significant changes. In the direct encoding, the number of parameters is fixed at 64, one for each input feature. Final champions using the NEAT representation have 240.7 weights on average. In contrast, final champions using the implicit encoding average 458.25 connections on average, but use only 34.12 parameters. The implicit encoding is therefore significantly more compact than either the direct encoding or the NEAT encoding ($p < 10^{-21}$). Thus the implicit encoding not only generates more efficient neural networks that are twice as complex as NEAT, but does so with an order of magnitude fewer parameters. Reducing the size of the parameter space alone may not necessarily simplify problem. However, the next section provides empirical evidence that the implicit encoding reduces the number of parameters while maintaining high evolvability, indicating that the implicit encoding is indeed able to reduce the problem complexity.

4.2.2 Acquired Evolvability

The evolvability of a genome can be approximated with the fitness of the local mutation landscape around that genome [19]. Run champion genomes using the implicit encoding are more robust against mutation than those using either the direct or NEAT encoding (figures 6, top two rows; since no significant differences were found between the direct and NEAT encoding, the direct encoding is omitted from this section for clarity). For the white role, the implicit encoding beats the random heuristic in approximately 96.1% of matches, while NEAT wins approximately 93.4% of matches. After a single point mutation, the performance of implicit encoding drops to 91.0% while that of NEAT drops to 81.3%. After two mutations the performances decrease to 87.0% and 71.1%, respectively. A similar trend is seen for the black role: Initially both the implicit encoding and NEAT win 86.0% of matches. However, after a single mutation, the implicit encoding only drops to 81.7%, while NEAT drops to 69.4%. As more mutations are made, the differences in the two means continue to increase, indicating that for both roles, genomes using the implicit representation are more robust against mutation than those using the NEAT encoding.

One explanation could be that the implicit encoding makes smaller mutations on average than NEAT. However, this is not the case: Using the network compatibility measure [21]

$$\delta(i,k) = \frac{E(i,k) + D(i,k)}{\max(||i||, ||k||)} + 3.0 \cdot \bar{W}(i,k), \qquad (4)$$

where i, k are the networks being compared, $||i||$ is the number of weights in network i, E is the number of excess weights between network i and k, D is the number of disjoint weights and $\bar{W}$ is the average difference in weights, the implicit encoding on average makes weight and topology mutations two to five times as strong as the NEAT encoding (figure 6, bottom two rows). Taken together, these results indicate that the implicit encoding is indeed more evolvable than the standard NEAT encoding; that is, the implicit encoding generates more *adaptive* variation than NEAT.

Much of the robustness achieved by the final champions is acquired during the course of evolution (figure 6a vs. 6b). For the white role, as mutations are made to the first generation champions, there is no significant difference in performance between the implicit and NEAT encoding at any mutation level tested. For the black role, the implicit encoding significantly outperforms the standard encoding after any number of mutations greater than one. However, the difference between the two means does not increase as fast as it does for the final champions: After one mutation, the first generation champions differ by 7.8 on average, while the final champions differ by 12.3; after two generations the differences increase to 13.7 and 18.4, after 5 generations to 17.1 and 24.25 and after 10 generations to 13.1 and 27.8. Thus, the implicit encoding has acquired evolvability with respect to the Nothello domain during evolution.

4.2.3 Canalization

Representational canalization, i.e. the degree to which genotypic changes affect the phenotype, can be measured by averaging phenotypic differences after a fixed number of mutations. Phenotypic distance is calculated using the compatibility measure described in the previous section. For

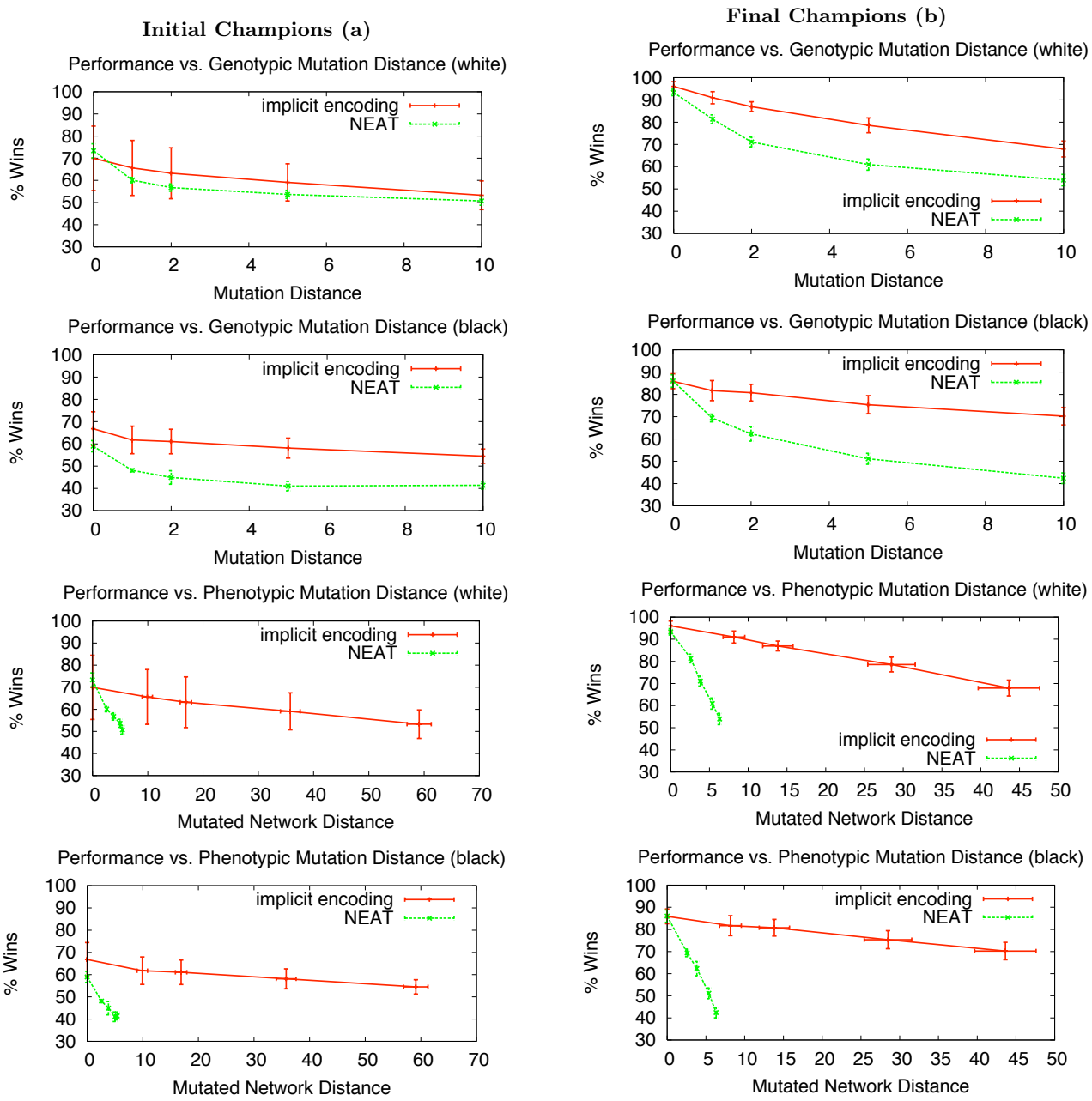

Figure 6: Mutation landscapes around the first generation (a) and run final champions (b). The top two graphs in both (a) and (b) show how the performance decreases with the number of mutations and the bottom two graphs show how it decreases with phenotypic distance, using the compatibility measure from NEAT. The implicit encoding's performance degrades significantly slower than NEAT and exhibits significantly higher phenotypic variation.

the NEAT encoding, making 10 mutations to a champion genome in the first generation results in networks with compatibility distance 5.41. Final champions, in contrast, have a distance of 6.3 on average ($p < 10^{-17}$), indicating that NEAT networks become biased towards larger mutations as evolution progresses. Using the implicit encoding, the first generation champions have compatibility distance 59.1 on average after 10 mutations, while final champions have compatibility only 43.6 ($p < 10^{-6}$). Since genotypic mutation rates remain constant throughout evolution, this trend indicates that effects of such mutations canalize over time.

Another indicator of a representation's canalization is how much fitness changes in response to mutation. Comparing figures 6a and 6b, it is clear that final fitnesses are less affected by mutation. For the white role, the standard error of the fitness mean after a single mutation in the first generation is 12.4, while the final champions' standard error is only 1.4. Similarly for the black role, the error decreases from 6.2 to 4.5. On the other hand, the standard error for the NEAT encoding increases with evolution from 1.2 to 2.0 as white and from 0.6 to 6.1 as black, indicating that genomes with less canalization are being selected. Thus, there is strong evidence that the properties of the implicit encoding not present in NEAT are responsible for canalization.

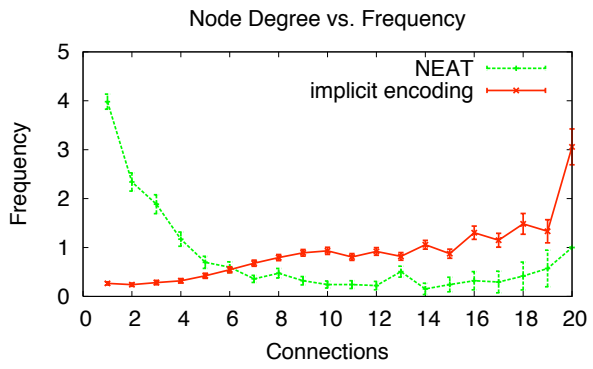

Figure 7: Neural Network Connectivity Analysis. With NEAT, the number of connections per node is distributed according to a power law, while networks evolved using the implicit encoding are more tightly connected. This result indicates that power law connectivity alone is not sufficient for evolvability.

4.3 Phenotypic Analysis

To understand the effects of representation on the evolved phenotypes, two methods are employed: comparing the degree distribution of nodes and analyzing the network motifs.

4.3.1 Degree Distribution

Figure 7 depicts the degree distribution, i.e. average number of connections per node for the final champion neural networks evolved with NEAT and with the implicit encoding. With NEAT, node degree is distributed roughly according to a power-law ($P(k) \sim k^{-\gamma}$), with exponentially more nodes of lower degree than of higher degree. Node degree ranges from one to 22, with an average of 4.0 nodes of degree one, 2.3 nodes of degree two, 1.9 nodes of degree three and 1.2 nodes of degree four. All other node degrees occur with frequency less than 1.0.

In contrast, implicit encoding results in more dense connection patterns. Number of connections ranges from one to 25, but few nodes are sparsely connected and most have ten or more connections. Overall, node frequency increases linearly with the number of connections (the data for nodes of degree 20 and higher is highly variable and is bucketed into 20). This result is counter-intuitive: In NEAT, learning such a complex, interconnected structure is not evolvable as each weight must be mutated in isolation.

4.3.2 Network Motifs

In order to characterize specific structural differences in evolved network structures, network motifs need to be analyzed. In motif analysis [14], the frequencies of small three-node subnetworks (triad motifs) are compared relative to random networks with the same number of nodes and connections. A significance score for each motif is calculated as its z-score:

$$Z_i = (N\mathrm{e}_i - \bar{N}\mathrm{r}_i)/\mathrm{std}(N\mathrm{r}_i), \quad (5)$$

where $N\mathrm{e}_i$ is the frequency of motif i in the evolved network, $\bar{N}\mathrm{r}_i$ is the average frequency of motif i in random networks and std is the standard deviation in the random networks. The z-score profiles are normalized and reported as significance profiles

$$SP_i = Z_i/(\sum_k Z_k^2)^{1/2}. \quad (6)$$

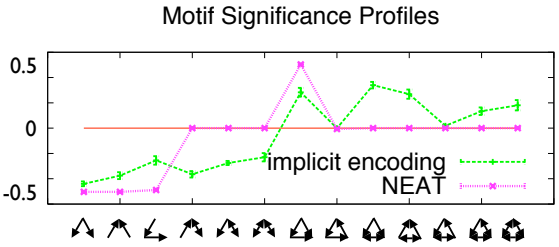

Figure 8: Network motifs in NEAT and in implicit encoding. For each motif (shown along the horizontal axis), the significance profiles are shown relative to those in randomly connected networks with the same number of nodes and connections. NEAT exhibits a strictly feed-forward profile, while the implicit encoding demonstrates a high degree of recurrent connectivity.

Figure 8 shows the significance profiles for the NEAT networks compared to the implicitly encoded networks. NEAT networks have a profile similar to that of feed-forward networks [13]: Only a single triad motif called the *feed-forward loop* occurs significantly more often in NEAT than in the random networks, and motifs with low connectivity (1-3) occur less often.

In contrast, the implicitly encoded networks have several more significant motifs of higher order (7-10, 12,13). These motifs are common in highly connected networks and minimize the number of edges connecting any two nodes. Such motif significance profiles are typical of neural networks found in nature [13]. Such highly recurrent network structures may be used to increase the *detectability* of mutations affecting evolvability [18].

5. DISCUSSION AND FUTURE WORK

Several observations suggest that the implicit encoding is highly evolvable: The networks in the local mutation space around the champion genomes are highly varied, but perform significantly better than mutated genomes from NEAT. Furthermore, such evolvability is acquired during evolution: The champions' fitness varies more in the beginning and much less in the end while the evolved network structures maintain high variance. One possible reason is that with the implicit encoding, evolution controls how the phenotypes are distributed, thereby transforming random mutations into structured phenotypic variation. Furthermore, since such mutations have an immediate and *detectable* impact on fitness, fewer mutations are required on average to create a gradient for selection [18].

This result suggests that in order to maximize evolvability, representations should be constructed in such a way that phenotypic variation can be adapted to match the structure of the fitness function in as few mutations as possible (i.e. mutations affecting phenotypic variation must be as easy to detect as possible [18]). Furthermore, representations must be able to adapt the genotype-phenotype mapping at all levels, ranging from fundamental design changes to small phenotypic tweaks, using as few mutations as possible.

Encodings based on GRNs and developmental systems allow for this kind of adaptation implicitly through overlapping gene expression domains [5, 11, 16]: (1) Weak linkages allow mutations to affect both fundamental and fine-tuned structure, (2) Expression domains can be easily copied between genes, and (3) Upstream mutations can shift expres-

sion domains. These mechanisms are powerful precisely because search becomes constrained, generating only highly adaptive phenotypes. In evolutionary computation, acquiring such constraints is akin to learning the underlying structure of a particular fitness function: such structure can then be exploited to make the search more efficient.

Scale-free organizations may be highly evolvable, as has been suggested in theoretical biology literature [1]. However, the phenotypic analysis in this paper shows that implicit encoding is more evolvable than NEAT. This result suggests that scale-free organization alone is not sufficient, and it may actually be more important to control the genotype-phenotype mapping.

There are two main areas of future work. First, the implicit encoding should be tested on a variety of domains to determine how general its performance benefits are. Second, the implicit encoding can serve as a basis upon which other aspects of developmental encodings can be built. The next logical step is to add entirely regulatory gene products, i.e. products that do not correspond directly to hidden neurons. Such functionality would allow for *upstream* mutations that can have complex phenotypic expression resulting in hierarchical modularity [17]. Encodings making use of such features may prove to be more efficient and evolvable, allowing more complex problems to be solved.

6. CONCLUSION

An implicit encoding of neural network weights and topologies was shown to outperform a direct encoding and the NEAT topology-and-weight evolving method on a complex board-game task. The implicit encoding represents complex networks with an order of magnitude fewer parameters than NEAT. Furthermore, the implicit encoding results in more adaptive variation in response to mutation as well as more phenotypic variability. Thus, the results indicate that adaptive representations are more evolvable than direct encodings, and that such encodings can acquire additional evolvability with respect to specific fitness functions as evolution progresses. Ultimately, such encodings will yield more efficient search in complex domains.

Acknowledgments

This research was supported in part by the National Science Foundation under CISE Research Infrastructure Grant EIA-0303609 and through a Graduate Research Fellowship to the first author.

7. REFERENCES

[1] A.-L. Barabási and Z. N. Oltvai. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*, 5:101–113, 2004.

[2] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 35–43, San Francisco, 1999. Kaufmann.

[3] A. Bergman and M. L. Siegal. Evolutionary capacitance as a general feature of complex gene networks. *Nature*, 424:549–552, 2003.

[4] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proc. of CEC 2002*, pages 1872–1877. IEEE Press, 2002.

[5] E. H. Davidson. *Genomic Regulatory Systems*. Academic Press, San Diego, 2001.

[6] E. de Jong. The maxsolve algorithm for coevolution. In *GECCO '05: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 483–489, New York, NY, USA, 2005. ACM Press.

[7] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. *Proc. of the 4th European Conf. on Artificial Life*, pages 205–213, 1997.

[8] S. G. Ficici. Monotonic solution concepts in coevolution. In *Proc. of the 2005 Conference on Genetic and Evolutionary Computation*, pages 499–506, New York, NY, USA, 2005. ACM Press.

[9] M. R. Genesereth, N. Love, and B. Pell. General game playing: Overview of the aaai competition. *AI Magazine*, 26(2):62–72, 2005.

[10] G. S. Hornby. Functional scalability through generative representations: The evolution of table designs. *Environment and Planning B: Planning and Design*, 31(4):569–587, 2004.

[11] M. Kirschner and J. Gerhart. Evolvability. *PNAS*, 95:8420–8427, 1998.

[12] D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Art. Intelligence*, 6:293–326, 1975.

[13] R. Milo, S. Itzkovitz, N. Kashtan, R. Levitt, S. Shen-Orr, I. Ayzenshtat, M. Sheffer, and U. Alon. Superfamilies of evolved and designed networks. *Science*, 303(5663):1538–1542, March 2004.

[14] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[15] N. J. Radcliffe. Genetic set recombination and its application to neural network topology optimization. *Neural computing and applications*, 1(1):67–90, 1993.

[16] R. A. Raff. *The Shape of Life: Genes, development, and the Evolution of Animal Form*. The University of Chicago Press, 1996.

[17] R. A. Raff and B. J. Sly. Modularity and dissociation in the evolution of gene expression territories in development. *Evolution and Development*, 2(2):102–113, 2000.

[18] J. Reisinger and R. Miikkulainen. Selecting for evolvable representations. In *Proc. of the 2006 Genetic and evolutionary computation conference*, pages 257–264, New York, NY, USA, 2006. ACM Press.

[19] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. Fitness landscapes and evolvability. *Evolutionary Computation*, 10(1):1–34, 2002.

[20] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Art. Life*, 9(2):93–130, 2003.

[21] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *J. of Artificial Intelligence Research*, 21:63–100, 2004.

[22] M. Toussaint. Compact representations as a search strategy: compression edas. *Theoretical Compututer Scicience*, 361(1):57–71, 2006.

[23] L. Van Valin. A new evolutionary law. *Evolution Theory*, 1:1–30, 1973.