In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO 05), pp. 1225-1232, Washington, D.C., June 2005.

Automatic Feature Selection in Neuroevolution

Shimon Whiteson
Dept. of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233
shimon@cs.utexas.edu

Peter Stone
Dept. of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233
pstone@cs.utexas.edu

Kenneth O. Stanley
Dept. of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233
kstanley@cs.utexas.edu

Risto Miikkulainen
Dept. of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233
risto@cs.utexas.edu

Nate Kohl
Dept. of Computer Sciences
University of Texas at Austin
1 University Station, C0500
Austin, TX 78712-0233
nate@cs.utexas.edu

ABSTRACT

Feature selection is the process of finding the set of inputs to a machine learning algorithm that will yield the best performance. Developing a way to solve this problem automatically would make current machine learning methods much more useful. Previous efforts to automate feature selection rely on expensive meta-learning or are applicable only when labeled training data is available. This paper presents a novel method called FS-NEAT which extends the NEAT neuroevolution method to automatically determine an appropriate set of inputs for the networks it evolves. By learning the network's inputs, topology, and weights simultaneously, FS-NEAT addresses the feature selection problem without relying on meta-learning or labeled data. Initial experiments in an autonomous car racing simulation demonstrate that FS-NEAT can learn better and faster than regular NEAT. In addition, the networks it evolves are smaller and require fewer inputs. Furthermore, FS-NEAT's performance remains robust even as the feature selection task it faces is made increasingly difficult.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*

General Terms

Experimentation

Keywords

genetic algorithms, neural networks, feature selection

1. INTRODUCTION

To implement a successful machine learning system, human designers must make several preliminary decisions. For example, setup usually requires choosing a representation for the solution, selecting relevant inputs, and setting all the parameters associated with the given learning method. Performing these tasks manually is time-consuming and may yield sub-optimal results if not done correctly. Hence, one important goal of machine learning research is to develop techniques that shift some of the burden off the human designer and place it onto the learning algorithm itself. Doing so will reduce the time and expertise required to use machine learning techniques and greatly broaden the set of tasks to which they can be practically applied.

One of the decisions that is usually addressed manually is the *feature selection* problem. In many real world tasks, the set of potential inputs, or features, that can be fed to the learning algorithm is quite large. Feature selection is the process of determining which subset of these inputs should be included to generate the best performance. Doing so correctly can be critical to the success of a machine learning system. If any important features are excluded, it may be impossible to find an optimal policy. On the other hand, including superfluous inputs can also impede learning. Since each input adds at least one dimension to the search space, even a few extraneous features can be detrimental. However, the consequences of sub-optimal feature selection are not limited just to the learner's performance. If adding inputs costs money (e.g. putting more sensors on a robot), then pruning out unnecessary features can be vital.

Feature selection can often be performed by a human with the appropriate domain expertise. However, in some domains, no one has the requisite knowledge and even when experts do exist, employing them can be expensive and time-consuming. In such domains, automatic feature selection is necessary. Langley [6] divides feature selection techniques into two categories: *filters* and *wrappers*. Filters [1, 5] analyze the value of a feature set without regard to the learning algorithm that will use those features. Instead, they rely on labeled data. The data is analyzed to determine which features are most useful in distinguishing between the category labels. This approach has been successful but works only in supervised learning tasks. In reinforcement learning scenarios, when no labeled data is available, filtering techniques are not applicable.

By contrast, wrappers [7, 8] test a feature set by applying it to the given learning algorithm and observing its performance. Labeled examples are not necessary so this approach can be used in reinforcement learning tasks as well. However, it requires a meta-learner to search through the space of feature sets; evaluating any point in that space requires an entire machine learning run of its own. For most real-world problems, this approach is computationally infeasible.

Feature Selective NeuroEvolution of Augmenting Topologies (FS-NEAT) is a new learning method that avoids such limitations by incorporating the feature selection problem into the learning task. FS-NEAT searches for good feature sets at the same time as it trains networks that receive those features as input. Hence, it does not depend on human expertise, labeled data sets, or meta-learning.

FS-NEAT is based on the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [10], which evolves both the topology and weights of a neural network. NEAT already goes a long way towards reducing the burden on human designers. While most neuroevolution methods [14] evolve only the weights of a network with fixed topology, NEAT discovers an appropriate topology automatically. This approach is highly effective: NEAT outperforms other neuroevolution methods on many difficult reinforcement learning tasks [10, 11]. FS-NEAT goes one step further than regular NEAT by learning the network's inputs too. Using evolution, it automatically and simultaneously determines the network's inputs, topology, and weights. Harvey et al. [4] also used neuroevolution to find useful subsets of available features though, unlike FS-NEAT, their system still requires a human to specify the size of that subset in advance. In [13], an earlier implementation of FS-NEAT was presented along with initial results on a contrived supervised learning problem. This article presents a simpler implementation and results on a reinforcement learning problem.

A critical feature of NEAT is that it begins with networks of minimal topology (i.e. with no hidden nodes and all inputs connected directly to the outputs). As evolution proceeds, NEAT adds links and hidden nodes through mutation. Since only those additions that improve performance are likely to be retained, it tends to find small networks without superfluous structure. Starting minimally also helps NEAT learn more quickly. When networks in its population are small, it is optimizing over a lower-dimensional search space; it jumps to a larger space only when performance in the smaller one stagnates.

FS-NEAT further exploits this same premise. It begins with a population of networks that are even smaller than in regular NEAT. These networks contain no connections at all (save those added by an initial mutation step) and are little more than pools of inputs and outputs. Evolution then proceeds as in regular NEAT, with hidden nodes and links added through mutation. Feature selection occurs implicitly as only those links emerging from useful inputs will tend to survive.

In addition to introducing this novel method, this paper presents experiments comparing FS-NEAT to regular NEAT in a challenging reinforcement learning domain: an autonomous car racing simulation called RARS [12]. The results of these experiments confirm that when some of the available inputs are redundant or irrelevant, FS-NEAT can learn better and faster than regular NEAT. In addition, these results demonstrate that the networks FS-NEAT evolves are smaller and require fewer inputs.

The remainder of this paper is organized as follows. Section 2 explains the NEAT algorithm and the modifications made to it to yield FS-NEAT. Section 3 introduces the RARS domain and presents the results of our experiments comparing FS-NEAT to regular NEAT. Section 4 discusses the implications of these results and Section 5 outlines some opportunities for future work.

2. METHOD

This section provides background on the NEAT algorithm. It also introduces FS-NEAT, our novel modification to the original NEAT method.

2.1 NeuroEvolution of Augmenting Topologies

In most domains, the optimal topology and complexity for a neural network is unknown. With most neuroevolution techniques [14], a human designer must try to guess it. Since the topology determines the size of the search space, the consequences of guessing wrong can be severe. Searching in too large a space is intractable while searching in too small a space limits solution quality. NeuroEvolution of Augmenting Topologies (NEAT) [10] is a neuroevolution technique that does not require a designer to choose a topology in advance. Instead, it automatically evolves the topology to fit the complexity of the problem. NEAT combines the usual search for appropriate network weights with complexification of the network structure.

This approach is highly effective: NEAT outperforms other neuroevolution methods, e.g. on the benchmark double pole balancing task [10]. In addition, because NEAT starts with simple networks and expands the search space only when beneficial, it is able to find significantly more sophisticated controllers than fixed-topology evolution, as demonstrated in a robotic strategy-learning domain [11]. These properties make NEAT an attractive method for evolving neural networks.

In this section, the NEAT method is briefly reviewed; a more comprehensive description of the NEAT method is given in [10].

2.1.1 Genetic Encoding with Historical Markings

Evolving network structure requires a flexible genetic encoding. 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 number*, which allows NEAT to find corresponding genes during crossover.

Mutations in NEAT can change both connection weights and network structure. Connection weights mutate as in any neuroevolution system whereas structural changes are caused by two special mutation operators that allow complexity to increase. Figure 1 shows one of these operators, which adds a new hidden node to the network. Figure 2 shows the other operator, which adds a new link to the network. Through these mutations, genomes of varying sizes are created, sometimes with completely different connections specified at the same positions.

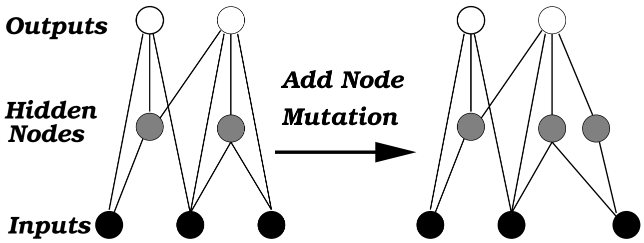

Figure 1: An example of NEAT's mutation operator for adding hidden nodes to a network. A new hidden node, shown on the right, is added to the network, splitting an existing link in two.

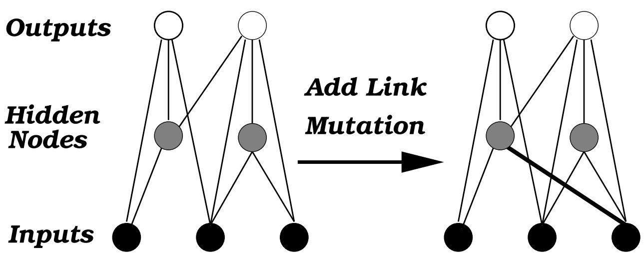

Figure 2: An example of NEAT's mutation operator for adding links to a network. The new link, shown with a thicker black line, is added between two existing nodes.

To perform crossover, the system must be able to tell which genes match up between *any* individuals in the population. For this purpose, NEAT keeps track of the historical origin of every gene. Whenever a new gene appears (through structural mutation), a *global innovation number* is incremented and assigned to that gene. The innovation numbers represent a chronology of every gene in the system. Whenever these genomes crossover, innovation numbers on inherited genes are preserved. Thus, the historical origin of every gene in the system is known throughout evolution.

Through innovation numbers, the system knows exactly which genes match up with which. Genes that do not match are either *disjoint* or *excess*, depending on whether they occur within or outside the range of the other parent's innovation numbers. When crossing over, the genes in both genomes with the same innovation numbers are lined up. Unmatched genes are inherited from the more fit parent, or if they are equally fit, from both parents randomly.

Historical markings allow NEAT to perform crossover without expensive topological analysis. Genomes of different organizations and sizes stay compatible throughout evolution, and the problem of matching different topologies [9] is essentially avoided.

2.1.2 Speciation

In many cases, adding new structure to a network initially reduces its fitness. However, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before they have to compete with other niches in the population.

Historical markings make it possible for the system to divide the population into species based on topological similarity. The distance δ between two network encodings is a simple linear combination of the number of excess (E) and disjoint (D) genes, as well as the average weight differences of matching genes ($\overline{W}$):

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

The coefficients c_1, c_2, and c_3 adjust the importance of the three factors, and the factor N, the number of genes in the larger genome, normalizes for genome size. Genomes are tested one at a time; if a genome's distance to a randomly chosen member of the species is less than δ_t, a compatibility threshold, it is placed into this species. Each genome is placed into the first species where this condition is satisfied, so that no genome is in more than one species.

The reproduction mechanism for NEAT is *explicit fitness sharing* [2], where organisms in the same species must share the fitness of their niche, preventing any one species from taking over the population.

2.1.3 Minimizing Dimensionality

Unlike other systems that evolve network topologies and weights [3, 14] NEAT begins with a uniform population of simple networks with no hidden nodes and inputs connected directly to outputs. New structure is introduced incrementally via the two special mutation operators described above. Since only those additions that improve performance are likely to be retained, it tends to discover small networks without superfluous structure. Starting minimally also helps NEAT learn more quickly. When networks in its population are small, it is optimizing over a lower-dimensional search space; it jumps to a larger space only when performance in the smaller one stagnates.

2.2 Feature Selective NeuroEvolution of Augmenting Topologies

NEAT's initial networks are small but not as small as possible. The structure of the initial networks, in which each input is connected directly to each output, reflects an assumption that all the available inputs are useful and should be connected to the rest of the network. In domains where the input set has been selected by a human expert, this assumption is reasonable. However, in many domains no such expert is available and the input set may contain many redundant or irrelevant features. In such cases, the initial connections used in regular NEAT can significantly harm performance by unnecessarily increasing the size of the search space.

FS-NEAT is an extension to NEAT that attempts to solve this problem by starting even more minimally: with networks having almost no links at all. As in regular NEAT, hidden nodes and links are added through mutation and only those additions that aid performance are likely to survive. Hence, FS-NEAT begins in even lower dimensional spaces than regular NEAT and feature selection occurs implicitly: only those links emerging from useful inputs will tend to survive.

Exactly how should we initialize the population in order to implement this idea? The most minimal initial topology possible would contain no hidden nodes or links at all. However, such networks would not generate any output. Obviously, spending a generation to evaluate a population of such networks would be wasteful. Therefore, for each network in the initial population, FS-NEAT randomly selects an input and an output and adds a link connecting them. Figure 3 compares the initial network topologies of regular NEAT and FS-NEAT. After the initial population is generated, FS-NEAT behaves exactly like regular NEAT[10].

In most tasks, FS-NEAT's initial networks will lack the structure necessary to perform well. However, some will likely connect a relevant input to an output in a useful way and hence outperform their peers. Such early distinctions provide an initial gradient to the evolutionary search. Complexification then drives that search towards networks that use the most appropriate inputs, topology and weights.

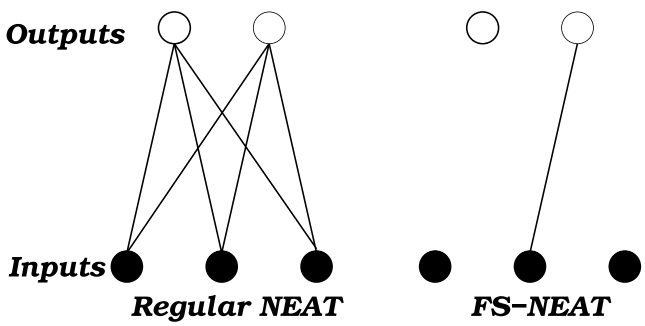

Figure 3: Examples of initial network topologies for both regular NEAT and FS-NEAT. In regular NEAT, networks in the initial population have all inputs connected directly to all outputs. In FS-NEAT, those networks have one link connecting a randomly selected input and output.

Since FS-NEAT incorporates the feature selection problem into the learning task itself, it avoids the need for expensive meta-learners used by wrappers. In addition, since it does not rely on labeled data like filters do, it can be applied to reinforcement learning problems. The next section describes one such application.

3. EXPERIMENTS

The experiments presented in this paper were conducted in the Robot Auto Racing Simulator (RARS) [12], a Java-based program that uses a two-dimensional model to simulate cars racing around a track. The simulation is quite realistic and takes into account effects such as skidding and traction. In addition, RARS models the noise that occurs in real-world effectors. For example, the coefficient of friction is stochastic such that the effect of trying to accelerate is not entirely predictable. The goal in this domain is to develop a controller that can race an automobile around the track as quickly as possible without damaging it.

The RARS simulator offers a plethora of raw data about the car's immediate environment. This data was consolidated into a rangefinder system, shown in Figure 4, that projects rays at different angles relative to the car's current heading. These rays measure the distance from the car to the edge of the road, which allows the agent to estimate its position in the road and perceive upcoming curves. This sensor system creates a very typical feature selection problem. How many rangefinders does the controller need in order to drive the car most effectively? If too few are included, the networks NEAT evolves will not have enough information to master the task. If too many are included, NEAT will be forced to search in an unnecessarily large search space, which may substantially reduce its performance.

To test the ability of FS-NEAT to automatically address this problem, the networks are provided with a set of 80 rangefinders (evenly distributed across the 180 degree range in front of the car), which we expect to be more than necessary. In addition, another 80 irrelevant inputs are included, each of which supplies random numbers drawn uniformly from the range $[0, 1]$. Finally, there is one input specifying the vehicle's current velocity and one bias unit, for a total of 162 inputs. If FS-NEAT can automatically discover a useful subset of these inputs, it should outperform regular NEAT, which is forced to use all 162.

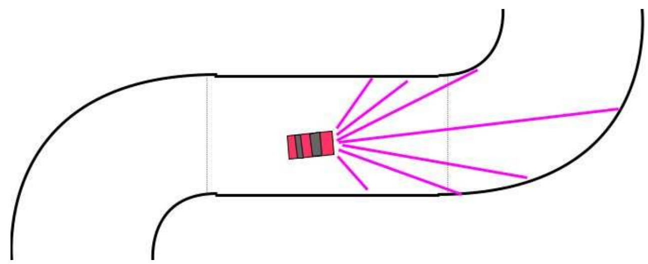

Figure 4: The range finder sensor system in RARS. A set of rays (seven in this case) are projected at different angles to allow the agent to estimate its position in the road and perceive upcoming curves.

In addition to these inputs, the networks have two outputs: one specifying the agent's desired speed and the other specifying the agent's desired heading. In our experiments, a trial consists of 2000 timesteps on the track called "clkwis," shown in Figure 5. This track was selected because it is small enough to allow efficient evaluations but still captures a wide range of driving challenges (i.e. straight sections, turns, and an S-shaped curve). During each timestep, input from the environment is fed into the network controlling the car. The network is then activated once and the values of the outputs are used to adjust the vehicle's heading and speed. At the end of each trial a score is computed as $S = 2d - b$, where d is the distance traveled and b is a damage penalty computed internally by RARS as a function of the time the vehicle spends off the track. Since the simulation is noisy, each fitness evaluation in NEAT consists of 10 trials; the agent's fitness is the average of the scores received in these trials. See the appendix for details about the NEAT parameter settings used in these experiments.

Using this setup, we performed experiments comparing regular NEAT to FS-NEAT. For each method, we conducted 10 runs, each of which ran for 200 generations. The results are summarized in Figure 6. Each line in the graph represents the score received by the best network from each generation, averaged over all 10 runs. The graph demonstrates that when some of the available inputs are redundant or irrelevant, FS-NEAT can learn better networks and learn

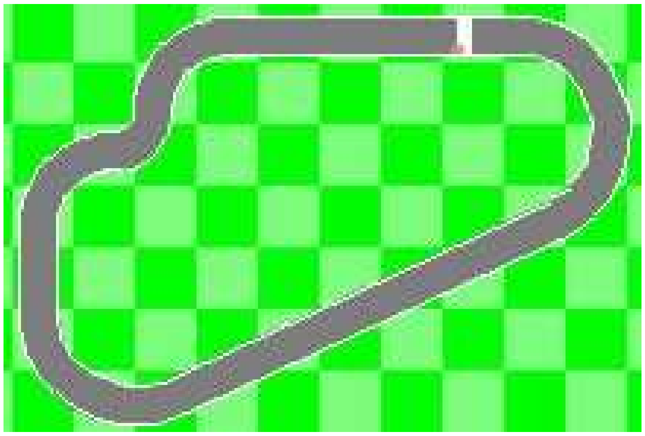

Figure 5: The "clkwis" track used in the FS-NEAT experiments. It captures a wide range of driving challenges (i.e. straight sections, turns, and an S-shaped curve).

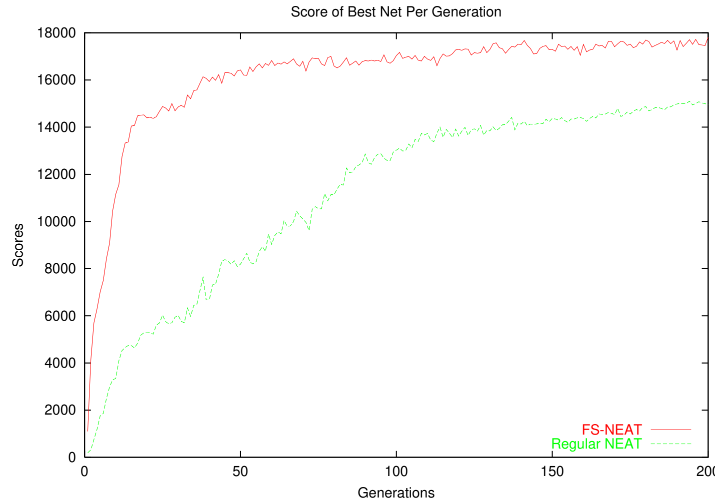

Figure 6: A comparison of the performance of regular NEAT and FS-NEAT in the RARS domain with 162 available inputs, 80 of which are irrelevant to the task. Each line in the graph represents the score received by the best network from each generation, averaged over all 10 runs. By learning appropriate feature sets, FS-NEAT learns significantly better networks and learns them faster than regular NEAT.

them faster than regular NEAT. In this graph and all those presented below, a Student's t-test verified, with 95% confidence, the statistical significance of the difference between FS-NEAT and regular NEAT.

Figure 7 shows, for the same experiments, how many inputs have at least one connection emerging from them in the best network of each generation. Regular NEAT always uses all 162 inputs but FS-NEAT finds better networks that use only a small fraction of them. In fact, when FS-NEAT's performance begins to plateau around generation 65, its performance is already 17.5% better than regular NEAT ever achieves, at which point its best network has on average only 10% as many connected inputs. FS-NEAT's performance continues to creep up slowly after generation 65, improving another 4.6% by generation 200, at which point its best network has on average 22.9% as many inputs as regular NEAT.

Figure 8 shows, for the same experiments, the size of the best network from each generation, where size is simply the total number of nodes (only connected inputs are counted) plus the total number of links. This graph demonstrates that FS-NEAT evolves substantially smaller networks than regular NEAT does. When FS-NEAT's performance begins to plateau around generation 65, its best network is on average only 9.7% as large as regular NEAT's. When the runs complete at generation 200, FS-NEAT's best network is on average only 18.5% as large as regular NEAT's.

In these experiments, FS-NEAT found high performing networks that use only 16 inputs, which implies that the feature set we supplied to the learners, with 80 rangefinders, was much larger than needed. How would the performance of FS-NEAT relative to regular NEAT change if the initial feature set were closer to ideal? How many redundant and irrelevant features must be present before FS-NEAT provides a significant advantage? Does FS-NEAT's performance improvement continue to increase as the feature set gets larger? To address these questions, we conducted several additional experiments with feature sets of different sizes. These experiments use the setup described above but, instead of 80 rangefinders, they include 5, 20, 40, or 160 rangefinders. In each case, the rangefinders are matched with an equal number of irrelevant inputs. Adding the velocity and bias inputs yields initial feature sets of size 12, 42, 82, and 322. For each size and for each method, we conducted 10 runs, each of which ran for 200 generations.

Figure 9 summarizes the results of these experiments by showing, for each method and feature set size, the performance of the best network in the entire run, averaged over all ten runs. Even when the initial feature set contains only 12 inputs, FS-NEAT still performs better. As the size of the feature set grows, the performance of regular NEAT deteriorates. By contrast, the performance of FS-NEAT remains nearly constant even as the feature selection task it faces becomes ever more difficult.

Figure 10 compares the number of connected inputs in the best network in the entire run, averaged over all ten runs. Regular NEAT always uses all available inputs while FS-NEAT learns to use much smaller subsets. Even as the size of the feature set grows, the number of inputs used by FS-NEAT's best networks stays nearly constant. Similarly, Figure 11 compares the sizes of these same networks. The size of regular NEAT's best networks increases linearly with respect to the number of available features, whereas FS-NEAT's best networks stay nearly constant in size. Therefore, FS-NEAT consistently uses features sets with many fewer extraneous inputs than regular NEAT and, in so doing, finds better solutions faster.

4. DISCUSSION

The empirical results presented in this paper demonstrate that when some of the available inputs are redundant or irrelevant, FS-NEAT can learn better networks and learn them faster than regular NEAT. In addition, the networks it learns are smaller and use fewer inputs. These results are consistent across feature sets of different sizes.

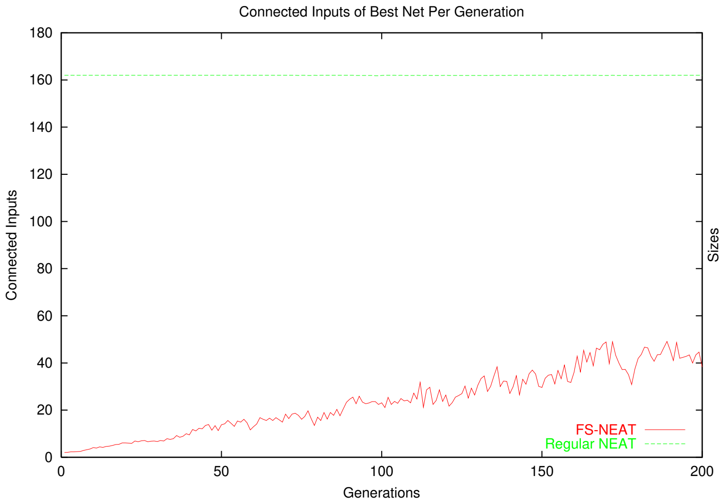

Figure 7: A comparison of the number of inputs used by regular NEAT and FS-NEAT in the RARS domain with 162 available inputs. Each line in the graph represents the number of inputs with at least one connection emerging from them in the best network of each generation. Regular NEAT always uses all 162 inputs but FS-NEAT evolves better networks that uses significantly fewer of them.

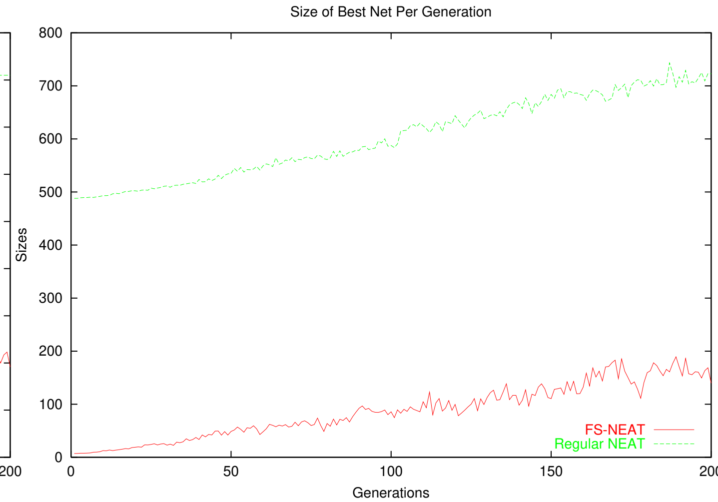

Figure 8: A comparison of the size of the networks evolved by regular NEAT and FS-NEAT in the RARS domain with 162 available inputs. Each line in the graph represents the number of nodes (only connected inputs are counted) plus the number of links in the best network of each generation. FS-NEAT evolves significantly smaller networks than regular NEAT does.

One interesting question raised by these results is why the size and number of inputs used by FS-NEAT do not plateau. For example, Figure 6 shows that performance improvements mostly level off by generation 65. However, Figures 7 and 8 show that the size and number of inputs used by FS-NEAT's best networks continue to grow linearly through generation 200. Shouldn't we expect them to plateau also once the "right" size has been found? Counterintuitively, the answer is no. The goal of both NEAT and FS-NEAT is to determine the right complexity to solve a given task. Hence, when performance at a certain complexity plateaus, these algorithms proceed to explore at higher complexities. In these experiments, that exploration pays few dividends after generation 65.

Nonetheless, even given such exploration, we would still expect to see size plateau if there were a strong selective pressure against larger networks since none of these networks would likely become generation champions. The fact that they do implies that FS-NEAT is not completely intolerant of redundant and irrelevant inputs. This behavior makes sense because the presence of such inputs may not be harmful if, for example, NEAT can learn to set the weights emerging from them close to zero. In this respect, FS-NEAT behaves exactly as we would wish: it selects against large networks only when their size presents a significant disadvantage to the learner.

In evolutionary search, it is critical that the fitness of the initial population have some variance: unless some individuals are more promising than others, progress is unlikely. This issue is of particular concern in FS-NEAT since its initial population consists of degenerate networks that are almost completely disconnected. While the experiments presented in this paper verify that FS-NEAT consistently finds an initial gradient for learning, those experiments tested only one population size: 100. We wondered if the relative performance of FS-NEAT would deteriorate for smaller populations since the probability of finding an initial promising network would decrease. However, this problem does not occur in the RARS domain. In fact, preliminary experiments with different population sizes indicate that both regular NEAT and FS-NEAT perform robustly with populations as small as 25 and that FS-NEAT retains its substantial advantage over regular NEAT. Hence, at least in RARS, FS-NEAT's smaller initial networks seem *more* likely to point evolution in the right direction.

The most revealing test of FS-NEAT's robustness is how its performance changes when the size of the initial feature set increases. As this set gets larger, feature selection becomes more important, as confirmed by the decline of regular NEAT's performance in Figure 9. FS-NEAT's performance, by contrast, does not decline at all. Most strikingly, the size and number of inputs used by FS-NEAT's best networks remains approximately constant across different feature set sizes, whereas regular NEAT's networks grow ever larger. Together, these results suggest that the efficacy of FS-NEAT scales well to large feature selection problems.

5. FUTURE WORK

An important question that this paper does not address is how the effects of irrelevant inputs might differ from those of redundant inputs. The experiments presented in this paper study these two types of inputs together to ensure that FS-NEAT faces the challenges of both. However, studying them separately would help tease apart their differences. Preliminary experiments suggest that, in the RARS domain, irrelevant features are more damaging than redundant ones.

In addition, this paper's experiments always add nodes and links to the networks at a rate that remains constant throughout each run. However, in most problems, this be-

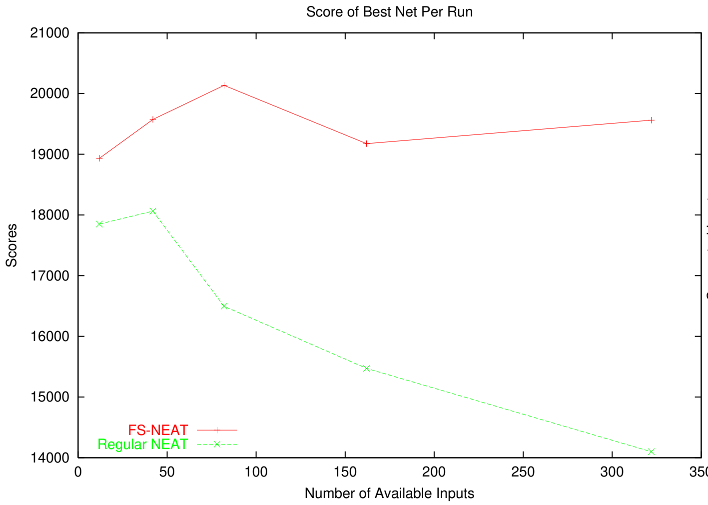

Figure 9: A comparison of the performance of regular NEAT and FS-NEAT across feature sets of different sizes. Each line in the graph represents the score received by the best network in the entire run, averaged over all 10 runs. The performance of regular NEAT gets significantly worse as the feature set gets larger whereas the performance of FS-NEAT stays nearly constant.

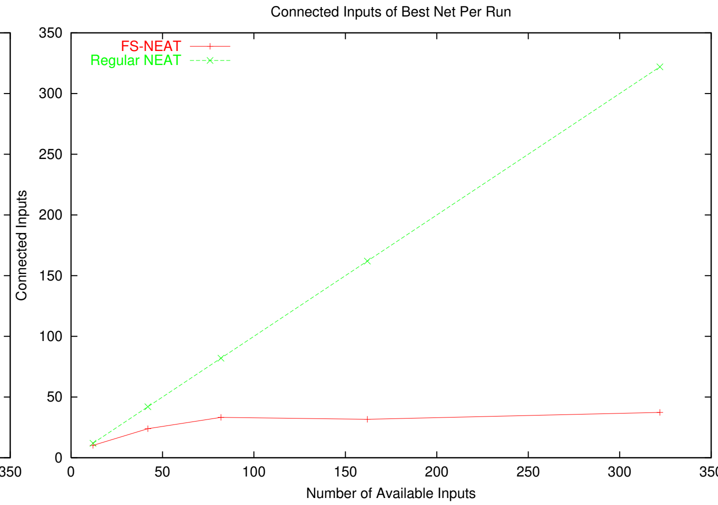

Figure 10: A comparison of the number of connected inputs in regular NEAT and FS-NEAT across feature sets of different sizes. Each line in the graph represents the number of inputs with at least one connection in the best network of the entire run, averaged over all 10 runs. Regular NEAT always uses all available inputs while FS-NEAT learns to use significantly smaller subsets.

havior is probably not optimal. Intuitively, such mutations are more likely to be helpful early in evolution, when most networks lack the topology and input connections necessary to solve the task. Later in the search, when good structures have already been found and need only fine-tuning, structural mutations are less likely to improve performance. Hence, FS-NEAT could be considerably improved by placing these mutations rates on an annealing schedule such that they decay over time. Of course, regular NEAT could also benefit from such a modification. However, since its initial networks already have some connections, the change in optimal mutation rates over time is probably less pronounced.

Furthermore, the mutation operators that add new structure are not balanced in any way by mutations that *remove* structure. Despite NEAT's focus on finding small networks, this may result in some network bloating unless there is strong selective pressure against large networks. Such bloating could potentially be reduced by adding new mutation operators that remove nodes and links from the networks. Then, networks would increase in size only when there is significant selective pressure favoring larger networks. This could be extremely useful in domains where minimizing network size is important even when it does not improve performance. For example, if the networks are to be implemented in hardware, minimizing size may be a critical factor. In addition, if we are preparing controllers for real robots, using the fewest possible inputs can save money on physical sensors.

In future work, we plan to implement and test the modifications mentioned above. We also intend to apply FS-NEAT to additional challenging reinforcement learning problems to further establish the scope of its efficacy.

6. CONCLUSION

The primary contribution of this paper is FS-NEAT, a novel extension to NEAT which enables it to perform automatic feature selection. Unlike other feature selection methods, FS-NEAT does not rely on expensive meta-learning or labeled data. Empirical results in a challenging autonomous car racing domain demonstrate that when some of the available inputs are redundant or irrelevant, FS-NEAT can learn better networks and learn them faster than regular NEAT. In addition, the networks it discovers are smaller and use fewer inputs. Experiments on initial feature sets of different sizes confirm that FS-NEAT's performance is robust even as the feature selection problem it faces grows increasingly difficult. By reducing the need for human experts to manually select appropriate features, we hope that FS-NEAT will increase the set of tasks to which the NEAT method can be practically applied.

Acknowledgments

This research is supported in part by a grant from Toyota USA, NSF CAREER award IIS-0237699, and an IBM faculty award.

APPENDIX

A. NEAT AND FS-NEAT SYSTEM PARAMETERS

The population size was 100. Each fitness evaluation required 10 trials, yielding 1000 trials per generation. The coefficients for measuring compatibility were $c_1 = 1.0$, $c_2 = 1.0$, and $c_3 = 3.0$. The compatibility distance was $\delta_t = 3.0$. The champion of each species with more than five networks was copied into the next generation unchanged. The inter-

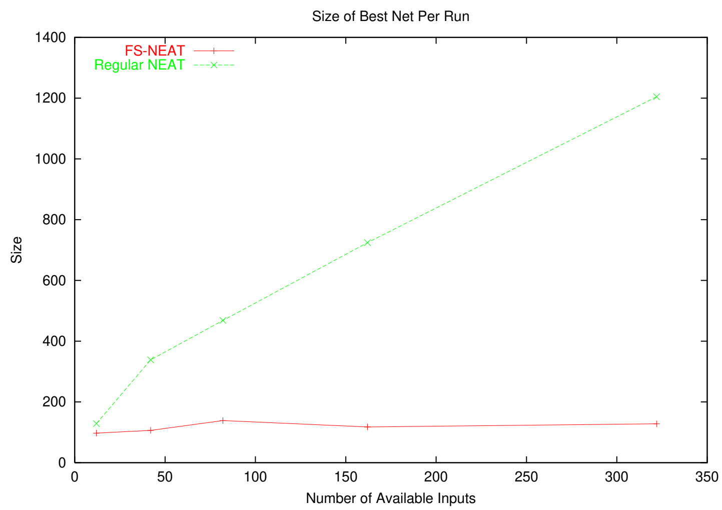

Figure 11: A comparison of network size in regular NEAT and FS-NEAT across feature sets of different sizes. Each line in the graph represents the size of the best network of the entire run, averaged over all 10 runs. Regular NEAT's best networks increase in size significantly as the number of available features grows, whereas FS-NEAT's best networks stay nearly constant in size.

species mating rate was 0.05. The probability of adding a new node was 0.02 and the probability of a new link mutation was 0.1.

B. REFERENCES

[1] B. V. Bonnlander and A. S. Weigend. Selecting input variables using mutual information and nonparametric density estimation. In *Proceedings of the 1994 International Symposium on Artificial Neural Networks (ISANN'94)*, pages 42–50, Tainan, Taiwan, 1994.

[2] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154, 1987.

[3] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 81–89. MIT Press, 1996.

[4] P. R. Harvey, D. M. Booth, and J. F. Boyce. Evolving the mapping between input neurons and multi-source imagery. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 1878–1883, 2002.

[5] K. Kira and L. Rendell. A practical approach to feature selection. In *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, Massachusetts, 1992. Morgan Kaufmann.

[6] P. Langley. Selection of relevant features in machine learning. In *Proceedings of AAAI Fall Symposium on Relevance*, 1994.

[7] P. M. Narendra and K. Fukunaga. A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, 26:917–922, 1977.

[8] J. Novovivova, P. Pudil, and J. Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125, 1994.

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

[10] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[11] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21, 2004. In press.

[12] M. E. Timin. The robot auto racing simulator, 1995. `http://rars.sourceforge.net`.

[13] S. Whiteson, K. O. Stanley, and R. Miikkulainen. Automatic feature selection in neuroevolution. In *GECCO 2004: Proceedings of the Genetic and Evolutionary Computation Conference Workshop on Self-Organization*, July 2004.

[14] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.