Sustaining Diversity using Behavioral Information Distance

Faustino J. Gomez
IDSIA, Galleria 2
6928 Manno-Lugano
Switzerland
tino@idsia.ch

ABSTRACT

Conventional similarity metrics used to sustain diversity in evolving populations are not well suited to sequential decision tasks. Genotypes and phenotypic structure are poor predictors of how solutions will actually behave in the environment. In this paper, we propose measuring similarity directly on the behavioral trajectories of evolving candidate policies using a universal similarity measure based on algorithmic information theory: normalized compression distance (NCD). NCD is compared to four other similarity measures in both genotype and phenotype space on the POMDP Tartarus problem, and shown to produce the most fit, general, and complex solutions.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*

General Terms

Algorithms

1. INTRODUCTION

Most challenging real-world problems can cause evolutionary algorithms to be lured into local minima by candidate solutions that afford better than average fitness, but do not exhibit the kind of true complexity that is ultimately required to solve the task. These sub-optimal solutions can quickly overtake the population, depleting diversity and, with it, the possibility of exploring other more promising regions of the search space. Reducing selective pressure will slow convergence, but may also prevent the right path from being taken. Some mechanism is needed to sustain diversity, but variation is not enough: ideally the population should maintain solutions that are not only fit but different from one another so that evolution can keep its options open for as long as possible.

Many algorithms have been developed to try to sustain diversity or at least postpone convergence, e.g., crowding [3], deterministic crowding [8], fitness sharing [3], implicit fitness sharing [10], restricted mating [4]. With the exception of recent work by Lehman and Stanley [5] these algorithms are always implemented to measure diversity by looking at the distance between individuals using standard metrics (e.g. Euclidean and Hamming distance) in either genotype or phenotype space. This is a natural approach for *static* tasks such as function optimization where the fitness of the candidate is represented directly by its parameters. However, for sequential decision, or *dynamic* tasks this notion of similarity is deceptive as phenotypes that are similar structurally, may behave very differently when evaluated in the environment, not only due to stochasticity, but also to discontinuities in the genotype-phenotype map.

This paper proposes using *behavioral distance* to more meaningfully compare individuals using a universal measure based on algorithmic information theory, called the normalized compression distance (NCD; [7]). This quasi-metric has the powerful property that it can measure similarity between sequences (e.g. state-action trajectories) of potentially different lengths by exploiting only their algorithmic regularities; no domain-specific features need to be specified.

The next section discusses the idea of measuring similarity in different evolution spaces. Section 3 explains normalized compression distance in more detail. In section 4, we present experiments comparing NCD to other similarity measures on the Tartarus (block packer) problem using a simple crowding based algorithm. The last two sections discuss the overall results, suggest directions to further research, and present our conclusions.

2. MEASURING SIMILARITY

In order to identify novel solutions during a search process there must be some way to measure how similar a new candidate is to the solution points already visited. In evolutionary algorithms, this measurement can be taken in the genotype space where the solutions are encoded as strings, or in the phenotype space where they are manifest.

For some problem classes and representations, applying a particular metric in one space is equivalent to applying it in the other: the genotype$\rightarrow$phenotype mapping, G, preserves the relative distance between objects in each. When this is not the case, it may be more informative to measure the similarity between phenotypes (figure 1), after all, what we are truly interested in is the similarity of solutions, not their encodings.

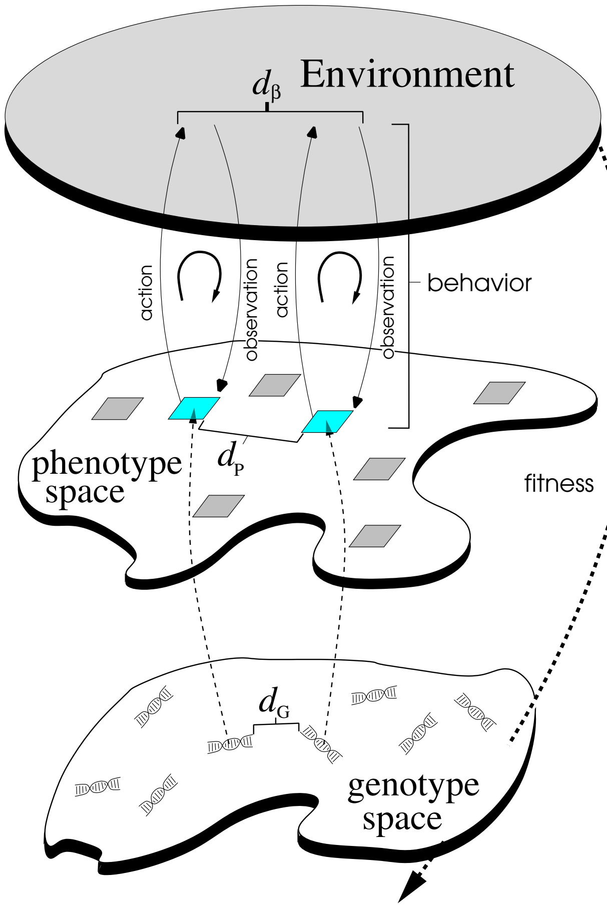

Figure 1: Genotype-Phenotype map. The similarity between evolving candidate solutions can be computed at different levels. In sequential decision tasks, comparing individuals structurally in phenotype space, d_P, may not give a reliable indication with respect to phenotype behavior (shown as the cycling of actions and observations of the two highlighted phenotypes). A more informative measure is to compare the behaviors directly, d_β.

In many cases, comparing phenotypes is usually not more complicated than comparing genotypes. This is because many optimization problems are static: n-dimensional genotype vectors are mapped to m-dimensional phenotypes (parameter vectors) which are then evaluated *once* on a cost function. As both are of fixed dimension, the same metrics (e.g. Euclidean distance) used for genotypes can be used for phenotypes.

For sequential decision tasks (e.g. reinforcement learning) the situation is different: G maps to some form of policy, π, that implements a probability distribution over a set of possible actions conditioned on the observation from the environment. More generally, the choice of action at time t can be conditioned on the entire *history* of previous observations, o, and actions, a:

$$a_t \leftarrow \pi(o_{t-1}, a_{t-1}, \ldots, o_0, a_0). \tag{1}$$

For example, such a policy could be implemented by a recurrent neural network. Comparing phenotypes of this form by only looking at structural similarity, e.g. network topology, and weight vectors, can be deceptive as policies that are structurally similar with respect to the chosen metric may be very different in terms of behavior when they interact with the environment. We define the *behavior*, β_x, of individual x to be a set of one or more histories of the form in equation 1 resulting from one or more evaluations in the environment. A behavior is therefore an approximation of the *true* behavior of the individual that can only be sampled by interaction with the environment.

In order to encourage useful *behavioral diversity*, similarity should be computed directly at the level of phenotype behavior. However, this is more complicated because behaviors can be very high dimensional, and two behaviors can have different durations.

The next section describes a powerful similarity measure that can be applied to sequences of arbitrary length without any knowledge domain-specific features.

3. NORMALIZED COMPRESSION DISTANCE

One universal way to measure the similarity of two objects is to use the *normalized information distance* [7]:

$$\mathrm{NID} = \frac{\max(K(x|y), K(y|x))}{\max(K(x), K(y))} \tag{2}$$

where $K(x|y)$, the *Kolmogorov complexity* of x given y, is the length of the shortest binary program, running on a universal Turing machine, that will output x given the input y, and $K(x)$ is the *Kolmogorov complexity* of x given no input. $K(x)$ provides a theoretical lower bound on the compressed size of x. The beauty of NID is that unlike all other similarity measures, which are feature-based, it does not rely on the *a priori* identification of salient domain features: NID automatically measures the similarity between two objects based on the dominant discriminating feature.

Unfortunately, $K(x)$ is not Turing computable. In order to apply NID, Kolmogorov complexity can be approximated by using a real-world compressor to arrive at the *normalized compression distance* (NCD):

$$\mathrm{NCD}(x,y) = \frac{C(xy) - \min(C(x), C(y))}{\max(C(x), C(y))}, \tag{3}$$

where $C(x)$ is the compressed length of sequence x, and xy is the concatenation of x and y. If we assume for the moment that $C(x) < C(y)$, then $\mathrm{NCD}(x,y)$ can be understood as the improvement in compressing y derived from using x as a compressed data base. If x has little regularity in common with y, then $C(xy) - C(x) \approx C(y)$ and NCD ≈ 1. If $x = y$, then a good compressor should be able to detect the xy contains two copies of the same sequence, so that $C(xy) \approx C(x) = C(y)$, and NCD ≈ 0.

Although it is not to possible compute how close the NCD is from the ideal NID for any two objects, in practice NCD has yielded impressive and surprising results. In Cilibrasi et al. [1] it was used to correctly classify disparate file types ranging from compiled java code to Jimi Hendrix songs in midi, all without any domain knowledge! And in Li et al. [6] it was used to construct phylogenetic trees from mitochondrial genomes that corresponded exactly to the existing maximum likelihood trees which used extensive domain knowledge.

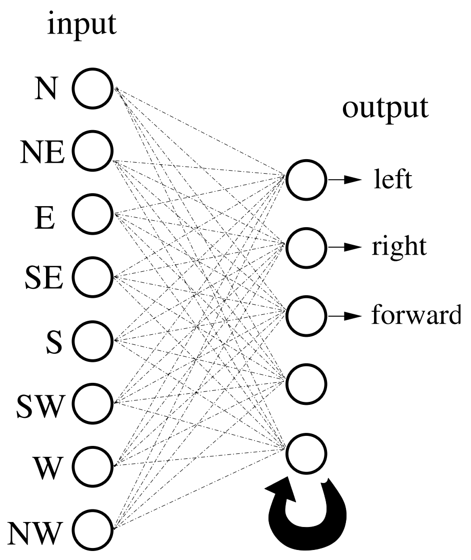

Figure 2: Bulldozer controller. The bulldozer is controlled by a fully recurrent neural network with five units. At each time step the network receives the state of the eight surrounding grid cells through its input layer. The input is propagated to the output layer, along with the activation of the output layer from the previous time step (i.e. the recurrent connections denoted by the large black arrow), and the action corresponding to action unit (left, right, forward) with the highest activation is chosen for next time step.

4. EXPERIMENTS

In the following experiments we compare the performance of NCD to four other similarity measures on the Tartarus POMDP problem.

4.1 The Tartarus Problem

Figure 3 shows the Tartarus problem [11], used in the experiments. The task consists of a 6×6 grid-world surrounded by walls within which six blocks and an agent, or *bulldozer*, are place away from the walls. The bulldozer is allotted 80 moves, during each of which it takes one of three actions: turn left, turn right, or go forward. If the agent goes forward it can push a block in its path as long as the block is not against a wall or another block. The objective is for the bulldozer to push the blocks against the walls. For each block that finishes against a wall, the agent receives one point, corners are worth two points; for a maximum score of 10.

Although the grid-world is quite small, the task is challenging because the bulldozer can only see the adjacent grid cells, so that many observations that require different actions look the same, i.e. perceptual aliasing. In order to perform the task successfully, the bulldozer must remember previous observations such that it can compute its location relative to the walls and record the locations of observed blocks for the purpose quickly acquiring them later. In short, the agent is quite blind, which means that evolutionary search can quickly discover simple, mechanical behaviors that produce better than random performance but do not exhibit the underlying memory capability to perform well on the task.

Algorithm 1: GenericCrowding(p, k, n, m)

```
1  Initialize the population P with $p$ individuals
2  and evaluate them
3  for $i$=1 to $k$ do
4      parentA ← TournamentSelect($n$)
5      parentB ← TournamentSelect($n$)
6      childA ← Crossover(parentA, parentB)
       childB ↙
7      Mutate(childA) ;        /* evaluate and mutate */
8      Evaluate(childA) ;        /* the two offspring */
9      Mutate(childB)
10     Evaluate(childB)
11     $l_A$ ← CrowdingSelect($P$, $m$, childA)
12     $l_B$ ← CrowdingSelect($P$, $m$, childB)
13     $P[l_A]$ ← childA ;        /* replace losers with */
14     $P[l_B]$ ← childB ;                   /* offspring */
   end
```

Function CrowdingSelect(P, n, x)

```
1  for $i$=1 to $n$ do
2      $j$ = rand($|P|$);   /* choose random individual */
3      distance ← $d(x, P[j])$;       /* compute distance */
4      if distance < min then
5          min ← distance
6          loser = $j$
7      end
8      return loser;      /* return the most similar */
9  end
```

4.2 Setup

Algorithm 1 presents pseudocode for the simple steady state GA used in the experiments. The algorithm takes four parameters: p, the size of the population, k, the number of iterations, n, the tournament size for selection, and m, the *crowding factor* [2] used for replacement. After the population of p individuals is initialized and evaluated, each iteration begins by selecting two parents based on fitness using tournament selection, and recombining them using 1-point crossover to produce two children. After mutating and evaluating the children, the CrowdingSelect function chooses, for each child, by tournament selection (tournament size m), the individual (i.e. the *loser*) that is most similar to the child according to the similarity measure $d(\cdot, \cdot)$. Each child replaces its corresponding loser, and the cycle repeats.

The Generic Crowding algorithm is not intended to advance the study of niching methods, rather its purpose is to provide a minimal framework within which to analyze the effect of various similarity measures on diversity and performance. This simple algorithm allows us to control the selective pressure that drives the population to convergence and at the same time an opposing, "replacement" pressure via the crowding factor that seeks to delay convergence–the efficacy of which depends the particular similarity measure used.

The five following similarity measures were compared:

Fitness:

$$d_{\text{fit}}(x, y) = |f(x) - f(y)| \tag{4}$$

where $f(x)$ is the fitness of genotype x. This is the simplest of the similarity measures used.

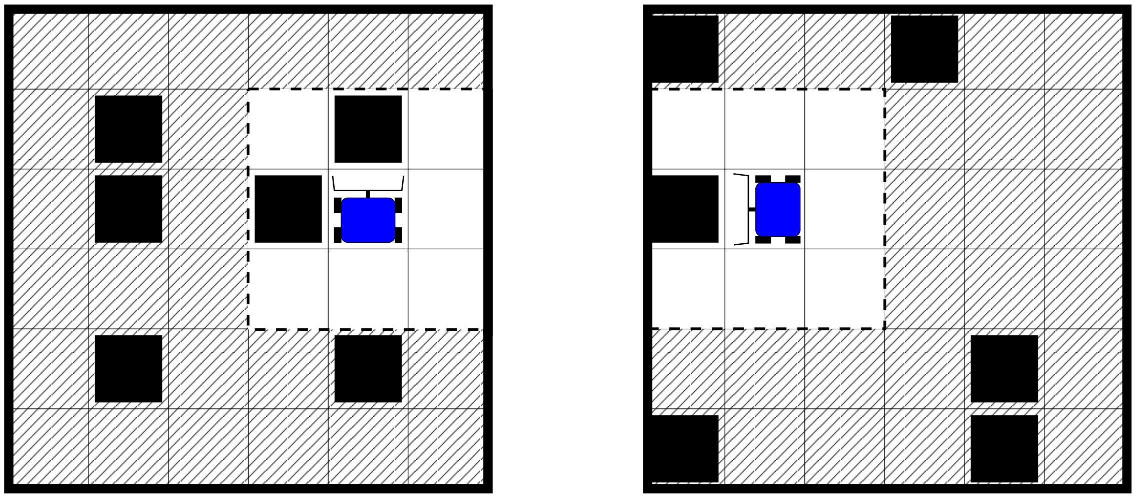

Figure 3: The Tartarus Problem. The Tartarus board on the left shows a possible initial state with the six blocks and the bulldozer placed at random squares away from the walls; the orientation the bulldozer is also random. The bulldozer must select an action (either turn left, turn right, or go forward) at each time-step based on the situation within its visual field (shown in white), and its internal state (memory). The bulldozer can only move forward if its path is unobstructed or the block in its way has no block behind it, otherwise it will remain its current position. On the right is a possible final state after the allotted 80 moves. The score for this configuration is 7: two blocks receive a score of two for being in the corner, plus one point for each of the three other blocks that are against a wall.

Euclidean distance:

$$d_{\text{euc}}(x, y) = \|x - y\|_2 \tag{5}$$

This is probably the most pervasive measure of similarity found in the literature. It is applied here in genotype space, but given the direct, 1-to-1 genotype-phenotype mapping used below for the neural network policies, applying it in phenotype space would have the same effect.

Hamming distance:

$$d_{\text{ham}}(x, y) = \sum_{i=0}^{T} \delta(\beta_x[i], \beta_y[i]) \tag{6}$$

where δ is the Kronecker delta, the action histories in each β are concatenated together, and $\beta[i]$ is the i-th entry in the concatenated sequence, so that the distance between two behaviors is just the number of positions i at which the two individuals chose different actions.

Relative Entropy:

$$\begin{aligned} d_{\text{RH}}(x, y) &= D(\beta_x \,\|\, \beta_y) + D(\beta_y \,\|\, \beta_x) &(7)\\ &= \sum_{a \in A} (p_x(a) - p_y(a)) \log \frac{p_x(a)}{p_y(a)} &(8) \end{aligned}$$

where $D(y\|x)$ is the Kullback-Leibler divergence of x from y, i.e. the expected number of bits (for log base 2) per symbol required to encode x using the optimal code for y. As $D(\cdot, \cdot)$ is not symmetric, d_{RH} combines the divergence in both directions so that $d_{RH}(x, y) = d_{RH}(y, x)$. Each $p_x(a)$ corresponds to the probability of symbol a in β_x, $(P(\beta_x[i] = a))$, and $p_y(a)$ for each symbol in β_y.

Normalized Compression Distance (NCD):

$$d_{\text{NCD}}(\beta_x, \beta_y) = \frac{C(\beta_x\beta_y) - \min(C(\beta_x), C(\beta_y))}{\max(C(\beta_x), C(\beta_y))} \tag{9}$$

where $C(\beta)$ is the compressed length of behavior β (see section 3).

Note that the last three distance measures ($d_{\text{ham}}, d_{\text{RH}}$ and d_{NCD}) are applied in behavior space. Hereafter, each distance will be referred to by the following convention: d_{fit}: Fitness, d_{euc}: Euclidean, d_{ham}: Hamming, d_{RH}: Entropy, d_{NCD}: NCD.

A set of experiments was run for each each similarity measure, each set consisting of four groups of 50 experiments. All group within set used a tournament size, $n = 10$, but differed in their crowding factor $m = 2, 5, 10, 15$ parameters, for a total of $5 \times 4 \times 50 = 1000$ experiments. Other values of n were also tried, but $n = 10$ produced the best performance for all methods, so only those results are presented.

In addition, two more groups of 50 simulations were run to provide a baseline:

Random : replace a random individual in the population. This is equivalent to setting the crowding factor to 1.

Worst : replace the least fit individual in the population. The conventional practice in many GAs.

All simulations were run for $k = 400,000$ iterations using a population of $p = 100$ bulldozer controllers represented by fully recurrent neural networks with five sigmoidal units (figure 2). Three of the five units served as the outputs, one for each of the actions. The network genotypes were repre-

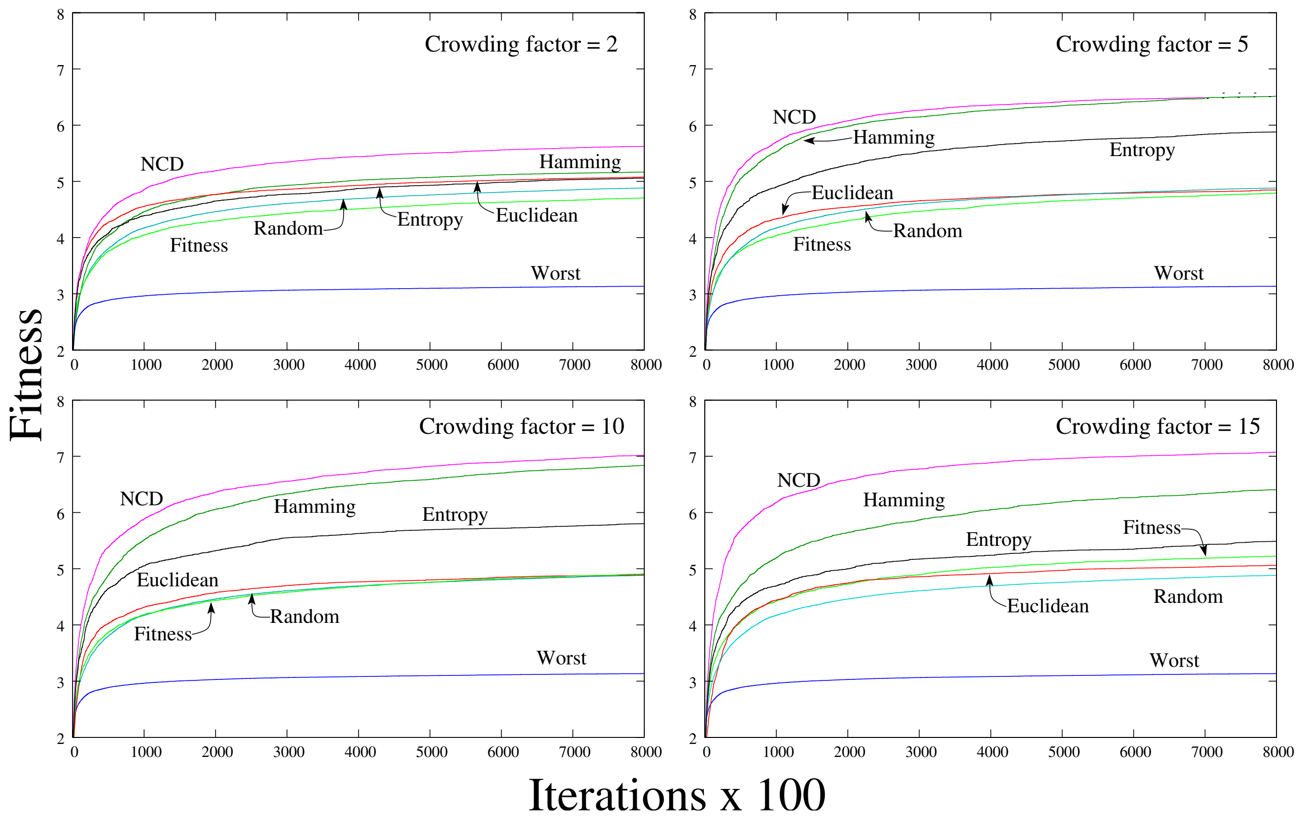

Figure 4: Comparison of similarity measures for Generic Crowding on the Tartarus problem. Each graph shows the performance in terms of average fitness for each of the replacement schemes described in section 4.2 using the Generic Crowding algorithm, for a particular value of m, the crowding factor. The two schemes based on behavioral similarity (HAMMING and NCD) show a clear advantage as the crowding factor increases. NCD dominates the other methods, and benefits more from high replacement pressure than HAMMING, which declines when m reaches 15. Each curve is the average of 50 simulations.

sented by real-valued vectors encoding the inputs and recurrent weights of each of the units using initial values chosen at random from [-10,10]. The mutation operator changed the value of each weight with probability α to a new value chosen at random from the same range as the initial weights. For all experiments $\alpha = 0.3$.

Each controller was evaluated on 100 random board configurations. To reduce evaluation noise, the set of 100 initial boards was chosen at random for each simulation, but remained fixed for the duration of the simulation. That is, in a given run all networks were evaluated on the same 100 initial boards.

For the three behavior distances, HAMMING, ENTROPY, and NCD, the behaviors consisted of sequences of 80 {Left=1, Right=2, Forward=3} actions executed in each of the 100 trials concatenated together, for a total of 8,000 actions. For this task it was not necessary to include observations in the behaviors because the environment is deterministic and the initial states were fixed for all individuals in a single run, so that each sequence of actions only has one corresponding sequence of observations. For NCD, `bzip2` [9] was used as the compressor, which uses the Burrows-Wheeler transform, and Huffman coding ($O(n)$). Note that the compressor has no explicit information about the segmented, trial-by-trial structure of the behaviors.

4.3 Results

Figure 4 summarizes the results of the experiments for the seven different approaches, with one graph for each of the studied crowding factor, m, values. For $m = 2$ there is very little replacement pressure so that all of the schemes, except WORST, perform close to the RANDOM baseline. However, even with only this slight bias toward similarity in the replacement, the simulations using NCD perform significantly better than the others, whereas the other behavior-based measures, ENTROPY and HAMMING, were not statistically different from the RANDOM replacement scheme.

At $5 \leq m \leq 10$, the performance of the behavioral distance methods clearly diverges from that of the others, and their absolute performance increases. For $m = 5$, there was no statistical difference between the average fitness of the best controller found for HAMMING and NCD; however, NCD learned more quickly. ENTROPY performed midway between HAMMING and NCD, and the non-behavioral methods. The fact that ENTROPY is measured in behavior space makes it more effective than FITNESS and EUCLIDEAN, but because it is not concerned with the structure of the behaviors, rather only their statistical properties, it is less informative about true behavioral similarity.

For $m = 15$, NCD improves upon HAMMING by 10% in the fitness, and reaches the same performance in about 80%

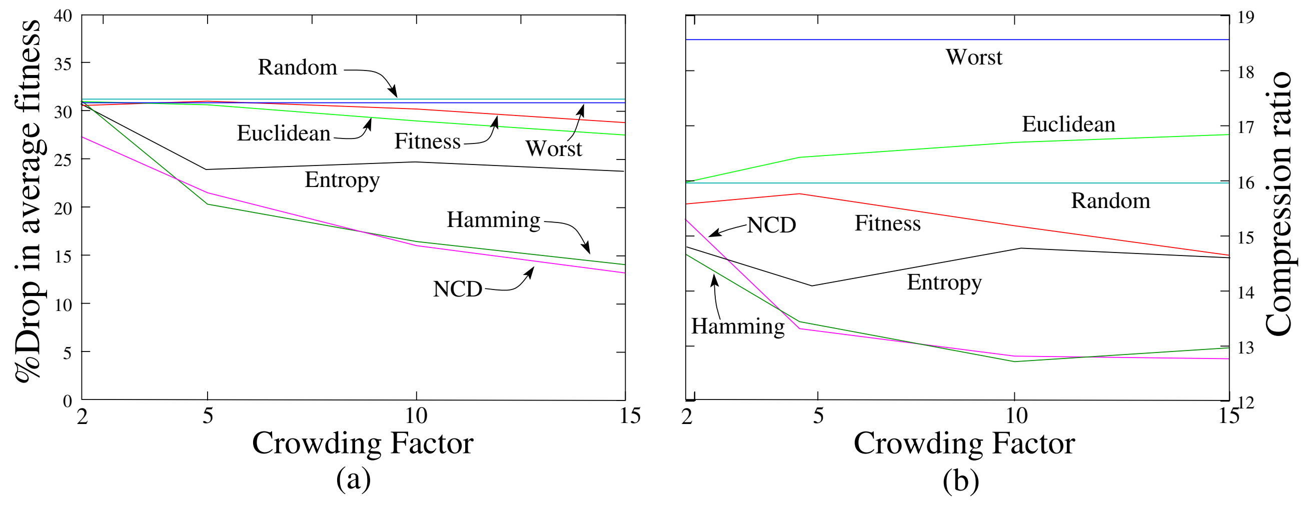

Figure 5: Generalization and complexity of evolved controllers. (a) The plot shows, for each of the four crowding factor values {2,5,10,15}, the average percent degradation in performance compared to the fitness obtained during evolution, when the best (most fit) network from each run is tested on 10,000 initial Tartarus boards. Overall, higher crowding factor yields better generalization, with a marked difference between the behavioral similarity measures and the other replacement schemes. HAMMING and NCD show less than a 15% reduction in average score. (b) This plot shows the complexity of the evolved controllers as measured by the ratio of the length of the behavior generated by the 10,000 test trials, and the length of the behavior after compressing it with `bzip2`. The behavioral similarity measures produce more complex controllers, requiring as many as 10,000 more symbols to represent in compressed form.

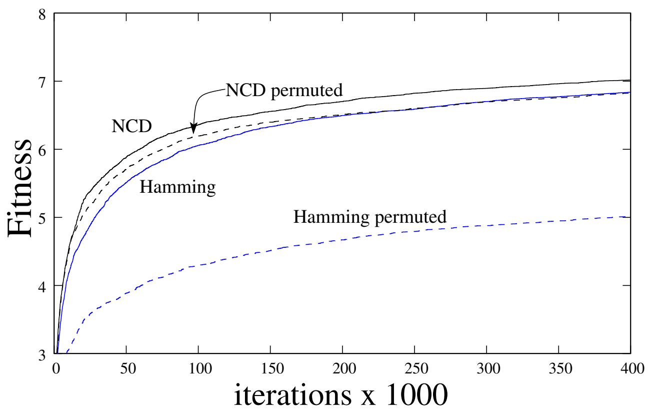

Figure 6: Permuted trials. The graph compares the performance of HAMMING and NCD when the order of the trials in the behavior are permuted. Because the trials of the two behaviors no longer line up, their Hamming distance is not a useful distance measure; meanwhile the compressor is still able to detect common regularity between the strings.

fewer iterations. This is significant as the difficulty of the task in relation to the number of blocks is not linear.

The performance advantage of NCD may not seem to justify the additional overhead of `bzip2` compared to HAMMING, even though both have complexity of $O(n)$. The more important gain from using NCD is that it is less sensitive to mis-alignments in the two behaviors. Figure 6 illustrates this point: for each evaluation, the 100 trials in each behavior were permuted. Unsurprisingly, Hamming breaks down completely, but NCD is still able to measure similarity in meaningful way. An entropy measure like d_{RH} would be completely unaffected by permutation, but, as demonstrated above, it is less useful for measuring similarity as it is not concerned with the order of the actions taken, only their expected occurrence.

4.4 Controller Complexity

The generalization of the evolved controllers was also examined to determine whether evolving with different similarity measures also had an effect on robustness. Figure 5a shows how much the performance of the controllers from each approach degrades on average after being tested on 10,000 random Tartarus boards. The robustness all of the similarity-based methods (i.e. all but RANDOM and WORST), improves with an increase in the crowding factor, especially for the behavioral distance methods which experience a drop of less than 15%, for $m \geq 10$.

It is not possible to de-couple the results of this generalization test from the absolute performance of each method. That is, one cannot determine from this data alone whether replacement by behavioral similarity encourages more general controllers to evolve or whether the increased generalization is a side-effect of the fact that these methods are better able to discover more fit controllers that by virtue of being more competent, are more capable of coping with novel boards. Further study is needed to compare the generalization of the controllers generated by the different similarity measures, at similar fitness levels.

Figure 5b looks at the “complexity” of the evolved controllers quantified in terms of the ratio between the length of the behavior generated during the generalization test (figure 5a) and its length after being compressed using `bzip2`, $C(\beta)/\beta$.

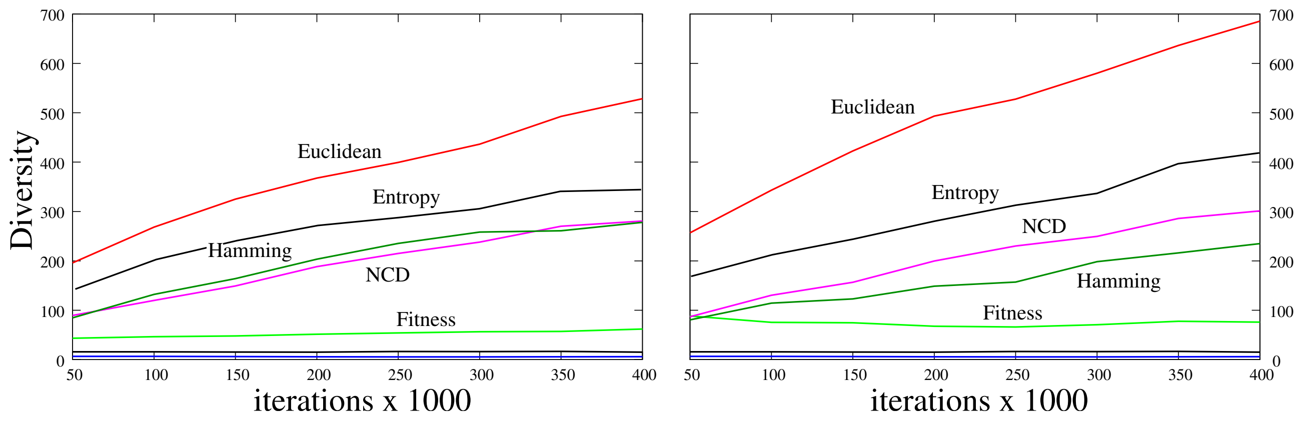

Figure 7: Genotypic Diversity. Each curve shows the average pairwise Euclidean distance between the chromosomes of a population for each method, averaged of 50 runs. The curves for Random and Worst are barely visible along the bottom of the graph (not labeled).

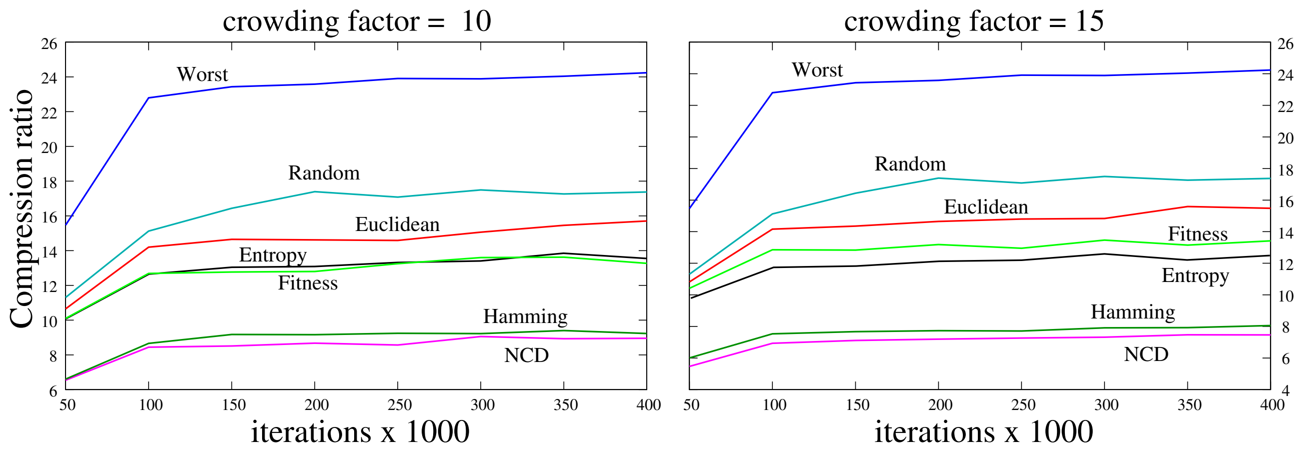

Figure 8: Behavioral Diversity. Each curve shows the average compression ratio (using `bzip2`) of the behaviors generated by the population every 50,000 iterations, averaged over 50 runs. Lower compression ratios indicate more behavioral diversity.

Intuitively, higher compression ratios signify less complex behaviors because they contain more easily identifiable regularity. For example, behaviors that consistently produce the same sequence of actions irrespective of the perceived state of the environment are simple, and this simplicity will be captured by the compressor since each repetition of the sequence can be coded by only a few symbols in the compressed representation.

In terms of complexity, in this heuristic sense at least, the picture is somewhat different from figure 5a where the baseline experiments, Random and Worst, were the poorest performers. Replacing the worst individual is again the least effective policy, leading to the simplest behaviors. This is to be expected given that the controllers for Worst were barely able to push three blocks against the walls, and did so by employing primitive, mechanical behaviors that loop around the board or go back-and-forth without regard for block locations. The Random approach, however, produces much more complex behaviors, with a compression ratio of about 16–again, this is correlated with the controller performance (figure 4).

The interesting result is for Fitness and Euclidean. While these two measures performed about same on the task, the complexity of their controllers diverges as the m increases: Fitness controllers gain in complexity, where Euclidean controllers lose complexity.

Finally, Hamming and NCD produced the least compressible, most complex behaviors.

4.5 Diversity

Figures 7 and 8 show how the different similarity measures affect diversity in both genotype and phenotype behavior space, respectively (only results for $m = 10$ and $m = 15$ are shown). The average genotype diversity (figure 7) is calculated by computing the average pair-wise Euclidean distance of the 100 real-valued chromosomes in each population, sampled every 50,000 iterations. Euclidean maintains the highest level of genotypic diversity, since it explicitly tries to maximize this measure. However, this does not translate into behavioral diversity (figure 8), which is measured by compressing all of the behaviors generated by a population, sampled at the same points during each evolutionary process as the genotypic diversity. The higher the compression ratio, the less diverse the behaviors of the population. By this measure EUCLIDEAN is less diverse than FITNESS which exhibits very low genotypic diversity.

The behavioral similarity measures maintain genotypic diversity indirectly by maintaining diversity where it matters most: in behavior space. The exception is ENTROPY which maintains relatively high genotypic diversity, but about the same behavioral diversity as FITNESS, even though it is acting directly on behaviors—the distance between the action probability distributions (i.e. K-L divergence) of two behaviors does not say much about the differences in the actual sequence of actions taken.

5. DISCUSSION AND FUTURE WORK

The overall results demonstrate that measuring similarity in behavior space can lead to faster learning by delaying convergence. Genotype Euclidean distance and Fitness distance proved to be very poor predictors of behavioral similarity, performing roughly equal to random replacement.

NCD outperformed the other measures and was less sensitive to the crowding factor parameter. While a measure like Hamming distance is easy to compute, it is limited to strings of equal length and generalizations such as Edit Distance are not practical (with complexity $O(|\beta_x||\beta_y|)$), and still only capable of detecting rather shallow sequence features. In contrast, NCD can exploit the deeper regularities favored by the particular compressor used, in sequences of arbitrary length. What is remarkable about NCD is that it does not care about the domain from which the data derives.

The immediate follow-up work will investigate a broader class of tasks where policies can generate behaviors of different lengths, and compare the performance of different lossless compressors plugged into NCD.

More analysis is required of not only the solutions discovered, but also of the evolving populations to gain more insight into how NCD affects the genetic composition of the whole population compared to other similarity measures. In the success stories using NCD for classification [1, 6, 7], the NCD is computed between all objects in the set, and then a clustering algorithm is applied to the resulting distance matrix. In our work, we have limited the application of NCD to $O(m)$ per iteration for reasons of efficiency. It would be interesting to compute the whole distance matrix, if not to enhance search, then as an off-line analysis tool for better understanding the topological structure of evolving populations in behavior space.

Another promising direction is to use NCD directly to drive the evolutionary process instead of the obligatory fitness-based selective pressure, as has been suggested by Lehman and Stanley [5]. In their work, evolutionary search is focused on finding novel solutions rather than fit ones. The belief is that the continual discovery of novelty will lead to ever-increasing complexity, which will inevitably generate interesting and possibly even useful behavior. So far, however, their experiments have employed only a very minimal, domain-specific concept of behavioral novelty. A clear test of the concept would be to use the much more general measure afforded by NCD.

6. CONCLUSION

This paper has presented an approach to sustaining diversity in evolving populations of policies by measuring behavioral similarity using *normalized compression distance*, a distance measure based on algorithmic information theory. The experiments compared this measure to three other similarity measures on the the Tartarus problem, using a simple crowding steady state genetic algorithm. The results showed that normalized compression distance produces more fit, general, complex controllers than the other measures. Given the extremely general applicability of normalized compression distance, these encouraging results bode well for its broader application to real-world domains.

Acknowledgments

This research was supported in part by the EU Projects IM-CLEVER (#231711), STIFF (#231576), and Humanobs (#231453), and the NSF under grant EIA-0303609. I would like to thank Jürgen Schmidhuber for providing invaluable input and inspiration for this work.

7. REFERENCES

[1] R. Cilibrasi and P. Vitanyi. Clustering by compression. *IEEE Transactions on Information Theory*, 51:1523–1545, 2005.

[2] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, The University of Michigan, Ann Arbor, MI, 1975. University Microfilms No. 76-09381.

[3] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 148–154. San Francisco, CA: Morgan Kaufmann, 1987.

[4] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In J. D. Farmer, C. Langton, S. Rasmussen, and C. Taylor, editors, *Artificial Life II*. Addison-Wesley, Reading, MA, 1991.

[5] J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life (ALIFE XI)*, Cambridge, MA, 2008. MIT Press.

[6] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. *Bioinformatics*, 17(2):149–154, 2001.

[7] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitanyi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[8] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1995.

[9] J. Seward. bzip2 and libbzip2, version 1.0.5: A program and library for data compression, 1996–2007. http://www.bzip.org.

[10] R. E. Smith, S. Forrest, and A. S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1992.

[11] A. Teller. *Advances in Genetic Programming*, chapter 9. MIT Press, 1994.