Changes in Genetic Representation

ABSTRACT

A theoretical framework for understanding the representation issue is presented. This framework is used to explain, as a corollary, conditions under which the genetic algorithm may be expected to eventually get to the absolute maximum of the search space. This framework also explains why utilizing random representations tends to improve performance in genetic algorithm optimization. Numerical simulations illustrating this phenomenon are presented on problems exhibiting complex fitness landscapes.

track:evolutionary combinatorial optimization

1. INTRODUCTION

The representation problem is an exceptionally difficult and important problem in the genetic and evolutionary algorithm world. More than that, it is an extremely important problem in nearly every engineering, physical science, or theoretical problem. The ability of the algorithm, theoretical study, physical design, etc. to complete the desired task is very strongly affected by the chosen representation.

In recent decades, several authors have explored the representation problem. Many of them have developed methodologies for generating representations that seem to be able to exploit properties of the problem efficiently. Others have spent time carefully crafting adaptive operators capable of molding themselves to the design problem. These methods are extremely effective if applied properly, but so far, no general methodology has existed.

During the same timeframe, studies have emerged which have indicated something very different from intuition. Several studies have explored the use of *arbitrary* representations in searching for optima in genetic algorithms. The analogy to this is a group of diverse individuals all discussing problems together, but each one having a vastly different point of view. It has been shown anecdotally and in various studies that this type of group action produces performance that can be significantly better than that of an expert system despite the sophistication of the expert.

The main problem with the application of these studies to the evolutionary algorithm community is the lack of theoretical backing for these results. While some evolutionary algorithms have been shown to benefit from the same type of diverse representation, this does not seem to be enough to warrent the use of this technique in general. A clear theoretical reason is much more satisfying in that it clearly delineates the conditions under which the method can be expected to be beneficial.

This paper examines the problem from a theoretical point of view. Major results of the paper include a theoretical description of the conditions under which a genetic algorithm can be expected to converge to the absolute optimum of the space. It is straightforward to infer from this an upper limit on the computation time though this is not attempted here. The discussion also includes a description of the reason for the random representation phenomenon in which random changing representations generate significantly superior results in comparison to single static representations. We present the results of the application of this methodology to several problems found in the literature.

The remainder of the paper is organized as follows. Firstly, a working knowledge of evolutionary algorithms is assumed. Section 2 describes the theoretical framework and the theoretical results of the paper. Section 3 presents the optimizations problems we investigate, the simulations used to carry out the optimizations, and the data obtained. Section 4 gives a brief discussion

and concluding remarks.

2. BASIC DEFINITIONS

This section will describe the theory used to examine the dynamics of evolutionary algorithms. The examination begins with a description of the basic assumptions of the class of evolutionary algorithms examined here. Of particular note is that this class does not include constructive algorithms such as genetic programming where the ultimate final solution size is unknown and the algorithm is essentially unbounded.

Once the basic assumptions have been clarified, we describe an *extended diversification operate* which is a generalized mutation/crossover operator. This operator is tied in with the reproduction operator, and can be used to understand why the genetic algorithm is not generally limited to specific subspaces.

We continue with an examination of the meaning of different representations in optimization algorithms. This examination leads to the idea that optimization algorithms behaving like evolutionary algorithms tend to move between regions of the search space. The movement is essentially connected to the connectiveness of the space, which itself is a result of the representation. The representation, therefore, is important in the sense that different regions of the search space may be "further" or "closer", according to the number of steps required to reach them, as a result of which representation is used. This view of representation is used to determine the effect of the various random representations.

2.1 Extended Diversification

We assume in the following work that the systems to which we apply this formalism have the following properties:

1. The algorithm maintains a population, which is a set of vectors, that are stored in memory. The vectors are stored in memory and acted on by operators which define the evolutionary algorithm. The state of the evolutionary algorithm is defined by the population and the current operator being employed. which is a set of individual vectors stored in memory.

2. The diversification operator(s) introduce new elements to the population using a combination of crossover and mutation events. Diversification operators do not include selection.

3. Selection culls the population by replacing some elements with others, preferentially replacing lower scoring individuals with higher scoring individuals.

4. We assume we are working with a finite search space Γ.

The details of these behaviors are not important for the arguments made below, though the arguments made from here on out will be true as long as these assumptions are true.

There are, of course, two radically different types of populations. In the first type, the population is finite (which is required for most practical applications involving EAs), and in the second, the population is infinite. In this paper, we will explore the finite population case. In this case, the population has a specific number of elements which is maintained by the evolutionary algorithm. Additions of new elements to the population must be accompanied by removals of elements from the population. This is an important consideration for what will follow.

The diversification operator generally operates in the following way: based on the population (or in some cases independently of the population), the diversification creates new vectors for consideration. These vectors either immediately become part of the population, replacing vectors in the population, or are subject some culling. The methods for including these are varied. They may include finding a vector whose score is lower than the new vector and replacing it, or randomly choosing a new vector to replace. When this step is combined with selection one typically finds a lower-scoring vector to replace with the new vector. If one cannot be found, the new vector is not added to the population. Generational selection typically consists of a removal of the lower scoring individuals from the population and replacement either with an individual produced by a diversification event or by an individual which is a copy of an existing individual in the population.

Often times, the diversification operator is limited in the sense that a single application of this operator cannot transform any given vector into any other given vector in the search space. As an example diversification operators derived from single point mutation and any finite-point crossover operations cannot change all binary vectors into any other binary vector. In this case, it is the repeated action of the diversification operator which allows the vector to be changed completely from one vector to another vector. In population-based search algorithms, it is possible for diversification steps to act additively, extending the capability of the diversification operator. We call such a pseudo-operator an ***extended diversification operator*** and each sequence of connected diversifications an ***extended diversification event***.

Let us now take the original population of N elements and enumerate them. We represent these as $\{x_1, \ldots, x_N\}$. Then, each element discovered by the extended diversification operator can be added to this list of elements. Thus, if M extended diversification events have occurred, then the sequence of elements is given by $\{x_1, \ldots, x_{N+M}\}$.

We may think of the current population as being the subset of elements from this sequence of elements that is being considered when the next element is developed. Thus, we may write the next element as a function of the current sequence, with an emphasis on the current population. That is, we may write a recursion relation as

$$x_{N+M+1} = f\left(P; x_1, \ldots, x_{N+M}\right) \quad (1)$$

where f is the diversification operator. Let us designate the entire set of numbers a sequence X.

What this means, then, is that the entire evolutionary algorithm can be likened to a method for generating an infinite sequence. The sequence of numbers is complex to analyze, but it is still a *deterministic* sequence of numbers.

The main problem is that the sequence is often times a *repeating* sequence of vectors containing multiple copies of many of the elements. Simply because an element has been removed from the population does not require this element to never again reappear in the population. Let us now consider the subsequence of elements of X which do not appear earlier in the sequence. Let us designate this sequence of elements as $Y = \{y_i\}_{i=1}^{Max}$ where Max represents the final unique discovery in the sequence X. For finite spaces, Max is finite; for infinite spaces, Max may be infinite.[1]

We can then define different types of algorithms. An optimization algorithm can be defined as *eventually stagnant* if $Max < |\Gamma|$ where Γ is the number of elements in the entire search space. An example of an algorithm that is eventually stagnant no matter the starting point is a hillclimbing algorithm, no matter the number of starting vectors or the position of the starting vectors.

One important question is whether or not evolutionary algorithms are eventually stagnant. This is dependant on the evolutionary algorithm's diversification operator. The following proposition addresses the future of optimization algorithms whose diversification operators or extended operators have no limitation in their reach. Let us define the probability of an extended diversification operator D changing vector $\overrightarrow{v_1}$ to vector $\overrightarrow{v_2}$ as $p_{D\left(\overrightarrow{v_1},\overrightarrow{v_2}\right)}$.

PROPOSITION 1 *Suppose that an evolutionary algorithm has an extended diversification operator D such that given any two vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ in the search space Γ, $p_{D\left(\overrightarrow{v_1},\overrightarrow{v_2}\right)} > 0$. Then the evolutionary algorithm is not eventually stagnant.*

PROOF. To prove that an evolutionary algorithm is eventually stagnant, we must show that there is a state in which there is no way to reach elements of the search space that have not been reached using the extended diversification operator. However, this would imply that $p_{D\left(\overrightarrow{v},\overrightarrow{u}\right)} = 0$ for some elements v and u in the population.. Since this is not true by assumption, the conclusion follows. □

The importance of this proposition comes from its application to the optimization, and forms the motivation for the use of evolutionary algorithms. The following Corollary illustrates its use in optimization.

COROLLARY 1 *An evolutionary algorithm which has an extended diversification operator d such that given any two vectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ in the search space Γ, $p_{D\left(\overrightarrow{v_1},\overrightarrow{v_2}\right)} > 0$ will always find the space's optimum.*

PROOF. Since the algorithm is not eventually stagnant and the search space Γ is finite, the sequence Y will eventually include the optimum. □

Corollary 1 provides a clear description of the motivation for using evolutionary algorithms. It is clear that, using evolutionary algorithms, the optimum will eventually be visited by the algorithm. In order to do this, it is merely necessary to either construct a diversification operator that, in one step, has a p which is nonzero for all possible mutations, or an extended diversification operator which does the same.

2.2 Representational issues

In fact, most evolutionary algorithms have a limited diversification capability. Binary mutation operators and crossover operators can typically not change any single vector into any other in a single iteration. However, repeated uses of the operators have this effect, and so the extended diversification operator can be seen to also have positive p in this case.

Proposition 1 and Corollary 1 also indicate that the longer the optimization operator is run, the better the result, as even those better results that are unlikely to be visited can be found, given enough time. This result provides the theoretical motivation for the recent result of Cantu-Paz and Goldberg [?] who both theoretically and empirically found the same result.

Despite the positive impact of these considerations, the simple fact is that no indication is given as to how long one might expect to wait in order to obtain these improvements. This is very important because we would like to know not only that the search will succeed, but also that the search will take a reasonable amount of time.

Let us assume that we have a population P, which is a subset of the sequence of elements X. In general, this population is made up of copies of the maximal element in the population, and other members that do not score as well as the maximal element. We may view

[1] Note that this sequence is identical to the one used in the arguments for the No Free Lunch theorems.

these as two types of elements. The first is the current best, and the others are intermediate states of a diversification event seeking a new best. In systems involving elitism, the current best is a protected element in that it will not be replaced by a less fit individual. In those not employing elitism, the best individual is not protected, and so it can be replaced by a less fit individual, depending on the selection operator employed. In what follows, we will assume that the methodology utilizes elitism.

We define a *basin of attraction of a local maximum* to be the the maximal set of vectors such that

1. the set is an open set in the set theoretic sense,
2. the local maximum is in the interior of the set,
3. the fitness values of all the elements in the set are lower than that of the local maximum.

We define the ***depth of a basin of attraction around a vector*** $\vec{v}$ to be the minimum number of diversification steps required during a diversification event in order to reach another vector whose fitness value is greater than the local maximum.

Using these definitions, we can generate the following immediate consequences.

PROPOSITION 2 *Suppose that an evolutionary algorithm utilizes a diversification operator in which each separate step is uncorrelated, unbiased, and chooses between the same number of potential alternatives. Then the computation time required to find a vector outside of the basin of attraction increases at least exponentially with the depth of the basin.*

PROOF. If the assumptions are as given above, then the probability of each step in the diversification event is constant. The probability of the extended diversification resulting in a specific new individual from the optimum is a function of it's relative position in the search space; that is, the probability of making s necessary diversification steps is p^s . This means that the number of diversification steps needed to reach the edge of the basin of attraction from the optimum is $\frac{1}{p^s}$. In general, the population will not be completely converged. This means that each individual will be a specific minimum distance to the boundary of the basin of attraction. If this diversity is maintained, then, on average, the population will have individuals a distance of s' diversification steps from the optimum. This then increases the probability to $p^{s-s'}$. □

This result is important because it indicates that an evolutionary algorithm is likely to get trapped in a basin of attraction for an exponentially increasing amount of time, depending on the depth of the basin. Thus, even though the algorithm cannot be trapped, as Corollary 2.2 indicated, the algorithm's completion time tends to become continually larger.

Note also that this result indicates, implicitly, that any population with a diversity large enough that $s - s' < 0$ is unstable with respect to the local optimum. As a result, the population may be expected to very quickly discover one or more superior vectors, and move away from the local optimum. This explains why many researchers have found, anecdotally, that one needs to keep the diversity of the population high in order for the algorithm to succeed.

In their 2005 paper, Rand and Riolo [6] demonstrate that a dynamic re-encoding of the search space has significant advantages over a static encoding of a search space. This result has no theoretical basis to date, but has been reported by other researchers as well. We turn now to a description of why this is so.

Since basins of attraction have depths that are functions of the diversification operator, and the diversification operator is a function of the encoding of the search operators, the encoding of the space has a very large effect on the search algorithm. Some encodings will cause some basins of attraction to be very shallow. In this case, the search algorithm should progress very quickly. However, another encoding might make the basins quite deep. This would make the optimization happen very slowly, with a relative speed exponentially related to the basin depths. As a result, a re-encoding might be able to increase the speed of the algorithm. This is the topic of the next corollary.

Before we delve into the corollary, we pause to consider what a re-encoding of the diversification operator means. The diversification operator may be thought of as a recursion relation which takes the current population and generates a new vector. That is,

$$D(\mathbf{P}) = v \tag{2}$$

where v is some vector. Re-encoding this diversification operator causes a change in the mapping. Random re-encoding changes the mapping in a random way, generating a completely new mapping. The basins of attraction for specific maxima may be shredded by this process, and the new locations of optima may be completely unrelated to their previous positions. Under these circumstances, a re-encoding of the diversification operator will have an unpredictable effect on the overall shape of basins of attraction around the optima. Moreover, it the relative distance between any two optima will generally change, with the new positions unrelated to the old ones.

The act of re-encoding the diversification operator can have a profound effect on a continuing optimization. Not only will the population have completely new connectivity, and therefore be able to spread itself throughout the space very differently, but this new connectivity

can change the distance (in number of diversifications) between any two vectors. This means that those reencodings which bring them closer together will tend to increase the speed of optimization, while those that make them further apart will tend to decrease the speed of optimization.

COROLLARY 2 *An evolutionary algorithm employing nonbiased mixing will converge to an optimum faster than an evolutionary algorithm which does not employ mixing with a probability that falls off as* 2^{-n} *with* n *mixing events.*

PROOF. Nonbiased mixing will as often connect two basins of attraction together via a reduced number of diversification steps as with an increased number of diversification steps. The probability of a random mixing event causing a beneficial change in the diversification-distance between two optima's basins of attraction as opposed to a neutral or non-beneficial change is $\frac{1}{2}$. In the event that this occurs, the probability of moving to the higher basin will increase exponentially, and the convergence will be improved. The probability that this will not happen in n mixing events is $\left(\frac{1}{2}\right)^n$. □

Note that while random mixing events are often times desired, mixing events that maintain continuity of the search space might also be beneficial. Mixing events that maintain the integrity of basins of attraction must be homeomorphic[2] transformations, which are a subset of the possible transformations. Moreover, they cannot be linear, as we are interested in the number of mutations needed to transform one to another, not their point in state space. Thus, we must create nonlinear homeomorphic transformations of the mutation and crossover operators in order to improve the performance of the algorithm. Continuity arguments may be dropped in discrete spaces, as they have no rigorous meaning.

3. TEST PROBLEMS

The results of the preceding sections would seem to indicate an approach to genetic search that is not immediately obvious. The "reach" of the evolutionary algorithm is the issue when considering how an EA can jump from one basin of attraction to another higher basin. Interestingly the reach is related to the representation of the search space. What this means is that changes in the representation of the search space can alter the functionality of the algorithm. Moreover, perhaps the most unintuitive result is that random representation changes would seem to be able to have a beneficial effect on the performance, in much the same way that uniform crossover had a beneficial effect on genetic algorithms. Since this result is dependent on the connectivity of the search space, it would seem to be broad enough to have an effect on the design of nearly all evolutionary algorithms, not only one of , and that

In this section, we examine the performance of simulations written with the representation alterations and compare it to that of simulations that do not employ the representation alteration. Our simulations are real-encoded genetic algorithms employing single point mutations and single point crossover. Each simulation contains populations of 100 initially randomly assigned individuals. The crossover and mutation probabilities are 0.1. The algorithm utilizes an elitest mechanism so as to stabilize the population against variations that tend to reverse progress.

In order to examine the effect of change of representation, we modify the general genetic algorithm by adding the following structure. At each step we calculate a matrix M and its inverse IM. These matrices are used to "recast" the populations into a new encoding. Initially, both matrices are assigned to I, the identity matrix. Once a new set of matrices is created, each element of the population transforms to another element as

$$v' = IM_{n+1}M_n v. \tag{3}$$

This "mixes up" the search space according to the matrix M_n, and so changes the effect of both the mutations and the crossovers. Each successive application reverses the last transformation and applies the current transformation. As a result of the transformation, the connectivity of the search space becomes affected, making some vectors "closer" to others in terms of the number of mutations and/or crossovers required to reach one of the vectors from the other vector. In our simulations, the recoding step is applied once every 10,000 iterations. This makes the total number of recodings 20, as each simulation runs for 200,000 iterations.

We investigate the method using two problems found in the literature and two homemade functions. The two problems found in the literature are the Rastrigin and the Griewangk problems. The problems that are homegrown are

$$J_1 = x_1^2+x_2^2+x_1x_2-14x_1-16\sin(x_1)+10\cos^2(x_3-10)+ 4\cos(x_4-5)+2\cos(x_6-1)+5x_7^2+\sin\left(7(x_8-11)^2\right)+ \sin^2(x_10)+\sin(x_1)+5\tan^{-1}\left(\frac{x_2}{x_6}\right)+x_1\cos\left(\frac{x_9}{3}\right)$$

and

$$J_2 = \cos(x_5x_{10})+\tan^{-1}\left(\frac{x_3}{x_4}\right)-2\cos\left((x_6-x_7)^2\right)+ \cos\left((x_3-x_8)^2\right)+\sum_{i=1}^{10}\sin(x_i)+4\tan^{-1}\left(\frac{x_8}{x_9}\right).$$

[2] A homeomorphic transformation is a bijection (one-to-one and onto) which preserves the neighborhood structure of the space. In other words, points that are nearby one another in the first set are still nearby one another in the transformed set.

Table 1: This table gives the performance of the modified genetic algorithm on various optimization problems. The first four columns represent the non transformed search space, while the second four represent the transformed search space.

	J_1	J_2	G	R	$J_{1,r}$	$J_{2,r}$	G_r	R_r
max	9.28	73.77	-0.015	0.00	13.03	97.29	0.00	0.00
average	8.16	73.76	-0.077	-7.46	8.03	92.58	-0.05	-9.86
stdv	1.17	0.035	0.037	9.06	1.66	4.39	0.05	8.72

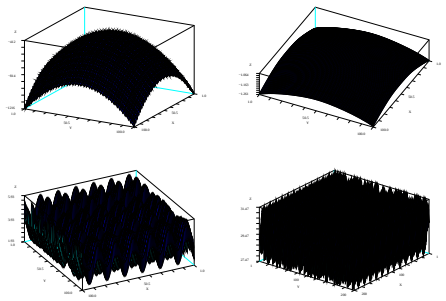

Figure 1: These problems are optimized using the genetic algorithm described above. The previously reported functions from the literature (top) have a very different structure from the new ones (bottom).

These problems are illustrated in Figure 1. Note that the gross structure of the J_1 and J_2 functions is very different from that of the Rastrigin and Griewangk functions. As a result, one might expect the performance of the GA to differ on these two "mini-classes" of functions.

Aside from the GA modifications described above, we use a generational GA with a standard multi-point mutation operator and uniform crossover operators. The mutation probability is 0.1 and the crossover probability is 0.9. The population is of size 100, and we utilize a proportional reproduction operator. We also utilize an elitist mechanism, preserving one vector with the highest (to date) score.

We report, for each problem, the average, maximum, and standard deviation of the final optimal vector with and without the sparse recoding. These data are given in Table 1.

In Table 1, we present the data of several optimization problems solved using the GA described above, both employing recoding and not employing recoding. In each case, the average and standard deviation are reported, along with the best performace over one hundred runs.

While three of the functions perform statistically identically, one of the functions (J_1) performs vastly better with the representation change. This is not suprising as the initial state of the recoded GA is identical to that of the non-recoded GA. Thus, the performance can only differ once the recoding takes place. In several of the problems, the performance of the recoded GA is very significantly superior to that of the standard GA. It is interesting that this performance increase can be achieved with no significant change to the algorithm outside of the periodic recoding.

4. SUMMARY AND CONCLUDING REMARKS

The main contributions of this paper lie in the cementing of a theoretical understanding of a somewhat heretofore misunderstood phenomenon in evolutionary computation. That is that the representation of the various operators, even if randomly assigned, can lead to very real improvements in the performance of evolutionary algorithms. While this isn't always the case, some problems that one can find have fitness landscapes with structure that doesn't match with a particular arbitrarily decided upon structure.

This result echoes the work of other researchers examining the various encodings of the search space [5, 3, 4, 2, 7]. These studies have examined the static representation of the genetic operators along with various dynamic adaptations. In view of the fact that the uniform crossover operators apparently needed no specific encoding to be effective, it would seem to be a straightforward result.

Nonetheless, our examples now join a large amount of previous work that implies what our theoretical results confirm. The utility of this result lies in its application to evolutionary algorithms, indicating that the algorithm could benefit from a random scrambling of the search space. The design of algorithms routinely utilizing such a scrambling operator would seem to be useful, and not obvious from the biological analogy.

5. REFERENCES

[1] E. Cantu-Paz and D. Goldberg. *Are multiple runs of genetic algorithm better than one?* **Lecture Notes in Computer Science, 2723**, pp. 801-812, 2003.

[2] H. Kargupta and B. Park. *Gene expression and fast construction of distributed evolutionary representation.* **Evolutionary Computation**, 9(1), 43-69, 2001.

[3] S. Kazadi. *Conjugate Schema in Genetic Search.* **Proceedings of the Seventh International Conference on Genetic Algorithms**, San Mateo, Ca: Morgan Kaufmann Publishers, pp. 10-17, 1997.

[4] S. Kazadi. *Conjugate Schema and Basis Representation of Crossover and Mutationi.* **Evolutionary Computation**, v6(2), 129-160, 1998.

[5] M. Munetomo and D. Goldberg. *Linkage identification by non-monotonicity detection for overlapping functions.* **Evolutionary Computation**, 7(4), 377-398.

[6] Rand and Riolo. *The problem with a self-adaptive mutation rate in some environments. A case study using the shaky ladder hyperplane-defined functions.* **Proceedings of GECCO 2005**, Washington D. C., USA, pp. 1493-1500, 2005.

[7] G. Syswerda. *Uniform crossover in genetic algorithms.* **Proceedings of the Third International Conference on Genetic Algorithms**. D. Schaffer, (Ed.), Morgan Kaufmann, San Mateo, CA, 2-9, 1993.