Rank Based Variation Operators for Genetic Algorithms

Jorge Cervantes
Universidad Autónoma Metropolitana
Departamento de Matemáticas Aplicadas y Sistemas, México D.F. 01120
JorgeCervantesO@aim.com

Christopher R. Stephens
Instituto de Ciencias Nucleares
Universidad Nacional Autónoma de México
A. Postal 70-543, México D.F. 04510
stephens@nucleares.unam.mx

ABSTRACT

We show how and *why* using genetic operators that are applied with probabilities that depend on the fitness rank of a genotype or phenotype offers a robust alternative to the Simple GA and avoids some questions of parameter tuning without having to introduce an explicit encoded self-adaptation mechanism. We motivate the algorithm by appealing to previous theoretic analysis that show how different landscapes and population states require different mutation rates to dynamically optimize the balance between exploration and exploitation. We test the algorithm on a range of model landscapes where we can see under what circumstances this Rank GA is likely to outperform the Simple GA and how it outperforms standard heuristics such as $1/N$. We try to explain the reasons behind this behaviour.

ACM Primary Classification:
I.2.8 Problem Solving, Control Methods and Search
Subjects: Heuristic methods.
ACM Additional Classification:
J.2 Subjects: Engineering.
J.3 Subjects: Biology and Genetics.
J.4 Social and behavioral sciences.
General Terms:
Algorithms, Performance, Reliability, Theory.
Keywords:
Parameter Tuning, Rank GA, Robustness.

1. INTRODUCTION

Optimal parameter setting and optimal operator tuning for a given class of Evolutionary Algorithms (EAs) [1, 2] and a given class of fitness landscapes is a difficult task. As has been shown in [3] for instance, "optimal" mutation rates can vary from 0 to nearly 1, depending on with respect to what one is "optimizing", and can depend on many factors, such as the fitness landscape, population size, population status and the set of operators being used. In this paper we extend previous work [4] to show how and specially *why* using rank-based genetic variation operators can greatly ameliorate the problem of parameter tuning in EAs while adding much more robustness to search.

There have been three recurring themes with respect to an "optimal" mutation rate: i) that a rate $p \sim 1/N$ is preferred [5], where N is the string length; ii) that a preferred rate should be dynamic [6, 7, 8, 9, 10] $p = p(t)$; and iii) that the error threshold p^* offers guidance as to an optimal rate [11, 12, 13]. Of course, the desire has been to derive useful heuristics for setting mutation rates and, the more universal the better. The well known $p = 1/N$ heuristic is too universal in that it is independent of the fitness landscape. The error threshold p^* of a signal in the fitness landscape is defined as the mutation rate above which this signal would not be able to influence the population's dynamics and thus it offers no preferential selective benefit. This error threshold does depend on the fitness landscape details and has been argued to offer guidance to an optimal balance between exploration and exploitation. However, in order to use it one has the problem of how to calculate or measure it for a "black box" landscape. Other work [9] has argued that an optimal mutation rate depends not only on the fitness landscape but also on the actual population, and hence is dynamic. To have to go to such a fine-grained level is clearly not practically feasible. A possible solution is to consider a self-adaptive mutation rate that evolves with the population as in [6, 9, 10]. However, this can be computationally expensive and can end up with a premature convergence of the mutation rate itself.

With this in mind, we see the need of a method for applying different mutation rates to different subpopulations all running at the same time. In [14] a fitness based rule is used to assign mutation and recombination rates, with higher rates being assigned to those genotypes that are most different in fitness from the fittest individual in the population. One disadvantage of this approach is that, if there is more than one individual with maximum fitness, i.e. they belong to the same optimal (or best found) neutral net, then all will be assigned mutation rate zero and hence there is no exploration on the neutral net.

Another adaptive scheme, is that proposed in [15] where a fitness rank based rule is used. However, there, the chosen mutation rate is not applied bit-wise but rather to the full genotype. This means that each individual can have only one mutation per generation if any. This limits the possibility of escaping local optima because the actual mutation rate can not be comparatively higher than 1/N and it has been shown in [3] that higher mutation rates can be optimal under certain circumstances.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

In this paper we justify (based on previous theoretical work) and demonstrate the potential utility of a method proposed in [4], where each subpopulation is assigned a mutation rate directly proportional to fitness rank and combined with a suitable recombination operator. Besides offering a more general framework and having a wider applicability, our analysis here is much more firmly grounded in previous theoretical analyses as well as being tested on a wider range of fitness landscapes than the previously referred works. This justification explains why the method is useful and also gives the possibility to apply the method almost directly to other areas in evolutionary computation.

2. METRICS AND LANDSCAPES

We will use as an evaluation metric the Run Length Distribution (RLD) [16]. These are curves that show the characteristic behaviour of a given search algorithm on a given environment, such as a fitness landscape and/or initial population. RLDs are obtained by performing many runs (here, throughout, we use 100) of an algorithm, evaluating the effort (number of fitness evaluations) needed to find the optimum associated with a given run, sorting these runs from minimum to maximum effort and graphing the resultant curve. The curves are noisy when the number of runs is small but, as this number grows, noise is reduced, and the curves are smoother. RLDs provide more information than simple summary statistics, such as min, max and average effort, since one can see at a glance the proportion of runs that are expected to find the optimum below any fixed level of effort, as well as phenomena that affect only a fraction of the runs.

We will examine a range of different well-known "model" landscapes, taking the point of view that a principal task of search is to distinguish "signal" from "noise" by a suitable balance between exploration and exploitation. It is also important to differentiate between offline and online signals, the former being the signals of the fitness landscape itself, and the latter those signals that exist in the current state of the algorithm, e.g. in the current population.

We assume that most of the real life landscapes are multimodal and that, often, a punctuated equilibrium takes place in the search process whenever the population gets temporarily stuck in a local optimum and then, when it finally gets off that suboptimum, it jumps (or transits) from a region of one type to another. In this sense we believe that a two signal model can tell you a lot about how genetic operators interact in the search process.

Throughout this paper we have considered "model" as opposed to "real-world" landscapes. Although we have tested the algorithms on a variety of the latter, with positive results, we here particularly wished to be able to relate this work to previous theoretical and empirical work on optimal mutation rates that was also carried out on "model" landscapes. By so doing we are patently not trying to create the "killer" GA but rather are trying to rectify in a fairly straightforward manner certain deficiencies of the Simple GA (SGA) by appealing to results and observations gleaned from more theoretical research. We consider:

1. The Counting Ones Landscape (ONEMAX). To understand what happens in a landscape that is very easy for the GA and which has been studied extensively by others. $ft(x) = 1 + NumOnes(x)/N$

2. A Deceptive Trap landscape (DTrap). The same as ONEMAX but with the string of all 0s having its $ft > 2$ thus being the optimum. This case is the hardest for the SGA.

3. Two anticorrelated signals. A two-needle NIAH (Needle In A Haystack) landscape with one suboptimal degenerate needle ($ft = 2$) placed at the antipode of an optimal single needle ($ft > 2$). The rest of the landscape is flat ($ft = 1$). A deceptive landscape too but with a different shape for the deceptive signal so that not all of the landscape pulls the population away from the optimum.

4. Two correlated signals. A two needle NIAH landscape with a suboptimal needle ($ft = 2$) placed at different Hamming distances from an optimal needle ($ft > 2$). The rest of the landscape is flat ($ft = 1$). To compare how well the algorithms exploit correlations.

5. Concatenated Blocks. The string is split into k blocks each one having its own NIAH or DTrap landscape. The overall fitness of a string is the sum of the fitness in each block. Here we test how well an algorithm uses the landscape modularity and reuses partial solutions found during a run.

3. OPTIMAL MUTATION RATES

In choosing a suitable mutation rate p for an EA, there are three standard approaches: find an optimal rate experimentally; use a standard heuristic; or use a self-adaptive algorithm. Each has its advantages and disadvantages. Therefore, in order to justify the method we are using [4] with respect to mutation rates, it is important to understand what are the traits that most affect the optimal value of p.

An investigation of this nature was carried out in [3] on a set of model landscapes in order to try to understand under what circumstances and why certain mutation rates were preferred. For example, using a "needle-in-a-haystack" landscape as a model for a landscape with a global optimum that is relatively uncorrelated with the rest of the landscape, i.e. only one "signal", it was shown that in this case the error threshold p^*, theoretically approximated in a straightforward manner[1], can be too high or too low. When this signal is present in the population, exploration of the search space with a p in the vicinity of this approximation is quite close to random search because there is a high probability that the signal is quickly lost in the population and also there is a low probability of its recovery since the rest of the landscape is flat. So, if one wants to preserve the signal in the population, one must use lower mutation rates that are about half of this approximation, in other words, the ""real" p^* is much lower. It was also shown (in [3]) that the "real" p^* decreases as a function of decreasing population size and increasing noise level in the fitness function and also if recombination is used (from [11]).

In this landscape the separation between exploration and exploitation is total [3] - optimal search being random with a mutation rate of 0.5, until the optimum is found, whereupon the optimal rate is 0 in order to preserve the optimum in the population. The standard heuristic $1/N$ works well in

[1] $p^* = ln(G)/N$, where $G = ft_{needle}/ft_{hay}$ is the fitness gain of the needle with respect to the hay.

this landscape in terms of the exploration phase, but not the exploitation phase. This example shows also how an optimal mutation rate naturally depends on the state of the population as well as what is the objective of the search. By making the "needle" degenerate (a plateau signal consisting of more than one genotype) it was also shown that there is a universal relation between the error threshold and the fraction of optimal configurations in the fitness landscape and, consequently, on the optimal mutation rate. In fact, beyond a redundancy of about 5% of the configuration space, the error threshold disappears completely.

In landscapes with more than one "signal", where the optimum's signal competed against a second local degenerate one, it was shown that there exists a mutation rate, p_f, above which the suboptimal signal is preferred asymptotically, thus clearly misleading the search. Further, it was seen that, more often than not, $p_f < p^*$ and hence was of greater practical relevance. Moreover, even when $p < p_f$ there is a substantial part of the evolution wherein the suboptimum is preferred [3]. p_f increases when the Hamming distance between the optimum and suboptimum is very low and decreases as the fitness differential between the optima and suboptima decreases [3]. It was shown also that, when the suboptimum is anti-correlated with the optimum (at the maximal Hamming distance), an optimal mutation rate for minimizing computational effort is close to one! So, generally, a suboptimum can either help or hamper the search for the optimum depending on its Hamming distance to the optimum. For an optimum and suboptimum quite close together, where the initial condition is to have all the population in the suboptimal state, there exists an optimal mutation rate for maximizing the success rate of the algorithm and it may be substantially lower than $p = 1/N$ or the theoretical error threshold $p^* = ln(G)/N$. For an optimum and suboptimum quite far away from each other the optimal p may be substantially higher than that [3].

In the counting ones landscape each Hamming class (the set of all genotypes at equal Hamming distance to the optimum), considered as degenerate signals, exhibits its own error threshold, there being an increase in this threshold as a function of the Hamming distance from the optimum. However, in this case, unlike the "needle-in-a-haystack" landscape, the optimal string dominates for any mutation rate, though the probability to find it decreases as the mutation rate increases. Cooperation here between "signals" (Hamming classes) is total and competition is absent resulting in a situation where these signals act effectively as one wide signal.

So, generally, the $1/N$ and error threshold heuristics are too universal. Canonically, they are too high for search in landscapes with multiple correlated peaks. Uncorrelated landscapes require a higher mutation rate than correlated ones, while landscapes with anti-correlation, i.e., with "deceptive" peaks, require even higher rates. Of course, one can argue that a complicated landscape can exhibit all these features. A mutation rate well below the $1/N$ and error thresholds is recommendable in the case where either there are weak peaks that are quite close in fitness value and Hamming distance to the optimum or when the optimum is present in the population, and values well above those are recommendable when there are robust suboptimal signals far away from the optimum and the optimum is not present in the population.

4. A RANK BASED VARIATION GENETIC ALGORITHM - RANK GA

In section 3 we argued that knowledge of the fitness landscape under consideration can aid in the choice of a suitable mutation rate. However, we saw that the range of potential optimal rates was very large, depending not only on the landscape but also on the current state of the population. In many search problems detailed knowledge of the fitness landscape is not readily at hand. The question then is, how can the above lessons be applied to such "black box" search problems? Below we describe the proposed method to deal with these difficulties. Since this method is an extension of that described in [4] we cite it accordingly.

4.1 Rank Proportional Mutation Regime

Given that the optimal mutation rate covers a large range, the first lesson to be learned is to assign a large range of possible mutation rates to the population [4]. Secondly, we have to choose which individuals will have a certain mutation rate applied [4]. We thus define a rank-based mutation operator such that in a given generation t it is applied in the following way:

The mutation rate range R_m will be specified by a minimum and maximum mutation rate - $p_{\min}$ and $p_{\max}$ respectively [4]. As we have seen in section 3 a natural range to cover "any" eventuality is $p_{\min} = 0$, $p_{\max} = 1$ [4]. Clearly, if there is knowledge of the fitness landscape details and the population status, then a lower $p_{\max}$ or a higher $p_{\min}$ may be appropriate but if not, a lower $p_{\max}$, for instance, could leave out some mutation rates that may be just the optimal ones for the current problem.

The next step is to divide the current population into m groups Obviously, if $m = 1$ then we have the configuration of the SGA, i.e. the same mutation rate is applied to the whole population. At the other extreme, if $m = s$, s being the population size, then every individual is associated with their own mutation rate. By making such a rule we are essentially turning the population from one where the exploration aspect of search is the same, to one where it is different for every individual (in [4] the concept of groups is not present but it is equivalent to using $m = s$).

Having defined m groups, the next step is to choose from R_m a set of m mutation rates. Here we will use the simple deterministic rule of choosing the m mutation rates such that $(p_{\max} - p_{\min})$ is divided into m equal parts (as in [4]).

The next step is to assign a given mutation rate to a given group. The most simple method would be to do it at random, but fixed in time, i.e. a given group is given a mutation rate and keeps that mutation rate throughout the evolution. Alternatively, a different mutation rate could be assigned randomly every generation. Of course, by doing so one is not taking advantage of any information that the population possesses, for example in terms of the population fitness values or the explicit genotypes present. Here, we will consider assigning mutation rates according to the fitness rank of the group, i.e. the average fitness of a group (a single individual for our tests) is evaluated then the m groups are ranked and mutation rates are assigned from $p_{\min}$ to $p_{\max}$ according to their rank. The lowest mutation rate is assigned to the top ranked group, the next lowest rate to the next ranked etc., until the highest rate is assigned to the lowest

ranked group. In our case (as in [4]) this is done linearly:

$$p_x = p_{\min} + (p_{\max} - p_{\min}) * (Rank(x) - 1)/(m - 1) \quad (1)$$

where p_x is the mutation rate assigned to individual x, and $Rank(x)$ is the rank (ranging from 1 to m with $m > 1$) of the group of x in the (average) fitness sorted population (by groups). The idea is to protect the top ranked groups and exploit their genotypes while the worst groups are used to perform exploration of the search space using all possible mutation rates.

In the rest of the paper we work with a rank-based mutation operator with the following parameters: $p_{\min} = 0$, $p_{\max} = 1$; $m = s$ (as in [4]). We use $p_{\min} = 0$ to force elitism (with respect to mutation), as the best individual will never mutate. Often, using elitism in the SGA can produce too little genetic diversity due to excessive convergence to the current optimum. With rank-based mutation using $p_{\max} = 1$ elitism is not a problem because there are always some individuals that use low and some that use high mutation rates, thus ensuring that part of the population is exploiting (totally and partially) the best found genotype, and part is exploring the search space. If a landscape has correlations to the optimum then it is not unreasonable to suppose that those individuals that are close to the optimum in Rank are also likely to be close to it in Hamming distance (although not necessarily). Thus, it is reasonable to perform few mutations on these individuals if one wants to follow the online signal provided by the current optimum. However, other individuals farther away in the fitness ranking will use higher mutation rates thus providing exploration further away and making it possible to escape from the currently followed landscape signal, which is desirable in the case that this signal leads to a local suboptimum.

4.2 Near-Rank Mating

When recombination is applied, selection has already acted on the population filtering the worst individuals out but maintaining some genetic diversity. So, this population contains some good and some not so good individuals. A recombination of the best individual with the worst one is very likely to destroy both rather than construct a better solution because the worst can contain genes that are quite incompatible with those of the best. The same can happen if any highly ranked individual is recombined with a low ranked one. Two good individuals containing very different genetic solutions but as good as each other have a high probability to produce better offspring when recombined in case they have found different building blocks of the global optimum. This individuals would have a high probability to be nearly ranked. Therefore one can also envision a rank-based recombination operator.

We chose a crossover operator such that only those individuals that are closely ranked in the fitness-sorted list are allowed to crossover with probability 1. We set a maximum rank difference between parents for mating as a fraction, $\alpha = 5\%$ of the number of individuals[2] (as in [4]). We also ensure that the current population best does not perform crossover in order to guarantee that the algorithm remains elitist (as in [4]).

We can anticipate that this Near-Rank mating procedure will work in the following way: In the fitness sorted list, if the landscape has some modularity, individuals will be sorted according to the number and value of the building blocks they have already found. If a bad individual eventually finds a block, it will escalate in the list and so it should have better chances of being recombined with the next "level" of individuals (with more building blocks) and, in case this discovered block is not already present at that level, it will have an enhanced probability of producing better offspring. Distinct but nearly ranked individuals will tend to have the same number of building blocks, some different and some in common. When their genomes are recombined, crossover points can be placed anywhere in the common blocks without damaging them, thus the probability of recombining the blocks that are distinct without damage increases.

The idea of recombining genotypically similar parents has been tested for instance in [17, 18], in the context of Evolutionary Multi-objective Optimization (EMOO). That work reported quicker convergence to the so called Pareto Front but also the negative effect of having low genetic diversity. To overcome this, some parameters need to be adjusted to every problem. In contrast, the method we are using, being based only on fitness rank similarity, allows for the recombination of two genotypically and/or phenotypically quite distinct individuals thus providing more genetic diversity.

Another related work is the aforementioned paper [14], where a fitness based probability of crossover as opposed to the rank-based one we propose was used. This results in a reduced probability of crossover for the best solutions available in an attempt to protect them. However, this does not recognize the possibility that recombining the very fit solutions could lead to a better one in the case that they contain equally valuable but different building blocks.

In [19] the probability of recombining similarly fit solutions is increased as the number of generations approaches a predefined number. The idea is that, after a number of generations, the best individuals are protected from being destroyed by recombinations with bad ones. One problem with this is that they give no guidance to set this predefined number of generations, so it can lead to stagnation in local optima when this number is set too low because genetic diversity is too limited, and if this number is set too high, then good solutions are not protected for too much time. So, when is it good to recombine dissimilarly fit parents? It is clear that it is sometimes good, so this must always be a possibility but also some kind of protection must be provided for fit individuals. When the fit individuals have found many building blocks of the global solution (highly fit) and there is another individual that has found only one building block (lowly fit) but that has not been found by the good ones yet, a recombination between these could produce the addition of all the found building blocks even if their fitness values are very different. In our rank-based recombination it is always possible to recombine individuals with very dissimilar fitness values while good ones are protected because most of them will be recombined with similar ones (similarly ranked does not necessarily mean similarly fit nor genetically similar but this is the most likely situation). For instance, suppose 80% of the population has the same high fitness value and the rest has the same low value. In such a situation, the method in [19] would tend to recombine only those with a similar fitness value while ours would recombine a few from both groups, thus providing more genotypically varied population but protecting most of the fit individuals.

[2]We have tested other values of α with the performance being robust to different choices.

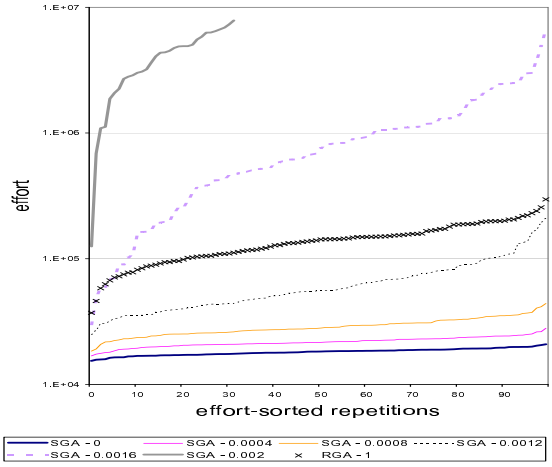

Figure 1: Run Length Distribution (RLD) curves for the Rank GA vs a "meta" SGA in the Counting Ones landscape.

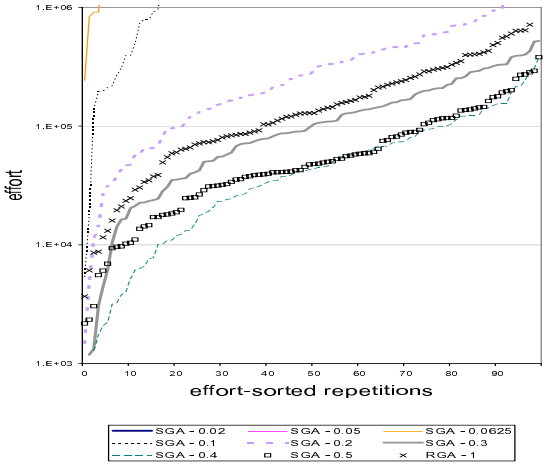

Figure 2: RLD curves for the Rank GA vs a "meta" SGA in the DTrap landscape.

4.3 Intermediate Evaluations

The above described Rank-based operators are designed under the assumption that all individuals are ranked according to their fitness value. As these operators are applied sequentially however, one must resort the population after every step. Thus, after applying recombination, the resulting population is resorted before applying mutation which implies a fitness reevaluation in case it changed.

Including this reevaluation after recombination is not so unnatural from a biological point of view if one thinks of mutation as an operator that can also act on the individuals during the period from their birth (after recombination) through to their reproductive maturity (adaptation to the environment). In this sense, Rank-Mutation affects more those individuals that inherited bad traits from their parents and less on those who inherited good ones.

5. PERFORMANCE OF THE RANK GA

In this section we present plots of the characteristic search curves by landscape type. We compare the Rank GA with a "meta" SGA. The latter is tested with several different fixed mutation rates in order to cover the best case for this parameter. The probability to perform crossover is set to 0.9 in the SGA. In both algorithms the following applies: The population size was always set to be $N + 1$; Fitness-proportional selection; 2-point crossover; Parent replacement by two complementary offspring.

We do not perform tests separately on each of the new operators because they depend on each other and there is little advantage in using them like that. The "normal" recombination operator with Rank-based mutation would simply be too destructive. Using Near-Rank recombination with a fixed (normally low) mutation rate p produces protection for the best individuals but also faster convergence which could be compensated by increasing p. This is good but needs fine tuning in each case. Besides, intermediate evaluations are only meaningful when both operators are Rank-based.

5.1 Counting Ones

This is the easiest of all landscapes since the whole landscape leads the population to the optimum. In Figure 1 we see the results with $N = 100$ and a population size of 101 individuals. Labels show Algorithm type and for the SGA also mutation rate, for the Rank GA $p_{max} = 1$ is shown. As can be seen, for the SGA the best mutation rate is 0 (circles) or at least something very close to 0. This means that the principal search operator in this case is recombination. For $p \sim 0.0012$ the corresponding SGA appears to be as good as the Rank GA (marked with Xs) in their worst case. Higher mutation rates in the SGA only make it worse. This is notable because this mutation rate is very low. The heuristic $1/N$ here gives $p \sim 0.01$, which is much higher than the plotted values, meaning that the performance of the SGA with this value must be much worse according to the tendency observed in the graph. So this results show how the Rank GA is quite well "tuned" for such an easy landscape.

5.2 Deceptive Trap - DTrap

The DTrap landscape is considered a difficult one since the whole landscape leads, by selection, the search away from the optimum. Our instance of this landscape was created from the counting ones landscape with the only difference that the string with all 0s has been assigned with a fitness value greater than that of the string with all 1s. In Figure 2 we show the Run Length Distribution curves for this landscape where $N = 16$ and population size is 17. The best case for the SGA is obtained by choosing a very high mutation rate (0.5), i.e., random search (squares at the bottom). Here we have that $1/N = 0.0625$, for which the corresponding curve of the SGA must be a little bit worse than the one for 0.05 (plotted in filled squares at the top) which is very bad compared to the Rank GA (marked with Xs). The Rank GA performs a little bit worse than random search but note that it is exactly the same algorithm used above for the counting ones landscape while the SGA had to use a radically different mutation rate in order to be "optimal".

5.3 Two Anticorrelated Signals

This landscape with $N = 20$ consists of an optimal needle corresponding to one genotype at one vertex of the hypercube, and another more robust signal consisting of the antipode of the needle and also those states at Hamming distance one from the antipode. The rest of the landscape is flat. As can be seen in Figure 3 the Rank GA (marked with Xs), performs just a little bit worse than the best case of the "meta" SGA which is the one that used a mutation rate $p = 1$, radically different than the usual values. If a typical setting of the mutation rate for the SGA or that of the

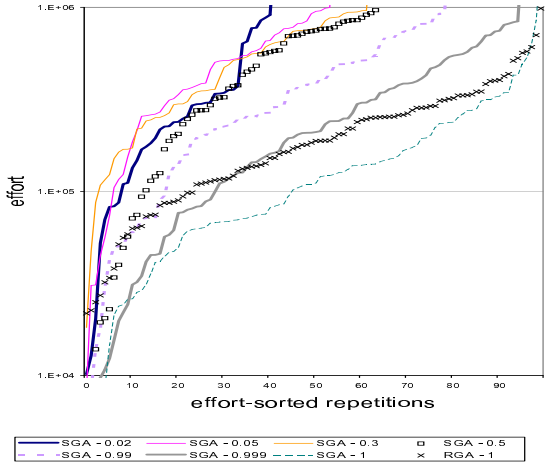

Figure 3: RLD curves for the Rank GA vs a "meta" SGA in a landscape with two anticorrelated signals.

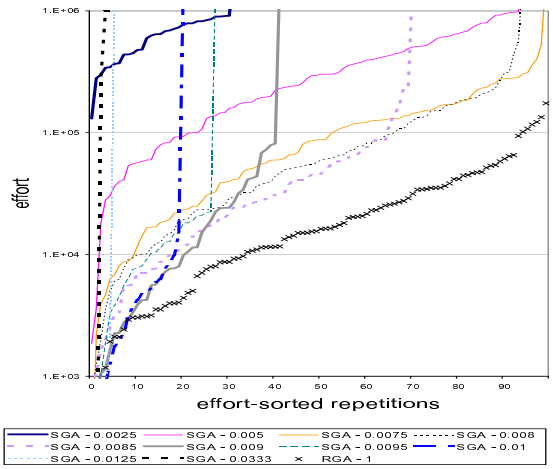

Figure 4: RLD curves for the Rank GA vs a "meta" SGA in a landscape with one suboptimal needle close to the optimal needle. The population is initialized at the subneedle.

$1/N = 0.05$ heuristic or a lower value is used, then the corresponding performance is bad. The corresponding algorithm basically never finds the optimum. The SGA can perform much better than random search but only by using a mutation rate very close to 1 which is very unusual. Interestingly, this mutation rate was not useful in the DTrap case above, although both landscapes have a deceptive signal that tends to take the population away from the optimum.

This difference with the DTrap in case of the SGA can be understood in terms of the *strengths* of the deceptive signal in the *neighborhoods* of both the optimum and its antipode. In the DTrap a mutation rate close to 1 does not help because, in that case, when the population is nearby the optimum's antipode, it is attracted (by selection) to this antipode with a strength say A and in the next generation, this population is then closer to the optimum (because p is close to 1), but now the deceptive signal repels the population away from the optimum with a strength say R. If both strengths A and R are equal then they cancel the overall effect so that the population never gets closer to the optimum's antipode and consequently neither to the optimum. If in a deceptive landscape A was weaker than R (this is the case if the deceptive signal has low gradient nearby the optimum's antipode and a high gradient nearby the optimum) the population gets gradually away from the optimum. In the landscape tested here $R = 0$ because nearby the optimum the landscape is flat and $A > 0$ because the anticorrelated signal is degenerate and individuals closer to the optimum's antipode are more selected, thus, selection attracts the population to the optimum's antipode in one generation and then, when the mutation rate close to 1 is applied, there is no repulsion from the optimum. This gets the population closer and closer to the optimum's antipode alternating in each generation with being closer to the optimum. When the optimum's antipode is reached, there is a high probability that in the next generation the optimum will be found.

The Rank GA also behaves differently than in the case of the DTrap but the explanation we found for the Rank GA is another one. In the DTrap, the rank of the individuals is always related with their Hamming distances to the optimum's antipode, while in this landscape, only a few of them have the same fitness than the optimum's antipode (2) and the rest have the same lower fitness value (1). In such a case, the rank of the good individuals ranges from 1 to some (low) value *randomly*, since any of them can be ranked as number 1. The rest of the individuals are ranked also randomly with the rest of the positions. Since selection eliminates unfit individuals replacing them with fit ones, when mutation is applied, there are more mutants that come from the fit individuals (at the suboptimal signal) which in turn happen to use low mutation rates. Thus, there are more *unfit* individuals close to the suboptimal signal than far away. The next time mutation is applied, some of these will use very high mutation rates and consequently explore the opposite side of the vicinity of the suboptimal signal which is close to the optimum. This effectively means that there are two "clouds" of individuals, one searching around the optimum's antipode and another around the optimum. This obviously increases the probability to eventually hit the optimum.

We think this landscape shows an extreme case of a non-cooperative signal and how a SGA needs a very unusual mutation rate to solve it efficiently, the usual mutation rate heuristics being very poor while the Rank GA shows that it can search at the antipode of the current best signal given that the rest of the landscape is flat.

5.4 Two Correlated Signals

Here we will see how well the algorithms perform search starting from a population that is concentrated at a suboptimal peak when the optimum is nearby. In this case we have a two needle NIAH landscape where we set the initial population to be located at a suboptimal needle with this needle twice as fit as the hay $G = 2$. The optimum was set at a Hamming distance 3 from the suboptimum. Here, $N = 30$. In Figure 4 we see the results for this case. It is clear here that the Rank GA (Xs) has an advantage over any of the SGA implementations. Random search in this case is very bad and is not plotted.

The reason for the bad performance of the SGA is that, in order to explore around the suboptimum, higher mutation rates are needed, but that proves to be too destructive and as a result the suboptimum frequently becomes extinct, thus leaving the population to be driven only by genetic drift since the rest of the landscape is flat and there is no signal in the population to direct the population (and search) close to the optimum. The Rank GA has no problem in keeping the suboptimum always present while exploring both close and

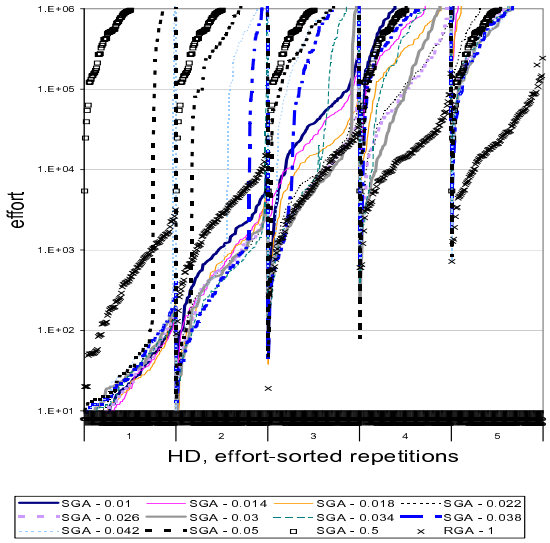

Figure 5: RLD curves for the Rank GA vs a "meta" SGA in a landscape with one suboptimal needle at different hamming distances from the optimal needle. The population is initialized at the subneedle.

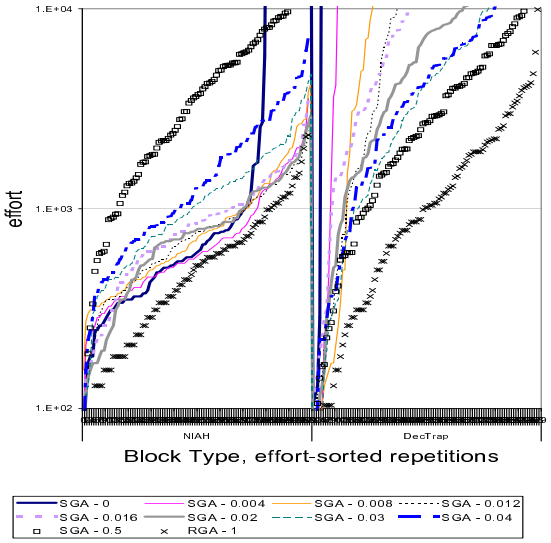

Figure 6: RLD curves for the Rank GA vs a "meta" SGA in the Concatenated Blocks landscape. NIAH blocks (left), DTrap blocks (right).

far away from it, irrespective of whether the suboptimum is very robust or not.

As a function of the Hamming distance of the optimum from the suboptimal peak, one can see in Figure 5 (made with $N = 20, G = 4$) that for low Hamming distances the SGA has better performance than the Rank GA. Nevertheless, those are the easiest cases and the Rank GA can solve them easily too. The advantage of the Rank GA becomes evident when the SGA cannot reach the optimum with low mutation rates and thus has to use higher mutation rates than the current optimum's critical mutation rate, with the consequent loss of the current signal in the population which then becomes just random search. The Rank GA, in contrast, is able to perform search locally and in a wider "radius" without losing the current optimum's signal, thereby increasing the chances of escaping from the local suboptimum and finding a nearby optimum. But, how much of this can be achieved using simple elitism in the SGA? In this case the population gets more concentrated around the current best, which is in fact a negative trait since exploration is limited. If then higher mutation rates are used, local search is affected.

5.5 Concatenated Blocks

In this case we test the ability of the algorithms to keep and exploit partial solutions of the landscape while at the same time keep on exploring the search space. We show in Figure 6, results for 3 concatenated 4-bit NIAH landscapes (left) and 3 concatenated 4-bit DTrap landscapes (right). In both cases the best results are those of the Rank GA (Xs). For the NIAH blocks (left), the SGA always performs better than random search (squares), the best mutation rate value being very low but not 0. Recombination is in this case the principal search operator. Since both algorithms perform better than random search, one can say that they are able to exploit the modularity of the landscape by "completing" the optimum in one of the blocks and exploiting it while exploring for the optimum in the other blocks. Here $p = 1/N = 0.0833$ is a little bit off the range we used in the plot but one can see that its curve would be between the one for $p = 0.5$ and $p = 0.04$ making it evident that it is far from optimal. On the other hand, for the Dec-Trap blocks landscape (right) we see a very different result for the SGA. Now all the SGA cases are worse than random search as high mutation rates are required in order to have sufficient exploration so that the optimum of a block may be found, while, at the same time, low mutation rates are needed in order to keep the optimum of a block while the other ones are being discovered. This is clearly contradictory and cannot be solved by the SGA. Note that the curve for $p = 1/N$ is very close to random search. On the other hand, in both cases the modularity of the landscape has strongly favoured the Rank GA (compared with the cases where there is only one big block) because it is able to keep any string with at least one optimal block while at the same time keeps on exploring. Then, when two individuals have found different optimal blocks, the algorithm via recombination manages to build a solution with more optimal blocks from these two and keep it while waiting for further improvements through recombinations with individuals that find the other optimal blocks. In this way the problem is effectively split into 3 smaller (4-bit) DTrap problems. This case is a very good example of a punctuated equilibrium that is very common in real world landscapes.

6. CONCLUSIONS

We have seen that the optimal mutation rate for the SGA depends on several features of the fitness landscape as well as the population state. This leads to the conclusion that if a problem is to be solved efficiently then a suitable mutation rate needs to be found for each problem instance and even at different stages of the same run of the algorithm. Consequently, the SGA with a fixed mutation rate is not such a general-purpose algorithm. When it is well tuned for some situations it is necessarily badly tuned for others. Universal heuristics such as $1/N$ or the error threshold in this sense are compromises trying to make the best of a bad job. Rank-based GAs offer a very simple to implement robust alternative to the SGA, offering an adequate dynamic balance

between exploration and exploitation for commonly found situations in search. They are also an interesting alternative to using a more computationally expensive self-adaptive algorithm that evolves the mutation rate. They need no tuning in the mutation rate when a single run transits from one regime to another (punctuated equilibria) wherein different mutation rates are required, not to mention when the landscape itself is time dependent. The chances of stagnating are minimal.

One could think of the Rank GAs presented here as models of societies, in which every individual (or group) is obsessed with being more successful than any other. To do this they are informed of their relative success (fitness) at any time. The top ranked individuals tend to stay unchanged while others tend to try different strategies (mutate) providing a chance to escape from local optima. At the same time, mating is performed only between similarly ranked individuals thus protecting successful genotypes while trying new combinations of their elements within a restricted scope which provides the possibility to solve modular landscapes. This mechanism has a positive effect in "black box" search spaces because it performs search in the near and far neighborhood of the current best individual, without loosing it, in two ways: by mutations and by recombinations.

We saw, particularly that the Rank GA offers several advantages when considering landscapes that have correlations (nearby peaks leading to the optimum) and/or modularity (with difficult to find block optima) - both of these being commonplace in real world problems. The Rank GA is able to follow every correlation present in the peaks of the landscape that lead to higher (better) ones regardless of their width or degeneracy (if two signals are equally fit, the more robust is followed) and having as worst case something very similar to random search (while a SGA performs much worse than random search). So if this algorithm performs better than random search it is an indicative that there are useful correlations in the landscape and viceversa. This algorithm can be useful, for instance, as an evaluation tool for different representations of the same problem. A good representation is one that contains the best "royal" road to the optimum by means of signals that attract the population faster to it. After evaluating many different representations with the Rank GA, one could make an analysis of what are the common traits present in those representations that were good and try to extract knowledge from that.

This kind of differential evolution could also, in principle, be applied to other adaptive systems such as game agents or evolving neural networks for instance.

7. REFERENCES

[1] J.J. Grefenstette, Optimisation of Control Parameters for Genetic Algorithms, IEEE Trans SMC, 16(1), 122-128 (1986).

[2] T. Bäck, Optimal Mutation Rates in Genetic Search. In Proceedings of ICGA 5, ed. S. Forrest, 2-8, Morgan Kaufmann (1993).

[3] J. Cervantes, C. R. Stephens, "Optimal" mutation rates for genetic search. Proceedings of Genetic and Evolutionary Computation Conference (GECCO) 06, pp 1313-1320, Maarten Keijzer et. al. (2006).

[4] J. Cervantes, C. R. Stephens, A Rank Proportional Generic Genetic Algorithm. Lecture Series on Computer and Computational Sciences, Volume 7, 2006, pp 71-74. Brill Academic Publishers.

[5] H. Mühlenbein, How Genetic Algorithms Really Work: Mutation and Hill Climbing, PPSN II, ed.s B. Männer and R. Manderick, 15-25, North Holland (1992).

[6] T. Bäck, Self-adaptation in Genetic Algorithms, In Proceedings of 1st European Conference on Artificial Life, 263-271 MIT Press (1991).

[7] J. Hesser and R. Männer, Towards an Optimal Mutation Probability for Genetic Algorithms, LNCS 496, ed. H.P. Schwefel, Springer Verlag (1991).

[8] T.C. Fogarty, Varying the Probability of Mutation in the Genetic Algorithm, ICGA 3, ed. J.D. Schaffer, 104-109, Morgan-Kaufmann (1989).

[9] C.R. Stephens, I. Garcia Olmedo, J. Mora Vargas and H. Waelbroeck, Self-Adaptation in Evolving Systems, Artificial Life, Vol. 4, Issue 2, 183-201 (1998).

[10] J.E. Smith and T.C. Fogarty, Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm, Proceedings of CEC, 318 - 323, IEEE Press (1996).

[11] G. Ochoa, I. Harvey and H. Buxton, Error Thresholds and their Relation to Optimal Mutation Rates. In D. Floreano, J-D. Nicoud, and F. Mondada (Eds.) ECAL'99, Lecture Notes in Artificial Intelligence 1674, pp 54-63, Springer-Verlag, Berlin (1999).

[12] G. Ochoa, Setting the Mutation Rate: Scope and Limitations of the 1/L Heuristics, GECCO-2002, pp 315-322, Morgan Kaufmann, San Francisco, CA (2002).

[13] G. Ochoa, Error Thresholds in Genetic Algorithms. Evolutionary Computation, MIT Press (in press).

[14] M. Srinivas, Lait M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Transactions on Systems, Man and Cybernetics, 24(4):656-667, April 1994.

[15] M. Sewell, J. Samarabandu, R. Rodrigo, and K. McIsaac. The Rank-scaled Mutation Rate for Genetic Algorithms. International Journal of Information Technology, Volume 3 Number 1, 2006.

[16] H. Hoos, T. Stützle. Evaluating Las Vegas Algorithms: Pitfalls and Remedies. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, UAI-98 pages 238-245, Morgan Kaufmann Publishers, San Francisco CA, 1998.

[17] H. Ishibuchi and Y. Shibata. A Similarity-Based Mating Scheme for Evolutionary Multiobjective Optimization. E.Cantú-Paz et al.(Eds.):GECCO 2003, LNCS 2723, pp.1065 1076,2003. Copyright Springer Verlag Berlin Heidelberg 2003.

[18] H. Ishibuchi and K. Narukawa. Recombination of similar parents in EMO algorithms. Lecture Notes in Computer Science 3410:Evolutionary Multi-Criterion Optimization, pp.265-279, Springer, Berlin, March 2005.

[19] G. Chakraborty, K. Hoshi. Rank Based Crossover - A new technique to improve the speed and Quality of convergence in GA. Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99.