Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm

Jim Smith
Faculty of Computer Studies & Mathematics
University of West of England
Bristol Bs16 1QY, U.K.
jim@btc.uwe.ac.uk

T.C.Fogarty
Faculty of Computer Studies & Mathematics
University of West of England
Bristol Bs16 1QY, U.K.
tcf@btc.uwe.ac.uk

1: Abstract

This paper investigates the use of genetically encoded mutation rates within a "steady state" genetic algorithm in order to provide a self-adapting mutation mechanism for incremental evolution.

One of the outcomes of this work will be a reduction in the number of parameters required to be set by the operator, thus facilitating the transfer of evolutionary computing techniques into an industrial setting.

The NK family of landscapes is used to provide a variety of different problems with known statistical features in order to examine the effects of changing various parameters on the performance of the search. A number of policies are considered for the replacement of members of the population with newly created individuals and recombination of material between parents, and a number of methods of encoding for mutation rate are investigated.

Empirical comparisons (using the "best-of current-population" metric) over a range of test problems show that a genetic algorithm incorporating the best "flavour" of the adaptive mutation operator outperformed the same algorithm when using any one of a variety of "standard" fixed mutation rates suggested by other authors.

2: Introduction

Mutation has long been regarded as a vital ingredient in evolutionary algorithms, and some paradigms e.g. Evolutionary Strategies [Rechenberg 1973], [Schwefel 1981] use it as their principal search mechanism. Within the field of Genetic Algorithms (GA's) [Holland 1975] there has been much work, both practical e.g. [Schaffer et al. 1989] and theoretical e.g. [Spears 1992] on the relative merits of mutation as a search mechanism. Much of the work has been concerned with finding suitable values for the rate of mutation to apply as a global constant during the search. There is, however, an increasing body of evidence, both empirical e.g. [Fogarty 1989] for learning control rules, and theoretical e.g. [Bäck 1992] that the optimal rate of mutation is not only different for every problem encoding but will vary with evolutionary time according to the state of the search and the nature of the landscape being searched.

This issue has been tackled successfully within Evolutionary Strategies by encoding the mutation step applied within the representation of each solution. This approach also means that the mutation rate is now governed by a distributed rather than a global rule (see [Hoffmeister & Bäck, 1991] for a good overview of the issues tackled and approaches taken). These ideas have been applied to a generational GA by adding a further 20 bits to the problem genotype, which were used to encode for the mutation rate [Bäck 1991]. The results showed that in a generational setting the mechanism proved competitive with a genetic algorithm using a fixed (optimal) mutation rate provided that a high selection pressure was maintained (this is referred to as "extinctive" selection).

In this paper an investigation is made of the issues confronted when this paradigm is implemented within the setting of a "steady state" genetic algorithm [Whitley & Kauth1988] where the methods used in Evolutionary Strategies for managing the population are not suitable. The class of landscapes chosen to investigate the behaviour of the algorithm is the well-studied NK family of landscapes which provide a tunable set of landscapes with increasing numbers of optima and decreasing fitness correlations as K is increased. The reader is referred to [Kauffman 1993] for a full description. Since these landscapes are randomly created, the fitness of the global optimum of each landscape is not known, and so the measure chosen to study the performance of the algorithm in various flavours is the current best in a population averaged over a number of runs (each with a different landscape).

3: Implementation Details

The steady state genetic algorithm is different to the generational models used in the work discussed above in that there is typically a single new member inserted to the population at any one time, which according to the standard nomenclature would correspond to a (μ+1) strategy. However it has been shown empirically (and theoretically for certain simple problems [Schwefel 1981]) that there is an optimal "acceptance" ratio of approximately 1:5. i.e. one individual should be incorporated into the population for every five created. This can be explained by considering that although repeated mutation of a single individual corresponds to a form of local search, which can be shown to enhance the performance of Genetic Algorithms e.g. [Bull & Fogarty 1994], there is a trade off between this local search and global search (either via recombination or simply by considering other members of the population)

which will affect both the convergence velocity and (possibly) the quality of the final solution.

The approach taken is to create a single individual via recombination from two parents and then "clone" that individual a number of times. Each of these offspring then undergoes the mutation process and is evaluated before one is selected according to some policy and inserted into the population according to the deletion strategy used. Effectively this inner loop can be thought of as a (1, CLUTCHSIZE) generational GA.

The algorithm used can be summarised as follows.

WHILE (evals_done < MAX_EVALS)
{ Select two parents
Create a new individual via recombination.
Repeat (CLUTCHSIZE) times
{ Copy new individual
Read genetically encoded Mutation rate
Use this value to mutate the mut.encoding
Get new mutation rate.
Apply mutation problem encoding.
Evaluate this individual }
Select one individual from the offspring.
Replace one individual from the population
evals_done = evals_done + CLUTCHSIZE }

It can be seen from the above that the following parameters will affect the algorithm:

1) ***Deletion Policy***: Two standard policies are frequently used with steady state GA's, namely Delete-Worst and Delete-Oldest. In addition there is the issue of whether a member of the population should be replaced only if it is less fit than the offspring which would replace it (Conditional Replacement) or always (Unconditional Replacement).

2) ***Selection Policy***: Parental selection is Fitness Proportionate ("Roulette Wheel"). Selection of an individual from the "clutch" to enter the population can use the same mechanism or be deterministic, i.e. the best offspring is always picked.

3) ***Recombination Policy***: What type of crossover is used, and should the mutation encoding be subject to crossover?

4) ***Mutation Encoding***: The genotypically encoded mutation rate must be decoded and then scaled onto the range 0-100%. Three types of encoding are used, namely binary, gray and exponential. In the latter (suggested by [Kaneko & Ikegami 1992]) the mutation encoding is first binary decoded to give a value *j* and then the mutation rate μ is given by μ_j = maxposs * $2^{(j - jmax)}$ (where *jmax* is the largest number allowed by the binary encoding).

5) ***Clutchsize***: i.e. the number of offspring cloned in a single iteration of the algorithm.This will affect the balance between local search and global search in the early stages of the run before the population has converged.

In order to test the effects of the above factors a set of "standard" values had to be adopted which could be kept constant during the testing of other factors. Since empirically derived "standards" (e.g. the 1/5 rule) existed for some of the factors, and previous work in a "Generational setting" [Bäck 1991] had shown that the selection pressure was probably the most important factor, experiments were run in the order shown above, with the optimal (or most robust) combination of parameters from one set of experiments being carried through to the next set.

All experiments were run using a population of 100 on 16 bit NK (i.e. N = 16) landscapes with values of K of 0, 4, 8 and 15 to represent a spread of landscapes with increasing complexity from a simple uni-modal hill (K = 0) to a randomly coupled landscape (K = 15). All experiments were averaged over fifty runs, each run being on a different landscape. For equivalence the same fifty seeds and landscapes were used with each of the alternatives under comparison.

4: Results

4.1: Selection/Deletion policies

A variety of policies for determining the insertion of new individuals into the population were tested, using one point crossover (at 70% probability), 16 bit gray encoding for the mutation rates and a "clutch" size of 5.

The results of the more successful methods are shown in Figure 1. Not shown are the curves for the combination of a delete-oldest policy with unconditional acceptance of an offspring, which showed very erratic behaviour and failed to reach optima on any but the simplest problem (K = 0). The following observations can be drawn from the plots:

1The combination of "delete-worst" with deterministic offspring selection (WC-B / WU-Best) performs well on the simpler problems, and there seems to be little effect whether the offspring is accepted conditionally or unconditionally.

2 The combination of "delete-worst" with stochastic offspring selection (WC-FP) performs less well than the deterministic selection policy. The use of stochastic offspring selection will reduce the selection pressure in the algorithm, which explains the relatively slower growth curves, but noticeably the runs also converge to lower optima as the complexity of the search space increases (i.e. as K increases), which suggest that although the rate of approaching optima is decreased this is not counter balanced by an increase in diversity which might have lead to a reduced likelihood of becoming trapped in sub-optima.

3The use of a "delete-oldest" policy is only successful if the insertion of an individual is conditional on its being better than the member it replaces. Even with the conditional acceptance policy, the overall selection pressure in the algorithm is much less than for the "delete-worst" policy, and this is reflected in reduced growth curves. The relaxation of the selection pressure also highlights the difference between the stochastic (OC-FP) and deterministic (OC-Best) selection policies, with the former showing very slow improvements,

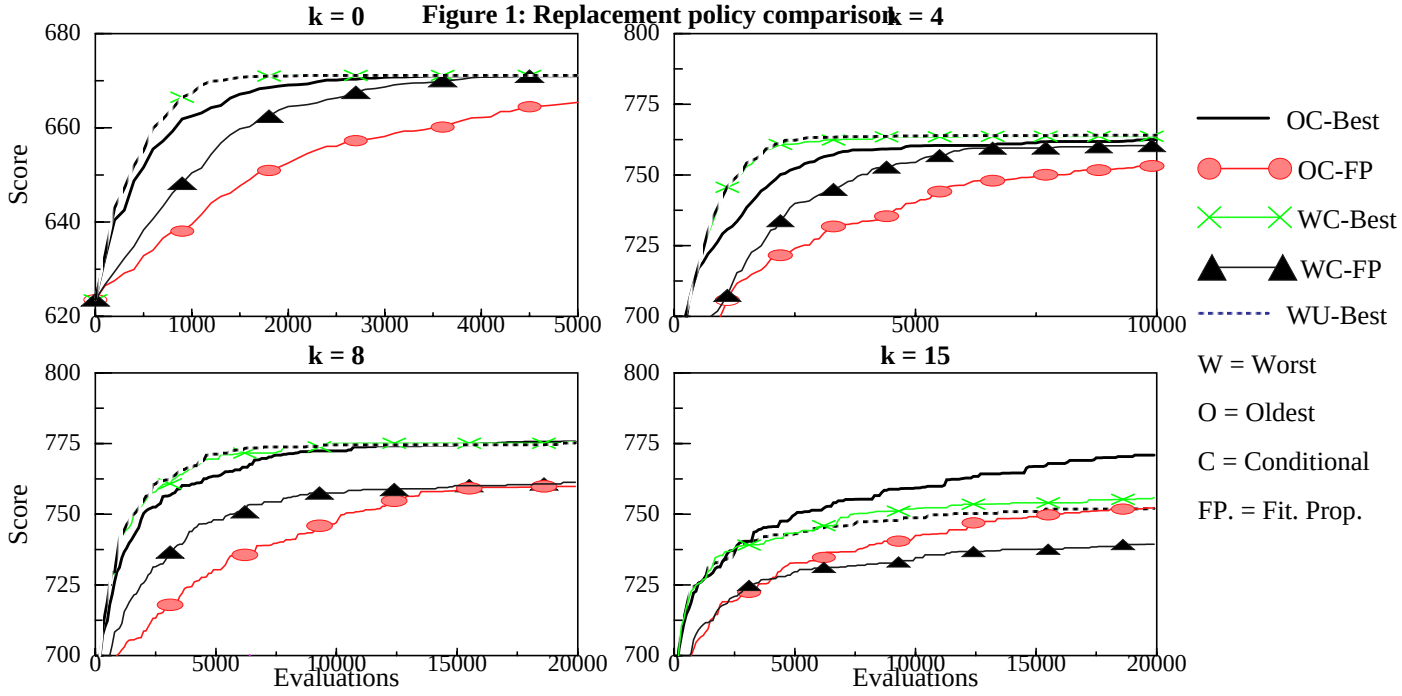

Figure 1: Replacement policy comparison

although the searches do not appear to "stagnate" on the more complex problems as the algorithms with "delete-worst" policies do.

4 Overall the best policy would appear to be the replacement of the oldest of the population with the fittest of the offspring, conditional on the latter being the fitter of the two (OC-B). This policy shows growth curves comparable with the other policies on the simpler problems (K = 0,4,8), but on the most complex problem it significantly outperforms all others, reaching much higher optima.

4.2: Recombination Policy

The "standard" algorithm above, (with OC-Best replacement) was run with a variety of recombination policies, namely every combination of 1point/uniform crossover, applied at 70% / 100% to the whole chromosome / the genome only. The results obtained showed very slight differences in performance compared to the large differences found above, and no significant pattern emerged, suggesting that either the algorithm is highly tolerant of crossover mechanism, or that the nature of the selection pressure is such that the algorithm proceeds via mutation - based search. In order to test this a set of experiments was run using no crossover. These results showed that for the simple problems the growth curves were very similar, but for the more complex curves e.g K = 8, the performance was significantly degraded if crossover was not employed, indicating that although the type of crossover used is not important, it does have a value in enabling the population to reach different optima.

3.3. Mutation Encoding

Of the three different encodings investigated, the Gray coding and Binary encoding showed similar performance, both substantially outperforming the "exponential" encoding. The use of 16 bits as opposed to 8 for the representation provides a slight advantage which is more noticeable on the more rugged landscapes, but the difference is small - for K = 15, the mean maximum values reached are shown in the table below

Mutation Encoding	Mean fitness	
	8 Bit	16 Bit
Binary	767.82	770.08
Gray	767.74	770.92
Exp.	735.54	736.26

Also tested was the importance of the maximum value which the mutation rate can take once decoded.

Overall there is a trade-off between retaining the ability to learn large mutation rates which enable faster search on less correlated landscapes, and the time taken to "learn" to low mutation rates on smooth landscapes. However these effects are small compared to those of selection and "clutchsize".

.It would appear from these results that if nothing is known about the landscape to which the mechanism is to be applied, initialising the population at random over the range of 0-25% represents a good compromise.

3.4. Clutchsize

A full investigation of the effects of changing the number of individuals cloned from which one is selected for inclusion into the population appeared to offer strong empirical evidence

for the 1:5 “acceptance ratio” referred to above. A single landscape and fifty populations were created for each value of K. Experiments were run using “clutch” sizes of 1,2 5 and 10, and on all but the simple problem (where all variants reached the same mean optimum) the mean fitness achieved at the end of the run by the variant with 5 offspring was higher than the mean fitness reached by all others. These results are summarised in the table below. Starred values are significantly different using Student’s t-test at the 5% confidence level

Number of Offspring	Mean Maximum Fitness		
	K = 4	K = 8	K = 15
1	791.48*	755.28*	751.18*
2	796.12*	754.84*	763.14
5	798.42	763.6	764.8
10	799.26	767.2	766.68

Although the results in this format do not show a significant difference between the performance as the amount of local search is doubled from 5 to 10 offspring, this can be demonstrated by considering the time taken to achieve the optimum for the simple problem, K = 0.

This was a mean of 5800 evaluations with 5 offspring and 9900 with 10. This near doubling of time to converge is also reflected in the time taken to reach lesser values such as 99% of the maximum (1300 evaluations vs. 2000) and 95% of the optimum (200 evaluations vs. 400)

4: Comparison with Standard Fixed Mutation Rates

The optimal set of parameters and policies identified above were tested against a steady state genetic algorithm using a “normal” mutation mechanism with the same recombination parameters, population size and parent selection mechanism as the adaptive algorithm. The replacement policy was also to delete the oldest member of the population if its fitness was worse than that of the new individual. Again given that the value of the global optimum was unknown (and different) for every problem, the algorithms were measured in terms of the performance of the best of the current population, tested every 200 evaluations. A variety of authors have attempted to determine fixed values for the mutation rate which will yield good results across a range of problems. A number of these common settings were tested including p_m = 0.001 [DeJong 1975], p_m = 0.01 [Grefenstette 1986], $p_m = 1/l$ (where l is the length of the string) and $p_m = 1.75 / (l * \text{popsize})$ (this comes from [Bäck 1991] as an empirical formulation of a result from [Schaffer et al. 1989]). These experiments were run 50 times and the mean results are shown in Figure 2. Also shown in this figure are the results for the adaptive mutation mechanism with a single offspring for comparison.

As can be seen, for the simplest problem the added overhead of learning mutation rates slows down the rate of evolution to the optimum, however the mean best value attained is the same for this algorithm as for the fixed rate flavours. Comparison of the curves for the adaptive mechanism with a single offspring suggests that most of the degradation in speed is due to the overheads of the local search, possibly because only a single new offspring is inserted into the population from each clutch

As the amount of epistasis in the problem increases the adaptive policy shows a markedly improved ability to continue to improve the fitness of the population compared with the fixed rate versions. This is true for both sizes of offspring although the single offspring is much slower to learn as noted above. The fixed mutation rate algorithms frequently show curves which either stagnate or show very slow improvements, suggesting that on most if not all of the 50 runs the search had become trapped in a local optimum. In contrast the adaptive mutation algorithms always shows continuing improvement. One reason for this may be that whether the encoding is binary or Gray coded there is always a chance that mutation of a single bit in a low mutation rate encoding will create an individual with a high mutation rate, providing a means of escaping local optima.

After noting that the version of the algorithm with 5 offspring is noticably better on the more complex problems than the single offspring version, further experiments were run in order to determine whether the improvements noted above were the result of adaptive mutation rates or simply the result of adding a form of local search to the GA via the use of a “clutch” of offspring. These experiments used the same suite of fixed mutation rates as above, but this time in exactly the same algorithm as the adaptive mechanism. The results are shown in Figure 3.

As can be seen the fixed rate algorithms are all improved by the addition of local search in that the problem of premature convergence to a local optimum is ameliorated (to some extent), and the setting of $p_m = 1/l$ is more competitive, but the adaptive mechanism still discovers higher optima in both of the more complex landscape families (K = 8 and K = 15).

In order to test the statistical significance of these results, as further set of experiments were run. For each value of K in the range (4 8 15) a landscape and fifty random populations were created. Each algorithm in turn was allowed to run for 20,000 evaluations, and the best value in the final population noted. The adaptive algorithm had the highest mean value on all three landscapes.

Using these fifty samples of each algorithm’s performance, Student’s t-test at the 5% confidence level was used to test the significance of the differences in performance. This showed that the adaptive algorithm was significantly better than all of the fixed algorithms other than that using p_m = 1.75

/ (l * popsize) on the landscapes with K =8 and K = 15. ForK = 4, the results with $p_m = 1/l$ were also not significantly worse than those of the adaptive algorithm.

These results confirm the trends shown in figure 3 although the order of the fixed algorithms changes slightly. One explanation for this could be that the significance tests were conducted using a single landscape for each value of K which may have been more suited to certain values of p_m. In fact it could well be argued that the changes in relative fortunes of the fixed rate algorithms demonstrates the "brittleness" of the approach of using fixed rate operators which are unable to adapt to their environment.

Finally it should be noted that testing binary vs. gray coding for the adaptive mutation rate showed that with K = 15, the gray coding was significantly better using this metric.

5: Conclusions

A mechanism has been presented which allows the incorporation within the "Steady State" genetic algorithm of a Self-Adaptive mutation rate. This mechanism also incorporates some local search as a by-product of the means whereby it "learns" good mutation rates. The nature of the mechanism is such that it provides a form of distributed control for the genetic algorithm whereby individuals in different parts of the search space may have different mutation rates.

Experiments with a variety of parameters have shown that the mechanism is robust in the face of major changes such as choice of crossover, mutation rate encoding, etc. The most sensitive choice appears to be that of which member of the population to replace, and with which offspring. The combination of a "Delete-Worst" policy with selection of the fittest offspring works well on all but the most complex problems, but overall the best performance came from using a "Delete-oldest" policy which also has the advantage of being computationally simpler as it does not require the re-ordering of the population after every insertion. Further investigation is needed on the sensitivity of the mechanism to the parental selection technique.

On the most complex, uncorrelated landscapes, gray coding worked significantly better than binary coding. This is not surprising as Gray coding provides a much smoother landscape for the system to learn mutation rates, and as the landscapes become less correlated, so mutation becomes more important to the search process.

This reason is why differences in performance generally become more noticable at higher values of K.

The comparisons with GA's using fixed rate operators with or without local learning showed two results. Firstly, for all non-trivial problems the inclusion of a form of local search can improve performance. Secondly the addition of the adaptive mutation mechanism significantly improves the performance of the Genetic Algorithm, as well as making it more suitable for application in industry via the removal of a parameter from the set of decisions faced by the operator.

6: References

Bäck T. 1991 *"Self Adaptation in Genetic Algorithms"* in "Towards a Practice on Autonomous Systems" ed. Varela & Bourgine (MIT Press 1992) pp 263-271

Bäck T. 1992 *"The Interaction of Mutation Rate, Selection & Self-Adaptation Within a Genetic Algorithm"* in "Parallel Problem Solving from Nature @" eds. Manner & Manderick (Elsevier Science) pp 85-94.

Bull L., & Fogarty T.C. 1994 *"An Evolutionary Strategy & Genetic Algorithm Hybrid: An Initial Implementation & First Results"* in "Evolutionary Computing ed. T.C.Fogarty (Springer Verlag) pp 95-102

Fogarty T.C. 1989 *"Varying the Probability of Mutation in the Genetic Algorithm"* in "Proceedings of 3rd International Conference on Genetic Algorithms", ed Schaffer (Morgan Kaufmann) pp 104-109.

Grefenstette J.J. (1986). *"Optimisation of control parameters for genetic algorithms"*. pp 122-128, IEEE Transactions on Systems, Man & Cybernetics, SMC-16(1)

Hoffmeister F., & Bäck T. 1991 *"Genetic Self Learning"* in "Towards a Practice on Autonomous Systems" ed. Varela & Bourgine (MIT Press 1992) pp 227-235

Holland, J. 1975 *"Adaptation in Natural & Artificial Systems"* (University of Michigan Press, Ann Arbor)

DeJong, K. (1975) *"An Analysis of the behaviour of a class of genetic adaptive systems"*. PhD. thesis, University of Michigan.

Kaneko, K. & Ikegami T. 1992. "*Homeochaos: dynamic stability of a symbiotic network with population dynamics and evolving mutation rates"* pp 406-429, Physica-D 56. North Holland

Kauffman S. 1993 *"The Origins of Order - Self Organisation & Selection in Evolution"* (Oxford University Press)

Rechenberg, I. 1973 *"Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution" (Frommann*-Holzboog Verlag, Stuttgart).

Schaffer J.D., Caruana R., Eshelman L.J., Das R. 1989 *"A Study of Control Parameters Affecting On-line Performance of Genetic Algorithms for Function Optimisation"*. in "Proceedings of 3rd International Conference on Genetic Algorithms", ed Schaffer (Morgan Kaufmann).pp 51-60

Schwefel, H-P. 1981 *"Numerical Optimization of Computer Models"* (Wiley, Chichester).

Spears W. 1992 *"Crossover or Mutation"* in "Foundations of Genetic Algorithms 2", ed.Whitley (Morgan Kaufmann)

Whitley D. & Kauth J. 1988 *".GENITOR: A different Genetic Algorithm"* in Proc. of the Rocky Mountain Conf. on Artificial Intelligence" Denver C*O, pp118-130*

Figure 2: Comparison of Adaptive GA with "Standard GA's".

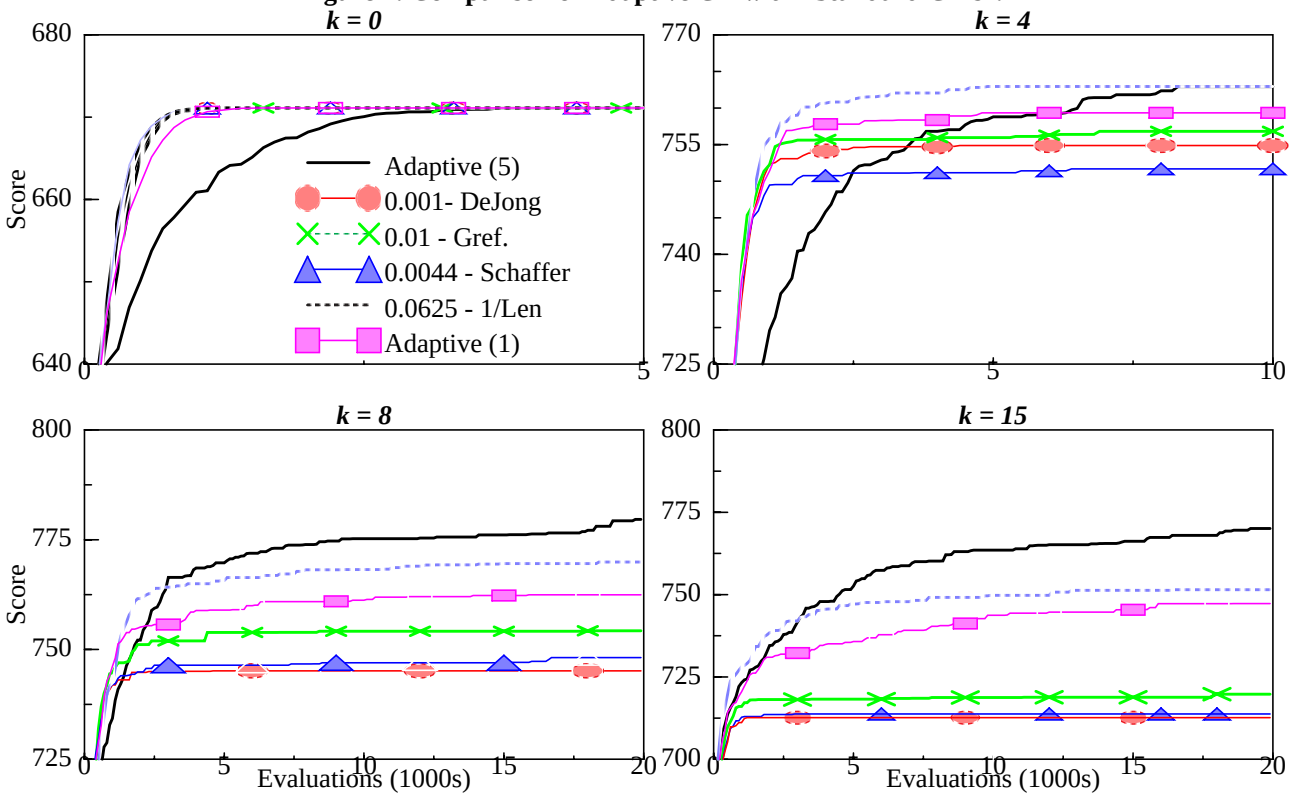

Figure 3: Comparison of Adaptive GA with "Hybrid" GA's.