Evolutionary Algorithm With A Novel Insertion Operator For Optimising Noisy Functions

Evan J. Hughes
Department of Aerospace, Power and Sensors,
Cranfield University,
Royal Military College of Science,
Shrivenham, England.
ejhughes@iee.org

Abstract- As more complex engineering optimisation problems are being tackled, optimisation algorithms are being stretched to their limits. The simulations are often subject to noise and uncertainties, leading to noisy objective functions and variable evaluation times.

With a noisy objective, it is often very difficult to identify the best solutions reliably as the noise can cause even an optimum value to appear 'good' rather than 'excellent'. Bad solutions are often much easier to spot as it takes a lot of noise to make them appear 'good'. Thus in a noisy environment, a strategy of *replacing* bad solutions may have an advantage over *selecting* excellent ones.

A new insertion process has been developed that allows the insertion strategy to be tuned smoothly between 'greedy' fitness based insertion and uniform random insertion. This new insertion function allows both the evolutionary processes of selection and insertion to be adjusted to suit the level of noise and complexity in the objective calculations.

This paper demonstrates that the insertion and selection operators can be tuned to suit the level of noise in the objective to maintain maximum algorithm efficiency and solution accuracy. Experiments have shown that for some noisy problems, the insertion process must dominate the selection operator for maximum efficiency, but the selection process has a significant effect on the accuracy of the final solutions.

1 Introduction

Simulations of real engineering systems are often subject to parameter uncertainties and noise. Detailed simulations are also often complex and therefore slow to evaluate. Optimising these systems is slow and difficult because of the noisy objective functions. Noise may also be added in order to identify robust solutions to the problems [1].

Many algorithms re-insert new chromosomes into the population using some form of 'greedy' operator where the worst individual in the population is replaced. This operator tends to suffer with a noisy objective as the population saturates with 'lucky' chromosomes where the noise has been favourable in the objective evaluation [2]. In dynamic situations such as on-line optimisation, a strategy that replaces the oldest individual is used [2].

Simulations of real systems often require functions such as variable step integrators etc. The time required to evaluate the objective function is often dependent on uncertain parameters and the trial solution being evaluated. Parallel steady-state evolutionary algorithms have been shown to be robust to out-of-order execution of individuals [3] and do not require any synchronisation processes (see section 2.1).

This paper introduces a new probabilistic insertion operator that allows the evolutionary effects of insertion to be controlled. The results demonstrate that in the presence of noise, the 'greedy' insertion process leads to a less efficient algorithm, and the evolutionary effects of selection need to be reduced in order to improve algorithm efficiency.

2 Algorithm

2.1 Algorithm Structure

Figure 1 shows the basic structure of the steady-state evolutionary algorithm. A population of p individuals is maintained by the algorithm. Each time a processor finishes evaluating an individual, the individual is inserted into the population using the method described in section 2.2. A set of parents are then chosen from the population and used to generate a new chromosome for evaluation. This child is then passed to the next processor that becomes available. If there is no chromosome waiting to be processed when a processor becomes available (as at startup), a random chromosome is generated for evaluation. The generation of child chromosomes is delayed until a population of p individuals has been collected. This allows for sufficient random chromosomes to be generated at startup.

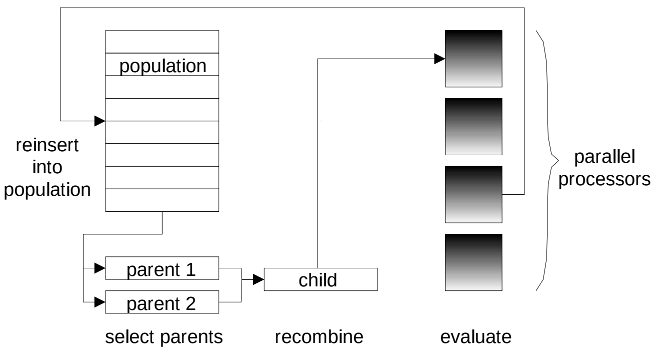

Figure 1: Asynchronous Parallel EA structure.

If k processors are used, a minimum of $p + k - 1$ and maximum of $p + k$ random individuals will be generated, depending on the time taken to evaluate each chromosome and the process scheduling. As the algorithm is robust to out-of-order chromosome insertion [3], the approach is ideally suited to situations where a network of machines with different processing powers is available, or the objective function gives wide differences in evaluation time dependent on the chromosome and the values of noisy parameters. The algorithm will also tolerate processors joining (a new random chromosome would be created) and leaving (a lost evaluation) the pool of available processing resources at a slow rate.

2.2 Reinsertion and the Probabilistic Cut Operator

Various reinsertion strategies have been developed, but one of the most often used is replacing the worst individual in the current population. This is simple to implement and provides an added evolutionary process alongside selection for breeding. With a noisy objective, the algorithm is 'greedy' and focuses attention on solutions that have been artificially improved by the noise. If a 'true' fitness based insertion process is adopted where the worst individual in the current population is only replaced if the new solution is better, the maximum evolutionary benefit is obtained. Unfortunately, if noise is present, the 'true' fitness based insertion often performs poorly. Uniform random insertion provides no evolutionary drive and allows even the best solution found in the population to be replaced. This is sometimes referred to as a *total replacement* strategy.

Alternatively, instead of viewing the insertion process as the new individual directly replacing an individual of the population, we may insert the new solution into the (sorted) list of individuals and then cut an individual from the list to maintain a population of p.

Figure 2 shows a plot of the probability density function for a probabilistic cutting function. The figure corresponds to equation 1. The function uses the *cut pressure* α to determine the shape of the probability density function of the cutting process for individual i. The cut pressure alters the evolutionary effect of the insertion process in a manner analogous to selective pressure in the selection process (section 2.3). This method of generating a selective pressure is effectively a subset of the cut pressure equation proposed here (see section 2.3). The cut pressure equation has been enhanced with respect to the selection equations by allowing only a subset of the population to be considered if necessary, in this case, the subset is of progressively worse individuals.

$$P(i) = \begin{cases} mi + c & i > q \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\begin{aligned} P(i <= x) &= \int_q^i P(i)\,di \\ &= \begin{cases} \frac{mi^2}{2} + ci - \frac{mq^2}{2} - cq & i > q \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

Where

$$q = \begin{cases} 0 & \alpha < 1/(p+1) \\ (p+1)\alpha - 1 & \text{otherwise} \end{cases} \tag{3}$$

$$m = \begin{cases} \frac{2\alpha}{p+1} & \alpha < 1/(p+1) \\ \frac{2}{(p+1-q)^2} & \text{otherwise} \end{cases} \tag{4}$$

$$c = \begin{cases} 1/(p+1) - \alpha & \alpha < 1/(p+1) \\ -mq & \text{otherwise} \end{cases} \tag{5}$$

With a value of the cut pressure $\alpha = 0$, the density function is uniform with a probability of $1/(p+1)$ of cutting any of the individuals in the population (total replacement strategy). With $\alpha = 1$, the density function is zero for all but the worst individual, corresponding to 'true' fitness based insertion. As the cut pressure is increased from zero, the probability that the better solutions will be cut reduces to zero, forming 'elitist' types of strategy.

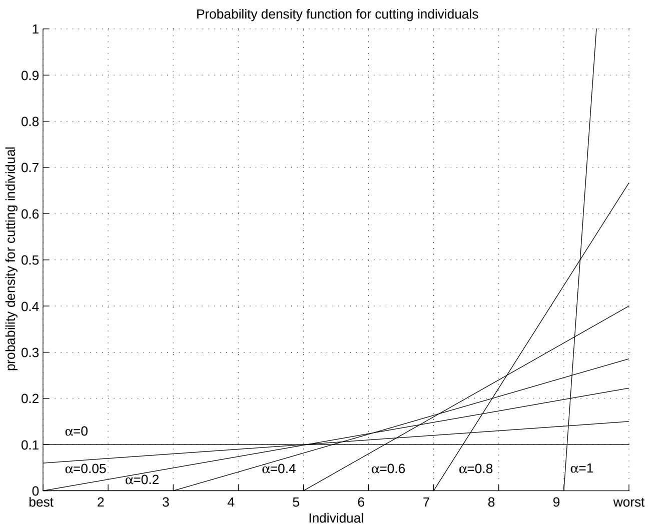

Figure 2: Variation of probability density function for cutting individuals (i) from the population as α is varied from 0.0 to 1.0 with $p = 9$

One of the problems with steady state evolutionary algorithms is premature convergence. In this paper, a process of removing duplicate solutions was used to improve the population diversity [4]. As trials were performed on problems with both no noise and noise, a duplicate solution was defined as having the chromosome *and* the objective values the same. This definition therefore only operates when no noise is present on the objective. Thus if the new individual matches an individual already in the population, the new individual is discarded. If it is known that there is no noise on the objective function, only the chromosomes need to be compared and the test can be made before evaluation to prevent solutions being discarded unnecessarily after evaluation. If a solution is discarded rather than being added to the population, the population size will remain at p and therefore no solutions need to be cut.

The probabilistic cut mechanism may also be applied to conventional generational evolutionary algorithms by concatenating the old population with the new individuals and then cutting solutions until the desired population size is obtained.

2.3 Selection and Child Generation

After an individual has been inserted into the population (or not if it was a duplicate), a single chromosome is generated from the current population for evaluation. Two parents are selected to form the child chromosome. Each parent is chosen by first ranking the raw objective values and then calculating fitness from the rank positions by using equation 6 [5, Chapter 1, section 1.2.2]

$$F(i) = 2 - s + 2(s-1)\frac{r_i - 1}{p - 1} \tag{6}$$

Where s is the selective pressure ($1 \leq s \leq 2$) and r_i is the rank of individual i with the rank p being most fit ($1 \leq r_i \leq p$).

The selective pressure, s, allows the evolutionary effect of selection to be controlled. A value of $s = 1$ means that the parents are chosen uniformly with no relation to the rank positions. A value of $s = 2$ will give the best individual a fitness of 2 and the worst a fitness of zero.

Roulette wheel selection is used independently for each parent with a probability of selection given by equation 7.

$$P(i) = \frac{F(i)}{\sum_{j=1}^{p} F(j)} \tag{7}$$

In the trials in section 3, real valued chromosomes were used. The child chromosome, $\mathbf{O}$, was created from the two parent chromosomes, $\mathbf{P_1}$ and $\mathbf{P_2}$, using intermediate recombination [5, Chapter 1, section 1.2.4.4] shown in equation 8.

$$\mathbf{O} = \mathbf{P_1} + \mathbf{k}(\mathbf{P_2} - \mathbf{P_1}) \tag{8}$$

The vector $\mathbf{k}$ is generated at random with values lying in the range [-0.25, 1.25]. This allows the child chromosome to be generated somewhere within a hypercube 25% larger than the hypercube defined with $\mathbf{P_1}$ and $\mathbf{P_2}$ at opposing corners. The crossover operator is applied with a probability of 0.9 in the trials.

After crossover, each of the genes is mutated with a probability of 0.3. Mutations consist of adding zero mean Gaussian white noise with a standard deviation, σ_m, of $1/6$ of the total range of the gene. This corresponds to $\pm 3\sigma_m$ covering the expected range of the gene, for a gene at the centre of its range.

3 Example Results

3.1 Test Functions

Two test functions were formulated as maximisation problems. Both functions have an optimum of 1.0 at point $(0,0)$ and many local optima. Both were formulated for a chromosome using two real valued genes, each lying in the range $[-10, 10]$. Test function 1, described by equation 9 and figure 3, has a narrow central spike as the global optimum point and then a series of concentric ridges.

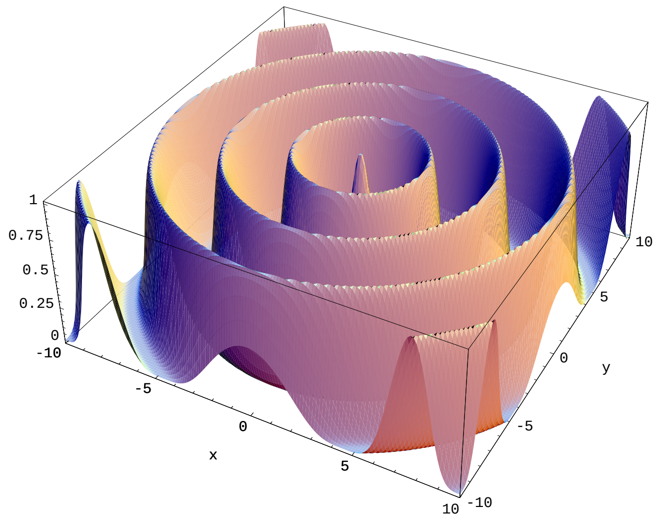

Figure 3: Test objective surface of equation 9

The gradient of each ridge is zero, giving ‘rings’ of local optima, rather than discrete points. Zero mean Gaussian noise ($N(0, \sigma^2)$), with a standard deviation σ, is effectively applied to the chromosome *before* the objective is evaluated. This is to simulate problems that suffer from internal ‘system’ noise. The output of the objective function still lies within the range $[0, 1]$.

$$\begin{aligned} \mathcal{O} &= \left(\frac{\cos\left(d^2\right)}{1 + d/1000}\right)^2 \\ d &= \sqrt{(x^2 + y^2)} + N(0, \sigma^2) \end{aligned} \tag{9}$$

Test function 2, described by equation 10 and figure 4, has a central ridge, with very similar ridges on each side. The ridges are perturbed to make the optimisation more difficult. Zero mean Gaussian noise is applied to the objective value *after* computation to simulate problems where measurement noise dominates. In this case, the observed objective value may lie outside of the range $[0, 1]$.

$$\mathcal{O} = \left(\frac{\cos\left(x/2 + \sin\left(y/2\right)^2\right)}{1 + \sqrt{(x^2 + y^2)}/1000}\right)^2 + N(0, \sigma^2) \tag{10}$$

3.2 Trials

A series of trials were run to assess the effects of α on various aspects of algorithm performance. The algorithm was coded using ‘C++’ with parallel *Message Passing Interface* (MPI)

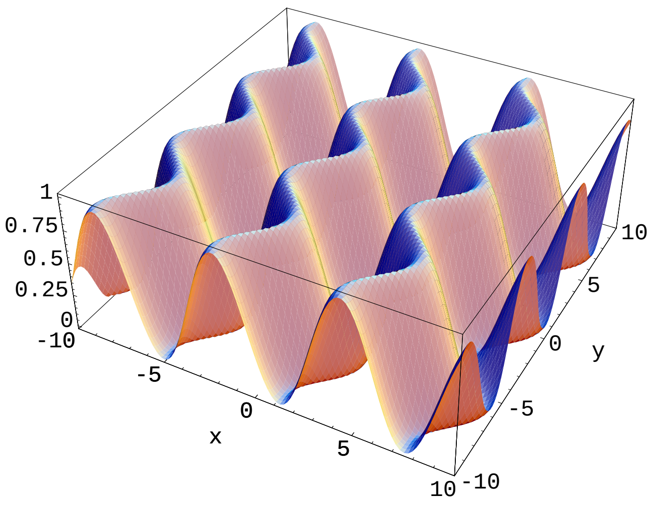

Figure 4: Test objective surface of equation 10

protocols. The trials were conducted on Cranfield University's SGI CRAY Origin 2000 supercomputer.

The following experiments were conducted:

1. $\alpha = [0, 1]$ in 21 steps of 0.05, obj. 1, selective pressure 1.3, applied noise $\sigma = [0, 0.8]$ in 9 steps of 0.1.
2. $\alpha = [0, 1]$ in 21 steps of 0.05, obj. 1, selective pressure 1.8, applied noise $\sigma = [0, 0.8]$ in 9 steps of 0.1.
3. $\alpha = [0, 1]$ in 21 steps of 0.05, obj. 2, selective pressure 1.3, applied noise $\sigma = [0, 0.45]$ in 10 steps of 0.05.
4. $\alpha = [0, 1]$ in 21 steps of 0.05, obj. 2, selective pressure 1.8, applied noise $\sigma = [0, 0.45]$ in 10 steps of 0.05.
5. Selective pressure= $[1, 2]$ in 11 steps of 0.1, $\alpha = 0.5$, obj. 1 with applied noise $\sigma = [0, 0.8]$ in 9 steps of 0.1, obj. 2 with applied noise $\sigma = [0, 0.45]$ in 10 steps of 0.05.

For each trial, 15000 objective calculations were performed with a population size of 200 individuals, mutation rate of 0.3, crossover rate of 0.9, and one new individual being generated from each selection cycle. The top 10 individuals (based on objective value) were selected from each trial, and each trial was repeated 1000 times (with different random number sets), giving 10,000 evolved objective locations for each experiment. Three parameters were monitored for each experiment:

1. Mean number of iterations of the algorithm to first locate the objective value
2. Probability that *global* optimum is found
3. Mean Euclidean difference between true optima location and recorded position.

The first two parameters, mean number of iterations ($\bar{N}$) and probability of finding optimum ($P(opt)$), were combined to give the *expected number of evaluations for success* (ENES) using $ENES = \bar{N}/P(opt)$.

3.3 Results

3.3.1 Experiment 1

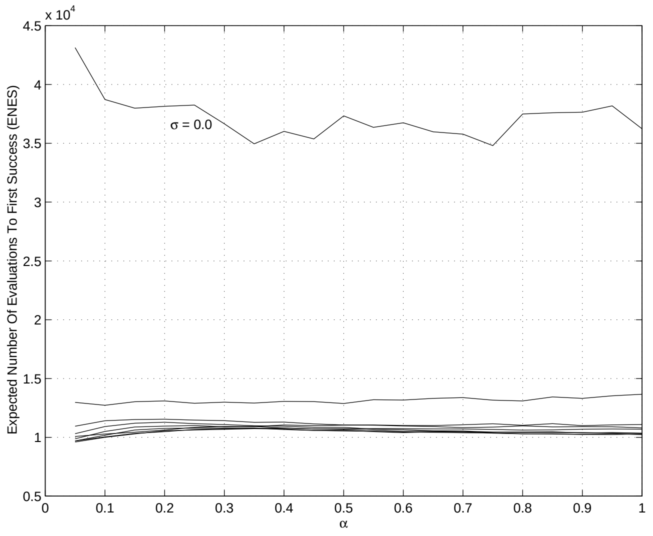

Figure 5: Experiment 1. ENES results for test objective 1 with selective pressure of 1.3 for $0 \le \alpha \le 1$ and applied noise of standard dev. $0 \le \sigma \le 0.8$.

Figure 5 shows the ENES for the experiment. In the plot, the values at $\alpha = 0.0$ are lost as $P(opt)$ is near zero and therefore the ENES has little meaning as there is little, if any, chance of finding the true optimum. As the noise is effectively applied to the chromosome before objective evaluation, the narrow central peak in the objective is moved around by the noise and becomes *easier* to find. This accounts for the ENES traces reducing with increasing noise. For $\sigma = 0.0$, i.e. no noise, values of α in the range [0.3,1] are acceptable for good ENES performance. In this experiment, a low selective pressure of 1.3 is used to reduce the effects of the selection process.

Figure 6 shows the region α in the range [0,0.2] in more detail (trace corresponding to $\sigma = 0.0$ removed for clarity). It is clear from the plot that the relationship between α and the applied noise is non-linear. For the higher noise levels, a value of $\alpha = 0.04$ is near optimum, although the differences in ENES are very small.

With a population size of 200, this value of $\alpha = 0.04$ corresponds to having a probability of zero of cutting the top 7 individuals, and an increasing probability of cutting the lower individuals. This is similar to some total replacement strategies 'with elitism', where the whole population is replaced by the child chromosomes, except for the best individual.

Figure 7 shows the positional errors in where the optimisation process converged, compared to where the true objective centre was located. As the noise standard deviation, σ, increased, the mean error also rose as expected. The values at $\alpha = 0.0$ are spurious due to the poor performance of the algorithm at this setting and should be ignored.

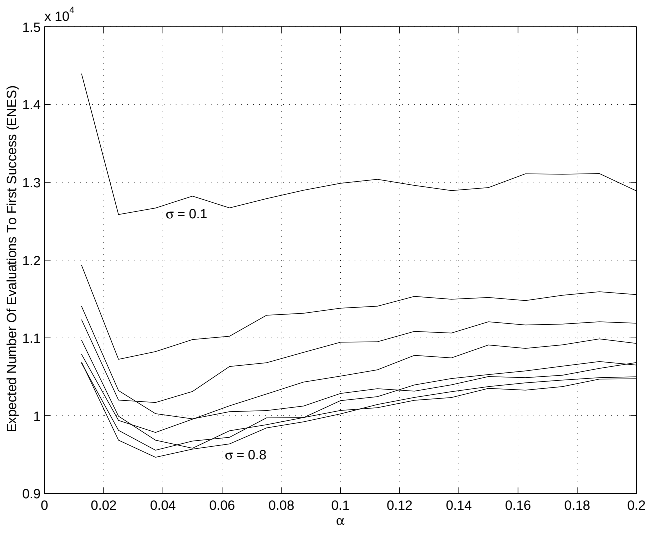

Figure 6: Experiment 1. ENES results for test objective 1 with selective pressure of 1.3 for $0 \leq \alpha \leq 0.2$ and applied noise of standard dev. $0.1 \leq \sigma \leq 0.8$.

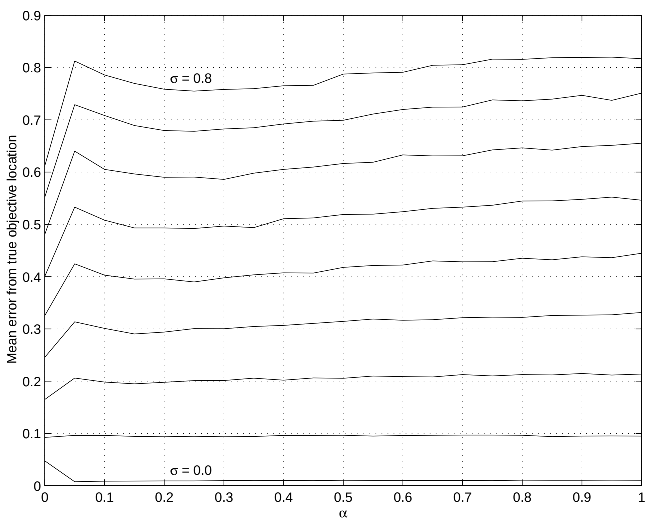

Figure 7: Experiment 1. Mean error from true objective location for test objective 1 with selective pressure of 1.3 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.8$.

It is clear that the best performance lies in the range $\alpha = [0.2, 0.3]$, although a difference is only really noticeable at high noise levels. This region tends to coincide with the areas of higher ENES where more iterations are needed to find the optimum, although the changes in ENES are small compared to the changes in accuracy at the high noise levels.

3.3.2 Experiment 2

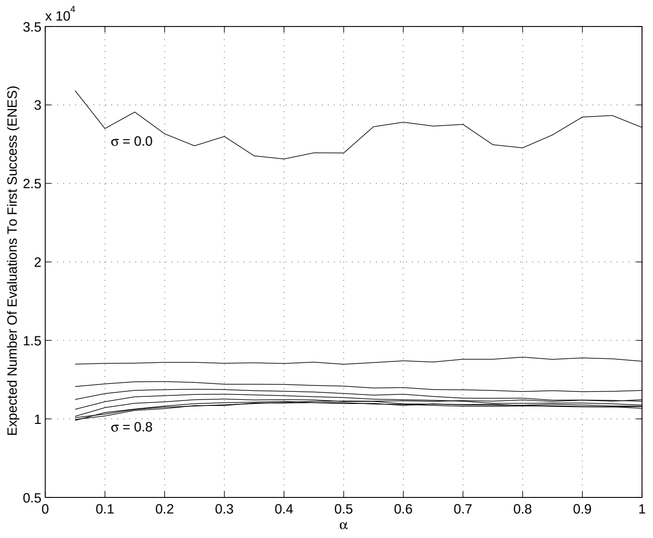

Figure 8: Experiment 2. Expected number of evaluations for success results for test objective 1 with selective pressure of 1.8 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.8$.

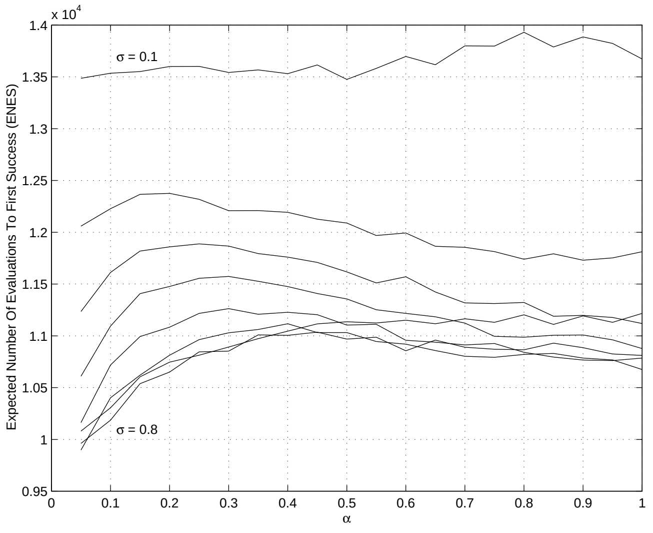

Figure 9: Experiment 2. Expected number of evaluations for success results for test objective 1 with selective pressure of 1.8 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0.1 \leq \sigma \leq 0.8$.

In this experiment, a higher value of selective pressure was used to examine the effects of the probabilistic cut operator in combination with a significant alternative route for evolution, i.e. selection. Figure 8 shows the ENES plot for the experiment. It is clear that for the no noise case, $\sigma = 0.0$, the algorithm is more efficient with the higher selective pressure (≈ 36000 for the lower selective pressure, compared to ≈ 27000 for this experiment). It is also noticeable that for the noisy cases, the performance is slightly *worse*, although the difference is very small.

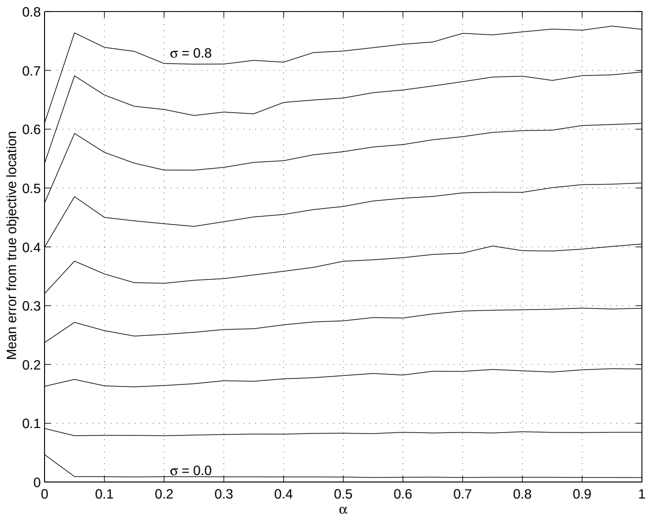

Figure 10: Experiment 2. Mean error from true objective location for test objective 1 with selective pressure of 1.8 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.8$.

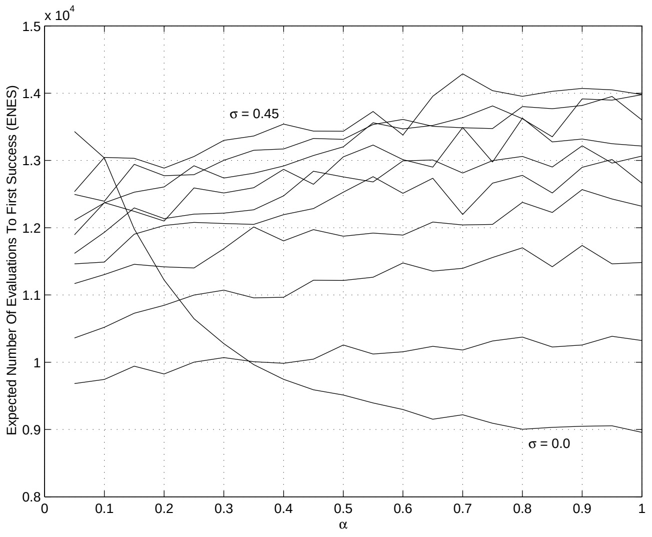

Figure 11: Experiment 3. ENES results for test objective 2 with selective pressure of 1.3 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

Again, figure 9 shows the previous plot with $\sigma = 0.0$ removed for clarity and again it is clear that the relationship between α and the applied noise is non-linear. Figure 10 shows that the positional errors are only slightly reduced with the higher selective pressure for this objective. Thus increasing the selective pressure has reduced the efficiency of the algorithm slightly as expected, but has improved the accuracy of the final result. This corresponds well with the initial hypothesis that the selection process focuses on improving the good solutions, whereas the insertion operator removes bad solutions. Thus in an environment with ‘system’ noise, both operators need to be controlled.

3.3.3 Experiment 3

Figure 11 shows the ENES results for objective 2 with a low selective pressure. It is clear that a value of $\alpha = 1$ is needed for best performance in the noise free case. As the applied noise increases, for this objective the noise is being applied *after* calculation, the optimum becomes harder to locate and so the ENES increases.

All the noisy cases for this experiment would benefit from using $\alpha = 0.05$. Again the relationship between α and σ is non-linear, but is also dependent on the objective and noise characteristics. Figure 12 shows the positional errors. As expected, the errors increase as the applied noise is increased. The value of α in the range [0.05,1] has very little influence on the errors with this objective.

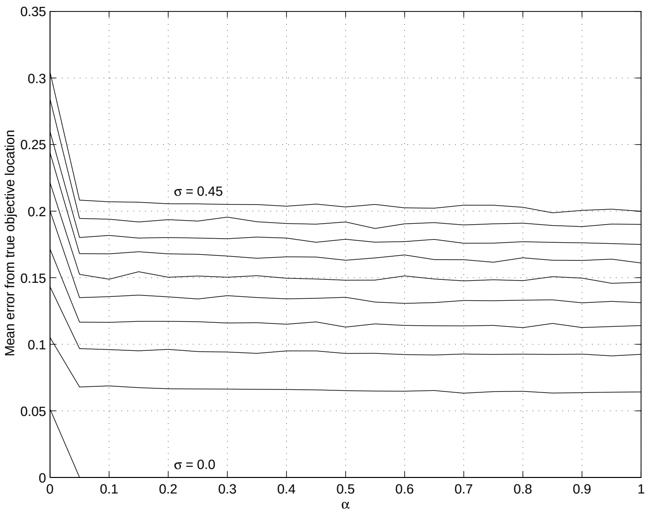

Figure 12: Experiment 3. Mean error from true objective location for test objective 2 with selective pressure of 1.3 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

3.3.4 Experiment 4

Experiment 3 has been repeated with a selective pressure of 1.8 to establish the influence on the algorithm performance with objective 2. Figure 13 shows the ENES results for the experiment. It is clear that the performance in the no noise case ($\sigma = 0$) has been improved significantly, but the performance on the noisy cases has changed very little.

Experiment 2 showed a similar result and suggests that for some noisy cases, the selective pressure has very little effect on algorithm performance and that the insertion operator dominates the evolutionary process. This fits well with our initial hypothesis that the selection process may suffer with noise.

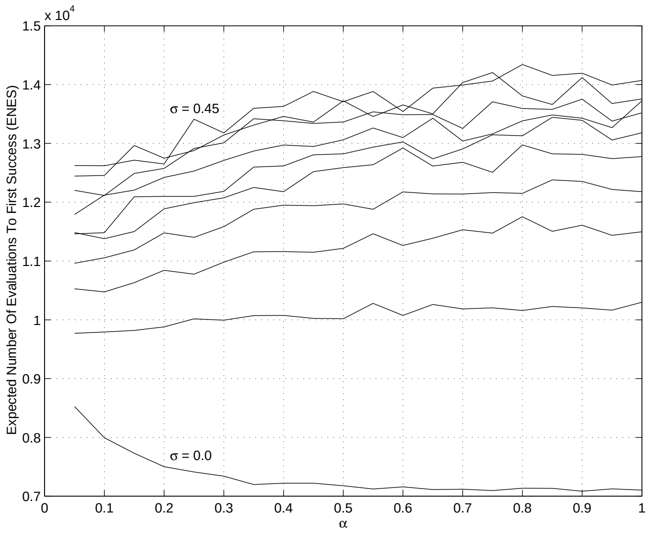

Figure 13: Experiment 4. ENES results for test objective 2 with selective pressure of 1.8 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

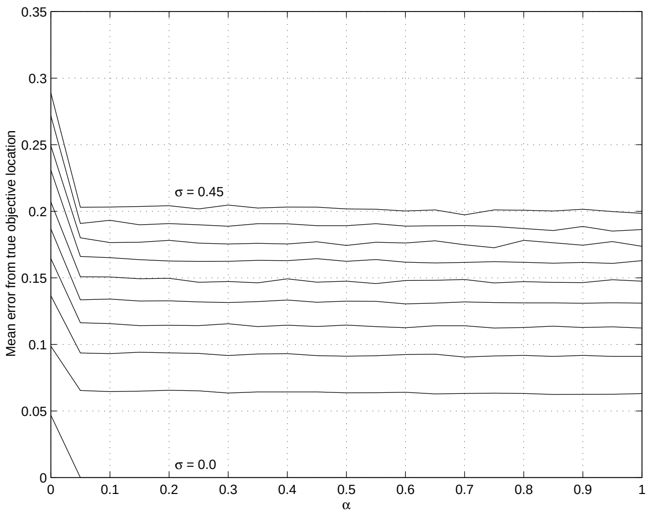

Figure 14: Experiment 4. Mean error from true objective location for test objective 2 with selective pressure of 1.8 for $0 \leq \alpha \leq 1$ and applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

Figure 14 shows the corresponding error plot for the experiment. Again, the change in selective pressure has had very little effect on the algorithm performance, although for this objective, the cut pressure, α, has very little effect on error performance either.

It is interesting to note that with the 'system' noise of objective 1, the objective value still lies in the range [0,1] and selection is important. With the 'measurement' noise of objective 2, the objective no longer lies within [0,1] and the selection process now has very little influence.

3.3.5 Experiment 5

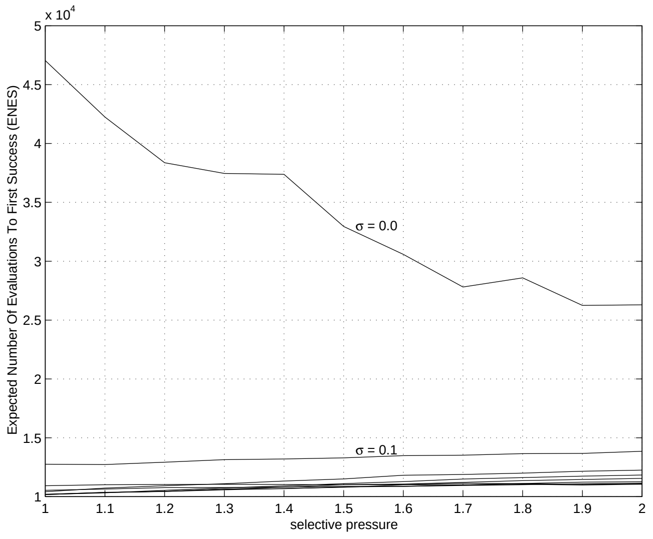

Figure 15: Experiment 5. ENES results for test objective 1 with selective pressure in range $[1,2]$ and with $\alpha = 0.5$ and applied noise of standard dev. $0 \leq \sigma \leq 0.8$.

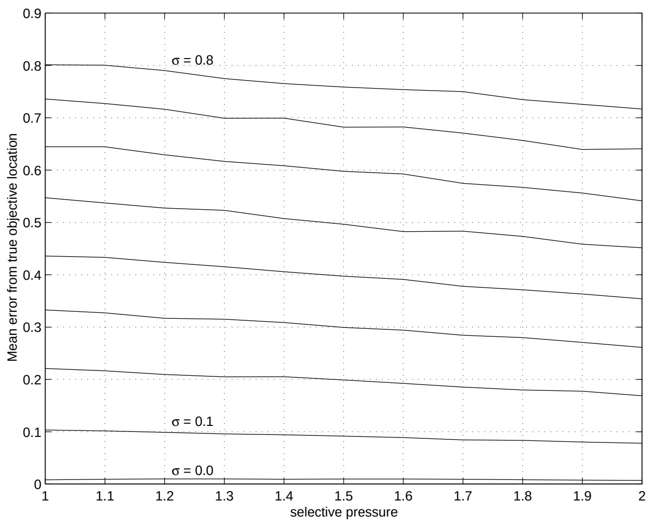

Figure 16: Experiment 5. Mean error from true objective location for test objective 1 with selective pressure in range $[1,2]$ and with $\alpha = 0.5$ applied noise of standard dev. $0 \leq \sigma \leq 0.8$.

To verify the contribution of the selection process further, in this experiment the selective pressure has been varied from $s = 1$, random selection, to $s = 2$, corresponding to the worst individual having zero probability of selection. A cut pressure of $\alpha = 0.5$ is used. Figure 15 shows the ENES results for objective 1.

Clearly for the no noise case, increasing the selective pressure improves the algorithm efficiency. Conversely, if noise is present, a selective pressure of 1 is best, i.e. random selection.

Figure 16 shows that reducing the effect of the selection operator has a detrimental influence on the accuracy of the final solution. Figures 17 & 18 show that for objective two, the selective pressure had no significant influence on either ENES or the accuracy of the solutions when noise was present.

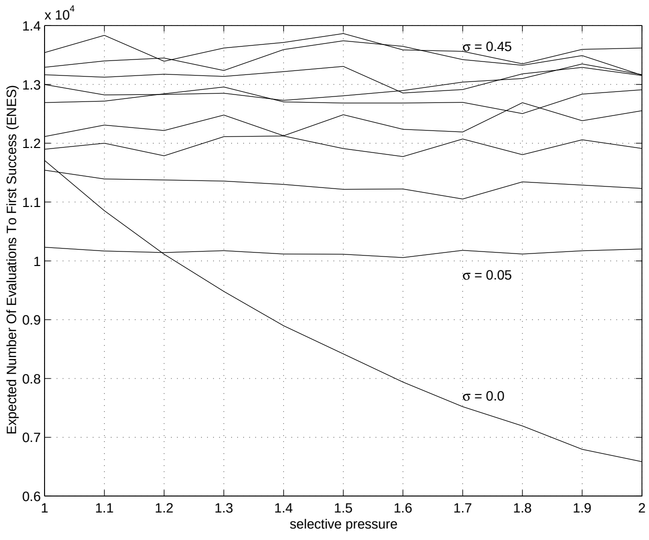

Figure 17: Experiment 5. ENES results for test objective 2 with selective pressure in range $[1, 2]$ and with $\alpha = 0.5$ and applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

4 Conclusions

This paper has introduced a new insertion process called the probabilistic cut operator that may be applied to any existing form of population based evolutionary algorithm and allows the evolutionary effects of the insertion process to be controlled. The ability of the operator to improve the performance of the evolutionary algorithm in the presence of noisy objectives has been demonstrated.

The algorithm is simple and has been implemented in an asynchronous parallel evolutionary algorithm and applied to functions that have similar noise characteristics to real engineering problems.

It has been established that the relationship between the cut pressure α, the level of noise, the selective pressure, and the algorithm performance in terms of efficiency and accuracy is non-linear and highly dependent on the objective function characteristics. It has also been demonstrated that with some noisy objective functions, the insertion process is the dominant evolutionary operator with parent selection playing only a minor role in determining algorithm efficiency, whereas the selection process is dominant in terms of the accuracy of the final solution. Both operators need to be adjusted to obtain maximum algorithm performance.

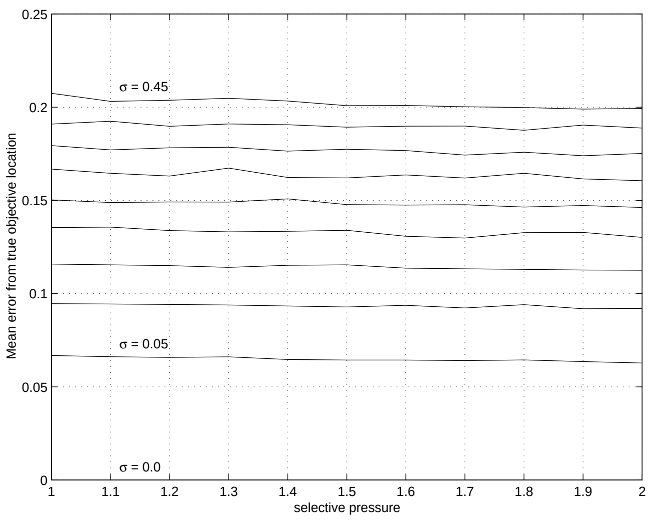

Figure 18: Experiment 5. Mean error from true objective location for test objective 2 with selective pressure in range $[1, 2]$ and with $\alpha = 0.5$ applied noise of standard dev. $0 \leq \sigma \leq 0.45$.

Acknowledgements

The author would like to acknowledge the use of Cranfield University's SGI CRAY Origin 2000 supercomputer facility at RMCS, Shrivenham, for performing the experimental calculations.

Bibliography

[1] S. Tsutsui and A. Ghosh, "Genetic algorithms with a robust searching scheme," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 201–8, Sept. 1997.

[2] F. Vavak and T. C. Fogarty, "Comparison of steady state and generational genetic algorithms for use in nonstationary environments," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 192 – 195, IEEE, 20-22 May 1996.

[3] K. Rasheed and B. D. Davison, "Effect of global parallelism on the behavior of a steady state genetic algorithm for design optimisation," in *Proceedings of the 1999 Congress on Evolutionary Computation - CEC99*, vol. 1, (Washington, DC, USA), pp. 534–41, IEEE, July 1999.

[4] S. Ronald, "Duplicate genotypes in a genetic algorithm," in *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence.*, (New York, NY, USA), pp. 793 – 8, IEEE, 1998.

[5] A. Zalzala and P. J. Flemming, eds., *Genetic algorithms in engineering systems*. The Institution of Electrical Engineers, 1997.