A Controlled Migration Genetic Algorithm Operator for Hardware-in-the-Loop Experimentation.

D. Gladwin,[a], P. Stewart[b] and J. Stewart[b]

[a]*Department of Electronic and Electrical Engineering, University of Sheffield. Mappin St. Sheffield S1 3JD U.K.*; [b]*School of Engineering, University of Lincoln, Lincoln LN6 7TS, UK. Corresponding Author: pstewart@lincoln.ac.uk*

***Abstract*—In this paper, we describe the development of an extended migration operator, which combats the negative effects of noise on the effective search capabilities of genetic algorithms. The research is motivated by the need to minimize the number of evaluations during hardware-in-the-loop experimentation, which can carry a significant cost penalty in terms of time or financial expense. The authors build on previous research, where convergence for search methods such as Simulated Annealing and Variable Neighbourhood search was accelerated by the implementation of an adaptive decision support operator. This methodology was found to be effective in searching noisy data surfaces. Providing that noise is not too significant, Genetic Algorithms can prove even more effective guiding experimentation. It will be shown that with the introduction of a Controlled Migration operator into the GA heuristic, data, which represents a significant signal-to-noise ratio, can be searched with significant beneficial effects on the efficiency of hardware-in-the-loop experimentation, without a priori parameter tuning. The method is tested on an engine-in-the-loop experimental example, and shown to bring significant performance benefits.**

I. Introduction

FOR the design, implementation and testing of systems, which are complex and/or difficult to represent to a sufficiently high degree of accuracy in simulation; it is common practice to adopt a hardware-in-the-loop approach, where some of the control loop components are real items of hardware [18], [19]. In this way, major systems components (such as engines in automotive applications) can be evaluated and control systems designed without the expense and complexities of a whole system empirical development programme [30]. However, the utilization of real hardware introduces the significant issue of sensor and measurement noise. The authors have previously conducted research into the development of techniques to improve the performance of several standard search heuristics such as gradient descent [31], [32], variable neighbourhood search [12] and simulated annealing [36] in supporting hardware-in-the-loop search in internal combustion engine development [38]. This produced a methodology, which makes use of the data evaluated by the heuristic during the search, and utilizes this to produce response surfaces. These response surfaces are used to generate probability surfaces to provide the search heuristic with weighted stochastic decision support (WSDS) (figure 1).

This operator supports the heuristic and guides the experimental process to predicted areas of interest in the search space. Basic gradient descent, Simulated Annealing (SA) and Variable Neighbourhood Search (VNS) were supplemented by the WSDS methodology, and performance compared to the basic form of the heuristics. The supplemented heuristics were shown to have significantly improved performance when searching over increasingly noisy surfaces.

Fig. 1. Decision Support Architecture

It would be expected that Genetic Algorithms (GAs) should be effective in noisy environments, and out-perform basic heuristics [6]. The GA allows for variance in fitness values, and providing that noise isnt overwhelming, this is effective, since the GA doesnt discard useful information too quickly. In comparison, local search may not identify improving moves or local optima without a priori information related to the nature of the noise. For this reason, GAs are studied in this paper. In comparative studies conducted at the time, an initial experimental investigation into the performance of GAs was carried out, resulting in a lower performance level than was anticipated. This motivated the current work, which addresses the application of GAs to real-life experimental decision support applications

GAs have been shown to be compromised when directing search over significantly rugged surfaces [15], [28], such as those applications discussed in this work. As the amount of noise inherent in the surface increases, it is likely that the number of local optima increases and, unless there is sufficient diversity within the populations of the GA, this often causes the GA to converge on these local optima, rather than the global, optimal solution [14]. Diversity is important in genetic algorithms [29], as crossing over a homogeneous population

does not yield new solutions [10]. The parameters of a GA can be improved for such problems, for example using a high, or directed mutation rate [23], [27], larger population sizes [1], [11], [16] or by suitable selection techniques [8], [13]. A priori knowledge is typically required to set these parameters, although solutions such as adapting the parameters throughout the search using deterministic control schemes have been produced [4], [5], [9]. However, a most important aspect to be considered here is that for many Hardware in the Loop applications, a priori knowledge is not available, and the search space can be considered unknown and unseen.

Another possible degree-of-freedom in GA implementation is Mutation, which is used to maintain the diversity of the entire population by changing individuals bit by bit with a small probability $pm[0, 1]$, termed *mutation rate*. There is much debate whether high or low mutation rates should be used and whether these should be static or adaptive. A high mutation rate increases the level of exploration creating a more diverse population according to [26], which is desirable for more complex combinatorial problems. However, there have been many proposed static mutation probabilities which are derived from experience or by trial-and-error. De Jong suggested $pm = 0.001$ in [6], with Schaffer et al extending this to a range of $[0.001, 0.005]$ [35]. Bck used Schaffers results in [3] to propose that the mutation rate should be set according to population size and length of individuals, giving $pm = 1.75/(N * L1/2)$, where N is the population size and L denotes the length of individuals. Mhlenbein [27] recommended that pm = 1 / L is an acceptable mutation rate and should be generally optimal. There is, however, evidence, both empirical [9] for learning control rules, and theoretical [3] that the optimal rate of mutation is not only different for every problem but will vary with evolutionary time according to the state of the search and the nature of the landscape being searched. Work by Thierens [39] proposes two simple adaptive mutation rate control schemes called constant gain and declining. Thierens compares these to fixed mutation rates, and other known self-adaptive mutation rates showing that they perform favourably in terms of performance with no initial parameters to configure. Qiu [34] proposes a new multi-objective evolutionary algorithm, called selective migration parallel genetic algorithm (SMPGA) in which a new migration strategy develops a searching population and a elite population evolve at the same time to keep and improve the convergence and diversity of the Pareto optimal set. Power [33] incorporates a diversity guided selection mechanism, selecting a diverse set of individuals for migration from the evolving populations, and reports good performance.

None of the cited methods report activity in noisy environments, and many require *a priori* knowledge of the problem domain. In particular, adaptive mutation schemes require considerable *a priori* knowledge and subsequent parameter tuning. The application domain in which we are working, in particular, the automotive and aerospace sectors have, in general, rugged or noisy search surfaces, with little *a priori* information. Often, the experimental evaluations of the controller are expensive, and hence it is preferred to use a methodology which requires a minimum of parameter tuning to achieve convergence. Given the prior success of weighted operators in raising the performance of local heuristics, and associated with no a priori tuning requirement, this approach will be investigated in this paper in conjunction with migration operators, which have been shown recently to have significant potential in this kind of application area.

II. Random Migration Operators

In this section, we introduce the random migration operator based upon the migration operator that is used in multi-deme (multiple population) GAs [24], [25], and apply this to single-deme GAs supported by a decision support operator to yield a novel operator called controlled migration. It should be noted that controlled migration is equally applicable to the multi-deme case although this is not investigated here. Multi-deme GAs make use of the migration operator to pass individuals between sub populations according to a pre determined migration rate and migration interval. During a search, sub populations will receive a new individual from another sub population that could be from anywhere in the global search space. The individual that is received is likely to have been evolved in a sub population that may be converging towards an alternative optimum, thus creating diversity in the receiving population. In a single-deme (single population) GA, a similar scheme can be applied where random individuals are introduced into each generation from the global search space, thus introducing an alternative source of diversity. A typical GA will use either the incremental/steady state genetic algorithm (IGA) model [2], [41] or the generational genetic algorithm (GGA) [7], [40]. Here we use the GGA that batch replaces an entire population each generation, as opposed to the IGA which in typical applications only replaces one individual at a time. Figure (2) represents the GGA methodology that is applied in this section.

Fig. 2. GGA architecture

To insert random individuals into a generation, the following changes are necessary: In step 4 select fewer individuals that

are required to create the next population; step 7 is then altered to insert the processed individuals from step 6 into a new generation, and to also introduce randomly generated individuals termed migrants to maintain the population size (figure 3). The term migration rate defines the number of migrants to insert into the new population, and hence the number of individuals to select in step 4 will be equal to the original population size less the migration rate.

Fig. 3. New generation compiled of processed individuals and random migrants

For the development of this methodology, a realistic data surface with multiple local minima, plateaus and one global minimum, representative of real-life experimental combinatorial surfaces is considered. Later in this paper, the developed methodology will be applied to a real-life hardware-in-the-loop experimental application. Inspection of the experimental surface (figure 22) reveals the fundamental similarities of this kind of real problem to the development surfaces presented here. The standard MATLAB peaks surface (figure 4) describes a combinatorial process in two variables (equation 1):

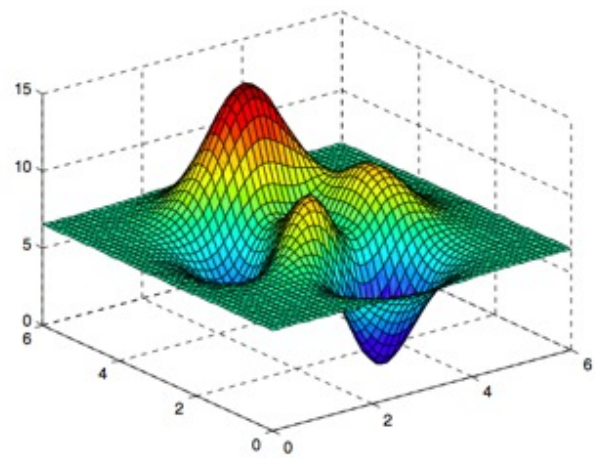

Fig. 4. Smooth algorithm development fitness landscape: peaks0

$$
\begin{aligned}
y &= 3\left(1-x_1\right)^2 .exp\left(-x_1^2-x_2^2\right) \\
&- 10\left(\frac{x_1}{5}-x_1^3-x_2^5\right).exp\left(-x_1^2-x_2^2\right) \\
&- \frac{1}{3}.exp\left(-(x_1+1)^2-x_2^2\right) \qquad (1)
\end{aligned}
$$

In order to investigate the effects of noise, progressively larger amounts of Gaussian noise are added to the smooth surface (peaks0) to give peaks 1,2,3 (figures 5, 6, 7). For the GA, performance is degraded by the number of local minima in the search space. Local optima are formed in this case by two mechanisms. The first mechanism is the underlying shape of the search space. Essentially, higher order functions tend to create more complex shapes with more local minima. Measurement or Process noise adds numerous local minima to the underlying surface. The magnitude of the noise is given as a fraction of the range of values of this input array. The addition of the noise is achieved by utilising the R function jitter written by Werner Stahel and Martin Maechler, ETH Zurich. The jitter function adds a small amount of Gaussian (white) or uniform noise to a vector, matrix or N-D array.

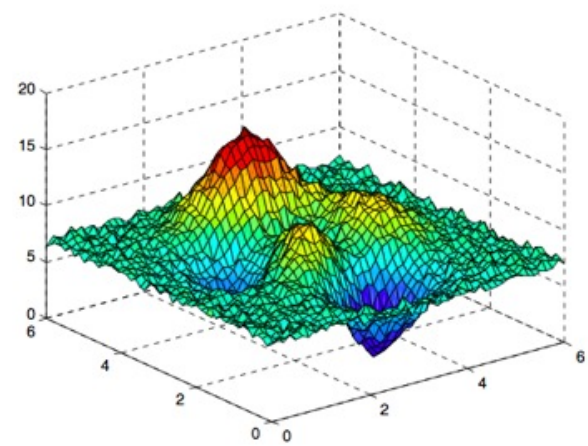

Fig. 5. Rugged algorithm development fitness landscape: peaks1

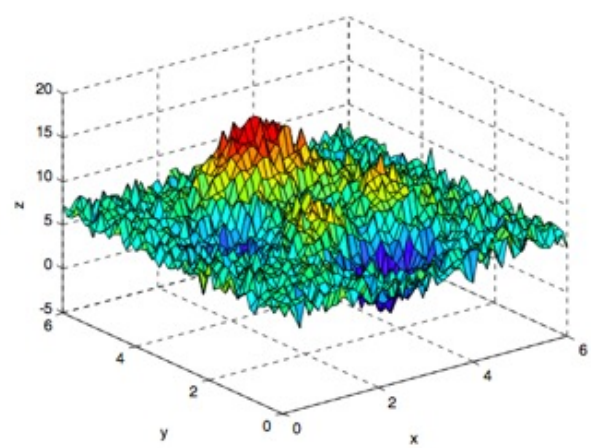

Fig. 6. Rugged algorithm development fitness landscape: peaks2

The development surfaces have increasing levels of Gaussian noise imposed on the Peaks0 surface according to:

- Peaks1 mean 0.1189, variance 0.0836
- Peaks2 mean 0.2842, variance 0.3705
- Peaks3 mean 1.7277, variance 0.7648

In order to examine the effectiveness of the method, another search space is introduced, namely the bump problem [21], which is a smooth surface comprising many peaks, all of a

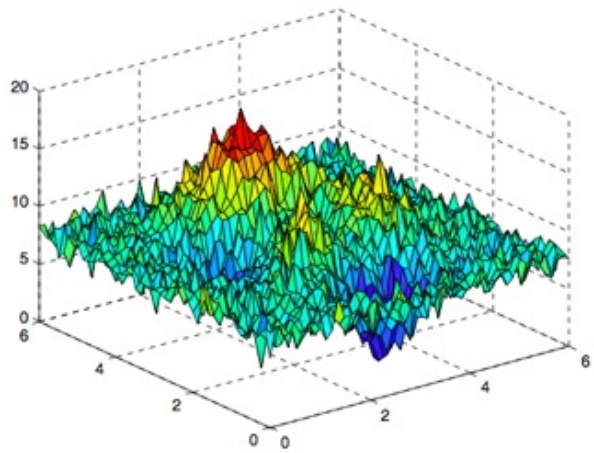

Fig. 7. Rugged algorithm development fitness landscape: peaks3

similar size. Also the optimal value is defined adjacent to a constraint boundary. The Bump problem is defined as:

$$max \frac{abs\left(\sum_{i=k}^{n} cos^4\left(x_i\right) - 2\prod_{i=1}^{n} cos^2\left(x_i\right)\right)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \quad (2)$$

for: $0 < x_i < 10,\ i = 1, ..., n$

subject to $\prod_{i=1}^{n} x_i < 0.75$ and $\sum_{i=1}^{n} x_i < 15n/2$

starting from: $x_i = 5,\ i = 1, ..., n$

where the x_i are the variables (in Radians) in the range 0 to 10 subject to two constraints, and n is the number of dimensions.

It has been noted that these features render it relatively difficult for most optimisers to deal with [22], (Figures 8, 9).

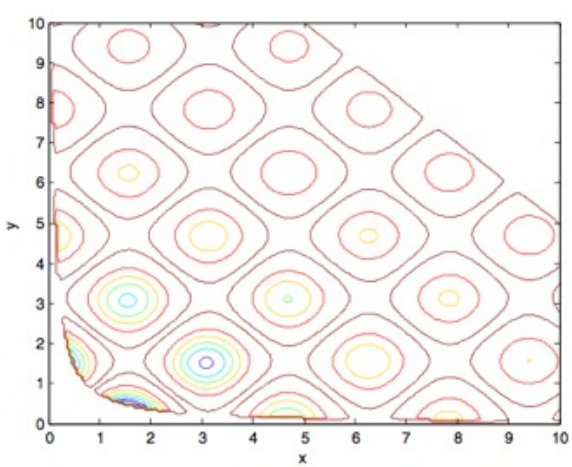

Fig. 8. Contour map for two-variable bumps function

Using this methodology with the GGA parameters as declared in Table (figure(10)), the range of surfaces *Peaks0* to *Peaks3* and *Bumps* are searched to identify the global minimum. The GGA is run 100 times per surface, producing

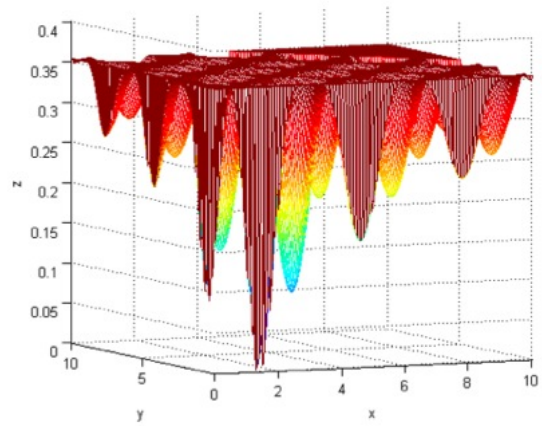

Fig. 9. Two-variable bumps function surface

mean results to negate the effects of the inherently stochastic heuristic.

Parameter	Value
Max number of generations	Infinity (stop criteria used)
No. individuals in a population	20
Survival Selection	Stochastic Universal Sampling
Recombination method	Multi-point crossover
Crossover probability	0.7
Mutation rate	0.00875

Fig. 10. GGA parameters used for search

Surface / Random Migrants	peaks0	peaks1	peaks2	peaks3	bump	Total
0	548 1012	1628 17934	10492 88650	255500 1290183	35552 408221	303720 1806000
2	553 1070	1088 5546	2447 25710	48113 363686	29028 160780	81229 597645
4	631 1145	1016 2401	1906 11743	17468 87780	38432 189995	59453 293064
6	682 1552	1306 4424	1937 7229	9310 53122	18475 107279	36982 173606
8	675 1744	1208 5685	2248 5303	7677 36729	12840 76894	24648 126355
10	804 2365	1580 4747	2417 10250	5259 28208	10455 56045	20515 101615
12	927 4065	1916 6688	2476 9364	8096 28825	9434 43765	22849 92707
14	1152 3842	2617 8566	6453 18447	5390 17123	7180 37563	22792 85541
16	1438 4822	3531 11502	11497 51503	12233 46523	9795 37062	38494 151412
18	4697 13662	13071 58641	31529 105001	48374 214102	12066 68060	109737 459466

Fig. 11. Effects of increasing random migrants across the test surfaces with a population size of 20, (upper value: mean, lower value: worst case)

Table (figure(11)) shows the effect on computations of inserting random migrants into a population of size 20 for a range of migration rates. A computation is counted as each evaluation of an individual. It shows that introducing random migration for complex surfaces such as peaks3 and the bump yields a considerable decrease in the number of computations compared to having no migrants. Figure 12 and figure 13

illustrate the effects of the different migration rates across these surfaces, clearly showing that increasing the migration rate reduces computations until a critical point where the search starts to degrade. A justification for this observation is that as the migration rate increases, then so does the diversity of the population with only a small number of highly ranked individuals surviving. As the migration rate nears the population size, then the search is comparable to a random search. Observing the results from the less complex surfaces it can also be seen that a critical point also exists, albeit to a lesser degree, where a random migration rate is present that increases the performance of the GA (Figure 14)).

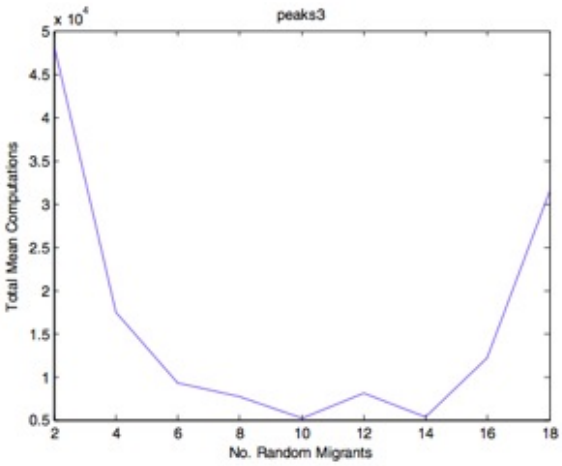

Fig. 12. Mean computations on peaks3 surface showing the effects of varying the number of random migrants for population of size 20

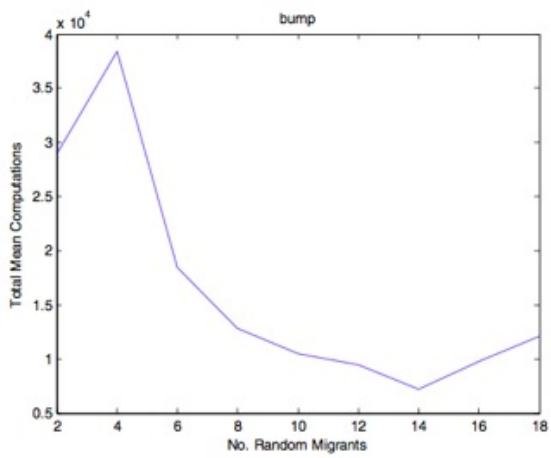

Fig. 13. Mean computations on bump surface showing the effects of the number of random migrants for population size of 20

The results show how introducing random migration into single-deme GAs can increase diversity, and hence lead to dramatic search improvements, particularly on rugged or complex surfaces. However, it is apparent that there is a critical migration rate that varies according to the complexity of the surface. A high random migration rate leads to excessive diversity analogous to high mutation rates, where previous

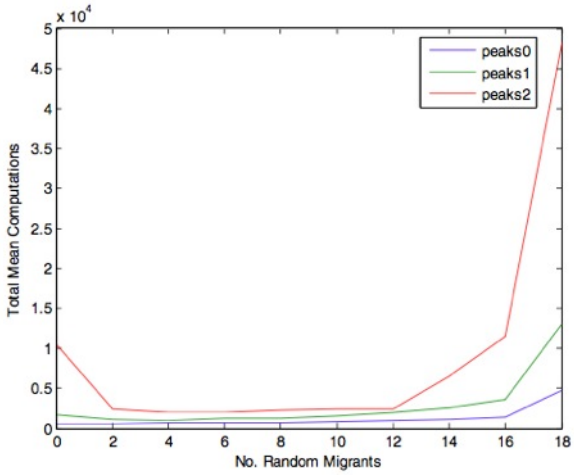

Fig. 14. Mean computations for peaks0, peaks1 and peaks2 showing the effects of the number of random migrants for population size of 20

work has shown a similar effect [37], [42]. It can be seen that low to mid random migration rates are a good trade-off between performance gains for complex surfaces, whilst minimising additional computation requirements for less complex surfaces. As with other genetic operators, the introduction of random migration has introduced another parameter that for improved effectiveness would require a priori knowledge of the surface to set a random migration rate. However, the next section will show that by applying decision support to random migration, it is possible to minimise penalties for higher random migration rates on less complex surfaces.

III. Controlled Migration

We have discussed earlier in this paper how Genetic Algorithms have been shown to be compromised when directing search over noisy evaluation surfaces. As the amount of noise inherent in the surface increases in experimental applications with additive process, measurement and sensor noise, it is likely that the number of local optima increases and, this often causes the GA to converge on these local optima. We have shown in the previous section that the introduction of a random migration operator can reduce the steps to convergence of a GA presented with noisy evaluation surfaces. The authors have previously produced a methodology, which makes use of the data evaluated by the heuristic during the search, and utilizes this to produce response surfaces. These response surfaces are used to generate probability surfaces to provide the search heuristic with weighted stochastic decision support (WSDS). Since this methodology has shown excellent results when applied with other heuristics, in this section, we aim to combine the beneficial effects of migration and weighted decision support with the aim of achieving even higher performance levels when searching noisy surfaces. A *Weighted Stochastic Decision Support (WSDS) Operator* method introduced in [38] is applied to random migration to create a novel operator termed controlled migration.

The methodology updates itself with data as it is gathered, to map areas of potential interest to direct experimentation

based upon previous results. It is an extremely compact and tractable representation, based upon polynomial response surfaces, which retains a generalized approximation of the search space. The method approximates the incoming and historical data with a polynomial function, often a second order of the form

$$\eta = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \sum_{j=1}^{k} \beta_{jj} x_j^2 + \sum_{i<j}\sum \beta_{ij} x_i x_j \quad (3)$$

It may be necessary to employ an approximating function greater than two, based upon standard Taylor series expansion. In this paper, a standard second order approximation is employed. The parameter set is estimated by least squares regression analysis. With $n < k$, an observed response
$y_1, Y_2, ..., y_n$
is associated with regression variables such that x_{ij} denotes the ith observation of variable x_j. Assuming that the error term ϵ has $E(\epsilon) = 0$ and $Var(\epsilon) = \sigma^2$ and the ϵ_i are uncorrelated variables. The model can now be expressed in terms of the observations

$$y_i = \beta_0 + \beta_1 x_{j1} + \beta_2 x_{j2} + ... + \beta_k x_{jk} + \epsilon_j \qquad j = 1, 2, ..., n \quad (4)$$

The β coefficients in (4) are chosen such that the sum of the squares of the errors ϵ_i are minimized via the least squares function

$$L = \sum_{j=1}^{n} \epsilon_i^2 = \sum_{j=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{n} \beta_j x_{ij} \right) \quad (5)$$

Thus, as data from the experimental results are gathered under the direction of the GA, it is possible to generate a surface approximation for the system under consideration. Since the true system response surface is unknown, this represents the current view of the likely response. It is this polynomial which forms the basis for the controlled migration. The y values are normalized according to

$$y_{norm} = 1 - \left(y - \frac{(max(y) + min(y))/2}{(max(y) - min(y))/2} + 1 \right) /2 \quad (6)$$

which yields a surface over the search space bounded between zero and one, where increasing value represents increasing interest, inferred from previous evaluations. These monotonically increasing values correspond to co-ordinates in a probability space from which migrants are chosen according to random selection, with probability of being chosen based on relative value in the probability space.

Figure 15 illustrates how the WSDS is integrated into the GGA methodology, with the additional steps coloured in red. During the first generation, the evaluation results from each individual in the population are used collectively to provide the data to fit the normalised response surface to. This response surface is then used to create the WSDS surface as defined in the accompanying paper. According to

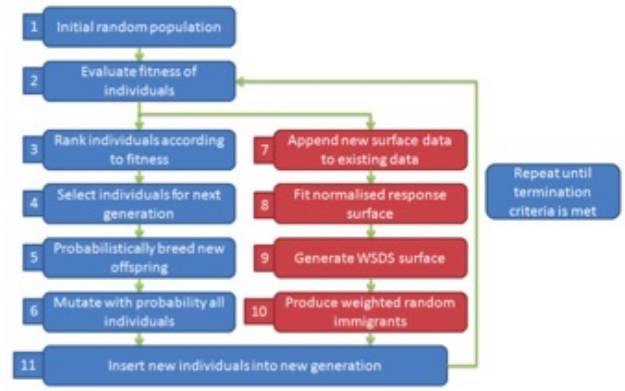

Fig. 15. GGA methodology with addition of WSDS random immigrants

the controlled migration rate, a number of migrants are then probabilistically selected from the WSDS surface and inserted into the new generation, along with the individuals processed by the standard GGA operators. This procedure is repeated for each generation, with the evaluation of each individual feeding into the data used to update the normalised response surface, thus the probability of selection of the next controlled migrants are based statistically on the results of the previous generations.

Using the GGA parameters previously presented, the experiment from the previous section is repeated replacing the random migrants with controlled migrants using the prescribed method for a range of controlled migration rates. Based on the results of the WSDS methodology applied to gradient descent methods, a 2nd order support surface is chosen. The search is conducted running the GGA on each surface 100 times to yield the mean and maximum number of computations as shown in table (figure(16)).

Surface / **Controlled Migrants**	**peaks0**	**peaks1**	**peaks2**	**peaks3**	**bump**	**Total**
2	606 1445	900 4398	1307 4666	26578 229532	4806 22919	34197 262960
4	539 1136	1063 4755	1393 5464	12563 132622	5608 19422	21166 163399
6	624 1682	1091 3974	1877 9767	5641 29022	3015 10319	12248 54764
8	602 1503	1099 3485	1683 7448	5725 31526	1804 9133	10911 53095
10	647 1630	1425 4729	2244 9585	2879 15922	1628 6895	8823 38761
12	714 2268	1625 6721	3318 16723	3393 13108	1693 5566	10742 44386
14	941 3462	1644 6681	4580 16622	5275 26703	1254 4450	13694 57918
16	989 4601	2677 9222	10897 32281	5411 22762	978 4983	20952 73849
18	2840 11727	7253 29322	35623 162742	17927 101061	1244 4799	64887 309651

Fig. 16. Effects of increasing controlled migrants across the test surfaces with a population size of 20

From the results it is immediately evident that using the controlled migration gives a 35% improvement in performance at migration rates of interest compared to using random

migration. Figure 17 and figure 18 illustrate this improvement using the total mean and max computations respectively across all the surfaces. Comparing the totals across all surfaces is justified, as although the more complex surfaces contribute most to the improvements observed, there are no significant declines in performance for less complex surfaces. Moreover, controlled migration appears to minimise penalties for higher migration rates on the basic surfaces (figure 19 and figure 20). This further endorses the generality of controlled migration as a viable operator to speed-up GA searches, as whilst it appears to cause no significant detrimental affect, it can provide a major performance boost on such surfaces as the bump (figure 21).

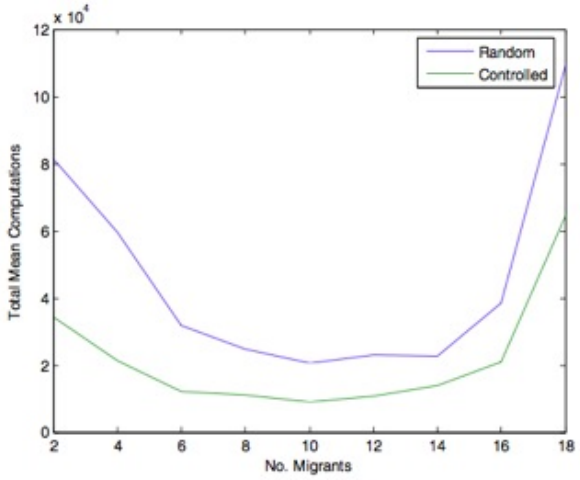

Fig. 17. Comparison of random migration vs. controlled migration for total mean computations across all surfaces

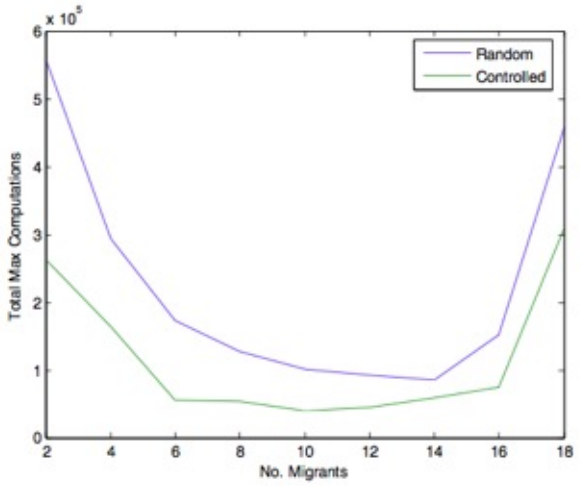

Fig. 18. Comparison of random migration vs. controlled migration for total maximum computations across all surfaces

IV. Experimental Hardware-in-the-Loop Application

The method, which in the previous section was applied to test surfaces, is now applied to an experimental automotive combinatorial search. It is desired to identify the

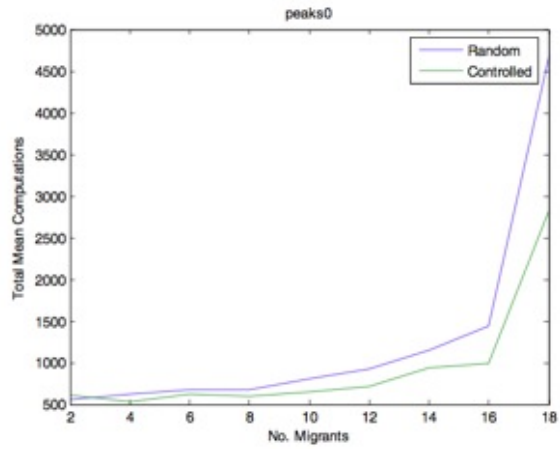

Fig. 19. Comparison of random migration vs. controlled migration for mean computation on peaks0

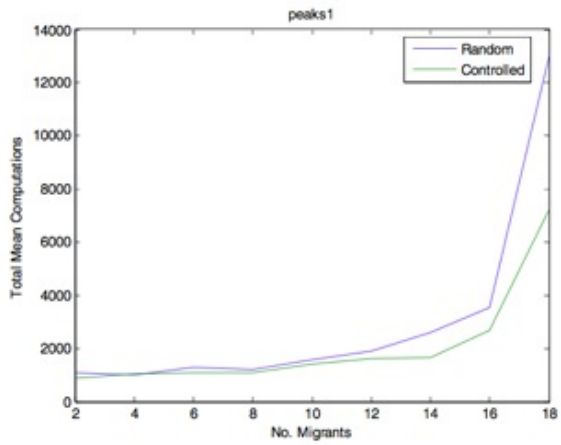

Fig. 20. Comparison of random migration vs. controlled migration for mean computation on peaks

maximum power output of an experimental single-cylinder spark-ignition engine operating under a novel control regime. The programme conducts peak power experiments under combinatorial conditions to identify global peak power for subsequent design procedures [38]. In order to evaluate the relative performance of the Controlled Migration method, the engine was characterized by exhaustive designed experiment, shown in figure 22.

In the experimental series, both standard and controlled migration, GAs were utilized to guide the search for a maximum power point. Due to the stochastic nature of the Metaheuristics, each method was run 100 times in order to produce mean performance evaluations. A comparison was also performed between population sizes of 20 (Figure 23) and 40 (Figure 24).

In both cases, at significant migration performance, the GA running a controlled migration policy outperforms the standard migration operator.

V. Conclusions

This paper introduces a novel decision support methodology based upon response surfaces. The method had been

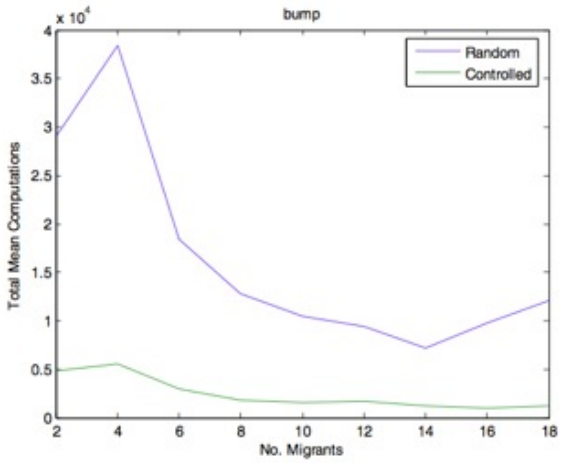

Fig. 21. Comparison of random migration vs. controlled migration for mean computations on bump

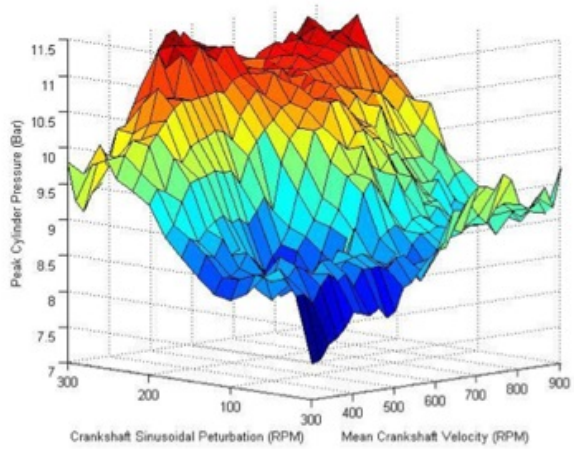

Fig. 22. Engine experimental map for peak cylinder pressure for given throttle, spark and injection settings

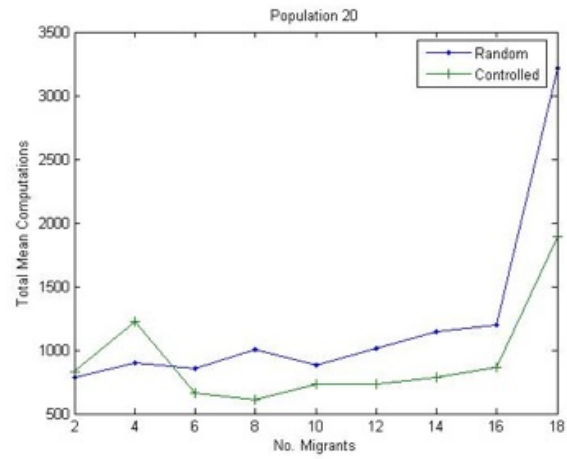

Fig. 23. Experimental results for 20 population

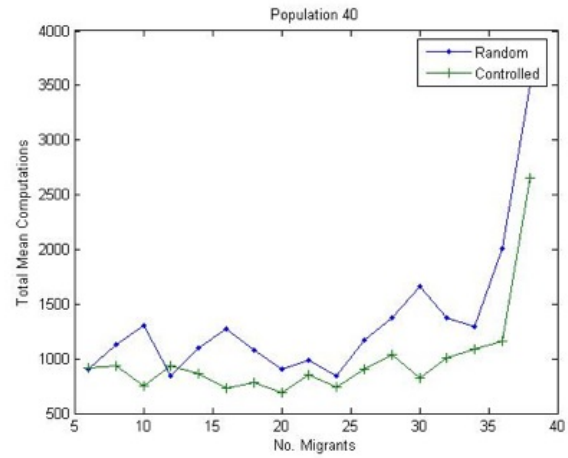

Fig. 24. Experimental results for 40 population

previously applied to add decision support to the previously random jumps of gradient descent methods commonly used in combinatorial experimentation [20]. The response surfaces are generated through exploitation of evaluated data that is performed during the search, valuable additional information which should not be discarded. These response surfaces are transformed into normalised contours of the search area, providing weighted stochastic decision support for migration.

The decision support methodology was applied to GAs, first investigating a mechanism for the introduction of a decision supported operator. Migration in single-deme GAs of random individuals was investigated as an alternative to mutation as a means of maintaining diversity in each generation. Random migration is then demonstrated to provide substantial improvements in the efficiency of a GA when faced with more complex or rugged surfaces that contain many local optima. Moreover, this migration operator provides the mechanism for which to apply decision support, and is introduced as controlled migration. Through a comparison with random migration, controlled migration is shown to provide an improvement in required computations by up to a factor of two. With both migration operators, it is apparent that there is a critical migration rate that varies according to the complexity of the surface. A high migration rate leads to excessive diversity analogous to high mutation rates. A low migration rate, whilst providing minimum risk for computation penalties for simple surfaces, does not exploit the benefits attainable when applied to more complex surfaces. However, using controlled migration over random migration is shown to allow higher migration rates, whilst minimising the detrimental effects on simple surfaces. This can be explained as whilst exploring simple surfaces, the decision support surface is more likely to provide a reliable estimate as to where the minimum or maximum lies. Using high controlled migration rates, it is more probable that good candidate individuals are chosen. In the context of hardware-in-the-loop experimentation, the methodology has the potential for significant cost savings, since each experimental evaluation in the search has associated expense in terms of time and hardware costs.

References

[1] Alander, T. (1992). On Optimal Population Size of Genetic Algorithms. IEEE International Conference on Computer Systems and Software Engineering (pp. 65-70). IEEE.

[2] AlSharafat W.S. and AlSharafat, M.S. (2010), Adaptive Steady State Genetic Algorithm for scheduling university exams, Networking and Information Technology (ICNIT), 2010 International Conference on, 70-74.

[3] Back, T. (1992). Self-Adaptation in Genetic Algorithms. Proceedings of the 1st European Conference on Artificial Life, (pp. 263-271).

[4] Baker, J. E. (1987). Adaptive Selection Methods for Genetic Algorithms. In Eribaum (Ed.), Proceedings of the 2nd International Conference on Genetic Algorithms and their Application, (pp. 14-21). Cambridge MA.

[5] Chen L. (2009), An Adaptive Genetic Algorithm Based on Population Diversity Strategy, Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on, 93-96.

[6] DeJong, K. A. (1975). An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. University of Michigan, Department of Computer and Communication Science. Ann Arbor: University of Michigan.

[7] DeJong, K. A. (1992). Are Genetic Algorithms Function Optimizers? Parallel Problem Solving from Nature PPSN2. Elsevier.

[8] Eiben A.E., Schut, M.C. and de Wilde, A.R. (2006), Evolutionary Computation, 2006. CEC 2006. IEEE Congress on, 477 - 482.

[9]]Fogarty, T. C. (1989). Varying the Probability of Mutation in the Genetic Algorithm. Proceedings of the 3rd International Conference on Genetic Algorithms, (pp. 104-109).

[10] Fogel, D. B. (2006). Evolutionary Computation: Towards a New Philosophy of Machine Intelligence. New York: IEEE Press.

[11] Furutani, H., Fujimaru, T., Yu-an Zhang and Sakamoto, M. (2007), Effects of Population Size on Computational Performance of Genetic Algorithm on Multiplicative Landscape, Natural Computation, 2007. ICNC 2007. Third International Conference on, 488 - 496.

[12] Garcia-Martinez C. and Lozano H. (2009), Continuous variable neighbourhood search algorithm based on evolutionary metaheuristic components: a scalability test. 9th Int. Conf. on Intelligent Systems Design and Applications, 1074-1079.

[13] Goldberg, D. E. (1991). A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In D. E. Goldberg, Genetic Algorithms: Foundations of Genetic Algorithms (pp. 69-93). San Mateo, CA, USA: Morgan Kauffmann.

[14] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison-Wesley.

[15] Goldberg, D. E., Deb, K. and Clark, J. (1992). Genetic Algorithms, Noise and the Sizing of Populations. Complex Systems , 6, 333-362.

[16] Gong D., Yuan J. and Ma X. (2008), Interactive genetic algorithms with large population size, Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, 1678 - 1685.

[17] Hu, S. and Zhu, Q. (2003). Stochastic optimal control and analysis of stability of networked control systems with long delay. Automatica , 39 (11), 1877-1884.

[18] Gouvenc B.A., Gouvenc L. and Karaman S. (2009). Robust yaw stability controller design and hardware in the loop testing for a road vehicle. IEEE Trans. Veh. Tech. 58 (2), 551-571.

[19] Isermann, R., Schaffnit, J. and Sinsel, S. (1999). Hardware-in-the-loop simulation for the design and testing of engine-control systems. Control Engineering Practice , 5 (7), 643-653.

[20] Jaskiewiscz, A. (1998). Genetic Local Search for Multiple Objective Combinatorial Optimisation. Technical Report RA-GL4/98, Poznan University, Institute of Computing Science, Poznan.

[21] Keane, A. J. (1994). Experience With optimizers in Structural Design. In I. C. Parmee (Ed.), Proceedings of the Conference on Adaptive Computing in Engineering Design and Control '94, (pp. 14-27). Plymouth, UK.

[22] Keane, A. J. (1995). Genetic Algorithm Optimisation of Multi-Peak Problems: Studies in Convergence and Robustness. International Journal of Artificial Intelligence in Engineering , 9 (2), 75-83.

[23] Li N and Ye F. (2006), Optimal Design of Discrete Structure with Directed Mutation Genetic Algorithms, Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on, 3663 - 3667.

[24] Liu L.-M., Wang N.-P. and Li F.-C. (2009), Study on convergence of self-adaptive and multi-population composite Genetic Algorithm, Machine Learning and Cybernetics, 2009 International Conference on, 2680 - 2685.

[25] Mann, K. F., Tang, K. S., Kwong, S. and Halang, W. A. (1999). Genetic Algorithms: Concepts and Designs. London: Springer-Verlag.

[26] Michalewicz, Z. and Fogel, D. B. (2006). How to Solve It: Algorithms for Engineering Systems. Cambridge, UK: Cambridge University Press.

[27] Muhlenbein, H. (1992). How Genetic Algorithms Really Work. Mutation and Hill-Climbing. In R. Manner and R. Manderick (Ed.), Parallel Problem Solving from Nature PPSN II, (pp. 15-25). Amsterdam.

[28] Nakama T. (2009), Transition and convergence properties of genetic algorithms applied to fitness functions perturbed concurrently by additive and multiplicative noise. 2009 IEEE Congress on Evolutionary Computation, 2662-2669.

[29] Nsakandaa A.L., Wilson L., Price B., Moustapha D. and Marc G. (2006), Ensuring population diversity in genetic algorithms: A technical note with application to the cell formation problem, European Journal of Operational Research, 178, (2), 634-638.

[30] Oh, S. C. (2005). Evaluation of motor characteristics for hybrid electric vehicles using the hardware-in-the-loop concept. IEEE Transactions on Vehicular Technology , 54 (3), 817-824.

[31] Petter, O., Fiorelli, E. and Leonard, N. E. (2004). Cooperative Control of Mobile Sensor Networks: Adaptive Gradient Climbing in a Distributed Environment. IEEE Transactions on Automatic control , 49 (8), 1292-1302.

[32] Po L.M., Ng K.H., Cheung K.-W., Wong K.-H., Uddin Y. and Ting C.-W. (2009), Novel directional gradient descent searches for fast block motion estimation. IEEE Trans. Circuits and Signals for Video Technology, 19 (8), 1189-1195.

[33] Power, D., Ryan C. and Azad, R.M.A. (2005), Promoting diversity using migration strategies in distributed genetic algorithms, Evolutionary Computation, 2005. The 2005 IEEE Congress on, 1831 - 1838 Vol. 2.

[34] Qiu T. and Ju G. (2010), A selective migration parallel multi-objective genetic algorithm, Control and Decision Conference (CCDC), 2010 Chinese, 463 - 467.

[35] Schaffer, J. D., Caruana, R. A., Eshelman, L. J. and Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimisation. Proceedings of the 3rd International Conference on Genetic Algorithms (pp. 51-60). Los Altos CA: Morgan Kaufman.

[36] Smith K.I., Eversen R.M., Murphy C. and Misra R. (2008), Dominance based multiobjective simulated annealing, IEEE Trans. Evolutionary Computation 12 (3), 323-342.

[37] Spears, W. M. (1992). Crossover or Mutation. In Whitley (Ed.), Foundations of Genetic Algorithms -2 (pp. 221-237). San Mateo, CA, USA: Morgan Kauffmann.

[38] Stewart, P. G., Gladwin, D., Stewart, J., Chen, R. and Winward, E. (2010). Improved Decision Support for Engine-in-the-Loop Experimental Design Optimisation. IMechE Part D - Automobile Engineering , In Press.

[39] Thierens, D. (2002). Adaptive Mutation Rate Control Schemes in Genetic Algorithms. IEEE International Conference on E-Commerce, Evolutionary Computation. 1, pp. 980-985. Piscataway: IEEE.

[40] Vavak, F. and Fogarty, T. C. (1996). A Comparative Study of Steady-State and Generational Genetic Algorithms. Selected Papers from AISB Workshop on Evolutionary Computing (pp. 297-304). AISB.

[41] Whitely, D. and Kauth, J. (1998). GENITOR: A Different Genetic Algorithm. Rocky Mountain Conference on Artificial Intelligence. Denver.

[42] Zhang Y., Sakamoto, M. and Furutani, H. (2009), Effects of String Length and Mutation Rate on Success Probability of Genetic Algorithm, Natural Computation, 2009. ICNC '09. Fifth International Conference on, 211 - 216.