ADAPTIVE ZOOMING GENETIC ALGORITHM FOR CONTINUOUS OPTIMISATION PROBLEMS

J. Peng[†], K. Li[†], S. Thompson[‡]

[†] *School of Electrical & Electronic Engineering*
Queen's University Belfast, UK
[‡]*School of Mechanical & Manufacturing Engineering*
Queen's University Belfast, UK

Abstract: This paper proposes an adaptive zooming genetic algorithm (AZGA) for continuous optimisation problems. Other than gradually reducing the search space with a fixed reduction rate during the evolution process, the upper and lower boundaries for variables in the objective function are dynamically adjusted based on the distribution information of variables in the whole population. This technique is evaluated on a suite of benchmark test functions to confirm its effectiveness over existing techniques in terms of convergence speed and robustness.

Keywords: Genetic algorithms, optimisation, convergence, adaptation, robustness

1. INTRODUCTION

Continuous optimisation problems widely exist in industrial applications, such as in system design, in dynamic system modelling and identification, and in plant operation and control, etc (Man, Tang and Kwong, 1999; Sabatini, 2000; Peng, Thompson, & Li, 2002). A typical continuous function optimisation problem is formulated as follows: given a set of n independent variables $x=\{x_1, x_2, \cdots, x_n\}$, and a real-valued objective function $f(\mathrm{x})$, where $x \in \Omega$, $\Omega \subseteq R^n$, Ω - the search space, and $f: \Omega \to R$ some real-valued continuous function, then an optimisation problem is to find the point x^* in Ω such that $f(x^*)$ is optimal for all $x \in \Omega$.

Conventional methods for multivariable continuous optimisation problems include calculus-based techniques and the dynamic programming technique. Calculus-based techniques require a smooth objective function and compactness of the search space Ω. Dynamic programming technique is severely restricted by its exponential increase of dimensionality in accordance with the number of generators. As stochastic global search methods, genetic algorithms (GAs) have been successfully applied to a number of optimisation problems (Goldberg, 1989).

Generally speaking, simple GAs need no auxiliary information like the first-order or second-order information of the objective functions, and they also do not need any initial estimates of the variables. However, due to the stochastic discrete sampling nature, simple GAs suffer the problem of slow convergence and the precision of solutions is limited especially when the search space is of high dimension. To speed up the convergence, a number of techniques have been proposed, among which the Successive Zooming Genetic Algorithm (SZGA) gradually reduce the search space along the evolution process (Kwon, Kwon, Jin, and Kim,

2003). As the search space is gradually narrowed, GA operations can then gradually focus on the neighbourhood of the optimal solution in the end, hence speeding up the convergence and improving the precision of solutions.

For continuous optimisation problems, the search space is normally represented as $l_i \le x_i \le u_i, i = 1, \cdots, n$, and l_i and u_i are the upper and lower boundaries for x_i. In SZGA, update of variable boundaries is performed after every N_{sub} (say, N_{sub}=100) generations and it is based on the candidate optimum point (the current best solution after N_{sub} generations) with the new boundaries set as $[X_{k,opt} - \alpha^k/2, X_{k,opt} + \alpha^k/2]$, where $X_{k,opt}$ is the candidate optimum point at the k^{th} boundary updating instant, a is a fixed zooming factor. Within the reduced search space, a new population is initiated and another round of GA searching is performed for N_{sub} generations. Obviously, micro-genetic algorithms (MGAs) technique is applied in SZGA. In SZGA, the reduction rate for the search space is set 'a priori'. However, this may cause GA operation fail to adapt to the dynamics of GA population in the evolution process. Moreover, each variable in the function normally has different convergence speed due to its varied sensitivity therefore to reduce the intervals for every variable in the same pre-defined rate obviously is not optimal.

In this paper a new search space zooming technique will be proposed. Other than setting the variable interval reduction rate 'a priori', update of the boundaries for each variable is based on its distribution characteristics over the whole population during the evolution process. Using this distribution based adaptive zooming method the search space can be updated dynamically in accordance with convergence speed of the GA population, therefore named adaptive zooming technique for genetic algorithms (AZGA).

2. THE NEW ADAPTIVE ZOOMING TECHNIQUE FOR GENETIC ALGORITHMS

In AZGA, the new interval for each variable is updated according to its distribution over the current population,

$$\begin{cases} g_u(k+1) = g_\mu(k) + b \cdot [g_u^{(\beta)}(k) - g_\mu(k)] \\ g_l(k+1) = g_\mu(k) - b \cdot [g_\mu(k) - g_l^{(\beta)}(k)] \end{cases} \quad (1)$$

where $g_u(k)$ and $g_l(k)$ denote the upper and lower boundaries of variable g at the k^{th} generation respectively, $g_\mu(k)$ denotes the mean value of the variable g over the whole population at the k^{th} generation, $b>1$ is some pre-determined positive constant referred to as the *zooming factor*, b is used to control the reduction rate of the interval and to reduce the risk of missing the true optimal point. $\beta \in (0, 1]$ is referred to as the *zooming fraction,* say if it is set to be 0.95, then 95 percent of the values for variable g on the whole population should be contained within the new boundaries. $g_u^{(\beta)}(k)$ and $g_l^{(\beta)}(k)$ are determined by:

$$\begin{aligned} & g_u^{(\beta)}(k) - g_l^{(\beta)}(k) = \min_i \{ g_{i+N_\beta} - g_{i+1} \} \\ & 0 \le i \le (N_{Pop} - N_\beta) \end{aligned} \quad (2)$$

where N_{Pop} is the population size, $\{ g_i, i = 1, \cdots, N_{Pop} \}$ contains all the values for variable g in the whole population in increasing order. $N_\beta = [\beta \times N_{Pop}]$ is the rounded integer of $\beta \times N_{Pop}$. Obviously $[g_l^{(\beta)}(k), g_u^{(\beta)}(k)]$ is the shortest interval containing N_β values for variable g, or $\beta \times 100\%$ values for variable g in current whole population, therefore the most densely distributed interval for variable g.

According to (2), only $(N_{Pop} - N_\beta + 1)$ values for variable g at the two ends of the sorted array are left out. β is always set close to 1, it is therefore unnecessary to sort all the values in the whole population as this is computationally expensive when the population size is large. Instead only the $(N_{Pop} - N_\beta + 1)$ largest values and the $(N_{Pop} - N_\beta + 1)$ smallest values for variable g over the whole population need to be identified, this will therefore significantly reduce the computation effort.

Statistically speaking, the interval $[g_l^{(\beta)}(k), g_u^{(\beta)}(k)]$ identified in (2) is the greatest distribution density region for variable g over the whole population, and therefore the larger the variable variance, the larger the interval $[g_l^{(\beta)}(k), g_u^{(\beta)}(k)]$. According to (1), the centre $\bar{g}(k+1)$ of the updated interval and its length for variable g are defined as

$$\begin{cases} \bar{g}(k+1) = [g_u(k+1) + g_l(k+1)]/2 \\ = g_\mu(k) + b \cdot [\bar{g}^{(\beta)}(k) - g_\mu(k)] \\ g_u(k+1) - g_l(k+1) \\ = b \cdot [g_u^{(\beta)}(k) - g_l^{(\beta)}(k)] \end{cases} \quad (3)$$

where $\bar{g}^{(\beta)}(k) = [g_u^{(\beta)}(k) + g_l^{(\beta)}(k)]/2$.

For a successful GA searching, if all the variables converge to their optimum as the evolution proceeds,

i.e. $g_\mu(k) \to g^*$, $g_u^{(\beta)}(k) \to g^*$, and $g_l^{(\beta)}(k) \to g^*$ when k is large enough, then, according to (3), we have $\bar{g}(k) \to g^*$, $g_u(k) \to g^*$ and $g_l(k) \to g^*$. This implies that the update of boundaries is in synchronisation with the converging speed of GA population. This encourages the GA search to focus around the true optimum point in the search space at the late stage of evolution process. Another technique used in this algorithm is the introduction of the zooming factor *b (>1)* in (1), which helps to reduce the risk of missing the true optimal point at early stage of evolution process due to gradual narrowing of the search space, or inappropriate initial setting of boundaries.

Similar to SZGA, in AZGA update of the variable intervals is only triggered after a certain number of generations say n_{g0} after start, and then performed for every n_g generations afterwards, under the consideration that update of the boundaries taking place immediately after each generation may lead to a premature of the GA searching.

Table 1 The test functions

index	Name	Search space	Dimension (n) tested	Global minimum
f_1	F15n	$[-10, 10]^n$	100	0 at $x_i = 1$
f_2	F5n	$[-10, 10]^n$	100	0 at $x_i = -1$
f_3	Brown3	$[-1, 4]^n$	20	0 at $x_i = 0$
f_4		$[-10, 10]^n$	100	0 at $x_i = 0$
f_5	Griewank	$[-600, 600]^n$	30	0 at $x_i = 0$
f_6		$[-10, 10]^n$	30	0 at $x_i = 0$
f_7	Ellipsoid	$[-10, 10]^n$	30	0 at $x_i = 0$
f_8	Cigar	$[-10, 10]^n$	30	0 at $x_i = 0$

3. TESTS

3.1 The test functions

To evaluate the performance of the proposed adaptive zooming technique, a suit of benchmark test functions have been used which are listed table 1. These functions have been widely used in the literature, and the reported results can be directly used for comparison.

These functions are defined as follows.

$$f_1(\boldsymbol{x}) = \frac{1}{10}[\sin^2 3\pi x_1 + \sum_{i=1}^{n-1}(x_i - 1)^2(1 + \sin^2 3\pi x_{i+1}) + (x_n - 1)^2(1 + \sin^2 2\pi x_{i+1})] \quad (4)$$

$$f_2(\boldsymbol{x}) = \frac{\pi}{n}[10\sin^2 \pi y_1 + \sum_{i=1}^{n-1}(y_i - 1)^2(1 + 10\sin^2 \pi y_{i+1}) + (y_n - 1)^2] \quad (5)$$

where $y_i = 1 + 0.25(x_i + 1), i = 1, \cdots, n$

$$f_3(\boldsymbol{x}) = \sum_{i=1}^{n-1}\left[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}\right] \quad (6)$$

$$f_4(\boldsymbol{x}) = \sum_{i=1}^{n}(0.2x_i^2 + 0.1x_i^2 \sin 2x_i) \quad (7)$$

$$f_5(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos(x_i/\sqrt{i}) + 1 \quad (8)$$

$$f_6(\boldsymbol{x}) = \sum_{i=1}^{n}|x_i| + \prod_{i=1}^{n}|x_i| \quad (9)$$

$$f_7(\boldsymbol{x}) = \sum_{i=1}^{n} i x_i^2 \quad (10)$$

$$f_8(\boldsymbol{x}) = x_1^2 + 100000\sum_{i=2}^{n}x_i^2 \quad (11)$$

Among these functions f_1, f_2, f_4,and f_5 are multi-modal and have many local minima, for example f_1 has 60n (where n is the dimension) local minima in the search space which is most challenging. Function f_7 is an ellipsoid with different eigenvalues on each axis, and f_8 is also an ellipsoid in which all but one eigenvalue are the same.

3.2 Selection and tuning of GA parameters

In this study, AZGA together with the SZGA and a simple GA (SGA) were tested and their performances were compared. Each algorithm had a set of parameters to be determined, and SGA was the simplest and has the least number of parameters. All other algorithms employed additional operations, but all were based on SGA, therefore it is necessary to first describe the SGA operations and how the parameters in SGA were selected.

In SGA real encoding scheme was employed, that is, for the continuous optimisation problems, each variable appeared in the chromosome representation as a floating-point code segment. SGA employed one population, with the initial random population uniformly generated within the search space. Linear ranking was employed to map the objective function value to the chromosome fitness. Selection was performed using the Stochastic Universal Sampling (SUS). The selected chromosomes were then collected into a mating pool, on which genetic crossover operation was performed. Intermediate recombination was employed as the genetic crossover operation, by which offspring are produced as:

$$\begin{cases} child_1 = a \cdot parent_1 + (1-a) \cdot parent_2 \\ child_2 = (1-a) \cdot parent_1 + a \cdot parent_2 \end{cases}$$

where *a,* referred to as the *blending factor*, is a randomly generated number with a uniform distribution within an interval $[a_L, a_U]$. The crossover rate was set to be $P_c = 1.0$. After crossover, the original population was replaced with the new chromosomes in the mating pool. No mutation operation was performed, i.e. the mutation rate P_m = 0. In addition, to prevent the best chromosomes in the current population failing to survive to the next generation, an *elitist strategy* was adopted. One or a few of the best chromosomes, say *Ne*, were reserved and copied directly into the next generation, and *Ne* was set at 1 in this paper.

Table 2 Success-rate of SGA with different parameter settings over function f_1 (%)

Pop size (N_{Pop})	Distribution interval $[a_L, a_U]$ for the blending factor					
	[-0.4, 1.4]		[-0.5, 1.5]		[-0.6, 1.6]	
	P_s= 1.8	P_s= 2.0	P_s= 1.8	P_s= 2.0	P_s= 1.8	P_s= 2.0
200	75	80	93	95	0	0
240	86	85	97	92	0	0
280	90	90	96	99	0	0
320	86	91	95	99	0	0
360	94	90	77	**100**	0	0
400	95	95	75	100	0	0

Except for the parameters described above, other parameters in SGA including the blending factor interval $[a_L, a_U]$, population size N_{Pop} and selection pressure P_s were tuned against function f_1 as f_1 has the largest number of local minima among all the functions and is most difficult to be optimised among all functions. To tune these parameters, a number of settings of values for GA parameters were tested. For each setting of parameters, SGA was tested for 100 times of run over f_1 in searching the global optimum. It should be noted that each of the 100 runs was driven by different random number series and each run terminated after 400 generations of evolution. The best chromosome in the population after 400 generations of evolution in each run was examined and if the solution was within area $[1+1/6, 1-1/6]^{100}$, then the search was regarded as a successful run (note that within area $[1+1/6, 1-1/6]^{100}$ function f_1 has only one minimum which is the global one). The success-rate (the number of successful runs out of 100) for each parameter setting was recorded and listed in Table 2.

In tuning of GA parameters, the success-rate and the population size N_{Pop} were the two important factors to consider. If the number of generations in GA searching were fixed in all runs, then the computation complexity depends on the population size. The larger the population size, the more time-consuming the GA searching is. Therefore, among all sets of GA parameters it was desirable to select the one that had the minimum population size with success-rate 100%. According to Table 2, the best set of parameters for SGA are the population size N_{pop}=360, the distribution interval $[a_L, a_U]$ for the blending factor [-0.5, 1.5], and the selection pressure P_s = 2.0. Along with other parameters (the number of generations N_{gen} = 400, the mutation rate P_m = 0, the crossover rate P_c = 1 and the elitist number *Ne* = 1), all the 100 runs using SGA succeed in locating the global minimum of function f_1. Therefore these parameters were used throughout for SGA in the following tests. Except for the population size, all other parameters selected for SGA were applied to the SZGA and AZGA. Only the parameters special to each of the three algorithms were tuned.

For SZGA, similar tests were performed over function f_1 with different settings of the population size N_{Pop}, the sub-generation number N_{Sub}, and the zooming factor *a*. These parameters were chosen as: N_{Sub} = 250, *a* = 0.02, N_{Pop} = 120.

For AZGA, similar tuning method was applied and the parameters were: the population size N_{Pop} = 240, the zooming fraction $\beta = 0.97$, the zooming factor *b* = 1.15, and variable interval update frequency $n_g = 4$ (i.e. interval update every 4 generations).

All algorithms used in this paper have the mechanism to generate new individuals outside the current search space, which helps to reduce the risk of missing the global optimum when it is located far away from the centre of the search space.

Table 3 Average minimum values of the 100 runs for each function when each algorithm is applied

	SGA	SZGA	AZGA
f_1	1.66E-04	7.88E-07	6.74E-07
f_2	6.18E-06	2.80E-03	2.59E-08
f_3	9.15E-24	1.44E-18	1.71E-24
f_4	5.98E-04	2.29E-02	2.01E-06
f_5	3.61E-14	1.13E-03	1.42E-15
f_6	1.18E-09	8.05E-08	2.95E-10
f_7	1.83E-15	2.18E-12	1.06E-16
f_8	1.12E-11	1.06E-08	6.03E-13

3.3 Test case 1

In this test case, the three algorithms, namely the SGA, SZGA, and AZGA were used to optimise the 8 test functions. Like most tests used in the literature, the initial search space in this case was set in a way that the true minimum of the test function was quite close to the centre of the search space. For each algorithm, 100 runs were performed over each test

function. The average minimum values of the 100 runs for each algorithm are summarised in Table 3. Note that in this test case, the global minimums for the test functions are all 0s, and each algorithm was performed for 200 generations in each run.

3.4 Computation complexity

The computation complexity in GA based optimisation is dominated by the objective function evaluation in the evolution process. The computation complexity of the three algorithms therefore can be roughly estimated using the total number of function evaluations in each search, i.e. the production of population size N_{Pop} and generations N_{Gen}. For the three algorithms, the average time for each of the 100 runs over function f_1 was measured. The computation costs of the three algorithms are listed in Table 4. The algorithms were programmed in C++ and the objective functions were written in Matlab, and the run times were measured on a desktop PC (P4 2.8GHz, 512M RAM) with Microsoft Windows XP operating system.

Table 4 Comparison of computation complexity

	N_{Pop}	N_{Gen}	Number of evaluations	Average run time over f_1 (second)
SGA	360	400	144000	5.13
SZGA	120	1000	120000	4.80
AZGA	240	400	96000	5.04

For SZGA, an update operation of the boundaries for each variable and generation of a new random population were performed at each 250 generations. In addition, more generation means more selection operations. For AZGA, an update operation of the boundaries for each variable was performed at each 4 generations. This mainly included operations like sorting the whole population in order to identify the 8 (1 plus 3% of 240) maximal and minimal values and the search for the minimum distribution interval.

3.5 Test case 2

In test case 1, the initial search spaces were set in such that the global minimum was close to the centre of the initial intervals. This search space setting method was adopted in most literatures. It can make GAs easily converge to the true minimum, as the initial population is uniformly generated within the initial search space and the mean values of the whole population are quite close to the centre of the search space. Therefore for the first test case, the precision of algorithms over 100 tests is mainly concerned.

In practice, the optimal solution is not necessarily close to centre of the initial search space, the ability of algorithms in adapting to different initial search space settings other than that used in case 1 need to be assessed. Two initial search space settings (referred to as settings I, and II respectively) were used. In initial search space setting I, the global minimum is half way from the centre of the initial search space. In initial search space setting II, the global minimum is quite close to the edge of the initial search space. For this two settings, the algorithms can not always locate the right region of the global optimum, therefore the success-rate of the algorithms are mainly concerned in this test case. Table 5 compares success-rate of the three algorithms over functions f_1 and f_2 with the two different initial search space settings. The global minimum for function f_1 is at $x_i = 1$, $i=1,...,100$, and for f_2 the global minimum is at $x_i = -1$, $i = 1,\ldots,100$.

Table 5 Success rates (%) of the three algorithms

Algorithm	Function f_1		Function f_2	
	I: $[-14, 6]^n$	II: $[-18, 2]^n$	I: $[-6, 14]^n$	II: $[-2, 18]^n$
SGA	99	99	100	100
SZGA	96	98	94	88
AZGA	99	99	100	99

3.6 Dynamics of the algorithms

To investigate dynamics of the whole population in the evolution process, statistic information such as the mean values of variables, boundaries and centres of variable intervals, and the best variable values in the whole population, are plotted. Fig.1~ Fig. 3 are typical curves taken from one of the 100 runs when the three algorithms were applied to test function f_1 with initial search space setting II. The average in the figures is the mean value of variable x_1 over the whole population in each generation. Boundaries in the figures refer to the upper and lower boundaries for variable x_1, and the centre refers to the interval centre of variable x_1 in each generation. The global minimum value for variable x_1 in function f_1 is also displayed in the figures.

For SZGA, the centre of variable intervals was always close to the best value, and the interval reduced at a constant rate in the whole evolution process (Fig. 2). In SZGA, update of variable boundaries was based on the best value after N_{sub} (say, N_{sub}=100) generations. However as shown in Fig.1 ~ Fig. 3 that the curve of the best values for a variable was rather erratic, this means that the variable boundaries along the whole evolution process formed a zigzag route which make lead the SZGA miss the true optimum.

For AZGA, the variable boundaries were updated dynamically based on the statistic distribution of values for the specific variable over the whole population (Fig. 3). Unlike SZGA, the variable interval in AZGA was not constantly decreasing, and

an increase in the variable interval implied an increase of the distribution density of values for the specific variable in the whole population, and this often happened when the GA searching jumped out of a local minimum.

Finally, the convergence speed of the three algorithms over function f_1 is graphically illustrated in Fig. 4, for which the y-axis represents the distance of the solution to the true minimum, and the x axis is the number of objective evaluations ($\times$ 1000).

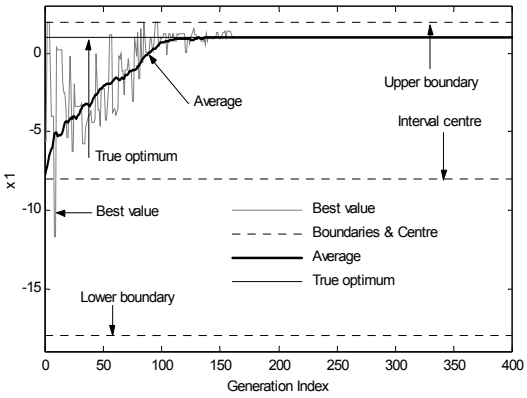

Fig. 1. Convergence curve for SGA when applied to f_1 with variable intervals [-18,2]

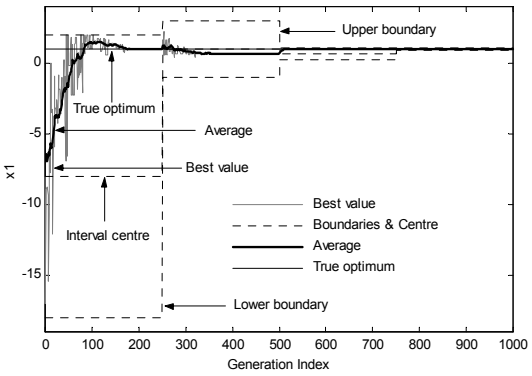

Fig. 2. Convergence curve for SZGA when applied to f_1 with variable intervals [-18,2]

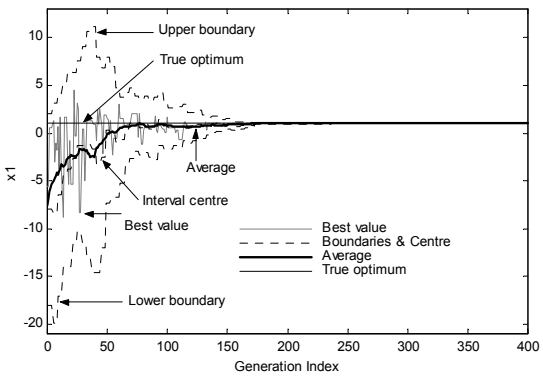

Fig. 3. Convergence curve for AZGA when applied to f_2 with variable intervals [-18,2]

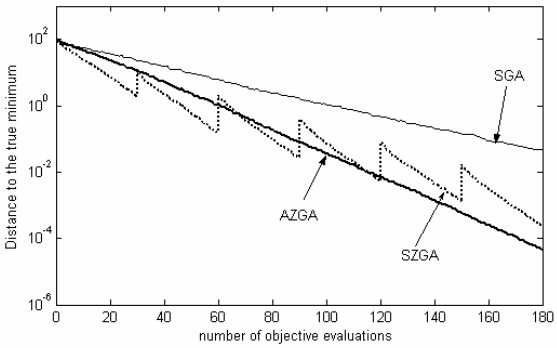

Fig. 4. Convergence speed of the four algorithms over f_1 with $\boldsymbol{x} \in [-10,10]^{100}$

4. CONCLUSION

A new search-space-update technique has been proposed to improve the performance of GAs for continuous optimisation problems. Other than setting a constant reduction rate 'a priori', the proposed technique dynamically adjusts the search space based on the statistic information of the population. Test results have shown that the proposed technique could effectively speed up the convergence thus improve the solution accuracy, and can also adapt to different initial search space settings effectively.

ACKNOWLEDGEMENTS

Dr K. Li wishes to acknowledge the financial support of the UK Engineering and Physical Sciences Research Council (EPSRC Grant GR/S85191/01).

REFERENCES

Goldberg, D.E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, Inc.

Kwon, Y., S. Kwon, S. Jin, J. Kim (2003). Convergence Enhanced Genetic Algorithm with Successive Zooming Method for Solving Continuous Optimization Problems. *Computers & Structures*, **81**, pp. 1715-1725.

Man, K.F., K.S. Tang and S. Kwong (1999). Genetic Algorithms. Springer-Verlag London Limited.

Peng, J, S. Thompson, K. Li (2002). A gradient-guided niching method in genetic algorithm for solving continuous optimisation problems. Proceedings of the 4th World Congress on Intelligent Control and Automation, Vol 4 , pp. 3333 – 3338, Shanghai, 2002.

Sabatini, A.M. (2000). A hybrid genetic algorithm for estimating the optimal time scale of linear systems approximations using Laguerre models. *IEEE Transactions on Automatic Control*, **45**, pp. 1007-1011.