Choice of Memes In Memetic Algorithm

Zhu Ning, Y. S. Ong, K. W. Wong and M. H. Lim
School of Computer Engineering, Nanyang Technological University, Singapore
{PG04169798, ASYSONG, ASKWWONG} @ntu.edu.sg
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
EMHLIM@ntu.edu.sg

Abstract

One of the fastest growing areas of evolutionary algorithm research is the enhancement of genetic algorithms by combination with local search methods or memes: often known otherwise as memetic algorithms. However there is often little theoretical basis on which to characterize the choice of memes that lead to successful memetic algorithm performance. In this paper, we investigate empirically the use of different memes in the memetic algorithms across a variety of benchmark test functions for function optimization. Empirical results show that the choice of memes affects the search performance significantly. Further, an investigation on the random choice of memes at each decision point during the memetic algorithm search arrives at interesting results.

1. Introduction

Many researchers [1] have shown that genetic algorithms (GAs) perform well for global searching, but they usually take a relatively long time to converge to the optimum. On the other hand, local search methods can quickly find the local optimum of a small region of the search space, but are typically poor global searchers. Therefore, local search methods have been incorporated into GAs in order to improve their performance through what could be termed as learning. Such hybrid GAs often known as memetic algorithms have been used successfully to solve a wide variety of realistic problems.

Davis [2] argues that genetic algorithms when hybridized with the most successful local search methods or memes for a particular problem give the best of both worlds. If implemented correctly, these algorithms should do better than the traditional genetic algorithms or local search alone. Nevertheless this also means that unless one correctly choose the right meme, a memetic algorithm may not perform at its optimum, or it may even be worse than using the genetic algorithm or the local improvement procedure itself.

On scheduling problems, Cowell et al. [3] recently coined the term hyper-heuristic to describe the idea of adaptively controlling several low-level heuristic search methods for use on each individual. But not much work has been carried out on the choice of successful memes in memetic algorithms in the area of function optimization. In this paper, our key focus is to investigate the choice of memes in hybrid genetic algorithm-local search or memetic algorithms. This paper is organized as follows: Section 2 describes and outlines the memetic algorithms. Section 3 presents and compares the search performance of various memetic algorithms on a large number of benchmark test functions commonly used in function optimization. In addition, the investigation results for a simple scheme of randomly choosing a local search method at each decision point is also presented in section 3. Section 4 summarizes the conclusions.

2. Memetic Algorithms

The basic steps of the canonical memetic algorithms based on GAs are outlined in Figure 1. In the first step, the GA population is initialized randomly. Subsequently, for each chromosome in the population, a meme is used for local improvement based on the Lamarckian evolution. Standard GA operators are then used to generate the next population until the stopping conditions are satisfied.

Procedure Canonical Memetic Algorithm

Begin
Initialize: Generate an initial GA population;
  While (Stopping conditions are not satisfied)
    Evaluate all individuals in the population
    For each individual in the population
      • Perform local search on it and replace it with locally improved solution
    End For
Apply standard genetic algorithm operators to create a new population.
    End while
End

Figure 1: Canonical Memetic Algorithm Pseudo-code

2.1. Genetic Algorithms

Genetic algorithms, based on the Darwinian survival-of-the-fittest theory, are efficient and broadly applicable global algorithm. They are a family of computational models inspired by evolution. The three basic operators of genetic algorithms are: selection, crossover and mutation. In a broader usage of the term, a genetic algorithm is any population-based model that uses these operators to generate new sample points in a search space.

2.2. Local Search Methods For Function Optimization

Schematically, a typical local search method can be sketched as in figure 2.

Procedure A Typical Local Search Method
Begin

1. Start from some given point x_1
2. Assign k=1;
3. Calculate a search direction D_k
   Determine an appropriate step length λ_k
   Replace $x_{k+1} = x_k + \lambda_k * D_k$
4. Converged?
   - Yes: stop and output results
   - No: goto step 3.

End

Figure 2: A typical Local Search Method

In general local search methods can be categorized as second, first or zeroth order techniques [4].

Second order method: This method requires the function values, its first (partial) derivative vector and the second derivative matrix - the Hessian. Newton-Raphson method is an example of this kind.

First order method: The method often used when the function values and first (partial) derivative vector are available during the search. Examples of first order methods include Steepest Descent, Conjugate Gradients and Quasi-Newtonian Methods.

Zeroth order method: The main feature of the method is that they only need the object function values, or even only need the relative rank of objective values. Example of such method is direct search techniques.

3. Empirical Study

In this section, we present and compare the search performance of various memetic algorithms on a number of benchmark test functions commonly used in function optimization. The memes or local search methods employed consist of both derivative and non-derivative methods. Further, we investigate the simple scheme of randomly choosing a local search method at each decision point of the memetic search.

3.1. Benchmark Problems for Function Optimization

The eight benchmark tests presented in table 1 that are employed in the study. They have diverse properties in term of modality, constraint and continuity.

3.2. Local search methods/Memes employed

Various locals search methods or memes from the OPTION suite [5] are employed in the empirical study. They consist of a variety of optimization methods from standard libraries [6] and others specially developed for suite. The eight local search methods used here are listed in table 2.

Bcl	Bit Climbing Algorithm by Davis [7].
Dscp	Schwefel library strategy of Davis, Swann and Compey [6,8].
Fib	Schwefel library Fibonacci search [6].
Fletch	Fletch's 1971 method by Siddall [9].
Gold	Schwefel Golden Section Search [6].
Pds	Powell's Direct Search Method [10].
Seek	Hooke and Jeeves Direct Search [11].
Simplex	Simplex strategy of Nelder & Meade [12].

Table 2: Local search methods or memes employed

3.3. Results for Benchmark Tests

In the empirical study, we employ a standard binary coded genetic algorithm for the memetic search. All results presented are averages over ten independent runs. Each run continues until the global optimum was found or the maximum of 40,000 function evaluation calls was reached, except for the Bump function where a maximum of up to 100,000 function calls was used[1]. In each run, the control parameters for the memetic algorithm were set as follows: population of 50, mutation rate of 0.1%, 2-point crossover with a rate of 60%, 10 bit binary encoding, and maximum local search length of 100 evaluations.

The performance of the different memetic algorithms across the eight-benchmark test is presented in table 3. We computed the mean of function fitness over function

[1] The Bump constrained problem is a very hard problem and therefore requires greater effort.

evaluation calls for each benchmark test function, and then ranked them from 1 to 8 since there is eight test functions in total. From the results presented in table 3, we can observe that GA-Bcl performs best on the Step function, but poor on the Sphere function. GA-Fletch performs best on the Sphere function, but quite poorly on 4 out of the other 7-benchmark functions. GA-Simplex works best on the Foxhole function, but its performance is relatively poor on the Bump function. The overall ranking performance of GA-Dscp across all benchmark tests seems to be the best. However, on the Sphere function, its performance is much poorer than that of Fletch.

From our study, it is evident that different memes would best suit a problem most. For example, the derivative local method, Fletch, performs best on continuous and symmetric functions, like the Sphere Function. But when there are break points in a function, it can be unusable. These characteristics make generalization in this field difficult and also the a priori selection of particular meme in a canonical memetic algorithm to suit a black box problem almost impossible. In the subsequent sub-section, we carry out further an investigation on randomly choosing a local search method or meme at each decision time during the memetic algorithm search lifetime.

Descriptions	**Benchmark Test Functions**	n (dimension)	S (Range of x_i)	*Global Optimum*
It is a continuous, unimodal, symmetric function.	$F_{bump} = \frac{abs\left[\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)\right]}{\sqrt{\sum_{i=1}^{n} i x_i^2}}$ $\prod_{i=1}^{n} x_i > 0.75$ and $\sum_{i=1}^{n} x_i \langle 15n/2$	20	$[0,10]^{20}$	≅0.81 (maximum)
It is the representative of the problem of flat surfaces. Flat surfaces do not give any information as to which direction is favorable, thus making them difficult for optimization algorithms.	$F_{foxhole} = 0.002 + \sum_{j=1}^{n} \frac{1}{c_j + \sum_{i=1}^{2}\left(x_i + \alpha_{ij}\right)^8}$ $a_{1i} = \begin{pmatrix} -32,-16,0,16,32,-32,-16,0, \\ 16,32,-32,-16,0,16,32,-32, \\ -16,0,16,32,-32,-16,0,16,32, \end{pmatrix}$ $a_{2i} = \begin{pmatrix} -32,-32,-32,-32,-32,-16, \\ -16,-16,-16,-1616,16,16, \\ 16,16,32,32,32,32,32,0,0,0,0,0 \end{pmatrix}$	2	$\begin{bmatrix} -65.536, \\ 65.536 \end{bmatrix}^2$	0.0
It is an example of many local optima.	$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{40000} - 2\prod_{i=1}^{n} \cos(x_i / \sqrt{i})$	10	$[-600,600]^{10}$	0.0
This function has a very narrow ridge. The tip of the ridge is very sharp, and it runs around a parabola.	$F_{Rastrigin} = n \times A + \sum_{i=1}^{n}\left(x_i^2 - A \times \cos(2\pi x_i)\right)$	20	$[-5.12,5.12]^{10}$	0.0
It is a non-linear multi-modal function; this function contains millions of local optima in the interval of consideration.	$F_{Rosenbrock} = \sum_{i=1}^{n-1} 100 \times \left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2$	30	$[-5.12,5.12]^{30}$	0.0
It is a non-linear multi-modal function. It is characterized by a second- best minimum that is far away from the global optimum.	$F_{Schwefel} = 418,9829 * n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	10	$[-500,500]^{10}$	0.0
It is a non-linear multi-modal function. This function is high dimension multimodal function with many local minima.	$F_{Sphere} = \sum_{i=1}^{n} x_i^2$	30	$[-5.12,5.12]^{30}$	0.0
This function has two non-linear inequality constraints. It gives a highly bumpy surface where the true global optimum is usually defined by the product constraint. It contains many peaks, all of similar heights and has strong inter-parameter linkage.	$F_{step} = 6 \times n + \sum_{i=1}^{n} \lvert x_i \rvert$	5	$[-5.12,5.12]^{5}$	0.0

Table 1: Benchmark Tests

Memetic Algorithms	*Bump Function*	*Foxhole Function*	*Griewank Function*	*Rastrigin Function*
	Ranking	Ranking	Ranking	Ranking
GA-Bcl	2	1	5	6
GA-Dscp	1	4	1	1
GA-Fib	7	2	7	7
GA-Fletch	5	5	3	4
GA-Gold	6	7	8	8
GA-Pds	3	3	2	2
GA-Seek	4	8	6	3
GA-Simplex	8	6	4	5
	Rosenbrock Function	*Schwefel Function*	*Sphere Function*	*Step Function*
	Ranking	Ranking	Ranking	Ranking
GA-Bcl	5	3	6	1
GA-Dscp	1	1	2	2
GA-Fib	8	8	7	4
GA-Fletch	2	6	1	6
GA-Gold	7	7	8	3
GA-Pds	3	5	3	7
GA-Seek	4	2	5	8
GA-Simplex	6	4	4	5

Table 3: Performance Ranking of Memetic Algorithms on Benchmark Problems

3.4. Simple Random Walk Scheme

The pseudo code of the Simple Random Walk memetic algorithm is quite similar to the canonical memetic algorithm in figure 1. It differs from the canonical memetic algorithm in that the meme used for learning during each Lamarckian evolution is randomly chosen from the pool of 8 memes employed in the search at each decision point.

The empirical results of the Simple Random Walk memetic algorithm in comparison with the 8 canonical memetic algorithms across the benchmark problems are presented in table 4. We illustrate the relative ranking of the Simple Random Walk in comparison with the other eight-memetic algorithms. Further we compute the average ranking of each memetic algorithm across all the benchmark functions with equal weights to get the overall performance rankings (see table 5). The performance of the best and worst performing memetic algorithms and the simple random walk memetic algorithm on each function are shown in figures 3-10. The empirical results show that the Simple Random Walk algorithm performs relatively well across all the benchmark problems. In fact, across all the 8-benchmark problem, the simple random walk memetic algorithm emerges as having the best overall ranking performance when compared to all 8 canonical memetic algorithms used in the investigation. The distributions across the various benchmark problems (see figure 11) in the Simple Random Walk Scheme indicate that although the meme selection process is stochastic at each decision point, the choice of each meme in the entire search of each benchmark problems appears to be near uniformity.

	Bump Function	*Foxhole Function*	*Griewank Function*	*Rastrigin Function*
	Rank	Rank	Rank	Rank
Simple Random Walk	2	1	4	2
	Rosenbrock Function	*Schwefel Function*	*Sphere Function*	*Step Function*
	Rank	Rank	Rank	Rank
Simple Random Walk	1	2	2	2

Table 4: Performance Ranking of the Simple Random Walk Memetic Algorithm in Comparison to the 8 Canonical Memetic Algorithms

	GA-BCL	*GA-DSCP*	*GA-FIB*	*GA-FLETCH*
Overall performance ranking	4	2	8	5
	GA-GOLD	*GA-PDS*	*GA-SEEK*	*GA-SIMPLEX*
Overall performance ranking	9	3	6	7
	The Simple Random Walk			
Overall performance ranking	1			

Table 5: Overall performance ranking of the different Memetic Algorithms across Benchmark Problems

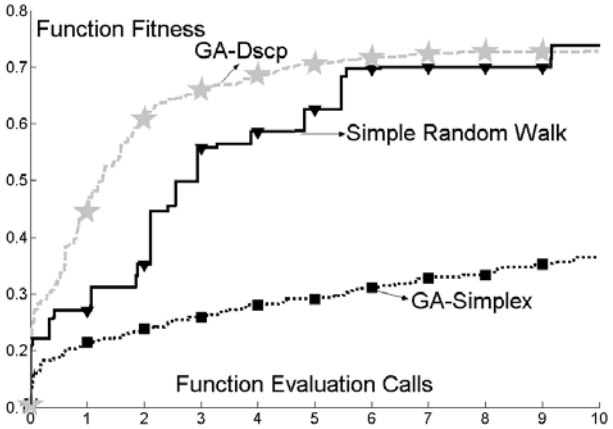

Figure 3: Search Traces (average of 10 runs) for maximizing 20D Bump Function.

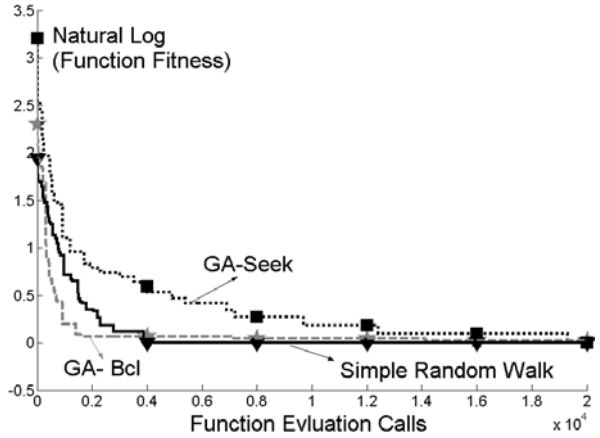

Figure 4: Search Traces (average of 10 runs) for minimizing 2D Foxhole Function.

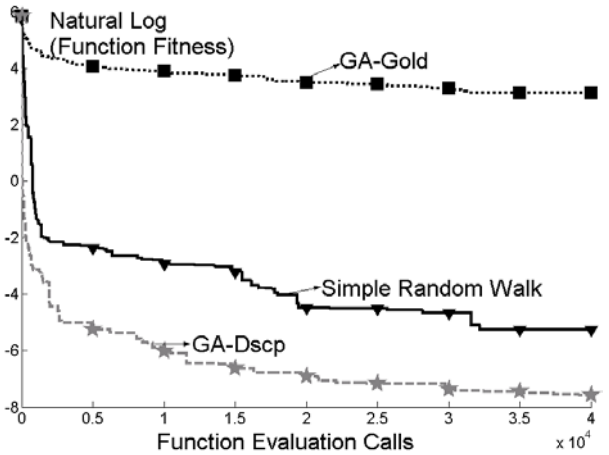

Figure 5: Search Traces (average of 10 runs) for minimizing 10D Griewank Function.

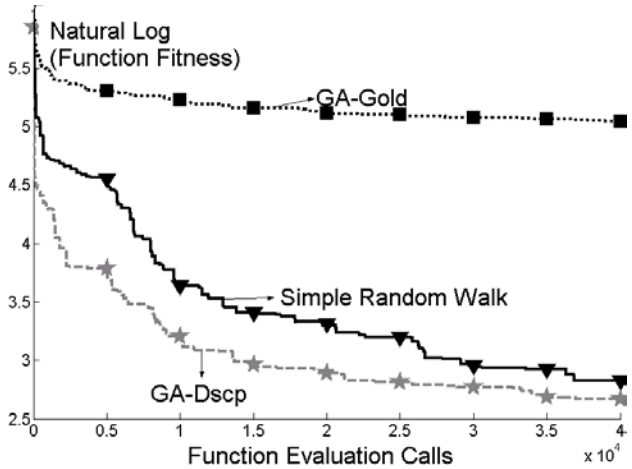

Figure 6: Search Traces (average of 10 runs) for minimizing 20D Rastrigin Function.

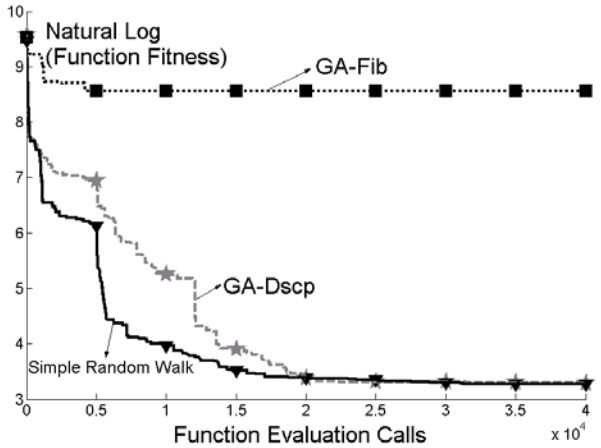

Figure 7: Search Traces (average of 10 runs) for minimizing 30D Rosenbrock Function.

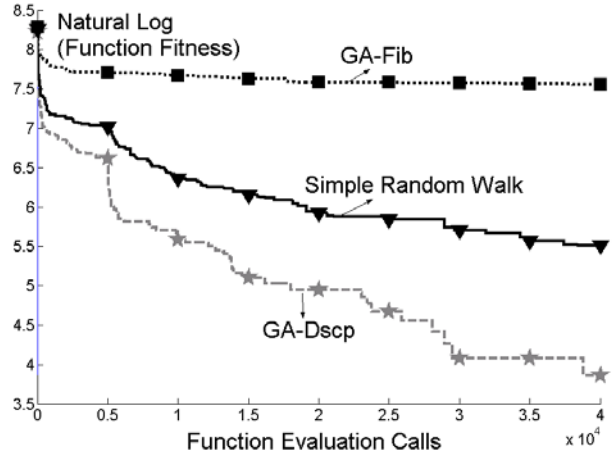

Figure 8: Search Traces (average of 10 runs) for minimizing 10D Schwefel Function.

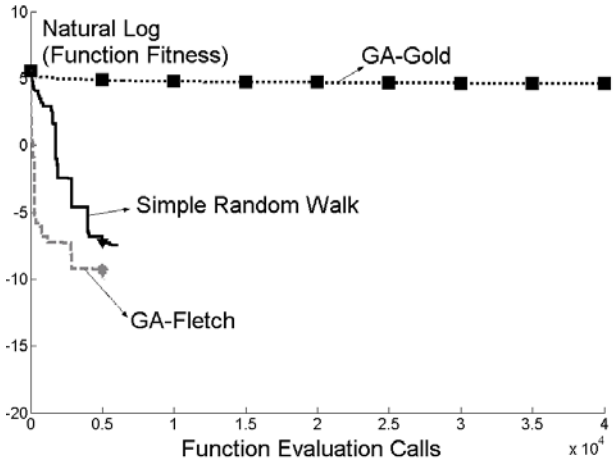

Figure 9: Search Traces (average of 10 runs) for minimizing 30D Sphere Function.

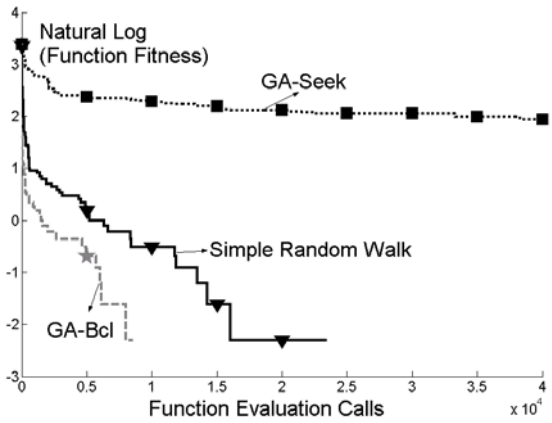

Figure 10: Search Traces (average of 10 runs) for minimizing 5D Step Function.

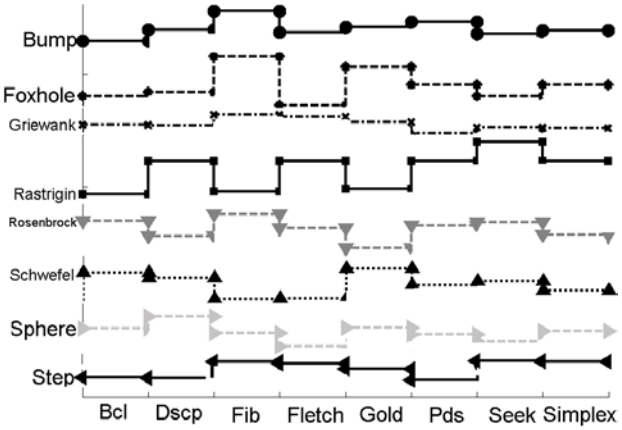

Figure 11: Distribution of choice of memes in the Simple Random Walk Memetic Algorithm on the Benchmark Problems, i.e., a plot of Distributions versus Memes.

4. Conclusions

In this paper, we presented an empirical investigation on the use of different local search methods or memes in the memetic algorithms across a variety of benchmark test functions for function optimization. We show that the choice of memes affects the search performance significantly. No single memetic algorithm always performs best on the diverse set of benchmark test functions. Further, an investigation on Simple Random Walk memetic algorithm that randomly selects a meme at each decision point during the search was conducted. The empirical results show that the simple random walk memetic algorithm performs relatively well and robust across all the benchmark problems and overall emerges as being superior to all the other canonical memetic algorithms employed. Hence, we believe that there is much evidence to warrant additional studies on the adaptive selection of memes, i.e., via some intelligent means while the memetic search is progressing.

Acknowledgements

This research work is supported by NTU/SCE grant number CE-SUG 3/03. The authors would like to thank the two research centers, Parallel and Distributed Computing Centre, Centre for Multimedia and Network Technology of Nanyang Technological University for their support in this work.

References

[1] Z. Michalwicz, *Genetic Algorithms + Data Structure= Evolution Programs*, AI Series, Springer-Verlag, New York, 3rd edition, 1996.

[2] L. Davis, *The handbook of Genetic Algorithms*, Van Nostrand Reingold, New Tork, 1991.

[3] P.I. Coweling, G. Kendall and E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, *3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT)*, pp.176-190, 2001.

[4] B. D. Bunday, *Basic Optimization Methods*, Edward Arnold (Publishers) Ltd, 1985.

[5] A.J. Keane, The OPTIONS Design Exploration System User Guide and Reference Manual, 2002, http://www.soton.ac.uk/~ajk/options.ps

[6] H. P. Schwefel, *Evolution and Optimum Seeking*, John Wiley &Son, 1995.

[7] L. Davis, Bit Climbing, Representational Bias, and Test Suite Design, *Proceeding of the 4th International conference on Genetic Algorithms,* pp. 18-23, 1991.

[8] W. H. Swann, *A report on the development of a new direct searching method of optimization*, ICI CENTER Instrument Laboratory, Middlesborough, 1964.

[9] R.Fletcher, A new approach to variable metric algorithm,*Computer Journal*, Vol.7(4),pp.303-307, 1964.

[10] M. J. D. Powell, An efficient method for finding the minimum of a function of several variables without calculating derivatives, *Computer Journal*, Vol. 7(4), pp. 303-307, 1964.

[11] J. L. Horner, R. G. White, R. Hooke and T. A. Jeeves, *Direct search solution of numerical and statistical problems*, J. Sound Vib. 147(1) pp. 87-103, Westinghouse Research Labs. Scientific Paper, 1960.

[12] J. A. Nelder and R. Meade, A simplex method for function minimization, *Computer Journal*, Vol. 7 pp. 308-313, 1965.