Meta-Lamarckian Learning in Memetic Algorithms

(Y.S. Ong and A.J. Keane)*
Y.S.Ong@soton.ac.uk, Andy.Keane@soton.ac.uk

ABSTRACT

Over the last decade, memetic algorithms have relied on the use of a variety of different methods as the local improvement procedure. Some recent studies on the choice of local search method employed have shown that this choice significantly affects the efficiency of problem searches. Given the restricted theoretical knowledge available in this area and the limited progress made on mitigating the effects of incorrect local search method choice, we present strategies for memetic algorithm control that decide, at runtime, which local method is chosen to locally improve the next chromosome. The use of multiple local methods during a memetic algorithm search in the spirit of Lamarckian learning is here termed Meta-Lamarckian Learning.

Two adaptive strategies for Meta-Lamarckian learning are proposed in the paper. Experimental studies with Meta-Lamarckian learning strategies on continuous parametric benchmark problems are also presented. Further, the best strategy proposed is applied to a real world aerodynamic wing design problem and encouraging results are obtained. It is shown that the proposed approaches aid designers working on complex engineering problems by reducing the probability of employing inappropriate local search methods in a memetic algorithm, while at the same time, yielding robust and improved design search performance.

Keywords: Continuous Parametric Design Optimization, Hybrid Genetic Algorithm–Local Search, Memetic Algorithm and Adaptive Meta-Lamarckian Learning.

---

* Manuscript received ; Y. S. Ong is with the Division of Information Systems, School of Computer Engineering at Nanyang Technological University, Block N4, 2b-39, Nanyang Avenue, Singapore 639798. A. J. Keane is with the Computational Engineering and Design Center, School of Engineering Sciences, University of Southampton, Highfield, Southampton, SO17 1BJ, United Kingdom.

I. INTRODUCTION

Genetic algorithms (GAs) are a powerful set of global search techniques that have been shown to produce very good results on a wide class of problems. GAs are capable of exploring and exploiting promising regions of the search space. They can, however, take a relatively long time to locate the local optimum in a region of convergence (and may sometimes not find the optimum with sufficient precision).

Torn and Zilinskas [1], in the section entitled *Global search methods: exploration and exploitation*, observe that two competing goals govern the design of global search methods: exploration is important to ensure global reliability; i.e., every part of the domain is searched enough to provide a reliable estimate of the global optimum; exploitation is also important since it concentrates the search effort around the best solutions found so far by searching their neighborhoods to produce better solutions. Many search algorithms achieve these two goals using a combination of dedicated global and local searches. These are commonly known as hybrid methods. Hybrid Genetic Algorithm-Local Search (GA-LS) methods, which incorporate local improvement procedures with traditional GAs may thus be used to improve the performance of GAs in search. Such hybrids have been used successfully to solve a wide variety of engineering design problems and experimental studies show that GA-LS hybrids not only often find better solutions than simple GAs, but also that they may search more efficiently [2-11]. In diverse contexts, hybrid GA-LSs are also known as Lamarckian Learning GAs, Baldwinian Learning GAs and Memetic Algorithms. Since we are here concerned with evolutionary algorithms where local search plays a significant role throughout the search, the term Memetic Algorithm (MA) [2] is used in this paper.

Davis [3] argues that hybridizing genetic algorithms with the most successful local search method for a particular problem gives one the best of both worlds: correctly implemented, these algorithms should do no worse than the traditional GA or LS[1] alone. Clearly, what this implies is that unless one knows which local search method most suits the problem in hand (along with its correct parameters settings), a MA may not perform at its optimum or worse, it may perform less well than using the GA alone. The influence of the local search method employed has been shown in [4] and [10] to have a major impact on the search performance of MAs. These experiments conducted on two different local methods, demonstrated that the performance obtained by MAs can indeed be worse than that obtained by the GA or LS alone. The varied suitability of LSs to different problems also helps explain why a variety of MAs have emerged in the literature.

The significance of local search method choice on the performance of MAs is therefore not a new observation. However, little work has been done to mitigate this problem. The greatest barrier to further progress is that, with so many local search methods available in the literature, it is almost impossible to know which is most relevant to a problem when one has only limited knowledge of its cost surface before one starts. Moreover, LS(s) by themselves are known to work very differently with different design problems, even among problems from the same design domain [12]. Depending on the complexity of a design problem, local search methods that may have proven to be successful in the past might not work so well, or at all, on others - an outcome that is often referred to as the "no free lunch theorem for search" [13].

With the restricted amount of theory currently available for choosing the LS that best matches a black box problem in MA search, it is reasonable to ask whether the effects of

[1] The term Local Search and the abbreviation LS are used interchangeably in the paper.

this choice on performance might be reduced via some intelligent means *while* the search is progressing. Many adaptive search systems already exist [14-18] and have been shown to solve some problems very effectively. The adaptation of GA crossover and mutation operators has been attempted in [14,15]. General surveys of various adaptive systems in evolutionary computations and their influences can be found in [16] and [17]. Nevertheless, the operator that has perhaps the greatest influence on MA performance, but one that has yet to be thoroughly explored for adaptation is the choice of LS(s) that best matches a design problem.

In this paper, we focus on investigations into the adaptive choice of local search methods to ensure *robustness* in memetic algorithm search. In particular, we consider the general nonlinear programming problem of the form:

$$\textbf{Minimize:} \qquad f(\boldsymbol{x})$$

$$\textbf{subject to:} \qquad g_i(\boldsymbol{x}) \le 0, \qquad i = 1,..., p$$

$$\boldsymbol{x}_l \le \boldsymbol{x} \le \boldsymbol{x}_u$$

where $\boldsymbol{x} \in \Re^n$ is the vector of continuous design variables, and $\boldsymbol{x}_l$ and $\boldsymbol{x}_u$ are the lower and upper bounds, respectively. Two adaptive strategies proposed for the selection of local method from a pool of LS(s) while the design search progresses are presented. Note that here we present work based on Darwinian and Lamarckian evolution. Lamarckian learning forces the genotype to reflect the result of improvement through placing the locally improved individual back into the population to compete for reproductive opportunities [4,7,10]. The approach of using multiple LSs during a MA search in the spirit of Lamarckian learning is here termed Meta-Lamarckian learning.

The paper is organized in the following manner. Section II presents traditional Lamarckian learning, a basic Meta-Lamarckian learning scheme and two potential adaptive strategies inspired by social evolution. Section III summarizes some empirical studies on continuous parametric benchmark functions, analyses the results and recommends the more competitive Meta-Lamarckian learning strategy. Section IV considers a real world aerodynamic design application. Finally, section V provides some brief conclusions.

II. META-LAMARCKIAN LEARNING

Traditionally, MAs with Lamarckian learning procedures are based on the use of the evolutionary algorithm and a single local search method for local improvements. Little published work has dealt with MAs using multiple local search methods in continuous parametric design optimization. Every search algorithm, except for uniform random search, introduces some kind of bias into its search. Different local search methods have different biases. It is these biases that make a method efficient for some classes of problems but not for others. Therefore the motivation for the use of multiple LSs, and thus Meta-Lamarckian learning in MA searches, is to achieve improved search performance and to reduce the probability of utilizing an inappropriate local method. This meta-learning approach is similar in spirit to the work of Krasnogor et al. [10,11], Hugo et al. [19] and Cowling et al. [20] where several heuristic methods are employed.

A. Basic Meta-Lamarckian Learning Scheme, MA-B

The most basic Meta-Lamarckian learning scheme for LS selection is a simple random walk over the available methods every time a chromosome is to be locally

improved - this does not adapt but has the advantage of at least giving all the available local methods in the LS pool a chance to improve each chromosome throughout the memetic algorithm run. It is purely stochastic in nature: each local method has an equal probability of being chosen at all GA iterations. This approach forms a base-line algorithm with which other Meta-Lamarckian learning strategies may be compared.

B. Adaptive Meta-Lamarckian Learning

Lamarckian learning in MA searches may be structured to promote cooperative, competitive, or individualistic efforts. The traditional approach in a MA where a single LS is used on the problem throughout the search is an example of individualistic effort. There has been a long history of research on these different kinds of effort in social evolution since the first study in 1898, where it was shown that cooperation and competition, as compared with individualistic efforts, typically result in higher achievement [21]. Inspired by these works, the adaptive strategies proposed here for Meta-Lamarckian learning in MA are structured to promote cooperation and competition among the different LSs, working together to accomplish the shared optimization goal.

The idea behind the adaptive strategies is that as the search progresses, the effectiveness of each LS in dealing with the current problem is learnt. Knowledge about the current population of solutions and each LS is thus built dynamically online, so identifying the strengths and weaknesses of the LSs for the problem currently being worked on, *given its current state*. To promote competition among the LSs, the local methods with higher fitness improvement measures are rewarded with greater chances of being chosen for subsequent chromosome optimizations. We define cooperation as the

act of operating together. Optimization problems in science and engineering commonly have large search spaces, which contains numerous local landscapes of diverse forms. The joint operation of diverse LSs to cope with the large search space is facilitated via problem decomposition or diversity in the LS selection.

Here during Meta-Lamarckian learning, the reward η is measured using the improvements contributed by the LS to each chromosome that has been searched using

$$\eta = \beta \frac{|pf - cf|}{\mu} \qquad (1)$$

where *pf* is the initial function fitness of a parent chromosome before local search and *cf* is the final function fitness of the child chromosome obtained after applying local search. μ is the number of LS function evaluation calls made to reach the improved child chromosome or solution. Alternatively, the actual wall-clock time may be used in place of the number of LS function calls made on the parent chromosome. Further, we distinguished between absolute and relative reward in the spirit of Cowling et al. [20]. The term $\frac{|pf - cf|}{\mu}$ provides a simple measure of the rate at which the local search improves a design and is an obvious component of absolute reward measure; see [22] for such a model demonstrated on an artificial problem. β signifies the relative reward which scales the absolute reward in proportion to the method's ability to produce high quality genotypes when compared to the best global solution obtained so far. Here, β is set as $\frac{\sigma}{cf}$ or $\frac{cf}{\sigma}$, for minimization and maximization problems, respectively. σ is the best solution encountered so far in the global search. We have tried a number of terms to provide this

measure such as $\frac{1}{pf - \sigma}$, but find that the ratio of σ to *cf* gives the best performance in practice. The reward obtained by each LS on the chromosomes then influences which method is selected from the pool of available methods to proceed with the local improvement.

According to [17], adaptive systems can be classified as *deterministic*, *adaptive* or *self-adaptive*. Here two a*daptive* Meta-Lamarckian learning strategies are studied: 1) A Heuristic Approach, Sub-problem Decomposition – MA-S1 and 2) A Stochastic Approach, Biased Roulette Wheel – MA-S2.

1) Heuristic Approach, Sub-problem Decomposition – MA-S1

Sub-problem Decomposition, MA-S1, represents a heuristic approach. At the start of the strategy, each LS is given an equal probability of being chosen as the local search method to be used. The reward of the chosen local method searching on a chromosome is measured using equation (1). All parent chromosomes and selected LSs together with the rewards obtained are archived in a database that is used later to guide future LS choice. The set of parent chromosome vectors archived in the database is denoted by $P \equiv \{p_j\}_{j=1}^{m}$, where *m* is the database size at any instant of search.

Next, after some pre-defined number of generations *g* has been completed, the mechanism of sub-problem decomposition takes over. For each unseen parent chromosome, denoted by $\hat{p}$, in the GA population to be searched, the strategy locates the *k* nearest neighbors from the archived database *P*, here using a simple Euclidean distance metric. Other metrics may be used of course. This subset of *k* chromosomes in *P* is

denoted by P_k. The local search methods associated with P_k then form the local sub-pool of candidate LS's that will compete, based on their rewards, to decide on which method proceeds with the local improvement of $\hat{p}$. After local search, all $\hat{p}$ and the chosen LS, together with the reward obtained are updated into the database. See Figure 1 for a pictorial illustration and pseudo-code of the strategy.

With the choice of LS involving only the rewards for candidate LS's that are applicable in the neighborhood of the chromosome to be improved, the strategy decomposes the original problem cost surface into many sub-partitions dynamically, and attempts to choose the most competitive local search method for each sub-partition. In the same manner, it creates opportunities for joint operations between different LSs in solving the problem as a whole, because the diverse LSs help improve the overall population based on their areas of specialization. Hence, the Sub-problem Decomposition strategy promotes both cooperation and competition among the LSs during MA searches.

2) A Stochastic Approach, Biased Roulette Wheel – MA-S2

The Biased Roulette Wheel strategy, MA-S2, is a stochastic approach making use of knowledge gained on-line to form biases. During the training stage, each local method is first given a single opportunity to hybridize with the GA in optimizing unseen parent chromosomes. After this, the probability that a local method will be chosen to work on any subsequent chromosome is biased according to its previous performance, which now changes dynamically as the overall search progresses. The measurement of a local method's reward on a chromosome is again based on equation (1), and a biased roulette wheel is used to pick the subsequent local search methods, based on the rewards taken

over all previous local searches. See Figure 1b for a pictorial illustration and pseudo-code of the MA-S2 strategy.

Since the choice of LS is biased according to the reward of each local search method, the Biased Roulette Wheel strategy is generally a competitive strategy. Likewise, the stochastic nature of the strategy guarantees diversity in the LS selection, hence restraining any LS from complete domination throughout the search. By ensuring diverse LS methods participation in the problem search, the strategy promotes joint operation and hence cooperation between local search methods.

III. STUDY OF META-LAMARCKIAN LEARNING ON BENCHMARK PROBLEMS

In this section, we present experimental studies on benchmark problems obtained by implementing Meta-Lamarckian Learning within a standard genetic algorithm. The basic steps of the MA search with Meta-Lamarckian Learning are outlined in Figure 2.

BEGIN
Initialize: Generate an initial GA population.
While (Stopping conditions are not satisfied**)**
Evaluation of All Individuals in the Population
For each individual in the population
- *Pseudo-code to select local search methods. For example, Select LS using the Meta-Lamarckian Learning Strategy employed.*
- Proceed with local improvement and replace the genotype in the population with the improved solution.

End For
Apply standard GA operators to create a new population; i.e., Selection, Mutation and Crossover.
End While
END

Figure 2: Proposed Framework for MA with Meta-Lamarckian Learning

In the first step, the GA population is initialized either randomly or using design of experiments techniques such as Latin hypercube sampling [23]. Subsequently, for each chromosome in the population, a local search is selected from the pool of multiple LSs, based on the Meta-Lamarckian learning strategy in use, and used for local improvement. The reward given to the local method is updated and this is followed by replacement of the genotype in the population with the locally improved solution (in the spirit of Lamarckian learning). Standard GA operators are then used to form the next population.

For the purpose of reproducibility, the GAC package developed in C by Spears [24] is employed as the Standard GA in the results presented here. A variety of local search methods were employed. These consist of various optimization methods from the Schwefel libraries [25], some briefly described by Siddall [26] with a few others available in the literature. These are both constrained and unconstrained nonlinear local search methods commonly used in engineering design optimization. Nine[2] hybrid MA-LSs are presented here: MA-BC, MA-CO, MA-DS, MA-HO, MA-FL, MA-LA, MA-NM, MA-PD and MA-PO. These abbreviations have the following meanings:

GA : Standard GA, GAC by Spears [24];
MA-AV: Expected Convergence Trace of a Traditional MA, given multiple LS methods.
MA-B : Base-line Meta-Lamarckian learning scheme for multiple LS selection.
MA-BC: GA with Bit Climbing Algorithm by Davis [27];
MA-CO: GA with Complex Method of M.J.Box as implemented by Schwefel [25];
MA-DS: GA with Davies, Swann and Campey Search with Gram-Schmidt orthogonalization as implemented by Schwefel [25];
MA-HO: GA with Hooke and Jeeves Direct Search by Siddall [26];
MA-FL: GA with Fletcher's 1972 method by Siddall [26];

[2] This pool of nine local search methods was selected to be representative of the over 30 used in the experimental studies. The local search methods that are not presented in the paper have search performances that lie between the best and worst performing memetic algorithms on these problems.

MA-LA: GA with Repeated Lagrangian Interpolation as implemented by Schwefel [25];
MA-NM: GA with Simplex Method by Nelder and Meade [28];
MA-PD: GA with Powell's Direct Search Method [29] as produced by AERE Harwell;
MA-PO: GA with Powell's Direct Search Method [29] as implemented by Schwefel[25].

A. Continuous Parametric Benchmark Test Problems

Three commonly used continuous parametric benchmark test problems already extensively discussed in the literature are used in this work[3]. They represent classes of constrained, unconstrained, unimodal and multimodal test functions summarized in Table 1. These functions make it possible to study the proposed Meta-Lamarckian learning in comparison with other approaches.

Benchmark Test Functions	**Range of x_i**	**Characteristics**				**Function Minimum At**
		Epi[*1]	**Mul**[*2]	**Disc**[*3]	**Con**[*4]	
$F_{Sphere} = \sum_{i=1}^{n} (x_i^2)$	$[-5.12, 5.12]^{30}$	none	none	none	no	0.0
$F_{Griewank} = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \left[\cos\left(x_i / \sqrt{i}\right)\right]$	$[-600, 600]^{10}$	weak	high	none	no	0.0
$F_{Bump} \quad \frac{abs\left(\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)\right)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}$ $\prod_{i=1}^{n} x_i > 0.75$ and $\sum_{i=1}^{n} x_i < 15n/2$	$[0, 10]^{20}$	high	high	none	yes	Maximum at ~ 0.81

Table 1: Classes of Constrained, Unconstrained, Unimodal and Multimodal Benchmark test functions. *1: Epistasis, *2:Multimodality, *3:Discontinuity, *4:Constrainted.

The first benchmark problem is the unconstrained unimodal Sphere function [30]. It is a smooth, symmetric function and is used here to provide a measure of the general

[3] A number of other commonly used test problems from the literature were employed in the investigation. However, due to limited space, only representatives of each benchmark problem class are presented here.

efficiency of the Meta-Lamarckian learning approach. It has a single minimum located at (0,…,0). A thirty dimensional (*n=30*) version of the sphere function is used here.

The second problem is the Griewank unconstrained function [1]. It is a high dimension multimodal function with many local minima and a global minimum located at (0,...,0). This function has inter-parameter linkage due to the presence of the product term. However, the effect decreases as the number of parameters increases. The Griewank function with 10 dimensions (*n=10*) has more than 500 local minima in the hybercube $[-600, 600]^{10}$. It has a very rugged landscape and is difficult to search for most optimizers.

The third is the Bump or Keane function [31], which is subject to two constraints. It is the most difficult for search methods/optimizers to deal with, among those considered here. This function gives a highly bumpy surface where the true global optimum is usually defined by the product constraint. It is quite smooth but contains many peaks, all of similar heights and has strong inter-parameter linkage. Its main purpose is to test how methods cope with optima that occur hard up against the constraint boundaries commonly found in engineering design. These properties make it suitable for the study of MA performance as well as in adaptive control of evolutionary optimization methods.

The surface for n=2 on the Sphere and Bump functions are shown in Figure 3a and 3c, respectively, while a one-dimensional slice of the Griewank function for $[-200, 200]^{10}$ is shown in Figure 3b.

B. Results for Benchmark Test Problems

To see how the choice of the local search method employed affects the efficiency of problem searches, the nine different local search methods used to form the traditional MAs were used to search the benchmark problems. Note that Traditional MA implies that throughout the entire search, a fixed local search method is used. The averaged convergence trends obtained for the test problems as a function of the total number of function evaluations are shown in Figures 4-6. All results presented are averages over twenty independent runs. Each run continues until the global optimum was found or a maximum of 40,000 trials (function evaluation calls) was reached, except for the Bump function where a maximum of up to 100,000 trials was used[4]. In each run, the control parameters for the hybrid MA-LS used in solving the benchmark problems were set as follows: Population size of 50, Mutation rate of 0.1%, 2-point Crossover with a rate of 60%, 10 bit binary encoding, Maximum Local Search Length[5] of 100 evaluations and the probability of applying local search on a parent chromosome is set to unity.

From the results of Figures 4-6, it is clear that the effect of local method choice on the efficiency of traditional MAs is significant. For example, MA-FL is seen to perform best on the Sphere function but very poorly on both Griewank and Bump. Moreover the majority of the nine traditional MA combinations do not show any improvement over the standard GA on the difficult Bump problem, with most having search capabilities closer to the least efficient traditional MA. In addition, the two different implementations of Powell's Direct Search (i.e., MA-PD and MA-PO) included in the investigation illustrate

[4] The Bump constrained problem is a very hard problem and therefore requires greater effort.
[5] Maximum Local Search Length refers to the maximum number of iterations or function calls allocated for local learning.

that the capability of a given local search method may differ even among different implementations of the same basic algorithm. These characteristics make generalization in this field very difficult and also the *a priori* selection of particular LS in a traditional MA to suit a black box optimization problem almost impossible.

1) Results of Meta-Lamarckian Learning

To analyze the new approaches proposed, the Meta-Lamarckian learning strategies were tested on the benchmark problems. In these experiments, the control parameters, stopping criteria, etc., were the same as in the previous experiments. The averaged convergence trends obtained for the strategies are shown in Figures 7–9. Note, in all cases, results are plotted against the total number of function evaluations calls made by the combined GA and LS searches.

The performance of the Meta-Lamarckian learning approach may be established by comparison with some of the traditional MAs and also with trace MA-AV. MA-AV represents the estimated performance one might expect to get when a traditional MA is randomly chosen for use. This is an average of the previous search traces in Figures 4-6 for the entire pool of nine traditional MA on each problem and is obtained from: MA-$\mathrm{AV}_j = \sum_{i=1}^{L} \left(fitness_{ji} \right) / L$, where L is the LS pool size used in the experimental studies (here nine). $fitness_{ji}$ is the expected (average) objective function fitness obtained from the traditional memetic algorithms, MA_i, at function evaluation call/count j.

From Figures 7-9, it is notable that although the baseline Meta-Lamarckian learning scheme, represented by search trace MA-B, performs generally better than MA-AV, it still performs poorly when compared to the best traditional MA on each problem, i.e.,

MA-FL for Sphere and MA-DS for Griewank and Bump. On the other hand, the strategies MA-S1 and MA-S2, display performances that are statistically significantly better than GA, MA-AV and MA-B and also close to that of the best LS hybrid on each benchmark problem.

To gain a better understanding of the two proposed strategies, MA-S1 and MA-S2, they have been further analyzed and compared to the Simple Inheritance Mechanism introduced recently in [10] for discrete combinatorial search according to the following aspects (see Figure 10 for pseudo code of SIM):

- Search Quality and Efficiency – the capability of the strategy to provide high search quality and efficiency over different problems types.
- Computational Cost – the amount of extra CPU effort incurred over and above traditional Lamarckian learning in MA.
- Robustness – the capability of the strategy to generate performances that are close to the best traditional MA (from among the pool tested), on different problems.
- Simplicity – ease of implementation. Simple strategies should require minimum effort to develop as well as a minimum numbers of control parameters that need to be managed.

a) Search Quality and Efficiency

The search quality and efficiency performance of MA-S1 is dependent on the initial period, g, allocated for learning before the mechanism of sub-problem decomposition steps in. It is also strongly dependent on the nearest neighbor parameter, k, which defines the candidate LSs that will compete for selection. Shown in Table 2a, is the mean search

quality (i.e., function fitness value) and associated standard deviations for various values of *g* and *k* applied to the benchmark problems. Likewise, the SIM approach also requires *a priori* specification of the innovation rate (*IR*) which controls the diversity for local search method selection. The search performances of SIM using various innovation rates (i.e., $0.10<IR<0.35$) are summarized in Table 2b. In contrast to both MA-S1 and SIM, MA-S2 has no extra parameters to set. Its search performances on the test functions are shown in Table 2c. For information on the behavior of MA-S2 on further test problems, with various local search methods and LS pool sizes, see Tables 5a and b in the annex.

	Sphere Function (Minimum)	**Griewank Function (Minimum)**		**Bump Function (Maximum)**	
	Global Optimum found at Eval. Count of	Mean at 40,000	Standard Deviation	Mean at 100,000	Standard Deviation
MA-S1 *g, k* = 1, 2*L*	9833	5.08	5.58	73.8	5.00
MA-S1 *g, k* = 1, 6*L*	10833	15.4	39.3	69.3	8.89
MA-S1 *g, k* = 3, 3*L*	12297	55.3	21.8	72.2	6.46

(a)

SIM *IR* = 0.10	17032	3.54	5.20	72.9	15.9
SIM *IR* = 0.15	12765	41.4	70.4	73.2	7.84
SIM *IR* = 0.35	13595	35.9	48.9	71.4	11.74

(b)

MA-S2	7198		2.80	8.28	73.4	2.22
Standard GA	Mean at 40,000 = 63.7	Standard Deviation = 8.19	15700	2400	66.7	7.33

(c)

Table 2: (a) Effects of *g* and *k* on MA-S1, (b) Effects of Innovation Rate (*IR*) on SIM, (c) Mean and Standard Deviation of MA-S2 and Standard GA, on the benchmark problems. Griewank Function fitnesses are multiplied by 10^{04}, Bump Function fitnesses are multiplied by 10^{02}. Results are taken over 20 runs.

Note that it was possible to obtain performances of MA-S1 and SIM that are competitive or superior to MA-S2 after fine-tuning of the control parameters and extensive empirical runs on each individual test problem. However, no fixed values of the parameters were always found to generate competitive performances on all three benchmark problems. Besides, on average, the search quality of MA-S2 is found to be statistically significantly better than both MA-S1 and SIM on the Sphere and Griewank benchmark problems, when using the Student-t test at the confidence level of 0.05. On the Bump function, however, the search qualities of three strategies do not differ significantly statistically.

b) Computational Cost

Of course, when searching across a domain where the algorithm function evaluations are expensive (order of minutes or more), such as aerodynamic wing design, all these adaptive strategies have negligible additional cost. The total computational costs incurred by MA-S1, MA-S2 and SIM over traditional MA are of the order of $O(c^2\gamma+c\tau)$, $O(c\tau)$ and $O(c\theta)$ respectively. c is the number of chromosomes evaluated so far (LS selections) made, γ is the time taken to perform the Euclidean distance measure between any two chromosomes, τ is the time required to evaluate equation (1) and choose a LS from among the pool, while θ is the time SIM requires for each LS selection.

On a Pentium III processor, τ (for a 20D problem) and γ are found to be around 2μsec and 7μsec, respectively. For a MA search with stopping criteria of 100,000 maximum function evaluation calls, c is around 1400. The total time incurred by the adaptive strategies over traditional MA search at MA-S1: 1400^2 *7μs + 1400*2μs ≈ 14 seconds

and MA-S2: 1400*2μs = 2.8 milliseconds, respectively is negligible. Likewise, the computational cost incurred by SIM is also negligible. Nonetheless, among the three strategies presented, MA-S1 is relatively more costly.

c) Robustness and Simplicity

Adopting the proposed adaptive strategies in MA search improves the robustness of the search performance greatly: this is one of the primary goals in this study. All three strategies are able to select a LS that matches the problem throughout the search, thus producing search performances that are close to the best traditional MA on the benchmark problems. However, MA-S2 is most robust considering that it has no control parameters requiring management, unlike the g and k parameters for MA-S1 and the innovation rate for SIM. These control parameters can result in very poor performance of the strategies if inappropriately set. In addition, both MA-S2 and SIM are generally much simpler to implement than MA-S1.

Based on these performance metrics, the Biased Roulette Wheel strategy, MA-S2, is considered the most competitive adaptive Meta-Lamarckian learning strategy for MA search, especially when strategy *robustness* is the main issue. Further discussion is thus restricted to this method in the current work. However, before demonstrating MA-S2 on a real-world engineering design problem, we address some further issues.

2) Other Issues

The success of Meta-Lamarckian learning in memetic algorithms for continuous parametric design optimization also involves the following issues:

- What is the effect of LS pool size on design search performance?

- What local search methods should be included in the LS pool?
- Can human expert knowledge be incorporated into the Meta-Lamarckian learning approach proposed (for example, by the choice of LS pool members)?

a) Effects of LS Pool Size

The chances of obtaining robust or better design search performance from a MA generally increase by using multiple LS during the search, especially when adaptive strategies are used to control the choice of LS. The effect of different sizes of LS pool (i.e., 2, 4, 9 and 25) on MA-S2 is presented for the Griewank benchmark problem in Table 3.

LS Methods within the Pool	**Mean Search Fitness at Function Evaluation Count of**				
	1000	**5000**	**10000**	**30000**	**40000**
Pool Size = 2 DS, PO	536	59.3	23.9	6.74	5.34
Pool Size = 4 CO, DS, NM, PO	973	62.0	23.2	7.05	5.81
Pool Size = 9	2770	165	41.1	5.17	2.80
Pool Size = 25	9140	287	96.6	20.9	8.40
Best Traditional MA-DS	443	65.3	23.1	4.63	3.45

Table 3: Effects of Changes in LS Pool Size on Griewank function, mean function fitnesses are multiplied by 10^{04}. When the pool size is 25, the other methods used were from the OPTIONS toolkit [32].

From the table, it may be seen that the use of a smaller LS pool size is often associated with quicker improvements during earlier stages, as less evaluations are needed to acquire sufficient knowledge about the LS before the learning strategy begins to bite. So, although it is advantageous to include a large pool of LS to maintain wide ranging robustness, one concern is the number of evaluations required before the LS decision

space is sufficiently explored. From extensive studies conducted on a range of test problems and LS methods, the MA-S2 strategy is found to remain generally effective even with a pool size of up to 25 different local search methods. However, a pool size of around 10 local search methods would be more practical when working with real world design problems and is thus recommended.

b) Choice of Local Search Methods in the Pool

The choice of which local search methods to include within the pool is yet another issue that is important, especially when one has little or no prior knowledge of which local method works best on a problem. Nevertheless, as a rule of thumb, the recommended LS pool should contain both derivative based and non-derivative local methods. Derivative based LS include quasi-Newton methods [26], quadratic programming [33] and conjugate-gradient. For non-derivatives methods, direct search methods [25], linear approximation methods [25, 34], and local evolutionary search [35] are all good choices. The best traditional memetic algorithm on the sphere function, MA-FL, FL is an example of the quasi-Newton methods, which have the ability to locate the local optimum rapidly. DS, Local Evolutionary Strategy (ES), Local Evolutionary Programming (EP) and Local Simulated Annealing (SA) are direct search or local evolutionary methods that makes a good combination with FL in the same pool as they can handle problems where derivative based LS may fare badly. Other considerations on the choice of LS pool would include the abilities of LS to handle constrained and non-linear problems.

c) Incorporation of Human Expert Knowledge

The proposed Meta-Lamarckian learning approach also permits the incorporation of a designer's intuition, experience and knowledge during design activities. For example, assume that a design specialist has knowledge about the Davies, Swann and Campey local search method (DS) [25], i.e., that it often performs relatively well on most design problems of interest to him/her. However, being not totally sure about its suitability for a new problem, he/she may not wish to commit to a traditional MA and instead incorporate this expert knowledge into the adaptive strategies by biasing the DS local method with higher probabilities of being selected. Trace MA-S2A of Figure 11 illustrates the case where the DS method is biased with twice the chance of being selected, as compared to the other local methods in the same pool. Shown also in Figure 11 is trace MA-S2B where the designer chooses to use six local methods (PO, NM, CO, BC, PD and DS) as the pool to perform a search on the Griewank function. From these results, it is seen that superior search performances are obtained when human expert knowledge is incorporated into the Meta-Lamarckian learning process. Improvements can also be found for the other benchmark problems when knowledge is incorporated in this way.

IV. HYBRID MA-LS WITH ADAPTIVE META-LAMARCKIAN LEARNING APPLIED TO AERODYNAMIC WING DESIGN

In aerospace companies, designers/design teams are often required to work on design problems that are new to them and accompanied by tight deadlines. In this paper, the parametric design of a civil transport aircraft wing for operation at Mach 0.785 and a Reynolds number of 7.3 million using strategy MA-S2 is considered [36]. The objective is to design a wing with minimal drag D/q meters^2 as calculated by using tools with

variety of levels of complexity, with target lift, wing weight, volume, pitch-up margin and root triangle layout chosen to be representative of a 220 seat wide body airliner.

Figure 12 shows a geometric view of streamlines over the transonic civil transport aircraft. The planform geometry is also shown in Figure 12, while the definitions of the wing design variable, nonlinear inequality constraints and optimization conditions considered are given in Table 4. The parameters used to describe the wing design problem considered here consist of the free-stream velocity and viscosity and coefficient of lift of the wing together with a small number of overall wing geometry variables. The wing geometry is characterized by the planform shape of the wing together with several span-wise functions. Here, a wing design is represented by eleven parameters (i.e., eleven optimization design variables). In order to prevent the optimizer from driving the designs to unworkable extremes, several constraints are placed on the wings designed. These are the under-carriage bay length (which must be accommodated within the root to kink section of the wing), the fuel tank volume (which must be accommodated between the main spars within the wing), the wing weight and the pitch-up margin. Here, one of the tools developed by BAE Systems in this area, the TADPOLE program, which is based on empirical models by Cousin and Metcalfe [37] is used to predict drag. TADPOLE returns the total drag coefficient defined by the wave drag due to the presence of shocks, viscous wake or profile drag due to the boundary layer and vortex or induced drag due to the tip vortex of the 3-d wing.

On this aerodynamic wing design problem, the worst and best traditional MA were found to be MA-BC and MA-CO, respectively (see Figure 13). In addition, both MA-FL and MA-DS fare very poorly compared to others in the LS pool of nine. Once again, MA-

S2 was able to generate design search performance that is as good as the best traditional MA on this realistic problem (MA-CO). With the incorporation of human expert knowledge, superior design search performance may be attained, as shown by the search trace MA-S2C of Figure 13 where the CO local method is biased with greater probability of being selected. Such results encourage the use of multiple local methods, rather than relying simply on one fixed, and possibly poor choice. Meta-Lamarckian learning clearly offers a high quality and robust approach for engineers working on continuous parametric design problems, regardless of whether *a priori* knowledge of the best LS is available.

V. CONCLUSION

Every search algorithm, except for uniform random search, introduces some unique form of bias, suitable for some classes of problems but not for others. Therefore, any traditional memetic algorithm using a fixed local search (LS) will include biases. Since *a priori* knowledge about problem cost surfaces is often scarce this makes selection of the appropriate LS for use in such fixed schemes difficult. The great advantage of Meta-Lamarckian learning in a memetic algorithm is that it is able to address this fundamental problem by allowing a range of local searches to cooperate and compete in finding good solutions.

Several strategies of Meta-Lamarckian learning have been described and analyzed here for continuous parametric design search. Empirical studies on three representative classes of benchmark problems have shown that the strategies presented are effective in producing search performances that are close to the best traditional MA with a LS chosen to suit the problem in hand. Given that such knowledge is often not available *a priori*,

this ability to tackle new problems in a robust way is of significant value. Overall, the Biased Roulette Wheel approach to method selection is the most competitive strategy considered here. It is shown to be capable of attaining robust, high quality and efficient performance on benchmark problems and a real-world aerodynamic design problem.

ACKNOWLEDGEMENT

The authors wish to thank the editors and anonymous referees for their constructive comments on an earlier draft of this paper. This work was funded by the University Technology Partnership for Design, which is a collaboration between BAE Systems and Rolls-Royce and the Universities of Cambridge, Sheffield and Southampton.

REFERENCES

[1] A. Torn and A. Zilinskas, "Global Optimization", Volume 350 of Lecture Notes in Computer Science, Springer-Verlag, 1989.

[2] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms", Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[3] L. Davis, "Handbook of Genetic Algorithms", Van Nostrand Reinhold, New York, 1991.

[4] W. E. Hart, "Adaptive Global Optimization with Local Search", PhD thesis, University of California, San Diego, May 1994.

[5] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolutionary Programs", Springer-Verlang, New York, 2nd edition, 1994.

[6] H. Bersini and B. Renders, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways", IEEE International Symposium Evolutionary Computation, pages 312-317, Orlando, Fl, 1994.

[7] C. Houck, J. Joines, and M. Kay, "Utilizing Lamarckian Evolution and the Baldwin Effect in Hybrid Genetic Algorithms," NCSU-IE Technical Report 96-01, Meta-Heuristic

Research and Applications Group, Department of Industrial Engineering, North Carolina State University, 1996.

[8] A. Vicini and D. Quagliarella, “Airfoil and wing design using hybrid optimization strategies”, AIAA Journal, Vol.37, No.5, May 1999.

[9] P. Merz, “On the Performance of Memetic Algorithms in Combinatorial Optimization”, 2nd Workshop on Memetic Algorithms, GECCO 2001, San Francisco, CA, USA.

[10] N. Krasnogor, “Studies on the Theory and Design Space of Memetic Algorithms”, Ph.D. Thesis, Faculty of Computing, Mathematics and Engineering, University of the West of England, Bristol, U.K, 2002.

[11] N.Krasnogor, B.Blackburne, J.D.Hirst and E.K. Burke N., “Multimeme Algorithms for the Structure Prediction and Structure Comparison of Proteins”, Proceedings of Parallel Problem Solving From Nature, Lecture Notes in Computer Science, 2002.

[12] Y. S. Ong and A. J. Keane, “A domain knowledge based search advisor for design problem solving environments”, Engineering Applications of Artificial Intelligence, Vol. 15, No. 1, pp. 105-116, 2002.

[13] D. H. Wolpert and W. G. Macready, “No free lunch theorems for optimization", IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 67--82, 1997.

[14] L. J. Fogel, P. J. Angeline, and D. B. Fogel, “A Preliminary Investigation on Extending Evolutionary Programming to Include Self-adaptation on Finite State Machines”, Informatica 18: 387-398, 1995.

[15] T. Back and M. Schutz, “Intelligent mutation rate control in canonical genetic algorithms”, Proceedings of the 9th International Symposium, ISMIS 96, pp. 158--167, Zakopane (Poland), 9.-13. June 1996. Springer-Verlag, Berlin (Germany).

[16] R. Hinterding, Z. Michalewicz, and A. E. Eiben, “Adaptation in evolutionary computation: A survey”, Fourth International Conference on Evolutionary Computation (ICEC 97), pages 65-69, IEEE Press, Piscataway (NJ), 1997.

[17] J. E. Smith and T. C. Fogarty, “Operator and parameter adaptation in genetic algorithms”, Soft Computing, Vol. 1, No. 2, pp. 81--87, 1997.

[18] G. Magyar, M. Johnsson and O. Nevalainen, “An adaptive hybrid genetic algorithm for the three-matching problem”, IEEE Transactions on Evolutionary Computation, vol. 4, issue 2 July 2000.

[19] T-M. Hugo, Ross, Peter, and V-R. Manuel, “Evolution of Constraint Satisfaction Strategies in Examination Timetabling”, Genetic and Evolutionary Conference, Orlando, FL, July 13-17, 1999, pp 635-642.

[20] P. Cowling, G. Kendall and E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit", Lecture Notes In Computer Science, Springer. Third International Conference on the Practice and Theory of Automated Timetabling, PATAT 2000.

[21] D. W. Johnson and R. T. Johnson, "Cooperation and competition: Theory and research", Edina, MN: Interaction Book Company, 1989.

[22] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm", IEEE International Conference on Evolutionary Computation (pp. 122-125), 1997, Piscataway, NJ: IEEE.

[23] M. D. McKay, R. J. Beckman and W. J. Conover, 1979, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", Technometrics, 21(2), pp.239-245.

[24] W. M. Spears, "A simple GA (conceptually based on Genesis) written in C", 1991. [Online] Available: http://www.aic.nrl.navy.mil:80/galist/src/#C

[25] H. P. Schwefel, "Evolution and Optimum Seeking", John Wiley&Sons, 1995.

[26] J. N. Siddall, "Optimal Engineering Design: Principles and Applications", Marcel Dekker Inc., New York, 1982.

[27] L. Davis, "Bit Climbing, Representational Bias, and Test Suite Design", 4th International Conference on Genetic Algorithms (ICGA IV), ed. Belew, R. K. and Booker, L. B. Morgan Kaufman Publishers, Inc., San Diego, 1991.

[28] J. A. Nelder and R. Meade, "A simplex method for function minimization", Computer Journal, 7:308-313, 1965.

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

[30] De Jong, "An analysis of the behaviour of a class of genetic adaptive systems", PhD thesis, University of Michigan, 1975.

[31] A. J. Keane, "Genetic algorithm optimization of multi-peak problems: studies in convergence and robustness", Artificial Intelligence in Engineering 9(2): 75-83, 1995.

[32] A. J. Keane, 1995, The OPTIONS Design Exploration System, http://www.soton.ac.uk/~ajk/options /welcome.html.

[33] C. Floudas and V. Visweswaran, "Quadratic optimization", Handbook of Global Optimization (R. Horst and P. M. Pardalos, eds.), Kluwer Academic Publishers, 1995.

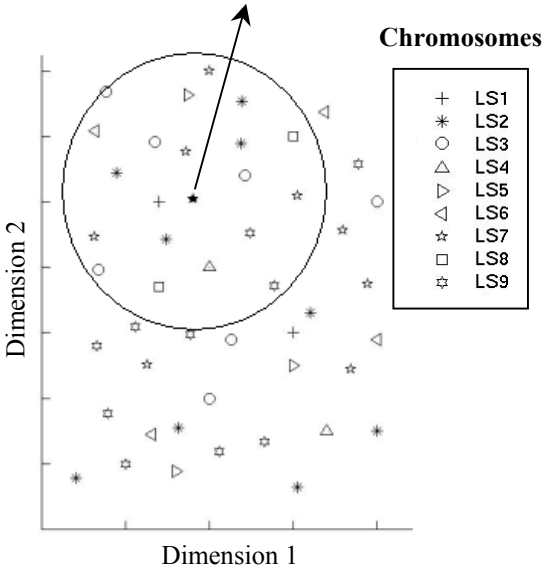
[34] M. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation", SIAM conference 1996, Virginia.

[35] H. Voigt and J. Lange, “Local evolutionary search enhancement by random memorizing”, IEEE International Conference on Evolutionary Computation, pp. 547-552, Piscataway, NJ, 1998. IEEE Press.

[36] A. J. Keane and N. Petruzzelli, “Aircraft wing design using GA-based multi-level strategies”, Proceedings of the 8th AIAA/USAF/NASSA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, Long Beach, 2000.

[37] J. Cousin and M. Metcalfe, “The BAE Ltd transport aircraft synthesis and optimization program”, AIAA 90-3295, September 1990.

For k=20, the local search method with the maximum average fitness among the 20 nearest chromosome in the database will be used for Lamarckian Learning on the new chromosome.



MA-S1 Pseudo-Code

IF (GA Generation < *g*)

1. Generate a random number between 1 and LS pool size;
2. Select the LS method that the number indicates;
3. Create/Update Database.

ELSE

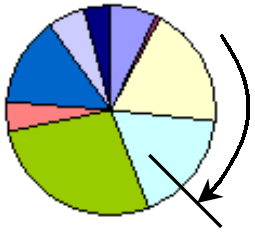
1. Locate k nearest chromosomes to $\hat{p}$ in database P using Simple Euclidean Measures; i.e.,

$$P_k \Rightarrow \left\{ k \min_{P_j \in P} \left\| \hat{p} - p_j \right\| \right\}$$

2. Find the average fitness of each member of the reduced LS pool based on P_k;
3. Select the LS method with the maximum average fitness;
4. Update Database.

(a)

Local search methods have equal space on roulette wheel initially, thus equal probability.



A single spin of the roulette wheel will pick the local method.

Space allocated according the LS's fitness

MA-S2 Pseudo-Code

IF (Training Stage)

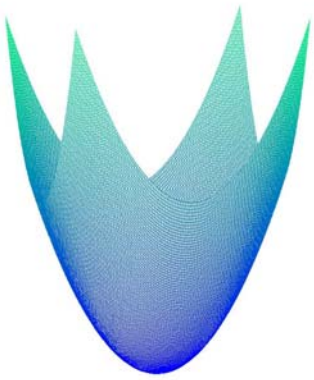
1. Ensure each LS is given one chance to participate; in a random order;
2. Update LS's Global fitness.

ELSE

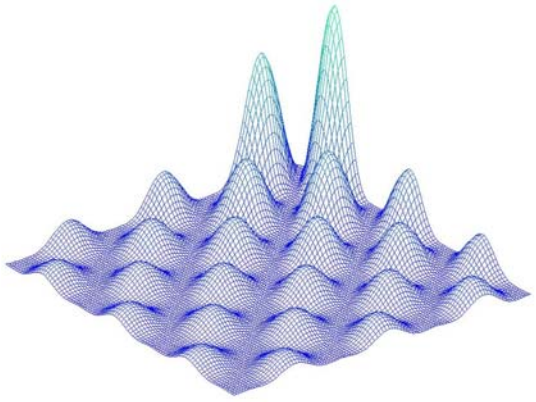
1. Sum the fitness of each member of the LS pool;
2. Determine the normalized relative fitness of each member of the LS pool;
3. Assign space on roulette wheel proportional to local method's fitness;
4. Generate a random number between zero and 1, select the LS method from where the random number falls;
5. Update LS's Global fitness.

(b)

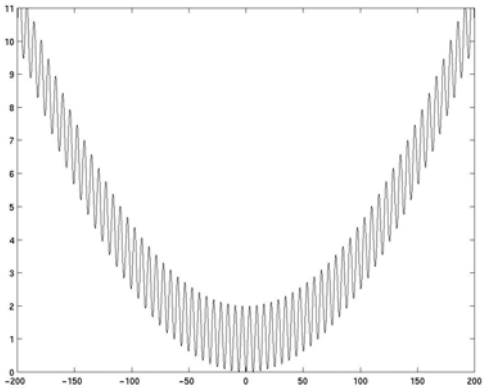
Figure 1: Pictorial Illustration and Pseudo Code of
(a) Sub-problem Decomposition Strategy, MA-S1.
(b) Biased Roulette Wheel Strategy, MA-S2.



(a)



(c)



(b)

Figure 3: (a) 2-D Sphere function
(b) 1-D slice of Griewank function
(c) 2-D Bump Constrained function

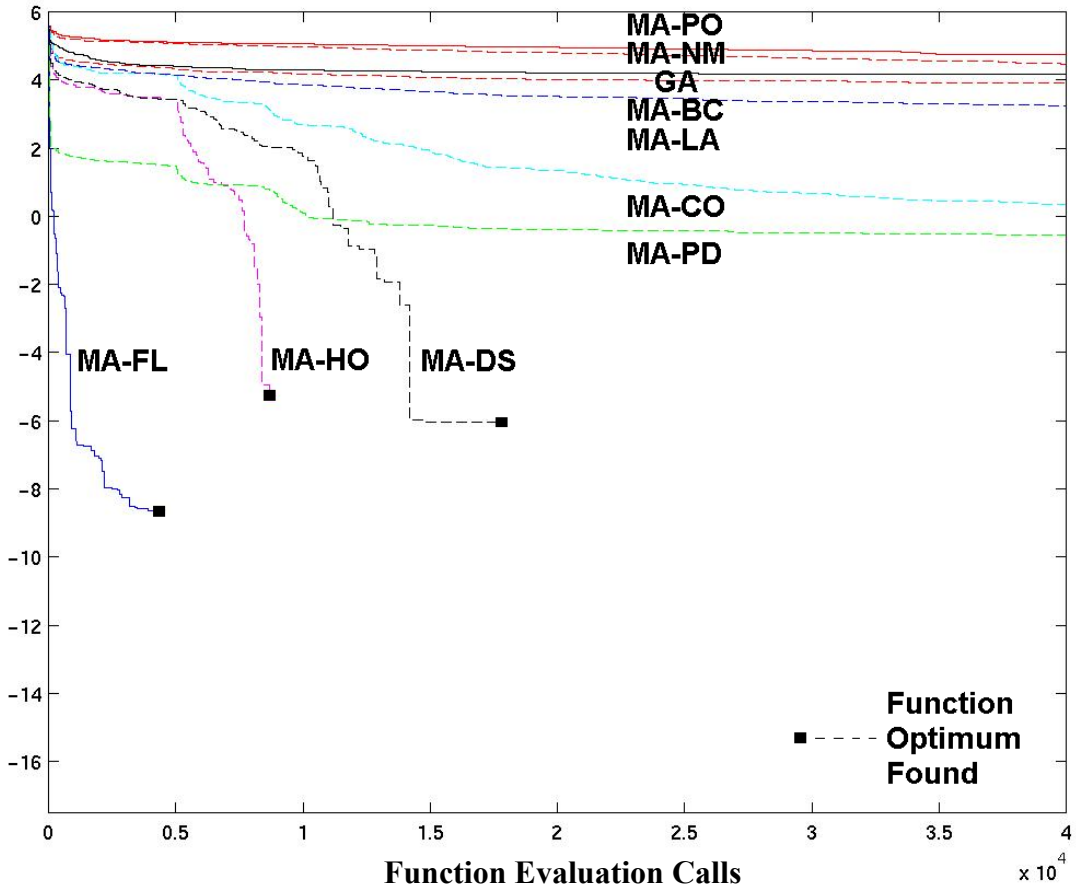


Figure 4: Search Traces (average of 20 runs) for minimizing 30D Sphere function using GA and various traditional memetic algorithms with Lamarckian Learning. (Sphere function minimum at 0.0 and as $\log(0.0) = \infty$ the search traces end.)

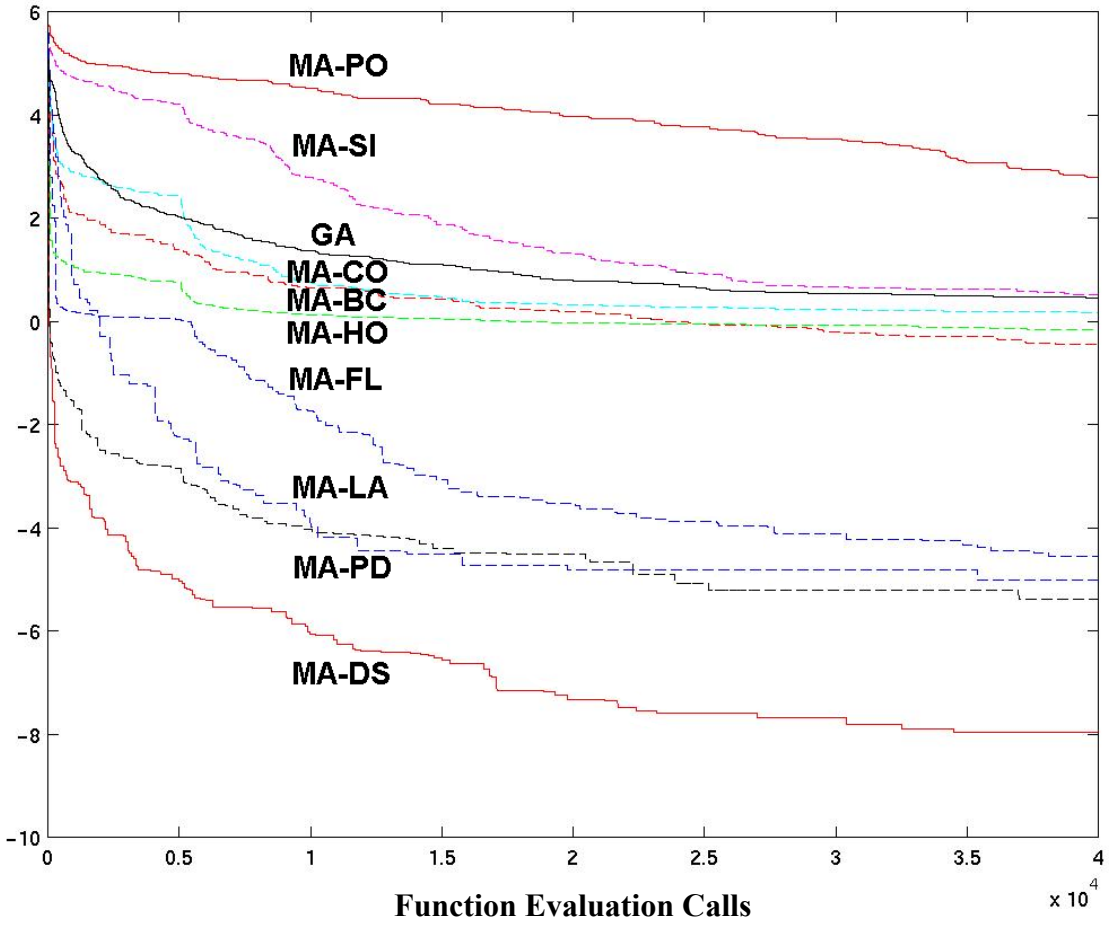


Figure 5: Search Traces (average of 20 runs) for minimizing 10D Griewank function using GA and various traditional memetic algorithms with Lamarckian Learning.

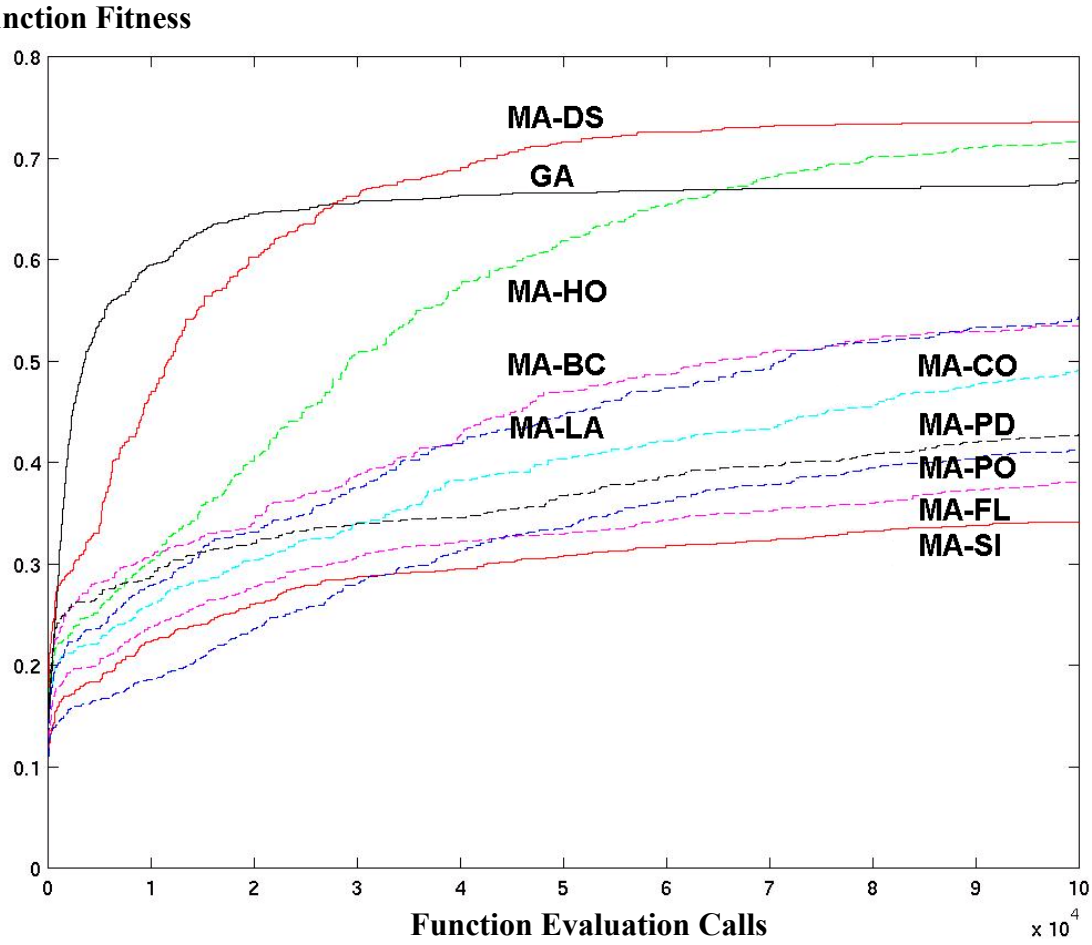


Figure 6: Search Traces (average of 20 runs) for maximizing 20D Bump function using GA and various traditional memetic algorithms with Lamarckian Learning.

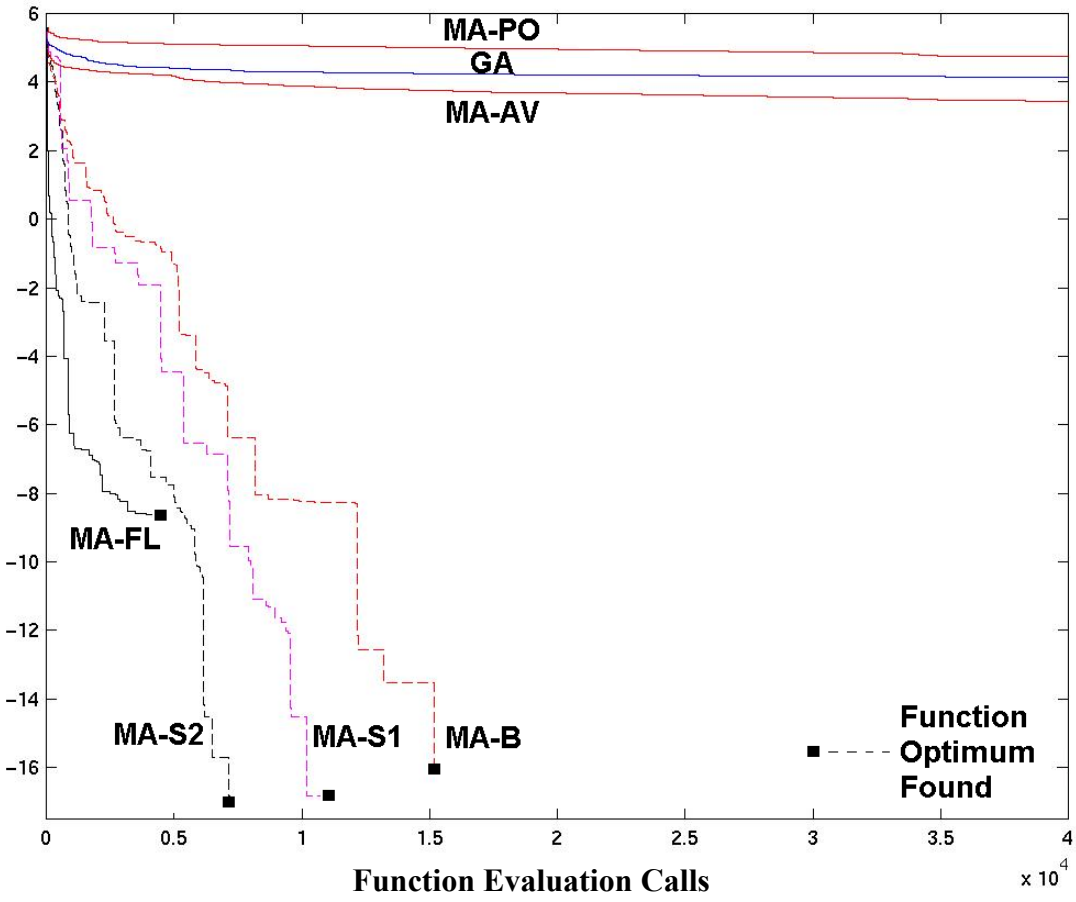


Figure 7: Search Traces (average of 20 runs) for minimizing 30D Sphere function using strategies MA-B, MA-S1$_{(g,k\,=\,1,6L)}$ and MA-S2. Shown also are the search traces for GA, MA-PO, MA-AV and MA-FL hybrids.

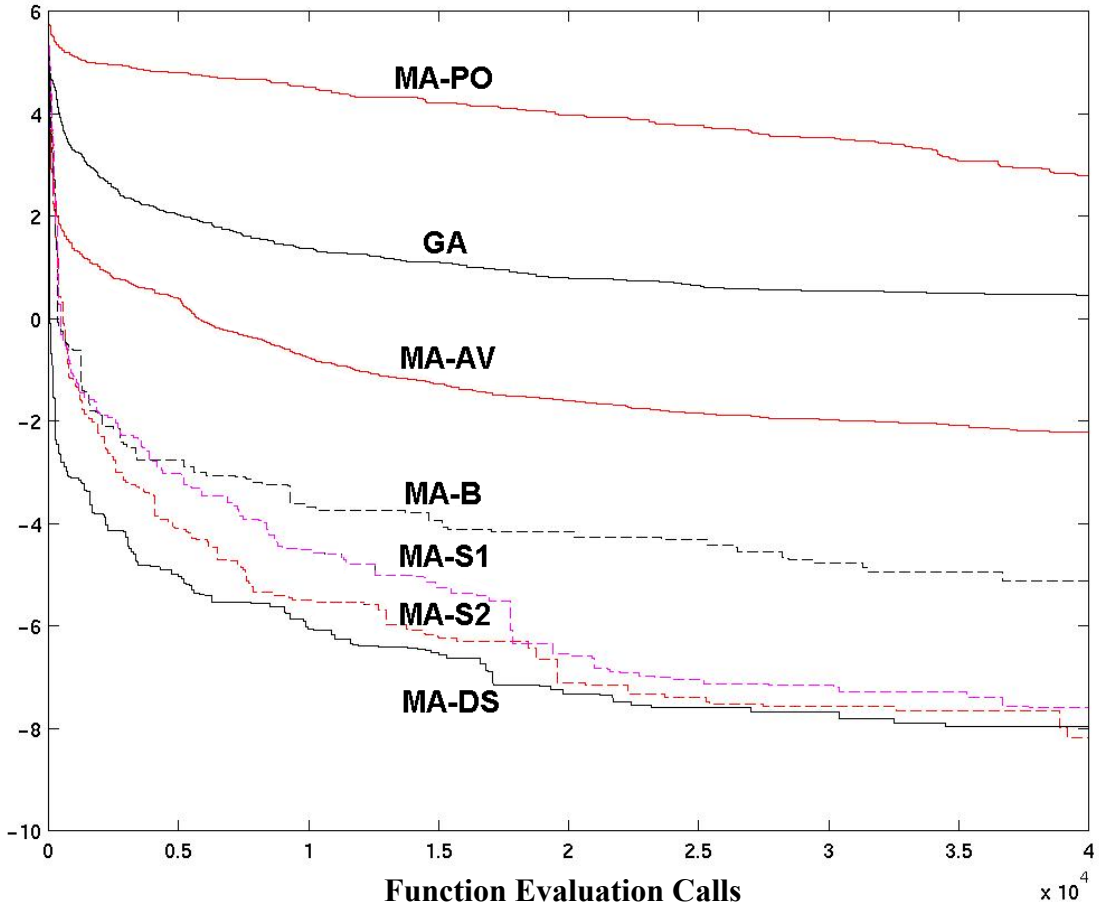


Figure 8: Search Traces (average of 20 runs) for minimizing 10D Griewank function using strategies MA-B, MA-S1$_{(g,k\,=\,1,2L)}$ and MA-S2. Shown also are the search traces for GA, MA-PO, MA-AV and MA-DS hybrids.

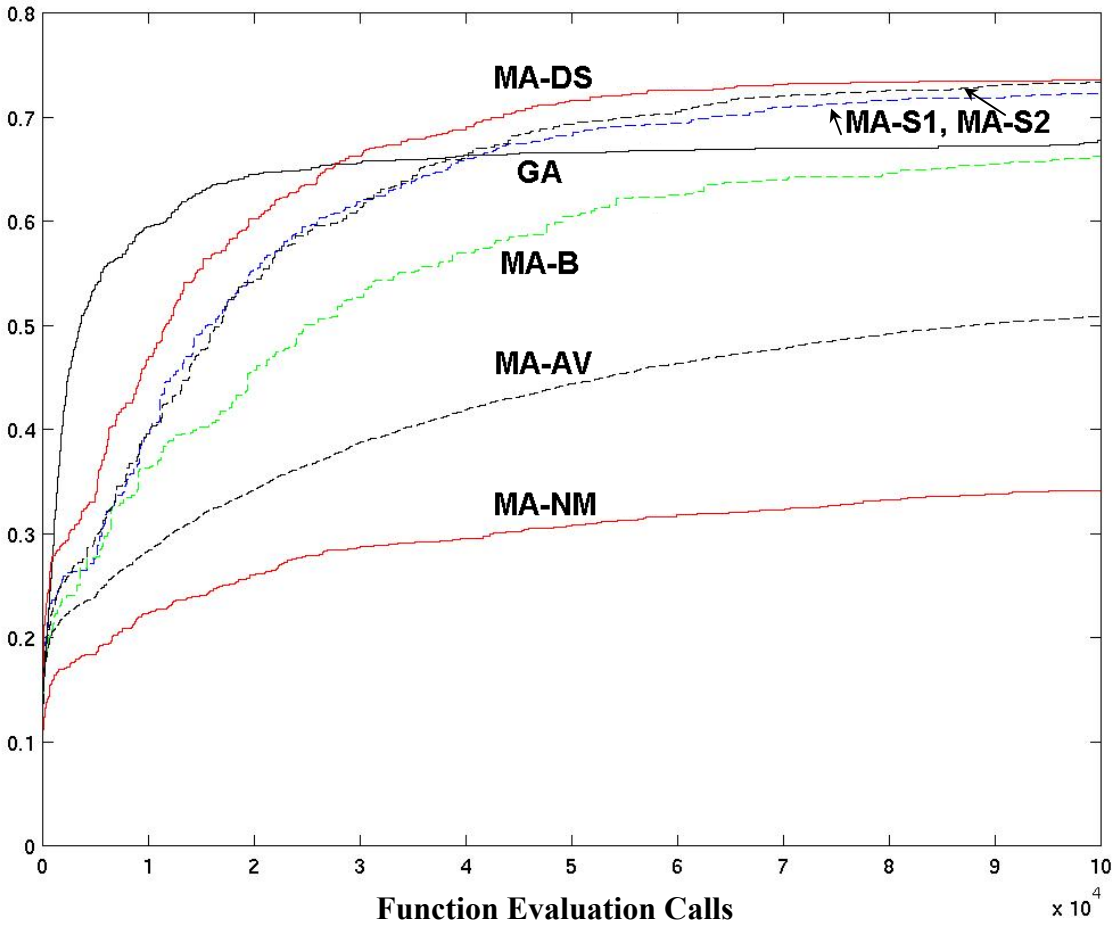


Figure 9: Search Traces (average of 20 runs) for maximizing 20D Bump function using strategies MA-B, MA-S1$_{(g,k\,=\,3,3L)}$ and MA-S2. Shown also are the search traces for GA, MA-DS, MA-AV and MA-NM hybrids.

BEGIN

Replace LS method of individual with randomly selected LS according to specified *Innovation Rate (IR)*. $IR \in [0,1]$, where *IR* of value 0 implies zero diversity in the choice of LS, i.e., any LS not in the population will not be re-introduced, and value 1 implies maximum diversity with every available LS utilized equally.

Select 2 parent chromosomes;
IF (both parents have the same LS**)**
    Inherit common LS to the offspring;
ELSE-IF (parents1.fitness == parents2.fitness)
    Random select one of the two attached LS to the offspring;
ELSE-IF (parents1.fitness > parents2.fitness)
    Inherit parent1 LS to the offspring;
ELSE /* (parents2.fitness < parents1.fitness) */
    Inherit parent2 LS to the offspring;
END
END

Figure 10: Pseudo Code of Simple Inheritance Mechanism

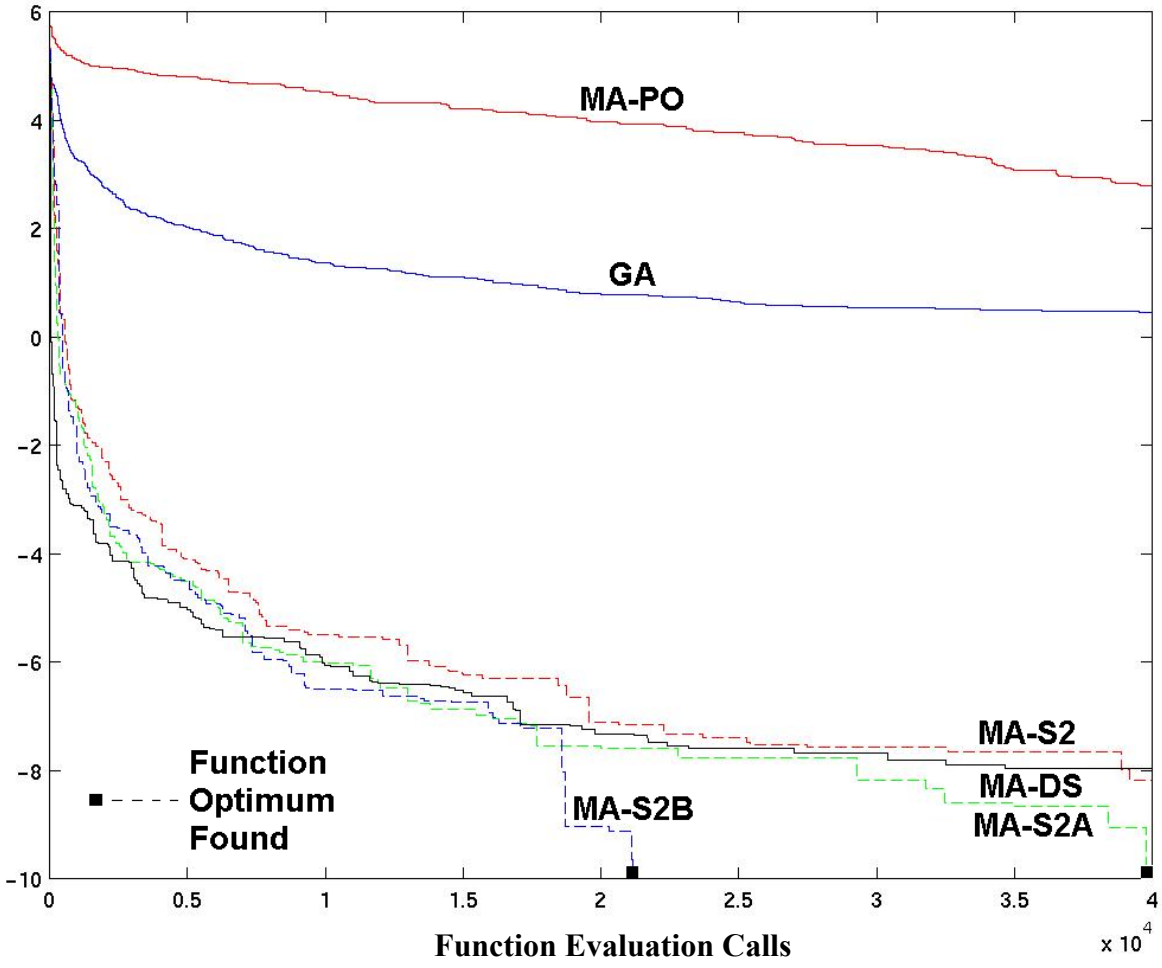
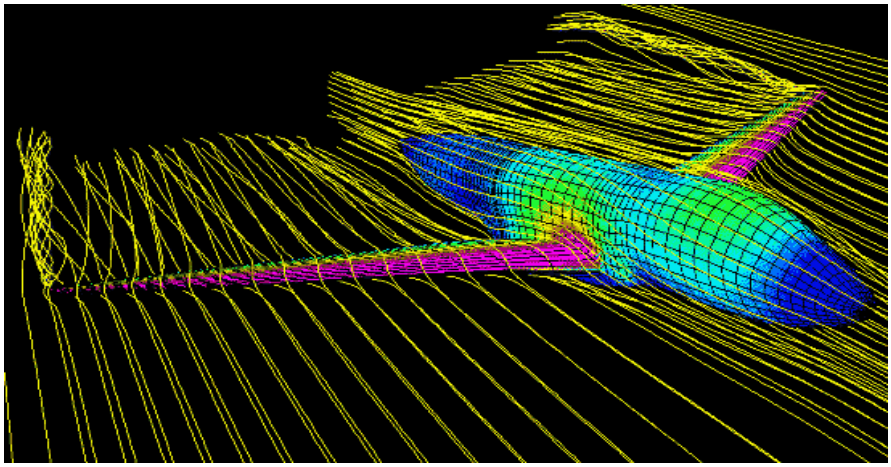


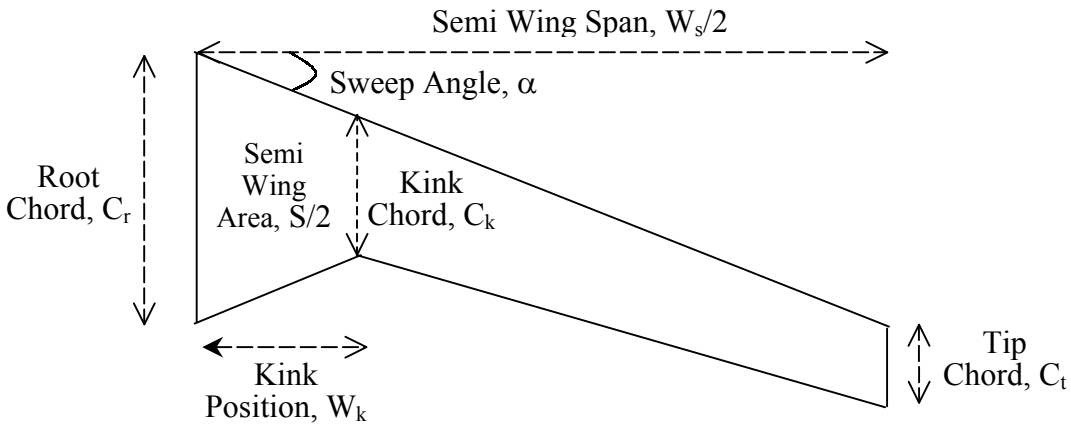
Figure 11: Search traces (average of 20 runs) for minimizing 10D Griewank function using MA-S2A and MA-S2B ***with the incorporation of Human Expert Knowledge***. Shown also are the search traces for GA, MA-PO, MA-DS and MA-S2 hybrids.

11 Wing Design Variable Definitions		
Lower Limit	**Upper Limit**	**Quantity (units)**
100	250	Wing Area (m^2), S
6	12	Aspect Ratio, W_s^2/S
0.2	0.45	Kink position, $2W_k/W_s$
25	45	Sweep angle (degrees), α
0.4	0.7	Inboard taper ratio, C_k/C_r
0.2	0.6	Outboard taper ratio, C_t/C_r
0.1	0.18	Root thickness/chord, T_r/C_r
0.06	0.14	Kink thickness/chord, T_k/C_k
0.06	0.14	Tip thickness/chord, T_t/ C_t
4.0	5.0	Tip wash, twist (degrees)
0.65	0.85	Kink washout fraction
Four Design Constraints		
2.5		Under-Carriage bay length
	135000	Wing weight (N)
40.0		Wing volume (m^3)
	5.4	Pitch-up margin

Table 4: Definitions of the wing design variable, nonlinear inequality constraints and optimization conditions considered.



(a)



(b)

Figure 12: (a) Geometric View of Streamlines over a Transport Aircraft. (b) 2-D transonic wing planform.

Aerodynamic Drag

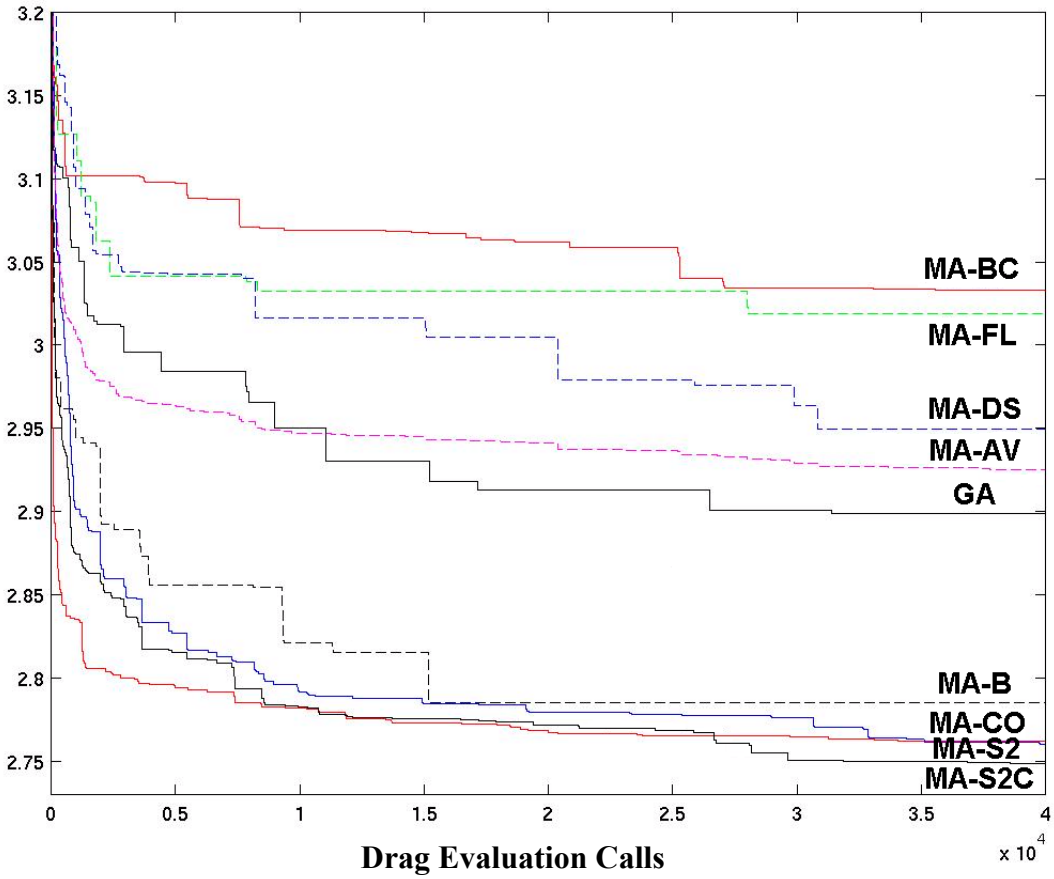


Figure 13: The design of a Transonic Civil Transport Aircraft Wing using Adaptive Meta-Lamarckian Learning hybrid MA-S2. ***MA-S2C differs from MA-S2 due to the incorporation of human knowledge (see text)***. Shown also are the search traces for GA, MA-BC, MA-FL, MA-DS, MA-AV, MA-B and MA-CO.

ANNEX

Benchmark Problems	Range of x_i	Characteristics				Function Minimum At
		Epi*1	Mul*2	Disc*3	Con*4	
$F_{Rosenbrock} = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	$[-2.048, 2.048]^{2, 30}$	high	none	none	no	0.0
$F_{Step} = 6n + \sum_{i=1}^{n} \lfloor x_i \rfloor$	$[-5.12, 5.12]^{30}$	none	none	medium	no	0.0
$F_{FoxHole} = \left[0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{n} (x_i - a_{ij})^6} \right]^{-1}$	$[-65.536, 65.536]^2$	none	low	high	no	1.0
$F_{Rastrigin} = (10 * n) + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right)$	$[-5.12, 5.12]^{20}$	none	high	none	no	0.0
$F_{Schwefel} = 418.9829 * \mathrm{n} + \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^{10}$	none	high	none	no	0.0

1: Epistasis, *2:Multimodality, *3:Discontinuity, *4:Constrainted

Table 5a: Further Benchmark Problems used for Experimental Studies

All the other benchmark problems noted here are commonly used in the literature. *n* is the parameter or dimension size for the respective problems used in the experiments. All benchmark functions were searched using the Biased Roulette Wheel strategy, MA-S2, over twenty independent runs. Each run continued until the global optimum was found or a maximum of 40,000 function evaluations was reached. The local search methods and pool size used were selected randomly. The local search methods not presented previously include: Successive Linear Approximation (AP), Dynamic Hill Climbing (DH), Evolutionary Strategy (ES), Evolutionary Programming (EP), Simulated Annealing (SA) and Rosenbrock's Rotating Coordinate Search (RO). The search results are tabulated in Table 5b where the MA-S2 strategy is shown to perform well, using a variety of local search methods, pool sizes and benchmark problems.

Benchmark Functions	Local search Method (LS) Pool	Standard GA		Best Traditional MA of Random LS Pool		Poorest Traditional MA of Random LS Pool		Adaptive Meta-Lamarckian Learning MA-S2	
		Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.
Generalized Rosenbrock [-2.048, 2.048]^2	BC,DG,EP, FL,MN	9.01	7.37	20/20 hits found Global Min. by Eval. 2217	NA	5320	1840	20/20 hits found Global Min. by Eval. 4177	NA
Shekel's FoxHole [-65.536, 65.536]^2	DS,EP,ES,FL, HO,NM,PD	20/20 hits found Global Min. by Eval. 29179	NA	20/20 hits found Global Min. by Eval. 6819	NA	20/20 hits found Global Min. by Eval. 28248	NA	20/20 hits found Global Min. by Eval. 9634	NA
Schwefel [-500, 500][10]	CO,DS,DP, LA,PD,RO	15.6	4.08	16/20 hits found Global and 4/20 found 2nd Minimum	NA	221	12.3	16/20 hits found Global and 4/20 found 2nd Minimum	NA
Generalized Rosenbrock [-2.048, 2.048][30]	BC,DG,EP, FL,MN	20.3	2.61	2.69	0.10	355	54.5	2.57	0.06
Step [-5.12, 5.12][30]	AP,BC,DH, DP,FL,SA	8.55	2.02	1.56	0.24	11.7	0.88	1.63	0.31
Rastrigin [-5.12, 5.12][20]	BC,CO,DS,HO, LA,MN,PD,PO	2.99	0.79	1.62	0.51	12.8	1.65	1.55	0.71

Table 5b: Statistical Results from studies on further test problems, with various LS combinations and Pool Sizes, using the Biased Roulette Wheel Strategy, MA-S2. The Local Search Method Pool for each benchmark function was randomly selected from a General LS Pool of over 30 taken from the OPTIONS toolkit [30]. Results of Generalized Rosenbrock function are multiple by 10^4 while all other functions are multiple by 10^{-1}.