# Memetic Gradient Search

Boyang Li, Yew-Soon Ong, Minh Nghia Le and Chi Keong Goh

***Abstract*—This paper reviews the different gradient-based schemes and the sources of gradient, their availability, precision and computational complexity, and explores the benefits of using gradient information within a memetic framework in the context of continuous parameter optimization, which is labeled here as Memetic Gradient Search. In particular, we considered a quasi-Newton method with analytical gradient and finite differencing, as well as simultaneous perturbation stochastic approximation, used as the local searches. Empirical study on the impact of using gradient information showed that Memetic Gradient Search outperformed the traditional GA and analytical, precise gradient brings considerable benefit to gradient-based local search (LS) schemes. Though gradient-based searches can sometimes get trapped in local optima, memetic gradient searches were still able to converge faster than the conventional GA.**

## I. Introduction

OPTIMIZATION problems often involve finding one or more optima of some objective functions $F: \Re^n \to \Re$, $n \geq 1$. In real-world engineering problems, the objective function of interest is usually multimodal with high dimensionality. The cost of one measurement of the function value is not limited to computational effort, but may also include human labor, time, fuel and many others. Any attempts to search exhaustively will be prohibitively expensive due to the huge domain space of these functions, yet the multimodality implies that a search too narrow in scope risks mistaking a local optimum for the global optimum. The design of such optimization algorithms has consequently become another optimization problem, minimizing the cost, which is usually bounded by the number of objective function evaluations, while maximizing chances of finding the global optimum. Throughout the paper, we only consider minimization problems as maximization can be trivially converted to minimization.

Over the years, researchers aiming to tackle these problems have developed various algorithms, of which the origin may be traced back to the time of Isaac Newton. Generally, optimization algorithms may be classified into deterministic methods, or conventional numerical methods, and stochastic methods, depending on whether or not the result is totally determined at the inception of the optimization. Depending on the scope of the search, they may also be roughly classified into global searches and local searches.

Manuscript received December 15, 2007.

Boyang Li, Yew-Soon Ong, and Minh Nghia Le are with the School of Computer Engineering, Nanyang Technological University, Singapore, 639798. (e-mail: libo0001@ntu.edu.sg, asysong@ntu.edu.sg, lemi0005@ntu.edu.sg)

Chi Keong Goh is with the Data Storage Institute, Agency for Science, Technology and Research, Singapore (e-mail: ckgoh@nus.edu.sg)

### A. Deterministic Search

Deterministic methods can be further classified into direct searches, or gradient-free algorithms, and gradient-based methods. In many methods of both classes, a line search is employed although the heuristics that determine the direction and step size vary. The strengths of the conventional numerical methods lie in fast convergence. However, their scope of exploration is usually limited compared to the whole objective space. Many methods can be "trapped" in the first local optimum they find and will not be able to improve further. Since these methods are completely determined, they are destined to produce poor results if the starting position is unfortunate. Thus they are commonly classified as local searches.

Direct search methods do not depend on other information other than the measurement of the objective function. No assumptions of differentiability are made. The very first direct search is now known as the coordinate strategy, or the Gauss-Seidel strategy among many other names [1-3]. This particular strategy singles out one dimension at a time, and performs a line search on that dimension. An important variation of this strategy, the Davies-Swann-Campey method, initially proposed by Swann[4], combined the idea of coordinate strategy with Rosenbrock's idea of rotating coordinates[5] so the line search is not limited to go parallel to the coordinate axes.

There are quite a number of gradient-based local searches, each having numerous variants. The most widely used include the method of steepest descent[6], conjugate gradients[7], Newton's method and a class of quasi-Newton methods. While steepest descent and conjugate gradients make use of only the gradient, Newton's method requires the costly computation of the inverse of the Hessian matrix $\nabla^2 f(\hat{x})$. Quasi-Newton methods were introduced to estimate the Hessian matrix based on first-order data and simplify computation. The assumptions of these methods are, of course, that the objective function is twice or once differentiable, for Newton's method and other method respectively. The interested reader is referred to the excellent books written by Schwefel[8] and Ruszczyński[9] for detailed discussions.

### B. Evolutionary Computing

Evolutionary computing, including genetic algorithms[10], evolutionary strategies[11, 12], and genetic programming [13], was inspired by the biological process that genes of creatures are optimized by the force of natural selection. In contrast to conventional numerical methods, genetic algorithm is able to explore the objective space more extensively, but it usually takes more computational effort and reaches the optima with less precision. Hybrid methods

that combine genetic algorithms with local searches have been proposed so as to perform efficient global searches and take the merits from the two distinct approaches. These methods are usually named memetic algorithms (MAs)[14] due to the resemblance of local searches to the evolution of culture in human societies, or the evolution of memes[15]. Memetic algorithm has been successfully applied in many disparate engineering fields, ranging from microarray gene selection[16], aerodynamic design[17], drug Therapies design[18] to assignment problems[19].

In addition to gradient-based LSes in MAs, other means to utilize gradient in traditional genetic algorithms or evolutionary strategies have also been proposed. Salomon[20] pointed out the similarities between finite differencing (FD) and Evolutionary Strategies and consequently introduced a new mutation operator based on approximated gradients in his Evolutionary Gradient Search.

In this paper, we studied the impact of gradients on memetic search performance. In Section II we review various gradient-based local search methods, from steepest descent to quasi-Newton methods. Section III discusses sources of gradients as well as their possible influence in a memetic algorithm. Finally, in Section IV, we present an empirical study comparing four algorithms, including a simple genetic algorithm, a MA utilizing gradients computed with FD, a MA utilizing analytical gradients and a MA with Simultaneous Perturbation Stochastic Approximation (SPSA) as the local search.

## II. GRADIENT-BASED LOCAL SEARCHES

A large number of widely-used deterministic algorithms depend on gradient information of the objective function to perform optimization. There are, for example, steepest descent, Newton's method, and a variety of quasi-Newton methods. Different from direct searches, in a gradient-based method a relationship between the input and output of the function must be established or approximated. While some methods, like steepest descent, rely on only first-order derivatives, the Newton's method requires second-order information. Thus it must be assumed that the objective function can be differentiated at least twice for application of the Newton's method and once for other methods.

One-dimensional line searches are usually employed iteratively in deterministic methods. Given a pre-defined search direction $\hat{v}_k \in \Re^n$, the search seeks the optimal step length $s_k$:

$$F(\hat{x}_k + s_k\hat{v}_k) = \min_s\{F(\hat{x}_k + s\hat{v}_k)\}$$

Common strategies include Fibonacci Division [21], Golden Section[21, 22], and so forth. Although the method of steepest descent and conjugate gradient requires precise one-dimensional searches, Newton's method and quasi-Newton method only needs imprecise searches that guarantees sufficient decrease[23].

### A. Steepest Descent

One of the simplest gradient-based LS is the method of steepest descent introduced by Curry[6], which follows from the fact that the gradient vector points to the direction of fastest increase in the function value. The search points are iteratively generated by stepping in the opposite direction of the gradient. Formally, letting $\hat{x}_k$ denote the current search point, we have

$$\hat{x}_{k+1} = \hat{x}_k - s_k\frac{\nabla F(\hat{x}_k)}{\|\nabla F(\hat{x}_k)\|}$$

Although steepest descent is easy to implement, it is commonly considered slow as the locally optimal direction often does not coincide with the optimal direction in the bigger picture. A zigzag path can often be observed. The convergence rate is no better than first order except for some special cases like concentric contours[24].

Some variants[25, 26] of the basic steepest descent involve deriving the current direction from historical search points in order to reduce the zigzag behavior and accelerate convergence.

### B. Conjugate gradient

In the method of conjugate gradient[7], the search directions are a set of **Q**-conjugate vectors. If a symmetric matrix **Q** is positive definite, non-zero vectors $\hat{v}_1, \hat{v}_2, \ldots \hat{v}_n$ are said to be **Q**-conjugate if

$$\hat{v}_i{}^T\mathbf{Q}\hat{v}_j = 0, \forall i, j \in 1 \ldots n, i \neq j$$

It can be shown all these vectors are linearly independent. With first-order derivatives, we may compute the direction in the $k$th iteration $\hat{v}_k$ with the following formula:

$$\hat{v}_k = -\nabla F(\hat{x}_k) + \alpha_k\hat{v}_{k-1}$$

where

$$\alpha_k = \frac{\nabla F(\hat{x}_k)^T[\nabla F(\hat{x}_k) - \nabla F(\hat{x}_{k-1})]}{\|\nabla F(\hat{x}_{k-1})\|^2}$$

with the initialization $\hat{v}_1 = -\nabla F(\hat{x}_1)$. It is shown to be beneficial to reinitialize the direction vector periodically so as to avoid possible linear dependence in the generated direction vectors $\hat{v}$ [27].

The power of conjugate gradient lies in the fact that $F(\hat{x}_k)$ will probably become separable in the new coordinate system, so each dimension can be minimized individually. For a quadratic function of D dimensions, conjugate gradient needs a maximum of D steps to converge.

### C. Newton's method

Newton's method, or the Newton-Raphson method, was first introduced as a method to solve equations iteratively. For a simple equation $f(x) = 0$ and the current position $x_0$ in the proximity of the solution, the next search point is taken as the intercept of the tangent line at $x_0$ and the $x$ axis. More formally,

$$x_{k+1} = x_k - \frac{f(x_k)}{\nabla f(x_k)}$$

As an optimization method, the equation to be solved is simply changed to $\nabla F(\hat{\theta}_k) = 0$. Substituting corresponding terms, we get:

$$\hat{x}_{k+1} = \hat{x}_k - \mathbf{H}^{-1}\nabla F(\hat{x}_k)$$

For a quadratic objective function with a positive-definite Hessian, a local optimum can be reached with Newton's method in one step. Nonetheless the fast convergence is achieved at a high cost, mainly the effort needed to compute the Hessian and the search direction can be distorted by errors introduced by finite differencing. This inconvenience inspired researchers to look for alternative ways to approximate the Hessian or its inverse and led to the quasi-Newton methods.

### *D. Quasi-Newton methods*

Davidon[28] originated the first method in the quasi-Newton family. It was later refined by Fletcher and Powell[29] and is now called the Davidon-Fletcher-Powell method (DFP). The search direction in DFP is given by:

$$\hat{v}_k = \frac{-\mathbf{H}_k\nabla F(\hat{x}_k)}{\|\mathbf{H}_k\nabla F(\hat{x}_k)\|}$$

where $H_0 = \mathbf{I}$ and

$$\hat{y}_k = \hat{x}_{k+1} - \hat{x}_k = s\hat{v}_k$$

$$\hat{z}_k = \nabla F(\hat{x}_{k+1}) - \nabla F(\hat{x}_k)$$

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\hat{y}_k\hat{y}_k{}^T}{\hat{y}_k{}^T\hat{z}_k} - \frac{\mathbf{H}_k\hat{z}_k(\mathbf{H}_k\hat{z}_k)^T}{\hat{z}_k{}^T\mathbf{H}_k\hat{z}_k}$$

The Broyden-Fletcher-Goldfarb-Shanno or BFGS [30-33] method is slightly different. However, both of the methods belong to the quasi-Newton class characterized by the formula

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\hat{y}_k\hat{y}_k{}^T}{\hat{y}_k{}^T\hat{z}_k} - \frac{\mathbf{H}_k\hat{z}_k(\mathbf{H}_k\hat{z}_k)^T}{\hat{z}_k{}^T H_k\hat{z}_k} + \omega_k(\hat{z}_k{}^T\mathbf{H}_k\hat{z}_k)\hat{p}_k\hat{p}_k{}^T$$

$$\hat{p}_k = \frac{\hat{y}_k}{\hat{y}_k{}^T\hat{z}_k} - \frac{\mathbf{H}_k\hat{z}_k}{\hat{z}_k{}^T\mathbf{H}_k\hat{z}_k}$$

Obviously, the DFP formula is the special case where $\omega_k = 0$ while in BFGS $\omega_k = 1$. It can be shown that $\mathbf{H}_{k+1}$ is positive definite as long as $\mathbf{H}_k$ is positive definite for all $\omega_k \in [0,1]$, if all computation and line searches are exact. In real-world applications, it is beneficial to periodically reset $\mathbf{H}_k$ to $\mathbf{I}$, so as to prevent loss of the positive-definiteness [34].

## III. MEMETIC ALGORITHM

Having seen how creatures incredibly adapt to their environment, one may reasonably conclude that nature is probably the most powerful optimizer. The astonishing elegance of the ways natural creatures solve their daily problems inspired researchers to mimic evolution, nature's way of optimization, in their algorithms. These algorithms simulate the natural selection process and blind mutation and crossover of genes in the hope that the genes will be optimized, in a sense, "automatically".

In 1960s, Holland invented genetic algorithm(GA)[10] and Rechenberg and Schwefel invented evolutionary strategies[11, 12]. In 1985 Cramer[13] conceived genetic programming(GP), in which individuals are represented by tree structures, and Koza[35] refined it later. The three methods differ mostly in the representation of individuals and use of operators such as crossover and mutation. Nevertheless they share more or less a common algorithmic sequence. The three methods are now sometimes grouped together under the collective term evolutionary computing. More recent developments includes Ferreira's Gene Expression Programming[36], which is alleged to be a closer metaphor to real biological processes.

Evolutionary computing, same as the natural processes it mimics, is stochastic. It is able to reach the global optimum with high probability and at lower cost than a multi-start deterministic search. However, while evolutionary computing is generally good at identifying promising regions that are worth exploring further, it may take time to converge within those regions and may not converge to precisely the optimum. On the other hand, deterministic methods converge rapidly near optima. Memetic algorithms that hybridize evolutionary computing and deterministic LSes are thus proposed to perform efficient search with good precision.

Based on the way results of local searches are used, memetic algorithms can be categorized into Baldwinian and Lamarckian. In a Baldwinian MA, the results are merely used in the selection of individuals. In a Lamarckian MA, the results are also translated back to the genotypes. The Lamarckian approach is used in most recent works.

### *A. Sources of Gradients*

While the exact search strategy may vary, the gradient information many method depend on does not differ, except for truncation and rounding errors. In this section we look at different sources of gradient, their availability and cost.

The expense we pay for the gradients is generally limited by our knowledge about the objective function. The most general and widely used method is finite differencing, which requires no other information than measurements of the objective functions. However, FD can be expensive. If more information of the objective function is available, for example, the algorithm to derive the function value, it can be exploited so that the gradient information can be obtained at a much lower cost. In some special cases the gradient may even be readily available. Techniques to compute gradients ranges from Automatic Differentiation and adjoint Euler solver to symbolic differentiation. Even when little information is known, we might still reduce the number of function measurements at the expense of accuracy of the gradient, thanks to stochastic gradient methods such as SPSA.

In the most special case, analytical gradient is readily available. A good example is the Lennard-Jones cluster problem, which aims to find the optimal positions of a cluster of atoms so that the potential energy of the atoms is minimized. The problem has been so well studied that one can easily find analytical gradient in relevant literature. For some other problems, the objective function may be completely specified in formula form while the derivative is not; this knowledge enables us to use computer-assisted symbolic differentiation or even analytical derivation by hand.

In cases where the function is so complex that manual

derivation or symbolic differentiation is unbearably slow, or the function is not known in exact formula form but only algorithmically, i.e. as a series of steps, for instance, in complex simulations, automatic differentiation may be used. Automatic differentiation (AD) is superior to finite differencing because it generates numerical results as precise as one might obtain with analytical gradients. Besides, the results may be generated with reasonable cost as the computational complexity in the reverse mode is independent of the dimensionality of the inputs.

Automatic differentiation is a transformation of the algorithm deriving the function value to the algorithm deriving its derivative. See for [37] a history of automatic differentiation. AD include two strategies, or modes: the forward mode and the reverse mode. For our function of interest $F: \Re^n \to \Re^1$, the reverse mode requires less computational cost than the forward mode and is preferred. However, the reverse mode requires the storage of intermediate results and may pose a challenge for memory capacity when used with very complex algorithms.

There are also cases where gradient is not available in analytical form, but still can be obtained at a much lower cost compared to finite differencing. A class of problems directly taken from engineering practice is the optimization of airfoil design. The optimization of airfoil shapes can be viewed as a constrained optimization problem, where the constraints appear as Euler or Navier-Stokes flow equations. Here, an adjoint Euler solver, first used in [38], may be utilized to compute the gradients at only 1.5 times or 2 times the cost of evaluating the objective function[39], which is, compared to finite differencing, a cheap method.

Nevertheless there are cases that the function value is generated as processes or experiments external to our computer, such as outputs from external circuitry. Little is known about the input-output relationship. In this case the function becomes a black box. Generally, only finite differencing is applicable in these situations.

Finite differencing is derived directly from the definition of partial derivatives. There are several ways to approximate a local derivative. For the objective function $F: \Re^n \to \Re^1$, let $\hat{u}_\ell$ denote the unit vector of the $\ell$th dimension, and $\tilde{g}(\cdot)_\ell$ denote the corresponding approximated partial derivative. For a small positive perturbation $c_k$, the one-sided estimate (forward difference) at the point $\hat{x}_k$ can be express as

$$\tilde{g}(\hat{x}_k)_\ell = \frac{1}{c_k}[F(\hat{x}_k + c_k\hat{u}_\ell) - F(\hat{x}_k)]$$

We might also use the central difference, or the two-sided estimate:

$$\tilde{g}(\hat{x}_k)_\ell = \frac{1}{2c_k}[F(\hat{x}_k + c_k\hat{u}_\ell) - F(\hat{x}_k - c_k\hat{u}_\ell)]$$

If it can be assumed the function is twice differentiable, from the expansion of Taylor series we may obtain:

$$\tilde{g}(\hat{x}_k)_\ell = -\frac{1}{2c_k}[F(\hat{x}_k + 2c_k\hat{u}_\ell) - 4F(\hat{x}_k + c_k\hat{u}_\ell) + 3F(\hat{x}_k)]$$

Although the two-point estimate and the result yielded from Taylor series produces generally more accurate results, the one-point estimate is usually preferred on the grounds of economics. However, if the objective function is rough or noise is present, using the central difference may reduce the noise level and achieve faster convergence[40].

Despite being universally applicable, finite differencing has its own drawbacks. Choosing the value of $c_k$ can be tricky because too great a $c_k$ will, by definition, yields an inaccurate gradient, and too small a $c_k$ may be obscured by the noise in the external system and susceptible to rounding errors. In addition, computing FD is expensive because it requires D+1 measurement for a D-dimensional objective function.

When gradients are not cheaply available, it may be advantageous to use stochastic gradient methods that stochastically approximate the gradient of the function rather than the combination of a deterministic method with gradient obtained from finite differencing[41]. Although the gradient information acquired by stochastic approximation can be tightly coupled with the strategy of its utilization, we still classify these methods mainly as a source of gradient instead of gradient-based methods. In the next a few paragraphs we review the simultaneous perturbation stochastic approximation method (SPSA).

SPSA is due to Spall [42, 43]. The approximated gradient $\hat{g}(\hat{x}_k)$ can be expressed as follows:

$$\tilde{g}(\hat{x}_k) = \frac{F(\hat{x}_k + c_k\Delta_k) - F(\hat{x}_k - c_k\Delta_k)}{2c_k}\begin{bmatrix}\Delta_{k1}^{-1}\\ \Delta_{k2}^{-1}\\ \vdots \\ \Delta_{kp}^{-1}\end{bmatrix}$$

where the components of the perturbation vector $\Delta_k$ is drawn from the Bernoulli ±1 distribution. The independent variable $\hat{x}$ is updated as

$$\hat{x}_{k+1} = \hat{x}_k - a_k\tilde{g}(\hat{x}_k)$$

The choice of the decreasing gain sequence $a_k$ and $c_k$ are respectively determined by $a_k = a/(A+k)^\alpha$ and $c_k = c/k^\gamma$. For effective values of the constants A, a, γ, α, and c, see [44]. For each search point, the algorithm only requires two evaluations of $F(\cdot)$, this is only 1/D of the computational effort required by central finite differencing for a D-dimensional problem.

## IV. Empirical studies

In this section we studied the characteristics of algorithms based on the pseudo-code shown in Fig.1, including three memetic algorithm and one traditional genetic algorithm, in order to illustrate the impact of gradient in MAs. In this study our focus is on the steps marked (6) and (7) in Fig. 1. The first memetic algorithm employed analytical gradient that is assumed to be readily available (MA-AG). The cost of evaluating the gradient is deemed as negligible. It is worth noting that gradients obtained by automatic differentiation are as precise as analytical gradient, only at a slightly higher cost. Consequently, characteristics exhibited by this particular algorithm can easily be extended to cases where AD is used by only multiplying the cost by a small constant.

The second algorithm employed gradient obtained from one-sided finite differencing (MA-FD). The first and second

method differ in the step marked as (6). We used DFP as the local search for these two methods, while the third algorithm utilized SPSA together with its stochastic approximation of the gradient (MA-SPSA). The third MA differs from the previous ones in both steps (6) and (7). Lastly we also studied a pure GA without local searches, in which all steps (6), (7) and (8) are not present.

(1) Initialize: Generate an initial population and randomize the genotypes
(2) Number of generation: k = 0
**while** Stopping conditions are not satisfied do
(3) Evaluate all individuals in the population.
(4) Select individuals to crossover by tournaments
(5) Mutate the descendants
After every $\theta$ generations, **for each** descendant do
(6) Obtain the gradient at the individual's position
(7) Perform individual learning using a gradient-based local search with probability of $f_k$, for at most $t_k$ function evaluations.
(8) Update the individual's genotype with solutions found by the local search
**end for**
(9) k++
**end while**

Fig. 1. Outline of the Memetic Algorithm of Interest

### A. Test Functions and parameters

We studied the characteristics of the above algorithms with four test functions: Sphere, Rastrigin, Ackley and Rosenbrock. See Table I for a list of all test functions and some of their properties. All functions in have 10-dimensional inputs.

In all the algorithms, a population of 100 individuals of real-valued representation with Gaussian mutation and two-point crossover was used. The mutation probability was set to 0.01 and crossover probability was set to 1. The tournament size was set to 2. The local searches were applied after a number of $\theta$ generations, with a probability $f_k$ = 0.05. Our observations showed SPSA usually takes more functions evaluations to converge than DFP with FD and too small a $t_k$ resulted in poor performance. In order to allow SPSA to converge and keep the race fair, we gave SPSA a higher $t_k$ and at the same time lowered its frequency, so that the expected number of function measurements taken by the local searches remained the same. See Table II for a list of $\theta$ against $t_k$. All algorithms employed generational replacement of individuals whereas 10 elites were preserved. 100,000 evaluations were considered for all algorithms. Once the fitness value fall below $10^{-8}$, the algorithm is deemed to have reached the global optimum successfully. The initial step size of the local search is configured to 0.08 Newton step.

The analytical gradients of the functions are derived as follows:

$$\frac{\partial F_{Sphere}}{\partial x_i} = 2x_i$$

$$\frac{\partial F_{Rastrigin}}{\partial x_i} = 2x_i + 20\pi sin\,(2\pi x_i)$$

$$\frac{\partial F_{Ackley}}{\partial x_i} = \frac{2e^{\frac{1}{D}\sum_{j=1}^{D}\cos(2\pi x_j))\pi \sin\,(2\pi x_i)}}{D} + \frac{4x_i e^{-0.2\sqrt{\frac{1}{D}\sum_{j=1}^{D} {x_j}^2}}}{D\sqrt{\frac{1}{D}\sum_{j=1}^{D} {x_j}^2}}$$

$$\frac{\partial F_{Rosenbrock}}{\partial x_i} = \begin{cases} 2(z_1 - 1) + 400x_1({z_1}^2 - z_2), & i = 1 \\ -2 - 200{z_{i-1}}^2 + 400{z_i}^3 + z_i(202 - 400z_{i+1}), & i \in (1, D) \\ -200({z_{D-1}}^2 - z_D), & i = D \end{cases}$$

$$z_i = x_i + 1, \text{for } i \in [1, D]$$

TABLE I TEST FUNCTIONS AND PROPERTIES

| Function | Range | Properties | |
|---|---|---|---|
| | | Epistasis | Multimodality |
| $F_{Sphere} = \sum_{i=1}^{D} {x_i}^2$ | $[-100, 100]^D$ | none | none |
| $F_{Rastrigin} = 10D + \sum_{i=1}^{D} \left({x_i}^2 - 10\cos\,(2\pi x_i)\right)$ | $[-5.12, 5.12]^D$ | none | high |
| $F_{Ackley} = 20 + e - 20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}\right)} - e^{\frac{1}{D}\sum_{i=1}^{D}\cos\,2\pi x_i}$ | $[-32, 32]^D$ | weak | moderate |
| $F_{Rosenbrock} = \sum_{i=1}^{D-1} (100(z_{i+1} - {z_i}^2)^2 + (1 - z_i)^2)\,, z_i = x_i + 1$ | $[-2.048, 2.048]^D$ | high | weak |

TABLE II LOCAL SEARCH PARAMETERS

| Algorithm | Generation gaps $\theta$ | Max function evaluations $t_k$ | Expected LS evaluation per 100 generations |
|---|---|---|---|
| MA-AD | 10 | 200 | 10000 |
| MA-FD | 10 | 200 | 10000 |
| MA-SPSA | 25 | 500 | 10000 |

### B. Result and analysis

The convergence plots of all four algorithms are shown in Fig. 2, and numbers of evaluations are also shown in Table III-VI.

The results obtained in Sphere are summarized in Fig. 2 (a) and Table III. MA-AG and MA-FD spent around 3,000 and 20,000 evaluations respectively before reaching the optimum. It should be noted that the convergence speed was compromised by the small initial step size mentioned earlier. Although SPSA took much more measurements of the function before reaching the optimum, it is still much faster than a pure GA. The results showed that, to a large extent, gradient-based local searches helped the MAs to converge in

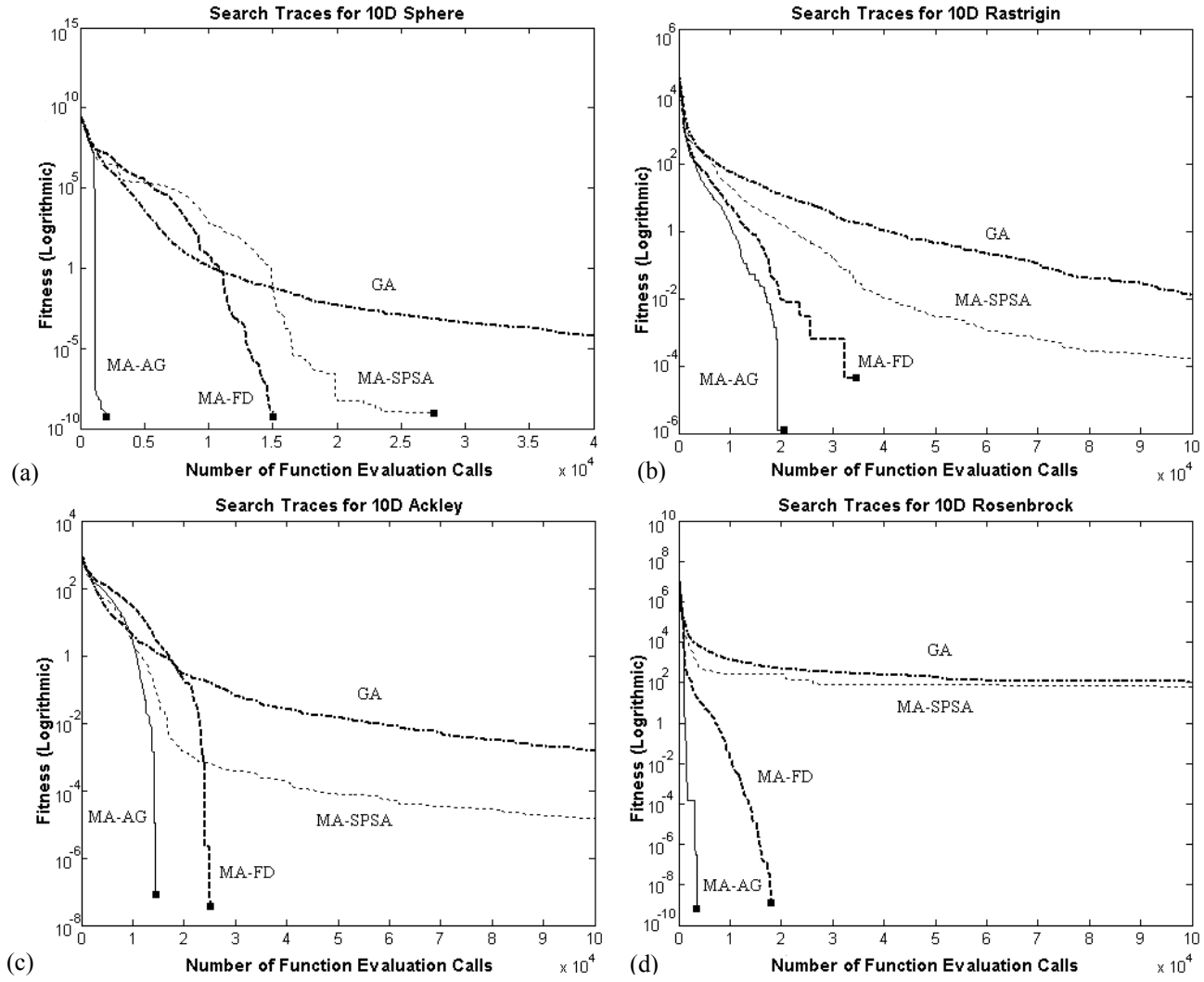


Fig. 2. Fitness versus number of function evaluation calls of test functions averaged over 25 runs.

a unimodal environment.

The next two test problems we studied was the multimodal 10D Rastrigin and Ackley, shown in Figs. 2 (b), (c) and Table IV, V. Even though multimodality can pose a challenge to gradient-based local searches since these methods tend to get trapped at local optima, the gradient-based MAs appears to balance the search exploration and exploitation relatively well. Fig. 3 shows a theoretical "trap" in a local optimum in a one-dimensional function. For any individual, e.g. point A or C, that lies in the basin of attraction, the local search pulls it towards the local minimum point B. For it to escape the local basin trap, the individual relies on the genetic operators, i.e. Gaussian mutation and crossover in the present work, to overcome some high barrier in order to proceed towards the global optimum, e.g. point D. In this manner, all the gradient-based MAs, i.e. MA-AG, MA-FD and MA-SPSA, converged significantly faster than the traditional GA on both the Rastrigin and Ackley problems.

Last but not least, we consider the strongly epistatic 10D Rosenbrock problem. Both MA-AG and MA-FD converged to the global optimum within 2000 evaluations on average. On the other hand, both GA and MA-SPSA did not work as well in comparison. In contrast to SPSA, the local search DFP used in MA-AG and MA-FD is observed to assist the GA in dealing with the strong epistasis of Rosenbrock much better.

In summary, the results indicate that MA-AG outperforms all the other algorithms on all four functions considered, with MA-FD ending up in second place. MA-FD is slower than MA-AG and displays higher standard deviation and lower success rate. The lower performance of MA-FD may be attributed to smaller number of function evaluation budget allocated in the local search phase, since most of this computational budget was spent on estimating the gradient. The effect is also illustrated by the dotted curve in Fig. 3, where a fully/partially converged local search will bring point E to that of nearest local optimum point G or some improved point F.

Although MA-SPSA takes much fewer function evaluation calls for one estimate of gradient than MA-FD, it was still outperformed by MA-FD. Nevertheless, since the cost of approximating the gradient grows with number of dimensions in finite differencing but not in SPSA, the strength of SPSA may not have been fully exploited in the present study where only problems of 10 input dimensions were considered.

Hence a study of MA-FD and MA-SPSA on higher dimensional problems may provide one with greater insights.

Overall, MA converged significantly faster than the traditional GA, suggesting that a gradient-based local search can help improve the performance of genetic algorithms. Further, the success of MA-AG suggests that one is recommended to employ analytical gradient if such precise information can be readily obtained at low cost.

### *C. Conclusion*

In the present study, we observed algorithms that employ gradient information outperform traditional GA. Analytical precise gradient brings considerable benefit to gradient-based local searches. The memetic algorithm with analytical gradient achieved exceptional convergence rate when dealing with functions with high multimodality and epistasis. Though gradient-based local search can be trapped in local optima, they have been shown to complement the traditional GA in forming successful MAs for search.

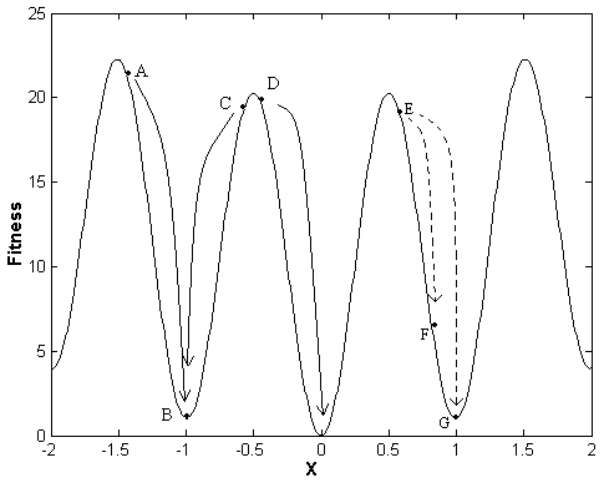


Fig. 3 Theoretical "traps" and effects of a fully/partially converged local search in one-dimensional Rastrigin

TABLE III NUMBER OF EVALUATION AND SUCCESS RATE (10 DIMENSIONAL SPHERE)

| Algorithm | Number of Evaluations and Success Rate | | | | | | | | Avg Errors in Failures |
|---|---|---|---|---|---|---|---|---|---|
| | 1st(best) | 7Th | 13th(med) | 19th | 25th(worst) | Mean | Std | Success | |
| MA-AG | 1795 | 2538 | 2734 | 3109 | 3499 | 2780.72 | 390.10 | 100% | - |
| MA-FD | 16973 | 18475 | 19392 | 20785 | 21805 | 19556 | 1443.84 | 100% | - |
| MA-SPSA | 10533 | 14103 | 31038 | 79129 | 100000 | 32034.04 | 27395.92 | 100% | - |
| GA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 1.72E-3 |

TABLE IV NUMBER OF EVALUATION AND SUCCESS RATE (10 DIMENSIONAL RASTRIGIN)

| Algorithm | Number of Evaluations and Success Rate | | | | | | | | Avg Errors in Failures |
|---|---|---|---|---|---|---|---|---|---|
| | 1st(best) | 7Th | 13th(med) | 19th | 25th(worst) | Mean | Std | Success | |
| MA-AG | 6471 | 11276 | 13048 | 15233 | 20765 | 13526.48 | 3472.18 | 100% | - |
| MA-FD | 14931 | 100000 | 100000 | 100000 | 100000 | 80676.8 | 35162.09 | 24% | 2.69E-8 |
| MA-SPSA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 2.32E-2 |
| GA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 0.150 |

TABLE V NUMBER OF EVALUATION AND SUCCESS RATE (10 DIMENSIONAL ACKLEY)

| Algorithm | Number of Evaluations and Success Rate | | | | | | | | Avg Errors in Failures |
|---|---|---|---|---|---|---|---|---|---|
| | 1st(best) | 7Th | 13th(med) | 19th | 25th(worst) | Mean | Std | Success | |
| MA-AG | 8894 | 12377 | 13194 | 14119 | 17386 | 13438.68 | 1782.47 | 100% | - |
| MA-FD | 81282 | 100000 | 100000 | 100000 | 100000 | 99251.28 | 3743.6 | 4% | 2.44E-8 |
| MA-SPSA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 8.10E-3 |
| GA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 0.0588 |

TABLE VI NUMBER OF EVALUATION AND SUCCESS RATE (10 DIMENSIONAL ROSENBROCK)

| Algorithm | Number of Evaluations and Success Rate | | | | | | | | Avg Errors in Failures |
|---|---|---|---|---|---|---|---|---|---|
| | 1st(best) | 7Th | 13th(med) | 19th | 25th(worst) | Mean | Std | Success | |
| MA-AG | 1159 | 1791 | 2597 | 2911 | 4679 | 2488.16 | 865.92 | 100% | - |
| MA-FD | 13052 | 17499 | 19652 | 21235 | 25069 | 19450.92 | 3100.81 | 100% | - |
| MA-SPSA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 6.02 |
| GA | 100000 | 100000 | 100000 | 100000 | 100000 | 100000 | 0.00 | 0% | 7.74 |

## REFERENCES


[1] R. V. Southwell, *Relaxation Methods in Engineering Science: A treatise on approximate computation*. Oxford: The Clarendon Press, 1940.

[2] M. Friendmann, L. and J. Savage, "Planning experiments seeking maxima," in *Eisenhart, Hastay and Wallis*, 1947, pp. 365-472.

[3] J. M. Ortega and M. L. Rockoff, "Nonlinear difference equations and Gauss-Seidel type iterative methods," *SIAM J. Numer. Anal.,* vol. 3, pp. 497-513, Sep. 1966.

[4] W. H. Swann, "Report on the development of a new direct search method of optimization," I.C.I. Central Instrument Lab, Middlesborough, Tech. Rep. 64/3, 1964.

[5] H. H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *Comp. J.,* vol. 3, pp. 175-184, Mar. 1960.

[6] H. B. Curry, "The method of steepest descent for non-linear minimization problems," *Quart. Appl. Math.,* vol. 2, pp. 258-261, 1944.

[7] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Natl. Bur. Stand.,* vol. 49, pp. 409-436, 1952.

[8] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.

[9] A. P. Ruszczyński, *Nonlinear Optimization*. Princeton, N.J.: Princeton University Press, 2006.

[10] J. H. Holland, *Adaptation In Natural And Artificial Systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Ann Arbor: University of Michigan Press, 1975.

[11] I. Rechenberg, *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Fommann-Holzboog, 1973.

[12] H.-P. Schwefel, *Numerische Optimierung von Computermodellen mittels der Evolutionsstrategie*. Basel: Birkhaeuser, 1977.

[13] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of the 1st International Conference on Genetic Algorithms*: Lawrence Erlbaum Associates, Inc., 1985.

[14] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms," California Institute of Technology, Pasadena, CA, Tech. Rep. 826, 1989.

[15] R. Dawkins, *The Selfish Gene*. Oxford: Oxford University Press, 1976.

[16] Z. Zhu, Y. S. Ong, and M. Dash, "Wrapper-filter feature selection algorithm using a memetic framework," *IEEE Trans. Syst., Man., Cybern., B.,* vol. 37, pp. 70-76, Feb. 2007.

[17] Y. S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust aerodynamic design," *IEEE Trans. Evol. Comput.,* vol. 10, pp. 392-404, Aug. 2006.

[18] F. Neri, J. Toivanen, G. L. Cascella, and Y. S. Ong, "An adaptive multimeme algorithm for designing HIV multidrug therapies," *IEEE/ACM Trans. Comput. Biol. Bioinformatics,* vol. 4, pp. 264-278, Apr. 2007.

[19] J. Tang, M. H. Lim, and Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Comput.,* vol. 11, pp. 873-888, 2007.

[20] R. Salomon, "Evolutionary algorithms and gradient search: similarities and differences," *IEEE Trans. Evol. Comput.,* vol. 2, pp. 45-55, Jul. 1998.

[21] J. Kiefer, "Sequential minimax search for a maximum," *Proc. Amer. Math. Soc.,* vol. 4, pp. 502-506, Jun. 1953.

[22] S. M. Johnson, "Best exploration for maximum is Fibonaccian," RAND Corporation, Santa Monica, CA, Tech. Rep. P-856, 1956.

[23] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 1999.

[24] G. E. Forsythe, "On the asymptotic directions of the s-dimensional optimum gradient method," *Numer. Math.,* vol. 11, pp. 57-76, Jan. 1968.

[25] G. E. Forsythe and T. S. Motzkin, "Acceleration of the optimum gradient method," *Bull. Amer. Math. Soc.,* vol. 57, pp. 304-305, Jul. 1951.

[26] R. J. Buehler, B. V. Shah, and O. Kempthrone, "Some properties of steepest ascent and related procedures for finding optimum conditions," Iowa State University, Statistical Laboratory, Ames, IA, Tech. Rep. 1, 1961.

[27] H. Crowder and P. Wolfe, "Linear convergence to the conjugate gradient method," IBM T.J. Watson Research Center, Yorktown Heights, NY, Tech. Rep. RC-3330, 1972.

[28] W. C. Davidon, "Variable metric method for minimization," Argonne National Laboratory, Lemont, IL, Tech. Rep. ANL-5990, 1959.

[29] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *Comp. J.,* vol. 6, pp. 163-168, 1963.

[30] C. G. Broyden, "The convergence of a class of double-rank minimization algorithms," *JIMA,* vol. 6, pp. 76-90, 1970.

[31] R. Fletcher, "A new approach to variable metric algorithms," *Comp. J.,* vol. 13, pp. 317-322, 1970.

[32] D. Goldfarb, "A family of variable metric updates derived by variational means," *Math. Comput.,* vol. 24, pp. 23-26, 1970.

[33] D. F. Shanno, "Conditioning of quasi-Newton methods for function minimization," *Math. Comput.,* vol. 24, pp. 647-656, Jul. 1970.

[34] Y. Bard, "On a numerical instability of Davidon-like methods," *Math. Comput.,* vol. 22, pp. 665-666, 1968.

[35] J. R. Koza, *Genetic Programming: On the programming of computers by means of natural selection*. Cambridge, Mass.: MIT Press, 1992.

[36] C. Ferreira, "Gene Expression Programming: a New Adaptive Algorithm for Solving Problems," *Compl. Sys.,* vol. 13, pp. 87-129, 2001.

[37] M. Iri, "History of automatic differentiation and rounding error estimation," in *Automatic Differentiation of Algorithms: Theory, implementation, and application*, A. Griewank and G. F. Corliss, Eds. Philadelphia, PA: SIAM, 1991, pp. 3-16.

[38] A. Jameson, "Aerodynamic design via control theory," *J. Sci. Comp.,* vol. 3, pp. 233-260, Sep. 1988.

[39] Y. S. Ong, K. Y. Lum, and P. B. Nair, "Hybrid evolutionary algorithm with Hermite radial basis function interpolants for computationally expensive adjoint solvers," *Comput. Optim. Appl.,* In press.

[40] H. J. Kushner and D. S. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. New York: Springer-Verlag, 1978.

[41] D. C. Chin, "Comparative study of stochastic algorithms for system optimization based on gradient approximations," *IEEE Trans. Syst., Man., Cybern., B.,* vol. 27, pp. 244-249, Apr. 1997.

[42] J. C. Spall, "A stochastic approximation algorithm for large-dimensional systems in the Kiefer-Wolfowitz setting," in *Proc. IEEE. Conf. Decision. and Control*, Austin, Texas, 1988, pp. 1544-1548.

[43] J. C. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Trans. Autom. Control,* vol. 37, pp. 332-341, Mar. 1992.

[44] J. C. Spall, "Implementation of the simultaneous perturbation algorithm for stochastic optimization," *IEEE Trans. Aerosp. Electro. Sys.,* vol. 34, pp. 817-823, Jul. 1998.