Optimization Using a Modified Second-Order Approach With Evolutionary Enhancement

Joel D. Hewlett, *Student Member, IEEE*, Bogdan M. Wilamowski, *Fellow, IEEE*, and Günhan Dündar, *Member, IEEE*

***Abstract*—An optimization algorithm is presented which effectively combines the desirable characteristics of both gradient descent and evolutionary computation into a single robust algorithm. The method uses a population-based gradient approximation which allows it to recognize surface behavior on both large and small scales. By adjusting the population radius between iterations, the algorithm is able to escape local minima by shifting its focus onto global trends rather than local behavior. The algorithm is compared experimentally with existing methods over a set of relevant test cases, and each method is ranked on the basis of both reliability and rate of convergence. For each case, the algorithm is shown to outperform other methods in terms of both measures of performance, truly making it the best of both worlds.**

***Index Terms*—Evolutionary algorithms, gradient descent, optimization.**

I. Introduction

Optimization algorithms usually come in one of two flavors, generally referred to as gradient descent and evolutionary computation. To put it plainly, a method is either fast or powerful. Although this distinction is not necessarily valid for every case, it is still a widely accepted generalization. Each method has its own niche, and comparison of any two methods is highly subjective on the basis of application. Still, the use of optimization algorithms continues to see rapid growth in a number of diverse fields ranging from robotics [1] and computational intelligence [2]–[5] to wireless transmission [6], [7] and digital filter design [8], [9]. With this continued increase in interest and application, the demand for a newer and more versatile approach has become evident. While attempts have been made at bringing the two sides together [10]–[12], it is the purpose of this paper to help further this cause, with the ultimate goal being a single robust algorithm which encompasses the strengths of both methods, making itself useful over a wider range of problems.

The content herein is the continuation of work previously published in [13]. Since its original publication, the method has been seeing a series of drastic changes and improvements, which are the subject of this paper. While the content presented is complete, and should not require any supplemental reading, the work presented in [13] may still be of interest to the reader. Although the method has changed significantly, the underlying principles are still the same.

The remainder of this paper is organized as follows. Section II presents an overview of the proposed method using a simple illustration. Section III contains a technical explanation which includes the details of the algorithm's operation. In Section IV, a set of test functions is presented, and the relevance of each function is discussed. A comparative analysis of the algorithm's performance is presented in Section V. This paper concludes with a short summary in Section VI.

Manuscript received June 11, 2008. First published July 9, 2008; last published August 29, 2008 (projected). This work was supported in part by the National Science Foundation under Contract NSF OISE 0352771.

J. D. Hewlett and B. M. Wilamowski are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: hewlejd@auburn.edu).

G. Dündar is with the Department of Electrical and Electronic Engineering, Boğaziçi University, Istanbul 34342, Turkey.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2008.927987

II. Bigger Picture

Gradient methods are often compared to a ball rolling along some surface. Neglecting momentum and assuming that the ball is infinitely small, it will follow the path of steepest descent exactly. This is a powerful illustration which, consequently, also highlights the method's fundamental flaws. With little effort, one can visualize any number of surfaces on which the ball might become permanently trapped before ever reaching its desired destination. Take, for example, a flight of stairs as in Fig. 1. Even though the behavior of the surface is relatively simple, it is clear that no matter where the ball on the left is placed, it will never find its way to the foot of the stairs. Although the staircase clearly exhibits a global trend which may be determined with only a limited amount of information, the ball is still incapable of proceeding. The advantage of this illustration is that it makes the solution blatantly obvious: Use a bigger ball. Although changing the size of the ball would not change the gradient of the surface, it will make the ball more sensitive to the large-scale behavior, allowing it to successfully descend the stairs. After all, all of the relevant information for descending a staircase can be found at the corners of the individual steps. While steps may be an essential part of a staircase, from the perspective of minimization, they are nothing more than noise. Furthermore, by actively allowing the ball to expand and contract as it descends, it is possible to navigate on as large or small a scale as necessary. This is the basic principle by which the proposed method operates.

III. Proposed Method

The key feature of the proposed method is the way in which the gradient is approximated. During each iteration, the

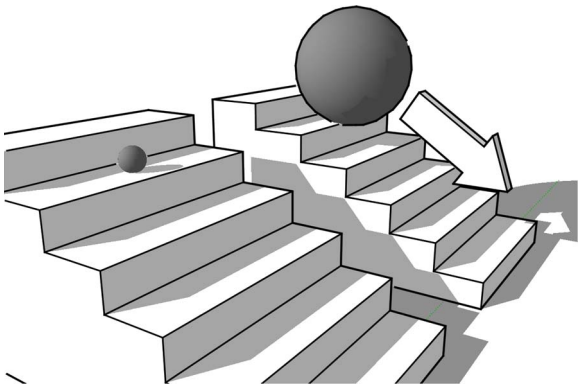

Fig. 1. (Left) Ordinary gradient methods are easily trapped on complex or piecewise constant optimization surfaces. (Right) The proposed method is able to overcome these obstacles by expanding the radius of its population-based gradient approximation.

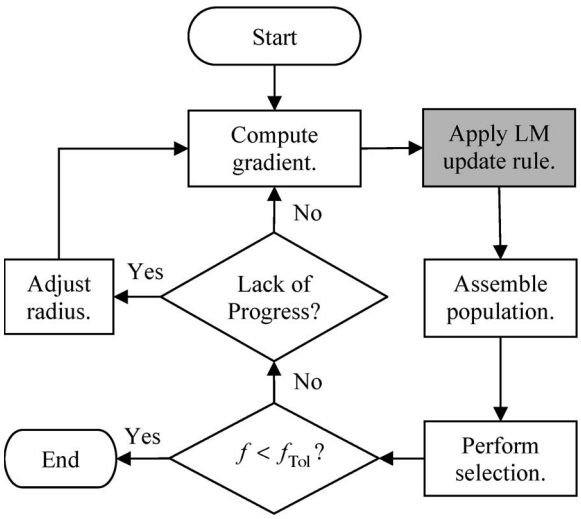

Fig. 2. Simplified flowchart depicts the basic operation of the proposed algorithm.

algorithm generates a "population" of points within a spherically bounded region. A portion of the generated set lies on the perimeter of the bounded region, and it is these members of the population which are used in the approximation of the gradient. Next, an additional point is generated in the direction of the approximated gradient and is added to the current population. Finally, the fittest member of the population is compared to the current best solution, and the population's center and radius are updated accordingly. The basic flow of the algorithm is shown in Fig. 2.

A. Approximating the Gradient

As previously noted, the gradient approximation is the defining feature of the proposed method. The generalized process for computing it is presented in the following in a step-by-step manner.

Let $\mathbf{x}_0 \in \mathbf{R}^n$ be the center of a population P with radius r, and let $y_0 = f(\mathbf{x}_0)$ be the value of the objective function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ at $\mathbf{x}_0$.

1) First, generate a set of n random vectors $\{\Delta\mathbf{x}_1, \Delta\mathbf{x}_2, \ldots, \Delta\mathbf{x}_n\}$, each of length r, and define an $n \times n$ matrix
$$\Delta\mathbf{X} = \begin{bmatrix} \Delta\mathbf{x}_1 \\ \vdots \\ \Delta\mathbf{x}_n \end{bmatrix}.$$
2) Now, let $\mathbf{x}_i = \Delta\mathbf{x}_i + \mathbf{x}_0$, and define the vector $\mathbf{y} = [y_1, \ldots, y_n]^{\mathrm{T}}$, where
$$y_i = f(\mathbf{x}_i) \tag{1}$$
for $i = 1, \ldots, n$. It should be clear from (1) that each value y_i corresponds to the value of the objective function at the randomly generated point $\mathbf{x}_i$, which lies on the perimeter of the population.
3) Finally, compute the gradient approximation using
$$\nabla f = \Delta\mathbf{X}^{-1} \cdot (\mathbf{y} - y_0). \tag{2}$$

It may appear, and understandably so, that a significant leap has been made between steps 2) and 3). Therefore, for the sake of clarity, a short derivation of (2) is included.

Let f be an n-dimensional scalar function. A linear approximation of f with respect to some point $\mathbf{x}_0 = (x_{01}, x_{02}, \ldots, x_{0n})$ may be found using the first two terms of the Taylor series expansion

$$\begin{aligned} f(\mathbf{x}) &\approx f(\mathbf{x}_0) + \frac{\partial f}{\partial x_1} \cdot \Delta x_1 + \frac{\partial f}{\partial x_2} \cdot \Delta x_2 + \cdots + \frac{\partial f}{\partial x_n} \cdot \Delta x_n \\ &= f(\mathbf{x}_0) + \nabla f|_{\mathbf{x}_0} \cdot \Delta\mathbf{x}. \end{aligned} \tag{3}$$

Solving this equation for the gradient yields

$$\nabla f|_{\mathbf{x}_0} \approx \frac{\Delta f}{\Delta\mathbf{x}} = \frac{f(\mathbf{x}) - f(\mathbf{x}_0)}{\mathbf{x} - \mathbf{x}_0}. \tag{4}$$

By generating a point $\mathbf{x}$ in the vicinity of $\mathbf{x}_0$, it is possible to obtain a numerical approximation of $\nabla f|_{\mathbf{x}_0}$ using (4). However, for $n > 1$, (4) is an invalid expression because it implies that $\Delta\mathbf{x}$ is a vector and therefore has no inverse. Thus, in order to generalize (4), f is evaluated for a set of n linearly independent points, allowing $\Delta\mathbf{x}$ and Δf to be replaced by the $n \times n$ matrix $\Delta\mathbf{X}$ and the $n \times 1$ vector of corresponding function values $\Delta\mathbf{f}$. This leads to the generalized form

$$\begin{aligned} \nabla f_{n\times 1} &\approx \begin{bmatrix} \Delta x_{11} & \cdots & \Delta x_{1n} \\ \vdots & \ddots & \vdots \\ \Delta x_{n1} & \cdots & \Delta x_{nn} \end{bmatrix}^{-1}_{n\times n} \begin{bmatrix} f(\mathbf{x}_1) - f(\mathbf{x}_0) \\ \vdots \\ f(\mathbf{x}_n) - f(\mathbf{x}_0) \end{bmatrix}_{n\times 1} \\ &\approx \Delta\mathbf{X}^{-1} \cdot \Delta\mathbf{f} \end{aligned} \tag{5}$$

which is equivalent to (2).

B. Modified LM Update Rule

Once the gradient is computed, an additional point $\mathbf{x}_{n+1}$ is generated in the direction of the approximation. The proposed method uses a modified version of the Levenberg–Marquardt (LM) algorithm [14] update rule for this purpose. For a scalar-valued function $f : \mathbf{R}^n \rightarrow \mathbf{R}$, the LM update rule may be written as

$$\mathbf{x}_{n+1}^{(k+1)} = \mathbf{x}_0^{(k)} - (\nabla f \cdot \nabla f^{\mathrm{T}} + \mu \mathbf{I})^{-1} \nabla f \cdot y_0^{(k)} \tag{6}$$

where k is the current iteration and μ acts as a damping factor which reduces oscillation by actively controlling the step size. One who is familiar with the LM algorithm may notice that the Jacobian has been replaced by the gradient in (6). This is because the Jacobian is defined as the matrix of all first-order partial derivatives of a *vector-valued* function. Thus, for a scalar-valued function, the Jacobian is, by definition, the transpose of the gradient. Although the proposed method may be extended to handle vector-valued functions, the version presented here is intended for use with scalar-valued problems. Thus, the Jacobian is replaced by the gradient. This is not the only modification. For the proposed implementation, the rule is also modified to account for the population radius r. This is done by adding a term which ensures that the length of the update step will be no smaller than the population radius r. This modification is needed to ensure that when the algorithm encounters local minima, the step size will exceed the radius of the current population, thus providing greater diversity. The modified version of the rule is written as

$$\mathbf{x}_{n+1}^{(k+1)} = \mathbf{x}_0^{(k)} - (\nabla f \cdot \nabla f^{\mathrm{T}} + \mu \mathbf{I})^{-1} \nabla f \cdot y_0^{(k)} + \frac{\nabla f}{\|\nabla f\|} \cdot r^{(k)}. \tag{7}$$

The added term in (7) is simply a vector of length r in the direction of ∇f.

It is important to note that the algorithm presented here, using the modified version of the LM update rule, is but one of many possible implementations. In other words, the shaded box in Fig. 2 may be replaced with any number of existing gradient methods without compromising the underlying principles. The only requirement is that the gradient be calculated in the previously described fashion. In fact, the proposed method was specifically designed with this sort of flexibility in mind. Therefore, the update rule may be readily exchanged with another second-order method such as the BFGS method [15] or the closely related Davidon–Fletcher–Powell algorithm, which has even been shown to outperform LM in certain applications [16].

C. Assembling the Population

The population P is generated in two parts with a combined size of λ. The primary population

$$A = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}\} \tag{8}$$

is created in the two previous steps, and is of size $n+1$, whereas the secondary population

$$B = \{\mathbf{x}_{n+2}, \ldots, \mathbf{x}_{\lambda}\} \tag{9}$$

consists of a set of m points which are randomly distributed within the region defined $\mathbf{x}_0$ and r. Here, the value of m is a user-defined parameter and may be assigned any nonnegative integer value including zero. This leads to the following definitions for P and λ:

$$P = \{\mathbf{x}_1, \ldots, \mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots, \mathbf{x}_{\lambda}\} = A \cup B \tag{10}$$

$$\lambda = n + 1 + m. \tag{11}$$

While the inclusion of A in P can provide a noticeable increase in the rate of convergence when applied to relatively complex optimization surfaces, in general, it is sufficient to let $m = 0$.

D. Selection Process

Once the P has been constructed according to (10), its members are ranked with respect to fitness. Next, the fittest member $\mathbf{x}_{\mathrm{opt}} \in P$ is chosen, and the following condition, also shown in Fig. 2, is evaluated:

IF $f(\mathbf{x}_{\mathrm{opt}}) < f_{\mathrm{Tol}}$ **THEN**
  Terminate algorithm.
ELSE IF $f(\mathbf{x}_{\mathrm{opt}}) \geq f(\mathbf{x}_0)$ **THEN**
  Adjust population radius r.
ELSE
  $\mathbf{x}_0 = \mathbf{x}_{\mathrm{opt}}$.
END IF

where f_{Tol} is the maximum acceptable value for the objective. Updating in this way essentially recycles the information used in the approximation of the gradient, which would otherwise be thrown away. This process also helps to guarantee the algorithm's stability because it is equivalent to running two methods in parallel. That is, if the generated step does not result in a lower value of the objective, the method will resort to selecting the best of the points used in the gradient approximation. In this way, it is still possible for the method to proceed. Furthermore, because the current best solution is included in the selection process, it is guaranteed that the value of the objective will not increase from one iteration to the next. Therefore, while the instability of the update rule may affect the algorithm's convergence, the algorithm itself will remain stable as a result of selection.

E. Adjustment of the Population Radius

Any number of radius update rules may be devised; however, the version presented here is perhaps the simplest. The method uses a fixed step size Δ to modify the radius over a bounded user-defined interval $[r_{\min}, r_{\max}]$, and is subject to the following conditions:

IF $r \geq r_{\max}$ **THEN**
  $r^{(k+1)} = r^{(k)} + \Delta$
ELSE
  $r^{(k+1)} = r_{\min}$
END IF

where k is the number of the current iteration. Although this method is simple, it proves to be adequate. The two features

which make the method so basic are the use of a resetting value of r, and a constant step size Δ. The constant step size serves the purpose of increasing the population radius when the algorithm encounters local minima, whereas the increase in r causes the gradient approximation to become less sensitive to local behavior, thereby allowing it to escape the traps introduced by complex local behaviors. Conversely, if greater accuracy of approximation becomes necessary for local search, the algorithm is still able to proceed once r is reset.

IV. Test Functions

Four unique functions were used for testing. Each function exhibits features deemed relevant for the purpose of comparison. All four functions are presented in generalized form, with n being the dimension of the search space.

A. Test Function 1

$$T_1(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{x_i}{4}\right)^4, \qquad x_i \in [-10, +10].$$

The first test function, T1, has a simple convex continuous parabolic surface with a minimum of $T_1(\mathbf{x}) = 0$ located at the origin. The function is used for two primary purposes. First, it offers a fair comparison of the proposed method with some of the more common gradient methods. Second, it highlights the strengths of such methods, which are superior to evolutionary methods when applied to problems of this type. Also, because the function is fourth order, it will highlight the difference between algorithms of higher and lower orders.

B. Test Function 2

$$T_2(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{\lfloor x_i \rfloor}{4}\right)^4, \qquad x_i \in [-10, +10].$$

The second test function, inspired by the illustration in Section II, is of an identical form to that of T1, except that the variables have been floored in order to make it piecewise constant. T2 has a minimum value of zero for all $\mathbf{x} = [x_1, \ldots, x_n]$ which satisfy $x_i \in [0, 1)$ for $i = 1, \ldots, n$. The features of this function are useful for testing the hypothesis made in Section II. If the proposed method operates as intended, there should be little difference in performance between T1 and T2. The surface of T2 is shown in Fig. 3.

C. Test Function 3

$$f(\mathbf{x}) = \frac{1}{2}\sqrt{\sum_{i=1}^{n} \lfloor x_i \rfloor^2}, \qquad x_i \in [-100, +100]$$

$$g(\mathbf{x}) = \frac{\mathbf{x}}{4} + (1 - \cos(\pi \mathbf{x})) \cdot \left(\tanh\left(\frac{\mathbf{x}}{4}\right) - 1\right)^2$$

$$T_3(\mathbf{x}) = g \circ f.$$

Like T2, T3 is also piecewise constant, but with the addition of local minima, which is the most common challenge faced by gradient methods. Although evolutionary methods may also become susceptible to these traps if population diversity becomes too low, they are still better suited for this type of problem. T3 has a minimum value of zero for all $\mathbf{x} = [x_1, \ldots, x_n]$ which satisfy $x_i \in [0, 1)$ for $i = 1, \ldots, n$. The surface of T3 is represented in Fig. 4.

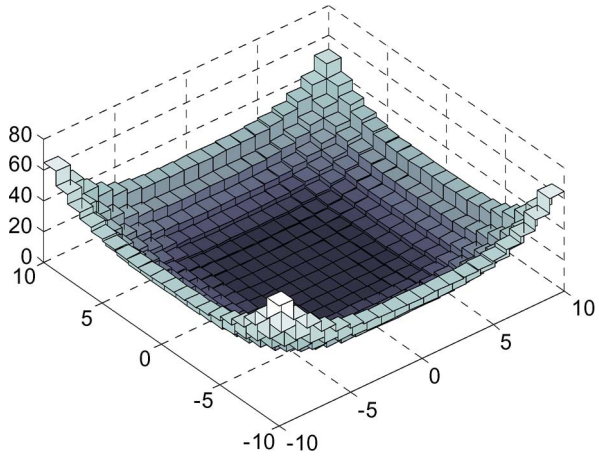

Fig. 3. Piecewise-constant surface of T2 has a gradient of zero everywhere, making it impossible for gradient methods to minimize.

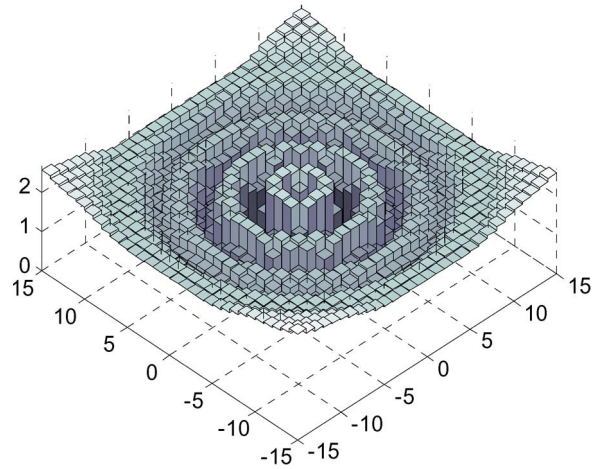

Fig. 4. T3 combines the zero-gradient characteristics of T2 with the added challenge of local minima.

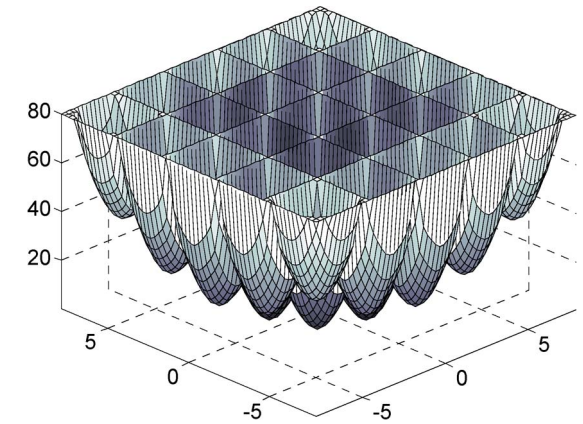

Fig. 5. Cross-sectional view of T4 shows the severity of the numerous local minima.

TABLE I
LIST OF ALGORITHM PARAMETERS USED IN TESTING

Algorithm	Parameters	T1	T2	T3	T4
MSD	α	1	1	1	1
LM	μ_0	0.1	0.1	0.1	0.1
SA-ES	μ, λ, σ_0	3, 12, 1	3, 12, 1	8, 16, 10	8, 16, 50
CMA-ES	μ, λ, σ_0	3, 12, 1	3, 12, 1	8, 16, 10	8, 16, 50
PM	$\mathrm{m}, \mathrm{r_{min}}, \mathrm{r_{max}}, \Delta$	0, 10^{-6}, 1, .1	0, 2, 6, 2	4, 2, 6, 2	0, 10^{-6}, 12, 3

TABLE II
SUCCESS RATE AND AVERAGE NUMBER OF FUNCTION EVALUATIONS

Success Rate		Test Function			
Mean evaluations		T1	T2	T3	T4
Algorithm	MSD	**100 %**	FAILURE	FAILURE	FAILURE
		5477.98	FAILURE	FAILURE	FAILURE
	LM	**100 %**	FAILURE	FAILURE	FAILURE
		43.6	FAILURE	FAILURE	FAILURE
	SA-ES	**100 %**	**100 %**	**100 %**	48 %
		214.48	426.28	355.72	6475.7
	CMA-ES	**100 %**	**100 %**	**100 %**	67 %
		186.68	138.32	772.24	815.61
	PM	**100 %**	**100 %**	**100 %**	**100 %**
		43.6	**28.72**	**148.21**	**382.36**

D. Test Function 4

$$T_4(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n}\left(x_i^2 + \tan(x_i)^2 - 10\cdot\cos(2\cdot x_i) + 10\right),$$

$$x_i \in [-100, +100].$$

T4 is the most extreme of the test functions. Although it is piecewise continuous, each of the regions of continuity is a deep local minimum. Thus, gradient methods are only capable of minimizing T4 on a local basis. Due to the large number of local minima, evolutionary methods which rely on modified mutation strength for accelerated convergence may also be susceptible to entrapment. The surface of T4 is shown in Fig. 5. In reality, the "walls" which separate the local minima in Fig. 5 are infinitely high. The minimum of T4 is zero and is located at the origin.

V. EXPERIMENTAL RESULTS

The proposed method is compared with four other known algorithms. It should be noted that the purpose of the comparison is not to show that the proposed method is superior to all algorithms over all problems; clearly, that is not the case. Instead, the goal is to show that while the algorithm shows the high rate of convergence and efficient local search characteristics of a second-order gradient method, it is also capable of minimizing complex classes of problems which are usually associated with evolutionary methods. Thus, the evolutionary methods used for comparison were selected on the basis of these same qualities. The following is a list of the compared methods.

1) MSD: The Method of Steepest Descent is perhaps the most well known of all gradient methods. MSD is a simple first-order method which, as the name implies, employs a user-defined step size to proceed in the direction of "steepest descent." The method was chosen for its high degree of stability as well as its reputation as the standard gradient method.
2) LM: The LM algorithm, described briefly in Section III-C, is regarded as one of the fastest gradient methods available. The method was chosen as a benchmark among second-order algorithms. Furthermore, the gradient portion of the proposed method uses an update rule directly inspired by the LM algorithm, making the method particularly relevant for comparison.
3) (μ, λ)-SA-ES: Self-Adaptive Evolution Strategy [17], a member of the larger family of algorithms known as Evolutionary Strategies [17], is a powerful evolutionary method which uses self-adapted mutation strength for optimal convergence as well as an accelerated local search. The method was chosen as a strong representative of the power of evolutionary methods.
4) CMA-ES: Covariance Matrix Adaptation Evolutionary Strategy [18], also a member of the family of Evolutionary Strategies, is an evolution-path-related technique which uses search-space information in a highly efficient manner, making it exceptionally fast with respect to other evolutionary methods.
5) PM: The proposed method.

The performance of each algorithm on each of the test functions was evaluated over a series of 100 simulation runs using the algorithm parameters listed in Table I. Each algorithm was then evaluated with respect to rate of success as well as the mean number of function evaluations per solution. The tabulated results are presented in Table II, which represents the case $n = 2$ for all four functions. For each of the test functions, success was defined by the following two conditions: 1) $f(\mathbf{x}) < 10^{-6}$ and 2) no more than 10^5 total evaluations of the objective.

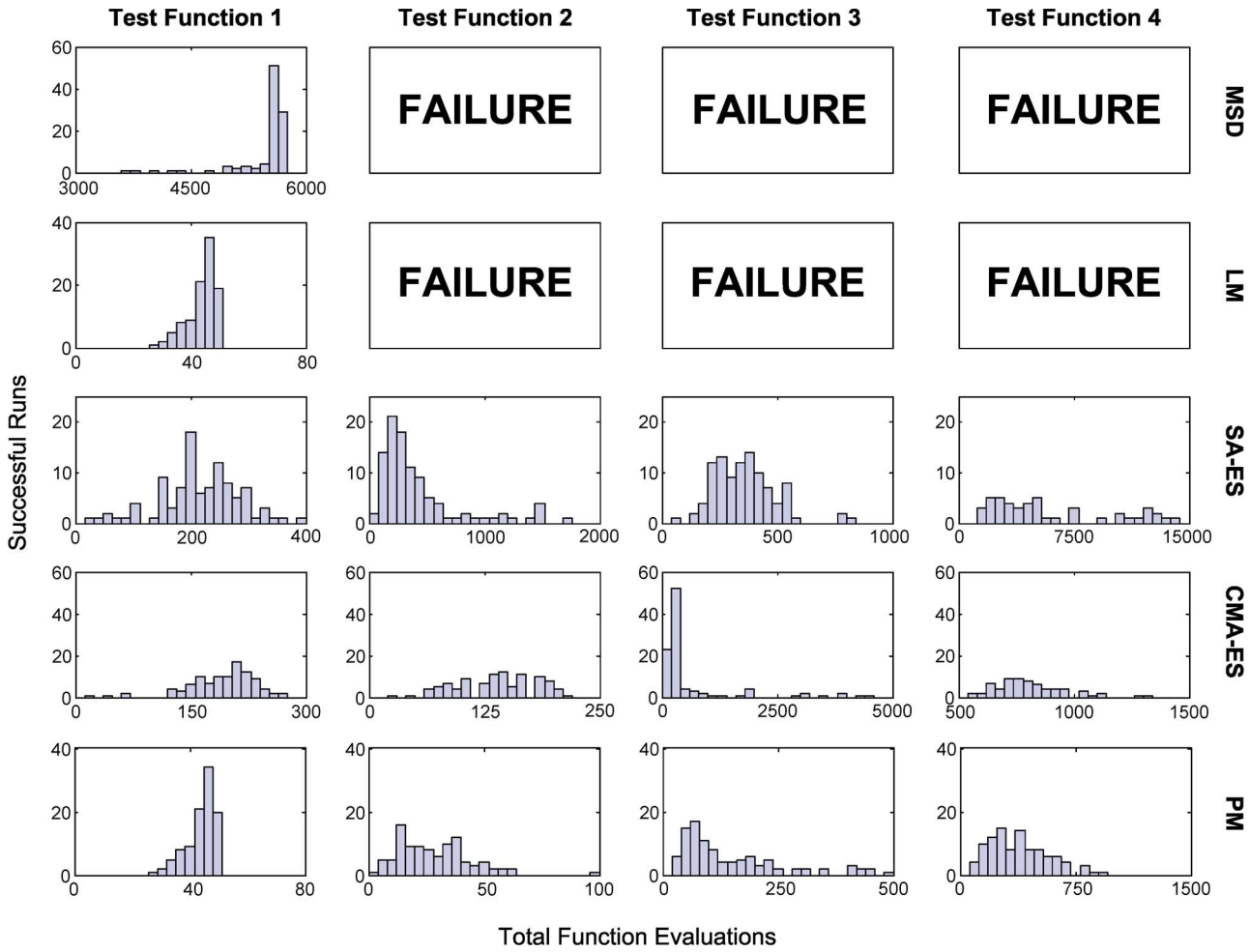

Fig. 6. Graphical summary of the test results.

The fields in Table II highlighted in bold font denote the top performers for each of the test cases. One may notice the similarity between the proposed method and the LM algorithm with regard to the first test function. Notice that, for this particular function, the behaviors of the two algorithms are nearly identical. This validates the earlier hypothesis that, in the absence of local minima, the two methods should behave in a similar manner. Note too the striking difference in the rate of convergence between the two second-order gradient methods when compared with the first-order method of steepest descent. Another point of interest is the performance of the proposed method on the second test. It was suggested in Section IV-B that, assuming proper performance, little difference should be observed between T1 and T2. This is confirmed by the results in Table II. In fact, the algorithm actually performs better. This behavior is a result of the increased population radius coupled with the modified LM update rule, which, when used together, increase the rate at which the proposed method traverses the optimization surface.

Finally, notice the similarity between the proposed method and that of CMA-ES with respect to T3 and T4. In fact, for T4, the similarities are striking. When applied to these more complex surfaces, the behavior of the algorithm changes drastically, and yet there seems to be little, if any, effect on the level of performance. What is most striking about the results in Fig. 6 is the change in behavior between T1 and T4. As the nature of the test functions becomes more complex, the behavior of the proposed method changes from a second-order gradient method to one which bears a striking resemblance to that of evolutionary computation.

VI. Conclusion

An optimization algorithm was presented, which incorporates features of both gradient and evolutionary methods. The algorithm was tested over a series of test functions, each chosen to typify commonly faced difficulties. The proposed method was then compared with a set of known algorithms which included both gradient and evolutionary methods. It was shown from the results that the proposed method was able to not only emulate both classes of algorithms based on the behavior of a given function but also outperform the compared methods in terms of rate of success and average number of evaluations of the objective. While the presented implementation shows a great deal of promise, its discussion is intended only as an introduction to the proposed method. Thus, for the sake of simplicity, a number of known improvements have been omitted for inclusion in future versions of the algorithm.

REFERENCES

[1] J. C. Derenick and J. R. Spletzer, "Convex optimization strategies for coordinating large-scale robot formations," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1252–1259, Dec. 2007.

[2] S.-B. Roh, W. Pedrycz, and S.-K. Oh, "Genetic optimization of fuzzy polynomial neural networks," *IEEE Trans. Ind. Electron.*, vol. 54, no. 4, pp. 2219–2238, Aug. 2007.

[3] L. dos Santos Coelho and B. M. Herrera, "Fuzzy identification based on a chaotic particle swarm optimization approach applied to a nonlinear yo-yo motion system," *IEEE Trans. Ind. Electron.*, vol. 54, no. 6, pp. 3234–3245, Dec. 2007.

[4] S.-K. Oh, W. Pedrycz, and H.-S. Park, "A new approach to the development of genetically optimized multilayer fuzzy polynomial neural networks," *IEEE Trans. Ind. Electron.*, vol. 53, no. 4, pp. 1309–1321, Aug. 2006.

[5] F.-J. Lin, P.-K. Huang, and W.-D. Chou, "Recurrent-fuzzy-neural-network-controlled linear induction motor servo drive using genetic algorithms," *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1449–1461, Jun. 2007.

[6] F. Grimaccia, M. Mussetta, P. Pirinoli, and R. E. Zich, "Genetical swarm optimization (GSO): A class of population-based algorithms for antenna design," in *Proc. 1st ICCE*, Oct. 10–11, 2006, pp. 467–471.

[7] F. Grimaccia, M. Mussetta, P. Pirinoli, and R. E. Zich, "Optimization of a reflectarray antenna via hybrid evolutionary algorithms," in *Proc. 17th Int. Symp. EMC-Zurich*, Feb. 27–Mar. 3, 2006, pp. 254–257.

[8] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Optimal design of digital IIR filters by using hybrid taguchi genetic algorithm," *IEEE Trans. Ind. Electron.*, vol. 53, no. 3, pp. 867–879, Jun. 2006.

[9] Y. Yu and Y. Xinjie, "Cooperative coevolutionary genetic algorithm for digital IIR filter design," *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1311–1318, Jun. 2007.

[10] R. Salomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Trans. Evol. Comput.*, vol. 2, no. 2, pp. 45–55, Jul. 1998.

[11] M. Bundzel and P. Sincak, "Combining gradient and evolutionary approaches to the artificial neural networks training according to principles of support vector machines," in *Proc. IEEE IJCNN*, Jul. 16–21, 2006, pp. 2068–2074.

[12] X. Hu, Z. Huang, and Z. Wang, "Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms," in *Proc. CEC*, Dec. 8–12, 2003, vol. 2, pp. 870–877.

[13] J. Hewlett, B. Wilamowski, and G. Dundar, "Merge of evolutionary computation with gradient based method for optimization problems," in *Proc. IEEE ISIE*, Vigo, Spain, Jun. 4–7, 2007, pp. 3304–3309.

[14] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM J. Appl. Math.*, vol. 11, no. 2, pp. 431–441, Jun. 1963.

[15] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York: Springer-Verlag, 2006, pp. 136–143.

[16] S. Abid, A. Mouelhi, and F. Fnaiech, "Accelerating the multilayer perceptron learning with the Davidon Fletcher Powell algorithm," in *Proc. IJCNN*, 2006, pp. 3389–3394.

[17] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Nat. Comput.*, vol. 1, no. 1, pp. 3–52, 2002.

[18] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.*, vol. 11, no. 1, pp. 1–18, 2003.

Joel D. Hewlett (S'08) received the B.S. degree in electrical engineering from Auburn University, Auburn, AL, where he is currently working toward the M.S. degree in electrical and computer engineering.

He is a Research Assistant with the Department of Electrical and Computer Engineering, Auburn University. He was a Systems Analyst with Delta Research, Inc., Huntsville, AL. His main research interests include intelligent systems, neural networks, numerical optimization, and evolutionary computation.

Mr. Hewlett is a Reviewer for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.

Bogdan M. Wilamowski (M'82–SM'83–F'00) received the M.S. degree in computer engineering in 1966, the Ph.D. degree in neural computing in 1970, and the Dr. Habil. degree in integrated circuit design in 1977.

He received the title of Full Professor from the President of Poland in 1987. He was the Director of the Institute of Electronics (1979–1981) and the Chair of the Solid State Electronics Department, Technical University of Gdansk (1987–1989), Gdansk, Poland. He was a Professor with Gdansk University of Technology (1987–1989), Gdansk; University of Wyoming, Laramie (1989–2000); University of Idaho, Moscow (2000–2003); and Auburn University, Auburn, AL (2003). He was also with the Communication Institute, Tohoku University, Sendai, Japan (1968–1970); Semiconductor Research Institute, Sendai (1975–1976); Auburn University (1981–1982 and 1995–1996); and University of Arizona, Tucson (1982–1984). Currently, he is the Director of the Alabama Micro/Nano Science and Technology Center and a Professor with the Department of Electrical and Computer Engineering, Auburn University. He is the author of four textbooks and about 300 refereed publications and is the holder of 28 patents. He was the Major Professor for over 130 graduate students. His main areas of interest include computational intelligence and soft computing, CAD development, solid-state electronics, mixed signal and analog signal processing, and network programming.

Dr. Wilamowski was the President of the IEEE Industrial Electronics Society (2004–2005). He served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS, IEEE TRANSACTIONS ON EDUCATION, IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, *Journal of Intelligent and Fuzzy Systems*, *Journal of Computing*, *International Journal of Circuit Systems*, and *IES Newsletter*. Currently, he is the Editor-in-Chief of the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.

Günhan Dündar (M'89) was born in Istanbul, Turkey, in 1969. He received the B.S. and M.S. degrees in electrical engineering from Boğaziçi University, Istanbul, Turkey, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1993.

In 1994, he was a Lecturer with Boğaziçi University, teaching courses on electronics, electronics laboratory, IC design, electronic design automation, and semiconductor devices. During August 1994–November 1995, he was with the Turkish Navy and taught courses on electronics, electronics laboratory, and signals and systems at the Turkish Naval Academy, Istanbul. Since 1995, he has been with Boğaziçi University, where he is currently a Professor in, and the Department Chairman of, the Department of Electrical and Electronic Engineering. Between 2002 and 2003, he was with the Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, on sabbatical leave. His research interests include analog integrated circuit design, CAD for analog design, and soft-computing circuits.