Application of Random-Based Step Length Selection Algorithm on Unconstrained Minimization with Steepest Descent Method

Kadir Kiran
Suleyman Demirel University

APPLICATION OF RANDOM-BASED STEP LENGTH SELECTION ALGORITHM ON UNCONSTRAINED MINIMIZATION WITH STEEPEST DESCENT METHOD

Kadir KIRAN[1,2]
[1]Suleyman Demirel University, School of Civil Aviation, Department of Airframe and Power Plant Maintenance, Isparta, Turkey
[2]Suleyman Demirel University, Innovative Technologies Application and Research Center, Design and Manufacturing Technologies Research Laboratory, Isparta, Turkey
kadirkiran@sdu.edu.tr , ORCID: 0000-0002-6109-435X

Abstract
In iterative unconstrained optimization, not only search direction but also step length have a great importance for the success and the performance of the method being used. The computation of the step length is actually a one-dimensional minimization problem and finding one of exact local minimizers of this problem brings high computational cost. To reduce this cost, in this paper, a random-based step length selection algorithm for the steepest descent (SD) method is proposed. The algorithm basically selects the step length, which ensures that the next function value is lower than the current one, from three randomly generated step lengths between $\mathbf{10^{-4}}$ and $\mathbf{10^{0}}$. The performance evaluation of the proposed algorithm is completed on the minimization of eleven two-dimensional test functions that have smooth and nonsmooth properties. For the performance measure, the total number of function evaluations, when converge occurs, are considered. The results have shown that the random-based step length selection algorithm is quite successful and provides much lower number of function evaluations to converge on most of the test functions compared to the exact local minimizer.

Keywords: Steepest descent, Step length, Line search, Optimization

1. INTRODUCTION

The steepest decent method is a practical and an easy to apply unconstrained optimization procedure. It is invented by Cauchy (1847) and the method uses negative gradient of the function in question to be a search direction. To control amount of the movement to the next point along this direction, a step length is employed. Generally speaking, for any optimization method, which uses line search procedure, the computation of the step length is actually a one-dimensional problem. It is pretty costly to compute one of exact local minimizers for this problem, which result in slow converge to the minimum. To overcome this issue and improve the SD method performance, the remarkable efforts have been completed in the literature. Barzilai and Borwein (1988), for instance, presented two-point step sizes for the SD method and they reported that the proposed algorithms provide less computation cost. By combining two methods (i.e., Cauchy and Barzilai-Borwein methods), Raydan and Svaiter (2002) developed the Cauchy-Barzilai-Borwein method for the choice of step length in optimization with the SD method. They obtained a superior performance with this method compared to Barzilai-Borwein and random Cauchy methods. Another attempt to efficiently compute the step length was carried out by Dai (2003). In this attempt, an alternate step length computation algorithm was proposed. According to the algorithm, step length is computed using Cauchy method at odd iterations while the Barzilai-Borwein is employed at even iterations. This switching process considerably improved SD method performance. Besides this study, Dai and Yuan (2003) presented an alternate minimization method that provides the step length minimizing the function value and norm of gradient. On the other hand, a Newton-like exact line search procedure was revealed by Wen et al. (2012) for the SD method. The authors performed a comparison between the proposed procedure and the well-known step length computation methods, and they achieved better performance with this new procedure. Different than the above studies, Moir (2021) considered the SD optimization as a control system. By doing so, the author used adaptive step size to reduce error in the control system. Another interesting study was completed by Kalousek (2017). In the study, the author proposes to use randomly generated step lengths in the SD method. Inspiring from this study, a random-based step length selection algorithm is developed in the current paper. Simply, the proposed algorithm selects step length, which provides lower function value than the current one, from three randomly generated steps lengths. From here on out, this paper is organized as follows: Section 2 covers the SD method. Section 3 describes the proposed algorithm. Section 4 includes test functions. In section 5, results and discussion are provided. Finally, Section 6 summaries and concludes the paper.

2. STEEPEST DESCENT METHOD

The steepest descent method, which was developed by Cauchy (1847), might be considered as a fundamental optimization method. It searches the point that gives function minimum value along with the opposite direction of the function gradient. This statement is mathematically:

$$x_{q+1} = x_q - \alpha \nabla f(x_q) \tag{1}$$

where x_q, x_{q+1}, $\alpha > 0$ and $\nabla f(x_q)$ are the current and next points, step length at the q. iteration and the function gradient at the current point, respectively. Generally speaking, Eq. (1) is also referred to as a line search because the next point is estimated on a line created by $\alpha \nabla f(x_q)$. This line search continues until the converge term (Griva et al. 2009) is satisfies which, in this study, is:

$$\frac{||\nabla f(x_{q+1})||_2}{1 + |f(x_{q+1})|} \leq 10^{-4} \tag{2}$$

On the other hand, the step length α is has a critical importance for success and performance of the SD method. Its computation is actually a one-dimensional minimization problem as follows:

$$\text{minimize}_{\alpha>0}\, g(\alpha) \equiv f(x_q - \alpha \nabla f(x_q)) \tag{3}$$

In this study, one of the exact local minimizers of the $g(\alpha)$ is numerically found using function gradient. Besides, a random-based algorithm, which is described in next section, is proposed to reduce computational expense. It is also noteworthy that the function gradient is calculated with the finite difference method.

3. PROPOSED ALGORITHM

The proposed algorithm bases on the random generation of the step lengths for each iteration. To do that, first, three random step lengths between $\alpha_{min} = 10^{-4}$ and $\alpha_{max} = 10^{0}$ are generated with a uniform distribution. Second, the function values (i.e., $f(x_q - \alpha_r \nabla f(x_q))$) for these step lengths are computed. In third, the step length (i.e., α_s) that gives minimum function value among them is determined and the minimum function value is compared with the current function value (i.e., $f(x_q)$) whether it is less than $f(x_q)$ or not. If so, the step length (i.e., α) for the line search is then set as α_s. If not so, new random step lengths are created and the same procedure is repeated until reaching defined number of iterations (i.e., N). As a safeguard of this algorithm, we set the α equal to the α_{min} if the algorithm can not find any random step length satisfying required condition within the defined N. As a convenience to reader, a pseudocode for this procedure is given in Algorithm 1.

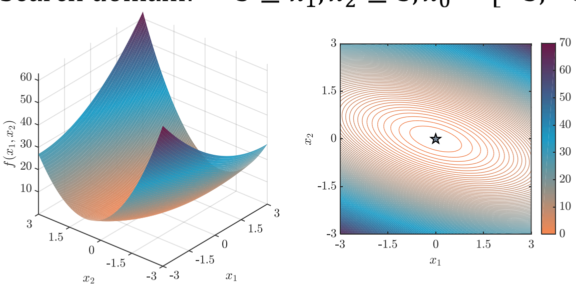
Algorithm 1 Random-based step length selection

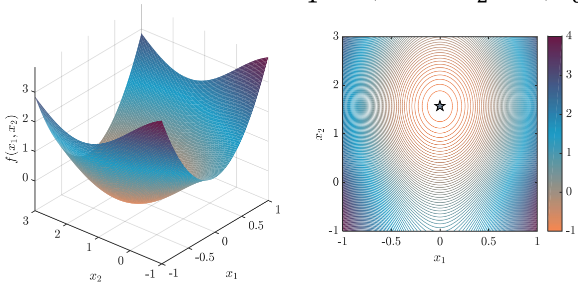
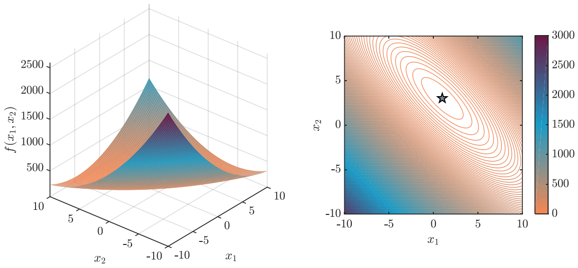
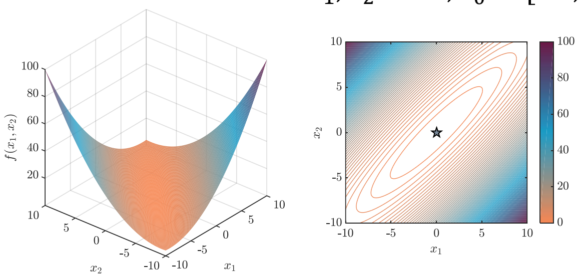
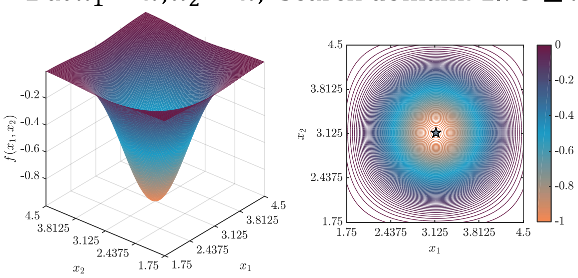
function RandomStepLength$(x_q, f, \nabla f(x_q))$
    define minimum step length $\alpha_{min} = 10^{-4}$
    define maximum step length $\alpha_{max} = 10^{0}$
    define number of iterations $N = 100$
    for $q \leftarrow 1$ to N **do**
        calculate 3 random step lengths: $\alpha_r = (\alpha_{max} - \alpha_{min}) * rand(1,3) + \alpha_{min}$
        find the step length giving smallest function value: $\alpha_s = argmin\, f(x_q - \alpha_r \nabla f(x_q))$
        if $f(x_q - \alpha_s \nabla f(x_q)) \leq f(x_q)$ **then**
            $\alpha = \alpha_s$
            break
        end if
        check number of iterations:
        if $q \geq N$ **then**
            $\alpha = \alpha_{min}$
            break
        end if
    end for
end function

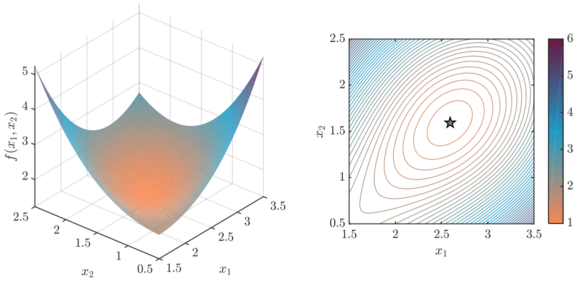
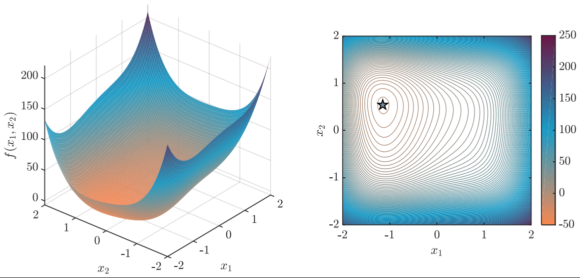
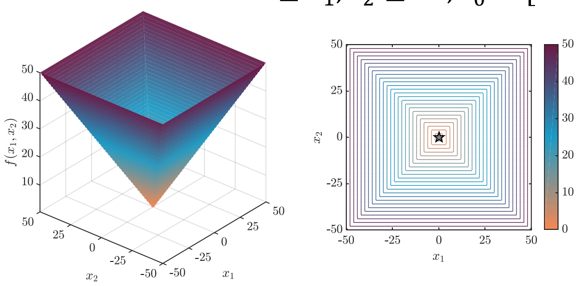
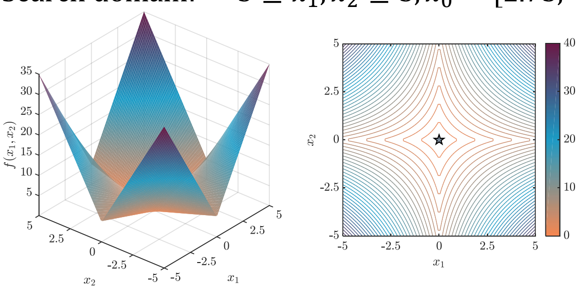
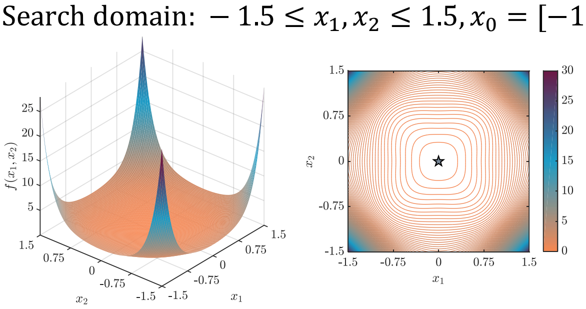
4. TEST FUNCTIONS

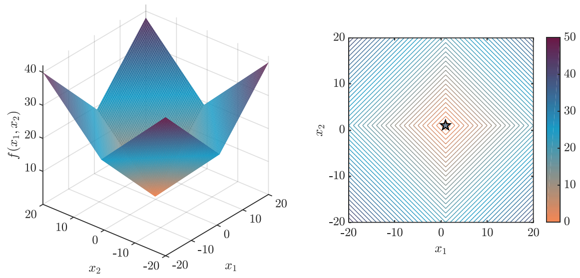
For performance evaluation of the proposed algorithm, the eleven two-dimensional test functions (Andrei, 2008; Jamil and Yang, 2013; Raeesi et al., 2020; Salonga et al., 2019), which are smooth and non-smooth, are used. Table 1. shows their mathematical descriptions, coordinates of the minimum points and corresponding function values, search domains, initial points for starting the search and plots.

Table 1. Test functions

Function Number	Function
1	$f(x_1, x_2) = x_1^2 + 4x_2^2 + 2x_1x_2, f_{min} = 0$ at $x_1 = 0, x_2 = 0,$ Search domain: $-3 \leq x_1, x_2 \leq 3, x_0 = [-3, -3]^T$ 

2	$f(x_1,x_2) = 3x_1^2 - \sin(x_2)$, $f_{min} = -1$ at $x_1 = 0, x_2 = 1.5$, Search domain: $-1 \le x_1 \le 1, -1 \le x_2 \le 3, x_0 = [0.9, -0.2]^T$ 
3	$f(x_1,x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$(Booth), $f_{min} = 0$ at $x_1 = 1, x_2 = 3$, Search domain: $-10 \le x_1, x_2 \le 10, x_0 = [-9.5, 9.5]^T$ 
4	$f(x_1,x_2) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ (Matyas), $f_{min} = 0$ at $x_1 = 0, x_2 = 0$, Search domain: $-10 \le x_1, x_2 \le 10, x_0 = [-8, -9.5]^T$ 
5	$f(x_1,x_2) = -\cos(x_1)\cos(x_2)\exp\left(-((x_1-\pi)^2 + (x_2-\pi)^2)\right)$ (Easom), $f_{min} = -1$ at $x_1 = \pi, x_2 = \pi$, Search domain: $1.75 \le x_1, x_2 \le 4.5, x_0 = [1.9, 4.35]^T$ 
6	$f(x_1,x_2) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$ (Mc – Cormick), $f_{min} = -1.9133$ at $x_1 = -0.54719, x_2 = -1.54719$, Search domain: $1.5 \le x_1 \le 3.5, 0.5 \le x_2 \le 2.5, x_0 = [1.6, 2.4]^T$

	
7	$f(x_1,x_2) = 5x_1^4 + 6x_2^4 - 6x_1^2 + 2x_1x_2 + 5x_2^2 + 15x_1 - 7x_2 + 13, f_{min} = -6.496$ at $x_1 = 0, x_2 = 0$, Search domain: $-2 \le x_1, x_2 \le 2, x_0 = [1.9, -1.9]^T$ 
8	$f(x_1,x_2) = \max(\lvert x_1\rvert, \lvert x_2\rvert), f_{min} = 0$ at $x_1 = 0, x_2 = 0$, Search domain: $-50 \le x_1, x_2 \le 50, x_0 = [-48, 43]^T$ 
9	$f(x_1,x_2) = \lvert x_1\rvert + \lvert x_2\rvert + \lvert x_1\rvert . \lvert x_2\rvert, f_{min} = 0$ at $x_1 = 0, x_2 = 0$, Search domain: $-5 \le x_1, x_2 \le 5, x_0 = [2.75, -5]^T$ 
10	$f(x_1,x_2) = (x_1^2)^{(x_2^2+1)} + (x_2^2)^{(x_1^2+1)}, f_{min} = 0$ at $x_1 = 0, x_2 = 0$, Search domain: $-1.5 \le x_1, x_2 \le 1.5, x_0 = [-1.5, 1.25]^T$ 
11	$f(x_1,x_2) = \lvert x_1 - 1\rvert + \lvert x_2 - 1\rvert, f_{min} = 0$ at $x_1 = 0, x_2 = 0$, Search domain: $-20 \le x_1, x_2 \le 20, x_0 = [-8, 20]^T$

5. RESULTS AND DISCUSSION

The minimization process is completed using the SD method along with random-based step length selection algorithm for all the test functions. To be able to provide a comparison between the proposed algorithm and the exact local minimizer, the minimization process is also completed using exact local minimizer. During these processes, the point, function and function gradient values are kept track at every iteration. As an example, for function 6, these outcomes versus iteration are illustrated in Figures 1. In addition, for a performance evaluation, the number of function evaluations at each iteration are computed and the algorithm outcomes are also plotted corresponding to the number of function evaluations (Nfe) at each iteration, as shown in Figure 2.

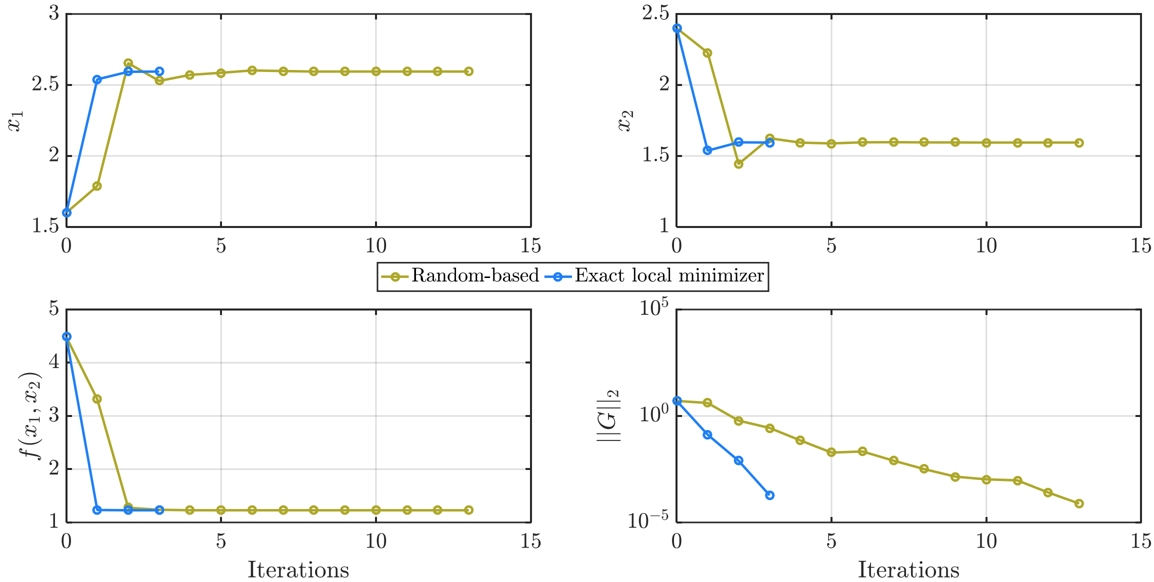


Figure 1. Minimization progress for function 6: Results versus iterations

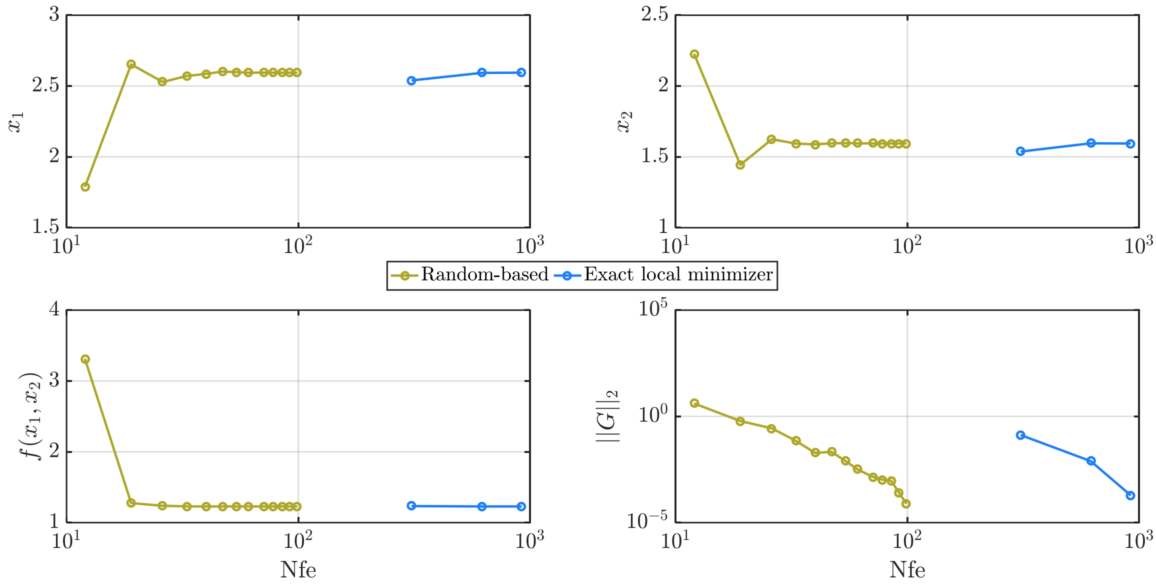


Figure 2. Minimization progress for function 6: Results versus number of function evaluations

As can be observed from Figure 1, SD method with both step length computation techniques successfully found the function minimum point with the defined converge term. However, the SD method with the exact local minimizer completes the search with 3 iterations whereas the random-based algorithm requires 13 iterations. At first glance, this might seem superiority of the exact local minimizer, but the amount of work to compute the corresponding step length at each iteration are quite high compared to random-based algorithm. This fact can be clearly seen from Figure 2. In other words, the random-based step length selection algorithm requires higher number of iterations to converge, but lower number of function evaluations compared to exact local minimizer. More specifically, for the function 6, the random-based algorithm totally evaluates the function 77 times while this is 921 for the exact local minimizer. The high number of function evaluations in the exact local minimizer are due to need of function gradient calculations for numerically finding one of the exact local minimizers of the $g(\alpha)$. However, for the random-based algorithm, it is not necessary to calculate function gradient (see Algorithm 1). Therefore, this new algorithm outperforms the exact local minimizer. As a convenience to reader, the step lengths computed via both methods are also compared in Figure 3(a), and Figure 3(b) shows the corresponding number of function evaluations at each iteration.

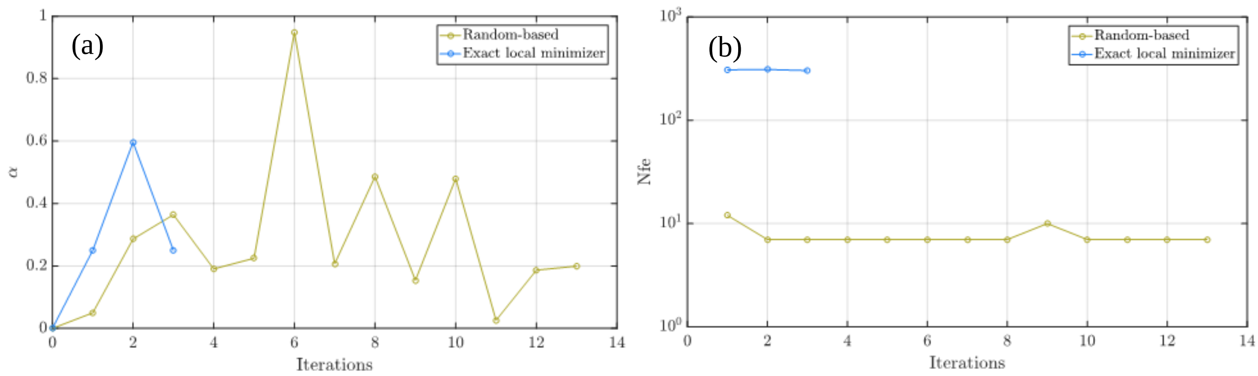


Figure 3. Comparison of calculated step lengths for each iteration

The reader may also note that the random-based algorithm is run 100 times and the mean values of number of iterations and of function evaluations are reported here because, as expected, the generated random step lengths vary at each iteration, which in turn variation in number of iteration (see Figure 4(a)) and function evaluations (see Figure 4(b)). This fact also leads to different search paths for every run, as shown in Figure 5.

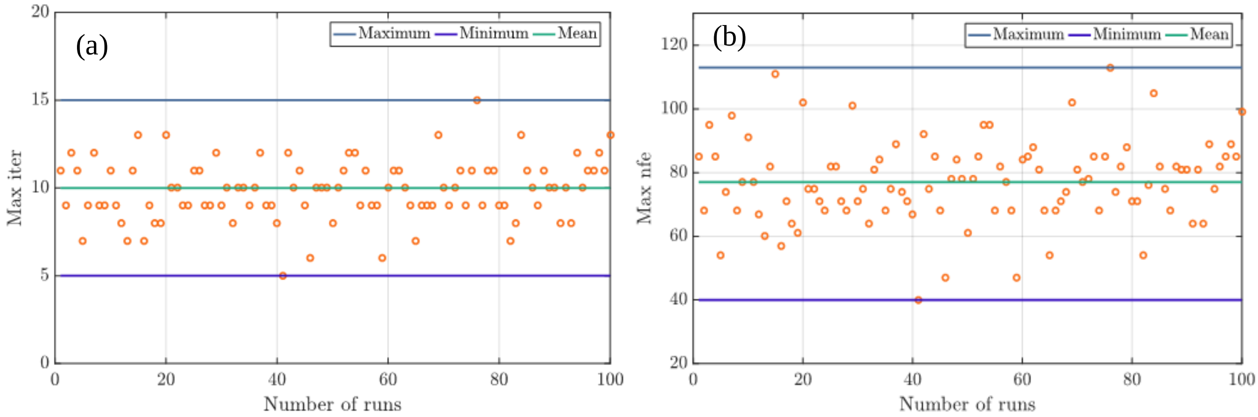


Figure 4. Variation of (a) number of iterations and (b) function evaluations in random-based algorithm

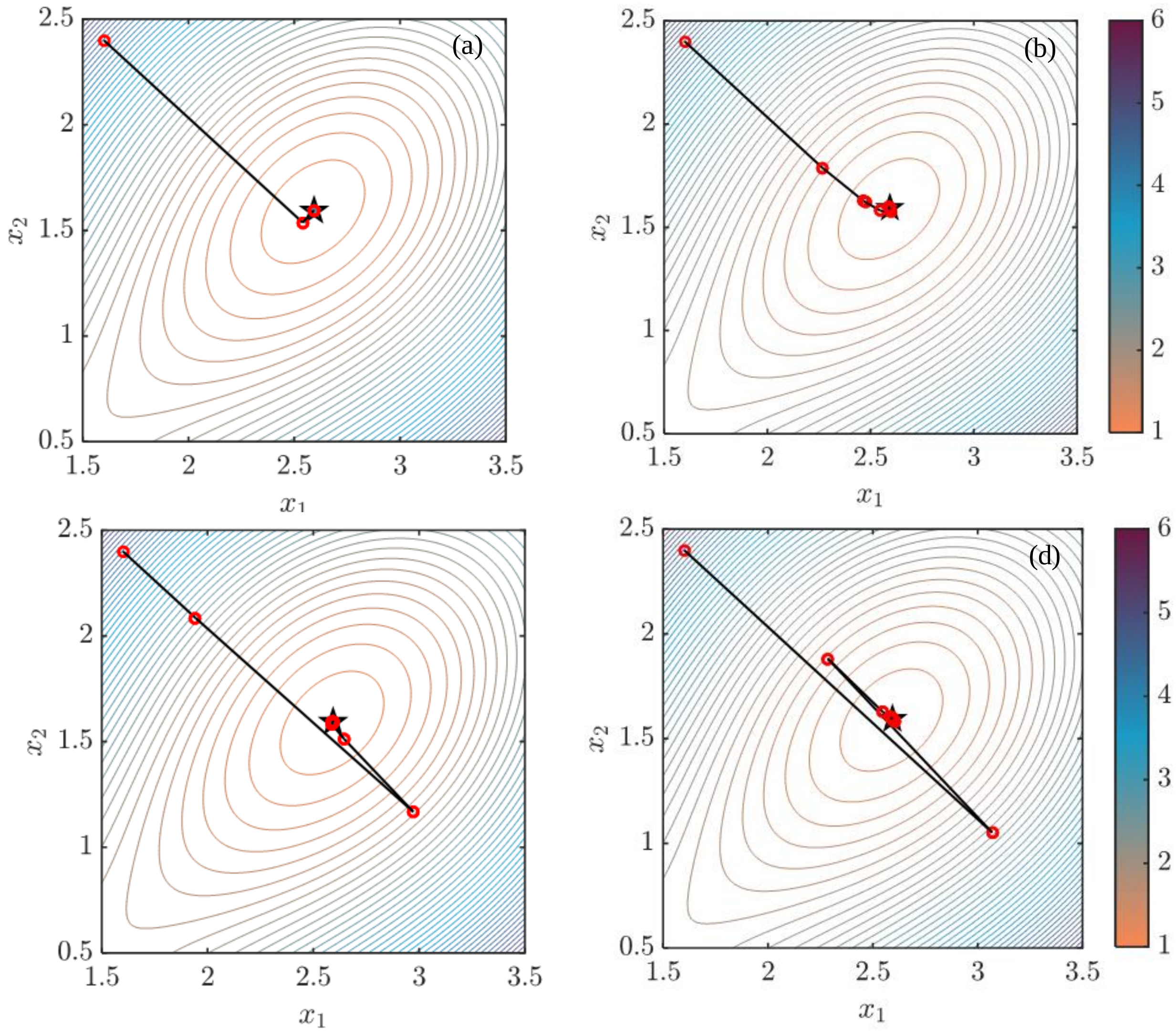


Figure 5. Search paths (Black lines): (a) Exact local minimizer (b) Random-based algorithm run-1 (c) Random-based algorithm run-2 (d) Random-based algorithm run-3

The same procedure reported so far is completed for all the test functions and the maximum number of iterations and the total number of function evaluations for all the test functions are comparatively indicated in Figures 6 and 7, respectively. As can be seen from Figure 6, using the exact local minimizer provides lower number of iterations in all the test functions. However, by looking at Figure 7, it requires higher number of function evaluations in 9 test functions out of 11 compared to random-based algorithm. As mentioned before, the function gradient computation, which is mandatory, brings high cost on the exact local minimizer. For the proposed random-based algorithm, we could state that it is a function gradient-free algorithm. Therefore, its performance is much better than the exact local minimizer.

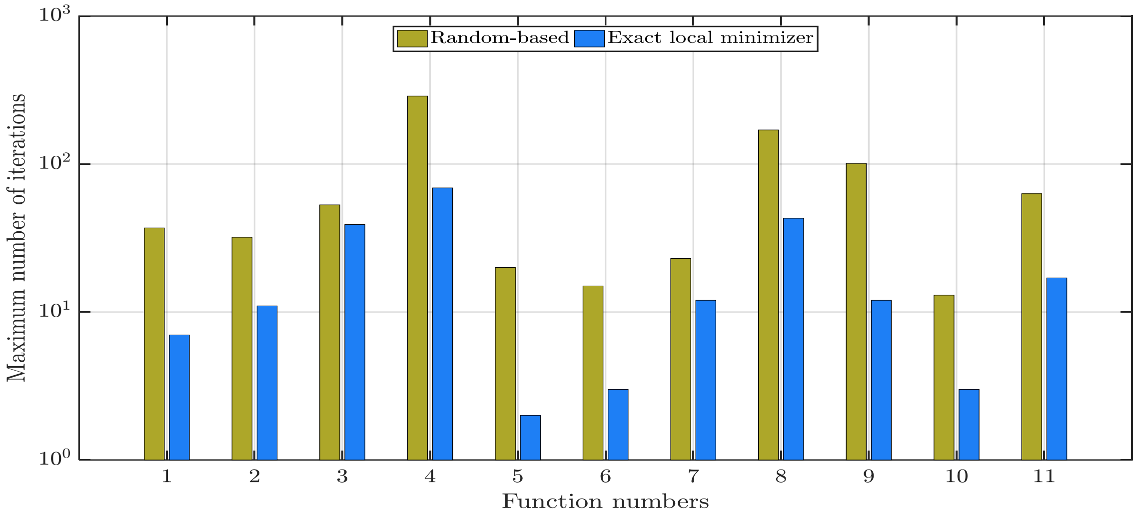


Figure 6. Maximum number of iterations for all functions

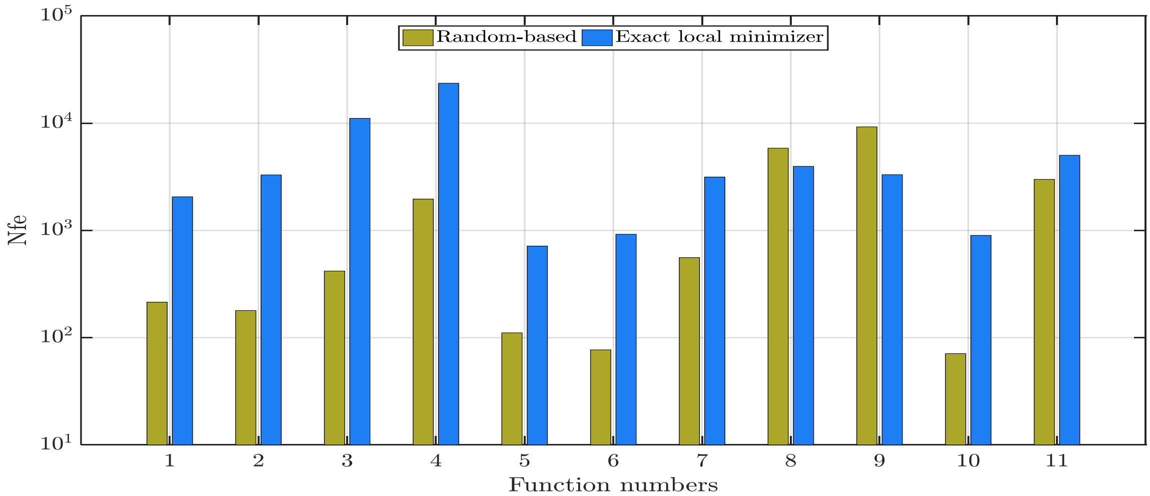


Figure 7. Total number of function evaluations for all functions

6. CONCLUSION

This paper was presented a random-based algorithm to select the step length for the minimization of functions using SD method. The eleven two-dimensional test functions were used to evaluate performance of the proposed algorithm. For the performance evaluations, the total number of function evaluations, when the converge term is satisfied, were considered. The results have shown that the proposed random-based step length selection algorithm exhibits much better performance in the most of test functions compared to exact local minimizer due to its function gradient computation-free property. These primary results show us that the proposed algorithm has a great potential for optimization algorithms. Thus, for a future study, we evaluate its performance in different applications such as curve fitting, geometry fitting etc.

REFERENCES

Andrei, N., 2008. An unconstrained optimization test functions collection. Adv. Model. Optim, 10(1), 147-161.

Barzilai, J. and Borwein, J.M., 1988. Two-point step size gradient methods. IMA Journal of Numerical Analysis, 8(1), 141-148.

Cauchy A., 1847. Methode generale pour la resolution des systemes d'equations simultanees. Comp. Rend. Sci. Paris, 25(2), 536-538.

Dai, Y.H., 2003. Alternate step gradient method. Optimization, 52(4-5), 395-415.
Dai, Y.H. and Yuan, Y.X., 2003. Alternate minimization gradient method. IMA Journal of Numerical Analysis, 23(3), 377-393.
Griva, I., Nash, S.G. and Sofer, A., 2009. Linear and nonlinear optimization. Vol. 108, Siam.
Jamil, M. and Yang, X.S., 2013. A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, 4(2), 150-194.
Kalousek, Z., 2017. Steepest descent method with random step lengths. Foundations of Computational Mathematics, 17(2), 359-422.
Moir, T.J., 2021. An adaptive stepsize plus integral controller for improvement to the steepest descent algorithm. International Journal of Adaptive Control and Signal Processing, 35(11), 2194-2213.
Raydan, M. and Svaiter, B.F., 2002. Relaxed steepest descent and Cauchy-Barzilai-Borwein method. Computational Optimization and Applications, 21(2), 155-167.
Raeesi, F., Azar, B.F., Veladi, H. and Talatahari, S., 2020. An inverse TSK model of MR damper for vibration control of nonlinear structures using an improved grasshopper optimization algorithm. Structures, 26, 406-416.
Salonga, P.K., Inaudito, J.M. and Mendoza, R., 2019. An unconstrained minimization technique using successive implementations of Golden Search algorithm. AIP Conference Proceedings, 2192(1), 060018.
Wen, G.K., Mamat, M., Mohd, I.B. and Dasril, Y., 2012. A novel of step size selection procedures for steepest descent method. Applied Mathematical Sciences, 6(51), 2507-2518.