A Benchmark Study on Steepest Descent and Conjugate Gradient Methods-Line Search Conditions Combinations in Unconstrained Optimization

Kadir Kiran[1,2,*]

[1] *Department of Airframe and Powerplant Maintenance, School of Civil Aviation, Suleyman Demirel University, Isparta, Turkey*

[2] *Design and Manufacturing Laboratory, Innovative Technologies Application and Research Center, Suleyman Demirel University, Isparta, Turkey*
E-mail: ⟨kadirkiran@sdu.edu.tr⟩

Abstract. In this paper, it is aimed to computationally conduct a performance benchmarking for the steepest descent and the three well-known conjugate gradient methods (i.e., Fletcher-Reeves, Polak-Ribiere and Hestenes-Stiefel) along with six different step length calculation techniques/conditions, namely Backtracking, Armijo-Backtracking, Goldstein, weak Wolfe, strong Wolfe, Exact local minimizer in the unconstrained optimization. To this end, a series of computational experiments on a test function set is completed using the combinations of those optimization methods and line search conditions. During these experiments, the number of function evaluations for every iteration are monitored and recorded for all the optimization method-line search condition combinations. The total number of function evaluations are then set a performance measure when the combination in question converges to the functions minimums within the given convergence tolerance. Through those data, the performance and data profiles are created for all the optimization method-line search condition combinations with the purpose of a reliable and an efficient benchmarking. It has been determined that, for this test function set, the steepest descent-Goldstein combination is the fastest one whereas the steepest descent-exact local minimizer is the most robust one with a high convergence accuracy. By making a trade-off between convergence speed and robustness, it has been identified that the steepest descent-weak Wolfe combination is the optimal choice for this test function set.

Keywords: conjugate gradient, line search, step length, steepest descent, optimization

Received: February 28, 2022; accepted: May 17, 2022; available online: July 12, 2022

DOI: 10.17535/crorr.2022.0006

1. Introduction

Steepest descent (SD), developed by Cauchy [8], is a fundamental optimization technique. Over the years, it is used in many studies such as those in [20, 28, 7, 33, 45, 37]. On the other hand, conjugate gradient (CG) method was first presented by Hestenes and Stiefel [21] to numerically solve the linear equations. Fletcher and Reeves [17] later improved it for nonlinear optimization. Since then, the CG methods and their variants [10, 15, 4, 16] have become the focus of many researchers. The details about the nonlinear CG methods in more depth can be found in the survey performed by Hager and Zhang [19]. All those methods or other optimization methods, which are available in the literature, might exhibit a different behavior and performance from problem-to-problem. Therefore, the benchmarking studies on the optimization methods have

*Corresponding author.

gained a great attention in recent years, because they provide the end-users an insight about the optimization methods and help them to select an optimal method for their problems. These studies can be seen in many areas such as computer science [12], engineering [35, 3, 11, 24], biology [43, 41], etc. In addition to those, Khan and Lobiyal [23] conducted a performance evaluation study on the SD, CG and Newton–Raphson methods in optimization of backbone based wireless networks. They recommended that the SD method should be used for accuracy whereas the CG method may be chosen for a higher solution speed. In [39], the authors performed a benchmarking of five global optimization methods (i.e., particle swarm, differential evolution, Bayesian, downhill simplex and limited-memory Broyden-Fletcher-Goldfarb-Shanno) on the nano-optical shape optimization and the parameter reconstruction. They reported that the Bayesian optimization method outperforms the other ones. Truong and Nguyen [42] have recently proposed new gradient descent algorithms for the large scale optimization. The authors showed that the performances of these new algorithms are better than the well-known ones (e.g., Momentum, Nesterov accelerated gradient, Adagrad, etc.). In [44], the well-known global optimization methods such as genetic algorithm, differential evolution and particle swarm methods are compared each other for switched reluctance machine design. The study showed that the differential evolution method stands out with better performance. On the other hand, a great performance benchmarking tool, namely performance profiles, was developed by Dolan and Moré [13]. This tool provides the end-users a valuable insight about the optimization methods being used on the problems. Therefore, many researchers have implemented this approach in their studies to be able to perform a reliable and an efficient benchmarking on the optimization methods. Andrei [2], for instance, employed the performance profiles to benchmark newly developed conjugate gradient method with the well-known ones and the limited memory quasi-Newton L-BFGS algorithm. Another study implementing performance profiles was carried out by Dolan et. al [14] in order to explore the effect of optimality measures on a set of solvers performances. In addition to the performance profiles, Moré and Wild [29] proposed the data profiles to perform an independent assessment on the optimation methods or the solvers. In other words, by means of these profiles, we may define the optimal optimization method within the given computational budget, which is not possible using the performance profiles. The performance and data profiles were successfully employed together in some studies such as those in [32, 27]. Besides, they have a great potential to be implemented in other areas including manufacturing, machining dynamics models [26, 25, 38], etc. Further information on benchmarking of optimization algorithms can be found in [6, 5] as well.

In the optimization using the SD and CG methods, the line search techniques [18, 30, 40, 9] are generally employed for a progress along with the given direction and the step length. In the line search methods, the direction and the step length play a crucial role for the performance of the optimization method being used. However, it is not found any comprehensive study on the performances of the SD and CG methods-line search conditions combinations in the literature. To make a contribution to this gap, therefore, in this paper, we focus on the effect of step length computation techniques or line search conditions such as Backtracking (BC), Armijo-Backtracking (ABC), Goldstein(GC), weak Wolfe(WWC), strong Wolfe (SWC) and local minimizer (LM) on the performances of the SD and three well-known CG methods (i.e., Fletcher-Reeves (FR), Polak-Ribiere (PR) and Hestenes-Stiefel (HS)). For benchmarking purpose, the optimization methods and line search conditions are considered as the combinations. More clearly, totally 24 combinations, which consist of 4 optimization methods and 6 line search conditions, are put to test on the 13 two-dimensional test functions. For each combination, the total number of function evaluations required to converge to the test functions minimums are recorded and set as a performance measure. Using those data, the performance and data profiles are created to benchmark the combinations. By means of these profiles, we are able to answer the following example questions: I) *What are the fastest, the most robust and the optimal optimization method-line search condition combinations for those test functions?*, II) *What* *is the percentage of the functions that a particular optimization method-line search condition combination is successful on?*, III) *What is the probability of finding functions minimums if we choose one of these combinations?* and IV) *How many functions can a particular optimization method-line search condition combination minimize within the given computational budget?*

From here on in, the remainder of this paper is structured as follows. Section 2 provides a summary on the mathematical background of the methods used in the study. Section 3 covers the test functions used in the computational experiments and the benchmarking procedures. Section 4 presents the benchmarking results with discussion. Finally, in Section 5, conclusions on the study are performed.

2. Mathematical Preliminaries

This section is intended to give a summary on the mathematical concepts of the SD and CG methods, and the line search conditions used in the study.

2.1. Steepest Descent Method

The steepest descent method, which was presented by Cauchy [8], is one of best known optimization methods in the literature. It performs a line search along the descent direction. In this method, the minimization of a function $f(x)$, which depends on real values (i.e., $x \in \mathbb{R}^n$ with $n \geq 1$) [30] and might be unimodal, multimodal, linear, nonlinear, continuous, discontinuous, convex, non-convex [22], can be completed as follows:

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

where the x_q and x_{q+1} are the current and next points, respectively. The $\nabla f(x_q)$ is the gradient of the function at the current point and the $\gamma_q > 0$ is the step length at the q. iteration. In (1), notice that the search direction in the SD method is the opposite of the function gradient. It is also noteworthy that $\gamma_q > 0$ is a scalar and it defines the amount of movement along the descent direction in the line search . Hence, the γ_q plays a major role on the performance of the optimization methods being applied as well as the search direction. Computationally exploring its role in the minimization with the SD and CG methods is the main focus of this paper, which is elaborated in Section 4.

2.2. Conjugate Gradient Methods

Hestenes and Stiefel [21] developed the linear CG method to iteratively solve the linear equations. Later on, the method was improved by Fletcher and Reeves [17] for solving nonlinear optimization problems. Polak and Ribiere [31] presented another version of the Fletcher and Reeves method as well. For the mathematical derivation of the CG methods, the interested reader is referred to Refs. [21, 17, 31, 18, 30]. Only a summary for those methods are provided in this section.

The next point in the CG methods is computed as follows:

$$x_{q+1} = x_q + \gamma_q s_q \tag{2}$$

where γ_q is the step length and s_q is the search direction. s_0 at the starting point x_0 is the same with the SD method which means that it is the negative direction of the function gradient at the current point. This is mathematically written in (3).

$$s_0 = -\nabla f(x_0) \tag{3}$$

The successive search directions are then computed as:

$$s_{q+1} = -\nabla f(x_{q+1}) + \beta_{q+1} s_q \tag{4}$$

where β_{q+1} is the CG coefficient. There are many formulas to calculate this coefficient in the literature. In this study, we use three best known formulas that are shown below.

Fletcher-Reeves:

$$\beta_{FR-q+1} = \frac{\nabla f(x_{q+1})^T \nabla f(x_{q+1})}{\nabla f(x_q)^T \nabla f(x_q)} \tag{5}$$

Polak-Ribiere:

$$\beta_{PR-q+1} = \frac{\nabla f(x_{q+1})^T (\nabla f(x_{q+1}) - \nabla f(x_q))}{\nabla f(x_q)^T \nabla f(x_q)} \tag{6}$$

Hestenes-Stiefel:

$$\beta_{HS-q+1} = \frac{\nabla f(x_{q+1})^T (\nabla f(x_{q+1}) - \nabla f(x_q))}{s_q^T (\nabla f(x_{q+1}) - \nabla f(x_q))} \tag{7}$$

In these equations, the $\nabla f(x_q)$ the $\nabla f(x_{q+1})$ are the function gradients at the previous and current points, respectively.

2.3. Line Search Conditions

This section covers the well-known line search conditions that are frequently used to compute the step length in a line search. For the sake of better understanding how the step length is computed, we first define the line search as follows:

$$x_{q+1} = x_q + \gamma_q s_q \tag{8}$$

As previously mentioned, in the line search, the next point is achieved along the given direction s_q. The amount of movement along this direction is defined by the step length γ_q. Its computation is actually one dimensional minimization problem, as described in (9).

$$\underset{\gamma>0}{\text{minimize}}\; g(\gamma) \equiv f(x_q + \gamma s_q) \tag{9}$$

The function g is a univariate function [30], and for a significant progress in the line search, one of the local minimizers or global minimizer of the g are desired to be found (see Figure 1). However, this requires an exact solution for (9) with a very high computational cost.

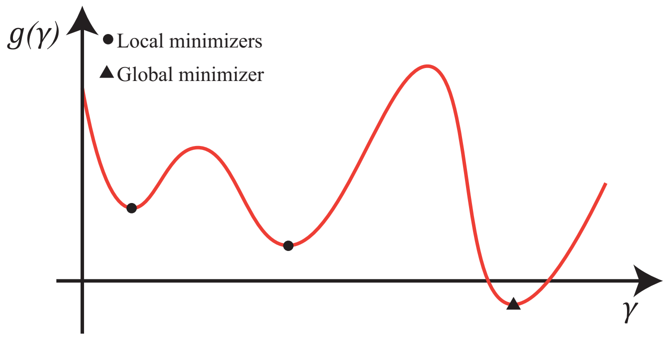


Figure 1: *Schematic description of step length computation problem*

On the other hand, inexact methods or line search conditions for computing step length provide reasonable progress in the line search with a minimal computational cost. In the following sections, five line search conditions as the inexact methods and an exact method, which numerically finds one of the local minimizers of the g, are given.

2.3.1. Backtracking

The backtracking condition focuses on only a reduction in the function value. The initial step length (e.g., $\gamma_0 = 1$) is decreased by multiplying with a coefficient ζ (e.g., $\zeta = 0.5$) until the condition below is met.

$$f(x_q + \gamma_q s_q) \leq f(x_q) \tag{10}$$

2.3.2. Armijo-Backtracking

In some cases, the BC may not provide sufficient decrease in the function value. This might lead to poor performance in the optimization method, even it could result in failure to converge [30]. To avoid this fact, Armijo-Backtracking condition, which is defined in (11), is available in the literature.

$$f(x_q + \gamma_q s_q) \leq f(x_q) + \mu\gamma_q \nabla f(x_q)^T s_q \tag{11}$$

In this inequality, $\mu \in (0, 1)$ is some scalar. Notice that the sufficient decrease in the function value is ensured by including the directional derivative $\nabla f(x_q)^T s_q$ [30].

2.3.3. Goldstein Conditions

The Goldstein conditions consist of two inequalities [30] as follows:

$$f(x_q) + (1-\nu)\gamma_q \nabla f(x_q)^T s_q \leq f(x_q + \gamma_q s_q) \leq f(x_q) + \nu\gamma_q \nabla f(x_q)^T s_q \tag{12}$$

where $\nu \in (0, 1/2)$ is some scalar. It is noteworthy that the ABC can be seen in the right inequality. The GC with this form provides a sufficient decrease in the function value as well.

2.3.4. Wolfe Conditions

The Wolfe conditions are other alternative ways to efficiently compute the step length. They can be achieved by adding the curvature condition to the sufficient decrease condition (i.e., ABC). This is mathematically described with two inequalities as follows:

$$f(x_q + \gamma_q s_q) \leq f(x_q) + \mu\gamma_q \nabla f(x_q)^T s_q \tag{13a}$$

$$\nabla f(x_q + \gamma_q s_q)^T s_q \geq \eta \nabla f(x_q)^T s_q \tag{13b}$$

(13a) provides a sufficient decrease, as been in the ABC, whereas (13b) denotes the curvature condition. In (13b), η is a scalar satisfying $0 < \mu < \eta < 1$. With the combination of these two conditions, the step length γ_q can be computed to be close to one of the local minimizers of the $g(\gamma)$ (i.e.,(9)). (13a) and (13b) are referred to as *weak Wolfe conditions*. To approach the one of the local minimizers of the $g(\gamma)$ as much as possible, the curvature condition can be modified to be:

$$|\nabla f(x_q + \gamma_q s_q)^T s_q| \leq \eta |\nabla f(x_q)^T s_q| \tag{14}$$

The combination of the inequalities (i.e., (13a) and (14)) is referred to as *strong Wolfe conditions* [30].

2.3.5. Local Minimizer

In this study, the exact local minimizer of the $g(\gamma)$ is numerically computed using a Golden section method based procedure. The procedure has the following steps.

(i) use the ABC so that γ_q provides sufficient decrease in the function value

(ii) ensure that slope of the $g(\gamma_q)$ is negative, if it is positive, reduce the γ_q until the slope is negative and set it as γ_q^{low}

(iii) increase the γ_q to find the γ_q^{up} where the slope of $g(\gamma)$ starts being positive

(iv) compute the local minimizer of the $g(\gamma)$ by applying the Golden section method between the γ_q^{low} and γ_q^{up}

3. Test Functions and Benchmarking Procedures

Table 1 shows the details about the test functions used for the benchmarking purpose. All the test functions are two-dimensional with having smooth and non-smooth properties. Note that those thirteen test functions are selected from the Refs. [34, 36, 1, 22, 18]

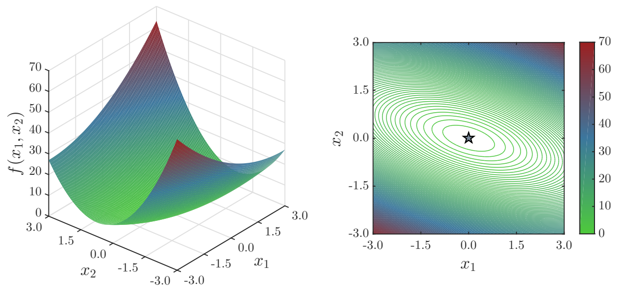
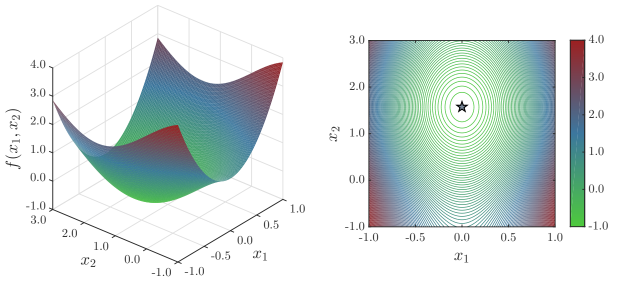
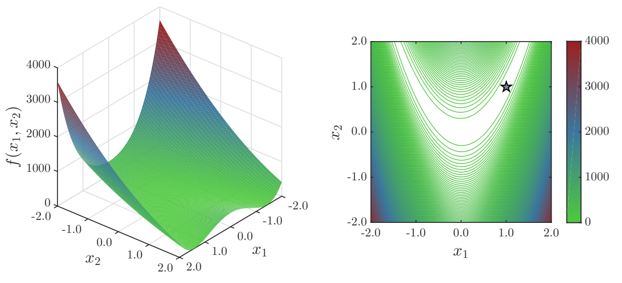
Function number	Function	Function plot
1	$f(x_1,x_2) = x_1^2 + 4x_2^2 + 2x_1x_2$ $f_{min} = 0$ at $x_1 = 0$, $x_2 = 0$ Search domain: -3 $\leq x_1, x_2 \leq$ 3 $x_0 = [-3,-3]^T$	
2	$f(x_1,x_2) = 3x_1^2 - \sin(x_2)$ f_{min} = -1 at $x_1 = 0$, $x_2 = \pi/2$ Search domain: -1 $\leq x_1 \leq$ 1, -1 $\leq x_2 \leq$ 3 $x_0 = [0.9,-0.2]^T$	
3	$f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$ (Rosenbrock) $f_{min} = 0$ at $x_1 = 1$, $x_2 = 1$ Search domain: -2 $\leq x_1, x_2 \leq$ 2 $x_0 = [-1.8,-1.8]^T$	

Table 1: *Test functions*

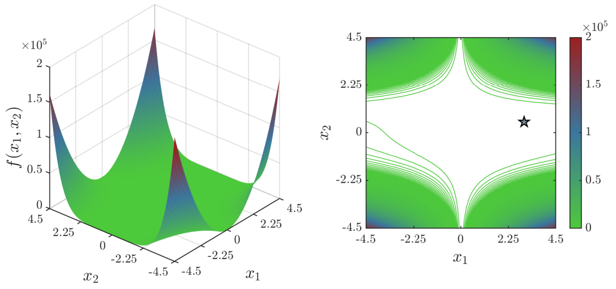
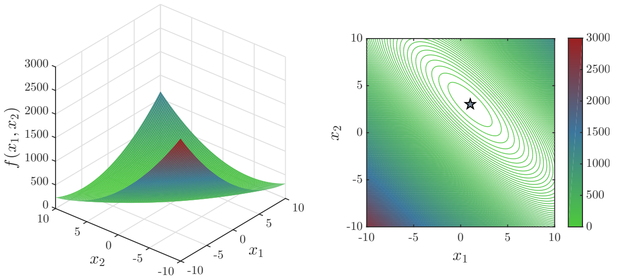
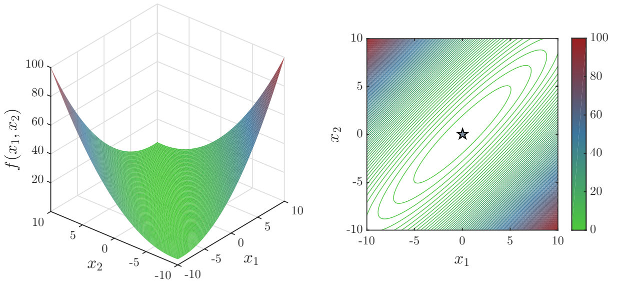
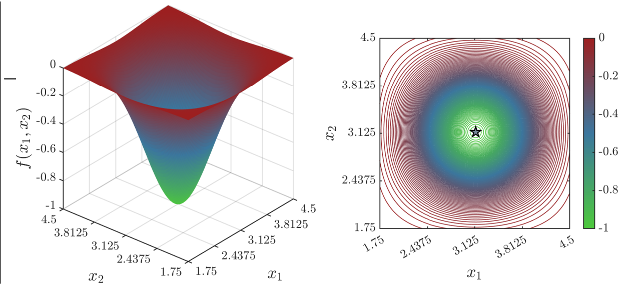
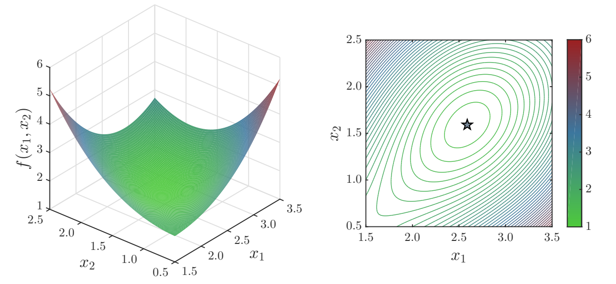
Function number	Function	Function plot
4	$f(x_1,x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$ (Beala) $f_{min} = 0$ at $x_1 = 3$, $x_2 = 0.5$ Search domain: -4.5 $\leq x_1, x_2 \leq$ 4.5 $x_0 = [-1.5, -3.75]^T$	
5	$f(x_1,x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ (Booth) $f_{min} = 0$ at $x_1 = 1$, $x_2 = 3$ Search domain: -10 $\leq x_1, x_2 \leq$ 10 $x_0 = [-9.5, 9.5]^T$	
6	$f(x_1,x_2) = 0.26(x_1^2+x_2^2) - 0.48x_1x_2$ (Matyas) $f_{min} = 0$ at $x_1 = 0$, $x_2 = 0$ Search domain: -10 $\leq x_1, x_2 \leq$ 10 $x_0 = [-8, -9.5]^T$	
7	$f(x_1,x_2) = -\cos(x_1)\cos(x_2)\exp(-((x_1-\pi)^2 + (x_2-\pi)^2))$ (Easom) f_{min} = -1 at $x_1 = \pi$, $x_2 = \pi$ Search domain: 1.75 $\leq x_1, x_2 \leq$ 4.5 $x_0 = [1.9, 4.35]^T$	
8	$f(x_1,x_2) = \sin(x_1+x_2) + (x_1-x_2)^2 - 1.5x_1 + 2.5x_2 + 1$ (McCormick) f_{min} = -1.9133 at x_1 = -0.54719, x_2 = -1.54719 Search domain: 1.5 $\leq x_1 \leq$ 3.5, 0.5 $\leq x_2 \leq$ 2.5 $x_0 = [1.6, 2.4]^T$	

Table 1: (*continued*)

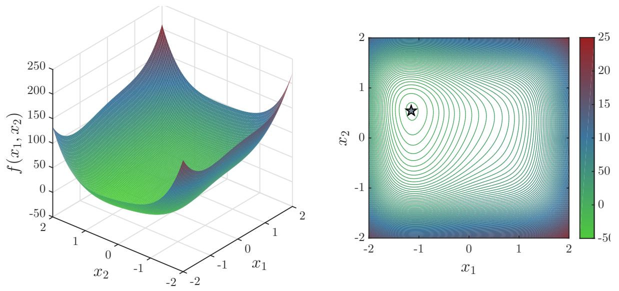
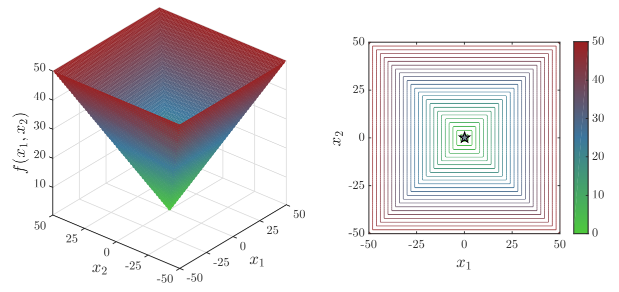
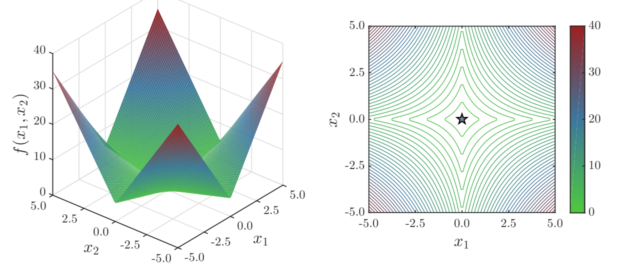
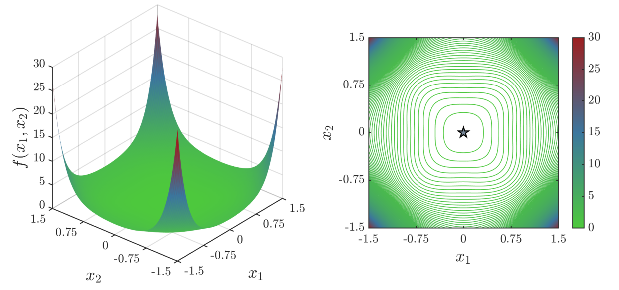
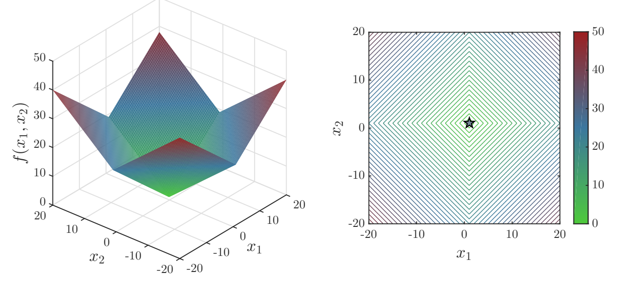
Function number	Function	Function plot
9	$f(x_1,x_2) = 5x_1^4 + 6x_2^4 - 6x_1^2 + 2x_1x_2 + 5x_2^2 + 15x_1 - 7x_2 + 13$ f_{min} = -6.4931 at x_1 = -1.1515, x_2 = 0.5455 Search domain: -2 $\leq x_1, x_2 \leq$ 2 $x_0 = [1.9, -1.9]^T$	
10	$f(x_1,x_2) = max(\lvert x_1\rvert, \lvert x_2\rvert)$ $f_{min} = 0$ at $x_1 = 0$, $x_2 = 0$ Search domain: -50 $\leq x_1, x_2 \leq$ 50 $x_0 = [-48, 43]^T$	
11	$f(x_1,x_2) = \lvert x_1\rvert + \lvert x_2\rvert + \lvert x_1\rvert.\lvert x_2\rvert$ $f_{min} = 0$ at $x_1 = 0$, $x_2 = 0$ Search domain: -5 $\leq x_1, x_2 \leq$ 5 $x_0 = [2.75, -5]^T$	
12	$f(x_1,x_2) = (x_1^2)^{(x_2^2+1)} + (x_2^2)^{(x_1^2+1)}$ $f_{min} = 0$ at $x_1 = 0$, $x_2 = 0$ Search domain: -1.5 $\leq x_1, x_2 \leq$ 1.5 $x_0 = [-1.5, 1.25]^T$	
13	$f(x_1,x_2) = \lvert x_1 - 1\rvert + \lvert x_2 - 1\rvert$ $f_{min} = 0$ at $x_1 = 1$, $x_2 = 1$ Search domain: -20 $\leq x_1, x_2 \leq$ 20 $x_0 = [-8, 20]^T$	

Table 1: (*continued*)

For a performance measure, we use the total number of function evaluations required to converge to test functions minimums among others such as algorithm running time, memory usage, success rate, computational accuracy, etc. [6]. Note that the central finite difference method is used to calculate functions gradients (i.e., $\nabla f(x)$) when they are required, and the number of function evaluations for those computations are also included in the total number of function evaluations. In case of the fact that a optimization method-line search condition combination fails to converge within the defined convergence tolerance, the total number of function evaluations for it are simply set as ∞. For search termination, the following expression [18] is used.

$$\frac{||\nabla f(x_{q+1})||_2}{1+|f(x_{q+1})|} \leq \epsilon \tag{15}$$

In (15), ϵ is the convergence tolerance value. To investigate influence of the convergence tolerance on the optimization method-line search condition combinations, the computational experiments are completed for both $\epsilon = 10^{-4}$ and $\epsilon = 10^{-3}$. On the other hand, for the sake of consistency and making a reliable benchmarking, all the coefficients (i.e., γ_0, μ, ν, η) used in the line search conditions and the starting point (i.e., x_0) for each test function are kept same for all the optimization method-line search condition combinations.

To conduct a reliable and an efficient benchmarking, the performance [13] and data [29] profiles are employed. For this purpose, we first define a test function and a optimization method-line search condition combination sets as $\mathcal{F}$ (i.e., Table 1) and $\mathcal{C}$ (i.e., Section 2), respectively. Then, the second step for the performance profiles is to compute the performance ratio to be:

$$R_{f,c} = \frac{NFE_{f,c}}{NFE_f^{min}} \tag{16}$$

where the $NFE_{f,c}$ is the total number of function evaluations required to converge to the minimum of the function $f : f \in \mathcal{F}$ while using the optimization method-line search condition combination $c : c \in \mathcal{C}$. The NFE_f^{min} is the minimum number of function evaluations provided by the fastest optimization method-line search condition combination for the function f. By using the corresponding ratio, finally the performance profile, $P_c(\upsilon) \in (0, 1)$, of a combination being used is then obtained as follows:

$$P_c(\upsilon) = \frac{1}{N_f} \sum_{f \in \mathcal{F}} \lambda_{f,c}(R_{f,c}, \upsilon) \tag{17}$$

where $\upsilon \geq 1$ is a performance factor and the N_f is the number of functions in the test set. The $\lambda_{f,c}(R_{f,c}, \upsilon)$ is defines as:

$$\lambda_{f,c}(R_{f,c}, \upsilon) = \begin{cases} 1, & \text{if } R_{f,c} \leq \upsilon \\ 0, & \text{otherwise} \end{cases}$$

For a quick interpretation of (17), it can be stated that the $P_c(\upsilon)$ provides information about how many test functions are able to be minimized by a specific optimization method-line search condition combination within a factor υ of the best combination. In addition, it can be achieved the number of wins for a combination by just looking at the $P_c(1)$ value, which also enables to rank the combinations. As noticed from (16) and (17), the performance profiles of the combinations always depend upon each other. Hence, it is not straightforward to make an independent assessment on the combinations. To overcome this issue, the data profiles [29] are used and they can be obtained with following expressions.

$$D_c(\psi) = \frac{1}{N_f} \sum_{f \in \mathcal{F}} \Gamma_{f,c}(NFE_{f,c}, \psi) \tag{18}$$

$\Gamma_{f,c}(NFE_{f,c},\psi)$ is:

$$\Gamma_{f,c}(NFE_{f,c},\psi) = \begin{cases} 1, & \text{if } NFE_{f,c} \leq \psi \\ 0, & \text{otherwise} \end{cases}$$

In (18), ψ is the number of function evaluations and $D_c(\psi)$ is the amount of test functions that can be minimized by the combination with that ψ. This information is very useful in case of having a computational budget (i.e., maximum allowed number of function evaluations).

4. Results and Discussion

A series of computational experiments have been completed to perform an overall assessment on the optimization method-line search condition combinations. Specifically, in total, 624 computational experiments, which consist of $f \in \mathcal{F} = 13$ and $c \in \mathcal{C} = 6$ for the SD method, and $f \in \mathcal{F} = 13$ and $c \in \mathcal{C} = 18$ for the CG methods, have been carried out with the two convergence tolerance values (i.e., $\epsilon = 10^{-4}$ and $\epsilon = 10^{-3}$). During each experiment, x_1 and x_2 values, the corresponding function and norm of gradient values (i.e., $f(x_1, x_2)$ and $||\nabla f(x_1, x_2)||_2$, respectively) are monitored for each iteration. Once the optimization method-line search condition combination converges to the function minimum, which means that the combination in question satisfies the converge condition (see (15)), the experiment is terminated and the total number of function evaluations are recorded for a performance measure of the combination. As a results of those experiments, the total number of function evaluations for all the optimization method-line search condition combinations are shown in Figures 2 and 3 for $\epsilon = 10^{-4}$ and $\epsilon = 10^{-3}$, respectively. Note that, in these figures, the total number of function evaluations are simply set as ∞ if the optimization method-line search condition combination fails to converge to the function minimum in question.

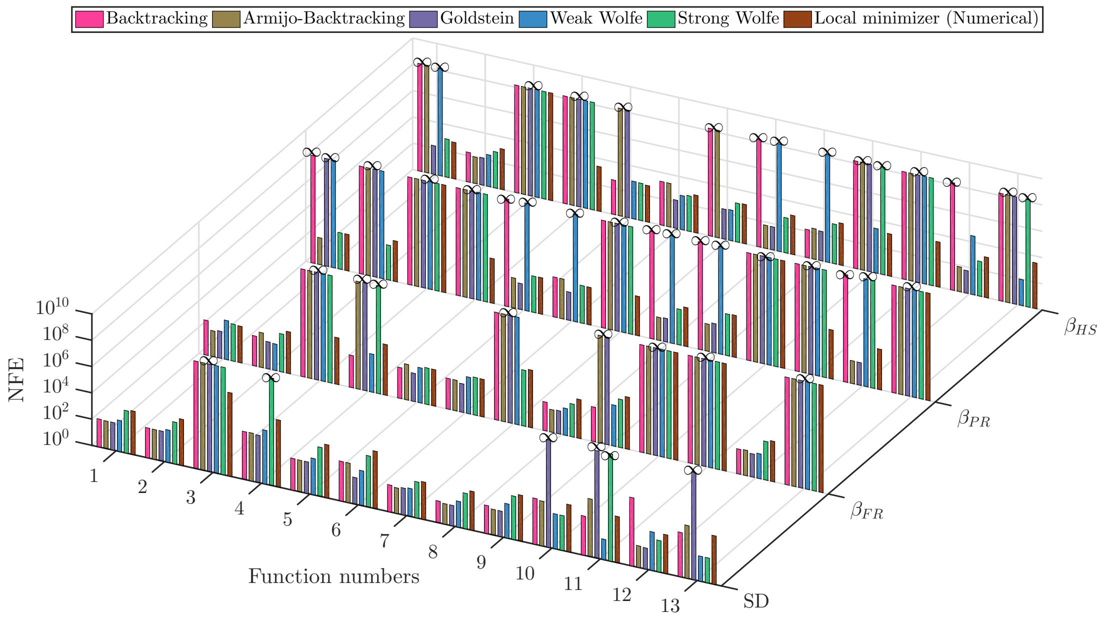


Figure 2: *Total number of function evaluations for all the optimization method-line search condition combinations on the test function set for* $\epsilon = 10^{-4}$

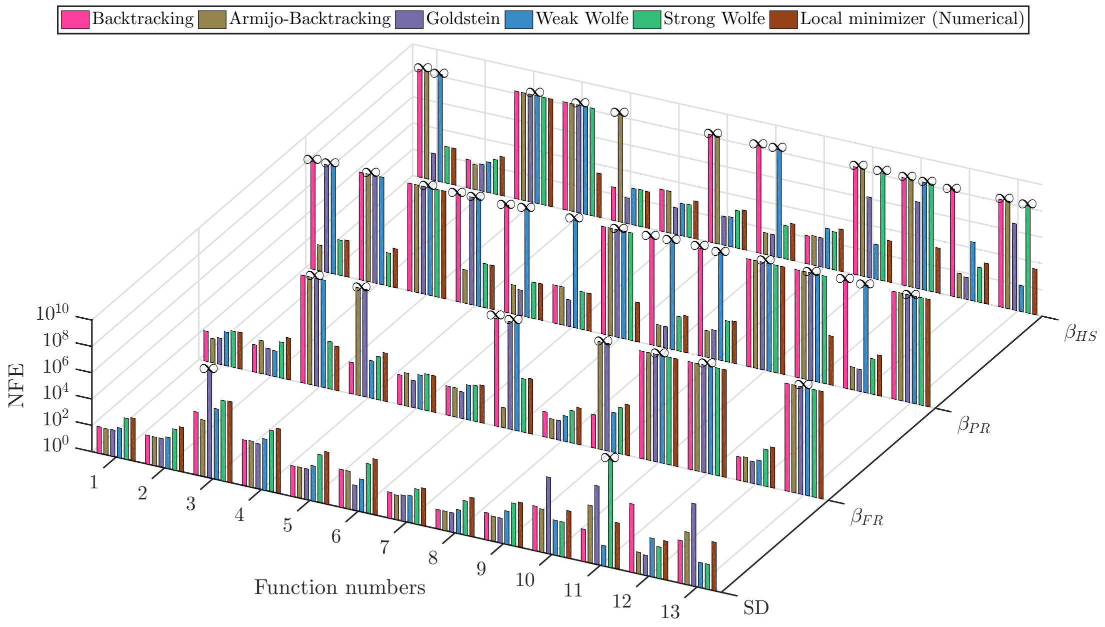
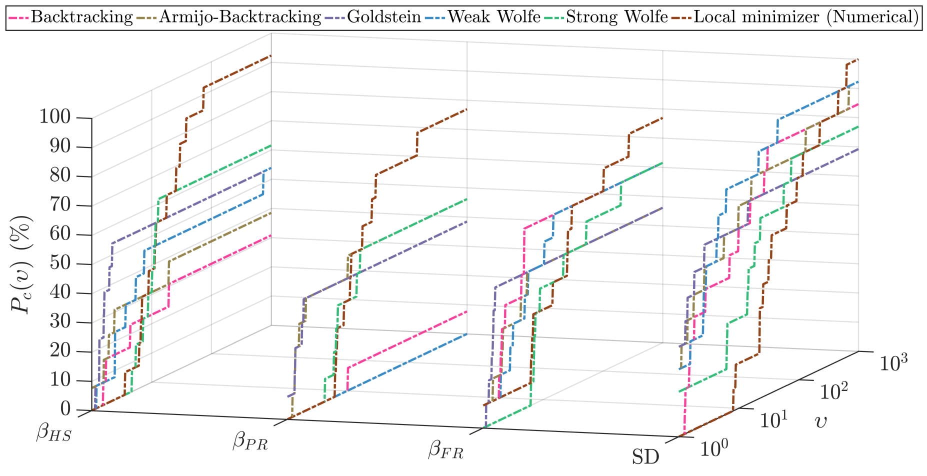


Figure 3: *Total number of function evaluations for all the optimization method-line search condition combinations on the test function set for* $\epsilon = 10^{-3}$

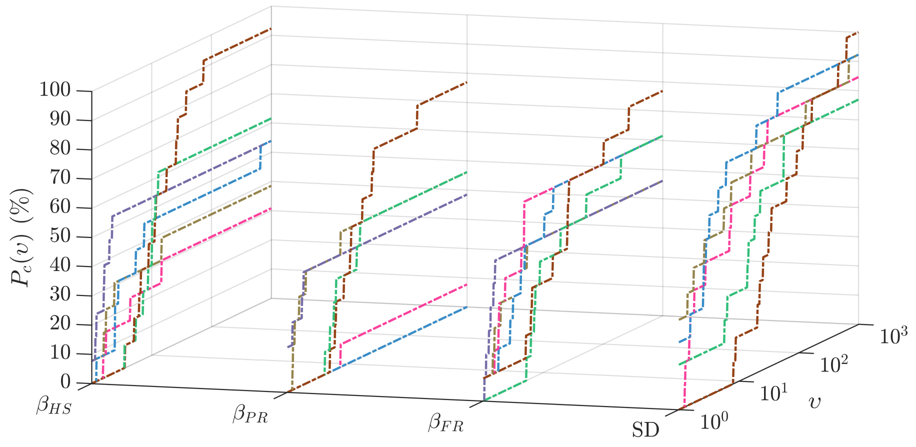
From these figures, we can simply say that the line search conditions have a great influence on the performance of the SD and CG methods. However, we can not obtain the answers for the following questions by just looking at these figures. What are the fastest, the most robust and the optimal optimization method-line search condition combinations for those test functions?, What is the percentage of the function that a particular optimization method-line search condition combination is successful on?, What is the probability of finding functions minimums if we choose one of these combinations?, etc. Those questions play a crucial role for an efficient and a reliable benchmarking of the optimization methods and line search conditions. Therefore, they are strongly required to be answered. To this end, the performance profiles of each optimization method-line search condition combination for $\epsilon = 10^{-4}$ and $\epsilon = 10^{-3}$ in a factor range from 10^0 to 10^3 (i.e., $10^0 \leq \upsilon \leq 10^3$) are illustrated in Figures 4 and 5. The reader may note that, in these figures, the performance profiles in three steps, as an example, are given for ranking purpose of the optimization method-line search condition combinations. The details about this ranking process will be provided later on. On the other hand, from these figures, we can determine the fastest combination/combinations on the test function set by looking at the probability values at the factor of $\upsilon = 1$ (i.e., $P_c(1)$). In Figure 4(a), for instance, the SD-GC combination is the fastest one with the probability of $P_c(1) = 30.77\%$. It means that the combination has the highest number of wins (i.e., finds the minimums of 30.77% of the test functions) compared to others. From the factor $\upsilon = 1$ to $\upsilon = 1.1$ of the fastest combination, the SD-GC combination preserves being the fastest one. At the factor $\upsilon = 1.1$, the SD-ABC combination catches up the SD-GC one with the same probability. Until the factor $\upsilon = 1.27$ of the fastest combination, the SD-GC combination keeps the same probability whereas the SD-ABC combination has the same probability till the factor $\upsilon = 1.38$. The fastest combination, the SD-GC, performance keeps rising as the υ increases until the $\upsilon = 14.28$. At that factor, the combination reaches maximum performance which is $P_c(14.28) = 69.23\%$. From that factor on, even if we increase the υ, the combination performance remains same. In other words, the SD-GC combination is able to find the minimums of 69.23% of test functions (i.e., 9 out of 13 test functions). The combination in question fails on 4 test functions which are the functions 3, 10, 11 and 13 (see Figure 2). However, the other combination options, which are more successful than the SD-GC combination, are available for $\upsilon \geq 14.28$. For instance, the SD-WWC combination is able to be successful on the 76.92% of the test functions for the factor $\upsilon = 14.28$ of the fastest one (i.e., the SD-GC). And its performance keeps rising until $\upsilon = 44.76$ where reaches up to its maximum probability $P_c(44.76) = 92.31\%$. The other options for the $\upsilon \geq 14.28$ might be the SD-BC, SD-ABC, SD-SWC, SD-LM, FR-LM, PR-LM and HS-LM combinations if we desire to find the minimums of more than 69.23% of the test functions. One could also notice that only the LM is able to provide highest success rates to CG methods whereas the SD is quite successful with the help of other line search conditions. This observation shows the sensitivity of the CG methods to the computation of the step length γ. To put a finer point on it, if we choose a line search condition other than LM, we would expect a significant drop in the performance of the CG methods. This is attributed to the fact that the second term in (4) may be dominant on defining search direction s_{q+1}, which results in a direction of ascent [30]. On the other hand, it is also evident from Figure4(a) that the choosing of the LM as a line search condition for the SD and CG methods brings high computational burden. However, the LM is the most robust one among others. In other words, it is the only one that is successful on all the test functions for the SD method within the factor $\upsilon \geq 634.1$ of the fastest combination . For the FR, PR and HS methods, the success rates are 76.92%, 76.92% and 92.31% within the factors $\upsilon \geq 272.4$, $\upsilon \geq 148.6$ and $\upsilon \geq 74.14$ of the fastest combination, respectively. Besides those, in case of choosing the LM for a line search condition of the CG methods, the HS one draws attention with the highest probability of 92.31% and the lowest factor $\upsilon \geq 74.14$ compared to other CG methods. This method could be the best one among other CG methods on this test function set with the convergence tolerance value $\epsilon = 10^{-4}$. In addition to that, the HS-LM combination with the factor $\upsilon = 74.14$ of the fastest one has more number of wins than SD-LM combination at $\upsilon = 74.14$ (i.e, $P_c(74.14) = 61.54\%$). However, at the first glance, the SD method generally exhibits better performance than CG methods in the line search conditions except for the LM. To elaborate this observation and rank the combinations according to convergence speed, the successive excluding procedure is utilized. The purpose of the successive excluding procedure is to determine the second, third and so on fastest combinations, because the performance profiles are relative, which means that they depend on each other. Therefore, it is not possible to extract the data for the speed sequence of the combination from Figure4(a). The idea behind the successive excluding procedure is to subtract the performance profile of the fastest combination in question in order to determine next fastest one. From Figure 4(a), for instance, the SD-GC combination is determined to be the fastest one as previously explained. With excluding this combination, we can generate new performance profiles with a new combination domain (i.e., $f \in \mathcal{F} = 13$ and $c \in \mathcal{C} = 5$ for the SD method, and $f \in \mathcal{F} = 13$ and $c \in \mathcal{C} = 18$ for the CG methods), as shown in Figure 4(b). The same procedure to determine fastest combination as conducted before is also valid for this figure. In this case, the fastest combination is the SD-ABC with the probability $P_c(1) = 30.77$. From global perspective, we can say that this combination is the second fastest one and it holds the second place until the factor $\upsilon = 2.18$ with the probability $P_c(2.18) = 46.15$. At that factor, the HS-GC combinations beats the SD-ABC with probability $P_c(2.18) = 53.85$. However, the HS-GC reach up the maximum performance at this factor whereas the SD-ABC performance keeps rising as the υ increases. Moreover, the SD-ABC is capable of converging to 92.31% of the test functions minimums within the factor $\upsilon \geq 683.9$. In a similar manner, the third fastest combinations are determined to be the SD-WWC and PR-GC with the probability $P_c(1) = 23.08$ (see Figure 4(c)). Totally 15 steps are completed to rank the combinations based on the convergence speed, but, to explain the combination ranking concept, the figures for only 3 steps are given here as examples. However, the exact speed sequence of the combinations with the probability values for $\epsilon = 10^{-4}$ are provided in Table 2. Here, the reader may note that the steps in Figure 4 and Table 2 denote corresponding fastest combination or combinations. For example, Step-4 in Table 2 points out that the SD-BC is the fourth fastest combination. This is achieved by consecutively excluding first, second and third fastest combinations corresponding to Step-1, Step-2 and Step-3, respectively. The same is also valid for the following figure and table related to performance profiles.

To be able explore influence of the convergence tolerance value on the optimization method-line search condition combinations, the same computational experiments are run for a higher tolerance value $\epsilon = 10^{-3}$. By using the data obtained through those experiments, the performance and data profiles are generated for this convergence tolerance value as well. Figure 5 displays the performance profiles, which includes three plots achieved via successive excluding procedures as previously explained. It is observed from Figure 5(a) that the SD-GC combination still keeps being fastest one with the same probability $P_c(1) = 30.77\%$ in this convergence tolerance value too. The second place, the SD-ABC, also remains same as before, but it has lower probability of 23.08%. This lower probability means that some other combinations performance improve with increasing convergence tolerance value because the performance profiles are relative. In other words, any changes in a combination performance are directly reflected on others performances. As an example on improving performance of the combination with increasing convergence tolerance value, the SD-WWC can be pointed out. More specifically, for $\epsilon = 10^{-4}$, this combination is able to find the minimums of 92.31% of the test functions within the factor $\upsilon \geq 44.76$ of fastest combination whereas its maximum probability is 100% within the factor $\upsilon \geq 104.89$ for $\epsilon = 10^{-3}$. In a similar manner, the HS-GC combination performance improves as the convergence tolerance value increases. As such, its maximum probability increases from $P_c(\upsilon \geq 2.18) = 53.85\%$ for $\epsilon = 10^{-4}$ to $P_c(\upsilon \geq 4.56) = 61.54\%$ for $\epsilon = 10^{-3}$. In other words, it finds one more function minimum (i.e., totally 8 out of 13 test functions) with lowering accuracy. On the other hand, some combinations, including SD-GC, SD-LM, FR-GC, FR-BC, FR-LM, PR-BC, PR-GC, PR-WWC, PR-LM, HS-BC, HS-ABC, HS-SWC, HS-LM, maximum probability are not influenced by the convergence tolerance value at all. There are only small reductions in the number of function evaluations. The reduction in the number of function evaluations with increasing convergence tolerance value (i.e., decreasing converge accuracy) is an inherent phenomena, because lower accuracy generally requires lower iterations. This means that we would expect lower number of function evaluations to compute step length γ. Similar observations on the influence of the convergence tolerance value, which have been conducted so far, can be completed for other combinations as well. As stated before, to rank the combinations, the successive excluding procedure is also applied for the $\epsilon = 10^{-3}$, as seen Figures 5(b) and (c). In addition, the complete speed sequence is given in Table 3. By performing a comparison between Tables 2 and 3, one could say that changing of the convergence tolerance value leads to different combination rank. The end-users should consider this fact while selecting the combination.

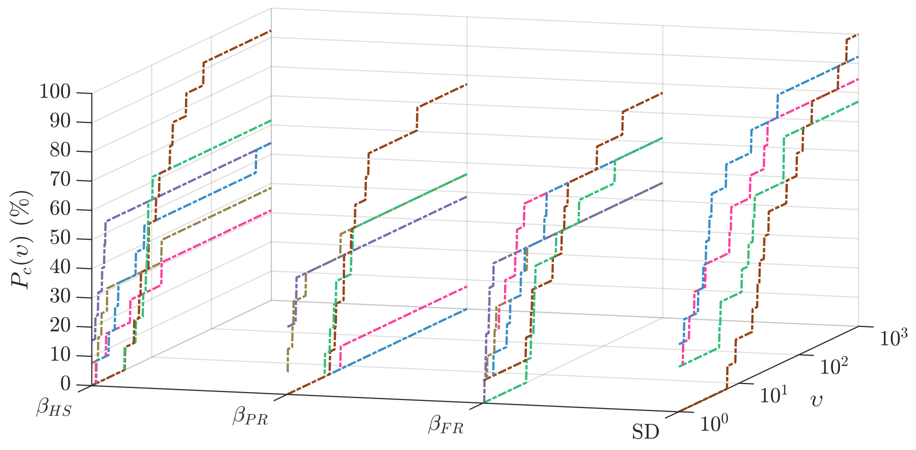
On the other hand, the end-users may have a computational budget (i.e., maximum number of function evaluation for this study) for finding minimums of the test functions. And they need to know how many function minimum can be found within the given budget. This requires an independent assessment on the optimization method-line search condition combinations, which can not be provided by the performance profiles. Hence, we use data profiles in this case. As such, Figures. 6(a) and (b) show the data profiles of all the combinations within a range $10^1 \leq \psi \leq 10^5$ for $\epsilon = 10^{-4}$ and 10^{-3}, respectively. Those plots provide valuable information for selecting the optimal combination based on the computational budget. Let's say, for instance, we are given a computational budget of 10^2 number of function evaluations (i.e., $\psi = 10^2$) and the convergence tolerance $\epsilon = 10^{-4}$ for finding minimums of at least 50% test functions. In these conditions, by looking at Figure 6(a), one can observe that there are no any combination options which have sufficient probability (i.e., $D_c(10^2) \geq 50\%$). For $\psi = 10^2$ and $\epsilon = 10^{-4}$, the maximum probability is 30.77% that belongs to the SD-ABC combination. To meet these terms, the computation budget has to be raised approximately two times. As such, for 2×10^2



(a) *Step-1*

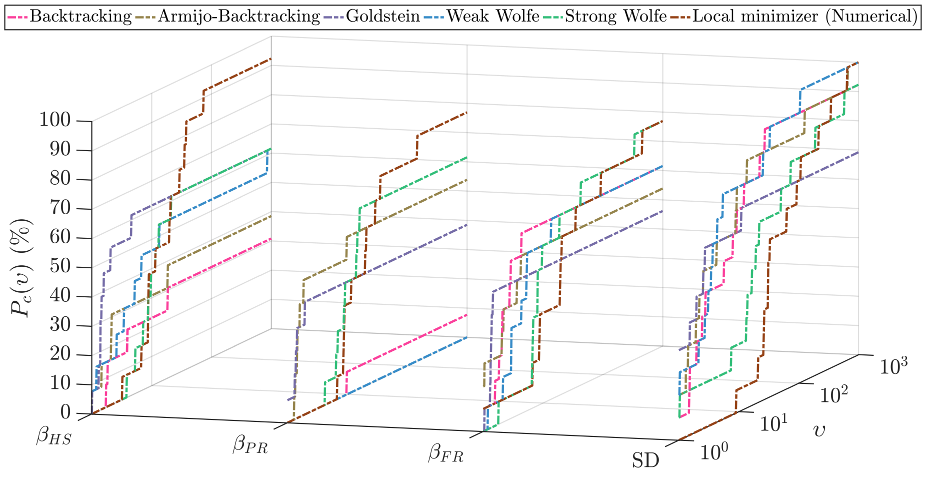


(b) *Step-2*

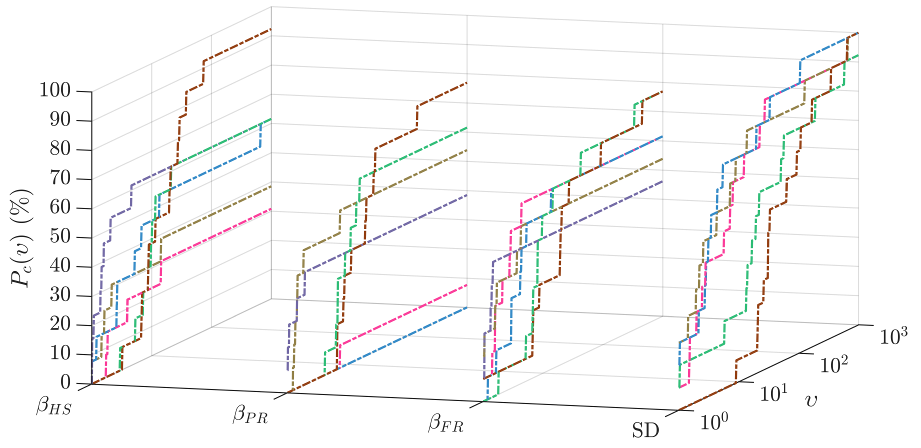


(c) *Step-3*

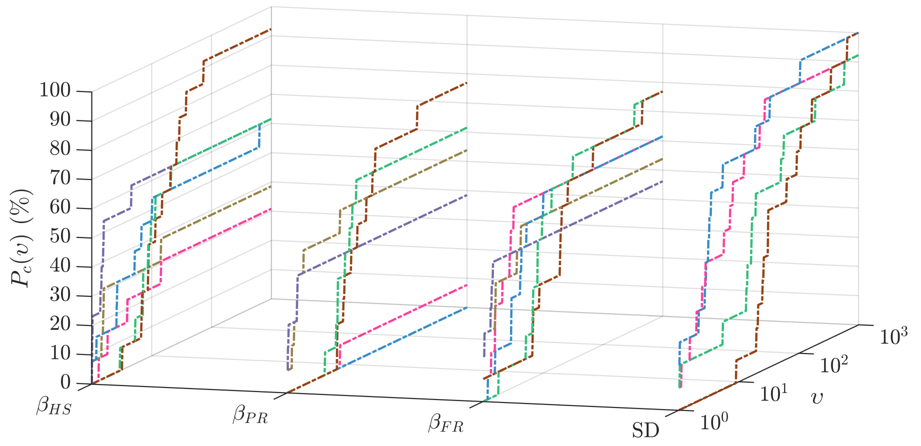
Figure 4: *Performance profiles of all the optimization method-line search condition combinations for* $\epsilon = 10^{-4}$



(a) *Step-1*



(b) *Step-2*



(c) *Step-3*

Figure 5: *Performance profiles of all the optimization method-line search condition combinations for $\epsilon = 10^{-3}$*

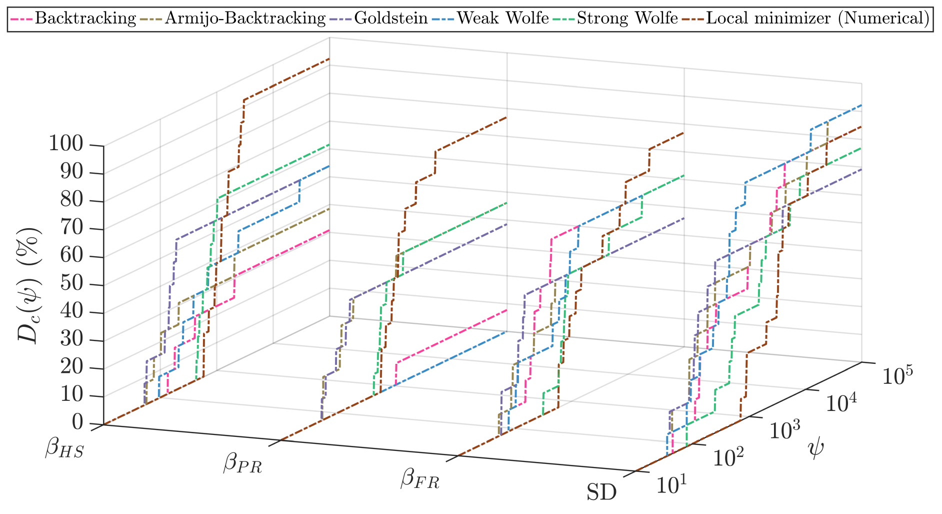
Table 2: *Speed sequence of all the optimization method-line search condition combinations for* $\epsilon = 10^{-4}$

#	Combination	$P_c(1)$ %
Step-1	SD-GC	30.77
Step-2	SD-ABC	30.77
Step-3	SD-WWC and PR-GC	23.08
Step-4	SD-BC	23.08
Step-5	HS-GC	23.08
Step-6	SD-SWC, FR-ABC, FR-GC and PR-ABC	15.38
Step-7	FR-BC	30.77
Step-8	HS-ABC and HS-WWC	23.08
Step-9	FR-WWC	30.77
Step-10	HS-LM	30.77
Step-11	SD-LM	30.77
Step-12	HS-BC and HS-SWC	23.08
Step-13	PR-SWC and PR-LM	30.77
Step-14	FR-SWC and FR-LM	38.46
Step-15	PR-BC	7.692
–	PR-WWC	0

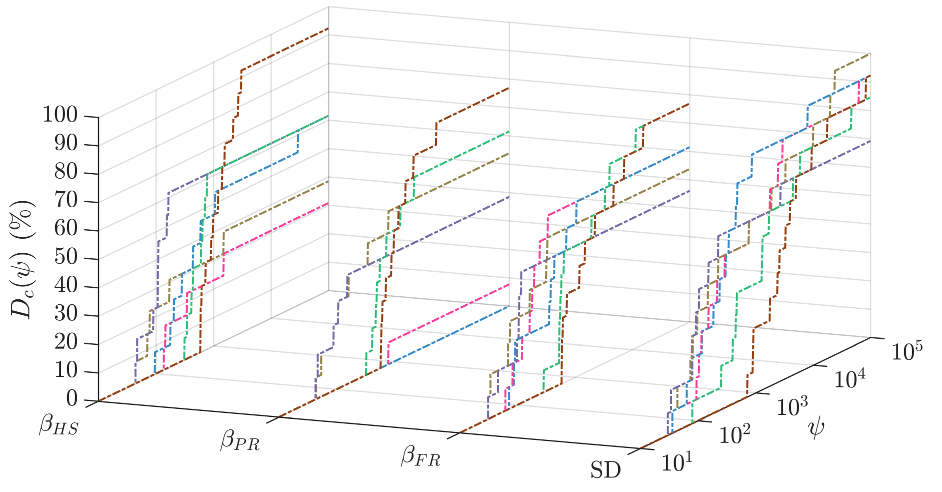
Table 3: *Speed sequence of all the optimization method-line search condition combinations for* $\epsilon = 10^{-3}$

#	Combination	$P_c(1)$ %
Step-1	SD-GC	30.77
Step-2	SD-ABC	23.08
Step-3	SD-WWC, FR-ABC and FR-GC	15.38
Step-4	SD-BC and PR-GC	23.08
Step-5	HS-GC	38.46
Step-6	PR-ABC	38.46
Step-7	FR-BC	38.46
Step-8	HS-ABC and HS-WWC	23.08
Step-9	FR-WWC	38.46
Step-10	SD-SWC	30.77
Step-11	HS-BC and HS-LM	23.08
Step-12	SD-LM	30.77
Step-13	HS-SWC	30.77
Step-14	PR-SWC	38.46
Step-15	FR-SWC	53.85
Step-16	PR-LM	69.23
Step-17	FR-LM	76.92
Step-18	PR-BC	7.692
–	PR-WWC	0

and $\epsilon = 10^{-4}$, two combination options, SD-GC and HS-GC, stand out with the probability of 53.85%. On the other hand, instead of increasing the computational budget, we may also reduce the converge accuracy to $\epsilon = 10^{-3}$. In this case (i.e., $\psi = 10^2$ and $\epsilon = 10^{-3}$), the maximum probability is 38.46% for the SD-ABC and SD-GC combinations (see Figure 6(b)). They still can not meet the solution requirement, $D_c(10^2) \geq 50\%$, but they are able to find one more function minimum compared to $\epsilon = 10^{-4}$ ones. Similar interpretations can be carried out for any computational budget within $10^1 \leq \psi \leq 10^5$, and for $\epsilon = 10^{-4}$ and 10^{-3} through those plots as well.



(a) $\epsilon = 10^{-4}$



(b) $\epsilon = 10^{-3}$

Figure 6: *Data profiles of all the optimization method-line search condition combinations for two convergence tolerance values*

So far, through Figures 2, 3, 4 and 5, and Tables 2 and 3, we have conducted an efficient and a reliable comparative evaluations on the optimization method-line search condition combinations. Based on those, now it is time to decide which optimization method-line search condition combination/combinations would be the optimal one for this test function set. As stated previously, the CG methods requires the LM for reasonable performance because their sensitivity to the step length. This fact brings high computational cost. Hence, the CG methods-LM combinations may be ruled out for this test function set. On the other hand, it is noteworthy that the CG methods-LM combinations exhibit better performance than the SD-LM one. Especially, the HS-LM combination outperforms the other CG and SD methods-LM combinations. Considering all those and by making a trade-off between convergence speed and robustness, the SD-WWC combination is determined as the optimal combination for this test function set. Note that, the most robust combination, having highest probability $P_c(\upsilon) = 100\%$ (i.e., finds all functions minimums), for the $\epsilon = 10^{-4}$ is the SD-LM whereas the SD-WWC and the SD-LM combinations are the most robust ones for the $\epsilon = 10^{-3}$.

5. Conclusion

This paper has presented a benchmarking study on the combinations of the well-known optimization methods and line search conditions. A series of computational experiments on the thirteen test functions have been completed with these combinations. An iterative procedure has been applied to find the test function minimums and number of function evaluations at each iteration have been recorded. For benchmarking purpose, the total number of function evaluations when the combination in question converges to the function minimum within the defined convergence tolerance have been set as a performance measure. The same procedure have been conducted for all the combinations and the test functions. The obtained data through those computational experiments have been used to create the performance and data profiles. These profiles have enabled us to perform a reliable and an efficient benchmarking on the combinations. The study has provided the following conclusions:

- The step length computation technique is crucial for the SD and CG methods success and performance. For optimal performance and remarkable success, both the step length and the search direction require a special care.

- Generally speaking, the WWC, SWC and LM, which are the gradient related step length computation methods, provide the lower number of iterations for converging to the functions minimums, but the number of function evaluations per iteration is notably high. Therefore, they are refereed to as computationally expensive techniques in this study.

- Considering the fact stated in the previous conclusion, it has been determined that the SD-GC combination is the fastest one for both convergence tolerance values. The SD-ABC combination takes second place. The fact behind this convergence speed is that the ABC and GC do not require gradient computation while trying to satisfy line search conditions for computing step length.

- It has been determined that the CG methods success rate with using line search conditions except for the LM is quite lower than those in SD method. This shows the sensitivity of the CG methods to the exact local minimizer. For a reasonable performance in the CG methods, the choice should be the LM or in some cases, the SWC can be also used even if they bring high computational burden.

- The optimal combination for the test function set used in this study is the SD-WWC whereas the most robust one is the SD-LM combination with a high convergence accuracy.

References

[1] Andrei, N. (2008). An unconstrained optimization test functions collection. Advanced Modeling and Optimization, 10(1), 147-161.

[2] Andrei, N. (2010). New accelerated conjugate gradient algorithms as a modification of dai–yuan's computational scheme for unconstrained optimization. Journal of Computational and Applied Mathematics, 234(12), 3397-3410. doi: 10.1016/j.cam.2010.05.002

[3] Arsenault, R., Poulin, A., Côté, P. and Brissette, F. (2014). Comparison of stochastic optimization algorithms in hydrological model calibration. Journal of Hydrologic Engineering, 19(7), 1374-1384. doi: 10.1061/(ASCE)HE.1943-5584.0000938

[4] Babaie-Kafaki, S., Ghanbari, R. and Mahdavi-Amiri, N. (2010). Two new conjugate gradient methods based on modified secant equations. Journal of Computational and Applied Mathematics, 234(5), 1374-1386. doi: 10.1016/j.cam.2010.01.052

[5] Bartz-Beielstein, T., Doerr, C., Berg, D.V.D., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., Malan, K.M., Moore, J. H., Naujoks, B., Orzechowski, P., Volz, V., Wagner, M. and Weise, T. (2020). Benchmarking in optimization: Best practice and open issues. Available at: https://ui.adsabs.harvard.edu/abs/2020arXiv200703488B/abstract

[6] Beiranvand, V., Hare, W. and Lucet, Y. (2017). Best practices for comparing optimization algorithms. Optimization and Engineering, 18(4), 815-848. doi: 10.48550/arXiv.1709.08242

[7] Bento, G., da Cruz Neto, J.X. and Santos, P. (2013). An inexact steepest descent method for multicriteria optimization on riemannian manifolds. Journal of Optimization Theory and Applications, 159(1), 108-124. doi: 10.1007/s10957-013-0305-9

[8] Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. Comp. Rend. Sci. Paris, 25(2), 536-538.

[9] Dai, Y. H. (2002). Conjugate gradient methods with armijo-type line searches. Acta Mathematicae Applicatae Sinica, 18(1), 123-130.

[10] Dai, Y. H. (2003). A family of hybrid conjugate gradient methods for unconstrained optimization. Mathematics of Computation, 72(243), 1317-1328. doi: 10.1090/S0025-5718-03-01491-1

[11] Diachin, L. F., Knupp, P., Munson, T. and Shontz, S. (2006). A comparison of two optimization methods for mesh quality improvement. Engineering with Computers, 22(2), 61-74. doi: 10.1007/s00366-006-0015-0

[12] Doerr, C., Ye, F., Horesh, N., Wang, H., Shir, O.M. and Bäck, T. (2020). Benchmarking discrete optimization heuristics with iohprofiler. Applied Soft Computing, 88, 106027. doi:10.48550/arXiv.1912.09237

[13] Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2), 201-213. doi:10.1007/s101070100263

[14] Dolan, E. D., Moré, J. J. and Munson, T. S. (2006). Optimality measures for performance profiles. SIAM Journal on Optimization, 16(3), 891-909.

[15] Fatemi, M. (2016). A new efficient conjugate gradient method for unconstrained optimization. Journal of Computational and Applied Mathematics, 300, 207-216. doi:10.1016/j.cam.2015.12.035

[16] Feng, D., Sun, M. and Wang, X. (2017). A family of conjugate gradient methods for large-scale nonlinear equations. Journal of Inequalities and Applications, 2017(1), 1-8. doi: 10.1186/s13660-017-1510-0

[17] Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. The Computer Journal, 7(2), 149-154. doi: 10.1093/comjnl/7.2.149

[18] Griva, I., Nash, S.G. and Sofer, A. (2009). Linear and nonlinear optimization, 2nd edition. Society for Industrial and Applied Mathematics.

[19] Hager, W.W. and Zhang, H. (2006). A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization, 2(1), 35-58.

[20] Haug, E., Arora, J. and Matsui, K. (1976). A steepest-descent method for optimization of mechanical systems. Journal of Optimization Theory and Applications, 19(3), 401-424. doi: 10.1007/BF00941484

[21] Hestenes, M. R., Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49(6), 409-436.

[22] 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. doi: 10.48550/arXiv.1308.4008

[23] Khan, K. and Lobiyal, D. (2017). Performance evaluation of different optimization techniques for coverage and connectivity control in backbone based wireless networks. Wireless Personal Communications, 96(3), 4329-4345. doi: 10.1007/s11277-017-4389-7

[24] Kiran, K. (2021). Performance analysis of steepest descent-line search condition combinations in nonlinear least squares fitting of CMM data. European Journal of Science and Technology, (28), 1190-1196. doi: 10.31590/ejosat.1012096

[25] Kiran, K. and Kayacan, M. C. (2019). Effect of material removal on workpiece dynamics in milling: Modeling and measurement. Precision Engineering, 60, 506-519. doi: 10.1016/j.precisioneng.2019.09.003

[26] Kiran, K., Rubeo, M., Kayacan, M. C. and Schmitz, T. (2017). Two degree of freedom frequency domain surface location error prediction. Precision Engineering, 48, 234-242. doi: 10.1016/j.precisioneng.2016.12.006

[27] Liu, Q., Chen, W.N., Deng, J.D., Gu, T., Zhang, H., Yu, Z. and Zhang, J. (2017). Benchmarking stochastic algorithms for global optimization problems by visualizing confidence intervals. IEEE Transactions on Cybernetics, 47(9), 2924-2937. doi:10.1109/TCYB.2017.2659659

[28] Liu, X. and Reynolds, A.C. (2016). A multiobjective steepest descent method with applications to optimal well control. Computational Geosciences, 20(2), 355-374.

[29] Moré, J. J. and Wild, S.M. (2009). Benchmarking derivative-free optimization algorithms. SIAM Journal on Optimization, 20(1), 172-191. doi:10.1137/080724083

[30] Nocedal, J. and Wright, S. (2006). Numerical optimization. Springer Science & Business Media.

[31] Polak, E. and Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. Revue Française D'informatique et de Recherche Opérationnelle, 3(R1), 35-43.

[32] Porcelli, M. and Toint, P.L. (2019). A note on using performance and data profiles for training algorithms. ACM Transactions on Mathematical Software (TOMS), 45(2), 1-10. doi: 10.1145/3310362

[33] Quiroz, E.P., Quispe, E. and Oliveira, P.R. (2008). Steepest descent method with a generalized armijo search for quasiconvex functions on riemannian manifolds. Journal of Mathematical Analysis and Applications, 341(1), 467-477. doi: 10.1016/j.jmaa.2007.10.010

[34] 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. doi:10.1016/j.istruc.2020.04.026

[35] Rojas-Labanda, S. and Stolpe, M. (2015). Benchmarking optimization solvers for structural topology optimization. Structural and Multidisciplinary Optimization, 52(3), 527-547. doi: 10.1007/s00158-015-1250-z

[36] Salonga, P.K., Inaudito, J.M. and Mendoza, R. (2019). An unconstrained minimization technique using successive implementations of golden search algorithm. In: AIP Conference Proceedings, vol. 2192, 060018, pp. 1-20. AIP Publishing LLC. doi: 10.1063/1.5139164

[37] Samir, C., Absil, P.A., Srivastava, A. and Klassen, E. (2012). A gradient-descent method for curve fitting on riemannian manifolds. Foundations of Computational Mathematics, 12(1), 49-73. doi: 10.1007/s10208-011-9091-7

[38] Schmitz, T. L. and Smith, K. S. (2009). Machining Dynamics: Frequency Response to Improved Productivity. New York: Springer. doi: 10.1007/978-0-387-09645-2

[39] Schneider, P. I., Garcia Santiago, X., Soltwisch, V., Hammerschmidt, M., Burger, S. and Rockstuhl, C. (2019). Benchmarking five global optimization approaches for nano-optical shape optimization and parameter reconstruction. ACS Photonics, 6(11), 2726-2733. doi:10.1021/acsphotonics.9b00706

[40] Shi, Z.J., Shen, J. (2005). New inexact line search method for unconstrained optimization. Journal of Optimization Theory and Applications, 127(2), 425-446. doi: 10.1007/s10957-005-6553-6

[41] Tangherloni, A., Spolaor, S., Cazzaniga, P., Besozzi, D., Rundo, L., Mauri, G. and Nobile, M. S. (2019). Biochemical parameter estimation vs. benchmark functions: A comparative study of optimization performance and representation design. Applied Soft Computing, 81, 105494. doi:10.1016/j.asoc.2019.105494

[42] Truong, T. T. and Nguyen, H.T. (2020). Backtracking gradient descent method and some appli-

cations in large scale optimisation. Part 2: Algorithms and Experiments. Applied Mathematics & Optimization, 84(3), 2557-2586. doi:10.1007/s00245-020-09718-8

[43] Villaverde, A. F., Fröhlich, F., Weindl, D., Hasenauer, J. and Banga, J. R. (2019). Benchmarking optimization methods for parameter estimation in large kinetic models. Bioinformatics, 35(5), 830-838. doi:10.1093/bioinformatics/bty736

[44] Zhang, S., Li, S., Harley, R.G. and Habetler, T.G. (2017). Performance evaluation and comparison of multi-objective optimization algorithms for the analytical design of switched reluctance machines. CES Transactions on Electrical Machines and Systems, 1(1), 58-65.

[45] Zhu, L.M., Ding, H. and Xiong, Y.L. (2003). A steepest descent algorithm for circularity evaluation. Computer-Aided Design, 35(3), 255-265. doi:10.1016/S0010-4485(01)00210-X