IJOCTA
An International Journal of
Optimization and Control:
Theories & Applications
2010

RESEARCH ARTICLE

Comparative assessment of smooth and non-smooth optimization solvers in HANSO software

Ali Hakan Tor[*]

Abdullah Gül University, Faculty of Computer Science, Department of Mathematics, Kayseri, Turkey
hakan.tor@agu.edu.tr

ARTICLE INFO	ABSTRACT
	The aim of this study is to compare the performance of smooth and nonsmooth optimization solvers from HANSO (Hybrid Algorithm for Nonsmooth Optimization) software. The smooth optimization solver is the implementation of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method and the nonsmooth optimization solver is the Hybrid Algorithm for Nonsmooth Optimization. More precisely, the nonsmooth optimization algorithm is the combination of the BFGS and the Gradient Sampling Algorithm (GSA). We use well-known collection of academic test problems for nonsmooth optimization containing both convex and nonconvex problems. The motivation for this research is the importance of the comparative assessment of smooth optimization methods for solving nonsmooth optimization problems. This assessment will demonstrate how successful is the BFGS method for solving nonsmooth optimization problems in comparison with the nonsmooth optimization solver from HANSO. Performance profiles using the number iterations, the number of function evaluations and the number of subgradient evaluations are used to compare solvers.

1. Introduction

Researchers working in different areas, for instance, in economics, engineering, data mining and machine learning encounter different types of optimization problems including those with smooth, non-smooth, convex or nonconvex objective and/or constraint functions. While these researchers search for an optimal solution of their real-life problems, they need suitable software. Most of existing optimization software and methods contain some user defined parameters and it is not always easy to choose these parameters for a particular problem. The choice of these parameters may strongly depend on the application area. Researchers, who apply optimization methods in their research, prefer optimization software that are robust to the choice of their parameters. For example, in the book [1], optimization problems arising in economics are analyzed and some robust

methods for solving them are discussed in [2, 3]. The theory and application of engineering problems can be found in [4], in particular, some genetic engineering problems are discussed in [5, 6]. In the areas such as data mining, machine learning and control theory the sources [7–9] are very useful and important for readers. In addition, you can find all the mathematical theory and application related to the nonsmooth optimization theory in the book [10], which used as a guide book while preparing this study. On the other hand, it is possible to increase the examples in the application area, but it is useful to talk about HANSO software as soon as possible without going beyond our purpose.

In this paper, we provide a comparative assessment of two nonsmooth optimization solvers.

---

*Corresponding Author

Both these solvers are available in HANSO software [11]. The first solver is the implementation of the BFGS method (Broyden-Fletcher-Goldfarb-Shanno method). The second solver is called HANSO (Hybrid Algorithm for Non-Smooth Optimization). This solver is the combination of the BFGS and Gradient Sampling Algorithm (GSA). Implementations of both algorithms are available in [11]. There are two main reasons to carry out this comparison. The BFGS is the smooth optimization algorithm, however the GSA is developed specifically for solving nonconvex nonsmooth optimization problems. This research will, in particular, show how comparable is smooth optimization solver for solving nonsmooth optimization problems. On the other side, BFGS is a part of HANSO algorithm. Our computational results will demonstrate how much the BFGS can improve the performance of the GSA.

We use the collection of 36 academic test problems to test algorithms. This collection contains convex and nonconvex nonsmooth optimization test problems and they different number of variables. The detailed description of these problems can be found in [10].

The rest of the paper is organized as follows. In the following Section 2, we provide the necessary information about the algorithms in HANSO software. In Section 3, the academic test problems used in this study are described. Summary of computational results is presented in Section 4. Discussion of computational results using performance profiles is given in Section 5. Some concluding remarks are provided in the final Section 6.

2. HANSO (Hybrid Algorithm for Non-Smooth Optimization)

Version 2.2 of HANSO developed by Michael Overton is used in this study. It has General Public License (GNU) as published by the Free Software Foundation, so anybody can redistribute it and/or modify it under the terms of this license.

HANSO is intended to seek a minimum value of non-smooth, non-convex functions, but also applicable to functions that are smooth, convex or both. It is based on the BFGS algorithm and GSA. You can find some details about BFGS and GSA in the following subsections.

2.1. BFGS Algorithm

BFGS algorithm suggested independently by Broyden, Fletcher, Goldfarb, and Shanno, in 1970

uses the Quasi-Newton algorithm which is a generalization of the secant method. The main difference between BFGS and Quasi-Newton algorithms is that it uses and maintains different properties of the matrix when updating formulas. In BFGS, the Hessian matrix is not calculated. Instead of this calculation, BFGS uses inverse Hessian matrix approximation using information from gradient evaluation. BFGS is normally used for optimizing smooth, not necessarily convex, functions, for which the convergence rate is generically superlinear. However, BFGS has acceptable performance even for non-smooth optimization problems, typically with a linear convergence rate as long as a weak Wolfe line search is used. This version of BFGS will work well both for smooth and non-smooth functions and has a stopping criterion that applies for both cases [12].

The weak Wolfe line search is far less complicated than the standard strong Wolfe line search. In addition to this fact, there is no disadvantage to using the weak Wolfe line search compared to the strong Wolfe line search when Newton or BFGS methods are used for smooth problems and BFGS or bundle methods are used for non-smooth problems. Thus, HANSO prefers to use the weak Wolfe line search with parameter $c_1 = 0$ for the sufficient decrease condition and parameter $c_2 = 0.5$ for the weak condition on the directional derivative. As indicated in the code [11],

> "For usual convergence theory for smooth functions, normally one requires $0 < c_1 < c_2 < 1$, but $c_1 = 0$ is fine in practice. May want $c_1 = c_2 = 0$ for some non-smooth optimization algorithms such as Shor or bundle, but not BFGS. Setting $c_2 = 0$ may interfere with superlinear convergence of BFGS in smooth case."

In this context, we can say that we can change the c_2 parameter, but it is not appropriate to make it 0. In our calculations, the code was not intervened and the results were obtained with the aforementioned parameters.

There are several options for the stopping criterion of BFGS algorithm in HANSO. First of all, it is possible to adjust the tolerance of a decent direction. If its norm is less than the given tolerance, the code is terminated. In this study, the default tolerance 10^{-6} is used. Another stopping criterion is that the distance of the gradient vector calculated in each step from the current iteration point is greater than the given tolerance value. The default tolerance value 10^{-4} is used in this

study again. Other stopping criteria are related to change of function values, the magnitude of the current iteration point and CPU time, but in this study numerical results have been obtained without any restrictions on them.

2.2. Gradient Sampling Algorithm (GSA)

Gradient sampling idea was used in [13,14] for the first time. Later, the gradient sampling method was used to approximate the Clarke subdifferential for locally Lipschitz functions in [15] and it was improved for non-smooth non-convex problems in [16], which is used in HANSO Software. Later, other versions of gradient sampling methods for some special optimization problems was developed such as [17–19].

GSA is intended for non-convex and locally Lipschitz functions that are differentiable almost everywhere, in other words, they are not differentiable on a set of the measure zero, so the subgradient at a randomly selected point is uniquely determined as the gradient at that point. Therefore, in GSA, gradients are computed on a set of randomly generated nearby points at current iteration. Consequently, by using gradient sampling, a local information of the function is obtained and the quadratic subproblem is formed. The ϵ-steepest descent direction is constructed by solving this quadratic subproblem, where ϵ is the sample radii.

The stopping criterion of GSA in HANSO is on descent directions. If the norm of the descent direction at current iteration is less then given tolerance, the algorithm is terminated. HANSO's default values 10^{-6} is used as a tolerance in this study.

3. Test Problem

The efficiency of HANSO was tested on the well-known non-smooth optimization academic test problems taken in [10]. Hanso's performance was discussed on this academic test problem according to the type of the non-smooth problems namely convex and non-convex. In Table 1, some information can be found for convex problems. n_{var} denotes the number of variables ,and f_{opt} denotes the optimal value of indicated problem.

Table 1. List of Convex Problems

Problem	n_{var}	f_{opt}
CB2	2	1,9522245
CB3	2	2
DEM	2	-3
QL	2	7,2
LQ	2	-1,4142136
Mifflin 1	2	-1
Wolfe	2	-8
Rosen_Suzuki	4	-44
Polak6	4	-44
Davidon 2	4	115,70644
Shor	5	22,600162
Wong 1	7	680,63006
Wong 2	10	24,306209
Polak 2	10	54,598150
Maxquad	10	-0,8414083
Polak 3	11	3,70348
Wong 3	20	93,90525
Watson	20	$0,14743027 \times 10^{-7}$
Maxq	20	0
Maxl	20	0
Gofflin	50	0
MXHILB	50	0
L1HILB	50	0

Similarly, some information can be found for non-convex problems In Table 2.

Table 2. List of Non-Convex Problems

Problem	n_{var}	f_{opt}
WF	2	0
SPIRAL	2	0
Rosenbrock	2	0
Crescent	2	0
Mifflin 2	2	-1
EVD52	3	3,5997193
OET5	4	$0,26359735 \times 10^{-2}$
OET6	4	$0,20160753 \times 10^{-2}$
El-Attar	6	0,5598131
Gill	10	9,7857721
Osborne 2	11	$0,48027401 \times 10^{-1}$
Steiner 2	12	16,703838
Shell Dual	15	32,348679

HANSO allows us to use the specified starting point or randomly generated starting point. However, in this study, 20 randomly generated starting points were used. Since the HANSO code was not suitable for running 20 different points, this code was modified to use these randomly generated 20 points by reading our starting point files. These numerical results are presented in the next section.

4. Numerical Results

Since it is not possible to give all of the 20 results obtained for each of the 36 test problems mentioned in the previous section, we first give the table below, which presents HANSO and BFGS solves how many problems related to the starting points successfully.

We can observe that there is no difference in the number of solving problems between HANSO and BFGS from Tables 3 and 4. On the other hand, looking at the values of the problems CB3, DEM, Polak 3, Wong 3, WF, SPIRAL, El-Attar, and Gill from Tables 3 and 4., one can observe that both HANSO and BFGS were able to solve them for some starting points. This means that the success of both software depends on the starting point. This is quite normal for non-smooth solvers. In addition, when there were 20 starting points for 36 test problems, in other words, these algorithms worked 720 times, they were successful in only %50 of these works.

Table 3. Number of convex problems solved successfully

Problem	HANSO	BFGS
CB2	20	20
CB3	14	14
DEM	1	1
QL	20	20
LQ	20	20
Mifflin 1	0	0
Wolfe	20	20
Rosen_Suzuki	0	0
Polak6	0	0
Davidon 2	20	20
Shor	20	20
Wong 1	0	0
Wong 2	20	20
Polak 2	0	0
Maxquad	20	20
Polak 3	5	5
Wong 3	10	10
Watson	0	0
Maxq	20	20
Maxl	20	20
Gofflin	20	20
MXHILB	20	20
L1HILB	20	20
TOTAL	**290**	**290**

Table 4. Number of non-convex problems solved successfully

Problem	HANSO	BFGS
WF	2	2
SPIRAL	18	18
Rosenbrock	20	20
Crescent	0	0
Mifflin 2	0	0
EVD52	20	20
OET5	0	0
OET6	0	0
El-Attar	8	8
Gill	3	3
Osborne 2	0	0
Steiner 2	0	0
Shell Dual	0	0
TOTAL	**71**	**71**

In particular, the rate of successfully solved attempts approximately is %63 and %27 for convex problems and non-convex problems respectively. This means that the number of iterations, function evaluations and gradient evaluations for these softwares should be compared for a more effective comparison. The reason that there is no comparison on CPU time in this study is HANSO software does not provide CPU time information. Of course, CPU time could be calculated by modifying the code, but it has not been done since the program ended in less than 2 seconds for both HANSO and BFGS. The reason that the program terminated such a short time is that HANSO's default maximum number of iterations is 1000, and this default number was used in the calculation.

5. Discussion by using Performance Profile

In order to compare the HANSO and BFGS effectively, as stated in the previous section, it is necessary to compare the numbers of iterations, function evaluations, and gradient evaluations obtained from numerical experiments. Because there is too much data in the numerical results, it is impossible to compare these data for each starting point of each problem. Therefore, the concept of performance profile presented to benchmark optimization software (see [20]) is used to compare HANSO and BFGS.

5.1. Performance Profile

The results are shown in Figures 1 and 2 by using the performance profiles introduced in [20]. In this section, the efficiency of the softwares are discussed in terms of performance measures,

which are iteration, function evaluation and gradient evaluation. The performance profile denoted by $\rho_s(\tau)$, where s is either software HANSO or BFGS, is a following

$$\rho_s(\tau) = \frac{\text{no. of the problems where } r_{p,s} \leq \tau}{\text{total no. of the problems}} \times 100$$
(1)

where $r_{p,s}$ denotes the performance ratio. If software s fails to solve problem p, the performance ratio is sufficiently large number (or infinity), otherwise

$$r_{p,s} = \frac{n_{p,s}}{\min\{n_{p,s}|\text{s is either HANSO or BFGS}\}}$$
(2)

where $n_{p,s}$ is the number of the performance measure for the software s for the problem p. In other word, $n_{p,s}$ is the number of the iteration (or function evaluation, gradient evaluation) for the software s for the problem p according to performance measure. One can easily observe from Equation 2 that the performance ratio is 1 for at least one of these two softwares and $r_{p,s} \geq 1$. If the value of $n_{p,s}$ in both softwares are equal, $r_{p,s}$ become 1 for both.

For both software HANSO and BFGS in terms of each of three aforementioned performance measures, graphs of the performance profiles $\rho_s(\tau)$ are given in Figures 1 and 2.

In the performance profiles, the value of $\rho_s(1)$ gives the percentage of test problems for which software s is the best. In other words, it uses the least performance measures, for example, least iteration number, function evaluations or gradient evaluations. If both software have the same values, both are considered the best.

The value of $\rho_s(\tau)$ at the rightmost abscissa gives the percentage of test problems that the corresponding solver can solve successfully, that is, the reliability of the solver. Moreover, the relative efficiency of each software can be directly seen from the performance profiles. The corresponding software of the higher curve is the better.

5.2. Discussion on Convex Problems

Performance profile graphs of the numerical results obtained by using 20 different starting points with 23 Convex problems given in Table 1 are given in Figure 1. Using 20 different starting points on 23 convex problems, 460 results were obtained, which means that the total number of problems in the denominator of the performance profile function which is Equation 1 is 460.

When looking at the performance profile graphs according to 3 different performance measures in Figure 1, we can say that HANSO is more reliable than BFGS in terms of these three measures, since the performance profile graph of HANSO (shortly the graph of HANSO) is above from the performance profile graph of BFGS (shortly the graph of BFGS).
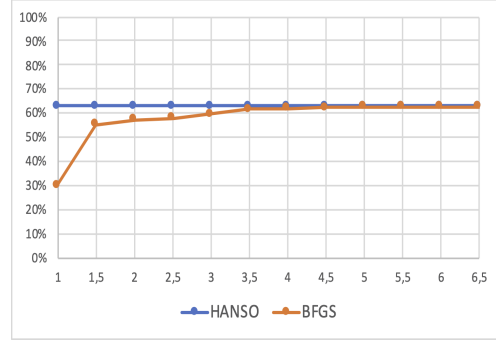
(a) Number of Iterations
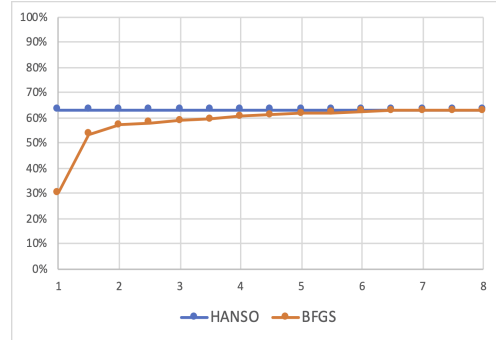
(b) Number of Function Evaluations
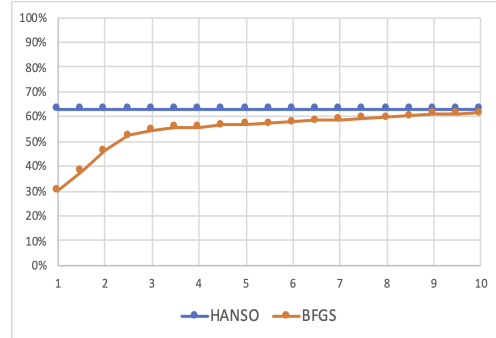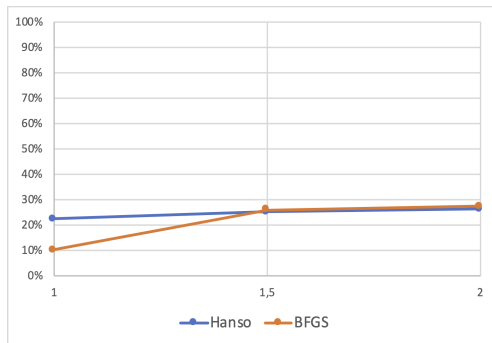

(c) Number of Gradient Evaluations

Figure 1. Performance Profiles for Convex Problems

On the other hand, it is possible to observe that both softwares have successfully solved an equal number of problems, which is %63. When we look at the $rho_s(1)$ value for both softwares, we can see that HANSO has the least number of iterations, functions, and gradients evaluations in all successfully solved problems. Similarly, when we look at the value of $rho_s(1)$ for BFGS, it can be said that BFGS has the least numbers in %30 of the successfully solved problems. This means that HANSO and BFGS have the same values for

all of these %30 problems. So there is no difference between HANSO and BFGS for this %30 problem solved successfully. One can observe that the graph of HANSO in these three graphs is a straight line. This means that for all convex problems, HANSO either has the same evaluation numbers as the BFGS or better.
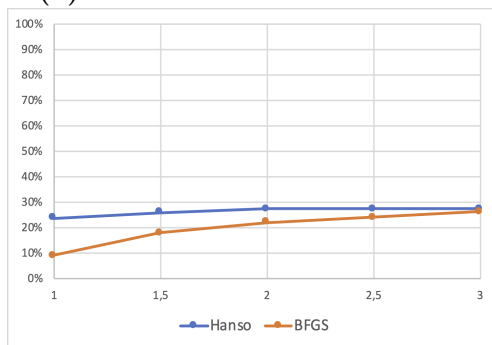
5.3. Discussion on Non-convex Problems

It can be said that the total number of problems in the performance profile function is 260 with 13 different problems for the non-convex case.

(a) Number of Iterations
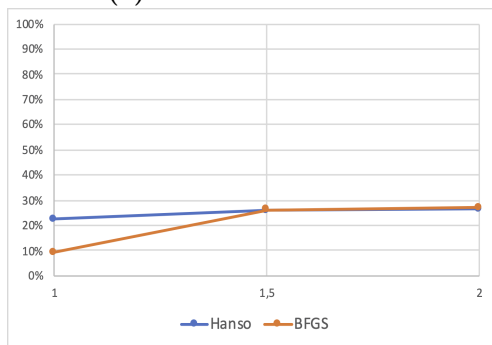
(b) Number of Function Evaluations

(c) Number of Gradient Evaluations

Figure 2. Performance Profiles for Non-convex Problems

When making a similar discussion for non-convex problems as It has done for convex problems, we will need to look at the graphs given in Figure 2. We can again say that HANSO is more reliable than BFGS for non-convex problems. However, this time the percentage %27 of successfully

solving problems for both softwares is very low. Apart from this remarkable percentage of solving success, we see that the graph of HANSO is not a straight line. The reason for this is that some problems are solved by the BFGS by making less iteration, function and gradient evaluation.

On the other hand, the meaning of the abscissa value 2 in Figures 2a and 2b is that the iteration and function evaluation numbers for these two softwares are at most 2 times higher compared to the other. This shows that there is not a huge improvement between HANSO and BFGS in terms of either performance measure. Similarly, this value is 3 in Figure 2c and the number of gradient evaluations has decreased by the factor 3 at most.

6. Conclusion

In this study, some information about HANSO software developed by Michael Overton and coded in MATLAB is presented. The parameters that HANSO uses in the line search algorithm are given and the stopping criteria of the algorithm to find the minimum value are specified. Afterwards, numerical results were obtained to compare HANSO and BFGS softwares by using some academic test problems. According to all these numerical results, it can be said that HANSO, which is a hybrid method, does not give very different results from BFGS in terms of accuracy, but it reduces the number of iterations and calculations efforts to some extent. On the other hand, when we look at the results obtained with different starting points, it can be said that for many problems, both HANSO and BFGS are sensitive to the starting point, that is, neither of them is robust. Nevertheless, this free software can be used for academic studies. Of course, when using this software, it would be more appropriate to use more than one starting point instead of one starting point for academic study. Finally, there is no need to use HANSO and BFGS separately to get results. The results obtained with only HANSO are sufficient.

It will be very useful to compare HANSO with other non-smooth solvers in the literature to shed light on academic studies. Another point not included in this study is how HANSO will give results when it comes to large-scale problems. However, this study demonstrated how HANSO software differs from BFGS by numerical calculations and discussion.

Acknowledgments

References

[1] Outrata, J., Kocvara, M., & Zowe, J. (2013). *Nonsmooth approach to optimization problems with equilibrium constraints: theory, applications and numerical results* (Vol. 28). Springer Science & Business Media.

[2] Özmen, A. & Weber, G-W. (2012). Robust conic generalized partial linear models using rcmars method-a robustification of cgplm. In *AIP Conference Proceedings*, vol.1499, 337–343.

[3] Özmen, A., Weber, G-W., Batmaz, İ. & Kropat, E. (2011). Rcmars: Robustification of cmars with different scenarios under polyhedral uncertainty set. *Communications in Nonlinear Science and Numerical Simulation*, 16(12), 4780–4787.

[4] Mistakidis, E.S. & Stavroulakis, G.E. (2013). *Nonconvex optimization in mechanics: algorithms, heuristics and engineering applications by the FEM*, volume 21. Springer Science & Business Media.

[5] Weber, G-W., Alparslan-Gök, Z.S. & Öyler, B.S. (2009). A new mathematical approach in environmental and life sciences: gene–environment networks and their dynamics. *Environmental Modeling & Assessment*, 14(2), 267–288.

[6] Weber, G-W., Tezel, A., Taylan, P., Soyler, A. & Çetin, M. (2008). Mathematical contributions to dynamics and optimization of gene-environment networks. *Optimization*, 57(2), 353–377.

[7] Bagirov, A.M., Karmitsa, N., & Taheri, S. (2020). *Partitional Clustering via Nonsmooth Optimization: Clustering via Optimization*. Springer Nature.

[8] Demyanov, V.F., Bagirov, A.M. & Rubinov, A.M. (2002). A method of truncated codifferential with application to some problems of cluster analysis. *Journal of Global Optimization*, 23(1), 63–80.

[9] Clarke, F.H., Ledyaev, Y.S., Stern, R.J. & Wolenski, P.R. (2008). *Nonsmooth analysis and control theory*, volume 178. Springer Science & Business Media.

[10] Bagirov, A.M., Karmitsa, N. & Mäkelä, M.M. (2014). *Introduction to Nonsmooth Optimization*. Theory, Practice and Software. Springer, Cham.

[11] Overton, M.L. Hanso: Hybrid algorithm for non-smooth optimization. https://cs.nyu.edu/faculty/overton/software/hanso/index.html. Accessed: 2020-08-15.

[12] Lewis, A.S. & Overton, M.L. (2013). Nonsmooth optimization via quasi-newton methods. *Mathematical Programming*, 141(1-2), 135–163.

[13] Ermoliev, Y.M. (1982). Methods of nondifferentiable and stochastic optimization and their applications. In *Progress in nondifferentiable optimization*, volume 8 of *IIASA Collaborative Proc. Ser. CP-82*, pages 5–27. International Institute for Applied Systems Analysis, Laxenburg.

[14] Shor, N.Z. (1985). *Minimization methods for nondifferentiable functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin. Translated from the Russian by K. C. Kiwiel and A. Ruszczyński.

[15] Burke, J.V., Lewis, A.S. & Overton, M.L. (2002). Approximating subdifferentials by random sampling of gradients. *Mathematics of Operations Research*, 27(3), 567–584.

[16] Burke, J.V., Lewis, A.S. & Overton, M.L. (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM Journal on Optimization*, 15(3), 751–779.

[17] Burke, J.V., Henrion, D., Lewis, A.S. & Overton, M.L. (2006). Stabilization via nonsmooth, nonconvex optimization. *Institute of Electrical and Electronics Engineers. Transactions on Automatic Control*, 51(11), 1760–1769.

[18] Burke, J.V., Lewis, A.S. & Overton, M.L. (2004). Pseudospectral components and the distance to uncontrollability. *SIAM Journal on Matrix Analysis and Applications*, 26(2), 350–361 (electronic).

[19] Kiwiel, K.C. (2007). Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2), 379.

[20] Dolan, E.D. & Moré, J.J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming. A Publication of the Mathematical Programming Society*, 91(2, Ser. A), 201–213.

Ali Hakan Tor *is an assistant professor at the Department of Mathematics, Abdullah Gul University, Kayseri, Turkey. He received his PhD degree from the Middle East Technical University, Ankara, Turkey in 2013. His research interests include non-smooth optimization theory and numerical non-smooth optimization.*

🆔 *https://orcid.org/0000-0003-3193-5004*