MCSFilter Performance: a Comparison Study

Luís Monteiro[1], José Rufino[2], Andrey Romanenko[3], and Florbela P. Fernandes[2]

[1] Instituto Politécnico de Bragança, IPB, Portugal
a36788@alunos.ipb.pt
[2] CeDRI, Instituto Politécnico de Bragança, Portugal
rufino@ipb.pt, fflor@ipb.pt
[3] MORE — Laboratório Colaborativo Montanhas de Investigação, Av. Cidade de Léon 506, 5300-358 Bragança, Portugal
aromanenko@morecolab.pt

Abstract

Optimization, together with process modelling and simulation, plays an important role in the industrial context because it is capable of generating considerable income or savings for the company.

The quality of an optimization algorithm and implementation is defined by its ability to determine an acceptable optimal solution and the time to reach this solution.

The algorithm used in this work is the MCSFilter [4,5]. This method is a multilocal method able to obtain local and global minimizers of a certain problem. The method does not use derivatives in this process and employs a multistart strategy coupled with a local coordinate search filter method to treat the constraints – simple bounds and/or general constraints. The MCSFilter method is a derivative-free method that can be used to deal with problems that involve discontinuous or non-differentiable functions.This kind of problems can formally be written as this:

$$\min_{x \in \Omega} f(x) \tag{1}$$

where the feasible region is given by:

$\Omega = \{x \in \mathbb{R}^n : l \leq x \leq u\,,\ g_i(x) \leq 0, i = 1, \cdots, k\}$ where l and u are, respectively, the lower and the upper bounds and g_i are the constraint functions.

These problems arise in many real situations, such as processes systems engineering, design and control, robotics, among other [6–8].

The initial implementation of MCSFilter in MATLAB has proved to be effective in applications whose details may be found elsewhere [1,2,7]. For more details about the algorithm, please see [4].

Further improvements included the implementation of the algorithm in C that resulted in a significant speedup compared with the original MATLAB implementation and the JAVA implementation [3] .

In this work, a new improved implementation in C was developed, using the previous JAVA code as reference. The objective is to compare the time needed by both implementations in C. A set of well-known problems were used to test and compare both implementations. These problems can be found in [4], respectively, from A1 to A18, and

they have different features: they are nonlinear problems, with simple bounds, with different dimensions (from 2 to 10), and a different number of solutions – some of them are local minimizers and other global minimizers.

Table 1 shows the results using this set of problems and were obtained in a Linux virtual machine (Ubuntu 20.04 64 bits, kernel 5.13) hosted in the CeDRI cluster, with 16 virtual CPU- cores assigned from an Intel Xeon W-2195 CPU, and 16 GB of RAM. The code was compiled using gcc (version 9.4.0) with the -O2 optimization level.

Regarding the stopping conditions, the following parameters were adopted for the execution of all problems: $\alpha_{min} = 10^{-5}$ for the local coordinate search filter and $\epsilon = 10^{-2}$ for the multistart part.

In Table 1, the first column identifies the problem, $Prob$, the second shows the known number of minimizers in the literature, min_{lit}. The third and the fourth columns display average values (from 100 executions for each problem) using the first implementation in C, $imp(1)$, related to the number of obtained minimizers, min_{avg}, and the execution time needed, t_{avg}, respectively. The next two columns show the same values, but now to the second implementation, $imp(2)$. Finally, the last column displays the speedup between both implementations given by $S = \frac{t_{avg}^{imp(1)}}{t_{avg}^{imp(2)}}$, being $t_{avg}^{imp(1)}$ the values in column four and $t_{avg}^{imp(2)}$ the values in column 6, for each problem.

Table 1. Results obtained from both implementations in C.

$Prob$ [4]	min_{lit} [4]	min_{avg} $imp(1)$	$t_{avg}(s)$ $imp(1)$	min_{avg} $imp(2)$	$t_{avg}(s)$ $imp(2)$	Speedup
A1	3	2,70	0,00126	2,96	0,00019	6,55
A2	6	4,60	0,00142	5,64	0,00019	7,42
A3	4	2,94	0,00311	3,49	0,00057	5,42
A9	760	20,03	0,00633	26,17	0,00295	2,15
A10	3	1,93	0,00095	2,21	0,00015	6,41
A11	4	3,85	0,00181	4,00	0,00033	5,55
A12	4	3,85	0,00169	4,00	0,00017	10,05
A13	8	7,50	0,00464	7,94	0,00041	11,24
A14	16	14,82	0,01239	15,91	0,00124	10,01
A15	32	29,46	0,03335	31,67	0,00359	9,29
A16	64	57,97	0,08533	63,13	0,01011	8,44
A17	256	225,53	0,54340	251,87	0,07396	7,35
A18	1024	884,00	3,21691	1002,53	0,47127	6,83

As it is possible to observe, the second implementation finds more minimizers than the first implementation in C. The new MCSFilter implementation in C is faster than the first one in C. Moreover, observing the last column, it is possible to reach the same conclusion since the speedup between both implementations for this set of problems varies between 2 and 11.

The improvements and the code optimization performed in this new implementation were successful, given the obtained results and its comparison to the old ones.

At this moment, the second implementation in C appears to be better than the first. Considering some characteristics of this method, namely the multistart part, the next

step (already under development) is to adapt the second MCSFilter code and develop a new parallel version. It will also be essential to enlarge the set of problems and use larger dimension problems to test the parallel version. It is expected that the parallel version overcomes the sequential version for larger dimension problems by far.

Keywords: MCSFilter method · Optimization · C language.

References

1. Amador, A., Fernandes, F.P., Santos, L.O., Romanenko, A., Rocha, A.M.A.C.: Parameter estimation of the kinetic α-pinene isomerization model using the mcsfilter algorithm. In: Computational Science and Its Applications — ICCSA 2018, Lecture Notes in Computer Science, vol. 10961, pp. 345–358. Springer Verlag (2018)
2. Amador, A., Fernandes, F.P., Santos, L.O., Romanenko, A.: Application of mcsfilter to estimate stiction control valve parameters. AIP Conference Proceedings **1863**(1), 2700051–2700054 (2017). https://doi.org/10.1063/1.4992427
3. Araújo, L., Pacheco, M.F., Rufino, J., Fernandes, F.P.: Towards a high-performance implementation of the mcsfilter optimization algorithm. In: Pereira, A.I., Fernandes, F.P., Coelho, J.P., Teixeira, J.P., Pacheco, M.F., Alves, P., Lopes, R.P. (eds.) Optimization, Learning Algorithms and Applications. pp. 15–30. Springer International Publishing, Cham (2021)
4. Fernandes, F.P.: Programação não linear inteira mista e não convexa sem derivadas. Ph.D. thesis, Univ. of Minho, Braga, Portugal (2014)
5. Fernandes, F., Costa, M., Fernandes, E.: Multilocal Programming: A Derivative-Free Filter Multistart Algorithm. LNCS **7971**(10), 103–118 (2013)
6. Floudas, C.: Recent advances in global optimization for process synthesis, design and control: enclosure of all solutions. Computers and Chemical Engineering **963**, 963–973 (1999)
7. Seiça, J.C., Romanenko, A., Fernandes, F.P., Santos, L.O., Fernandes, N.C.P.: Parameter estimation of a pulp digester model with derivative-free optimization strategies. AIP Conference Proceedings **1863**(1), 270006 (2017). https://doi.org/10.1063/1.4992428
8. Yang, X.S.: Optimization Techniques and Applications with Examples. Wiley (2018)