Interrupted Searches in the BBMCSFilter Context for MINLP Problems

Florbela P. Fernandes*, M. Fernanda P. Costa† and Edite M.G.P. Fernandes**

*ESTiG, Polytechnic Institute of Bragança, 5301-857 Bragança, Portugal,
†Centre of Mathematics, University of Minho, 4710-057 Braga, Portugal,
**Algoritmi Research Centre, University of Minho, 4710-057 Braga, Portugal

Abstract. The BBMCSFilter method was developed to solve mixed integer nonlinear programming problems. This kind of problems have integer and continuous variables and they appear very frequently in process engineering problems. The objective of this work is to analyze the performance of the method when the coordinate searches are interrupted in the context of the multistart strategy. From the numerical experiments, we observed a reduction on the number of function evaluations and on the CPU time.

Keywords: nonconvex MINLP, branch and bound, multistart, coordinate search, filter method
PACS: 02.60.Pn

1. INTRODUCTION

A wide range of problems arising in practical applications, such as, process engineering, water, gas, energy and transportation networks [1], involve both discrete decisions and nonlinear real-world phenomena. They are modeled as mixed-integer nonlinear programming (MINLP) problems. For a review of MINLP applications see [2].

There are different strategies to solve MINLP problems. For a review of the available solvers, we refer the reader to [2]. A large number of algorithms are based on a branch and bound (BB) framework (like the BBMCSFilter algorithm). Recently, the nonconvex MINLP research area became more interesting since metaheuristics and derivative-free algorithms, like the genetic algorithm, ant colony, differential evolution, pattern search algorithms, and the multistart Hooke and Jeeves algorithm [3] have been used in MINLP. (For details see the references included in [4].)

The MINLP problem to be solved is of the form

$$\begin{array}{rl} \min & f(x,y) \\ \text{subject to} & g_j(x,y) \le 0,\ j \in J \\ & x \in X,\, y \in Y \end{array} \tag{1}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is the objective function, g_j, $j \in J = \{1, \cdots, m\}$, are the constraint functions and J is the index set of the g functions, the vector x contains the continuous variables, with $X \subset \mathbb{R}^{n_c}$ being the set of simple bounds on x, the vector y contains the integer variables, where $Y \subset \mathbb{Z}^{n_i}$ is the set of simple bounds on y, and $n = n_c + n_i$ represents the total number of variables. It is assumed that at least one of the functions is nonlinear since this study aims to focus on MINLP problems.

Many MINLP problems arise in the context of the engineering applications being the time needed to obtain the solution of the MINLP problems sometimes a crucial issue. However, the engineering applications often require the use of complex black-box modeling routines that link the design variables with the objective and constraint functions. This MINLP problems are in general NP-hard nonsmooth and nonconvex ones. In the present study we are particularly interested in solving this type of problems. In [4], the BBMCSFilter method, which relies on a BB framework and a derivative-free methodology to solve nonsmooth and nonconvex MINLP problems, is presented. This study aims to modify the BBMCSFilter method so that a reduction on the number of function evaluations and, consequently, on the CPU time, required to reach a global solution to the MINLP problems, is obtained. In the BBMCSFilter method, the continuous NLP problems that appear in the BB tree search are solved to optimality by the MCSFilter method. This is a derivative-free global method that relies on a multistart algorithm coupled with a coordinate search filter method as the local procedure [4, 5]. The MCSFilter has shown to be an efficient algorithm for finding multiple minima of nonsmooth and nonconvex NLP problems, and hence the global one [6].

The remaining part of the paper is organized as follows. In Section 2, a brief description of the BBMCSFilter method is provided and the proposed interrupted search process is presented. Section 3 reports on the computational experiments and the last section contains the conclusions.

2. BBMCSFILTER WITH INTERRUPTED SEARCHES

In the present study we focus on nonsmooth and nonconvex MINLP problems. With this kind of problems two major issues have to be addressed. One is related with the integrality of some variables and the other is concerned with the lack of smoothness and convexity of the functions. The BBMCSFilter method is capable of solving this type of problems. However, to improve its efficiency, an interrupted search process is incorporated into the local coordinate search procedure of the MCSFilter method.

2.1. The BBMCSFilter Method

A detail description of the BBMCSFilter algorithm is available in [4]. To settle the first issue pointed out above, the algorithm uses a BB paradigm. A tree search is performed in the space of the integer variables and in each node of the tree search a continuous relaxation NLP subproblem is solved. The process of branching, bounding, relaxation and elimination of nodes, as well as the update of the solution, are performed along of the tree search until there are no more nodes to be explored. The updated solution (with all integer variables taking integer values) with the lowest function value is the solution of the MINLP problem. To settle the second issue, the BBMCSFilter algorithm uses the MCSFilter solver. This is a multistart (MS) strategy that invokes the local coordinate search filter (CSFilter) method. To explore the search space more effectively, the MS algorithm sequentially and randomly generates points $x_i = l_i + \lambda(u_i - l_i)$, $i = 1, \dots, n$, where λ is a random number uniformly distributed in $[0,1]$, and applies the CSFilter method to these points to converge to the optimal solutions of the problem. To each located minimizer a region of attraction is constructed. The MS algorithm uses the information produced by each region of attraction to check if a new randomly generated point x_i is likely to be inside a region of attraction. If this is so, the local CSFilter method is not invoked and x_i is discarded. Otherwise, the CSFilter method is invoked to converge to a new minimizer x^* [6].

The CSFilter method combines the classical derivative-free coordinate search method with a filter methodology for handling the constraints of the problem. In this approach, the continuous relaxation problem is rewritten as a bi-objective optimization problem of the form: $\min_x (\theta(x), f(x))$, where $\theta(x)$ is a nonnegative continuous aggregate constraint violation function defined by $\theta(x) = \|g(x)_+\|^2 + \|(l-x)_+\|^2 + \|(x-u)_+\|^2$, with $v_+ = \max\{0, v\}$. The filter technique uses the concept of nondominance, to construct a filter that is able to accept trial approximations if they improve the constraint violation or the objective function value.

The search in the CSFilter algorithm begins with a central point, denoted by $\tilde{x}$, and computes $2n$ trial approximations $t_c^i = \tilde{x} + \alpha e_i$, using vectors $e_i \in \mathscr{D}_\oplus$, where $\mathscr{D}_\oplus$ denotes the set of $2n$ coordinate directions and $\alpha > 0$ is a step size. Then, the two criteria θ and f are evaluated for all these trial points and if one of the conditions

$$\theta(t_c^i) < (1-\gamma_\theta)\,\theta(\tilde{x}) \text{ or } f(t_c^i) \le f(\tilde{x}) - \gamma_f\,\theta(\tilde{x}),$$

holds, for fixed constants $\gamma_\theta, \gamma_f \in (0,1)$, and the trial point is acceptable by the filter, then filter is updated. Whenever the filter is updated, the entries dominated by new ones are removed from de filter.

However, when $\tilde{x}$ is nearly feasible, the trial approximation t_c^i has to satisfy only the condition $f(t_c^i) \le f(\tilde{x}) - \gamma_f\,\theta(\tilde{x})$ in order to be acceptable.

The best trial point, t_c^{best}, is selected among all nondominated trial points and will replace the current approximation. However, if it is not possible to find a non-dominated best trial approximation, even after invoking a restoration phase, the step size is reduced. The algorithm stops when α falls below a specified tolerance.

2.2. Interrupted CSFilter

In the context of the MCSFilter, the proposed interrupted version is a clever extension of the CSFilter, which avoids solving the relaxed problem through the algorithm CSFilter till optimality when the probability of converging to an already located minimizer is high. The idea consist of interrupting the coordinate searches whenever a non-dominated best trial approximation enters an ε-neighborhood of a previously located solution. This way, the CSFilter avoids convergence to previous solutions and efficiency is improved. To check if an approximation x is inside the ε-neighborhood of a minimizer x_i^*, the following strategy is followed.

Let M be the set with all the minimizers found so far. Then, the approximation x is considered to be in the ε-neighborhood of a minimizer $x_i^* \in M$, if the distance to the minimizer verifies

$$||x - x_i^*|| < \varepsilon, \qquad (2)$$

with a small constant $\varepsilon > 0$. In this case, the likelihood that the search will converge to the minimizer x_i^* is high and the local search is interrupted. If the condition (2) does not hold for all $x_i^* \in M$, then the search process continues.

3. NUMERICAL RESULTS

To analyze and compare the behavior of the new interrupted coordinate search strategy (within the BBMCSFilter algorithm) with the BBMCSFilter itself, we set all the parameters as presented in [4], and in equation (2) we set $\varepsilon = 0.1$. The new algorithm is tested using a benchmark set of problems (from the engineering field) from the MINLPLib library available online [7]. This is the same set of benchmark problems used in [4]. Similarly, each problem was solved 30 times. Table 1 shows the numerical results produced by the proposed interrupted coordinate search version of BBMCSFilter. The columns in the table show the known global optimum, f^*; the average value of the obtained objective function values, 'f_{avg}'; the standard deviation of obtained function values, 'SD'; the average number of function evaluations, 'nfe_{avg}'; the average CPU time (in seconds), 'T_{avg}'; the average number of nodes, '$Nodes_{avg}$'; the successful rate (percentage of runs that found a solution within an error of 10^{-2} of the known global optimal solution), 'SR' (%). These results are compared with the results taken from [4] and listed in Table 2.

TABLE 1. Numerical results produced by the interrupted CS version of BBMCSFilter.

Problem	f^*	f_{avg}	SD	T_{avg}	nfe_{avg}	$Nodes_{avg}$	SR
f1	2	2.001406	0.0E+00	3.3	2440	1.0	100
f2	2.124	2.124799	0.0E+00	0.6	785	1.0	100
f3	1.07654	1.077146	0.0E+00	2.0	3616	3.0	100
f4	99.245209	99.239636	0.0E+00	0.1	627	1.0	100
f5	3.557463	3.561929	2.8E-03	29.3	44224	14.5	77
f6	4.579582	4.586224	0.0E+00	17.9	30684	7.0	100
f7	-17	-16.998524	2.2E-03	9.4	2709	2.6	100
f8	-32217.4	-32217.42778	0.0E+00	2.6	17062	5.0	0
f9	7.6671801	7.667919	9.6E-04	19.9	19022	3.7	100
f10	-2.4444	-2.444444	1.0E-08	1.1	1796	5.0	100
f11	3.2361	3.236266	0.0E+00	17.4	36639	10.0	100
f12	1.125	1.125616	0.0E+00	1.0	1413	3.0	100
f13	87.5	87.502735	2.1E-03	99.3	22463	3.0	87
f14	-6.666667	-6.666392	0.0E+00	0.4	856	1.0	100
f15	-5.6848	-5.611664	4.7E-02	1199.9	214025	54.1	6
f16	2	2.000000	0.0E+00	19.5	15560	4.7	97
f17	3.4455	3.445795	0.0E+00	1.6	1969	6.0	100
f18	2.2	2.200000	0.0E+00	3.9	7056	3.7	100
f19	6.0098	6.010985	5.1E-04	14.3	29849	6.8	87
f20	-17	-16.991320	9.9E-03	39.9	22354	1.1	63
f21	-4.514202	-4.513490	0.0E+00	22.1	19385	4.0	100
f22	-13.401904	-13.401902	5.4E-04	41.9	72790	14.8	100
f23	-1.08333	-1.083304	0.0E+00	2.6	2061	1.0	100

The computed solutions are of good quality since the f_{avg} for all problems are very close to the minimum known from the literature. The values of SD are equal to zero on 14 problems and moderate on the remaining 9 problems, ranging from $4.7E-2$ to $1.0E-8$ and showing the consistency of the algorithm. If a comparison is made using the third columns of both tables, it could be stated that the results are very similar or slightly worst than the results previously obtained. On the other hand, if a comparison is based on nfe_{avg} and T_{avg}, then clearly the new strategy has positively influenced the results, i.e., the number of function evaluations has been reduced and, consequently, the CPU time required to reach a global solution almost halved.

TABLE 2. Numerical results produced by BBMCSFilter from [4].

Problem	f^*	f_{avg}	SD	T_{avg}	nfe_{avg}	$Nodes_{avg}$	SR
f1	2	2.00082	3.6E-04	10.0	3530	1.1	100
f2	2.124	2.124590	1.4E-06	1.5	1259	1.0	100
f3	1.07654	1.081640	8.1E-03	3.8	5274	3.0	87
f4	99.245209	99.239635	1.0E-07	0.2	670	1.0	100
f5	3.557463	3.560848	2.0E-03	59.3	76775	11.0	97
f6	4.579582	4.582322	9.3E-04	54.9	75413	10.7	100
f7	-17	-16.998054	2.3E-03	9.5	4296	1.8	100
f8	-32217.4	-32217.42778	0.0E00	3.2	18051	5.7	0
f9	7.6671801	7.667583	9.5E-04	30.3	28090	3.9	100
f10	-2.4444	-2.444444	0.0E00	2.4	2736	5.0	100
f11	3.2361	3.236121	8.7E-05	18.8	41635	10.0	100
f12	1.125	1.125115	2.9E-04	42.3	7770	2.8	100
f13	87.5	87.507043	1.7E-02	252.5	41852	3.0	90
f14	-6.666667	-6.666131	1.8E-04	0.5	1122	1.0	100
f15	-5.6848	-5.651952	2.6E-02	3567.4	393345	59.8	30
f16	2.000	2.000000	0.0E00	52.7	29847	4.9	100
f17	3.4455	3.445808	2.1E-04	18.2	5469	6.0	100
f18	2.2000	2.200000	0.0E00	8.6	11182	3.7	100
f19	6.0098	6.010714	6.6E-04	21.1	37132	5.3	100
f20	-17.0000	-16.994605	5.5E-03	52.5	27149	1.1	80
f21	-4.514202	-4.513448	6.8E-04	84.4	50146	4.6	100
f22	-13.401904	-13.401930	3.6E-04	57.8	84790	14.0	100
f23	-1.08333	-1.083245	5.4E-05	2.6	2458	1.0	100

4. CONCLUSIONS

Since MINLP problems may appear in real engineering applications, the time required to reach a solution is most of the time a critical issue. To address this issue, a new strategy based on interrupting the coordinate searches within the MS paradigm of the BBMCSFilter method is devised. The numerical experiments carried out until now show that the average number of function evaluations and the average CPU time required to obtain the solution have been reduced. Since the quality of the solutions has not been improved for all the tested problems, future developments will focus on using a different and more rigorous step size choice.

REFERENCES

1. S. Burer and A. Letchford, *Surveys in Operations Research and Management Science* **17**, 97–106 (2012).
2. P. Belotti, C. Kirches, S. Leyffer, J. Linderoth and A. Mahajan, *Acta Numer.* **22**, 1–131 (2013).
3. M.F.P. Costa, F.P. Fernandes, E.M.G.P. Fernandes and A.M.A.C. Rocha, *Applied Mathematical Sciences*, **8**(44), 2163–2179 (2014)
4. F.P. Fernandes, M.F.P. Costa and E.M.G.P. Fernandes, in: *ICCSA 2014 Part II*, edited by Murgante, B. et al., LNCS 8580, Elsevier, 2014, pp. 140–153.
5. F.P. Fernandes, M.F.P. Costa and E.M.G.P. Fernandes, in: *ICCSA 2012 Part III*, edited by Murgante, B. et al., LCNS 7335, Elsevier, 2012, pp. 103–118.
6. F.P. Fernandes, M.F.P. Costa and E.M.G.P. Fernandes, in: *ICCSA 2013 Part I*, edited by Murgante, B. et al., LNCS 7971, Elsevier, 2013, pp. 333–346.
7. M.R. Bussieck, A.S. Drud and A. Meeraus, *INFORMS J. Comput.* **15**(1), 1–5 (2003) URL http://www.gamsworld.org/minlp/minlplib/minlpstat.htm