2nd International Conference on Engineering Optimization
September 6-9, 2010, Lisbon, Portugal

A Deterministic-Stochastic Method for Nonconvex MINLP Problems

Florbela P. Fernandes[1], M. Fernanda P. Costa[2] and Edite M.G.P. Fernandes[3]

[1] CMAT-University of Minho, Braga & Polytechnic Institute of Bragança, Bragança, Portugal, fflor@ipb.pt

[2] Department of Mathematics and Applications, University of Minho, Braga, Portugal, mfc@math.uminho.pt

[3] Department of Production and Systems, University of Minho, Braga, Portugal, emgpf@dps.uminho.pt

Abstract
A mixed-integer programming problem is one where some of the variables must have only integer values. Although some real practical problems can be solved with mixed-integer linear methods, there are problems occurring in the engineering area that are modelled as mixed-integer nonlinear programming (MINLP) problems. When they contain nonconvex functions then they are the most difficult of all since they combine all the difficulties arising from the two sub-classes: mixed-integer linear programming and nonconvex nonlinear programming (NLP). Efficient deterministic methods for solving MINLP are clever combinations of Branch-and-Bound (B&B) and Outer-Approximations classes. When solving nonconvex NLP relaxation problems that arise in the nodes of a tree in a B&B algorithm, using local search methods, only convergence to local optimal solutions is guaranteed. Pruning criteria cannot be used to avoid an exhaustive search in the solution space. To address this issue, we propose the use of a simulated annealing algorithm to guarantee convergence, at least with probability one, to a global optimum of the nonconvex NLP relaxation problem. We present some preliminary tests with our algorithm.
Keywords: Mixed-Integer Programming, Branch-and-Bound, Stochastic Method.

1. Introduction

Many optimization problems involve discrete and continuous variables that can be modeled as mixed-integer nonlinear programming (MINLP) problems. For instance, integer variables can represent the number of workers needed to perform a certain task whereas continuous variables can denote physical values, such as pressure or temperature. This has led to a wide range of applications in the field of process systems engineering [8]. In particular, one may find applications which include gas network problems, nuclear core reloaded problems, cyclic scheduling trim-loss optimization in the paper industry, synthesis problems, layout problems [1], thermal insulation systems [2]. Other examples are efficient management of electricity transmission, contingency analysis and blackout prevention of electric power systems [16].

A mixed-integer nonlinear program formulation can be represented as:

$$\begin{array}{ll} \min & f(x,y) \\ \text{s. t.} & g_j(x,y) \leq 0,\ j \in J \\ & x \in X,\ y \in Y \end{array} \tag{1}$$

where $X \subseteq \mathbb{R}^n$ is assumed to be a convex compact set, $Y \subseteq \mathbb{Z}^p$ corresponds to a polyhedral set of integer points, $f : \mathbb{R}^{n+p} \to \mathbb{R}$ and $g : \mathbb{R}^{n+p} \to \mathbb{R}^m$ are continuously differentiable functions, J is the index set of inequality constraints, and x and y are the continuous and discrete/integer variables, respectively. If the objective function f and the constraint functions g are convex, the problem is known as convex, otherwise the problem is a non-convex MINLP [3].

Until now, significant progress has been made in the solution techniques for convex MINLP [1, 4, 8, 5, 19]. However, techniques and solvers for nonconvex MINLP problem have just started to appear in the literature [15, 17]. Since this kind of problems appears very frequently in industrial processes, it is crucial to develop solution techniques to efficiently solve nonconvex MINLP problems. This is the goal of our study: to analyze and propose a method for nonconvex MINLP problems. First, we have been considering unconstrained MINLP problems, subject only to simple bounds. Our proposal combines two strategies: a Branch-and-Bound (B&B) method to find integer solutions and the simulated annealing

search to promote convergence to global solutions of nonconvex nonlinear programming (NLP) relaxation problems. A comparison between our proposal and another B&B-type method, which relies on a deterministic local search method for NLP problem solving – a function available in the Optimization Toolbox of MATLAB^{TM} – is presented.

2. The proposed method

To solve nonconvex MINLP problems we use a B&B-type method. The relaxed NLP problem that appears at each node of the B&B tree search is solved by a heuristic. Our research focus on problems with the form:

$$\begin{array}{c} \min \\ x \in X,\, y \in Y \end{array} \quad f(x,y) \tag{2}$$

where $X = \{x \in \mathbb{R}^n : l_x \leq x \leq u_x\}$ with $l_x, u_x \in \mathbb{R}^n$ and $Y = \{y \in \mathbb{Z}^p : l_y \leq y \leq u_y\}$ with $l_y, u_y \in \mathbb{Z}^p$. Most existing methods can be classified into two categories [12]: stochastic methods and deterministic methods. Stochastic methods sample the objective function for a small number of points, with an outcome that is random. They are particularly suited for problems that possess no known structure that can be exploited, and in general do not require derivative information. In these methods, a probabilistic convergence guarantee can be provided. The simulated annealing method is an example of a point-to-point stochastic method. On the other hand, deterministic methods exploit analytical properties of the problem to generate a sequence of points converging to a global solution. They typically provide a mathematical guarantee for convergence to a minimum in a finite number of steps. The B&B method is a deterministic method.

In this paper, a new methodology to solve problem in Eq. (2) is presented. It relies on a B&B scheme and uses a simulated annealing algorithm to guarantee convergence, at least with probability one, to a global optimum of the nonconvex NLP relaxation problem (that arises in each node of a tree in the B&B algorithm). A brief description of the two strategies combined in the herein proposed method is presented below.

2.1. Branch-and-Bound method

Although B&B was originally devised for MILP (Mixed Integer Linear Program), it can be applied to mixed-integer nonlinear problems too. The reader is referred to one of the first references to nonlinear Branch-and-Bound [6] and also to MINLP problems (see [13, 14] and the references therein included).

The B&B methodology can be explained in terms of a tree-search. Initially, all integer variables are relaxed and the resulting NLP relaxation problem is solved. If all integer variables take an integer value at the solution then this solution also solves the MINLP. Usually, some integer variables take non-integer values. Next, the algorithm selects those integer variables (which take non-integer values) and branches on it. Branching generates new NLP problems by adding simple bounds respectively to the new NLP relaxation problems.

Next, one of these new NLP problems is selected and solved. If the integer variables take non-integer values, then branching is repeated, generating a B&B tree whose nodes correspond to new NLP problems. The solution of each subproblem provides a lower bound for the subproblems in the descent nodes of the tree. This process continues until the lower bound exceeds the current upper bound, the NLP subproblem is infeasible, or the solution provides integer values for the integer variables. The integer solutions (at the nodes of the tree) give upper bounds on the optimal integer solution.

This process continues until there are no more nodes to explore.

2.2. The SAHPS Method

To solve the nonconvex MINLP problem we choose an approach that combines the B&B method (described above) and a simulated annealing heuristic pattern search (SAHPS) [10], which is used to solve the nonconvex NLP relaxation problem at each node of the B&B tree search.

Simulated annealing (SA) is one of the most effective metaheuristics for continuous global optimization. The SA algorithm successively generates a trial point in a neighborhood of the current solution and determines whether or not the current solution is replaced by the trial point based on a probability depending on the difference between their function values.

The SA approach is combined with the heuristic pattern search to form the hybrid method SAHPS [10]. The SAHPS tries to get better movements through the SA acceptance procedure or by using a

heuristic pattern search (HPS) procedure. More specifically, a new exploring neighborhood search is introduced to generate a number of SA trial points. If some of these trial points can be accepted by the SA acceptance procedure, this means that the search can go further and there is no need to use a local search method. Otherwise, some iterations of the HPS method are applied to generate more local exploratory trial points.

The HPS method is based on two main ideas. First, it uses a derivative-free heuristic method to produce an approximate descent direction at the current solution – the therein denoted approximate descent direction (ADD) method. Next, uses the ADD method to design a new pattern search method called the HPS. The ADD method is recalled to obtain an approximate descent direction v at the current iterate x_k. The direction v is computed at x_k, after generating m points $\{y_i\}_{i=1}^m$ close to x_k, as:

$$v = \sum_{i=1}^{m} w_i e_i \tag{3}$$

where

$$w_i = \frac{\Delta f_i}{\sum_{j=1}^{m} |\Delta f_j|}, \quad \Delta f_i = f(y_i) - f(x_k), \quad i = 1, \ldots, m$$
$$e_i = -\frac{y_i - x_k}{\|y_i - x_k\|} \quad i = 1, \ldots, m. \tag{4}$$

If no improvement is obtained along the vector v, then this is used to prune the set D of pattern search directions to generate other exploratory moves. To check if v is a descent direction for f at x_k, a small step size $\alpha > 0$ is used, and if $f(x_k + \alpha v) < f(x_k)$ then v is considered a descent direction. In this case, the set D_k^p of positive spanning directions in $\mathbb{R}^n$ for the pattern search procedure is obtained as

$$D_k^p = \left\{ d \in D : d^{\mathsf{T}} v \geq \beta \|d\| \|v\| \right\}, \text{ for } \beta \in (-1, 1) \tag{5}$$

whereas, if v is not a descent direction, the set D_k^p is obtained from

$$D_k^p = \left\{ d \in D : d^{\mathsf{T}} v \leq -\beta \|d\| \|v\| \right\}. \tag{6}$$

In the final stage of the search, a direct search method is applied to refine the best solution obtained so far. The algorithm uses a modified version of the Nelder-Mead method. All the details about this method can be found in [10] and the algorithm can be briefly described as below.

SAHPS Algorithm

1. Initialization of the parameters

2. Repeat Global SA Search m_1 times (in Step 2.1). If more than m_{ac} out of m_1 trial points are accepted, then skip the Local HPS (in Step 2.2) and proceed to Step 3.

2.1. Global SA Search:

2.1.1. Given the iterate x_k generate a trial point x_{SA} in its neighborhood using a radius $r > 0$

2.1.2. Evaluate f on the trial point x_{SA} and test the SA acceptance procedure

2.2. Local HPS: Repeat the following procedures m_2 times:

2.2.1. ADD: Compute the direction v, using Eq. (3). If a better movement along direction v is obtained, with a certain step size Δ_k, then proceed to the next iteration of the Local HPS

2.2.2. PS: Obtain D_k^p using either Eq. (5) or Eq. (6)

2.2.3. Parameter update: if an improvement is obtained update x_{k+1}; otherwise decrease Δ_k.

3. If the number of iterations in Global SA Search is less than $2n$, then go to Step 2.

4. If a typical cooling schedule of SA is completed or the function values of two consecutive improvement trials become close or the maximum number of iterations $50n$ is exceeded then go to Step 5. Otherwise decrease the temperature, increase the number of Local HPS repetition (m_2) and decrease the radius (r) for generating SA trials points. Go to Step 2.

5. Apply the modified Nelder-Mead method to the best point.

3. Numerical Results

The proposed method was implemented in MATLAB^{TM} and uses a B&B method combined with the SAHPS method of Hedar and Fukushima [10]. The MATLAB code of SAHPS was integrated in our solver and called inside our B&B algorithm (to solve the relaxed NLP in each node of the tree). The solver is herein called BBSAHPS. The values of the parameters inside the SAHPS are the default values as described in [10].

A collection of 14 test functions was used to analyze the practical behavior of BBSAHPS. A comparison with a B&B method that uses a local search method to solve the NLP relaxation problems – *fmincon* function from the MATLAB^{TM} Optimization Toolbox – is presented. The set of test functions, with the feasible region, the minimizers and the global minimum are displayed in the appendix at the end of the paper. For each function the solver was run 100 times (using the same initial point x_0). All experiments were run on a HP 2230s computer with an Intel(R) Core(TM)2 Duo CPU P7370 2.00GHz processor and 3,00 GB of memory.

Some minor changes were made to incorporate the SAHPS method into the B&B algorithm. For instance, when solving a NLP relaxation subproblem by SAHPS method, the initial approximation x_0 is not randomly generated but instead set equal to the optimum solution of its immediately previous subproblem in the ascent node of the tree, if that solution is feasible for the NLP relaxation subproblem. Otherwise, x_0 is the projection of that solution onto the set defined by the bounds.

Table 1 summarizes the BBSAHPS results obtained for each test problem. Some runs, out of the 100 runs, converged to different global minimizers. For instance, when solving Dixon and Price (DP) function, 36 runs (out of 100) converged to one global minimizer and in 26 runs (out of 100) the solver converged to the other global minimizer. The overall percentage of successful runs – converging to a global solution with mixed-integer variable – is then 62.

In Table 1, the best objective function values "Best f^*" obtained from 100 runs is reported for each test problem. In order to show more details concerning the quality of the obtained solutions, the average "Average f^*" and the standard deviation "σ" of the best obtained function values are also reported in Table 1. Moreover, success rates of obtaining the global minimum "% success", the average numbers of CPU time "time" (in seconds), major cycles of the B&B tree "cycles" and function evaluations "f eval.", are shown in columns 2–5 of Table 1.

The results obtained by the BBSAHPS method are quite satisfactory, except for problem R_2 (Rosenbrock function) which has only a success rate of 4%. However, it was observed that, if the solution of the NLP relaxation subproblem is rounded to an approximation with four decimal digits to maintain feasibility and avoid discarding that solution, the success rate of the BBSAHPS method increases. In practice, we used this heuristic when solving problems marked with "§" in Table 1. Table 2 shows the improvement on the success rate obtained for these problems. As it can be seen, the percentage of successful runs were improved to 96% with the Rosenbrock function and to 92% with de Beale (B) function. The success rate slightly improved in the Himmelblau n^o 3 (HM_3) function.

It is noteworthy that the BBSAHPS method has a success rate superior to 85% for 8 problems as shown in Table 1. With all these functions the BBSAHPS method could successfully find the global minimum, taking into account the integrality of some variables. Further, the accuracy of the achieved solutions, in terms of "Best f^*", is very good, and the standard deviations are close to zero, meaning that the consistency of our proposed deterministic-stochastic procedure is high. The computational cost in terms of CPU time required by the proposed BBSAHPS is rather small, as shown in Table 1. The L_8 function is the one that requires a larger CPU time when compared with the other functions. We remark that the L_8 function has four variables while the others have only two.

As far as DP function is concerned, and despite the fact that the point $x^* = (1, -\sqrt{2}/2)^\mathsf{T}$ is not listed in the literature [7, 12], BBSAHPS method is able to find it, and this is indeed a global minimizer of the DP function.

Table 1: Numerical results obtained during the 100 runs and using BBSAHPS

f	% success	Average time (s)	Average cycles	Average f eval.	Best f^*	Average f^*	σ
B §	46	0.08553	5	991	6.83325×10^{-13}	4.57638×10^{-9}	6.65897×10^{-12}
Boa	99	0.16958	13	2204	0.0×10^{0}	6.58918×10^{-13}	3.66562×10^{-12}
Boo	61	0.13181	12	1131	0.0×10^{0}	5.67883×10^{-14}	3.05624×10^{-13}
DP	62						
	36	0.07681	6	744	8.08397×10^{-16}	4.17957×10^{-9}	6.60461×10^{-9}
	26	0.07592	6	733	8.78736×10^{-14}	3.61570×10^{-9}	3.66593×10^{-9}
GP	95	0.07050	6	690	3.0×10^{0}	3.0×10^{0}	1.06526×10^{-10}
f_{26}	87	0.07936	7	966	-3.52386×10^{-1}	-3.52386×10^{-1}	1.09059×10^{-9}
HM_3 §	42	0.09331	8	656	0.0×10^{0}	1.09418×10^{-12}	3.52609×10^{-12}
L_8	88	0.65901	24	2442	1.49976×10^{-32}	2.30771×10^{-14}	1.48765×10^{-13}
P §	49						
	19	0.07179	5	918	1.07746×10^{-12}	1.71264×10^{-9}	1.60441×10^{-9}
	18	0.07399	5	842	3.63046×10^{-12}	1.35319×10^{-9}	1.24148×10^{-9}
	6	0.07104	5	859	1.15036×10^{-10}	1.76213×10^{-9}	2.33087×10^{-9}
	6	0.07244	5	865	7.45027×10^{-11}	1.16198×10^{-9}	1.85512×10^{-9}
R_2 §	4	0.10677	9	951	0.0×10^{0}	4.65454×10^{-11}	8.48159×10^{-11}
ST_1	89						
	49	0.04737	4	555	-4.07462×10^{-1}	-4.07462×10^{-1}	1.66109×10^{-9}
	40	0.04630	4	562	-4.07462×10^{-1}	-4.07462×10^{-1}	2.30832×10^{-9}
ST_2	92						
	49	0.05363	5	610	-1.80587×10^{1}	-1.80587×10^{1}	1.45725×10^{-9}
	43	0.05701	4	593	-1.80587×10^{1}	-1.80587×10^{1}	2.74686×10^{-9}
ST_3	93						
	52	0.05616	4	643	-2.27766×10^{2}	-2.27766×10^{2}	3.58893×10^{-9}
	41	0.04835	3	573	-2.27766×10^{2}	-2.27766×10^{2}	4.69358×10^{-9}
T	87	0.06028	5	513	-2.0×10^{0}	-2.0×10^{0}	2.34821×10^{-11}

Table 2: Improvement in BBSAHPS using the rounded solution of the NLP relaxation subproblem

f	% success
B	92
HM_3	45
P	59
R_2	96

Finally, we compare the results of our study with those of the B&B method with the *fmincon* solver of MATLAB. The *fmincon* parameters assigned to all fourteen executions are equal and set to their default values. The initial approximation x_0 used in each function was the same as that chosen for BBSAHPS.

Table 3 summarizes the results obtained with the B&B method with the *fmincon* solver for each test problem. Table 3 reports information similar to that shown in Table 1, concerning the CPU time, B&B cycles, average number of function evaluations and the best objective function value. The last two columns are from Table 1, to allow a more expedite comparison between the two methods.

Table 3: Comparison between B&B with *fmincon* and BBSAHPS

	fmincon					
f	time(s)	cycles	f eval.	f^*	Best f^*	Average f^*
B	1.35996	5	91	2.02450×10^{-7}	6.83325×10^{-13}	4.57638×10^{-9}
Boa	0.40489	18	83	0.0×10^{0}	0.0×10^{0}	6.58918×10^{-13}
Boo	0.75533	12	101	0.0×10^{0}	0.0×10^{0}	5.67883×10^{-14}
DP	0.16860	7	30	1.0×10^{0}	8.08397×10^{-16}	4.17957×10^{-9}
GP	0.30107	12	156	3.0×10^{0}	3.0×10^{0}	3.0×10^{0}
f_{26}	2.02499	7	34	-1.52639×10^{-1}	-3.52386×10^{-1}	-3.52386×10^{-1}
HM_3	0.16777	9	80	0.0×10^{0}	0.0×10^{0}	1.09418×10^{-12}
L_8	1.16483	52	326	1.49976×10^{-32}	1.49976×10^{-32}	2.30771×10^{-14}
P	0.12354	6	37	4.91198×10^{-10}	1.07746×10^{-12}	1.71264×10^{-9}
R_2	0.21735	11	134	0.0×10^{0}	0.0×10^{0}	4.65454×10^{-11}
ST_1	0.10171	1	27	-4.07462×10^{-1}	-4.07462×10^{-1}	-4.07462×10^{-1}
ST_2	0.29644	5	109	-1.80587×10^{1}	-1.80587×10^{1}	-1.80587×10^{1}
ST_3	0.05574	1	31	-2.27766×10^{2}	-2.27766×10^{2}	-2.27766×10^{2}
T	0.04051	1	3	6.79367×10^{-1}	-2.0×10^{0}	-2.0×10^{0}

The results shown in Table 3 indicate that BBSAHPS generally outperforms the B&B with *fmincon*. We observe that the B&B with *fmincon* requires less function evaluations but, on the other hand, the BBSAHPS has higher accuracy. Further, although the BBSAHPS method invokes the objective function more often than the B&B with *fmincon*, it is less time consuming.

To illustrate the behavior of the B&B with *fmincon* when compared with the BBSAHPS method, we consider the f_{26} function. Figure 1 is a graphical representation of f_{26}, near the global minimum. B&B with *fmincon* stops at -1.52639×10^{-1} (a local minimum), while BBSAHPS converges to -3.52386×10^{-1} (the global minimum). We observed that the initial point x_0 affects the performance of the B&B with *fmincon*. A similar behavior occurred with the DP and Tsoulos (T) functions.

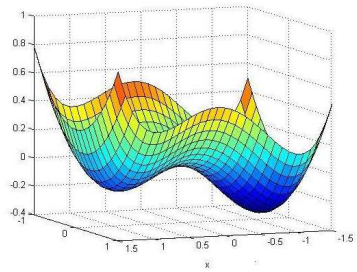

Figure 1: Graphical representation of f_{26}

4. Conclusions and future work

In this paper we have presented a deterministic-stochastic method – BBSAHPS – in which a Branch-and-Bound procedure is combined with a hybrid global search to find the minimum of MINLP problems with simple bounds. BBSAHPS method was implemented using MATLAB^{TM} and some results are shown considering 14 test functions. This method is able to find the global optimum, with integer restrictions for some variables. The performance of the BBSAHPS method is quite satisfactory.

A comparison between BBSAHPS and a local search method (based on B&B and *fmincon* – a function from MATLAB^{TM} Optimization Toolbox) was presented. The results obtained with B&B and the classical *fmincon* are worst than ours, and in some cases, it converged to a local minimum. This is due to the fact that the NLP relaxation problems – that are solved in the nodes of the B&B tree – are non-convex and *fmincon* is not able to detect the global minimum. On the other hand, BBSAHPS method was able to find the global minimum in all 14 test functions.

Numerical results indicate that BBSAHPS outperforms the B&B with *fmincon*. Future developments will be focused on efficient constraint-handling by the classical filter methodology.

Acknowledgements

This work was financially supported by the Research Centre of Mathematics of the University of Minho through the FCT Pluriannual Funding Program.

References

[**1**] K. Abhishek, S. Leyffer and J. Linderoth, FilMINT: An Outer-Approximation-Based Solver for Nonlinear Mixed Integer Programs. Technical Report ANL/MCS-P1374-0906, Mathematics and Computer Science Division, Argonne National Laboratory, 2006.

[**2**] K. Abhishek, S. Leyffer and J.T. Linderoth, Modeling without categorical variables: a mixed-integer nonlinear program for the optimization of thermal insulation systems, *Optimization and Engineering*, 11, 185–212, 2010.

[**3**] L. Biegler and I. Grossmann, Retrospective on optimization, *Computers and Chemical Engineering*, 28, 1169-1192, 2004.

[**4**] P. Bonami, L. Biegler, A. Conn, G. Cornuejols, I. Grossmann, C. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya and A. Wachter, An algorihmic framework for convex mixed integer nonlinear programs, Technical Report RC23771, IBM Research Report, 2005.

[**5**] R. H. Byrd, J. Nocedal, and R. A. Waltz,(2006). KNITRO: An Integrated Package for Nonlinear Optimization in Large-Scale Nonlinear Optimization, G. di Pillo and M. Roma (Eds.), 35-59, Springer-Verlag, 2006.

[**6**] R.J. Dakin. A tree search algorithm for mixed integer programming problems, *Computer Journal*, 8, 250-255, 1965.

[**7**] M.J. Hirsch, P.M. Pardalos, M.G.C. Resende, Speeding up continuous GRASP, *European Journal of Operational Research*, 205, 507-521, 2010.

[**8**] I. Grossmann, Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques, *Optimization and Engineering*, 3, 227-252, 2002.

[**9**] C.A. Floudas, P.M. Pardalos, C.S. Adjiman, W.R. Esposito Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer and C.A. Schweiger, *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers, Dordrecht/Boston/London, 1999.

[**10**] A. Hedar and M. Fukushima, Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization, *Optimization Methods and Software*, 19, 291-308, 2004.

[**12**] J. Lee, A novel three-phase trajectory informed search methodology for global optimization, *J Glob Optim*, 38, 6177, 2007.

[**13**] S. Leyffer, Deterministic Methods for Mixed Integer Nonlinear Programming. PhD Thesis, University of Dundee, United Kingdom, 1993.

[**14**] S. Leyffer, Integrating SQP and branch and bound for mixed integer nonlinear programming. *Computational Optimization and Applications*, 18, 295-309, 2001.

[**15**] S. Leyffer, A. Sartenaer, and E. Wanufelle, Branch-and-Refine for Mixed Integer Nonconvex Global Optimization, Preprint ANL/MCS-P1547-0908, Mathematics and Computer Science Division, Argonne National Laboratory, 2008.

[**16**] S. Leyffer, J. Linderoth, J . Luedtke, A. Miller and T. Munson, Applications and Algorithms for Mixed Integer Nonlinear Programming, Preprint ANL/MCS-P1630-0509, Mathematics and Computer Science Division, Argonne National Laboratory, 2009.

[**17**] I. Nowak and S Vigerske, LaGO: a (heuristic) branch and cut algorithm for nonconvex MINLPs, *Central European Journal of Operations Research*, 16, 127-138, 2008.

[**18**] R. Storn and K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11, 341-359, 1997.

[**19**] M.H. Van der Vlerk, Convex approximations for a class of mixed-integer recourse models, *Annals of Operations Research*, 177, 139–150, 2010.

Appendix - List of test functions

The collection of 14 test functions used in this study are listed below [7, 9, 10, 12, 18].

- **Beale function** (B)
  - Number of variables: $n = 2$, x_1 integer.
  - Definition: $B(x_1, x_2) = (1.5 - x_1(1 - x_2))^2 + (2.25 - x_1(1 - x_2^2))^2 + (2.625 - x_1(1 - x_2^3))^2$.
  - Feasible region: $x_1 \in [-5, 5]\,,\ \ x_2 \in [-4.5, 4.5]$.
  - Global minimum: $x^* = \left(3, \frac{1}{2}\right)^\mathsf{T}$ and $B(x_1, x_2) = 0$.
- **Bohachevsky function** (Boa)
  - Number of variables: $n = 2$, all integer.
  - Definition: $B_{oa}(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$.
  - Feasible region: $x \in [-100, 100]^2$.
  - Global minimum: $x^* = (0, 0)^\mathsf{T}$ and $B_{oa}(x^*) = 0$.
- **Booth function** (Boo)
  - Number of variables: $n = 2$, all integer.
  - Definition: $B_{oo}(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$.
  - Feasible region: $x \in [-10, 10]^2$.
  - Global minimum: $x^* = (1, 3)^\mathsf{T}$ and $B_{oo}(x^*) = 0$.
- **Dixon and Price function** (DP)
  - Number of variables: $n = 2$, x_1 integer.
  - Definition: $DP(x_1, x_2) = 2(2x_2^2 - x_1)^2 + (x_1 - 1)^2$.
  - Feasible region: $x \in [-10, 10]^2$.
  - Global minima: $x^* = \left(1, -\frac{\sqrt{2}}{2}\right)^\mathsf{T}, \left(1, \frac{\sqrt{2}}{2}\right)^\mathsf{T}$ and $DP(x^*) = 0$.
- **Goldstein and Price function** (GP)
  - Number of variables: $n = 2$, all integer.
  - Definition:

$$\begin{aligned} GP(x_1, x_2) = \quad & (1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2))\times \\ & \times(30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)) \end{aligned}$$

  - Feasible region: $x \in [-2, 2]^2$.
  - Global minimum: $x^* = (0, -1)^\mathsf{T}$ and $G(x^*) = 3$.

- f_{26} **function** (f_{26})
  - Number of variables: $n = 2$, x_2 integer.
  - Definition: $f_{26}(x_1, x_2) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$.
  - Feasible region: $x \in [-10, 10]^2$
  - Global minimum: $x^* = (-1.0466805696, 0)^\mathsf{T}$ and $G(x^*) = -3.5239 \times 10^{-1}$.
- **Himmelblau n^o 3 function** (HM_3)
  - Number of variables: $n = 2$, x_1 integer.
  - Definition: $HM_3(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$.
  - Feasible region: $x \in [-2, 4]^2$
  - Global minimum: $x^* = (3, 2)^\mathsf{T}$ and $HM_3(x^*) = 0$.
- **Levy n^o 8 function** (L_8)
  - Number of variables: $n = 4$, all integer.
  - Definition:

    $$L_8(x) = \sin^2(\pi y_1) + \sum_{i=1}^{3} \left((y_i - 1)^2 \left(1 + 10\sin^2(\pi y_{i+1})\right)\right) + (y_4 - 1)^2 \left(1 + \sin^2(2\pi y_4)\right)$$

    where $y_i = 1 + \dfrac{x_i - 1}{4}$, for $i = 1, \cdots, 4$.
  - Feasible region: $x \in [-10, 10]^4$
  - Global minimum: $x^* = (1, 1, 1, 1)^\mathsf{T}$ and $L_8(x^*) = 0$.
- **Parsopoulos function** (P)
  - Number of variables: $n = 2$, x_2 integer.
  - Definition: $P(x) = \cos^2(x_1) + \sin^2(x_2)$.
  - Feasible region: $x \in [-5, 5]^2$
  - Global minima: $x^* = \left(\dfrac{\pi}{2}, 0\right)^\mathsf{T}, \left(-\dfrac{\pi}{2}, 0\right)^\mathsf{T}, \left(\dfrac{3\pi}{2}, 0\right)^\mathsf{T}, \left(-\dfrac{3\pi}{2}, 0\right)^\mathsf{T}$ and $P(x^*) = 0$.
- **Rosenbrock function** (R_2)
  - Number of variables: $n = 2$, all integer.
  - Definition: $R_2(x) = (-1 + x_1)^2 + 100(x_2 - x_1^2)^2$.
  - Feasible region: $x \in [-5, 10]^2$
  - Global minimum: $x^* = (1, 1)^\mathsf{T}$ and $R_2(x^*) = 0$.

- **Storn function** (ST_m)
  - Number of variables: $n = 2$, x_1 integer.
  - Definition: $ST_m(x) = 10^m x_1^2 + x_2^2 - (x_1^2 + x_2^2)^2 + 10^{-m}(x_1^2 + x_2^2)^4$.

  ST_1 Feasible region: $x \in [-2, 2]^2$

  Global minima: $x^* = (0, 1.38695)^\mathsf{T}$, $(0, -1.38695)^\mathsf{T}$ and $ST_1(x^*) = -4.07462 \times 10^{-1}$.

  ST_2 Feasible region: $x \in [-4, 4]^2$

  Global minima: $x^* = (0, 2.60891)^\mathsf{T}$, $(0, -2.60891)^\mathsf{T}$ and $ST_2(x^*) = -1.80587 \times 10^1$.

ST_3 Feasible region: $x \in [-8, 8]^2$

Global minima: $x^* = (0, 4.70174)^\mathsf{T}$, $(0, -4.70174)^\mathsf{T}$ and $ST_2(x^*) = -2.27766 \times 10^2$.

- **Tsoulos function** (T)
  - Number of variables: $n = 2$, all integer.
  - Definition: $T(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$.
  - Feasible region: $x \in [-1, 1]^2$
  - Global minimum: $x^* = (0, 0)^\mathsf{T}$ and $T(x^*) = -2$.