Experiments with a Feasibility Pump approach for nonconvex MINLPs

Claudia D'Ambrosio[1], Antonio Frangioni[2],
Leo Liberti[3], and Andrea Lodi[1]

[1] DEIS, University of Bologna, Italy
{c.dambrosio; andrea.lodi}@unibo.it
[2] Dipartimento di Informatica, University of Pisa, Italy
frangio@di.unipi.it
[3] LIX, École Polytechnique, France
liberti@lix.polytechnique.fr

Abstract. We present a new Feasibility Pump algorithm tailored for nonconvex Mixed Integer Nonlinear Programming problems. Differences with the previously proposed Feasibility Pump algorithms and difficulties arising from nonconvexities in the models are extensively discussed. The main methodological innovations of this variant are: (a) the first subproblem is a nonconvex continuous Nonlinear Program, which is solved using global optimization techniques; (b) the solution method for the second subproblem is complemented by a tabu list. We exhibit computational results showing the good performance of the algorithm on instances taken from the MINLPLib.

Key words: Mixed-Integer Nonlinear Programming, nonconvex, heuristic method, experiments

1 Introduction

Heuristic algorithms have always played a fundamental role in optimization, both as independent tools and as part of general-purpose solvers. Heuristics can be classified into two broad categories: those which are based on a specific problem structure (e.g., heuristics for Set Covering, or Knapsack, or Quadratic Assignment problems) and those which target a large class of problems, such as Mixed Integer Linear Programming (MILP), or Mixed Integer Nonlinear Programming (MINLP). Far fewer heuristics (including the present one) belong to the second class with respect to the first one because of the inherent difficulty of devising general-purpose methods. In the rest of the paper we focus on algorithms belonging to the second class.

Starting from MILPs, different kinds of heuristics have been proposed: their aim is finding a good feasible solution rapidly or improving the best solution found so far. Within a MILP solver context, both types are used. Examples are rounding heuristics, metaheuristics (e.g. [1]), Feasibility Pump (FP) [2–4], Local Branching [5] and Relaxation Induced Neighborhoods Search [6]. Since the early 1990's, MINLP has attracted rising interest from the operations research and the chemical engineering communities. Typically, MINLP solution techniques complement continuous Nonlinear Programming (NLP) algorithms with combinatorial-type searches. Chemical plant design and operations are amongst the early applications of this field. Special attention has been devoted to convex MINLPs, a class of MINLP problems whose objective function and constraints are convex (thus, any local optimum of the continuous relaxation is also its global optimum). Furthermore, standard linearization inequalities such as Outer Approximation (OA) cuts [7] are valid. OA cuts linearly approximate each nonlinear convex constraint, say $f(x) \leq 0$, at a given point $\bar{x}$:

$$f(\bar{x}) + \nabla f(\bar{x})^T \, (x - \bar{x}) \leq 0. \tag{1}$$

It is easy to show that (1) does not cut off any feasible point of the original convex MINLP. Heuristics have been proposed recently also for convex MINLPs. Basically the ideas originally tailored on MILP problems have been extended to convex MINLPs, for example, Feasibility Pump [8–10] and diving heuristics [10].

This paper proposes a heuristic algorithm for nonconvex MINLPs. These problems are in general very difficult to solve to optimality and, often, also finding a feasible solution is practically difficult (besides being NP-hard in theory, since they generalize NLP feasibility [11]). For this reason, heuristic algorithms are a fundamental part of any solution process. General-purpose nonconvex MINLP heuristics proposed so far are, for example, Variable Neighborhood Search [12] and Local Branching [13]. All existing exact convex MINLP methods [7, 8, 14–19] can be heuristically deployed on nonconvex MINLPs. This field, however, is still relatively young and current heuristics leave a lot of room for improvement.

We already mentioned FP for 0-1 MILP, introduced by Fischetti et al. [2]. The algorithm has been extended to general integer variables by Bertacco et al. [3] and improved with respect to solution quality by Achterberg and Berthold [4]. The idea is to iteratively solve subproblems of the original (difficult) problem with the aim of "pumping" the feasibility in the solution. More precisely, Feasibility Pump iteratively solves the continous relaxation of the problem trying to minimize the distance to a target (infeasible) integer solution, then rounding the fractional solution obtained to become the next integer target. Few years later a similar technique applied to convex MINLPs was proposed by Bonami et al. [8]. In that case the subproblems are a convex NLP and a MILP. The authors also prove the convergence of the algorithm and extend the same result to MINLP problems with nonconvex constraints, defining, however, a convex feasible region. More recently Bonami and Gonçalves [10] proposed a more efficient version in which the MILP solution process is replaced by a rounding phase similar to that originally proposed by Fischetti et al. [2] for MILPs. Finally, an enhancement for the MILP case has been recently studied by Fischetti and Salvagnin [20] by using domain propagation during rounding.

In this paper, we propose a FP algorithm for general nonconvex MINLPs. The remainder of the paper is organized as follows. In Section 2 we present the structure of the algorithm. Details on implementation issues are given in Section 3. In Section 4 we present computational results on MINLPLib instances. Section 5 concludes the paper.

2 The Algorithm

We address the following nonconvex MINLP:

$$
\begin{aligned}
(P) \qquad\qquad \min f(x,y) & \qquad (2)\\
g(x,y) \le 0 & \qquad (3)\\
x \in X \cap \mathbb{Z}^n & \qquad (4)\\
y \in Y, & \qquad (5)
\end{aligned}
$$

where X and Y are two polyhedra of appropriate dimension (possibly including variable bounds), $f : \mathbb{R}^{n+p} \to \mathbb{R}$ is *convex*, but $g : \mathbb{R}^{n+p} \to \mathbb{R}^m$ is *nonconvex*. We will denote by

$$\mathcal{P} = \{\ (x,y) \mid g(x,y) \le 0 \wedge x \in X \wedge y \in Y\ \} \subseteq \mathbb{R}^{n+p}$$

the (nonconvex) feasible region of the continuous relaxation of the problem, by $\mathcal{X}$ the set $\{1,\dots,n\}$ and by $\mathcal{Y}$ the set $\{1,\dots,p\}$. We will also denote by $\mathcal{N} \subseteq \{1,\dots,m\}$ the subset of (indices of) nonconvex constraints, so that $\mathcal{C} = \{1,\dots,m\} \setminus \mathcal{N}$ is the set of (indices of) convex constraints. Note that the convexity assumption on the objective function f does not involve a loss of generality: one can always introduce an additional variable v to be minimized, and add the $(m+1)-$th constraint $f(x,y)-v \le 0$ to deal with the case where f is nonconvex.

Problem (P) presents two sources of nonconvexities:

Algorithm 1 The general scheme of Feasibility Pump

```
1: $i = 0$;
2: initialize $(\hat{x}^0, \hat{y}^0)$ and $(\bar{x}^0, \bar{y}^0)$;
3: while $((\hat{x}^i, \hat{y}^i) \neq (\bar{x}^i, \bar{y}^i)$ and CPU time $<$ limit) do
4:     increase $i$;
5:     solve $(P1)$ (minimize distance to $(\hat{x}^{i-1}, \hat{y}^{i-1})$ subject to $(x, y) \in \mathcal{P}$) to yield
       $(\bar{x}^i, \bar{y}^i)$;
6:     solve $(P2)$ (minimize distance to $(\bar{x}^i, \bar{y}^i)$ subject to $(x, y) \in (X \cap \mathbb{Z}^n) \times Y$) to
       yield $(\hat{x}^i, \hat{y}^i)$;
7: end while
```

1. integrality requirements on x variables;
2. constraints $g_j(x, y) \leq 0$ with $j \in \mathcal{N}$, defining a nonconvex feasible region.

The basic idea of FP is to decompose the original problem in two easier subproblems. One, called $(P1)$, is obtained by relaxing the integrality requirements; the other, called $(P2)$, by relaxing the nonlinear constraints. Both problems are solved at each iteration, yielding a pair of solutions $(\bar{x}, \bar{y})$ and $(\hat{x}, \hat{y})$ respectively. The aim of the algorithm is to make the trajectories of the two solutions converge to a unique point, satisfying all the constraints and the integrality requirements (see Algorithm 1).

In the next two sections we discuss the general framework by specializing it to our context, i.e., the nonconvex MINLP case.

2.1 Subproblem $(P1)$

At iteration i the subproblem $(P1)$ is denoted by $(P1)^i$ and has the form

$$\min_{x \in X,\ y \in Y} ||x - \hat{x}^{i-1}|| \tag{6}$$

$$g(x, y) \leq 0, \tag{7}$$

where $(\hat{x}^{i-1},\ \hat{y}^{i-1})$ is the solution of subproblem $(P2)^{i-1}$ which satisfies the integrality requirements on x (see Section 2.2) and in (6) we used the 2-norm. Either (a) the globally optimal objective function value of $(P1)^i$ is 0, implying a feasible solution of (P) with $x = \hat{x}^{i-1}$; or (b) no feasible point of (P) exists with $x = \hat{x}^{i-1}$. Unfortunately, solving $(P1)^i$ to global optimality is too computationally expensive to be considered as a viable option. Using a local NLP solver to solve $(P1)^i$ is not a viable alternative either: if $(\bar{x}^i, \bar{y}^i)$ is a local optimum of $(P1)^i$ with value > 0, (b) no longer holds and there might exist a feasible solution of (P) with $x = \hat{x}^{i-1}$. We would then erroneously consider the solution with $x = \hat{x}^{i-1}$ as infeasible and continue iterating. We therefore propose the following strategy:

1. Solve $(P1)^i$ using a simple multistart heuristic [21] to maximize chances of finding the global optimum.
2. If no solution yielding 0 as objective function value was found, solve the following problem denoted as $(P1fix)^i$:
$$\min f(\hat{x}^i, y) \tag{8}$$
$$g(\hat{x}^i, y) \leq 0. \tag{9}$$
Problem $(P1fix)^i$ differs with respect to problem $(P1)^i$ because the objective (6) (constant if x is fixed to $\hat{x}^i$ like in this case) is replaced by the original objective f, and it is solved in the attempt of finding a MINLP feasible solution with $\hat{x}^i$ values of x variables.

The solution proposed does not give any guarantee that the global optimum will be found and, consequentely, that no feasible solution of (P) will be ignored, but, since we propose a heuristic algorithm, we consider this simplification as a good compromise. Our computational experiments show that for some classes of nonconvex MINLP the approach is sound. Consider, for example, a problem (P) that, once variables x are fixed, is convex: in this case solving problem $(P1fix)^i$ would provide the global optimum.

2.2 Subproblem $(P2)$

At iteration i subproblem $(P2)$, denoted as $(P2)^i$, has the form
$$\min ||x - \bar{x}^i|| \tag{10}$$
$$g_j(\bar{x}^k, \bar{y}^k) + \nabla g_j(\bar{x}^k, \bar{y}^k)^T \begin{pmatrix} x - \bar{x}^k \\ y - \bar{y}^k \end{pmatrix} \leq 0 \quad k = 1, \ldots, i; j \in M^k \tag{11}$$
$$x \in X \cap \mathbb{Z}^n \tag{12}$$
$$y \in Y, \tag{13}$$
where $(\bar{x}^i, \bar{y}^i)$ is the solution of subproblem $(P1)^i$ and $M^k \subseteq \{1, \ldots, m\}$ is the set (possibly empty) of (indices of) constraints from which OA cuts are generated at point $(\bar{x}^k, \bar{y}^k)$. We remark that M^k will most usually be a proper subset of $\{1, \ldots, m\}$ because, when nonconvex constraints are involved, not all the possible OA cuts generated are "safe", i.e., do not cut off feasible solutions of (P) (see Figure 1). We remark that the OA cut generated from a convex constraint $g(z)$ is valid for (P). In order to model subproblem $(P2)^i$ as a MILP, in (10) we use the 1-norm.

Generation of OA cuts involves essentially two issues, one stemming from a practical consideration, the other from a theoretical point of view.

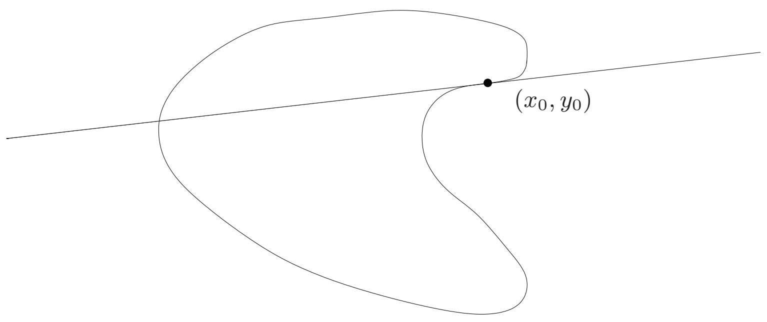

Fig. 1. Outer Approximation constraint cutting off part of the nonconvex feasible region.

The first issue is that discriminating convex and nonconvex constraints is a hard task in practice. We will describe in Section 4 how we simplified this step on the implementation side. The second issue is that OA cuts play a fundamental role on the convergence of the algorithm for convex MINLPs (see Bonami et al. [8]): if at one iteration no OA cut can be added, the algorithm may cycle. However, in the nonconvex case, even if an OA cut is added, there is no guarantee that it would cut off the solution of the previous iteration, as shown by the following example.

Example 2.1. In Figure 2 a nonconvex feasible region and its current linear approximation are depicted. Let us consider $\bar{x}$ being the current solution of subproblem $(P1)$. In this case, only one Outer Approximation cut can be generated, the one corresponding to convex constraint γ. However, this OA cut does not cut off solution $\hat{x}$, i.e., the solution of $(P2)$ at the previous iteration. In this example, the FP would not immediately cycle, because $\hat{x}$ is not the solution of $(P2)$ which is closest to $\bar{x}$. This shows that there is a distinction between cutting off and cycling. However, in the long(er) term not cutting off previously generated integer solutions might lead to cycling. $\Box$

A possible solution to this issue is using a tabu list for the last solutions obtained from $(P2)$: the MILP solver will discard integer feasible solutions in the tabu list. The integer part of the solution is compared with the one of the solutions in the tabu list and it is accepted only if its integer part is different from that of the forbidden solutions. Otherwise it is discarded: this prevents algorithmic cycling as long as the cycle length is shorter than the tabu list length. This simple idea works with both

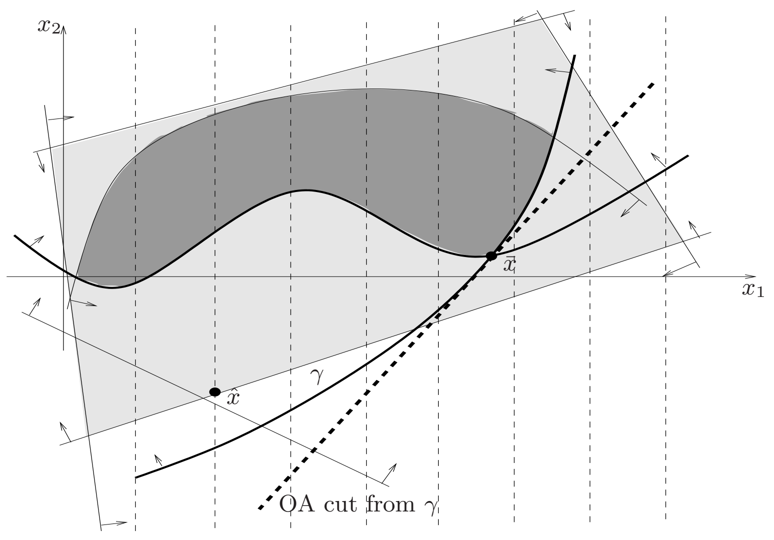

Fig. 2. The OA cut from γ does not cut off $\bar{x}$.

binary and general integer variables.

Before ending Section 2, we discuss previous FP implementations with respect to the general framework described above.

First, when the original problem (P) is an MILP, $(P1)$ is simply the LP relaxation of the current problem and $(P2)$ is the original MILP itself but with a different objective function. However, because in such a case problem $(P2)$ is probably as difficult as (P), Fischetti et al. [2] iteratively solved a trivial relaxation in which all the constraints are relaxed, i.e., an integer solution is obtained by rounding the fractional solution of $(P1)$.

Moreover, when the original problem (P) is a convex MINLP, i.e., $\mathcal{N} = \emptyset$, $(P1)$ is the NLP relaxation of the problem and $(P2)$ is a MILP relaxation of (P). In this case, we know that: (a) $(P1)$ is convex as well and it can ideally be solved to global optimality; and (b) $(P2)$ can be defined as the OA of (P) (see, e.g., Bonami et al. [8]) or replaced by a rounding phase (see Bonami and Gonçalves [10]).

Finally, when $\mathcal{N} \neq \emptyset$, as previously discussed, things get much more complicated because we have two different sources of nonconvexity. This is the main difference with respect to the previous FP algorithms and both $(P1)$ and $(P2)$ require specialized algorithmic techniques.

3 Code Structure

The algorithm was implemented within the AMPL environment [22]. We chose to use this framework to make it easy to change subsolver. In practice, the user can select the preferred solver to solve NLPs or MILPs, exploiting their advantages. In our case, problems $(P1)$ and $(P1fix)$ are solved using IPOPT [23]. Problem $(P2)$ is solved by CPLEX [24] modified by a tabu list hooked up to the solver via the incumbent callback function. This allows the user to define a function which is called during execution whenever CPLEX finds a new integer feasible solution. The tabu list is stored in a text file which is then exchanged between AMPL and CPLEX. Every time CPLEX finds an integer feasible solution, the specialized incumbent callback checks whether the new solution appears in the tabu list. If this is the case, the solution is rejected, otherwise the solution is accepted. CPLEX continues executing until either the optimal solution (excluding those forbidden) is found or a time limit is reached. In the case where an integer solution (x', y') found by CPLEX at the root node appears in the tabu list, CPLEX stops and no new integer feasible solution is passed to FP. In such a case, we amended problem $(P2)$ with a no-good cut [25] which excludes (x', y') and we call CPLEX again.

We also use a new solver/reformulator called ROSE (Reformulation Optimization Software Engine, see [26]), of which we exploit the following features.

1. Model analysis: getting information about nonlinearity and convexity of the constraints and integrality requirements of the variables (necessary to define $(P1)$ and $(P2)$).
2. Solution feasibility analysis: necessary to verify feasibility of the provided solutions.
3. OA cut generation: necessary to update $(P2)$.

We remark that some of the above features were added to ROSE within the context of the present work. In order to determine whether a constraint is convex, ROSE performs a recursive analysis of its expression tree [27] to determine whether it is an affine combination of convex functions. We call such a function *evidently convex* [26]. Evident convexity is a stricter notion than convexity: evidently convex functions are convex but the converse may not hold. Thus, it might happen that a convex constraint is labelled nonconvex; the information provided is in any case safe for our purposes, i.e., we generate OA cuts only from constraints which are certified to be convex.

4 Computational Results

In this section we report the results of preliminary computational experiments performed on an Intel Xeon 2.4 GHz with 8 GB RAM running Linux. We present the results in Tables 1-3. The algorithm terminates after the first MINLP feasible solution is found or a time limit is reached. The parameters are set in the following way: time limit of 2 hours of user CPU time, the absolute feasibility tolerance to evaluate constraints is 1e-6, and the relative feasibility tolerance is 1e-3 (used if absolute feasibility test fails). The tabu list length was set not to a fixed value, but to a value which was inversely proportional to the number of integer variables of the instance, i.e., the number of values to be stored for each solution of the tabu list. The value was 60,000 divided by the number of integer variables. The actual mean value of the solutions stored in the tabu list for the instances of the test bed was 35. The NLP solver used is IPOPT 3.5 trunk [23]. As a test set we use 243 instances taken from MINLPLib [28] (all those used in [12] excluding `oil` and `oil2` because the `log10` function is not supported by ROSE). We found an MINLP feasible solution for 199 of the instances as reported in Table 1. For each instance we report the CPU time (in seconds) and the number of iterations needed to find the (first) feasible solution (0 if it was found in less than 1 second) and the objective function value of such a solution. For 15 of the 199 solved instances, the algorithm found a feasible solution whose value is equal to the best-known solution reported in `http://www.gamsworld.org/minlp/minlplib/` within a 0.1% tolerance. The names of these instances are marked in boldface in Table 1. The instances for which the time limit is reached without finding any MINLP feasible solution are 19 and their names are reported in Table 2. The remaining 25 instances encounter some numerical problems during the execution (see Table 3). In general, finding a MINLP feasible solution for 199 of these 243 very difficult instances can be considered a very good performance for our algorithm. Of course the algorithm can be highly improved by taking into account the quality of the solution. First of all there is the possibility of continuing the execution of the algorithm instead of stopping it when the first feasible solution is found. Moreover, in most of the subproblems we solve, the original objective function is completely neglected. Using it in some way, i.e., combining it with the objective functions of subproblems ($P1$) and ($P2$) or adding a constraint which limits the value of the objective function, might lead to an improve-

Table 1. Instances for which a feasible solution was found within the time limit (199/243). The solution found is also the best-known solution for the instances marked in boldface.

instance	time	# iter	value	instance	time	# iter	value	instance	time	# iter	value
alan	0	1	4.2222E+00	gear3	0	1	7.3226E-01	ortez	0	1	-3.9039E-01
batchdes	0	1	2.2840E+05	gear4	0	1	9.6154E+05	parallel	0	1	4.0000E+10
batch	0	1	3.5274E+05	gear	0	1	7.3226E-01	**prob02**	0	1	1.1224E+05
contvar	608	1	1.9442E+07	gkocis	0	1	-2.7840E-03	**prob03**	0	1	1.0000E+01
csched1a	0	1	-2.5153E+04	hmittelman	0	1	2.1000E+01	prob10	0	2	4.7854E+00
csched1	0	1	-2.1049E+04	johnall	615	1	-2.0129E+02	procsel	0	1	-6.7200E-04
csched2a	4	1	-1.0287E+05	m3	0	1	2.4000E+06	product	17	1	-1.7772E+03
csched2	203	1	-1.2007E+05	m6	0	1	6.4800E+06	qap	0	1	4.9951E+05
deb6	4	3	2.3710E+02	m7_ar2_1	1	1	7.8800E+06	qapw	610	1	4.6012E+05
deb7	13	2	3.6983E+02	m7_ar25_1	1	1	7.8800E+06	ravem	171	1	7.6441E+05
deb8	2	1	1.4515E+06	m7_ar3_1	0	1	7.8800E+06	ravempb	185	1	7.6441E+05
deb9	18	3	4.2657E+02	m7_ar4_1	0	1	7.8800E+06	risk2bpb	2	1	-1.0195E+01
detf1	128	1	1.5976E+04	m7_ar5_1	0	1	7.8800E+06	saa_2	169	1	1.5976E+04
du-opt5	0	1	9.7825E+03	m7	0	1	7.8800E+06	sep1	0	1	-3.6123E+02
du-opt	0	1	1.1988E+04	mbtd	5,058	1	9.8529E+01	space25a	134	17	6.5069E+02
eg_all_s	62	4	1.0000E+05	meanvarx	0	1	2.1362E+01	space25	446	17	6.5069E+02
eg_disc2_s	7	1	1.0000E+05	no7_ar25_1	2,986	879	4.0000E+06	spectra2	0	1	3.0479E+02
eg_disc_s	9	1	1.0001E+05	no7_ar3_1	14	1	4.0000E+06	spring	1	2	1.3073E+00
elf	0	1	2.3992E+06	no7_ar4_1	138	162	4.0000E+06	st_e13	0	1	2.6004E+00
eniplac	2	3	-1.0209E+05	no7_ar5_1	7	1	4.0000E+06	st_e14	0	1	1.4555E+01
enpro48	609	1	1.6422E+06	nous1	0	1	2.1055E+00	st_e15	0	1	8.4763E+00
enpro48pb	610	1	1.6422E+06	nous2	0	1	2.1328E+00	st_e27	0	1	2.0020E+00
enpro56	607	1	7.0730E+05	nuclear14a	1,839	130	-1.1136E+00	st_e29	0	1	-2.9550E-01
enpro56pb	607	1	7.0730E+05	nuclear14b	670	5	-1.1007E+00	st_e31	1	1	-4.1766E-01
ex1221	0	1	8.4763E+00	nuclear14	647	3	-1.1213E+00	**st_e32**	0	1	-1.4303E+00
ex1222	0	1	1.0800E+00	nuclear24a	1,826	130	-1.1136E+00	st_e35	0	1	1.3270E+05
ex1223a	0	1	1.4556E+01	nuclear24b	668	5	-1.0979E+00	st_e36	1	3	-1.6644E+02
ex1223b	0	1	1.4556E+01	nuclear24	830	15	-1.1145E+00	st_e38	0	1	7.4478E+03
ex1223	0	1	1.4555E+01	nuclear25a	902	32	-1.0927E+00	**st_miqp1**	0	1	2.8100E+02
ex1224	0	1	-2.9550E-01	nuclear25b	627	2	-1.0750E+00	st_miqp2	0	1	7.0000E+00
ex1225	0	2	3.4000E+01	nuclear25	1,213	26	-1.1050E+00	st_miqp3	0	1	-1.0900E-04
ex1226	0	1	-6.1179E+00	nuclearva	132	1	-1.0068E+00	st_miqp4	0	1	-1.8889E-01
ex1233	1	1	2.5338E+05	nuclearvb	111	1	-1.0248E+00	st_miqp5	0	1	1.6881E+03
ex1243	0	1	1.6850E+05	nuclearvc	119	3	-9.8701E-01	stockcycle	5	1	2.1821E+05
ex1244	0	1	1.3109E+05	**nuclearvd**	424	1	-1.0370E+00	**st_test1**	0	1	-2.2230E-03
ex1263a	1	11	3.1000E+01	**nuclearve**	406	1	-1.0344E+00	st_test2	0	1	4.5600E-04
ex1263	66	137	1.2100E+02	**nuclearvf**	373	1	-1.0195E+00	st_test3	0	1	3.1900E-04
ex1264a	0	3	1.2000E+01	nvs01	0	1	4.9199E+02	st_test4	0	1	7.0000E+00
ex1264	29	12	1.8300E+01	nvs02	0	3	7.0780E+00	**st_test5**	0	1	-1.1000E+02
ex1265a	2	5	1.6500E+01	**nvs03**	0	2	1.6000E+01	st_test6	0	1	5.6700E+02
ex1265	158	6	1.4305E+01	nvs04	0	1	1.0040E+10	st_test8	0	1	2.4728E+04
ex1266a	0	1	1.6300E+01	nvs06	0	1	1.1596E+01	st_testgr1	0	1	0.0000E+00
ex1266	628	36	3.4600E+01	nvs07	0	2	6.0000E+00	st_testgr3	0	1	-6.1000E-05
ex3	0	1	1.1501E+02	nvs08	0	1	2.4119E+04	st_testph4	1	1	-1.2820E-03
ex3pb	0	1	1.1501E+02	nvs09	0	1	1.2972E+01	synheat	0	1	2.4872E+05
ex4	0	1	2.5566E+06	nvs10	0	1	-1.0240E+02	synthes1	0	1	9.9980E+00
fac1	0	1	5.4299E+08	nvs11	0	1	-1.5300E+02	synthes2	0	1	1.3608E+02
fac2	1	1	1.9520E+09	nvs12	0	1	-1.8800E+02	synthes3	0	1	1.1074E+02
fac3	0	1	1.0497E+08	nvs13	0	1	-1.6640E+02	tln2	1	27	2.8300E+01
feedtray2	2	1	8.7100E-04	nvs14	0	3	-2.9220E+04	tln4	1	4	1.2000E+01
feedtray	0	1	-1.2414E+01	**nvs15**	0	1	1.0000E+00	tln5	0	6	1.6500E+01
fo7_2	11	28	1.2000E+06	nvs16	0	1	1.4203E+01	tln6	1	8	2.5100E+01
fo7_ar25_1	3,066	879	1.2000E+06	nvs17	0	1	-2.7900E+02	tln7	1,072	397	1.0780E+02
fo7_ar3_1	8	1	1.2000E+06	nvs18	0	1	-2.0900E+02	tloss	5	7	2.4100E+01
fo7_ar4_1	141	162	1.2000E+06	nvs19	0	1	-2.8240E+02	**tls2**	1	4	5.3000E+00
fo7_ar5_1	7	1	1.2000E+06	nvs20	0	1	1.3869E+08	tls4	22	7	1.0000E+01
fo8_ar4_1	430	237	1.4000E+06	nvs21	0	1	-2.0000E-05	tls5	60	20	2.2500E+01
fo8_ar5_1	13	11	1.4000E+06	nvs23	1	2	-4.5480E+02	**tltr**	0	1	4.8073E+01
fo8	1,330	532	1.4000E+06	nvs24	1	3	-5.3620E+02	uselinear	51	1	1.9514E+03
fo9_ar3_1	417	186	1.6000E+06	o7_2	10	28	4.8000E+06	util	1	1	4.3393E+03
fo9_ar4_1	3,986	698	1.6000E+06	o7_ar25_1	2,987	879	4.8000E+06	var_con10	10	4	4.6317E+02
fo9_ar5_1	15	1	1.6000E+06	o7_ar3_1	7	1	4.8000E+06	var_con5	7	2	3.1517E+02
fo9	196	152	1.6000E+06	o7_ar4_1	136	162	4.8000E+06	water4	5	6	3.3353E+06
fuel	0	1	1.5155E+04	o7_ar5_1	8	1	4.8000E+06	waterx	0	1	3.3362E+06
gasnet	62	1	1.0246E+07	o8_ar4_1	622	235	8.2000E+06	waterz	3	4	3.3555E+06
gbd	0	1	3.7264E+00	o9_ar4_1	3,953	697	8.2000E+06				
gear2	0	1	7.3226E-01	oaer	0	1	-6.0000E-06				

ment of the quality of the solutions obtained with the proposed algorithm. That would be in the spirit of [4].

Table 2. Instances for which no feasible solution was found within the time limit (19/243)

deb10	fo9_ar2_1	lop97icx	nuclear49	tls12
ex1252	fo9_ar25_1	nuclear10a	product2	tls6
fo8_ar25_1	gastrans	nuclear49a	space960	tls7
fo8_ar3_1	lop97ic	nuclear49b	tln12	

Table 3. Instances with numerical problems during the execution (25/243)

4stufen	fo_7_ar_2_1	nuclear104	o7	super2
beuster	fo7	nuclear10b	pump	super3
cecil_13	fo_8_ar_2_1	nvs05	risk2b	super3t
eg_int_s	minlphix	nvs22	st_e40	waste
ex1252a	no_7_ar_2_1	o_7_ar_2_1	super1	windfac

5 Conclusions

In this paper we presented a Feasibility Pump algorithm aimed at solving nonconvex Mixed Integer Nonlinear Programming problems. The proposed algorithm is tailored to limit the impact of the nonconvexities in the MINLPs. These difficulties are extensively discussed. The preliminary results show that the algorithm behaves well with general problems on instances taken from MINLPLib.

References

1. Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers, Dordrecht, The Netherlands (2003)
2. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Mathematical Programming **104** (2004) 91–104
3. Bertacco, L., Fischetti, M., Lodi, A.: A feasibility pump heuristic for general mixed-integer problems. Discrete Optimization **4** (2007) 63–76
4. Achterberg, T., Berthold, T.: Improving the feasibility pump. Discrete Optimization **4** (2007) 77–86
5. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming **98** (2002) 23–47
6. Danna, E., Rothberg, E., Pape, C.L.: Exploiting relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming **102** (2005) 71–90
7. Duran, M., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Mathematical Programming **36** (1986) 307–339
8. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Mathematical Programming **119** (2009) 331–352
9. Abhishek, K.: Topics in Mixed Integer Nonlinear Programming. PhD thesis, Lehigh University (2008)
10. Bonami, P., Gonçalves, J.: Primal heuristics for mixed integer nonlinear programs. Technical report, IBM Research Report RC24639 (2008)
11. Vavasis, S.: Nonlinear Optimization: Complexity Issues. Oxford University Press, Oxford (1991)
12. Liberti, L., Nannicini, G., Mladenovic, N.: A good recipe for solving MINLPs. In Maniezzo, V., Stützle, T., Voß, S., eds.: Matheuristics. Volume 10 of Annals of Information Systems. Springer US (2008) 231–244

13. Nannicini, G., Belotti, P., Liberti, L.: A local branching heuristic for MINLPs. ArXiv, paper 0812.2188 (2009)
14. Fletcher, R., Leyffer, S.: Solving mixed integer nonlinear programs by outer approximation. Mathematical Programming **66** (1994) 327–349
15. Fletcher, R., Leyffer, S.: Numerical experience with lower bounds for MIQP branch-and-bound. SIAM Journal of Optimization **8** (1998) 604–616
16. Westerlund, T., Pörn, R.: Solving pseudo-convex mixed integer optimization problems by cutting plane techniques. Optimization and Engineering **3** (2002) 235–280
17. Westerlund, T.: Some transformation techniques in global optimization. In Liberti, L., Maculan, N., eds.: Global Optimization: from Theory to Implementation. Springer, Berlin (2006) 45–74
18. Consulting, A., Development: SBB Release Notes. (2002)
19. Abhishek, K., Leyffer, S., Linderoth, J.: FilMINT: An outer-approximation based solver for nonlinear mixed-integer programs. Technical report, Argonne National Laboratory (2007)
20. Fischetti, M., Salvagnin, D.: Feasibility pump 2.0. Technical report, DEI, University of Padova (September 2008)
21. Schoen, F.: Two-phase methods for global optimization. In Pardalos, P., Romeijn, H., eds.: Handbook of Global Optimization. Volume 2. Kluwer Academic Publishers, Dordrecht (2002) 151–177
22. Fourer, R., Gay, D., Kernighan, B.: AMPL: A Modeling Language for Mathematical Programming. Second edn. Duxbury Press/Brooks/Cole Publishing Co. (2003)
23. Wächter, A., Biegler, L.T.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming **106** (2006) 25–57
24. Ilog-Cplex. `www.ilog.com/products/cplex` (v. 11.0)
25. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: On Interval-subgradient and No-good Cuts. Technical report, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna (2009)
26. Liberti, L., Cafieri, S., Tarissan, F.: Reformulations in mathematical programming: a computational approach. In Abraham, A., Hassanien, A.E., Siarry, P., Engelbrecht, A., eds.: Foundations of Computational Intelligence Vol. 3. Studies in Computational Intelligence. Springer, Berlin (2009) 153–234
27. Liberti, L.: Writing global optimization software. In Liberti, L., Maculan, N., eds.: Global Optimization: from Theory to Implementation. Springer, Berlin (2006) 211–262
28. Bussieck, M., Drud, A., Meeraus, A.: MINLPLib - a collection of test models for mixed-integer nonlinear programming. INFORMS Journal on Computing **15** (2003) 114–119