Noname manuscript No.
(will be inserted by the editor)

Strengthening the Sequential Convex MINLP Technique by Perspective Reformulations

Claudia D'Ambrosio · Antonio Frangioni · Claudio Gentile

Received: date / Accepted: date

Abstract The Sequential Convex MINLP (`SC-MINLP`) technique is a solution method for nonconvex Mixed-Integer NonLinear Problems where the nonconvexities are separable. It is based on solving a sequence of convex MINLPs which trade a better and better relaxation of the nonconvex part of the problem with the introduction of more and more piecewise-linear nonconvex terms, and therefore binary variables. The convex MINLPs are obtained by partitioning the domain of each separable nonconvex term in the intervals in which it is convex and those in which it is concave. In the former, the term is left in its original form, while in the latter it is piecewise-linearized. Since each interval corresponds to a semi-continuous variable, we propose to modify the convex terms using the Perspective Reformulation technique to strengthen the bounds. We show by means of experimental results on different classes of instances that doing so significantly decreases the solution time of the convex MINLPs, which is the most time consuming part of the approach, and has therefore the potential to improving the overall effectiveness of `SC-MINLP`.

Keywords Global Optimization · NonConvex Separable Functions · Sequential Convex MINLP Technique · Perspective Reformulation

C. D'Ambrosio
LIX UMR 7161, École Polytechnique
Route de Saclay, 91128 Palaiseau – France
E-mail: dambrosio@lix.polytechnique.fr

A. Frangioni
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa – Italy
E-mail: frangio@di.unipi.it

C. Gentile
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", C.N.R.
Via dei Taurini 19, 00185 Rome – Italy
E-mail: gentile@iasi.cnr.it

1 Introduction and Motivation

We consider the following Mixed Integer NonLinear Programming (MINLP)

$$\text{(P)} \qquad \begin{array}{lll} \min & \sum_{j\in N} c_j x_j & \\ & f_i(x) + \sum_{j\in H(i)} g_{ij}(x_j) \leq 0 & i \in M \\ & l_j \leq x_j \leq u_j & j \in N \\ & x_j \in \mathbb{Z} & j \in I \end{array} ,$$

where M, N, $I \subseteq N$ and $H(i) \subseteq N$ are finite index sets, $f_i : \mathbb{R}^n \to \mathbb{R}$ are convex multivariate functions, whereas $g_{ij} : \mathbb{R} \to \mathbb{R}$ are nonconvex univariate functions; that is, (P) has *separable nonconvexities*. This simplified notation purposely hides many forms of structure that can, and will, be exploited if present but are not central in our discussion. For instance, $H(i) = \emptyset$ is possible, meaning that the i-th constraint is convex, and f_i may well be linear (affine). Also, the objective function need not be linear, and can have the same form $f(x) + \sum_{j\in H} g_j(x_j)$ as the constraints. Indeed, such a problem can always be brought in the form of (P) with the well-known reformulation trick whereby one introduces a new variable ω, redefines the objective function as "min ω", and adds the constraint $f(x) + \sum_{j\in H} g_j(x_j) - \omega \leq 0$. As this example shows, not all variables need necessarily appear in some nonconvex term, i.e., $\cup_{i\in M} H(i) \subsetneqq N$ is possible. Also, while bounds need be finite ($-\infty < l_j < u_j < \infty$) for all x_j that appear in at least one of the g_{ij}, this need not necessarily be so for those that do not. Finally, integrality constraints do not play a significant role in our development, i.e., $I = \emptyset$ is possible; (P) is still a $\mathcal{NP}$-hard problem in general.

The Sequential Convex MINLP (`SC-MINLP`) technique [3,4] for problems with this structure is based on the idea that for many univariate functions g_{ij} it is possible to automatically find $s(ij)+1$ points $l_j = l_{ij}^1 < l_{ij}^2 < \ldots < l_{ij}^{s(ij)} < l_{ij}^{s(ij)+1} = u_j$ so that g_{ij} is either convex or concave when restricted to each sub-interval $S_{ij}^s = [l_{ij}^s, l_{ij}^{s+1}]$ for $s \in \{1, \ldots, s(ij)\}$. For an algebraic C^2 function this amounts at computing the second derivative and finding all its roots in $[l_j, u_j]$; although this is not always practical (say, the roots may be very large numbers, or they may be exceedingly hard to compute), it is so for many cases of interest. Clearly, g_{ij} can also be defined piecewise (for a finite set of pieces) in $[l_j, u_j]$, and therefore can be nondifferentiable and even noncontinuous in a finite set of points, provided that it has the above property in each piece separately. In the following we will assume that this indeed holds. Then, for fixed i and $j \in H(i)$, we denote by $\check{S}(ij) = \{\, s \,:\, g_{ij}$ is *convex* in the sub-interval $[\, l_{ij}^s \,,\, l_{ij}^{s+1} \,]\,\}$, by $\hat{S}(ij) = \{\, s \,:\, g_{ij}$ is *concave* in the sub-interval $[\, l_{ij}^s \,,\, l_{ij}^{s+1} \,]\,\}$, and by $S(ij) = \check{S}(ij) \cup \hat{S}(ij)$.

Once this is done, it is easy to define a convex MINLP problem that provides a lower bound to (P) by just "keeping the convex parts of g_{ij}", while replacing g_{ij} with its best possible convex relaxation—a linear function—on the intervals where it is concave. This requires defining extra continuous and

binary variables x^s_{ij} and y^s_{ij}, subject to the constraints

$$x_j = l_j + \textstyle\sum_{s\in S(ij)} x^s_{ij} \tag{1}$$

$$(l^{s+1}_{ij} - l^s_{ij})y^{s+1}_{ij} \le x^s_{ij} \le (l^{s+1}_{ij} - l^s_{ij})y^s_{ij} \qquad s \in S(ij) \tag{2}$$

$$y^s_{ij} \in \{0,1\} \qquad s \in S(ij) \tag{3}$$

where for simplicity of notation we have introduced the constant $y^{s(ij)+1}_{ij} = 0$. Then, we replace the term $g_{ij}(x_j)$ with

$$g_{ij}(l^1_{ij}) + \textstyle\sum_{s\in\check{S}(ij)} \big(g_{ij}(l^s_{ij} + x^s_{ij}) - g_{ij}(l^s_{ij}) \big) + \sum_{s\in\hat{S}(ij)} \alpha^s_{ij} x^s_{ij} \tag{4}$$

where $\alpha^s_{ij} = (g_{ij}(l^{s+1}_{ij}) - g_{ij}(l^s_{ij}))/(l^{s+1}_{ij} - l^s_{ij})$. Clearly, (4) together with (1)–(3) defines a piecewise-convex underestimator of g_{ij}. Hence, this defines a relaxation of (P), say ($\underline{\text{P}}$). The relaxation is pictorially illustrated in Figure 1, where the continuous line is the original g_{ij} (that is neither differentiable nor continuous), and the dashed line shows how the—piecewise convex—function in ($\underline{\text{P}}$) differs from the original one.

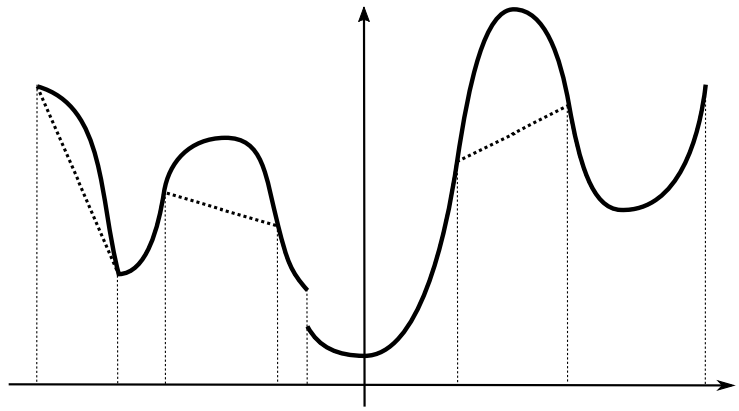

Fig. 1 Example of the piecewise-convex underestimator

It is immediately clear from the picture that solving ($\underline{\text{P}}$) (actually, solving any relaxation thereof, like the continuous one) provides a global valid lower bound on the optimal value of (P). Also, the approach suggests ways to produce valid global upper bounds that can be effective in practice [3,4]. If the thusly obtained bounds are not close enough to each other, it is possible to refine the piecewise-linear part of (P) by appropriately selecting some "concave" subinterval $s \in \hat{S}(ij)$ (for a properly chosen i and j) and further subdividing it in smaller intervals, obviously improving the approximation of the corresponding g_{ij}. This defines the `SC-MINLP` approach that, under appropriate assumptions, is globally convergent. We refrain from providing further details here since they are largely irrelevant for the development in this work.

Although `SC-MINLP` can be quite effective, its drawback is that it requires

the solution of a convex MINLP at each iteration. Even if convex MINLP are usually solved much more efficiently in practice than nonconvex ones, this is still a daunting task in general. Yet, the convex MINLPs produced by the SC-MINLP approach have one particularly valuable form of structure, that of *semi continuous variables with nonlinear convex cost*, that can be exploited by appropriate reformulation techniques to significantly improve the lower bounds obtained by the continuous relaxation, and hence (hopefully) the efficiency of the overall solution process. This is discussed in the next section.

2 Convex MINLP Relaxation Strengthening

To ease the notation, we will consider (P) written as

$$\min cx \tag{5}$$

$$\bar{f}_i(x) + \sum_{j\in H(i)} \sum_{s\in\check{S}(ij)} z^s_{ij} \leq 0 \qquad i \in M \tag{6}$$

$$z^s_{ij} \geq g_{ij}(l^s_{ij} + x^s_{ij}) - g_{ij}(l^s_{ij}) \qquad i \in M,\ j \in H(i),\ s \in \check{S}(ij) \tag{7}$$

$$x_j = l_j + \sum_{s\in S(ij)} x^s_{ij} \qquad i \in M, j \in H(i) \tag{8}$$

$$(l^{s+1}_{ij} - l^s_{ij})y^{s+1}_{ij} \leq x^s_{ij} \leq (l^{s+1}_{ij} - l^s_{ij})y^s_{ij} \quad i \in M,\ j \in H(i),\ s \in S(ij) \tag{9}$$

$$y^s_{ij} \in \{0,1\} \qquad i \in M,\ j \in H(i),\ s \in S(ij) \tag{10}$$

$$x_j \in \mathbb{Z} \qquad j \in I \tag{11}$$

where $\bar{f}_i(x) = f_i(x) + \sum_{j\in H(i)} g_{ij}(l^1_{ij}) + \sum_{s\in\hat{S}(ij)} \alpha^s_{ij} x^s_{ij}$; clearly, $\bar{f}_i$ is convex since f_i was. Furthermore, the g_{ij} terms are restricted to the sub-intervals where they are convex, hence as expected (5)–(11) is a convex MINLP. The introduction of the extra variables z^s_{ij} and the corresponding constraints (7) may look gratuitous at this point, but it both makes the following discussion easier, and it is what one actually does in practice, in many cases, to implement the *Perspective Reformulation* technique [7]. This allows to construct a reformulation of the problem whose continuous relaxation provides stronger bounds, which is simply obtained by replacing (7) with

$$z^s_{ij} \geq y^s_{ij}\left[\, g_{ij}(l^s_{ij} + x^s_{ij}/y^s_{ij}) - g_{ij}(l^s_{ij}) \,\right] \qquad s \in \check{S}(ij)\ ,\ j \in H(i)\ ,\ i \in M\ . \tag{12}$$

We refer to (5)–(6), (12), (8)–(11) as the *Perspective Reformulation* (PR) of (P). It is well-known that, if $h(x)$ is a convex function, then its *Perspective Function* (PF) $yh(x/y)$ describes its convex envelope when restricted to the mixed-integer set $\{\,(x,y) : 0 \leq x \leq uy\ ,\ y \in \{0,1\}\,\}$ corresponding to the semi-continuous variables definition. Hence, the continuous relaxation of (PR), the *Perspective Relaxation* (PR) of (P), provides tighter (usually, significantly so) lower bounds to the optimal value of (P) than the continuous relaxation of the standard formulation (5)–(11). Thus, the (PR) is often a better formulation of (P). Note that $\check{S}(ij)$ and $\hat{S}(ij)$ have nonempty intersection if g_{ij} is linear in some sub-interval s. In this case, clearly only one of the two (equivalent)

terms would be inserted in (4). Furthermore, (7) and (12) then coincide, as $yh(x/y) = h(x)$ if h is linear. As this case is easily dealt with, for simplicity we will avoid to further mention this occurrence.

A nontrivial issue, however, is how (PR) actually is solved. The point is that the PF in (12), although convex (if $y^s_{ij} \geq 0$), is nondifferentiable at $y^s_{ij} = 0$ even if g_{ij} is smooth. Therefore, just passing the constraint (12) to a MINLP solver incurs a substantial risk of numerical instability. Fortunately, a number of alternative approaches that can be used:

- *Perspective Cuts* (PC). The right-hand-side in (12) being a convex function, it can be represented as the supremum of (infinitely many) linear functions. In particular, in [7] it is shown that for a perspective reformulation constraint
$$z \geq yf(x/y) \qquad ly \leq x \leq ux$$
the associated perspective cuts are defined as
$$z \geq f'(\bar{x})x + [f(\bar{x}) - f'(\bar{x})\bar{x}]y \qquad \text{for any} \qquad \bar{x} \in [l, u].$$
Therefore, (12) can be replaced by the perspective cuts
$$z^s_{ij} \geq g'_{ij}(l^s_{ij} + \bar{x}^s_{ij})x^s_{ij} + [\, g_{ij}(l^s_{ij} + \bar{x}^s_{ij}) - g'_{ij}(l^s_{ij} + \bar{x}^s_{ij})\bar{x}^s_{ij}\,]y^s_{ij} \qquad (13)$$
for all $0 \leq \bar{x}^s_{ij} \leq (l^{s+1}_{ij} - l^s_{ij})$. The formula (13) assumes g_{ij} smooth, but it can be easily generalized to the nonsmooth case. While this would in principle require an infinite set of inequalities, it is trivial to dynamically separate (13) at the solution $z^{s*}_{ij}, x^{s*}_{ij}, y^{s*}_{ij}$ (> 0) of the continuous relaxation, just like any ordinary valid inequality, by just checking if it is satisfied with $\bar{x}^s_{ij} = x^{s*}_{ij}/y^{s*}_{ij}$; if not, the corresponding cut is inserted in the model. This has been shown to be a quite effective implementation in many cases. A significant side effect of this choice is that it completely "hides" the original g_{ij}. That is, if—as it happens in our test cases—everything but the g_{ij} in (P) is linear, then (PR) can be solved by using a MILP solver.
- *Specific reformulations.* Under mild assumptions on g_{ij}, it is possible to write its perspective function so as to eliminate the numerical difficulty corresponding to the nonsmoothness. In particular, if g_{ij} is representable as a Second Order Conic Programming problem (hereafter *SOCP-representable*), then ("almost always") so is its PF. For instance, in the quadratic case $g_{ij}(x_j) = a_{ij}x^2_j$ one could write (12) by means of the simple *rotated SOCP constraint* $z^s_{ij}y^s_{ij} \geq a_{ij}(x^s_{ij})^2$ (when $l_{ij} = l^s_{ij} = 0$), which is natively handled by any SOCP solver. Unfortunately, not all functions are SOCP-representable, so this approach is not completely general.
- *Projected reformulations.* If there had been no constraints on the binary variables y^s_{ij} except those pertaining to the corresponding continuous one x^s_{ij}, it would have been possible—subject to mild assumptions on g_{ij}—to construct a *Projected PR* [8] where the binary variables are eliminated, at the cost of making each term related to each g_{ij} in (12) a two- or four-piecewise one. This is not possible in this case because the constraints

(9) link both y_{ij}^s and y_{ij}^{s+1} with x_{ij}^s, making component-wise projection impossible. Projecting more than one variable at a time is conceptually possible, but it already becomes rather complex with disjoint pairs [2]. The *Approximated Projected PR* [5] can still be used in this case; it yields an intermediate relaxation that provides a stronger bound than the original continuous relaxation, although possibly weaker than that of the (PR). The approach can also be improved by using dual information [6] so that the bound is the same, but only at the root node of the enumeration tree, while it becomes weaker than that of the true (PR) as branching proceeds. Furthermore, the advantage of the approach—that of producing a problem with basically the same shape as the original one—can also be a disadvantage with general nonlinear terms g_{ij}, as it requires use of general (convex) nonlinear solvers for tackling continuous relaxations. This may be less efficient than linearizing them as PC does, especially if the rest of the problem is linear so that an LP solver can be used. Our computational results will show that, for the applications we tested, this is indeed the case: linearization is by far the most efficient approach.

The PC approach using (13) is therefore both quite general, not requiring any assumption on the original terms g_{ij}, and particularly well-suited to using efficient LP technology. This is why we have only tested that one. The results clearly indicate that, for the applications we tested, this is very likely to be the most efficient approach.

3 Computational Results

3.1 The instances

We tested our approach on two classes of MINLPs with the required structure, namely the Nonlinear Continuous Knapsack (NCK) problem and the Uncapacitated Facility Location (UFL) problem.

NCK is the nonlinear version of the classical (continuous) knapsack problem, where one is given a set of items N with associated weight and profit functions and a knapsack with capacity C. The aim is finding the quantities of each item to be inserted in the knapsack so as to maximize the overall profit while satisfying the capacity constraint. This arises in many applications, and it can be easily solved when the profit function is convex [9], but it is $\mathcal{NP}$-hard in general in the nonconvex case. Our specific instances stem from the problem of partitioning a given budget among advertisements for different products, maximizing the overall return [3, 4]. The return function is nonconvex because a small amount of allocation provides only a small return, up to a threshold when the advertisements are noticed by the consumers and result in substantial sales; however, as the advertisements quantity grows saturation sets in, yielding a

law of diminishing returns. The formulation we used is

$$\text{(NCK)} \qquad \begin{array}{ll} \max & \sum_{j\in N} p_j \\ & p_j - \frac{c_j}{1+b_j \exp(\,-a_j(x_j+d_j)\,)} \le 0 \quad j \in N \\ & \sum_{j\in N} x_j \le C \;\;,\;\; 0 \le x_j \le U \quad j \in N \end{array}$$

(note that the aforementioned trick has been used to reformulate the nonlinear objective function as nonlinear constraints). Hence, in this problem the original constraints are linear (and very simple), and in each nonconvex constraint there is only one nonlinear term, whose domain has exactly two sections, a convex one and a concave one. For each value of $|N| \in \{10, 20, 50, 100, 200, 500\}$ we randomly generated 10 instances, where a_j, b_j, c_j, and d_j were uniformly drawn from the intervals $[\,0.1\,,\,0.2\,]$, $[\,0\,,\,100\,]$, $[\,0\,,\,100\,]$, and $[\,-100\,,\,0\,]$, respectively. As for U and C, they are always fixed to 100 and $100|N|/2$, respectively.

In UFL we are given a set of customers, denoted with T, and set of facilities denoted with K. Each facility can satisfy a fraction of demand of each customer, but the shipment costs are univariate nonconvex functions. Apart from that, the mathematical model is linear (and quite classical):

$$\text{(UFL)} \qquad \begin{array}{lll} \min & \sum_{k\in K} C_k y_k + \sum_{t\in T}\sum_{k\in K} s_{kt} & \\ & a_{kt}(\sin(b_{kt}w_{kt}) + c_{kt}w_{kt})^2 - s_{kt} \le 0 & t \in T \;\;,\;\; k \in K \\ & \sum_{k\in K} w_{kt} = 1 & t \in T \\ & 0 \le w_{kt} \le y_k & t \in T \;\;,\;\; k \in K \\ & y_k \in \{0,1\} & k \in K \end{array}$$

where C_k is uniformly drawn in $[1, 100]$, a_{kt} can take the values in $\{-15, -25\}$, b_{kt} in $[2, 13]$, and c_{kt} in $[1, 13]$. Hence, (UFL) has $|K|\cdot|T|$ nonconvex constraints, again actually representing the nonlinear objective function, as well as binary variables. For each combination $(|K|, |T|) \in \{\,(6, 12)\,,\,(12, 24)\,,\,(24, 48)\,\}$ we generated 3 instances of increasing difficulty from the viewpoint of the nonlinear functions $g_{kt}(w_{kt})$, that have 1, 2, or 3 convex sections (and 1 to 2 concave ones). The results of the most difficult instance generated for $(|K|, |T|) = (24, 48)$ are not reported because it was exceedingly difficult to solve for all the methods we tested.

We refer the interested reader to [4] for more details on these applications.

3.2 Solvers and computational environment

We tested our approach, based on separation of Perspective Cuts (PC) implemented within a `Cplex` [11] cut callback, against 4 alternatives: a standard outer approximations (STD) for the nonlinear functions, and the packages `Bonmin 1.8.6` [1], `Minotaur 0.2.1` [12], and `Scip 4.0.0` [10]. All options solve the model (5)–(11). Perspective Cuts are applied on constraints (12) and substitute (7). PC and STD are implemented using the same algorithmic framework, i.e., passing the linearized model to `Cplex` and using its callbacks

(both the "lazy constraints" and the "user cuts" ones) to separate either PC or the standard gradient-based linearizations. Thus, how often and on which points separation is performed is completely controlled by `Cplex`. For `Bonmin` we tested three different algorithmic options (`bonmin.algorithm`): `B-BB`, `B-OA` and `B-Hyb`, corresponding to a Branch&Bound using a nonlinear solver, an outer-approximation approach using linear cuts (much similar in spirit to STD), and a hybrid approach. Both `B-OA`, and `B-Hyb` use an inner MILP solver (`bonmin.milp_solver`), for which we have tested two options: the default `CBC 2.9.9`, and `Cplex 12.7.0` (these denoted as `B-OA-C` and `B-Hy-C`, respectively). For `Minotaur` we tested different algorithmic options: `BNB`, `QG`, and `QPD` with either default `nlp_engine` or using `Ipopt 3.12.8` (denoted as `BNB-I`, `QG-I`, and `QPD-I`). `Scip` has been tested with default parameters, having set `Cplex 12.7.0` as the inner MILP solver and the `assumeconvex` option set to `true`. For all solvers we set a time limit of 10000 seconds and a required relative gap of `1e-4`. All the solvers have been compiled with `g++ 4.9.2` and ran, single-threaded, on a computer sporting a 16-core Intel Xeon E312xx (Sandy Bridge) processor at 2.3Ghz, with 32Gb of RAM, under Debian GNU/Linux 8.8.

3.3 Results for NCK

In Table 1 we report results obtained with different algorithms for `Bonmin` and `Minotaur`. For each option we report the (average) time in seconds and the gap reached within the time limit if some instance did not terminate (otherwise we report "-"). If all the instances hit the time limit we just report "tl". Since `B-OA-C` solved all instances within the allotted time, we do not report the gap.

	Bonmin							Minotaur						
size	B-BB		B-OA		B-Hyb		B-OA-C	BNB-I		QG-I		QPD-I		
	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap	time	
10	1.06	-	0.25	-	0.59	-	0.27	0.22	-	0.11	-	0.09	-	
20	2.99	-	0.34	-	2.12	-	0.32	0.53	-	0.22	-	0.16	-	
50	13.8	-	0.65	-	8.05	-	0.62	2.97	-	1.07	-	0.63	-	
100	78.9	-	9.16	-	7936	1.00	1.07	13.0	-	4.25	-	3.44	-	
200	1000	-	5035	0.62	4019	0.88	2.24	88.5	-	37.8	-	28.6	-	
500	tl	0.12	8035	0.62	9027	1.49	8.41	8621	0.07	7080	0.15	7692	0.16	

Table 1 NCK: `Bonmin` and `Minotaur` options comparison

Clearly, `B-OA-C` is by far the best option. We also tested other possible options (`B-QG`, `B-Ecp`, and `B-Hy-C`), but some of the instances were not correctly solved: either they self-aborted, or they needed to be aborted because the execution time was much larger than 10000 seconds. Therefore we do not report the corresponding results. For `Minotaur` we experienced issues with the `default nlp_engine`, as some instances were declared as correctly solved, but in fact the reported solution had a gap $\gg$ `1e-4`. Therefore we decided to report only the results obtained with `nlp_engine=ipopt`; anyway, `Ipopt` appeared also to be the more efficient option, in particular with the largest instances. In this case there is no clear dominance, with `QPD-I` being better

for the smaller instances and QG-I for the largest ones. This is why in the summary Table 2, where we compare the best options for each of the five algorithmic schemes, PC, STD, Scip, Bonmin, and Minotaur, for the latter we report the results with QPD-I n up to 200, and those with QG-I for $n = 500$. For PC and STD the column "cuts" reports the number of user cuts added by Cplex. Whenever not all the instances are solved within the time limit, we report both the (average) *inherent gap* (column "gap"), i.e., the gap between the upper and lower bounds produced by the solver, as well as the *best gap* (column "bgap"), i.e., the gap between the best bound produced by the relaxation in the solver and the best known feasible solution, although not necessarily the latter is actually produced by the solver.

size	PC		STD		Bonmin	MINOTAUR			SCIP
	time	cuts	time	cuts	time	time	gap	bgap	time
10	0.014	96	0.015	102	0.267	0.09	-	-	0.07
20	0.021	155	0.019	195	0.324	0.16	-	-	0.10
50	0.048	431	0.085	678	0.617	0.63	-	-	0.21
100	0.072	947	0.183	1182	1.067	3.44	-	-	0.66
200	0.105	1780	0.565	2461	2.237	28.6	-	-	131.2
500	0.380	4681	3.593	7821	8.406	7080	0.15	0.05	181.4

Table 2 NCK: comparison among the different algorithms

Table 2 clearly shows that PC is the best option. It performs quite close to STD, actually slightly losing out for $n = 20$, but as size grows the performances gap widens, reaching an order of magnitude for $n = 500$. Not surprisingly, the Outer Linearization algorithm in Bonmin (B-OA-C) is not far from STD, as they share the same basic approach; the difference is likely to be primarily attributable to the smaller overhead of an ad-hoc implementation w.r.t. a general-purpose MINLP solver. This is particularly true since the problem has an overall quite simple structure. However, improving the formulation with PR techniques clearly has a positive impact.

3.4 Results for UFL

The objective function in our UFL instances has trigonometric terms that SCIP 4.0 did not support, and therefore we do not report results for this solver. The results of the initial tuning for Bonmin and Minotaur are reported in Table 3, where a gap of ∞ means that no feasible solution was found. For B-OA and B-Hyb we used Cplex as MILP solver; not only this was (as expected) more efficient, but also the solver actually failed in at least one instance when using CBC instead. We tested other Bonmin options, B-Ecp and B-QG, but these failed on some of the instances as well. We also remark that B-BB had to be tested without setting bonmin.allowable_fraction_gap=1e-4, because when the option was set the solver did not terminate on instance 6x12x2, did not find any feasible solution for instance 24x48x2, and was also slightly worse on the other instances.

In this case there is even less clear dominance among the options, with

	Bonmin						Minotaur					
instance	B-BB		B-OA-C		B-Hy-C		BNB		QPD		QG-I	
	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap
6x12x1	176	-	1.76	-	1.37	-	538	-	24.8	-	4.66	-
6x12x2	tl	1.16	7.25	-	5.64	-	tl	29.17	tl	51.08	65.5	-
6x12x3	tl	657.6	tl	∞	tl	∞	tl	∞	tl	315.5	tl	260.3
12x24x1	1592	-	9.68	-	7.14	-	tl	8.07	tl	66.57	57.4	-
12x24x2	tl	18.77	93.8	-	57.9	-	tl	∞	tl	∞	tl	17.40
12x24x3	tl	∞	tl	∞	tl	∞	tl	∞	tl	∞	tl	271.6
24x48x1	tl	84.70	116	-	132	-	tl	∞	tl	∞	2844	-
24x48x2	tl	73.44	tl	∞	tl	∞	tl	∞	tl	∞	tl	31.49

Table 3 UFL: Comparison among Bonmin and Minotaur options

the B-BB option being often much slower, but at least succeeding in finding feasible solutions in cases where the other approaches find none. The results for Minotaur are limited to the algorithmic options BNB, QPD, and QG-I, as all the other options we tested could not solve some of the instances (we had to abort the run after many hours). At least, in this case QG-I clearly emerges as the best option.

Finally, Table 4 reports comparison of the four algorithmic schemes; for Bonmin, the best option was hand-picked on an instance-per-instance basis. Note that in some cases the "best gap" is larger than the "gap", because the lower bound found by the method was even negative. Similarly to the NCK case, the two ad-hoc linearization approaches clearly outperformed the use of general-purpose solvers. An analogous trend, even more pronounced, also shows up when comparing PC with STD. The latter can be more efficient on smaller or "easier" instances; note that the "complexity" of the objective function (number of convex pieces) grows when going from "x1" to "x2" to "x3" instances. For the most complex instances PC always significantly outperformed STD, either in terms of running time or of final gap (or both). A close examination of the solver logs showed that PC—as expected—always produced significantly better lower bounds; often (although not always) this also translated in significantly better upper bounds. Occasionally STD produced better upper bounds, but in these cases the difference was much less relevant; anyway, any advantage in the upper bound was largely negated by the worse lower bound. The cuts generated by PC were also much more effective in terms of time, in the sense that much fewer of them were needed to solve a single relaxation. That it, PC usually required significantly fewer separation passes than STD, and hence less LP solutions. For instances that hit the time limit, this allowed PC to explore more nodes, and hence usually find also better solutions.

instance	PC				STD				Bonmin			Minotaur		
	time	gap	bgap	cuts	time	gap	bgap	cuts	time	gap	bgap	time	gap	bgap
6x12x1	0.35	-	-	1673	0.26	-	-	1531	1.37	-	-	4.66	-	-
6x12x2	0.45	-	-	1842	0.42	-	-	1796	5.64	-	-	65.6	-	-
6x12x3	7921	-	-	33417	tl	54.3	52.4	180561	tl	657	796	tl	260	615
12x24x1	3.36	-	-	9565	2.55	-	-	8971	7.14	-	-	57.4	-	-
12x24x2	46.1	-	-	19653	27.3	-	-	17384	57.9	-	-	tl	17.4	10.5
12x24x3	tl	23.9	23.9	127380	tl	121	134	284557	tl	∞	1524	tl	272	1447
24x48x1	261	-	-	81372	316	-	-	102160	116	-	-	2844	-	-
24x48x2	tl	5.93	5.67	164809	tl	9.66	9.66	409177	tl	73.4	26.4	tl	31.5	24.6

Table 4 UFL: Comparison among different algorithms

4 Conclusions and Perspectives

This work proposes a conceptually simple modification of the Sequential Convex MINLP (`SC-MINLP`) approach whereby, after the first relaxation step aimed at removing nonconvex terms, a second reformulation step occurs where the representation of the convex terms is tightened using the Perspective Reformulation technique. This is especially attractive for problems where it is anyway convenient to linearize the convex terms (e.g., because apart from them the problem is linear), since then the implementation differs only slightly from that of a standard Outer Approximation (OA) one: the cuts to be separated are only slightly different, but the bound they provide are significantly better. Preliminary computational results show that this approach typically outperforms the standard OA algorithm, even if implemented ad-hoc, and even more so general-purpose OA implementations as well as many other different available options for solving the convex MINLPs. For easier and/or smaller instances the standard OA may be preferable, but using the PR seems to be the best option as the size and the complexity increases. Since the implementation effort within an existing OA solver is minimal, this approach is worth considering for implementations of `SC-MINLP`. The results are clearly partial, in the sense that the approach has not yet been tested in a "live" `SC-MINLP`. Yet, the idea appears promising, because the PC generated with one specific instantiation of ($\underline{\text{P}}$) can be reused when solving the next one, with some binary variables added. It is also possible that (PR) could be strengthened even further by considering the convex hull of a larger set of variables together. These could be fruitful directions for future research.

Acknowledgements The second and the third authors are partially supported by the project MIUR-PRIN 2015B5F27W. This paper has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement n. 764759.

References

1. Bonami, P., Lee, J.: Bonmin users' manual. Tech. rep. (June 2006)
2. Castro, J., Frangioni, A., C. Gentile: Perspective reformulations of the CTA problem with l_2 distances. Operations Research **62**(4), 891–909 (2014)
3. D'Ambrosio, C., Lee, J., Wächter, A.: A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity. In: Algorithms—ESA 2009, *LNCS*, vol. 5757, pp. 107–118. Springer, Berlin (2009)
4. D'Ambrosio, C., Lee, J., Wächter, A.: An algorithmic framework for MINLP with separable non-convexity. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, *The IMA Volumes in Mathematics and its Applications*, vol. 154, pp. 315–347. Springer New York (2012)
5. Frangioni, A., Furini, F., Gentile, C.: Approximated perspective relaxations: a project&lift approach. Computational Optimization and Applications **63**(3), 705–735 (2016)
6. Frangioni, A., Furini, F., Gentile, C.: Improving the Approximated Projected Perspective Reformulation by Dual Information. Operations Research Letters **45**, 519–524 (2017)
7. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0-1 mixed integer programs. Mathematical Programming **106**(2), 225–236 (2006)

8. Frangioni, A., Gentile, C., Grande, E., Pacifici, A.: Projected perspective reformulations with applications in design problems. Operations Research **59**(5), 1225–1232 (2011)
9. Frangioni, A., Gorgone, E.: A library for continuous convex separable quadratic knapsack problems. European Journal of Operational Research **229**(1), 37–40 (2013)
10. Gleixner, A., Eifler, L., Gally, T., Gamrath, G., Gemander, P., Gottwald, R.L., Hendel, G., Hojny, C., Koch, T., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schlösser, F., Serrano, F., Shinano, Y., Viernickel, J.M., Vigerske, S., Weninger, D., Witt, J.T., Witzig, J.: The SCIP optimization suite 5.0. Tech. Rep. 17-61, ZIB, Takustr.7, 14195 Berlin (2017)
11. IBM: ILOG CPLEX 12.7 User's Manual. IBM (2016)
12. Mahajan, A., Leyffer, S., Linderoth, J., Luedtke, J., Munson, T.: Minotaur: A mixed-integer nonlinear optimization toolkit. Optimization Online 6275 (2017)