ISTITUTO DI ANALISI DEI SISTEMI ED INFORMATICA
“Antonio Ruberti”
CONSIGLIO NAZIONALE DELLE RICERCHE

C. D’Ambrosio, A. Frangioni, C. Gentile

STRENGTHENING THE SEQUENTIAL CONVEX MINLP TECHNIQUE BY PERSPECTIVE REFORMULATIONS

R. 17-01, 2017

Claudia D’Ambrosio – LIX UMR 7161, École Polytechnique, Palaiseau (France).
Email: `dambrosio@lix.polytechnique.fr`.

Antonio Frangioni – Dipartimento di Informatica, Università di Pisa (Italy).
Email: `frangio@di.unipi.it`.

Claudio Gentile – Istituto di Analisi dei Sistemi ed Informatica “A. Ruberti”, Consiglio Nazionale delle Ricerche, Rome (Italy). Email: `gentile@iasi.cnr.it`.

ISSN: 1128–3378

Collana dei Rapporti dell'Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", CNR

via dei Taurini 19, 00185 ROMA, Italy

tel. ++39-06-49937101/02
fax ++39-06-49937106
email: `iasi@iasi.cnr.it`
URL: `http://www.iasi.cnr.it`

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 nonlinear 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 separately considering each separable nonconvex term in the intervals in which it is convex and those in which it is concave, where the former are left in their original form while the latter are piecewise-linearized. Because each interval corresponds to a semi-continuous variable, we rather 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

1. Introduction and Motivation

We consider Mixed Integer NonLinear Programming (MINLP) problems with separable nonconvexities, i.e.,

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

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. 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). The objective function need not be linear, but it can always be made so with a well-known reformulation trick. Not all variables need necessarily appear in some nonconvex term, i.e., $\cup_{i \in M} H(i) \subsetneq N$ is possible. Also, while bounds need be finite ($-\infty < l_j < u_j < \infty$) for all x_j that do appear in at least one of the g_{ij}, this needs 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; the problem, that we denote as (P) in the following, is still $\mathcal{NP}$-hard in general.

The Sequential Convex MINLP (`SC-MINLP`) technique [2, 3] has been proposed for problems with this structure. The idea behind the approach is that for many univariate functions g_{ij} it is possible to automatically partition the interval $[l_j, u_j]$ by finding $s(ij) + 1$ points $l_j = l^1_{ij} < l^2_{ij} < \ldots < l^{s(ij)}_{ij} < l^{s(ij)+1}_{ij} = u_j$ so that g_{ij} is either convex or concave when restricted to each of the sub-intervals $S^s_{ij} = [l^s_{ij}, l^{s+1}_{ij}]$ for each $s \in \{1, \ldots, s(ij)\}$. 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 possible (say, the roots may not always be finite, or they may be exceedingly hard to compute), it is so for many cases of practical interest. Clearly, the approach can easily be extended to functions that have the required property piecewise (for a finite set of pieces) in $[l_j, u_j]$, and therefore that can be nondifferentiable, and even noncontinuous, in a finite set of points. In the following we will assume that this can be indeed done. Then, for fixed i and $j \in H(i)$, we denote by $\check{S}(ij) = \{\, s : g_{ij}$ is *convex* in the sub-interval $[\, l^s_{ij} \,,\, l^{s+1}_{ij} \,] \,\}$, by $\hat{S}(ij) = \{\, s : g_{ij}$ is *concave* in the sub-interval $[\, l^s_{ij} \,,\, l^{s+1}_{ij} \,] \,\}$, 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

$$
\begin{array}{lll}
x_j = l_j + \sum_{s \in S(ij)} x^s_{ij} & & (1) \\
(l^{s+1}_{ij} - l^s_{ij}) y^{s+1}_{ij} \leq x^s_{ij} \leq (l^{s+1}_{ij} - l^s_{ij}) y^s_{ij} & s \in S(ij) & (2) \\
y^s_{ij} \in \{0, 1\} & s \in S(ij) & (3)
\end{array}
$$

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

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

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 ($\underline{\text{P}}$), that is a *convex* MINLP. 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 [2, 3]. If the thusly obtained bounds are not close enough to each other, it is possible to refine the piecewise linear part of the relaxation by appropriately selecting some "concave" sub-interval $s \in \hat{S}(ij)$ (for a properly chosen i and j) and further subdividing it in smaller intervals, correspondingly improving the relaxation of the concave g_{ij} in the obvious way. This defines the `SC-MINLP` approach that, under appropriate assumptions, is globally convergent. We refrain of providing further details of the approach 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. The basic observation underlying our work is that, however, the convex MINLPs produced by the `SC-MINLP` have one particularly valuable form of structure, that of *semi continuous variables with nonlinear convex cost*, that can in turn 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 the relaxation ($\underline{\text{P}}$) written as

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

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

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

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

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

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

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

where $\bar{f}_i = f_i(x) + \sum_{j\in H(i)} g_{ij}(l^1_{ij}) + \sum_{s\in\hat{S}(ij)} \alpha^s_{ij} x^s_{ij}$; clearly, $\bar{f}_i$ is convex since f_i was. Now, however, also the g_{ij} are so, hence (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 is actually justified, in most cases, when one uses the *Perspective Reformulation* technique [6] to construct an alternative model of the problem that is equivalent for integer values of the y^s_{ij} variables, but whose continuous relaxation provides stronger bounds. This is simply obtained by replacing (7) with

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

We refer to (5)–(6), (12), (8)–(11) as the *Perspective Reformulation* (PR) of ($\underline{\text{P}}$). It is well-known that, if $h(x)$ is a convex function, then its *perspective function* $yh(x/y)$ 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* ($\underline{\text{PR}}$) of ($\underline{\text{P}}$), provides tighter (usually, significantly so) lower bounds to the optimal value of ($\underline{\text{P}}$) than the continuous relaxation of the standard formulation (5)–(11). Thus, the (PR) is often a better formulation of ($\underline{\text{P}}$).

A nontrivial issue, however, is how ($\underline{\text{PR}}$) actually is solved. The point is that the perspective function 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, there are a number of alternative approaches that can be used:

- *Perspective Cuts* (P/C). Because the right-hand-side in (12) is a convex function, it can be represented as the supremum of (infinitely many) linear functions. In particular, it can be seen that (12) is satisfied on the mixed-integer set $\{ (x^s_{ij}, y^s_{ij}) \, : \, 0 \leq x^s_{ij} \leq (l^{s+1}_{ij} - l^s_{ij})y^s_{ij} \, , \, y^s_{ij} \in \{0,1\} \}$ if and only if

$$z^s_{ij} \geq g'_{ij}(l^s_{ij} + \bar{x}^s_{ij})x^s_{ij} + [\, g_{ij}(l^s_{ij} + \bar{x}^s_{ij}) - g'_{ij}(l^s_{ij} + \bar{x}^s_{ij})\bar{x}^s_{ij} \,]y^s_{ij} \tag{13}$$

  holds *for all* $0 \leq \bar{x}^s_{ij} \leq (l^{s+1}_{ij} - l^s_{ij})$. Although (13) only works if g_{ij} is smooth, it can be easily generalized to the nonsmooth case [6]. While this approach would in principle require an infinite set of inequalities, it is very easy to enforce (13) approximately by dynamically separating them at the solution of the continuous relaxation just like any ordinary valid inequality: given a solution z^{s*}_{ij}, x^{s*}_{ij}, y^{s*}_{ij} of the continuous relaxation with $y^{s*}_{ij} > 0$, it is enough to test if (13) is satisfied with $\bar{x}^s_{ij} = x^{s*}_{ij}/y^{s*}_{ij}$, and if not to insert the corresponding cut 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 specific nonlinear features of g_{ij}; for instance, if everything else but the g_{ij} in ($\underline{\text{P}}$) is linear, then the problem can be solved by using a MILP solver. This actually happens in our test cases.

- *Specific reformulations.* When g_{ij} has some specific property, 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 SOCP-representable, then ("almost always") its perspective function is also so. For instance, in the quadratic case $g_{ij}(x_j) = a_{ij}x_j^2$ 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 can be handled without any problem by any SOCP solver. Unfortunately, not all functions are SOCP-representable (in particular transcendental ones), so this approach cannot always be used.

- *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 Perspective Reformulation* [7] 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^s_{ij} and y^{s+1}_{ij} with x^s_{ij}, 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 [1]. The *Approximated Projected*

Perspective Reformulation [4] 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 ($\underline{\text{PR}}$). The approach can also be improved by using dual information [5] 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 ($\underline{\text{PR}}$) 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 P/C 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 P/C approach using (13) is therefore both completely 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, at least for the applications we tested, this is anyway very likely to be the most efficient approach.

3. Computational Results

3.1. The instances

We tested our instances 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, say, the profit function is convex quadratic [8], but it is $\mathcal{NP}$-hard in general in the nonconvex case. Our specific instances stem from a continuous resource-allocation problem, where one has to decide how to partition a given budget among advertisements for different products, maximizing the overall return from all categories [2, 3]. 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, and the return increase becomes negligible. For each $j \in N$, we used a linear weight function and the profit function $c_j / (\, 1 + b_j \exp(\, -a_j(x_j + d_j)\,)\,)$, where x_j is the variable representing the quantity of item j to insert in the knapsack. Hence, in this problem the original constraints are linear (and very simple), and there is only a single nonconvex constraint (corresponding to the objective function) which is subdivided into 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.

- 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; the rest of the mathematical model is linear (and quite classical), although binary variables are used to model fixed costs. Hence, in this case there are $|K| \cdot |T|$ nonconvex constraints. For each combination $(K, T) \in \{(6, 12), (12, 24), (24, 48)\}$ we generated 3 instances of increasing difficulty from viewpoint of the nonlinear functions involved, that is with 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 [3] 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` cut callback, against 4 alternatives: standard outer approximations (STD) for the nonlinear functions implemented within the same scheme as PC, and the packages `Bonmin`, `Minotaur`, and `Scip`. 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` solve MILP problems; for these we have tested two options for the MILP solver (`bonmin.milp_solver`), the default `CBC`, and `Cplex 12.7.0`. For `Minotaur` we tested different algorithmic options: `BNB`, `QG`, and `QPD` with either default `nlp_engine` or with `nlp_engine=ipopt` (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	BB		OA		Hyb		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	13.83	-	0.65	-	8.05	-	0.62	2.97	-	1.07	-	0.63	-
100	78.91	-	9.16	-	7936.7	1.00	1.07	12.98	-	4.25	-	3.44	-
200	1000.4	-	5035.6	0.62	4019.4	0.88	2.24	88.36	-	37.81	-	28.59	-
500	tl	0.12	8035.4	0.62	9026.9	1.49	8.41	8621.8	0.07	7080.4	0.15	7691.8	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 where not correctly solved: either they self-aborted, or they needed to be aborted because the execution time was much more 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 $n = 500$.

size	PC	STD	Bonmin	MINOTAUR		SCIP
	time	time	time	time	gap	time
10	0.014	0.015	0.267	0.086	-	0.067
20	0.021	0.019	0.324	0.157	-	0.099
50	0.048	0.085	0.617	0.627	-	0.214
100	0.072	0.183	1.067	3.441	-	0.663
200	0.105	0.565	2.237	28.588	-	131.97
500	0.380	3.593	8.406	7080.3	0.15	181.36

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 the current version of `SCIP` does 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 `OA` and `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, `Ecp` and `QG`, but these failed on some of the instances as well. We also remark that the option `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 the NLP-based Branch&Bound 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` (with `nlp_engine=ipopt`), 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.

	`Bonmin`						`Minotaur`					
instance	`BB`		`OA-C`		`Hy-C`		`BNB`		`QPD`		`QG-I`	
	time	gap	time	gap	time	gap	time	gap	time	gap	time	gap
6x12x1	176.1	-	1.76	-	1.37	-	538.0	-	24.82	-	4.66	-
6x12x2	tl	1.16	7.25	-	5.64	-	tl	29.17	tl	51.08	65.49	-
6x12x3	tl	657.6	tl	∞	tl	∞	tl	∞	tl	315.5	tl	260.3
12x24x1	1592.2	-	9.68	-	7.14	-	tl	8.07	tl	66.57	57.42	-
12x24x2	tl	18.77	93.84	-	57.87	-	tl	∞	tl	∞	tl	17.40
12x24x3	tl	∞	tl	∞	tl	∞	tl	∞	tl	∞	tl	271.6
24x48x1	tl	84.70	115.8	-	132.1	-	tl	∞	tl	∞	2844.3	-
24x48x2	tl	73.44	tl	∞	tl	∞	tl	∞	tl	∞	tl	31.49

Table 3: UFL: Comparison among `Bonmin` and `Minotaur` options

Finally, Table 4 reports comparison of the four algorithmic schemes; for `Bonmin`, the best option was hand-picked on an instance-per-instance basis. 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 number of convex pieces (hence, the "complexity" of the objective function) 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	time	gap	time	gap	time	gap
6x12x1	0.35	-	0.26	-	1.37	-	4.66	-
6x12x2	0.45	-	0.42	-	5.64	-	65.49	-
6x12x3	7921.1	-	tl	54.3	tl	657.6	tl	260.3
12x24x1	3.36	-	2.55	-	7.14	-	57.42	-
12x24x2	46.09	-	27.29	-	57.87	-	tl	17.40
12x24x3	tl	23.9	tl	121.2	tl	∞	tl	271.6
24x48x1	261.0	-	315.6	-	115.8	-	2844.3	-
24x48x2	tl	5.93	tl	9.66	tl	73.44	tl	31.49

Table 4: UFL: Comparison among different algorithms

4. Conclusions and Perspectives

We proposed a conceptually simple modification of the Sequential Convex MINLP (`SC-MINLP`) technique, whereby, after the first reformulation 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. Our approach is especially attractive for problems where it is anyway computationally convenient to linearize the convex terms (e.g., because apart from them the problem is linear), since then the implementation of our approach differs only slightly from that of a standard Outer Approximation (OA) one: the cuts to be separated are just a bit different, but they still provide significantly better bounds. Our preliminary computational results show that this approach typically outperforms the standard OA algorithm, even if implemented ad-hoc, and even more so its general-purpose implementation in `Bonmin` as well as many other different available options for solving the convex MINLPs. For easier and/or smaller instances standard OA may be preferable, but, at least in our test set, using the PR is clearly the best option as the size and the complexity increases. Since the implementation effort within an existing OA approach is minimal, we can confidently state that our approach is worth considering for implementations of `SC-MINLP`. Our results are clearly partial, in the sense that we have not yet tested our approach in a "live" `SC-MINLP`. Yet, we feel that the idea is promising, because one can then reuse the PR cuts generated with one specific instantiation of ($\underline{P}$) when solving the next one (with some binary variables added). It is also possible that the relaxation 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.

References

[1] J. Castro, A. Frangioni, and C. Gentile, "Perspective Reformulations of the CTA Problem with L_2 Distances," *Operations Research*, vol. 62, no. 4, pp. 891–909, 2014.

[2] C. D'Ambrosio, J. Lee, and A. Wächter, "A global-optimization algorithm for mixed-integer nonlinear programs having separable non-convexity," in *Algorithms—ESA 2009*, vol. 5757 of *LNCS*, pp. 107–118, Springer, Berlin, 2009.

[3] C. D'Ambrosio, J. Lee, and A. Wächter, "An algorithmic framework for minlp with separable non-convexity," in *Mixed Integer Nonlinear Programming* (J. Lee and S. Leyffer, eds.), vol. 154 of *The IMA Volumes in Mathematics and its Applications*, pp. 315–347, Springer New York, 2012.

[4] A. Frangioni, F. Furini, and C. Gentile, "Approximated Perspective Relaxations: a Project&Lift Approach," *Computational Optimization and Applications*, vol. 63, no. 3, pp. 705–735, 2016.

[5] A. Frangioni, F. Furini, and C. Gentile, "Improving the Approximated Projected Perspective Reformulation by Dual Information," *Operations Research Letters*, vol. 45, pp. 519–524, 2017.

[6] A. Frangioni and C. Gentile, "Perspective cuts for a class of convex 0-1 mixed integer programs," *Mathematical Programming*, vol. 106, no. 2, pp. 225–236, 2006.

[7] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici, "Projected Perspective Reformulations with Applications in Design Problems," *Operations Research*, vol. 59, no. 5, pp. 1225–1232, 2011.

[8] A. Frangioni and E. Gorgone, "A Library for Continuous Convex Separable Quadratic Knapsack Problems," *European Journal of Operational Research*, vol. 229, no. 1, pp. 37–40, 2013.