Università di Pisa

Dipartimento di Informatica

Technical Report

Improving the Approximated Projected Perspective Reformulation by Dual Information

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa – Italy
frangio@di.unipi.it

Fabio Furini
LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, 75775 Paris – France
fabio.furini@dauphine.fr

Claudio Gentile
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti"
Consiglio Nazionale delle Ricerche (IASI-CNR)
Via dei Taurini 19, 00185 Roma – Italy
gentile@iasi.cnr.it

September 2, 2016

address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. tel: +39 050 2212700 fax: +39 050 2212726

Improving the Approximated Projected Perspective Reformulation by Dual Information

Antonio Frangioni
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 Pisa – Italy
frangio@di.unipi.it

Fabio Furini
LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, 75775 Paris – France
fabio.furini@dauphine.fr

Claudio Gentile
Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti"
Consiglio Nazionale delle Ricerche (IASI-CNR)
Via dei Taurini 19, 00185 Roma – Italy
gentile@iasi.cnr.it

September 2, 2016

Abstract

We propose an improvement of the Approximated Projected Perspective Reformulation (AP^2R) of [1] for the case in which constraints linking the binary variables exist. The new approach requires to solve the Perspective Reformulation (PR) once, and then use the corresponding dual information to reformulate the problem prior to applying AP^2R, thereby combining the root bound quality of the PR with the reduced relaxation computing time of AP^2R. Computational results for the cardinality-constrained Mean-Variance portfolio optimization problem show that the new approach is competitive with state-of-the-art ones.

Keywords: *Mixed-Integer NonLinear Problems, Semi-continuous Variables, Perspective Reformulation, Projection, Lagrangian Relaxation, Portfolio Optimization*

1 Introduction

We study solution techniques for convex separable Mixed-Integer NonLinear Programs (MINLP) with n semi-continuous variables $x_i \in \mathbb{R}$ for $i \in N = \{1, \dots, n\}$. That is, each x_i either assumes the value 0, or lies in some given compact nonempty interval $\mathcal{X}_i = [\underline{x}_i, \bar{x}_i]$ ($-\infty < \underline{x}_i < \bar{x}_i < \infty$): this can be expressed, introducing $y_i \in \{0, 1\}$ for $i \in N$, as

$$\begin{aligned} \text{(P)} \qquad & \min h(z) + \textstyle\sum_{i \in N} f_i(x_i) + c_i y_i && (1)\\ & A(x) + By + C(z) = b && (2)\\ & (x, z) \in \mathcal{O} && (3)\\ & \underline{x}_i y_i \le x_i \le \bar{x}_i y_i \quad , \quad y_i \in \{0,1\}^n \quad , \quad x_i \in \mathbb{R}^n \qquad i \in N. && (4) \end{aligned}$$

We assume the functions f_i to be closed convex, one time continuously differentiable and finite in the interval $(\underline{x}_i, \bar{x}_i)$; w.l.o.g. we also assume $f_i(0) = 0$. In (P) we single out the *linking constraints* (2) that

link the binary variables y_i with anything except the corresponding x_i, the latter being done by (4). These are the "problematic" constraints in [1, 7], and the aim of this paper is to deal with them in the a cost-effective way. For our approach to work they must have a compatible structure with that of (1); we initially assume linearity in x for simplicity ($A(x) = Ax$), but extensions are discussed in §3. Linearity in y can be assumed without loss of generality, as long as separability holds. Because our approach hinges on availability of dual information, we assume that the functions $h(\cdot)$ and $C(\cdot)$ in the "other variables z" and the "other constraints (3)" are convex, i.e., (P) is a convex MINLP. Actually, in many applications everything but (1) is linear. When we can ignore the structure of (2), we just refer to (3)–(4) as "$(x, y, z) \in \mathcal{P}$".

Often, the most pressing issue in solving (P) is to derive tight lower bounds on its optimal value ν(P), which is typically done by solving its (convex) continuous relaxation ($\underline{\text{P}}$) (we denote by ν(X) and ($\underline{\text{X}}$), respectively, the optimal value and the continuous relaxation of any problem (X)). However, often $\nu(\underline{\text{P}}) \ll \nu(\text{P})$, making solution approaches inefficient. The presence of semi-continuous variables has been exploited to propose *reformulations* (P′) of (P) such that $\nu(\underline{\text{P}'}) \gg \nu(\underline{\text{P}})$, and that therefore are better suited for enumerative approaches. This starts from considering (1) as $h(z) + \sum_{i \in N} f_i(x_i, y_i)$, where $f_i(x_i, y_i) = f_i(x_i) + c_i$ if $y_i = 1$ and $\underline{x}_i \leq x_i \leq \bar{x}_i$, $f_i(0, 0) = 0$, and $f_i(x_i, y_i) = \infty$ otherwise. The *convex envelope* of $f_i(x_i, y_i)$ is known [4] to be $\tilde{f}_i(x_i, y_i) = y_i f_i(x_i/y_i) + c_i y_i$—the *perspective function* of f_i—which suggests the *Perspective Reformulation* of (P)

$$\text{(PR)} \qquad \min \left\{ \; h(z) + \textstyle\sum_{i \in N} \tilde{f}_i(x_i, y_i) \; : \; (2) \; , \; (x, y, z) \in \mathcal{P} \; \right\} \; .$$

As f_i is convex, $\tilde{f}_i$ is convex for $y_i \geq 0$; since $x_i = 0$ if $y_i = 0$, $\tilde{f}_i$ can be extended by continuity assuming $0 f_i(0/0) = 0$. Hence, (PR) is a convex MINLP if (P) is. Its continuous relaxation ($\underline{\text{PR}}$)—the *Perspective Relaxation* of (P)—usually has $\nu(\underline{\text{PR}}) \gg \nu(\underline{\text{P}})$, making (PR) a more convenient formulation [8, 9]. If f_i is SOCP-representable then so is $\tilde{f}_i$, hence the PR of a Mixed-Integer Second-Order Cone Program (MI-SOCP) is still a MI-SOCP. Hence, ($\underline{\text{PR}}$) is not necessarily more complex to solve—and, sometimes, even less so [2]—than ($\underline{\text{P}}$). Alternatively, one can consider a Semi-Infinite MINLP (SI-MINLP) reformulation of (PR), where *Perspective Cuts* [4]—linear outer approximations of $\tilde{f}_i$—are dynamically added. This is often the best approach [6], in particular for "general" (P) where no other structure is available. It is appropriate to remark that the (PR) approach also applies if the x_i are *vectors* such that $y_i = 0 \Longrightarrow x_i = 0$ and $y_i = 1 \Longrightarrow x_i \in \mathcal{X}_i$, with $\mathcal{X}_i$ a polytope; yet, here, as in [1, 7], each x_i must be a single variable.

It is clearly useful that solving ($\underline{\text{PR}}$) be not too much more time consuming than solving ($\underline{\text{P}}$), despite the fact that $\tilde{f}_i$ is "more complex" than f_i, in order not to negate the advantage corresponding to a tighter bound. This trade-off is nontrivial, in particular if f_i is "simple". For instance, if f_i is quadratic and everything else is linear, (P) is a Mixed-Integer Quadratic Program (MIQP) whereas (PR) is a MI-SOCP; hence, ($\underline{\text{P}}$)—a QP—can be significantly cheaper to solve than ($\underline{\text{PR}}$)—a SOCP. The *Projected PR* (P^2R) idea underpinning the approach studied here was indeed proposed in [7] for the quadratic case, and $\underline{x}_i \geq 0$. It was then extended in [1] to a more general class of functions, and allowing $\underline{x}_i < 0$. However, $\underline{x}_i < 0 < \bar{x}_i$ renders some of the arguments significantly more complex, hence for the sake of simplicity we will only deal here with the case where $\underline{x}_i \geq 0$, the extension to the more general one being immediate. The P^2R idea is to analyze $\tilde{f}_i$ *as a function of* x_i *only*, i.e., projecting away y_i: under appropriate assumptions, *and if there are no linking constraints* (2), this turns out to be a piecewise-convex functions with a "small" number of pieces, that can be characterized by just looking at the data of (P). Hence, ($\underline{\text{PR}}$) can be reformulated in terms of piecewise-convex objective functions, which makes it easier to solve, especially when $\mathcal{O}$ has some valuable structure (e.g., flow or knapsack) [7]. However, in several applications (2) are indeed present. Furthermore, since the binary variables y_i are removed from the formulation, branching has to be done "indirectly" in P^2R, which rules out off-the-shelf solvers. To overcome these two limitations, in [1] the *Approximated* P^2R (AP^2R) reformulation has been proposed whereby the y_i, after having been eliminated, are re-introduced in the formulation in order to encode the piecewise nature of $\tilde{f}_i$. This is possible even if (2) are present, and it has the advantage that (AP^2R) still is a MIQP if (P) is. However, $\nu(\underline{\text{AP}^2\text{R}}) = \nu(\underline{\text{PR}})$ only if (2) is empty; otherwise, it yields weaker bounds (whence the "Approximate" moniker). This is advantageous in some cases, but it may happen that the weaker bounds outweigh the faster solution time, making the approach not competitive with more straightforward implementations of the PR [1].

Aim of this paper is to improve the AP^2R by presenting a simple and effective way to ensure that $\nu(\underline{\text{AP}^2\text{R}}) = \nu(\underline{\text{PR}})$ even if (2) is nonempty, while keeping the shape of the formulation—and therefore, hopefully, the cost of ($\underline{\text{AP}^2\text{R}}$)—exactly the same. Since the bound at intermediate nodes of the enumeration tree can be weaker that that of ($\underline{\text{PR}}$), it is not obvious that the approach, despite the quicker solution

times of (AP^{2}R), is competitive. However, this is shown to be true in at least one relevant application, the Mean-Variance problem (with min buy-in and cardinality constraints) in portfolio optimization.

2 A quick overview of AP^{2}R

We now quickly summarize the analysis in [1], albeit limited to the case $\underline{x}_i \geq 0$, in order to prepare the ground for the new extension. Since we only consider *one* pair (x_i, y_i) (and the corresponding constraints) at a time, we drop the index "i" and we consider the problem

$$\min \left\{ \; f(x) + cy \; : \; \underline{x}y \leq x \leq \bar{x}y \; , \; y \in \{0,1\} \; \right\} \; . \tag{5}$$

The analysis starts by recasting the (PR) of (5) as

$$\min_x \left\{ \; p(x) = \min_y \left\{ \; \tilde{f}(x,y) \; : \; \underline{x}y \leq x \leq \bar{x}y \; , \; y \in [0,1] \; \right\} \; : \; x \in [0, \bar{x}] \; \right\} \; , \tag{6}$$

i.e., first minimizing $\tilde{f}(x,y)$ with respect to y, and then minimizing the resulting function $p(x)$ with respect to x. This is particularly attractive if y does not appear anywhere but in (4), for then the bound remains the same. This happens in some relevant applications [7]; however, the reformulation is possible also when (2) is not empty, even if at the cost of a weaker bound [1]. The function $p(x)$ is convex, and can be characterized by studying the optimal solution $y^*(x)$ of the inner problem in (6). Differentiability of f now yields that $y^*(x)$ can be easily found by assuming that the first-order optimality conditions

$$c + f(x/y) - f'(x/y)x/y = 0 \tag{7}$$

only have (at most) one solution, whose dependency on y is easy:

Property 1 (7) *has at most one solution for $x \geq 0$, which has the form $\tilde{y}(x) = gx$.*

In Property 1, $g \geq 0$ is a constant that can be determined by the data of the problem. For instance, (7) for the quadratic case $f(x) = ax^2 + bx$ is $c - ax^2/y^2 = 0$, whence $\tilde{y}(x) = |x|\sqrt{a/c}$ if $c > 0$, and there is no solution otherwise. Property 1 is satisfied by a surprisingly large set of functions, and a more general version can be stated for the case $\underline{x} < 0$ [1]. If (7) has a solution then $y^*(x)$ can be found by projecting $\tilde{y}(x)$ over the feasible set $[\, x/\bar{x} \,,\, \min\{\, 1 \,,\, x/\underline{x} \,\}]$; if no solution exists then $y^*(x)$ is in one of the two extremes. In all cases one can then write $p(x) = \tilde{f}(x, y^*(x))$. All this gives that there exists some $\underline{x} \leq \check{x} \leq \bar{x}$ such that

$$p(x) = \left\{ \; (\, f(\check{x})/\check{x} + c/\check{x} \,)x \quad \text{if } 0 \leq x \leq \check{x} \quad , \quad f(x) + c \quad \text{if } \check{x} \leq x \leq \bar{x} \; \right\} \; . \tag{8}$$

Thus, $p(x)$ is piecewise-convex with at most two pieces (although these become four if $\underline{x} < 0$), one of which is linear and the other is the original objective function. The crucial breakpoint $\check{x}$ can be determined a-priori: in particular, $\check{x} = 1/g$ if (7) has a solution and $1/g \in [\underline{x}, \bar{x}]$, and $\check{x} \in \{\, \underline{x} \,,\, \bar{x} \,\}$ otherwise [1]. For a numerical illustration, the quadratic case

$$\min \left\{ \; 2x^2 + 8y \; : \; y \leq x \leq 10y \; , \; y \in \{0,1\} \; \right\} \tag{9}$$

has $\tilde{y}(x) = x\sqrt{a/c} = x/2$, $g = 1/2$, and therefore $1/g = 2$: hence,

$$p(x) = \left\{ \; 8x \quad \text{if } 0 \leq x \leq 2 \quad , \quad 2x^2 + 8 \quad \text{if } 2 \leq x \leq 10 \; \right\} \; . \tag{10}$$

Writing (8) as the objective function is typically done with the "variable splitting" approach [7], whereby two new variables $0 \leq x' \leq \check{x}$ and $0 \leq x'' \leq \bar{x} - \check{x}$ are introduced such that $x = x' + x''$ (although a different form is sometimes preferable [3]): x' gets the linear cost, while x'' has cost $f(x'')$. This yields a reformulation with the same form of the original problem (say, a MIQP if (P) was one), with at most (and, often, less than) twice as many variables. Such a reformulation might be more efficient to solve, especially if (P) has some structure that allows application of specialized approaches [7].

However, removing the y_i variables from the formulation prevents from using off-the-shelf software to solve the integer problem. This is why in [1] it was proposed to "lift back" (8) in the original (x, y) space by using y in place of x' as follows:

$$p(x) = \left\{ \begin{array}{ll} \min_{y, x''} & yp(\check{x}) + f(x'' + \check{x}) + c - p(\check{x}) \\ & (\underline{x} - \check{x})y \leq x'' \leq (\bar{x} - \check{x})y \; , \; x = \check{x}y + x'' \; , \; y \in [0,1] \end{array} \right. \; . \tag{11}$$

Adding an integrality constraint $y \in \{0,1\}$ to (11) yields a *reformulation* of (5): for integer values of y the two are equivalent, but typically $\nu(11) \gg \nu(\underline{5})$. For illustration, consider (9): plugging (10) into (11) gives

$$\left\{ \begin{array}{ll} \min & [\,2(x''+2)^2+8\,] + 16y - 16 = 2(x'')^2 + 8x'' + 16y \\ & -y \le x'' \le 8y \;\;,\;\; x = 2y + x'' \;\;,\;\; y \in \{0,1\} \end{array} \right. . \tag{12}$$

It can be verified that $\nu(\underline{12}) \ge \nu(\underline{9})$ for any fixed x. For instance, for $x = 2$ the optimal solution to ($\underline{9}$) is $y = 1/5$, yielding $\nu(\underline{9}) = 9 + 3/5$, while the optimal solution to ($\underline{12}$) is $(y, x'') = (1, 0)$, yielding $\nu(\underline{12}) = 16$. This latter estimate is the same as the one provided by the ($\underline{\text{PR}}$) (for $x = 2$): in fact,

$$\nu(\underline{\text{PR}}) = \min\left\{\, 2x^2/y + 8y \;:\; y \le x \le 10y \;\;,\;\; y \in [0,1] \,\right\} = 16$$

since $\min\{\, 8y + 8/y \;:\; y \in [1/5, 1] \,\}$ has optimal solution $y = 1$.

We will denote by (AP^2R) the reformulation of (P) where (11) is separately applied to each block $i \in N$. If (2) is empty, $\nu(\underline{\text{AP}^2\text{R}}) = \nu(\underline{\text{PR}})$, whereas in presence of linking constraints (6) is a *relaxation* of the true projection problem, which, besides on x, also depends on all the other variables that y is linked with. Hence, $\nu(\underline{\text{AP}^2\text{R}}) < \nu(\underline{\text{PR}})$ can happen, and it does in practice. For illustration consider the problem

$$\text{(P)} \quad \min\; 2x_1^2 + 2x_2^2 + 8y_1 + 8y_2 \tag{13}$$

$$y_1 \le x_1 \le 10y_1 \quad , \quad y_2 \le x_2 \le 10y_2 \tag{14}$$

$$y_1 \in \{0,1\} \;\;,\;\; y_1 + y_2 = 1 \;\;,\;\; y_2 \in \{0,1\} \;\;,\;\; x_1 + x_2 = 8 \tag{15}$$

obtained by "duplicating" (9) and adding the linking constraint $y_1 + y_2 = 1$. The optimal solution of ($\underline{\text{P}}$) is $x_1 = x_2 = 4$, $y_1 = y_2 = 1/2$, yielding $\nu(\underline{\text{P}}) = 72 \ll \nu(\text{P}) = 136$, the latter obtained by setting $x_1 = 8$, $y_1 = 1$, $x_2 = y_2 = 0$ (or the symmetric solution). The ($\underline{\text{PR}}$), obtained by replacing (13) with

$$\min\; 2x_1^2/y_1 + 2x_2^2/y_2 + 8y_1 + 8y_2$$

has the same optimal solution as ($\underline{\text{P}}$): however, that same solution yields the much stronger (in fact, exact) bound of 136. The (AP^2R) is instead

$$\begin{array}{l} \min\; 2(x_1'')^2 + 2(x_2'')^2 + 8x_1'' + 8x_2'' + 16y_1 + 16y_2 \\ \quad -y_1 \le x_1'' \le 8y_1 \;\;,\;\; -y_2 \le x_2'' \le 8y_2 \;\;,\;\; x_1 = 2y_1 + x_1'' \;\;,\;\; x_2 = 2y_2 + x_2'' \;\;, \end{array} \tag{15}$$

(cf. (12)). The optimal solution of ($\underline{\text{AP}^2\text{R}}$) is $x_1 = x_2 = 4$, $y_1 = y_2 = 1/2$, $x_1'' = x_2'' = 3$, yielding $\nu(\underline{\text{P}}) = 72 \ll \nu(\underline{\text{AP}^2\text{R}}) = 100 \ll \nu(\underline{\text{PR}}) = 136$. In the next section we modify the AP^2R to increase its lower bound, avoiding the bound disadvantage with the (PR)—at least at the root node—while retaining the simpler (hence, cheaper) model shape.

3 Improving AP^2R using dual information

The idea is to reformulate (P) to include information about the linking constraints (2) in the objective function (1), so that it can be "processed" by the AP^2R. This hinges on the availability of dual information: assuming for simplicity that $C(z) = Cz$ is linear, the Lagrangian relaxation of (P) w.r.t. (2)

$$\min\; \left\{\, h(z) + \textstyle\sum_{i \in N} f_i(x_i) + c_i y_i + \lambda\big(\, Ax + By + Cz - b \,\big) \;:\; (x,y,z) \in \mathcal{P} \,\right\} \tag{16}$$

has an objective function that is still separable in the x_i

$$[-\lambda b +] \quad h(z) + \lambda Cz + \textstyle\sum_{i \in N} \big(\, f_i(x_i) + \lambda A^i x_i + (c_i + \lambda B^i) y_i \,\big) \;\; . \tag{17}$$

Hence one can apply $\underline{\text{PR}}$ to (16), which simply yields the modified objective

$$[-\lambda b +] \quad h(z) + \lambda Cz + \textstyle\sum_{i \in N} \big(\, y_i f_i(x_i/y_i) + \lambda A^i x_i + (c_i + \lambda B^i) y_i \,\big) \tag{18}$$

(note that the perspective function does not change linear functions). Thus, the ($\underline{\text{PR}}$) of (16) is

$$\phi(\lambda) = \min\; \big\{\, (18) \;:\; (3)\,,\; (4)\,,\; y \in [0,1]^n \;\;,\;\; x \in \mathbb{R}^n \;\;,\;\; z \in \mathbb{R}^q \,\big\}\, [-\lambda b] \;\; . \tag{19}$$

It is well known that the corresponding Lagrangian dual satisfies $\max_\lambda\{\, \phi(\lambda) \,\} = \nu(\underline{\text{PR}})$, due to convexity. In particular, the optimal dual solution λ^* satisfies $\phi(\lambda^*) = \nu(\underline{\text{PR}})$, and it is available at the cost of solving ($\underline{\text{PR}}$), since any solver provides dual information at termination. Also, (17) clearly satisfies Property 1 if f_i does: (7) for $f_i(x_i) + \lambda A^i x_i$ only differs from (7) for f_i for the constant λA^i. Hence one can apply AP^2R to (19), for which the following result holds:

Theorem 2 *With $\lambda = \lambda^*$, consider the reformulation of* (P)

$$\min \left\{ (17) \ : \ (2) \ , \ (x, y, z) \in \mathcal{P} \right\} [-\lambda^* b] \tag{20}$$

and denote by (AP^{2}R+) *its* AP^{2}R*; then,* $\nu(\underline{\text{AP}^2\text{R+}}) = \nu(\underline{\text{PR}})$.

Proof. First of all, (20) is a *valid reformulation* of (P). In fact, unlike (16), it contains (2); hence, the Lagrangian term $\lambda^* \big(Ax + By + Cz - b \big)$ is always null. Therefore, $\nu(\text{P}) = \nu(\text{AP}^2\text{R+})$. However, note that (1) and (17) are different (for $\lambda \neq 0$): in fact, the equivalence between the two optimal values only holds when taking into account the constant term $-\lambda^* b$. Since (16) is a relaxation of (20), the same holds for their (AP^{2}R), and hence for their ($\underline{\text{AP}^2\text{R}}$). Therefore, $\nu(\underline{\text{AP}^2\text{R+}}) \geq \phi(\lambda^*)$: in fact, for any λ the ($\underline{\text{AP}^2\text{R}}$) and the ($\underline{\text{PR}}$) of (16) are equivalent. Hence $\nu(\underline{\text{AP}^2\text{R+}}) \geq \nu(\underline{\text{PR}})$, and the inverse inequality is obvious. ■

To illustrate Theorem 2, consider again (13)–(15). The optimal dual multiplier of the linking constraint $y_1 + y_2 = 1$ in the ($\underline{\text{PR}}$) is $\lambda^* = 120$. Hence, (16) is

$$\min \left\{ 2x_1^2 + 2x_2^2 + 128y_1 + 128y_2 \ : \ (14) \ , \ (15) \right\} \ [-120] \ .$$

In its AP^{2}R, $c = 128$ gives $g = \sqrt{a/c} = \sqrt{2/128} = 1/8$, i.e., $\check{x} = 8$. Hence,

$$p(x) = yp(\check{x}) + f(x'' + \check{x}) + c - p(\check{x}) = 256y + 2(x'')^2 + 32x'' \ ,$$

and the (AP^{2}R+) is

$$\begin{aligned} \min \ & 2(x_1'')^2 + 2(x_2'')^2 + 32x_1'' + 32x_2'' + 256y_1 + 256y_2 \\ & -7y_1 \leq x_1'' \leq 2y_1 \ , \ -7y_2 \leq x_2'' \leq 2y_2 \ , \ x_1 = 8y_1 + x_1'' \ , \ x_2 = 8y_2 + x_2'' \ , \ (15) \end{aligned}$$

The optimal solution is $x_1 = x_2 = 4$, $y_1 = y_2 = 1/2$, $x_1'' = x_2'' = 0$, giving an optimal value of 256: counting the constant $-\lambda^* b = -120$, this finally gives $\nu(\underline{\text{AP}^2\text{R+}}) = 136$: (much) better than $\nu(\underline{\text{AP}^2\text{R}}) = 100$, and in fact precisely equal to $\nu(\underline{\text{PR}})$ as predicted.

We end this section by remarking that the assumptions that (2) is an equality constraint and that $A(x) = Ax$ and $C(z) = Cz$ are linear can be relaxed somewhat:

1. Inequality linking constraints $Ax + By + Cz \leq b$ can be transformed into equalities by the addition of slack variables: $Ax + By + Cz + s = b$, $s \geq 0$. However, note that this has to be done in (20), so that after the reformulation they will have cost $\lambda^* s$ in the objective function (i.e., they no longer will be slack variables). In this case nothing prevents $C(z)$ from being a (convex) nonlinear function.

2. If $A(x)$ is nonlinear in x it needs to have the same structure as (1), i.e., $A(x) = \sum_{i \in N} A^i(x_i)$ with each $A^i(\cdot)$ convex. In order for (2) to be convex they necessarily have to be inequalities; hence, the optimal dual multipliers λ^* will be non-negative, and therefore $\lambda^* A^i(x_i)$ will also be convex. Assuming that Property 1 holds for $f_i(x_i) + \lambda^* A^i(x_i)$, the approach readily extends.

4 Computational results

In this section we report results of computational tests of the proposed approach for the Mean-Variance cardinality-constrained portfolio optimization problem on n risky assets

$$\text{(MV)} \qquad \min \left\{ x^T Q x \ : \ \sum_{i \in N} x_i = 1 \ , \ \sum_{i \in N} \mu_i x_i \geq \rho \ , \ \sum_{i \in N} y_i \leq k \ , \ (4) \right\} \ ,$$

where μ is the vector of expected unitary returns, ρ is the prescribed total return, Q is the variance-covariance matrix, and $k \leq n$ is the maximum number of purchasable assets. Without the cardinality constraint ($k = n$), (MV) is well suited for AP^{2}R: the bound is the same as that of the PR, and the computation time per node is greatly reduced with respect to the Perspective Cut (P/C) technique. While AP^{2}R is competitive also for $k \ll n$, it becomes less so as the quality of the bound significantly deteriorates [1]. Hence, (MV) is a promising application for AP^{2}R+. Since (MV) is a *non-separable* MIQP, a diagonal matrix D has to be determined such that $Q - D$ is positive semidefinite: the PR technique is applied to $\sum_{i \in N} D_{ii} x_i^2$, leaving the remaining part $x^T (Q - D) x$ untouched. Choosing D is nontrivial: one can use e.g., a "small" SDP as advocated in [5], or a "large" SDP as proposed in [10]. We

denote these two by D_s and D_l. Although D_l provides a better root node bound, it is not necessarily the best choice throughout the enumeration tree: sometimes a convex combination between the two, denoted by D_c, works better [10].

For our tests we used the 90 randomly-generated instances, 30 for each value of $n \in \{200, 300, 400\}$, already employed in [1, 4, 5, 6, 10] to which the interested reader is referred for details. Here we only remark that "$^+$" instances are strongly diagonally dominant, "^{0}" ones are weakly diagonally dominant, and "$^-$" ones are not diagonally dominant; the less diagonally dominant, the harder an instance is. We have set $k = 10$, as in [1, 10]; this is a "tight" value, since the maximum number of assets that the model can choose, due to the lower limits $\underline{x}_i > 0$, without the cardinality constraint is ≈ 20 for all n. The (MV) instances and the diagonals used in the experiments are available at `http://www.di.unipi.it/optimize/Data/MV.html`.

The experiments have been performed on a computer with a 3.40 Ghz 8-core Intel Core i7-3770 processor and 16Gb RAM, running a 64 bits Linux operating system. All the codes were compiled with `g++` (version 4.8.4) using `-O3` as optimization option. We have tested AP^{2}R+ vs. AP^{2}R using `Cplex 12.6.0`, single-threaded, with all default parameters (save for one explicitly described below, and only for the tests with the D_l diagonal). We have obtained the (PR) root node bound with the P/C technique, implemented through callbacks, which is typically the best choice when AP^{2}R is not available [5]; hence, for completeness we also report results for the full B&C using P/C. Since we are not interested in comparing the cost/effectiveness of the different diagonal choices, this having been done in [10], we don't report detailed SDP times beyond saying that the "small" SDP requires on average about 0.2, 0.7, and 1.6 seconds while the "large" SDP requires 9, 21 and 47 seconds, respectively for $n = 200$, 300 and 400 (with little variance for the same n).

The results are reported in Table 1, 2 and 3 for the three diagonals D_s, D_c, and D_l, respectively. In the tables we report the (average) total B&C time and root time when using P/C. For AP^{2}R and AP^{2}R+ we report the (average) total number of B&C nodes, total B&C time, root node time and root gap (in percentage). As predicted by Theorem 2 the root node gap of AP^{2}R+ and P/C was identical, which is why we do not report it for P/C. The total time of AP^{2}R+ already includes the P/C root time, since it is needed to compute λ^* prior to performing the reformulation, and therefore starting the AP^{2}R+ B&C.

	P/C		AP^{2}R				AP^{2}R+			
	time		nodes	time		root	nodes	time		root
	tot	root		tot	root	gap		tot	root	gap
200^+	3.68	0.27	212	0.43	0.19	0.77	116	0.59	0.20	0.51
200^0	153.75	0.27	24868	22.01	0.21	3.14	9423	9.40	0.20	2.75
200^-	674.13	0.29	173844	157.54	0.23	4.75	40225	38.47	0.18	4.17
300^+	18.02	0.74	1303	2.66	0.81	1.08	322	1.79	0.65	0.49
300^0	824.02	0.83	69706	109.02	0.83	2.91	20006	33.77	0.78	2.34
300^-	3409.24	0.74	440656	704.32	0.82	3.92	85997	143.66	0.89	3.57
400^+	28.39	1.65	985	3.68	1.75	0.85	184	3.17	1.62	0.41
400^0	3608.04	1.70	329242	849.38	1.25	3.00	48967	129.54	1.90	2.34
400^-	27824.09	1.70	1821932	4769.89	1.28	4.53	334612	856.08	1.38	3.80

Table 1: Results with diagonal D_s

Tables 1 and 2 show that AP^{2}R+ is highly competitive with AP^{2}R, and *a fortiori* with P/C, reducing total time of up to an order of magnitude. While the exact ratio depends on n, the type of instance and the diagonal, the trend is clear: the improvement is due to the much reduced number of nodes, itself a consequence of the much improved bound, while the computing time per node remains the same. The results are somewhat different for D_l, which therefore requires separate discussion. In Table 3, although the nodes count does decrease, the running time does not nearly as much, and can actually increase. While the average time per node of AP^{2}R and AP^{2}R+ is almost identical for D_s and D_c, for D_l that of AP^{2}R+ is roughly an order of magnitude larger. Investigating the issue showed that D_l causes `Cplex` to change the (automatic) selection of the relaxation algorithm at the nodes, settling to one that turns out to be much less efficient. Tests determined that setting `CPX_PARAM_SUBALG = 5`, i.e., using the "sifting" approach, restored an average time per node similar to that of AP^{2}R. This is why Table 3 reports, besides AP^{2}R+, also "AP^{2}R++" which is just obtained changing that parameter. To save on space, root times

	P/C		AP^{2}R				AP^{2}R+			
	time		nodes	time		root	nodes	time		root
	tot	root		tot	root	gap		tot	root	gap
200^+	3.07	0.27	131	0.37	0.13	0.59	75	0.55	0.12	0.27
200^0	41.80	0.28	6693	7.08	0.13	1.91	1831	2.50	0.11	1.44
200^-	176.34	0.29	43940	40.66	0.14	3.01	6031	6.78	0.12	2.33
300^+	8.84	0.80	539	1.73	0.35	1.02	121	1.52	0.32	0.23
300^0	128.95	0.81	21354	38.47	0.33	1.86	3229	7.13	0.32	1.14
300^-	783.38	0.83	130738	229.55	0.37	2.49	16870	31.58	0.36	2.00
400^+	12.62	1.69	512	2.78	0.75	0.88	56	2.86	0.71	0.21
400^0	442.64	1.79	88124	240.84	0.72	2.01	7071	22.71	0.72	1.19
400^-	2663.34	1.87	368468	1060.82	0.72	2.97	38104	108.04	0.75	2.00

Table 2: Results with diagonal D_c

for AP^{2}R$*$ are not reported; they are, however, similar, even between AP^{2}R+ and AP^{2}R++, since it was subproblem time at the inner nodes that made a difference. The table shows that AP^{2}R++ is competitive w.r.t. AP^{2}R, albeit with a somewhat smallest ratio. This is due to another frankly unfathomable—but not unheard-of with today's complex MIP solvers—phenomenon: just by changing the relaxation solver, the number of nodes significantly increases w.r.t. AP^{2}R+. Yet, on the largest and hardest instances AP^{2}R++ enumerates a third of the nodes of AP^{2}R, with the corresponding time advantage.

	P/C		AP^{2}R			AP^{2}R+		AP^{2}R++		
	time		nodes	time	root	nodes	time	nodes	time	root
	tot	root		tot	gap		tot		tot	gap
200^+	3.81	0.51	129	0.83	0.93	65	1.79	480	1.76	0.09
200^0	8.24	0.63	873	2.34	1.52	168	3.52	947	3.28	0.38
200^-	20.68	0.77	5944	9.36	2.12	473	4.94	2324	6.33	0.77
300^+	9.70	2.34	749	4.60	1.89	14	3.91	217	3.92	0.03
300^0	22.33	2.93	2091	9.30	2.00	195	14.75	1520	11.58	0.18
300^-	49.85	2.87	13011	38.05	2.04	1403	55.34	9704	42.69	0.64
400^+	23.30	5.85	1116	11.01	1.94	14	9.05	190	8.61	0.06
400^0	150.04	6.62	10420	60.60	2.22	364	59.95	3092	32.84	0.26
400^-	393.69	7.92	21143	131.86	2.65	1921	254.94	6430	64.97	0.48

Table 3: Results with diagonal D_l, default and with option `CPX_PARAM_SUBALG = 5`

5 Conclusions

The main advantage of the proposed AP^{2}R+ technique is its simplicity: just solving (PR)—possibly even approximately with a dual approach—produces the dual solution λ^* which can be used to first construct (20) and then compute its AP^{2}R (an inexpensive task). Yet, this improves many-fold the performances over plain AP^{2}R, and even more so over P/C. Notably, AP^{2}R+ is quite general and applies to a much larger class than MIQP. It may be worth contrasting 27824 seconds (P/C in Table 1) with 65 seconds (AP^{2}R++ in Table 3) for 400^-: this is over two orders of magnitude difference for solving the same instances with the same underlying solver, and the gap with the standard MIQP formulation would be even more humungous. This nicely illustrates the power of reformulation techniques like AP^{2}R+.

Acknowledgements

The first and third authors acknowledge the contribution of the Italian Ministry for University and Research under the PRIN 2012 Project 2012JXB3YF "Mixed-Integer Nonlinear Optimization: Approaches and Applications". All the authors would like to acknowledge networking support by the COST Action TD1207.

References

[1] A. Frangioni, F. Furini, and C. Gentile. Approximated Perspective Relaxations: a Project&Lift Approach. *Computational Optimization and Applications*, 63(3):705–735, 2016.

[2] A. Frangioni, L. Galli, and M.G. Scutellà. Delay-Constrained Shortest Paths: Approximation Algorithms and Second-Order Cone Models. *Journal of Optimization Theory and Applications*, 164(3):1051–1077, 2015.

[3] A. Frangioni, L. Galli, and G. Stea. Delay-constrained routing problems: Accurate scheduling models and admission control. Technical report, Dipartimento di Informatica, Università di Pisa, 2015.

[4] A. Frangioni and C. Gentile. Perspective Cuts for 0-1 Mixed Integer Programs. *Mathematical Programming*, 106(2):225–236, 2006.

[5] A. Frangioni and C. Gentile. SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP. *Operations Research Letters*, 35(2):181 – 185, 2007.

[6] A. Frangioni and C. Gentile. A Computational Comparison of Reformulations of the Perspective Relaxation: SOCP vs. Cutting Planes. *Operations Research Letters*, 37(3):206 – 210, 2009.

[7] A. Frangioni, C. Gentile, E. Grande, and A. Pacifici. Projected Perspective Reformulations with Applications in Design Problems. *Operations Research*, 59(5):1225–1232, 2011.

[8] A. Frangioni, C. Gentile, and F. Lacalandra. Tighter Approximated MILP Formulations for Unit Commitment Problems. *IEEE Transactions on Power Systems*, 24(1):105–113, 2009.

[9] O. Günlük and J. Linderoth. Perspective Relaxation of MINLPs with Indicator Variables. In A. Lodi, A. Panconesi, and G. Rinaldi, editors, *Proceedings 13^{th} IPCO*, volume 5035 of *Lecture Notes in Computer Science*, pages 1–16, 2008.

[10] X. Zheng, X. Sun, and D. Li. Improving the Performance of MIQP Solvers for Quadratic Programs with Cardinality and Minimum Threshold Constraints: A Semidefinite Program Approach. *INFORMS Journal on Computing*, 26(4):690–703, 2014.