Università di Pisa
Dipartimento di Informatica

Technical Report: TR-06-01

SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP

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

Claudio Gentile
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", C.N.R.
Viale Manzoni 30, 00185 Rome – Italy
gentile@iasi.cnr.it

Address: via Largo B. Pontecorvo 3, 56127 Pisa, Italy. Tel: +39 050 2212700 Fax: +39 050 2212726

SDP Diagonalizations and Perspective Cuts for a Class of Nonseparable MIQP

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

Claudio Gentile*
Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti", C.N.R.
Viale Manzoni 30, 00185 Rome – Italy
gentile@iasi.cnr.it

Abstract

Perspective cuts are a computationally effective family of valid inequalities, belonging to the general family of disjunctive cuts, for Mixed-Integer Convex NonLinear Programming problems with a specific structure. The required structure can be forced upon models that would not originally display it by decomposing the Hessian of the problem into the sum of two positive semidefinite matrices, a generic and a diagonal one, so that the latter is "as large as possible". We compare two ways for computing the diagonal matrix: an inexpensive approach requiring a minimum eigenvalue computation and a more costly procedure which require the solution of a SemiDefinite Programming problem. The latter dramatically outperforms the former at least upon instances of the Mean-Variance problem in portfolio optimization.

Keywords: *Mixed-Integer Quadratic Programs, Valid Inequalities, SemiDefinite Programming, Portfolio Optimization*

*This author has been partly supported by the UE Marie Curie Research Training Network no. 504438 ADONET

1 Introduction

Perspective cuts are a family of valid inequalities, belonging to the general family of disjunctive cuts [1], for Mixed-Integer NonLinear Programming (MINLP) problems which exhibit blocks of the form

$$\min \left\{ f(x,y) = f(x) + cy \ : \ ly \le x \le uy \ , \ y \in \{0,1\} \right\} \qquad (1)$$

where $f(x)$ is (closed) convex. That is, x is a *semi-continuous* variable whose domain is the disconnected set $0 \cup [l,u]$ (if $0 \notin [l,u]$), and/or one has a *fixed-charge* cost function whereby one pays the fixed cost c whenever $x \neq 0$ (if $c \neq 0$). Actually, the approach can be extended to more general cases where several x variables depend on the same y and the linking constraints are different [6], but for the sake of clarity of the present discussion the simpler form (1) is more appropriate.

Perspective cuts are a way to strengthen the continuous relaxation of (1); they are obtained by computing the *convex envelope* of f (the "best" convex function approximating f) on its domain $(0,0) \cup [l,u] \times \{1\}$, which turns out to be

$$\overline{co} f(x,y) = \begin{cases} 0 & \text{if } x = 0 \text{ and } y = 0 \\ yf(x/y) + cy & \text{if } x \in [l,u] \ , \ y \in (0,1] \\ +\infty & \text{otherwise} \end{cases} .$$

This function (which is strongly related with a well-known object in convex analysis, the perspective function of $f(x)$, whence the name of the cuts) provides a better continuous relaxation to the original problem; this is easily seen in the quadratic case $f(x) = ax^2 + bx$ $(a > 0)$, where one obtains

$$\overline{co} f(x,y) = (1/y)ax^2 + bx + cy > f(x,y)$$

for all $y \in (0,1)$ and feasible x. Note that nonlinearity is required for the approach to have any impact: for $a = 0$, $\overline{co} f = f$.

However, using $\overline{co} f$ as the objective function instead of f has a serious drawback, especially in the "simple" quadratic case, in that $\overline{co} f$ is much "more nonlinear" than f, and it is nondifferentiable at $(0,0)$. The interior-point method of [3] could be used, but efficient implementations of that approach are not widely available, and have not yet been shown to be competitive with the sophisticated QP solvers available; furthermore, interior-point methods are less well-suited than simplex-like methods in the context of enumerative approaches [6], since the latter reoptimize more efficiently.

A possible alternative is to mimic what is done in most NonDifferentiable Optimization algorithms [5], using a *polyhedral approximation* of $\overline{co} f$ as the

objective function; this requires characterizing the subdifferential of $\overline{co}f$ and using the subgradient inequalities in the epigraphical space of f ((v, x, y) such that $v \geq f(x, y)$). After proper analysis, all this boils down to simple formulae; in the quadratic case, for instance, the perspective cut obtained at the feasible (fractional) point (x^*, y^*) is just

$$v \geq (2a\bar{x} + b)x + (c - a\bar{x}^2)y \tag{2}$$

where $\bar{x} = x^*/y^*$. This can be seen as an application of the general lift-and-project approach [1], but it is much easier to implement since there is no need to set up and solve a large-scale, nonlinearly-constrained separation problem, and the obtained cuts are global by nature and do not require lifting. Despite the low dimensionality of the faces that they represent, perspective cuts have been shown to significanlty improve the efficiency of enumerative approaches to Mixed-Integer Quadratic Problems (MIQP) with (many blocks with) structure (1) [6].

Using perspective cuts crucially requires the objective function to be separable among the blocks of semi-continuous variables; yet, there are applications, e.g., in Financial Trading and Planning problems [7], which sport semi-continuous variables and a nonseparable objective function. For these problems, a general reformulation technique was proposed in [6] which selects "the nonseparable part" of the objective function and moves it to newly introduced variables, leaving a separable objective function to which perspective cuts can then be applied. Different ways exist for selecting how to "decompose" the nonseparable objective function; in [6] a simple procedure, based on computing the minimum eigenvalue of the Hessian, was found to already provide good enough results to prove the worthiness of the approach. In this paper, we propose a more sophisticated—but still relatively simple to implement—procedure which requires the solution of a SemiDefinite Programming problem, and we report about the impact of using the resulting approach upon instances of the Mean-Variance problem with minimum and maximum buy-in thresholds in portfolio optimization. Our results show that the significantly larger cost of reformulating the instance in a "better" way dramatically pays off in terms of quality of the obtained lower bounds, allowing us to routinely solving instances of much larger size than those solvable (within the same time limit) with the simpler approach.

2 The reformulation technique

Assume the following (MIQP) is given

$$\begin{array}{ll} \min & x^T Q x + qx + cy \\ & Ax + By \geq b \\ & l_i y_i \leq x_i \leq u_i y_i \ \ , \ \ y_i \in \{0,1\} \quad i = 1, \ldots, n \end{array} \tag{3}$$

where Q is positive semidefinite (and not diagonal). Perspective cuts (2) cannot be directly applied to (3), as the quadratic objective function is nonseparable on the x variables. In [6] a decomposition technique of the objective function was proposed: select any non-negative diagonal $D \in \mathbb{R}^{n \times n}$ such that $Q - D$ still is positive semidefinite, replace $x^T Q x$ in the objective function with $x^T D x + z^T (Q - D) z$ and add contraints enforcing $x = z$. The resulting model is

$$\begin{array}{ll} \min & x^T D x + z^T (Q - D) z + qx + cy \\ & Ax + By \geq b \ \ , \ \ z = x \\ & l_i y_i \leq x_i \leq u_i y_i \ \ , \ \ y_i \in \{0,1\} \qquad i = 1, \ldots, n \end{array} \tag{4}$$

Model (4) is directly amenable to application of perspective cuts, and retains most of the structure of the original problem by just introducing a copy of the x variables and assigning it all the non-separability in the objective function. Intuitively, the "larger" D (the "fraction" of the overall objective function that is properly reflected on the separable costs) is, the more perspective cuts could be expected to improve the lower bound. As we will see, this is clearly confirmed by the computational results. Thus, procedures have to be devised for (efficiently) finding a "large" D.

In the context of the original experiments of [6], aimed at evaluating the effectiveness and efficiency of perspective cuts in general, a simple rule was used: computing the minimum eigenvalue λ_{min} of Q and setting $D = \lambda_{min} I$. There are many efficient ways for computing the minimum eigenvalue of a symmetric matrix; for our experiments, we have chosen to just compute all the eigenvalues of Q using the `eig()` function of the open-source package `octave 2.1` [4], and then extract the smallest one. We will refer to this as the Minimum Eigenvalue (ME) approach. More efficient methods for finding the minimum eigenvalue exist, but this is irrelevant for our application because the time required for this operation is negligible with respect to the total time of the B&C approach (cf. Table 1). We remark that the ME approach obviously requires Q to be *strictly* positive definite (for otherwise $\lambda_{min} = 0 \Rightarrow D = 0$); although this was the case in all the instances of our

test set, there may exist cases of nonseparable (MIQP) with non-full-rank Hessian matrix, to which the ME approach could not be applied.

More sophisticated techniques allow to find "larger" matrices D. In want of a better metric, we assumed $tr(D)$—the sum of the diagonal elements of D—to be a relevant indicator of the quality of D as a diagonal approximation of Q. Thus, finding the largest diagonal approximation that still leaves $Q-D$ positive semidefinite can be cast as the following dual pair of SemiDefinite Programming (SDP) problems

$$(PD) \qquad \max \left\{ \sum_{i=1}^{n} d_i \ : \ Q - \sum_{i=1}^{n} d_i (e_i e_i^T) = R \ , \ R \succeq 0 \ , \ d \geq 0 \right\}$$

$$(DD) \qquad \min \left\{ tr(QX_{11}) \ : \ x_{ii} - x_{i+n,i+n} = 1 \ \ i = 1, \ldots, n \ , \ X \succeq 0 \right\}$$

where $\Gamma \succeq 0$ means that Γ belongs to the cone of symmetric positive semidefinite matrices of proper dimension, e_i is the i-th vector of the canonical base of $\mathbb{R}^n$ and X_{11} is the $n \times n$ principal submatrix of the $2n \times 2n$ matrix X.

The above dual pair of small-scale SDP problems can be solved by means of the currently available interior-point SDP codes; for our tests we used the open-source package `csdp 4.8` [2], which proved to be efficient and reliable and directly delivered the required solution d_i, $i = 1, \ldots, n$ (we remark that (DD) is the primal problem according to the `csdp` standards, so d_i are actually optimal dual variables). We will refer to this as the SemiDefinite Programming (SDP) approach. Clearly, the SDP approach is immediately extended to the case where each of the d_i variables is given a different (nonnegative) weight to indicate different relevance of having a large quadratic coefficient for x_i in the resulting reformulation; however, up to now no sensible rules have been devised for computing those weights, so for our computational results we have always used identical (unitary) weights. Also, note that, unlike ME, the SDP approach can in principle be applied even to cases where Q is not strictly positive definite, i.e., $\lambda_{min} = 0$.

For both SDP and ME, as a further safeguard measure to avoid that $Q - D$ turs out not to be positive semidefinite due to numerical errors, we subtracted from D (hence, added to $Q - D$) a "small" positive definite matrix of form εI for a suitably chosen "small" ε.

3 Computational results

We have tested the influence of the two procedures ME and SDP on the overall efficiency of a B&C approach using perspective cuts to nonseparable

(MIQP) with semi-continuous variables; in particular, as in [6] we have applied the approach to instances of the Mean-Variance (MV) model in portfolio optimization [8], which is as follows.

A set of n risky assets are available; for each asset $i = 1, \ldots, n$, the expected unitary return μ_i for the considered time horizon is known. Also, the $n \times n$ variance-covariance matrix Q defined for the assets is available. Denoting by $x_i \in [0, 1]$ the fraction of the portfolio value invested in asset i, any vector x with $ex = 1$ (e being the vector of all ones) is a feasible allocation of the available resources over the assets, μx is the corresponding expected return and $x^T Q x$ is a measure of the associated risk (volatility). Thus, the problem faced by the "rational investor" is that of trading returns versus risk. With no further constraints, the problem of fixing a desired level of return ρ and minimizing the associated risk is an easy convex (QP); thus, one can effectively trace all the *risk-return efficient frontier*, with a classical procedure that has been considered the very beginning of rational financial analysis. However, in many real cases a number of further constraints over portfolio decisions exist. Typically, *minimum and maximum buy-in thresholds* l_i and u_i are set on each asset i, turning the problem into the much harder (MIQP)

$$\min \left\{ x^T Q x \;\middle|\; \begin{array}{l} ex = 1 \;\;,\;\; \mu x \geq \rho \;, \\ l_i y_i \leq x_i \leq u_i y_i \;,\; y_i \in \{0,1\} \;\; i = 1, \ldots, n \end{array} \right\} \quad . \tag{5}$$

Further constraints can be easily imposed, such as maximum and minimum numbers of purchased assets, or fixed purchase costs can be considered; however, in the following we will stick to the basic formulation (5). This problem is quite demanding for general-purpose (MIQP) solvers, most likely due to the fact that it has very few constraints with basically no structure, so that the classical polyhedral approaches to improve the lower bounds are ineffective. Also, as shown in Table 2 (column "`Cplex`") the root node gaps of the standard continuous relaxation are huge; as a result, instances with $n = 200$ are already practically unsolvable by standard methods.

For our tests, we have generated 10 (MV) instances for each value of $n \in \{200, 300, 400\}$. The variance-covariance matrices Q have been generated using the well-known random generator of [9]. The desired level of return ρ has been randomly chosen in the interval $[0.002, 0.01]$, and the minimum and maximum buy-in thresholds l_i and u_i have been randomly generated in the intervals $[0.075, 0.125]$ and $[0.375, 0.425]$, respectively. The data required for reproducing the instances is available upon request by the authors.

For our experiments we have implemented a B&C based on perspective cuts. This is not entirely straightforward, as some of the required operations—such as changing the quadratic part of the objective function during the execution of the B&C—are not supported by the API of available (MIQP) solvers such as `Cplex`. Furthermore, a number of different algorithmic choices, some unique to perspective cuts and some quite standard, have to made which have an impact on the overall effectiveness of the approach. However, these issues are mostly irrelevant for the present discussion, since a preliminary computational investigation showed that the effect of the different implementation choices is basically the same for both diagonalization procedures ME and SDP; thus, we just followed the guidelines developed in [6], to which the interested reader is referred for further details upon the implementation issues of the B&C approach with perspective cuts.

The experiments were performed on a PC with an Opteron 252 processor and 2Gb RAM, running the Linux Fedora operating system (kernel 2.6.11). The codes were compiled with `gcc 4.0` using aggressive optimizations `-O3`; our B&C code uses `Cplex 9.1` to solve the continuous relaxations at each node of the enumeration tree. We also solved, on the same machine, the same instances with the general-purpose B&C algorithm of `Cplex 9.1`. For each code, we set a global time limit of 10000 seconds.

In Table 1 we report some data that describes the average results of the initialization phase alone on all the instances of each of the three sizes. For both approaches, column "time" report the time (in seconds) required for computing D; then, for SDP columns "d_{max}", "d_{min}" and "d_{avg}" report respectively the maximum, minimum and average element of the diagonal of D, while for ME the column "λ_{min}" reports the minimum eigenvalue (note that $\lambda_{min} = d_{max} = d_{min} = d_{avg}$ for ME). On these instances, SDP finds diagonal elements of D which range from about 0.97 to about 1.97 times the minimum eigenvalue, with average 1.47 times; for doing so, it requires over two orders of magnitude more time than ME.

	SDP				ME	
n	d_{max}	d_{min}	d_{avg}	time	λ_{min}	time
200	3918	1933	2934	7.24	2001	0.13
300	5910	2905	4392	24.08	2996	0.23
400	7894	3887	5886	44.25	3994	0.39

Table 1: Comparison of SDP and ME initializations

Yet, the longer time is well-spent when one consider the total effective-

ness of the B&C approach, as shown in Table 2. For each code and instance, columns "nodes" and "time" report respectively the total number of explored nodes and the total time (in seconds) for the B&C approach, comprised the initialization phase, while columns "r.gap", "p.gap" and "d.gap" report respectively the root node gap and the gap of the best primal solution and the best lower bound attained at the end of the enumerative process (in percentage); a blank entry corresponds to a gap less than 0.01%—the optimality tolerance of the B&C—while a "*" in column "p.gap" means that the algorithm found no feasible solution. We have avoided to report column "time" for `Cplex` since it never terminated before the time limit, as well as columns "p.gap" and "d.gap" for SDP since it solved all the instances to the required precision.

As already seen in [6], the standard continuous relaxation has a huge root node gap, usually in the 80-90% range, that is only reduced to the 25-75% range (with the only exception of the "easy" pard400b instance) within 10000 seconds of the standard B&C. It is worth remarking that, owing to the extremely simple structure of the feasible set, `Cplex` only adds a handful of standard MIP cuts for each instance, so it basically behaves like a "pure" B&B approach. The same simple structure, however, helps the standard MIP rounding heuristics of `Cplex` to attain relatively good primal solutions (although with several exceptions); in comparison, the relatively "more complex" structure created by the perspective cuts makes life more difficult to the rounding heuristics, up to the point that, more often than not, no primal solution at all is obtained by ME. On the other hand, even with ME diagonalization perspective cuts close the root node gap to a much more manageable 5-8%; this allows to solve most of the smallest instances, but it is not enough for the largest ones.

The SDP diagonalization further reduces the root node gap to around 1.5%, making it possible to solve all the instances of our test bed up to $n = 400$. It is worth noting that not all instances are equally difficult, with over two orders of magnitude of difference in time and node count between the easiest and the most difficult ones; on the easy ones, solving the SDP program for computing D takes more than 50% of the total time, but it is clearly time well-spent.

We remark that in [6] (MV) instances were found quite difficult to solve, and thereby a heuristic approach had been developed, based on perspective cuts (and ME initialization), that was often—but not always—capable of providing reasonably accurate solutions for instances with n up to 300. A better choice of the diagonalization approach produces an *exact* B&C algorithm which is blatantly more efficient than our previous approximate

	SDP			ME					Cplex			
	time	nodes	r.gap	time	nodes	r.gap	p.gap	d.gap	nodes	r.gap	p.gap	d.gap
pard200a	93	7615	1.26	9537	695228	5.28			11479562	89.23	0.21	54.53
pard200b	13	357	0.97	1145	77509	6.19			5100085	77.82		27.45
pard200c	24	1182	1.62	349	25777	6.07			4404025	77.49	1.07	26.07
pard200d	10	203	0.75	1364	92785	5.31			3863561	81.34		35.04
pard200e	20	1051	0.69	7246	536692	7.51			11801638	89.65	0.14	55.29
pard200f	20	1035	0.89	1745	126090	5.74			11645001	89.14		54.18
pard200g	16	531	0.85	1025	71814	5.65			3985115	83.45		35.95
pard200h	12	343	0.47	10000	679364	8.57	*	1.04	11708754	89.61		55.66
pard200i	153	12167	2.00	2160	177523	5.81			11594165	88.99		53.72
pard200j	1275	105789	1.89	10000	711113	8.61	*	1.83	11605554	89.54		55.73
pard300a	1419	46607	1.55	10000	272460	4.81	*	0.50	5246001	92.61		66.19
pard300b	64	1305	0.46	10000	263120	5.66	*	1.29	2989191	92.76	0.16	67.53
pard300c	184	5501	1.30	10000	284900	5.64	13.31	0.48	2066676	91.55	2.55	62.56
pard300d	214	6857	1.16	10000	251600	5.96	*	1.32	4598355	92.59		65.62
pard300e	133	3465	0.96	10000	244400	5.18	*	0.57	4739540	92.55		66.06
pard300f	1486	49851	1.61	10000	266789	5.90	*	1.27	5037373	92.64	0.04	66.10
pard300g	48	758	1.14	1091	29957	4.81			1864199	87.94		54.14
pard300h	3293	114694	2.05	10000	275115	5.48	2.09	1.07	5044579	92.67	0.07	66.60
pard300i	353	10624	1.20	10000	259493	6.12	5.46	1.79	2189540	92.18	0.85	66.26
pard300j	2354	75672	1.48	10000	273821	5.89	*	1.02	5077818	92.60	0.23	66.84
pard400a	1577	321082	1.47	10000	122184	6.42	*	3.07	1583601	94.45	0.88	77.01
pard400b	129	660	1.89	228	3516	6.07			918151	63.51		6.48
pard400c	60	341	0.49	10000	147532	6.97		0.29	1080227	86.81		50.70
pard400d	1614	29680	1.23	10000	141442	5.95	*	2.10	1565201	94.35	7.46	75.29
pard400e	9689	169024	1.74	10000	119398	6.02	*	2.16	2838399	94.42		75.69
pard400f	257	1889	1.69	5396	48868	8.09			1405341	80.29	0.19	32.93
pard400g	3617	57793	1.42	10000	140572	5.51	*	1.35	2828635	94.37	0.26	75.84
pard400h	2327	36860	2.07	10000	154412	5.10	2.33	1.02	1108111	93.60	1.69	70.73
pard400i	123	1424	0.51	10000	138389	4.96	*	0.61	2787859	94.40		75.43
pard400j	7238	116128	1.76	10000	130460	6.40	*	1.93	2886807	94.41		75.21

Table 2: Results of the three B&C approaches

approach. We also remark that the much larger root node gaps of [6, Table 2] are *actual* gaps (between the upper and lower bound at the root node) while Table 2 in this paper displays *theoretical* gaps with respect to the optimal solution.

4 Conclusion

We have shown that reformulating (MIQP)s with semi-continuous variables and nonseparable cost matrix is possible in such a way that a B&C approach using perspective cuts can be quite efficient in the solution of large-scale

instances. For this to be true, however, a reasonably sophisticated approach has to be used for extracting "as large as possible" a part of the original nonseparable objective function; luckily, the current available interior-point approaches to SemiDefinite Programming offer reliable and efficient tools for performing this operation. An interesting issue that still remains open is the development of different weighting schemes for the variables in the SDP problem used to construct the reformulation to further improve the performances of the B&C approach.

Acknowledgements

We are grateful to Brian Borchers for maintaining `csdp` and helping us using it.

References

[1] E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[2] B. Borchers. `http://infohost.nmt.edu/~borchers/csdp.html`.

[3] S. Ceria and J. Soares. Convex programming for disjunctive convex optimization. *Mathematical Programming*, 86:595–614, 1999.

[4] J.W. Eaton et al. `http://www.octave.org`.

[5] A. Frangioni. About Lagrangian Methods in Integer Optimization. *Annals of Operations Research*, 139:163–193, 2005.

[6] A. Frangioni and C. Gentile. Perspective Cuts for 0-1 Mixed Integer Programs. *Mathematical Programming*, to appear, 2006.

[7] N.J. Jobst, M.D. Horniman, C.A. Lucas, and G. Mitra. Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. In *Quantitative Finance*, volume 1, pages 1–13. Wiley, Chichester, 2001.

[8] H.M. Markowitz. Portfolio Selection. *Journal of Finance*, 7:77–91, 1952.

[9] P.M. Pardalos and G.P. Rodgers. Computing aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45:131–144, 1990.