A Simplicial Decomposition - Branch and Price for convex quadratic mixed binary problems

Enrico Bettiol[1], Lucas Létocart[2], Francesco Rinaldi[3], and Emiliano Traversi[4]

[1] Fakultät für Mathematik, TU Dortmund, Vogelpothsweg 87, Dortmund, Germany
enrico.bettiol@math.tu-dortmund.de

[2] LIPN UMR CNRS 7030 Université Paris 13, Sorbonne Paris Cité, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
lucas.letocart@lipn.univ-paris13.fr

[3] Università di Padova, Dipartimento di Matematica, Via Trieste, 63, 35121 Padova, Italy
rinaldi@math.unipd.it

[4] LIPN UMR CNRS 7030 Université Paris 13, Sorbonne Paris Cité, 99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
emiliano.traversi@lipn.univ-paris13.fr

Abstract

Simplicial Decomposition (SD) is a column generation method which can be useful for convex quadratic problems. In this work we show that this method can be efficiently exploited to solve quadratic convex mixed binary problems, with linear constraints. We propose a Branch-and-Price algorithm, in which at each node we adapt a SD - based algorithm. This allows to efficiently reuse the variables generated in one node of the Branch-and-bound tree in its child nodes, providing a speed-up in the overall computing time. We also analyze the interaction of additional techniques for speeding up the overall framework. We provide computational results, which show the efficiency of the proposed method for Quadratic Shortest Path Problems, when compared to the commercial solver *Cplex*. We also provide some preliminary results for the Quadratic Minimum Spanning Tree Problem.

Keywords : *Convex Quadratic Programming, Mixed Binary Quadratic Optimization, Simplicial Decomposition, Column Generation, Branch-and-Bound.*

1 Introduction

Many real-world applications can be modelled as *Mixed Binary Quadratic Problems* (MBQPs): this means the optimization of a quadratic objective function subject to linear or quadratic constraints, where all or a part of the variables must be binary numbers, i.e. 0 or 1. We consider general mixed binary problems with quadratic objective function, subject to linear constraints. The form of these problems is the following:

$$\begin{aligned} \min\ & f(x) = x^\top Q x + c^\top x \\ \text{s. t.}\quad & Ax \geq b, \\ & Cx = d, \\ & l \leq x \leq u \\ & x_i \in \{0,1\} \quad \forall i \in I \subseteq \{1,\dots,n\} \end{aligned} \tag{1}$$

with $Q \in \mathbb{R}^{n\times n}$, $c, l, u \in \mathbb{R}^n$, $A \in \mathbb{R}^{m_1\times n}$, $b \in \mathbb{R}^{m_1}$, $C \in \mathbb{R}^{m_2\times n}$, $d \in \mathbb{R}^{m_2}$, $n, m_1, m_2 \in \mathbb{N}$.

We focus on convex problems, so we assume that the Hessian matrix Q is positive semidefinite. Moreover, we assume that $X = \{x \in \mathbb{R}^n : \ Ax \ \geq \ b, \ Cx \ = \ d, \ l \leq \ x \leq u\}$ is non-empty and bounded.

Furthermore, among all possible problems of type (1), we are particularly interested in those in which the matrix Q is dense and with the following additional property: there exists an efficient method for minimizing a linear function over the feasible set X, i.e., there exists an efficient linear minimization oracle that for a given $y \in \mathbb{R}^n$ solves the problem:

$$\min_{x \in X} y^\top x.$$

The class of these problems is NP-Hard and we present an algorithm to solve them. It is based on the classic Branch and Bound (B&B) method, but integrates it with a *Simplicial Decomposition (SD)* type approach, to solve the continuous relaxation of the mixed binary problem. Indeed, in [1] it is showed that SD is specifically tailored to solve the continuous relaxation of problems with the aforementioned features. However, it is worth noting that the proposed algorithm can handle any convex problem of type (1) and can be easily modified in order to deal with mixed integer problems and also problems having a general convex objective function. Within this framework we obtain promising results compared with the state-of-the-art solver *Cplex* on some sets of instances.

2 Simplicial Decomposition (SD)

Simplicial Decomposition (SD) can be seen as an application of the Dantzig - Wolfe decomposition method [2]. The idea behind it is described in [3, 4]: in order to solve the original continuous problem, it is decomposed into simpler ones, which are called respectively pricing and master programs, and are solved alternatively and repeatedly. The pricing solves the original problem with a linear objective function and the master program, instead, is a problem with the original objective function, but with lower dimension and simplified constraints. More precisely, given a set of extreme points of the domain, the master solves the original problem in the convex hull of them and the pricing linearizes the objective function in the optimal point of the master, to get a new extreme point; this cycle is repeated until the optimum of the original problem is found. In this way, the problem is solved with an inner approximation of its domain.

3 SD integrated in a Branch and Bound

In our B&B algorithm, at each node we solve the continuous relaxation with SD; in particular, to solve the master problems we use the *ACDM* algorithm proposed in [1], which is the most efficient in this case. Embedding the SD algorithm in this structure is useful for several reasons. The first one is that this algorithm has a good performance in terms of computational time with respect to *Cplex*, so it can improve the performance in solving each node. Then, since the extreme points given by a SD algorithm are always feasible, if they are also binary every node of the B&B is feasible; moreover, we can store information at each node to simplify the computations of the following nodes, as described in the next paragraphs.

Dual bound The pricing problem of SD, at each cycle, gives a valid lower bound on the solution: indeed, it solves a linear underestimator of the original objective function. Alternatively, it can be seen as the dual bound given by the Dantzig-Wolfe decomposition method. Such a lower bound can be exploited for pruning the nodes and it actually gives remarkable improvements. Hence, if the node can be pruned, performing the complete column generation algorithm is generally not needed, and most of the times only very few iterations are sufficient.

Column exploitation Another useful enhancement of the SD algorithm is that many columns generated at each node can be reused in the children nodes, if they satisfy the constraints given by the branching. Indeed, when at a given node a fractional solution is found for the continuous relaxation, and branching to a variable - say i - is performed, all the columns with i^{th} component equal to 1 or to 0 can be stored for the left and right child nodes. In this way, we can have an initial set of extreme columns for every node of the B&B tree. In particular, if the columns are binary, all the columns generated at every node are reused in the children nodes. This makes us able to *warmstart* the SD algorithm at every node. Results which evidence the reduction of computational time are presented later.

Branching strategy, branching rules The branching strategy that has been used is the *depth first search*: at each branching, the left child is the next node to explore. The advantages of this choice are the following:

- the number of open nodes is kept small: indeed, at any step of the algorithm, at most $n + 1$ nodes are opened, and this is the least possible;
- combined with SD, it allows to find rapidly both upper and lower bounds;
- it allows to efficiently reuse columns previously generated.

We choose the following branching rule: at each branching, we fix to 1 the variable with the largest fractional part. This choice generally allows us to keep several columns feasible in the left child node. Moreover, since the solutions of our problems are often sparse, this choice also allows us to find rapidly a good upper bound.

4 Computational results

We give a description of the computational results obtained with the SD based algorithm we presented in the previous sections. In our tests we consider instances of combinatorial problems. In particular, we focus on quadratic shortest path (QSP) problem and quadratic minimum spanning tree (QMST) problem.

Regarding the QSP problem, we use two types of instance: Quadratic Grid Shortest Path (QGSP) problem, that is graphs represented by a squared grid, and Quadratic Random Shortest Path (QRSP) problem, that is randomly generated graphs. The benchmark consists of 12 QGSP instances (with 180 to 420 variables, and 100 to 225 constraints) and 66 QRSP instances (1000, 3000 or 5000 variables, 100 to 300 constraints).

We also provide some preliminary results for the QMST problem, obtained on graphs with a grid structure. We have a benchmark of 18 instances, with 40 to 84 edges and 25 to 49 nodes.

4.1 Numerical results

In Table 1, average results are shown for the quadratic shortest path problem, for the grid and the randomly generated graphs. We compare the results of *Cplex*, with the results of our algorithm. We show the difference between reusing the columns in the children nodes or not. We show, on average, the CPU time (in seconds) to solve the instances for *Cplex*, and for our algorithm, without storing columns in the nodes (*BBSD*) or with the column reuse (*BBSD+col*), respectively; the number of B&B nodes of Cplex and of our algorithm; and the number of columns generated by our algorithm, in the two settings.

We observe that all the algorithms solve every instance to optimality, within the imposed timelimit of three hours. These results show that the computational time of the proposed algorithm is always better than that of the state-of-the-art software *Cplex*. Our algorithm generates on average more nodes than *Cplex*: since the overall time is shorter, it means that SD solves every node faster. It can be noticed that the reuse of columns from a node to the children is effective: while the number of nodes is almost unchanged, reusing columns allows to

Type	Time (s)			# nodes			# columns	
	Cplex	SDBB	SDBB+col	Cplex	SDBB	SDBB+col	SDBB	SDBB+col
QGSP	811.0	735.6	472.7	155005	267773	267641	2482845	1645443
QRSP	75.0	41.7	13.2	47.7	829.4	829.5	9173.7	2372.2

TAB. 1: Quadratic Shortest Path Problem.

save a considerable percentage of the CPU time, mainly due to the smaller number of iterations which are needed to be computed.

In Table 2 we present some preliminary results for the QMST problem. We show the number of vertices and edges, the computational time without storing the columns and with their reuse, and the average number of columns which are generated in the whole B&B tree.

Vertices	Edges	Time (s)		# nodes		# columns	
		SDBB	SDBB+col	SDBB	SDBB+col	SDBB	SDBB+col
25	40	0.27	0.15	1557	1553	13080	6523
36	60	54.35	33.48	190663	189746	1871141	1031868
49	84	5128.80	3276.50	9330521	9276077	124934508	69266606

TAB. 2: Quadratic Minimum Spanning Tree Problem.

We can see that, also in this case, the column exploitation is useful, since it allows to reduce the number of SD iterations and hence the overall computational time.

5 Conclusions

We presented an efficient combination of the SD framework with a branch and bound scheme, to solve mixed binary convex quadratic problems. It embeds in its structure the ad-hoc method for solving the master problem. It exploits all the advantages of the SD algorithm, as the lower (or dual) bound and the warmstart given by the structure of the simplices in an efficient way. We showed, through a numerical experience, that our algorithm performs better than *Cplex* when dealing with instances of Quadratic Shortest Path problems with a dense Hessian matrix. We also provided results for the Quadratic Minimum Spanning Tree problem. In conclusion, we showed that the SD algorithm, originally designed for continuous problems, can be profitably embedded in a framework for mixed binary quadratic problems.

References

[1] Enrico Bettiol, Lucas Létocart, Francesco Rinaldi, and Emiliano Traversi. A conjugate direction based simplicial decomposition framework for solving a specific class of dense convex quadratic programs. *Computational Optimization and Applications*, pages 1–40, 2019.

[2] George B Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations research*, 8(1):101–111, 1960.

[3] Charles A Holloway. An extension of the frank and wolfe method of feasible directions. *Mathematical Programming*, 6(1):14–27, 1974.

[4] Balder Von Hohenbalken. Simplicial decomposition in nonlinear programming algorithms. *Mathematical Programming*, 13(1):49–68, 1977.