Dantzig Wolfe decomposition and objective function convexification for binary quadratic problems: the cardinality constrained quadratic knapsack case.

Alberto Ceselli[1] [*], Lucas Létocart[2], Emiliano Traversi[2]

March 25, 2017

1. Dipartimento di Informatica
Università degli Studi di Milano
Via Bramante, 65
26013 Crema, Italy.
alberto.ceselli@unimi.it

2. Université Paris 13, Sorbonne Paris Cité
LIPN, CNRS, (UMR 7030)
F-93430 Villetaneuse, France.
{*lucas.letocart, emiliano.traversi*}*@lipn.univ-paris13.fr*

Abstract

The purpose of this paper is to provide strong reformulations for binary quadratic problems. We propose a first methodological analysis on a family of reformulations combining Dantzig-Wolfe decomposition and Quadratic Convex Reformulation principles. As a representative case study, we apply them to a cardinality constrained quadratic knapsack problem, providing extensive experimental insights. We show that a few reformulations of our family yield to continuous relaxations that are strong in terms of dual bounds and computationally efficient to optimize. We report and analyze in depth a particular reformulation providing continuous relaxations whose solutions turn out to be integer optima in all our tests. We finally discuss on how to exploit these results in a generic binary quadratic programming context.

1 Introduction

Generic solvers for mathematical programming models have been steadly improving since long time, in terms of both computing capabilities and modeling flexibility. While up to few years ago the focus was on effectively solving Mixed Integer Linear

[*]Partially funded by Regione Lombardia - Fondazione Cariplo, grant n. 2015-0717, project "RED-NEAT", and partially undertook when A. Ceselli was visiting LIPN - Université Paris 13, Sorbonne Paris Cité.

Programs (MILP), the interest is currently shifting to more general problems, possibly coping with nonlinearities in either the objective function or the constraints. Binary Quadratic Problems (BQPs) are among the special classes of Mixed Integer Non-Linear Problems (MINLP) which are currently subject to wider investigation.

In its generic form a BQP reads as follows:

$$(BQP) \quad \max\{x^\intercal Qx + L^\intercal x | Ax \leq b, x \in \{0,1\}\} . \tag{1}$$

with $Q \in \mathcal{Q}^{n\times n}$, not restricted to be convex, and $L \in \mathcal{Q}^n$. Several generic solvers are already available to tackle BQPs. To mention just a few examples BiqCrunch [22], CPLEX [19], GloMIQO [27], Gurobi [17] and SCIP [33] support direct optimization of either BQPs or even MINLP. In the vast majority of the cases, these solvers rely on Branch-and-Bound techniques, embedding special relaxations of the models provided by the user.

Restricting to the BQP case, a first popular choice for such relaxations is to reformulate the original problem by linearizing the quadratic terms [1, 14, 15, 16, 26, 34], thereby obtaining a MILP. A second promising way to obtain dual bounds for BQPs is to rely on Semidefinite Programming (SDP) relaxations, like for example [32] and [22].

When the objective function is concave, even the simple continuous relaxation of a BQP is appealing, as it yields to convex optimization subproblems, for which effective algorithms are engineered in state-of-the-art solvers like CPLEX and Gurobi. A third alternative for generic BQPs is therefore to use a reformulation obtained via the so-called *convexification* of the quadratic objective function. This convexification can be obtained by applying Quadratic Convex Reformulation (QCR) methods like those introduced in [5] and extended in [7].

On the other hand, Dantzig-Wolfe Decomposition (DWD) is a well known technique used to obtain tight bounds for MILPs (see for example [10, 11, 35]). Its principle is to replace the feasibility region corresponding to a subset of the constraints of a model by the convex hull of its extreme points through an inner representation. DWD can be applied in principle also to non linear mathematical programming models, provided such a subset of constraints exists, whose corresponding feasibility region can be represented as a polyhedron. In fact, the extension of DWD to non linear problems has been analyzed in several theoretical papers in the past years (see for example [2, 23, 18]). However, whether or not its application to MINLP may yield to successful computational methods is still an open research question.

One of the main issues is the following: the application of DWD leads to a formulation with an exponential number of variables. It can be solved via iterative procedures like Column Generation (we refer the reader to [12] for an extensive review of such a method), but particular problem convexity conditions have to be fulfilled to ensure convergence.

In this paper we present a methodology to systematically combine DWD and convexification methods like [5] and [7] to effectively solve BQPs. We therefore show that applying DWD to BQPs is not only theoretically possible, but also computationally profitable.

Our approach is both methodological and computational: in order to get key insights and ease the presentation we consider as test-bed a particular BQP in which the objective function is generic, and the set of constraints is composed by a single inequality and a single equality. Remarkably, on that particular problem we have found a reformulation method producing models whose continuous relaxation has an optimal integer solution in all the experiments we have performed.

The paper is organized as follows: we first detail our test problem (Section 2). Then, we describe different reformulations combining objective function convexification and DWD (Section 3) and we propose a theoretical analysis (Section 4) and an experimental comparison (Section 5) of them. As sketched above, one of these reformulations yields relaxations whose optimal solution is integral on all the instances we have tested. Therefore we analyse more deeply its structural properties, deriving its key factors (Section 6), finally reporting how such an understanding may substantially help in solving generic BQPs (Section 7).

2 A benchmark Binary Quadratic Problem

As a case study we focus on the 0-1 exact k-item quadratic knapsack problem [24, 25] (kQKP). The kQKP is a BQP with a generic objective function, but only two constraints (one equality and one inequality). It is hence perfect to be used as test problem, allowing us to focus on the methodology rather than the technical issues. More precisely, the kQKP consists of maximizing a quadratic function subject to a capacity and a cardinality constraint, which are both linear. It can be formulated as follows:

$$
\begin{aligned}
(\mathcal{F}) \quad \max \quad & f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_i x_j && (2)\\
\text{s.t.} \quad & \sum_{j=1}^{n} a_j x_j \le b && (3)\\
& \sum_{j=1}^{n} x_j = k && (4)\\
& x_j \in \{0,1\} \qquad j = 1,\dots,n && (5)
\end{aligned}
$$

where n denotes the number of items, and all the data, k (number of items to be filled in the knapsack), a_j (weight of item j), c_{ij} (profit associated with the selection of items i and j) and b (capacity of the knapsack) are nonnegative integers. Without loss of generality, the matrix $C = (c_{ij})$ is assumed to be symmetric.
Moreover, we assume that $\max_{j=1,\dots,n} a_j \le b < \sum_{j=1}^{n} a_j$ in order to avoid either trivial solutions or trivial variable fixing via constraint (3). Let us denote by k_{max} the largest number of items which could be filled in the knapsack, that is the largest k such that the sum of the k smallest a_j values does not exceed b. Therefore, we can assume that $k \in \{2,\dots,k_{max}\}$, where k_{max} can be found in O(n) time [4, 13]. Outside this range, either the value of the problem is equal to $\max_{i=1,\dots,n} c_{ii}$ (for k=1), or the domain of $(\mathcal{F})$ is empty (for $k > k_{max}$).

3 On improving the formulation of a BQP

The formulation ($\mathcal{F}$) presented in the previous section has two main drawbacks: first, its objective function is not guaranteed to be concave, hence leading to a possibly non-convex optimization problem; second, its continuous relaxation can be potentially weak.

In order to overcome these drawbacks, in subsection 3.1 we present quadratic convex reformulations, allowing to remap $\mathcal{F}$ into a convex problem with a stronger continuous relaxation, and in subsection 3.2 we exploit DWD to strengthen the formulation of the feasible region, possibly improving the bounds even further. Their integration is discussed in subsection 3.3.

3.1 A family of reformulations of the objective function $f(x)$

The objective function $f(x)$ can be reformulated by exploiting the fact that on the feasible region we have

$$x_j^2 = x_j \forall j = 1 \dots n \tag{6}$$

and

$$\sum_{j=1}^{n} x_j = k. \tag{7}$$

This idea is exploited in the recent work developed by Billionnet, Elloumi and M-C. Plateau [5, 31] and Billionnet, Elloumi and Lambert [7]. In this work, they highlight the great interest of an adequate convex reformulation of $f(x)$. Namely, the performance of commercial solvers is greatly improved when a rewriting is performed according to their general method called Quadratic Convex Reformulation (QCR). The QCR method consists of reformulating a nonconvex 0-1 quadratic maximization problem into an equivalent 0-1 program with a concave quadratic function. Then, the reformulated problem can be efficiently solved by a classical branch-and-bound algorithm based on continuous relaxation. So, the objective of this convexification method is to construct a model whose continuous relaxation is as tight as possible.

Due to (6) and (7) the objective function $f(x)$ is equivalent to the following function $f_{u,v}(x)$, depending on two parameters u in $\mathbb{R}^n$ and v in $\mathbb{R}$:

$$f_{u,v}(x) = f(x) - \sum_{i=1}^{n} u_i \left(x_i^2 - x_i\right) - v \left(\sum_{j=1}^{n} x_j - k\right)^2 \tag{8}$$

Leading to the following reformulations of ($\mathcal{F}$):

$$
\begin{aligned}
(\mathcal{F}^{\text{conv}}) \quad \max \quad & f_{u,v}(x) && && (9)\\
\text{s.t.} \quad & \sum_{j=1}^{n} a_j x_j \leq b && && (10)\\
& \sum_{j=1}^{n} x_j = k && && (11)\\
& x_j \in \{0,1\} && j = 1, \dots, n && (12)
\end{aligned}
$$

QCR determines the parameters $v^* \in \mathbb{R}$ and $u^* \in \mathbb{R}^n$ among those

- allowing to obtain a new function $f_{u^*,v^*}(x)$ which is concave (and hence leading to a problem with a convex continuous relaxation) and
- such that the upper bound obtained by the continuous relaxation is minimal.

That is, v^* and u^* realize the optimum of the following problem:

$$(P_{QCR}) \quad \min_{u\in\mathbb{R}^n, v\in\mathbb{R}} \quad \max_{x\in[0,1]^n \,:\, \sum_{j=1}^n a_j x_j \leq b, \quad \sum_{j=1}^n x_j = k} \quad f_{u,v}(x) \tag{13}$$

To obtain these optimal parameters, Billionnet et al. [5, 31] prove that a Semi Definite Programming (SDP) relaxation of $(\mathcal{F})$ has to be solved. For the sake of completeness, we report such a SDP model in Appendix A.

We remark that $(\mathcal{F}^{\text{conv}})$ presents the big advantage of being convex. This means that its continuous relaxation is usually easier to handle in comparison of the continuous relaxation of $(\mathcal{F})$. However, any possible set of coefficients u and v provides a valid reformulation of the objective function.

In [20] and [7], the authors build a convex 0-1 quadratic program reformulation stronger than $(\mathcal{F}^{\text{conv}})$. They first introduce additional y_{ij} variables representing the product of x_i and x_j variables, leading to the following formulation:

$$
\begin{aligned}
(\mathcal{F}^{\text{impr. conv}}) \quad & \max \quad f_{u,v,P,N}(x,y) && && (14)\\
& \text{s.t.} \quad \sum_{j=1}^n a_j x_j \leq b && && (15)\\
& \sum_{j=1}^n x_j = k && && (16)\\
& y_{ij} \leq x_i, \;\; y_{ij} \leq x_j && i,j = 1,\dots,n && (17)\\
& y_{ij} \geq 0, \;\; y_{ij} \geq x_i + x_j - 1 && i,j = 1,\dots,n && (18)\\
& x_j \in \{0,1\} && j = 1,\dots,n && (19)
\end{aligned}
$$

with

$$f_{u,v,P,N}(x,y) = x^T(C - Diag(u) - P - N)x + u^T x + \sum_{i,j=1}^n (P_{ij} + N_{ij}) y_{ij} - v\left(\sum_{j=1}^n x_j - k\right)^2$$

where the optimal u^*, v^*, P^*, N^* (i.e. the ones providing the strongest continuous relaxation) can be found by solving another associated SDP problem. The full construction of the improved convex 0-1 quadratic program reformulation (MQCR) of the model is provided in Appendix B.

Summarizing, by applying one of the two convexification methods presented above, we always obtain a convex quadratic (binary) optimization problem, whose objective function is of the form:

$$\max \quad \sum_{i=1}^n \sum_{j=1}^n \tilde{q}_{ij} x_i x_j + \sum_{j=1}^n \tilde{l}_j x_j + \sum_{i=1}^n \sum_{j=1}^n \tilde{w}_{ij} y_{ij}.$$

3.2 Dantzig-Wolfe Decomposition of $(\mathcal{F})$

A popular technique to obtain tighter bounds for Integer Linear Programs is DWD. Although DWD was initially devised to solve Linear Programs, its main ingredients can be applied whenever the feasibility region identified by a subset of the constraints of the problem can be represented as a polyhedron. In fact, the main idea behind DWD is to replace such a feasibility region with the convex hull of its extreme points, exploiting the Minkovsky and Weyl theorem (see for example [21]). From an algebraic point of view, this amounts to force the vector of decision variables to be represented as a linear convex combination of a finite set of (extreme) points. For this reason, in the literature this procedure is commonly known as partial “convexification”; however, in order to avoid misunderstanding with the QCR methods that deal with the convexification of the objective function, we refer to the effect of DWD as “strengthening” of subsets of constraints.

In the following we present all steps of DWD for the kQKP . In fact, a key point of our techniques is to exploit partial reformulations arising at an intermediate stage of the DWD process.

We first observe that $(\mathcal{F}^{\text{conv}})$ can be rewritten as follows:

$$
\begin{aligned}
(\mathcal{F}^{\text{conv}}_{\text{DWD}(\Omega)})\ \max\quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{q}_{ij}x_ix_j + \sum_{j=1}^{n}\tilde{l}_jx_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{w}_{ij}y_{ij} && && (20)\\
\text{s.t.}\quad & \sum_{j=1}^{n}a_jx_j \le b && && (21)\\
& \sum_{j=1}^{n}x_j = k && && (22)\\
& x_j \in \{0,1\} && j=1,\ldots,n && (23)\\
& y_{ij} = x_ix_j && i,j=1,\ldots,n && (24)\\
& x_j = \sum_{p\in\mathcal{P}} x_j^p\lambda^p && j=1,\ldots,n && (25)\\
& y_{ij} = \sum_{p\in\mathcal{P}} y_{ij}^p\lambda^p && i,j=1,\ldots,n && (26)\\
& \sum_{p\in\mathcal{P}}\lambda^p = 1 && && (27)\\
& \lambda^p \ge 0 && p\in\mathcal{P}
\end{aligned}
$$

where constraints (25), (26) and (27) impose x and y to belong to a given polyhedron Ω, whose set of extreme points is denoted by $\mathcal{P}$. In our case, the following choices of Ω are possible:

$$
\begin{aligned}
\Omega_{\text{knap}} &= \text{conv. hull}\{(x,y) : \sum_{j=1}^{n}a_jx_j \le b, y_{ij} = x_ix_j, i,j=1,\ldots,n, x_j\in\{0,1\}, j=1,\ldots,n\}\\
\Omega_{\text{card}} &= \text{conv. hull}\{(x,y) : \sum_{j=1}^{n}x_j = k, y_{ij} = x_ix_j, i,j=1,\ldots,n, x_j\in\{0,1\}, j=1,\ldots,n\}\\
\Omega_{\text{knap, card}} &= \text{conv. hull}\{(x,y) : \sum_{j=1}^{n}a_jx_j \le b, \sum_{j=1}^{n}x_j = k, y_{ij} = x_ix_j, i,j=1,\ldots,n, x_j\in\{0,1\}, j=1,\ldots,n\}
\end{aligned}
$$

with $\mathcal{P}_{\text{knap}}$, $\mathcal{P}_{\text{card}}$, $\mathcal{P}_{\text{knap, card}}$ being the corresponding sets of extreme points. According to the choice of Ω, at least one of the two constraints (21) and (22) becomes redundant and therefore can be dropped. For instance, if $\Omega = \Omega_{\text{knap}}$ then constraint (21) can be eliminated. Suitable choices of Ω can lead to formulations that are always as tight as $\mathcal{F}^{\text{conv}}$, and potentially much tighter. We remark that after this first reformulation step, $(\mathcal{F}^{\text{conv}}_{\text{DWD}(\Omega)})$ has a quadratic objective function in the x_i variables only: λ^p variables do not appear in it.

Then, we can rewrite the objective function of $(\mathcal{F}^{\text{conv}}_{\text{DWD}(\Omega)})$ by performing variable substitution according to constraints (24), (25) and (26):

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{q}_{ij}\sum_{p\in\mathcal{P}}y^p_{ij}\lambda^p + \sum_{j=1}^{n}\tilde{l}_j\sum_{p\in\mathcal{P}}x^p_j\lambda^p + \sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{w}_{ij}\sum_{p\in\mathcal{P}}y^p_{ij}\lambda^p$$

that by further exploiting constraints $y_{ij} = x_i x_j$ in the definition of Ω on the first term, and after trivial simplifications, can be written as

$$\sum_{p\in\mathcal{P}}(\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{q}_{ij}x^p_i x^p_j)\lambda^p + \sum_{p\in\mathcal{P}}(\sum_{j=1}^{n}\tilde{l}_j x^p_j)\lambda^p + \sum_{p\in\mathcal{P}}(\sum_{i=1}^{n}\sum_{j=1}^{n}\tilde{w}_{ij}y^p_{ij})\lambda^p$$

We remark that after this second reformulation step, the objective function is linear.

Variable substitution can be performed also on constraints (21) and (22), obtaining

$$\sum_{p\in\mathcal{P}}(\sum_{j=1}^{n}a_j x^p_j)\lambda^p \le b$$

and

$$\sum_{p\in\mathcal{P}}(\sum_{j=1}^{n}x^p_j)\lambda^p = k.$$

We denote as $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(\Omega)}})$ the model obtained in this way.

Computationally speaking, further steps may be pertinent in special cases. First, when $\tilde{w}_{ij} = 0$ (as for the function $f_{u,v}(x)$ produced by the pure QCR method), there is no need to keep the y_{ij} variables in the model, and therefore also constraints (24) and (26) can be dropped. Second, in order to work with linear constraints only, constraints (24) can be replaced by:

$$y_{ij} \le x_i, y_{ij} \le x_j, y_{ij} \ge x_i + x_j - 1, y_{ij} \ge 0$$

for each $i, j = 1, \ldots, n$.

3.3 Formulations Overview

While from an integer optimization point of view all the formulations presented in the previous subsections are equivalent, their continuous relaxation bounds can be substantially different.

In turn, since the set of λ_p variables is exponential, it is not viable to straightly solve the corresponding continuous relaxations. Instead, column generation techniques need to be employed [11]. They consist in an iterative process: the set $\mathcal{P}$ is replaced by a small subset $\mathcal{S} \subset \mathcal{P}$. The continuous relaxation of the resulting *master problem* is

			Convexification		
			orig	conv	impr conv
Strenthening	quad master	knap	`NCM`	$\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)}$	$\mathcal{F}^{\text{impr. conv}}_{\text{DWD}(knap)}$
		card	`NCM`	`IPP`	`IPP`
		knap+card	`NCM`	$\mathcal{F}^{\text{conv}}_{\text{DWD}(card,knap)}$	$\mathcal{F}^{\text{impr. conv}}_{\text{DWD}(card,knap)}$
	lin master	knap	$\mathcal{F}_{\overline{\text{DWD}}(knap)}$	$\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$	$\mathcal{F}^{\text{impr. conv}}_{\overline{\text{DWD}}(knap)}$
		card	$\mathcal{F}_{\overline{\text{DWD}}(card)}$	$\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(card)}$	$\mathcal{F}^{\text{impr. conv}}_{\overline{\text{DWD}}(card)}$
		knap+card	`POP`	`POP`	`POP`

Table 1: Summary of possible formulations combining strengthening and convexification.

solved. In both linear and convex quadratic optimization problems solution algorithms exist exploiting strong duality, that is providing both a globally optimal primal solution and a corresponding dual solution. Such a dual solution is therefore used to search for elements of $\mathcal{P} \setminus \mathcal{S}$ corresponding to positive pseudo-cost columns. The problem of finding a column of maximum pseudo-cost is called the *pricing problem*. If any is found, the set $\mathcal{S}$ is enlarged, and the process is iterated by solving the master problem again. If no positive pseudo-cost column is found, instead, column generation stops, and the optimal solution of the master problem is guaranteed to be optimal also for the full problem, that is involving the whole set $\mathcal{P}$.

In Table 1 we present an overview of the possible options arising from the combination of convexification and strengthening. The Table is organized as follows: DWD options are reported in rows. In particular, *quad master* refers to the formulations obtained by stopping DWD after the first step, that is keeping the quadratic terms involving the x_i variables in the master problem, while *linear master* refers to the formulations where full variable substitution is performed, leading to a linear objective function in the master problem. Instead, *knap*, *card* and *knap+card* refer to the possible choice of constraints to be strengthned, that is the choice of Ω.

Convexification options are reported in columns: column *orig* refers to the use of the original (possibily non convex) objective function, while column *conv* (resp. *impr conv*) refers to the option of reformulating the objective function with a method like QCR (resp. MQCR).

We first note that not every combination is relevant:

- `NCM`
  the combination marked with `NCM` (Non Convex Master) are those leading to a possibly non convex master problem. On these cases it is not possible, in general, to ensure through KKT conditions that the solution achieved during master optimization steps is a global optimum, and therefore it is not possible to guarantee global convergence of the column generation process. For this reason we discarded these combinations from our analysis.

- `IPP`
  The combination marked with `IPP` (Integrality Property on Pricing) are those where the pricing proplem has the so-called *integrality property*, i.e. the continuous relaxation of the pricing problem has always an optimal integer solution.

In fact, the pricing problem is linear, and amounts in selecting a set of k items of maximum prize. That is, even if we apply DWD, no improvement in the continuous relaxation bound is obtained. We therefore decided to discard these configurations.

- `POP`
The combinations marked with `POP` (Pricing is Original Problem) are those where the pricing problem presents a quadratic objective function, and the same feasible region as the original problem. Therefore, a single column generation iteration would be as hard as the original problem. Also these configurations were discarded from further analysis.

Among the remaining combinations, we show in details the master and pricing problems of $\mathcal{F}^{\text{impr. conv}}_{\text{DWD}(card,knap)}$ and $\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$, being representative of the whole set of options.

Indeed, it is worth noting that $\mathcal{F}^{\text{impr. conv}}_{\text{DWD}(card,knap)}$ provides the strongest reformulation of the feasible region; the pricing problem is kept of manageable complexity by leaving the quadratic part of the objective function in the master.

$$\begin{aligned}
\mathcal{F}^{\text{impr. conv}}_{\text{DWD}(card,knap)} \max \quad & f_{u,v,P,N}(x,y) & \\
\text{s.t.} \quad & x_j = \sum_{p\in\mathcal{P}_{card,knap}} x_j^p \lambda^p j = 1,\ldots,n & [\delta_j] \\
& y_{ij} = \sum_{p\in\mathcal{P}_{card,knap}} y_{ij}^p \lambda^p i,j = 1,\ldots,n & [\mu_{ij}] \\
& \sum_{p\in\mathcal{P}_{card,knap}} \lambda^p = 1 & [\sigma] \\
& 0 \le x_j \le 1 j = 1,\ldots,n & \\
& 0 \le y_{ij} \le 1 i,j = 1,\ldots,n & \\
& \lambda^p \ge 0 p \in \mathcal{P}_{card,knap} &
\end{aligned}$$

with $\mathcal{P}_{card,knap}$ being the set of extreme points of $conv\{(x,y)|0 \le x \le 1; \sum_{j=1}^n x_j = k; \sum_{j=1}^n a_j x_j \le b; y_{ij} \le x_i,\ y_{ij} \le x_j, y_{ij} \ge 0,\ y_{ij} \ge x_i + x_j - 1, i,\ j = 1,\ldots,n, x \in \{0,1\}\}$ and δ, μ, σ being the dual variables associated to the constraints in the master.

Let δ^*, μ^*, σ^* be the optimal dual variables of a master solution during a generic column generation iteration; the pricing problem can be written as follows:

$$\begin{aligned}
\max \quad & \sum_{j=1}^n \delta_j^* x_j + \sum_{i,j=1}^n \mu_{ij}^* y_{ij} + \sigma^* & \\
\text{s.t.} \quad & \sum_{j=1}^n a_j x_j \le b & \\
& \sum_{j=1}^n x_j = k & \\
& y_{ij} \le x_i,\ y_{ij} \le x_j & i,j = 1,\ldots,n \\
& y_{ij} \ge 0,\ y_{ij} \ge x_i + x_j - 1 & i,j = 1,\ldots,n \\
& x_j \in \{0,1\} & j = 1,\ldots,n.
\end{aligned}$$

Also $\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$ is appealing, since the corresponding master is linear and contains only two constraints, and the pricing is a quadratic binary knapsack problem, for which effective ad-hoc combinatorial algorithms exist [9, 6, 29, 28].

$$
\begin{array}{lllr}
\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}\ \max & & \displaystyle\sum_{p\in\mathcal{P}_{\text{knap}}} c^p_{u^*,v^*}\lambda^p & \\
 & \text{s.t.} & \displaystyle\sum_{p\in\mathcal{P}_{\text{knap}}}\sum_{j=1}^{n} x^p_j\lambda^p = k & [\gamma] \\
 & & \displaystyle\sum_{p\in\mathcal{P}_{\text{knap}}} \lambda^p = 1 & [\theta] \\
 & & \lambda^p \geq 0 p \in \mathcal{P}_{\text{knap}} &
\end{array}
$$

with $\mathcal{P}_{\text{knap}}$ being the set of extreme points of $conv\{x| \sum_{j=1}^{n} a_j x_j \leq b, x \in \{0,1\}\}$, and γ and θ being the dual variables associated to the constraints in the master.

Denoting as γ^* and θ^* the optimal dual variables of a restricted master solution during a column generation iteration, the pricing problem can be written as follows:

$$
\begin{array}{lll}
\max & f_{u^*,v^*}(x) + \displaystyle\sum_{i=1}^{n} \phi^*_i x_i + \gamma^* + \theta^* & \\
\text{s.t.} & \displaystyle\sum_{j=1}^{n} a_j x_j \leq b & \\
 & x_j \in \{0,1\} & j = 1,\ldots,n
\end{array}
$$

4 Theoretical comparison

An important way to compare the proposed reformulations is to evaluate the strength of their continuous relaxations, i.e. when constraints $x \in \{0,1\}$ are substituted by $x \in [0,1]$. In the previous sections we discussed several alternatives for the application of convexification and strengthening. The following propositions build a hierarchy of decompositions in terms of their bounds; the corresponding computational issues are discussed in Section 5.

In the following, we use $\mathcal{F}_{\text{DWD}}$, $\mathcal{F}^{\text{conv}}_{\text{DWD}}$ and $\mathcal{F}^{\text{impr conv}}_{\text{DWD}}$ to identify all the decompositions using either the original objective function, its reformulations with a method like QCR or those with a method like MQCR.

Let $\mathcal{F}^{(\cdot)}_{\text{DWD}(\cdot)}$ and $\mathcal{F}^{(\cdot)}_{\overline{\text{DWD}}(\cdot)}$ identify all pairs of decompositions sharing the same objective function convexification and feasiblity region strengthening, thereby differing only on the stage in which DWD is stopped (that is, either before of after the variable substitution step).

Proposition 1. *For a generic $f(x)$, $\mathcal{F}^{(.)}_{DWD(knap,\ card)}$ is stronger than both $\mathcal{F}^{(.)}_{DWD(knap)}$ and $\mathcal{F}^{(.)}_{DWD(card)}$,*

directly following from the Dantzig-Wolfe decomposition principle.

Proposition 2. *For a generic $f(x)$, $\mathcal{F}^{impr.conv}_{DWD(.)}$ is stronger than $\mathcal{F}^{conv}_{DWD(.)}$.*

Such a result holds for every BQP [7].

Proposition 3. *For a generic $f(x)$, and a particular setting of u and v, $\mathcal{F}^{impr.conv}_{\overline{DWD}(.)}$ and $\mathcal{F}^{conv}_{\overline{DWD}(.)}$ are equivalent*

Proof. We observe that pricing problems differ only in their objective function. In particular, the terms involving the master problem dual variables are equal; the remaining terms in the objective function of $\mathcal{F}^{impr.conv}_{\overline{\mathrm{DWD}}(.)}$ are the following:

$$x^T(C - Diag(u) - P - N)x + u^T x + \sum_{i,j=1}^{n}(P_{ij} + N_{ij})y_{ij} - v\left(\sum_{j=1}^{n} x_j - k\right)^2$$

but since the pricing problem is solved to integer optimality, y_{ij} is always equal to $x_i \cdot x_j$, and therefore

$$x^T(-P - N)x = \sum_{i,j=1,\ldots,n}(-P_{ij} - N_{ij})x_i x_j = \sum_{i,j=1,\ldots,n}(-P_{ij} - N_{ij})y_{ij},$$

reducing $f_{u,v,P,N}(x, y)$ to

$$x^T(C - Diag(u))x + u^T x - v\left(\sum_{j=1}^{n} x_j - k\right)^2$$

which is equal to the remaining terms in the objective function of $\mathcal{F}^{conv}_{\overline{\mathrm{DWD}}(.)}$. □

Proposition 3 allows us to exclude the family of formulations $\mathcal{F}^{\text{imp conv}}_{\overline{DWD}(\cdot)}$ from our computational analysis, being equivalent to $\mathcal{F}^{\text{conv}}_{\overline{DWD}(\cdot)}$, but requiring to solve a more complex pricing problem.

Proposition 4. *For a generic $f(x)$, if $v > 0$ (resp. $v < 0$) $\mathcal{F}^{conv}_{\overline{DWD}(.)}$ is stronger (resp. weaker) than $\mathcal{F}_{\overline{DWD}(.)}$.*

Proof. We again observe that pricing problems differ only in their objective function, and the terms involving the master problem dual variables are equal. The remaining terms in the objective function of $\mathcal{F}^{conv}_{\overline{\mathrm{DWD}}(.)}$ are the following:

$$f(x) - \sum_{i=1}^{n} u_i\left(x_i^2 - x_i\right) - v\left(\sum_{j=1}^{n} x_j - k\right)^2$$

but since the pricing problem is solved to integer optimality, $x_i = x_i^2$, and therefore the objective function reduces to

$$f(x) - v\left(\sum_{j=1}^{n} x_j - k\right)^2.$$

If $v > 0$ (resp. $v < 0$), the cost of each column in $\mathcal{F}^{conv}_{\overline{\mathrm{DWD}}(.)}$ is always lower (resp. greater) than the cost of a corresponding column in $\mathcal{F}_{\overline{\mathrm{DWD}}(.)}$. Since the feasibility regions are the same, it is always possible to consider an optimal solution for $\mathcal{F}^{conv}_{\overline{\mathrm{DWD}}(.)}$ (resp. $\mathcal{F}_{\overline{\mathrm{DWD}}(.)}$): this is always feasible for $\mathcal{F}_{\overline{\mathrm{DWD}}(.)}$ (resp. $\mathcal{F}^{conv}_{\overline{\mathrm{DWD}}(.)}$), of lower cost, and possibly sub-optimal, thereby proving our claim. □

$$\begin{tikzcd}
 & \mathcal{F}^{\text{conv}}_{DWD(\cdot)} & \mathcal{F}^{\text{imp conv}}_{DWD(\cdot)} \arrow[l, Leftarrow, "\text{Prop.2}"] \\
\mathcal{F}_{\overline{DWD}(\cdot)} \arrow[ur, Leftrightarrow, "/" marking, "\text{Prop.6}"] & \mathcal{F}^{\text{conv}}_{\overline{DWD}(\cdot)} \arrow[l, Rightarrow, "\text{Prop.4,}v>0"'] \arrow[u, Rightarrow, "\text{Prop.5}"] \arrow[r, Leftrightarrow, "\text{Prop.3}"] & \mathcal{F}^{\text{imp conv}}_{\overline{DWD}(\cdot)} \arrow[u, Rightarrow, "\text{Prop.5}"]
\end{tikzcd}$$

Figure 1: Hierarchy of reformulations.

Practically speaking, we found no instance with $v^* < 0$ in our experiments. This last observation opens key insights into the structure of our methods, and is therefore discussed in full details in Section 6.

Proposition 5. *If $f(x)$ is concave (and hence the problem is convex), $\mathcal{F}^{(\cdot)}_{\overline{DWD}(\cdot)}$ is stronger than $\mathcal{F}^{(\cdot)}_{DWD(\cdot)}$.*

Proof. Let $(\tilde{x}, \tilde{\lambda})$ be a feasible point for $\mathcal{F}^{(\cdot)}_{\text{DWD}(\cdot)}$, it will always be feasible for $\mathcal{F}^{(\cdot)}_{\overline{\text{DWD}}(\cdot)}$ and viceversa, this is true because both models have the same feasible region. In $\mathcal{F}^{(\cdot)}_{\overline{\text{DWD}}(\cdot)}$, the objective function value of $(\tilde{x}, \tilde{\lambda})$ is the following: $\sum_{p\in\mathcal{P}} c^p \tilde{\lambda}^p$; hence

$$\sum_{p\in\mathcal{P}} c^p \tilde{\lambda}^p = \sum_{p\in\mathcal{P}} f(x^p)\tilde{\lambda}^p \le f(\sum_{p\in\mathcal{P}} x^p \tilde{\lambda}^p)$$

due to the concavity of $f(x)$ and the constraint imposing $\sum_{p\in\mathcal{P}} \lambda^p = 1$. At the same time, $f(\sum_{p\in\mathcal{P}} x^p \tilde{\lambda}^p) = f(\tilde{x})$, that is the objective function value of $(\tilde{x}, \tilde{\lambda})$ in $\mathcal{F}^{(\cdot)}_{\text{DWD}(\cdot)}$. □

Proposition 5 suggests that with a purely concave objective function the reformulations containing a quadratic pricing problem are preferable in terms of bounds. A similar reasoning is valid for the case of a purely convex objective function. In this case a reformulation with a quadratic master problem would always produce non weaker bounds. This consideration is however of theoretical interest only, as we do not have global optimality guarantees for nonconvex master problems.

Finally, while Proposition 5 implies that $\mathcal{F}^{conv}_{\overline{\text{DWD}}(\cdot)}$ dominates $\mathcal{F}^{conv}_{\text{DWD}(\cdot)}$, against the intuition we experimentally observed the following:

Proposition 6. *There is no dominance between $\mathcal{F}_{\overline{DWD}(\cdot)}$ and $\mathcal{F}^{conv}_{DWD(\cdot)}$.*

We experimentally found an even stronger result: no dominance was observed even between $\mathcal{F}_{\overline{\text{DWD}}(\cdot)}$ and $\mathcal{F}^{conv}$ (that is the pure application of QCR to the original formulation), proving interesting degrees of orthogonality to exist between convexification and strengthening.

The overall hierarchy of reformulations is provided in Figure 1: for each pair of formulation families A and B, we depict $A \Rightarrow B$ when A is stronger than B, $A \Leftrightarrow B$ when A and B are equivalent, and $A \not\Leftrightarrow B$ when no a priori dominance can be proved between A and B. Near each dominance arrow we also reference the proposition yielding to the statement.

5 Computational comparison

Building on the results of Sections 3.3 and 4, we designed an experimental campaign in order to investigate the computational properties of our reformulations.

Our main aim is to assess the tradeoff that can be obtained in terms of dual bound quality versus computing time.

We therefore implemented a `C++` framework, in which the master problem is solved by the continuous solver of IBM `CPLEX` [19], and the pricing problem, unless otherwise indicated, is solved by using the integer (either quadratic or linear) solver of IBM `CPLEX`. In both cases we kept default settings.

In this section, the following set of instances from the literature is used as testbed:

- Density-dependent (`DD`) instances.
  This set consists of randomly generated kQKP instances introduced in [24]. The number of items takes the following values: $n = 50, 60, \dots, 100$; b, a_j and c_{ij} are positive integers. The values of $k \in \{1, \dots, \lfloor \frac{n}{4} \rfloor\}$ are chosen in a way to make the instances harder than the corresponding Quadratic Knapsack instance. The density of the objective function δ (defined as the percentage ratio between the number of non zero coefficients c_{ij} and n^2) takes the following values: $\delta = 25, 50, 75$. For any combination of n and δ, 10 random instances are included in the data set. This test-bed consists in total of 180 instances.

5.1 DWD with a quadratic master

In Table 2 the results concerning the families of formulations $\mathcal{F}^{(\cdot)}_{\text{DWD}(\cdot)}$, that are those with a quadratic master and a linear pricing problem, are presented.

As explained in Section 3.2, to obtain a reformulation amenable to be solved by DWD, the objective function in the master needs to be concave, for this reason we modified the objective function using u^*, v^*, P^*, N^*.

The table is divided into four vertical blocks. The first and second block, which are $(\mathcal{F}^{\text{conv}})$ and $(\mathcal{F}^{\text{impr conv}})$, report the results concerning the model obtained after convexifying the objective function with the techniques introduced in Section 3.1, and performing no DWD strengthening. The third and forth block, which are $(\mathcal{F}^{\text{conv}}_{\text{DWD(knap)}})$ and $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$, report the results concerning the models obtained after applying also DWD as explained in Section 3.2.

In particular, $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$ is the best possible DWD strengthening of the feasible region (not yielding back to the original problem). In the $(\mathcal{F}^{\text{conv}}_{\text{DWD(knap)}})$ case, instead, the pricing problem is a binary (linear) knaspack problem, for which effective ad-hoc algorithms [30] can be potentially used in place of the `CPLEX` solver.

The results on the two remaining options $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap)}})$ and $(\mathcal{F}^{\text{conv}}_{\text{DWD(knap, card)}})$ provided no further insights, and were therefore dropped from the Table.

The table shows horizontally the average of tests concerning the ten instances with the same value of n and δ. For each block we report the computing time (excluding the pre-processing time needed to compute optimal multipliers by solving SDPs) and the duality gap (defined as the difference between the continous relaxation bound and the optimal value, divided by the optimal value). In the last four lines the average over all the instances with the same density and the total average are reported.

		$(\mathcal{F}^{\text{conv}})$		$(\mathcal{F}^{\text{impr conv}})$		$(\mathcal{F}^{\text{conv}}_{\text{DWD(knap)}})$		$(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$	
		Gap	Time	Gap	Time	Gap	Time	Gap	Time
n	$\delta(\%)$								
50	*25*	102.65	0.02	30.89	1.05	38.36	1.97	29.15	9.30
	50	150.56	0.06	25.25	0.94	31.05	3.04	23.66	9.71
	75	230.29	0.12	105.16	1.09	114.26	1.55	100.88	8.06
60	*25*	60.76	0.04	130.92	0.04	149.25	1.55	126.07	10.89
	50	93.73	0.11	15.08	2.61	19.48	3.86	14.19	19.05
	75	212.67	0.25	141.08	2.09	151.22	1.67	136.22	8.99
70	*25*	130.23	0.06	38.03	5.11	46.84	4.82	36.52	33.25
	50	177.07	0.19	72.81	4.27	80.44	6.83	70.77	54.37
	75	382.36	0.44	56.26	3.45	63.77	3.25	54.57	22.19
80	*25*	111.24	0.08	34.05	7.98	41.90	5.64	32.87	71.19
	50	271.64	0.26	55.44	9.59	64.09	4.67	53.65	43.98
	75	313.33	0.66	83.58	7.42	92.31	4.64	81.47	43.42
90	*25*	118.45	0.13	112.80	13.75	129.31	4.74	109.63	44.66
	50	248.57	0.48	83.15	12.38	92.19	4.52	81.65	66.75
	75	388.68	1.06	37.90	5.63	42.13	6.95	37.12	102.54
100	*25*	169.43	0.16	73.90	23.49	82.78	6.80	72.72	99.90
	50	145.72	0.49	17.38	28.06	21.83	8.58	17.19	219.77
	75	260.26	1.25	21.67	18.37	27.22	6.23	21.50	158.30
Avg	*25*	115.46	0.08	70.10	8.57	81.41	4.25	67.83	44.86
	50	181.22	0.26	44.85	9.64	51.51	5.25	43.52	68.94
	75	297.93	0.63	74.28	6.34	81.82	4.05	71.96	57.25
Avg		198.20	0.32	63.08	8.18	71.58	4.52	61.10	57.02

Table 2: `DD` instances, formulations $(\mathcal{F}^{(\cdot)})$ and $(\mathcal{F}^{(\cdot)}_{\text{DWD}(\cdot)})$, continuous relaxation bounds.

By comparing the first two blocks, it is clear that convexifying the objective function with a method like MQCR improves significantly the strength of the dual bound; on the other hand, the computing time grows by one order of magnitude.

As theoretically expected, the duality gaps reported in the third block are smaller than those in the first block, at the expense of higher computing time.

Using MQCR in place of QCR, without DWD, or strengthening by DWD(knap) after QCR yields a similar effect both in terms of dual bound improvement and computing time increase, proving DWD to be an appealing alternative to complex convexification methods.

Even if, from a theoretical point of view, $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$ dominates $(\mathcal{F}^{\text{impr conv}})$, the experimental improvement in the duality gap that can be seen by comparing blocks two and four is limited, requiring at the same a substantial increase of computational effort.

Blocks three and four correspond to formulations in which both a more accurate convexification is performed and a larger feasibility region is strengthened. Nevertheless, the bound improvement is not as outstanding as expected, and the computing time grows by one order of magnitude. This suggests that while using DWD strengthening, employing sophisticated convexification mechanisms may not be worthwhile.

Among all our test, the most surprising result is arising from the block related to $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$. It is well known in combinatorial optimization that strenghtening a feasibility region usually helps significantly in improving the dual bound. For this reason, the modest improvement of the duality gap obtained after strenghtening the knapsack and the cardinality constraints (i.e. optimizing over the convex hull of the whole feasible region) is, at first glance, counter-intuitive.

A possible reason for this behaviour would be that the optimal solution is attained in the interior of the convex hull of the feasible region. Due to the fact that the master function is not linear (but still concave), the optimum of a constrained BQP can coincide with the optimum of its unconstrained counterpart. More precisely, it is sufficient to have one optimal solution in the interior of the convex hull of the feasible region to obtain no improvement in the bound after strenghtening.

In an effort for experimentally checking if the explanation holds, in Table 3 we report the number of `DD` instances with an optimal solution in the interior of the convex hull of the feasible region (over a total of ten instances).

n	50			60			70			80			90			100		
%	25	50	75	25	50	75	25	50	75	25	50	75	25	50	75	25	50	75
	0	2	0	0	2	0	0	2	0	0	1	0	0	1	0	0	1	0

Table 3: $(\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}})$, Number of instances with optimum of the continuous relaxation in the interior of the convex hull of the feasible region

The results clearly show that the modest improvement in the bound does not depend on having an optimal solution in the interior of the convex hull.

An a-posteriori explanation for this phenomenon is that strenghtening has no effect only if the system of linear equations

$$2x^{\mathsf{T}}Q + L = 0$$

allows a solution in the convex hull of the feasible region, that is unlikely in a generic BQP instance.

An alternative explanation of the behaviour showed in Table 2 is related to the fact that the convexification reformulates the original objective functions in an "artificially" convex objective function, which is known to be "flatter" than the original one (see for example [3]).

Intuitively, an objective function is flat if it has gradual descent directions, along which the value of the objective function does not change significantly. This consideration finally explains the results of Table 2.

5.2 DWD with a quadratic pricing

In this section we add to the analysis the formulations with a quadratic pricing problem. Table 4 is organized in the same way as Table 2. In this case it is not mandatory to convexify the objective function; for this reason we compare both the DWD with the original objective function ($\mathcal{F}_{\overline{\text{DWD}}(knap)}$) and the one with a convexified objective function ($\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$), ($\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}}$) and ($\mathcal{F}^{\text{conv}}$).

		($\mathcal{F}^{\text{conv}}$)		($\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}}$)		$\mathcal{F}_{\overline{\text{DWD}}(knap)}$		$\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$	
n	δ(%)	Gap	Time	Gap	Time	Gap	Time	Gap	Time
50	*25*	102.65	0.02	29.15	9.30	19.43	0.30	0.00	40.13
	50	150.56	0.06	23.66	9.71	28.97	0.08	0.00	34.72
	75	230.29	0.12	100.88	8.06	119.44	0.05	0.00	14.65
60	*25*	60.76	0.04	126.07	10.89	79.77	0.14	0.00	27.93
	50	93.73	0.11	14.19	19.05	13.13	0.25	0.00	32.54
	75	212.67	0.25	136.22	8.99	157.01	0.07	0.00	16.89
70	*25*	130.23	0.06	36.52	33.25	16.80	1.20	0.00	64.44
	50	177.07	0.19	70.77	54.37	47.85	4.30	0.00	72.07
	75	382.36	0.44	54.57	22.19	65.16	0.15	0.00	55.08
80	*25*	111.24	0.08	32.87	71.19	11.67	210.59	0.00	78.04
	50	271.64	0.26	53.65	43.98	42.69	0.77	0.00	66.70
	75	313.33	0.66	81.47	43.42	95.92	14.00	0.00	85.52
90	*25*	118.45	0.13	109.63	44.66	57.50	1692.40	0.00	89.42
	50	248.57	0.48	81.65	66.75	63.79	190.64	0.00	102.25
	75	388.68	1.06	37.12	102.54	59.35	15.16	0.00	491.36
100	*25*	169.43	0.16	72.72	99.90	41.07	926.10	0.00	145.81
	50	145.72	0.49	17.19	219.77	14.75	577.86	0.00	289.12
	75	260.26	1.25	21.50	158.30	47.82	3.77	0.00	656.70
Avg	*25*	115.46	0.08	67.83	44.86	37.71	471.79	0.00	74.30
	50	181.22	0.26	43.52	68.94	35.20	128.98	0.00	99.57
	75	297.93	0.63	71.96	57.25	90.79	5.53	0.00	220.03
Avg		198.20	0.32	61.10	57.02	54.56	202.10	0.00	131.30

Table 4: `DD` instances, formulations ($\mathcal{F}^{(\cdot)}_{\overline{\text{DWD}}(knap)}$), continuous relaxation bounds.

The results show that it is not possible to identify a dominant reformulation between ($\mathcal{F}_{\overline{\text{DWD}}(knap)}$) and ($\mathcal{F}^{\text{impr conv}}_{\text{DWD(knap, card)}}$). On the other hand, the last column of the Table shows how ($\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}}(knap)}$) provides an impressive bound, closing 100% of the open gap of all the instances in a reasonable amount of time. This striking result is analysed in details in Section 6.

The design of an exact approach based on the relaxation proposed goes beyond the scope of this work. However, the fact that with $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$ we are always able to close the whole integrality gap, suggests that a comparison with the overall solving time of CPLEX is pertinent to assess the actual computational potential of the decomposition proposed. Therefore, as last test of this section, in Table 5 we compare the time needed for computing the root node of $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$ (that in each test corresponds to the time needed to solve the problem to global integer optimality) with the overall time needed by CPLEX to solve the original formulation $(\mathcal{F})$ and $(\mathcal{F}^{\text{conv}})$.

		opt		root
		$(\mathcal{F})$	$(\mathcal{F}^{\text{conv}})$	$(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$
		Time	Time	Time
n	$\delta(\%)$			
50	*25*	132.64	1.16	40.13
	50	507.66	1.12	34.72
	75	384.37	0.89	14.65
60	*25*	12.85	1.95	27.93
	50	1089.85	2.37	32.54
	75	430.88	4.87	16.89
70	*25*	987.44	7.41	64.44
	50	2092.01	19.45	72.07
	75	1560.38	21.51	55.08
80	*25*	2324.91	29.38	78.04
	50	2184.51	48.57	66.70
	75	2032.01	88.06	85.52
90	*25*	1438.01	62.14	89.42
	50	2413.05	239.24	102.25
	75	2986.10	1024.02	491.36
100	*25*	2201.38	234.21	145.81
	50	3157.54	1271.16	289.12
	75	3246.99	1419.84	656.70
Avg		1621.25	248.74	131.30

Table 5: $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$ root node vs CPLEX

The results shows how with $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$ we are able to close the whole integrality gap faster than the time needed by CPLEX to solve to optimality the same instance, after convexifying the objective function. This results shows the extremely interesting potential of embedding $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$ in a Branch-and-Bound framework.

6 An alternative look to $(\mathcal{F}^{\mathbf{conv}}_{\mathbf{DWD}(knap)})$

In an effort for understanding its extreme behaviour, we analyze more deeply the $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$ case. Indeed, one may focus on the objective function of the pricing problem of $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$ (see Section 5.2), that reads as follows

$$f_{v,u}(x) = \sum_{i=1}^{n} u_i \left(x_i^2 - x_i\right) + x^\intercal Q x + L^\intercal x - v \left(\sum_{j=1}^{n} x_j - k\right)^2 + \tau^{*\intercal} x + \beta^* \ .$$

and straightly implies the following observation.

Proposition 7. *By setting $v = +\infty$, solving each pricing subproblem is equivalent to solving a kQKP with objective function $x^\intercal Q x + L^\intercal x + \tau^{*\intercal} x + \beta^*$*

Proof. The capacity constraint is explicitly enforced in the model, and the term $v\left(\sum_{j=1}^{n} x_j - k\right)^2$ in the pricing objective can be viewed as a penalty for violating the equality constraint. This implies that for a sufficiently large value of v, the pricing problem always produces solutions which are feasible also with respect to the cardinality constraint, and are therefore feasible for the original kQKP . Furthermore, since the pricing problem is solved to integer optimality, all terms $(x_i^2 - x_i)$ equal zero; finally, when v is sufficiently high to enforce $\sum_{j=1}^{n} x_j = k$, also the term $v(\sum_{j=1}^{n} x_j - k)$ equals zero. That is, also the pricing objective equals the objective function. □

Proposition 7 states that, for a sufficiently large value of v, the pricing problem solves *de facto* a kQKP . As a double check, in a preliminary experiment we therefore tested the following reformulation of $\mathcal{F}$:

$$(\mathcal{F}_M) \quad \max \quad f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_i x_j - M \left(\sum_{j=1}^{n} x_j - k\right)^2 \tag{28}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} a_j x_j \le b \tag{29}$$

$$x_j \in \{0,1\} \qquad j = 1, \dots, n \tag{30}$$

to solve the DD instances. We do not report the results in details because, as expected, using $\mathcal{F}_M$ instead of $\mathcal{F}$ turned out to be, in terms of computing time, way worse.

Summarizing, such a big-M approach proves not to be viable, but at the same time providing sufficiently high penalties for violating the cardinality constaint seems to be a key ingredient for a tight relaxation. We therefore argue that a critical issue is the choice of the multiplier v, that is by far non trivial. In the following, we report a set of experiments aimed at obtaining a better understanding of such a phenomenon.

		Average Dual. Gap			Average Time (s)			Average Feasible Cols			Column generation iter.		
$\delta(\%)$	n	$v = 0$	$v = \bar{v}$	$v = v*$	$v = 0$	$v = \bar{v}$	$v = v*$	$v = 0$	$v = \bar{v}$	$v = v*$	$v = 0$	$v = \bar{v}$	$v = v*$
25	50	-19.43%	0.00%	0.00%	0.3	63.5	48.0	15.00%	59.52%	73.33%	3.10	5.50	3.40
	60	-79.77%	-1.26%	0.00%	0.2	44.5	39.8	4.00%	44.65%	58.90%	3.80	5.50	4.40
	70	-16.80%	-4.08%	0.00%	1.2	59.3	65.7	0.00%	45.61%	68.57%	3.10	4.50	3.90
	80	-11.67%	0.00%	0.00%	198.7	95.7	84.0	9.00%	54.45%	71.07%	3.60	5.60	4.00
	90	-63.88%	0.00%	0.00%	296.9	166.7	165.4	0.00%	54.61%	73.17%	3.00	7.30	6.30
	100	-41.27%	0.00%	0.00%	8348.3	322.9	355.2	0.00%	50.52%	79.68%	3.90	8.40	7.50
50	50	-28.97%	0.00%	0.00%	0.1	51.5	49.4	15.00%	60.86%	76.67%	2.60	5.30	3.80
	60	-13.13%	0.00%	0.00%	0.3	58.2	46.4	33.33%	69.50%	83.57%	2.80	4.70	3.60
	70	-47.85%	-3.52%	0.00%	4.7	86.3	94.9	4.00%	53.74%	73.72%	2.70	5.30	5.30
	80	-42.69%	-2.79%	0.00%	0.9	121.7	120.7	0.00%	46.50%	78.57%	2.90	6.30	4.50
	90	-63.79%	0.00%	0.00%	191.6	279.2	216.4	10.00%	65.62%	77.42%	2.60	7.90	5.90
	100	-14.75%	0.00%	0.00%	491.3	1009.1	697.8	10.67%	59.90%	74.57%	2.70	7.80	7.10
75	50	-119.44%	0.00%	0.00%	0.0	31.0	27.4	0.00%	48.14%	66.67%	2.00	4.80	3.40
	60	-157.01%	0.00%	0.00%	0.1	42.2	35.1	10.00%	50.85%	68.24%	2.10	5.30	3.70
	70	-65.16%	0.00%	0.00%	0.2	86.9	71.0	0.00%	44.08%	73.57%	2.00	6.00	3.50
	80	-95.92%	-0.74%	0.00%	13.8	184.2	165.9	0.00%	53.60%	71.36%	2.10	6.50	5.90
	90	-59.35%	-0.27%	0.00%	16.9	1777.5	2044.9	0.00%	59.02%	81.79%	2.20	8.20	8.30
	100	-47.82%	-0.13%	0.00%	4.1	3431.6	2833.0	0.00%	52.99%	92.07%	2.10	8.50	10.30
100	50	-96.34%	-6.72%	0.00%	0.0	37.0	51.2	11.67%	42.33%	68.44%	1.90	4.40	4.60
	60	-77.53%	-0.50%	0.00%	0.1	48.9	60.4	0.00%	45.81%	79.83%	2.10	4.60	4.20
	70	-111.61%	-0.10%	0.00%	0.1	150.5	132.0	0.00%	49.93%	83.81%	1.70	6.70	5.80
	80	-74.78%	-0.58%	0.00%	0.1	368.5	294.0	0.00%	46.99%	91.03%	1.50	5.90	4.90
	90	-61.68%	0.00%	0.00%	0.2	528.8	519.3	0.00%	61.27%	79.24%	1.40	7.30	6.20
	100	-154.35%	-8.98%	0.00%	0.4	3137.1	1899.2	0.00%	43.74%	66.93%	1.90	8.00	6.10
Average		-65.20%	-1.19%	0.00%	399.2	507.6	421.5	5.13%	52.68%	75.51%	2.49	6.26	5.28

Table 6: DD set. Influence of v on the performances of $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$.

6.1 Choosing v

From a geometric point of view, the multiplier v fully controls the convexity of the pricing objective function, as the dual variables act only on its linear part. By setting $v = 0$, the pricing objective is convex in none of the DD instances, while large values of v ensure it to be convex. Indeed, we found that setting $v = v^*$ is sufficient to make each pricing objective convex in each of our test instances.

We therefore performed the following experiment: we compared the bounds and CPU times yielded by three versions of our algorithms, obtained by setting either $v = 0$, $v = \bar{v}$ or $v = v^*$, where $\bar{v}$ is the *minimum* value of v such that the pricing objective remains convex. The latter was found by means of a simple bisection routine in a preprocessing phase. The results on the DD set are reported in Table 6. The table includes one row for each value of density and size, reporting average values over the corresponding instances, and is composed by four blocks reporting, respectively, the average duality gaps (the ratio of the difference between the optimal solution and the dual bound obtained, to the bound obtained), the average CPU time spent in the column generation process, the fraction of columns in the final master problem for which the cardinality constraint is satisfied and the average number of column generation iterations to reach convergence. Each block includes three columns, one for each choice of the parameter v, as indicated in the leading row.

The results in Table 6 leads us to the following observations:

Experimental Observation 1. *Ensuring the pricing objective to be convex improves the quality of the final dual bound, but it is not enough to ensure the same high quality bounds achieved with $v = v^*$.*

Indeed, setting $v = 0$ leaves a huge duality gap, but also setting $v = \bar{v}$ still leaves an average duality gap of 1.19%.

Experimental Observation 2. *Choosing a high value of v tends to make the pricing problem harder to solve.*

The CPU time required after setting $v = 0$ is much lower than by setting $v = \bar{v}$ or $v = v^*$. Surprisingly, the strong bound obtained by setting $v = v^*$ can be obtained even faster than by setting $v = \bar{v}$.

Experimental Observation 3. *Decreasing v from v^* to $\bar{v}$ yields an increase in the total number of iterations needed to reach convergence, and therefore an overall increase in CPU time.*

Experimental Observation 4. *The higher is the value of v, the higher is the quantity of columns generated by the pricing algorithms that respect the cardinality constraint.*

Observation 4 is theoretically supported by Proposition 7. However, it is fundamental to notice that v^* is by far *not a big-M value*: even when $v = v^*$, we have that, in about one fourth of the times, the optimal column found by the pricing problem does not respect the cardinality constraint. That is, v^* helps to drive towards the feasibility region of the cardinality constraint without imposing to the column generated to belong to it. This is, in our opinion, the main rationale for the good computational behaviour of our method: to partially penalize the violation of the cardinality constraint with a QCR convexification of the objective function on one side, and to complete that penalization process with the feasibility region strengthening provided by DWD on the other side. This choice allows to obtain strong dual bounds without falling into the numerical troubles given by big-M penalization mechanisms.

6.2 Using heuristics

As reported above, in about one column generation every four the pricing problem finds as optimal one column violating the cardinality constraint. During the last column generation iteration, instead, a column representing an integer optimum for kQKP (thereby respecting also the cardinality constraint) was always generated.

Proposition 8. *If an optimal set of dual multipliers (τ^*, β^*) is known in advance, a single column generation iteration is sufficient to reach convergence (under the assumption that the optimal solution is unique).*

Regardless of the number of optimal solutions, Proposition 8 implies that once the values of the dual multipliers are close to the optimal ones, the convergence of the column generation will speed up significantly. Such an observation strongly motivates the use of a heuristic pricing algorithm: rather than solving exactly the pricing at each iteration, it is sufficient to heuristically find a positive reduced cost column to continue the column generation process, resorting to demanding exact methods only at the last iterations.

We indeed designed and tested the following pricing algorithm for $(\mathcal{F}^{\text{conv}}_{\overline{\text{DWD}(knap)}})$:

- No Heuristics `OPT`
  the pricing problem is solved at each iteration to optimality with CPLEX; the computation is stopped as soon as any column of positive reduced cost is found. This settings guarantees to have an exact separation procedure, because column generation never stops if positive reduced cost columns still exist.
- Early Stopping `ES`
  the pricing problem is solved at each iteration to optimality with CPLEX. However, a pricing solution value cutoff of 0.0 is set, and the optimization process is stopped as soon as a certain number of improving solutions are found, or when a certain time limit is reached.
- Truncated Branch-and-Bound Heuristic `TBB`
  a heuristic procedure based on the algorithm proposed in [9] is employed. In [9], the authors provide an algorithm to solve exactly the QKP with integer coefficients. We modified the algorithm to be able to handle continuous coefficients and we impose a time limit to its execution.

In our best pricing policy these pricing algorithms are used in sequence: first, `TBB` is called; if a solution with positive reduced cost is found within its time limit, the algorithm returns that solution and pricing stops. Otherwise, `ES` is called with a similar policy. `OPT` is called only if all other options fail. We experimented on various parameter setting in a preliminary round of tests, finding a timeout of $10s$ for `TBB` and $60s$ for `ES`, together with a solution limit of 1 in `ES` to produce best results.

In Table 7 we report the results comparing our best pricing policy (`TBB+ES+OPT`) with the policy using only `OPT` at each pricing iteration, that is turning off all pricing heuristics. Besides instance details, we include the number of instances for which the time limit was reached (column timeout) and the average number of column generation iterations (CG iter), restricting to those instances not hitting the time limit. Timeout was reached 21 times in absence of heuristics, proving their key importance.

We have also observed that our `TBB` was successful in a large number of pricing iterations, producing an optimal solution well before its time limit. However, in a few critical ones a radically different behaviour was observed. In an effort for understanding such a phenomenon, we compared the values of the dual variables in the iterations where `TBB` was successful to those in which either `ES` or `ES` and `OPT` were required. The corresponding results are reported in the two blocks of Table 8. Each block contains the average pricing time and the average value of γ and θ dual variables (see Section 3.3).

The results showed in Table 8 lead us to the following observation.

Experimental Observation 5. *The heuristic procedure `TBB` is able to find a column with positive reduced cost when the absolute values of the dual costs are significantly high.*

When the absolute values of the dual costs are high, the role of the dual variables is more significant in the objective function, this implies a higher variance of its coefficients. With such a huge gap among the coefficients, The heuristic procedure `TBB`

		TBB+ES+OPT		OPT only	
$\delta(\%)$	n	CG Iter	timeout	CG Iter	timeout
25	50	3.4	0	7.7	3
	60	4.4	0	5.7	0
	70	3.9	0	5.8	0
	80	4.0	0	6.6	2
	90	6.3	0	6.8	1
	100	7.5	0	7.6	1
50	50	3.8	0	5.9	1
	60	3.6	0	5.9	1
	70	5.3	0	7.1	1
	80	4.5	0	8.7	1
	90	5.9	0	8.7	1
	100	7.1	0	9.1	0
75	50	3.4	0	6.1	0
	60	3.7	0	6.5	0
	70	3.5	0	6.1	0
	80	5.9	0	6.9	1
	90	8.3	0	9.8	2
	100	10.3	0	7.6	3
100	50	4.6	0	6.6	0
	60	4.2	0	7.0	0
	70	5.8	0	8.9	1
	80	4.9	0	8.0	1
	90	6.2	0	6.8	0
	100	6.1	0	7.7	1
Avg.		5.3	0	7.2	21

Table 7: Impact of QKP pricing heuristics in $(\mathcal{F}^{conv}_{DWD(knap)})$

		TBB success			TBB failure		
$\delta(\%)$	n	γ	θ	Time	γ	θ	Time
25	50	-80294.40	0.00	7.21	298.65	0.00	18.10
	60	-206634.62	0.00	3.43	250.01	0.00	16.72
	70	-69565.17	0.00	9.23	-27575.31	418497.78	20.61
	80	-60485.72	0.00	10.77	-6853.32	101782.61	23.86
	90	-157250.41	0.00	5.85	379.09	0.00	35.77
	100	-100549.55	0.00	9.32	415.37	0.00	60.43
50	50	-78669.73	0.00	5.71	470.22	0.00	16.73
	60	-54558.69	0.00	5.15	511.07	0.00	18.49
	70	-92340.20	0.00	7.97	583.87	0.00	23.94
	80	-84350.57	0.00	8.55	471.64	0.00	36.89
	90	-92959.70	0.00	8.70	605.62	0.00	54.68
	100	-45970.56	0.00	10.63	-4668.16	86731.21	116.10
75	50	-169402.49	0.00	1.97	446.70	0.00	16.41
	60	-197580.50	0.00	1.87	422.70	0.00	20.64
	70	-101031.54	0.00	6.04	-17134.59	158696.32	29.12
	80	-94539.18	0.00	7.34	-7983.21	151077.11	37.67
	90	-48629.48	0.00	10.55	848.96	0.00	329.70
	100	-43727.78	0.00	11.68	811.47	0.00	333.23
100	50	-122817.48	0.00	2.88	544.26	0.00	18.81
	60	-121950.81	0.00	5.75	679.24	0.00	20.25
	70	-107172.20	0.00	8.70	801.18	0.00	27.86
	80	-116457.57	0.00	8.30	813.74	0.00	80.38
	90	-57083.79	0.00	10.23	-9996.81	142742.83	107.11
	100	-126918.45	0.00	7.92	-24053.72	549007.17	408.57
Avg.		-103870.43	0.00	7.18	-3549.88	70586.55	122.14

Table 8: Effect of the combinatorial pricing heuristics in $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$

can easily identify a core of promising items and hence it is able to quickly identify good solutions providing a column with positive reduced cost. Furthermore, using TBB heuristic, allows even to *reduce* the average number of column generation iterations, further stressing that a key point is to find a right penalization pattern.

The results concerning the pricing performed with the use of CPLEX provides less homogeneous information and hence are harder to read. However we report that the behaviour of CPLEX is somehow complementary to that of TBB: in preliminary tests we observed that CPLEX can incur in numerical problems when the coefficients of the objective function present a significant variance. Indeed, the results in Table 7 confirm this observation: our sequential heuristic pricing procedure is able to exploit the strength of both TBB and CPLEX in efficiently finding a column with positive reduced cost.

6.3 Convexity analysis

As mentioned in Section 3.1, the convexification of the objective function is not needed when the instance to be solved is convex. This observation suggests that convexity of the objective function is an aspect that should be analyzed more in detail.

In the following, we use the *fraction of positive eigenvalues* (κ) of the matrix representing the objective function ($Q + e \times L$) to measure the convexity of an instance of kQKP . Since we deal with problems in maximization form, an instance with a value of $\kappa = 0$ has an objective function with only negative eigenvalues and hence solving the corresponding continuous relaxation is a convex optimization problem. Instead, instances with values of $\kappa = 1$ have convex objective functions, yielding to nonconvex maximization problems.

All instances of the DD set have, by construction, a value of κ approximately equal to 0.5. To better understand the impact of convexity on the performance of our algorithms, we created the following second set of instances:

- Convexity-dependent (CD) instances.
  This set takes the 40 DD instances with $n = 100$ and modifies their objective function to obtain a set of instances with controlled values of κ. More precisely, for each original instance, we generate 11 new instances with the same values of a, b and k and with $\kappa = 0.0, 0.1, 0.2, \ldots, 0.9, 1.0$. The procedure (similar to the one described in [8]) used to obtain an objective function with a given convexity κ^* is the following:
  - We define a $n \times n$ matrix B' of real values randomly generated in the range $[-1, 1]$, and we orthonormalize it. The vectors composing B' are then used as eigenvectors of a matrix B''.
  - We randomly draw $\lceil \kappa^* \cdot n \rceil$ values in the range $[-1, 0]$ and $(n - \lceil \kappa^* \cdot n \rceil)$ values in the range $[0, 1]$, to be set as eigenvalues of B''.
  - We rebuild the matrix B'' using the spectral decomposition theorem.
  - We set $L = I \cdot B''$ and $Q = B'' - L$
  - We sort the diagonal values of L according to the order induced by the values w_j. This last reordering is done in order to have the linear objective terms correlated with the item weights.

	Time			
κ(%)	Avg. CPU	Max. CPU	timeout	feas. cols
0	4157.39	29995.30	1	72.45%
10	4372.54	33937.50	1	72.12%
20	4810.76	34362.20	1	75.27%
30	4491.33	37402.60	2	72.74%
40	6220.74	36431.80	1	73.55%
50	6264.31	36033.80	3	71.36%
60	5790.34	30320.00	2	68.57%
70	7262.65	34574.80	1	71.71%
80	9639.85	37713.30	6	71.08%
90	10291.02	38285.00	3	73.86%
100	4660.53	39224.50	1	70.90%
Avg.	6178.31	39224.50	22	72.15%

Table 9: Results on CD instances of $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$.

This procedure leads to a total of 440 instances.

In Table 9 we report the results concerning the CD instances. In each row, we provide the average computing time (Avg. CPU) and the maximum computing time of an instance in the set (Max CPU), the number of instances that are not solved within the time limit (timeout) and the fraction of columns respecting the cardinality constraint in the final master problem (feas. cols.). For the sake of readability, the content of column "Avg. CPU" is also plotted in Figure 2: the values of κ are reported on the horizontal axis, while the average CPU time of the corresponding instances is reported on the vertical axis.

The results in Table 9 and in Figure 2 lead to the following statement:

Experimental Observation 6. *Instances whose objective function is perfectly convex or perfectly concave are easier to solve than instances with a fractional value of κ. Moreover, among the instances with fractional values of κ, the one with $\kappa = 0.8$ and $\kappa = 0.9$ are the most difficult to solve by $(\mathcal{F}^{conv}_{DWD(knap)})$.*

In general, excluding the case $\kappa = 1$, the problem complexity seems to increase as the κ value increases.

7 Conclusion

We presented a first comprehensive methodological and experimental study on the application of Dantzig Wolfe Decomposition (DWD) to Binary Quadratic Problems (BQP), by combining it with objective function convexification techniques like the Quadratic Convex Reformulation (QCR). As case study we considered the cardinality constrained quadratic knapsack problem (kQKP). In particular we proposed a series of reformulations arising after applying DWD and QCR in diverse combinations, and we framed them in a hierarchy of dominance, based on the theoretical strength

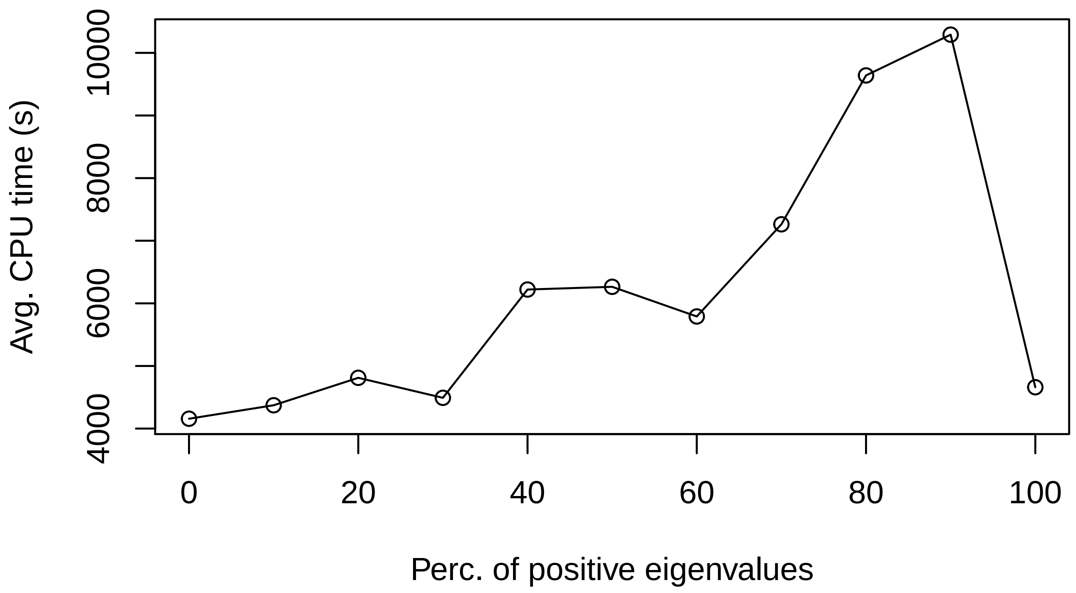

Figure 2: Average CPU time of $(\mathcal{F}^{\text{conv}}_{\text{DWD}(knap)})$ on `CD` instances.

of their continuous relaxation. We also tested the computational behaviour of the proposed reformulations on a large set of instances from the literature. We found a reformulation mechanism of particular interest, since the continuous relaxation of the models produced in this way were computed in competitive computational time, and always produced solutions automatically respecting the integrality conditions, thereby experimentally behaving like an exact integer optimization method.

Our in depth analysis of reformulations revealed that DWD and QCR are complementary in effectively handling the equality and inequality constraints of kQKP .

In particular, the equality constraint is of key importance in the convexification process of the objective function; a side effect of accurate convexification and DWD variables projection is then to implicitly enforce the same equality to be respected in a column generation framework already at a pricing stage. This effect is obtained by suitable penalties, whose main component is fixed by a single semidefinite optimization in a preprocessing step, and then corrected at each column generation iteration by the master dual variables. The combination of fixed and correction components, together with the possibility offered by DWD of applying pricing heuristics, allows to avoid undesirable "big-M" effects.

DWD is finally needed to strengthen the formulation by tightening the inequality constraints, explicitly handing them at a pricing stage.

Our results suggest, in principle, that our methods can be extended to generic BQPs by simply partitioning the set of constraints splitting equalities and inequalities, clearly highlighting their potential as elements of a black box solver. Our future research will therefore focus on computationally evaluating such an extension.

References

[1] Warren P. Adams, Richard J. Forrester, and Fred W. Glover. Comparisons and enhancement strategies for linearizing mixed 0-1 quadratic programs. *Discret. Optim.*, 1(2):99–120, November 2004.

[2] M. Aganagic and S. Mokhtari. Security constrained economic dispatch using nonlinear dantzig-wolfe decomposition. *IEEE Transactions on Power Systems*, 12(1):105–112, 1997.

[3] Aykut Ahlatçioglu, Michael R. Bussieck, Mustafa Esen, Monique Guignard, Jan-Hendrick Jagla, and Alexander Meeraus. Combining QCR and CHR for convex quadratic pure 0-1 programming problems with linear constraints. *Annals OR*, 199(1):33–49, 2012.

[4] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1130–1154, 1980.

[5] A. Billionnet, S. Elloumi, and M.-C. Plateau. Improving the performance of standard solvers via a tighter convex reformulation of constrained quadratic 0-1 programs: the QCR method. *Discrete Applied Mathematics*, 157:1185–1197, 2009.

[6] A. Billionnet and E. Soutif. An exact method based on lagrangean decomposition for the 0-1 quadratic knapsak problem. *European Journal of Operational Research*, 157:565–575, 2004.

[7] Alain Billionnet, Sourour Elloumi, and Amélie Lambert. Extending the QCR method to general mixed-integer programs. *Mathematical Programming*, 131(1-2):381–401, 2012.

[8] Christoph Buchheim and Angelika Wiegele. Semidefinite relaxations for non-convex quadratic mixed-integer programming. *Math. Program.*, 141(1-2):435–452, 2013.

[9] Alberto Caprara, David Pisinger, and Paolo Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

[10] GB. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.

[11] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors. *Column generation.* GERAD 25th anniversary series. Springer, New York, 2005. GERAD : Groupe d'tudes et de recherche en analyse des dcisions.

[12] Guy Desaulniers, Jacques Desrosiers, and Marius M Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[13] D. Fayard and G. Plateau. Algorithm 47: an algorithm for the solution of the 0-1 knapsack problem. *Computing*, 28:269–287, 1982.

[14] R. Fortet. Applications de l'algèbre de boole en recherche opérationnelle. *Revue Française de Recherche Opérationnelle*, 4:17–26, 1960.

[15] F. Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[16] Serigne Gueye and Philippe Michelon. "miniaturized" linearizations for quadratic 0/1 problems. *Annals of Operations Research*, 140(1):235–261, 2005.

[17] Gurobi Optimization. Gurobi, 2016. Version 7.0, http://www.gurobi.com/.

[18] Kaj Holmberg. Minlp: Generalized cross decomposition. In Christodoulos A. Floudas and Panos M. Pardalos, editors, *Encyclopedia of Optimization*, pages 2148–2155. Springer, 2009.

[19] IBM. Cplex, 2016. Version 12.6.3, https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[20] S. Ji, X. Zheng, and X. Sun. An improved convex 0-1 quadratic program reformulation for chance-constrained quadratic knapsack problems. *Asia-Pacific Journal of Operational Research*, 30(3):1340009, 2013.

[21] M. Jünger, Th.M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors. Springer, 2009.

[22] Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Trans. Math. Softw.*, 43(4):32:1–32:23, January 2017.

[23] Siriphong Lawphongpanich. Simplicial with truncated Dantzig-Wolfe decomposition for nonlinear multicommodity network flow problems with side constraints. *Operations Research Letters*, 26(1):33–41, 2000.

[24] Lucas Létocart, Marie-Christine Plateau, and Gérard Plateau. An efficient hybrid heuristic method for the 0-1 exact k-item quadratic knapsack problem. *Pesquisa Operacional*, 34(1):49–72, 2014.

[25] Lucas Létocart and Angelika Wiegele. Exact solution methods for the k-item quadratic knapsack problem. In *Lecture Notes in Computer Science*, volume 9849 of *Combinatorial Optimization*, pages 166–176. Springer, 2016.

[26] Leo Liberti. Compact linearization for binary quadratic problems. *4OR*, 5(3):231–245, 2007.

[27] R. Misener and C. Floudas. GloMIQO: Global Mixed-Integer Quadratic Optimizer. *Journal of Global Optimization*, 57:3–50, 2013.

[28] D. Pisinger. The quadratic knapsack problem: a survey. *Discrete Applied Mathematics*, 155:623–648, 2007.

[29] D. Pisinger, A. B. Rasmussen, and R. Sandvik. Solution of large quadratic knapsack problems through agressive reduction. *INFORMS Journal on Computing*, 19(2):280–290, 2007.

[30] David Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45(5):758–767, 1997.

[31] M.-C. Plateau. Quadratic convex reformulations for quadratic 0-1 programming. *4OR*, 6:187–190, 2008.

[32] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. *Math. Programming*, 121(2):307, 2010.

[33] SCIP. Scip, 2016. Version 3.2.1, http://scip.zib.de/.

[34] H.D. Sherali and W.P. Adams. A tight linearization and an algorithm for 0-1 quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.

[35] François Vanderbeck and Martin W. P. Savelsbergh. A generic view of dantzig-wolfe decomposition in mixed integer programming. *Oper. Res. Lett.*, 34(3):296–306, May 2006.

A Semidefinite model providing u^* and v^*

To obtain these optimal parameters, Billionnet et al. [5, 31] prove that the following SDP relaxation of $(\mathcal{F})$ has to be solved:

$$\mathcal{F}_{(SDP)} \begin{cases} \max & \sum_{i=1}^n \sum_{j=1}^n c_{ij} X_{ij} \\ \text{s.t.} & X_{ii} = x_i \qquad i = 1, \dots, n \quad (50) \\ & \sum\limits_{i=1}^n \sum\limits_{j=1}^n X_{ij} - 2k \sum\limits_{j=1}^n x_j = -k^2 \quad (51) \\ & \sum_{j=1}^n a_j x_j \le b \\ & \sum_{j=1}^n x_j = k \\ & \begin{pmatrix} 1 & x^t \\ x & X \end{pmatrix} \geq 0 \\ & x \in \mathbb{R}^n, \; X \in \mathbb{R}^{n \times n} \end{cases}$$

Let us note that in this model, the classical semidefinite constraint has been relaxed and the product constraints (51) which consist of multiplying the equality cardinality constraint by each variable x_i, have been introduced to tighten the optimal value.

Namely, the optimal values u_i^* $(i = 1, \dots, n)$ and v^* of problem (P_{QCR}) are simply given by the optimal values of the dual variables of $(\mathcal{F}_{SDP})$ associated with constraints (50) (resp. (51)).

B An improved convex 0-1 quadratic program reformulation

In this section, we present the result obtained by [20] on constructing the improved convex 0-1 quadratic program reformulation. They employ matrix decomposition and piecewise linear representation of quadratic term for 0-1 variables and they show that the optimal matrix decomposition can be found by solving a SDP problem.

Let $\mathcal{P}$ and $\mathcal{N}$ denote the sets of non-negative and non-positive $n \times n$ matrices, i.e.,

$$\mathcal{P} := \{P = (P_{ij}) \in \mathbb{R}^{n \times n} | P \geq 0\}.$$
$$\mathcal{N} := \{N = (N_{ij}) \in \mathbb{R}^{n \times n} | N \leq 0\}.$$

They consider the following decomposition of $f(x)$:

$$f(x) = x^T(C - M)x + x^T M x,$$

where $C - M \preceq 0$ and $M = Diag(u) + P + N \in \mathcal{S}$ with $u \in \mathbb{R}^n$, $P \in \mathcal{P}$ and $N \in \mathcal{N}$. Let variables y_{ij} represent the product of variables x_i and x_j; for any $x_i, x_j \in \{0,1\}$, we impose

$$y_{ij} = x_i \cdot x_j$$

that in turn can be linearized by enforcing

$$y_{ij} = \min(x_i, x_j) \text{ and } y_{ij} = \max(0, x_i + x_j - 1).$$

Thus, for any $x \in \{0,1\}^n$, they can rewrite f(x) as

$$\begin{aligned} q(x) &= x^T(Q - Diag(u) - P - N)x + x^T(Diag(u) + P + N)x \\ &= x^T(Q - Diag(u) - P - N)x + u^T x + \textstyle\sum_{i,j=1}^n [P_{ij}y_{ij} + N_{ij}y_{ij}] \\ &= x^T(Q - Diag(u) - P - N)x + u^T x + \textstyle\sum_{i,j=1}^n [P_{ij}s_{ij} + N_{ij}t_{ij}] \end{aligned}$$

where $s_{ij} = \min(x_i, x_j)$ and $t_{ij} = -\max(0, x_i + x_j - 1)$.

They obtain the following equivalent reformulation :

$$\begin{array}{lll} \max & f(x) = x^T(C - Diag(u) - P - N)x + u^T x + \sum_{i,j=1}^n [P_{ij}s_{ij} + N_{ij}t_{ij}] & \\ \text{s.t.} & \sum_{j=1}^n a_j x_j \leq b & (3) \\ & \sum_{j=1}^n x_j = k & (4) \\ & s_{ij} = \min(x_i, x_j), t_{ij} = \max(0, x_i + x_j - 1) \quad i,j = 1,\dots,n & \\ & x_j \in \{0,1\} \quad j = 1,\dots,n & \end{array}$$

where $Q - Diag(u) - P - N \preceq 0$, $P \in \mathcal{P}$ and $N \in \mathcal{N}$. Since $P_{ij} \geq 0$, the constraint $s_{ij} = min(x_i, x_j)$ can be relaxed to two linear inequalities $s_{ij} \leq x_i$ and $s_{ij} \leq x_j$ without affecting the optimal solution of the above problem. Similarly, we can relax $t_{ij} = max(0, x_i + x_j - 1)$ to $t_{ij} \geq 0$ and $t_{ij} \geq x_i + x_j - 1$.
Therefore, the above reformulation is equivalent to the following convex 0-1 quadratic program:

$$\begin{array}{lll} \max & f(x) = x^T(C - Diag(u) - P - N)x + u^T x + \sum_{i,j=1}^n [P_{ij}s_{ij} + N_{ij}t_{ij}] & \\ \text{s.t.} & \sum_{j=1}^n a_j x_j \leq b & (3) \\ & \sum_{j=1}^n x_j = k & (4) \\ & s_{ij} \leq x_i, \ s_{ij} \leq x_j \quad i,j = 1,\dots,n & \\ & t_{ij} \leq 0, \ t_{ij} \geq x_i - x_j - 1 \quad i,j = 1,\dots,n & \\ & x_j \in \{0,1\} \quad j = 1,\dots,n & \end{array}$$

By combining QCR and this reformulation, the objective function can be further rewritten as

$$f_{u,v,P,N}(x, s, t)$$

and the optimal parameters (u, v, P, N) can be found by solving the following SDP problem :

$$(\mathcal{F}_{SDP_{EXT}})\left\{\begin{array}{lllr} \max & \sum_{i=1}^n \sum_{j=1}^n c_{ij} X_{ij} & & \\ \text{s.t.} & X_{ii} = x_i & i = 1, \ldots, n & (50) \\ & \sum\limits_{i=1}^n \sum\limits_{j=1}^n X_{ij} - 2k \sum\limits_{j=1}^n x_j = -k^2 & i = 1, \ldots, n & (51) \\ & \sum_{j=1}^n a_j x_j \leq b & & \\ & \sum_{j=1}^n x_j = k & & \\ & X_{ij} \leq x_i,\ X_{ij} \leq x_j & i, j = 1, \ldots, n & (52) \\ & X_{ij} \geq x_i + x_j - 1,\ X_{ij} \geq 0 & i, j = 1, \ldots, n & (53) \\ & \begin{pmatrix} 1 & x^t \\ x & X \end{pmatrix} \succeq 0 & & \\ & x \in \mathbb{R}^n,\ X \in S_n & & \end{array}\right.$$

with x^*, u^*, v^*, P^*, N^* an optimal dual solution of $(\mathcal{F}^{\text{impr. conv}})$.

Also $(\mathcal{F}^{\text{impr. conv}})$ presents the big advantage of being convex. Moreover, in comparison to $(\mathcal{F}^{\text{conv}})$, it presents a stronger continuous relaxation.
It is easy to see that, without loss of generality, we can always assume that either $P_{ij} > 0$ or $N_{ij} < 0$; at the same time, each s_{ij} (resp. t_{ij}) variable whose corresponding P_{ij} (resp. N_{ij}) coefficient is zero can be simply dropped, and therefore the remaning s_{ij} and t_{ij} variables can be viewed as a partition of the y_{ij} variables.