Linear Reformulations of Integer Quadratic Programs

Alain Billionnet and Sourour Elloumi and Amélie Lambert

February 21, 2010

Abstract

Let (QP) be an integer quadratic program that consists in minimizing a quadratic function subject to linear constraints. In this paper, we present several linearizations of (QP). Many linearization methods for the quadratic 0-1 programs are known. A natural approach when considering (QP) is to reformulate it into a quadratic 0-1 program. However, this method, that we denote `BBL` (Binary Binary Linearization), leads to a quadratic program with a large number of variables and constraints.

Our new approach, `BIL` (Binary Integer Linearization), consists in reformulating (QP) into a particular quadratic integer program where each quadratic term is the product of an integer variable by a 0-1 variable. The obtained integer linear program is significantly smaller than in the `BBL` approach.

Each reformulation leads to an integer linear program that we improve by adding valid inequalities. Finally, we get 4 different programs that we compare from the computational point of view.

keywords : Integer programming, quadratic programming, linear reformulations

1 Introduction

Consider the following linearly-constrained integer quadratic program:

$$(QP) \begin{cases} Min & f(x) = x^T Qx + c^T x \\ s.t & x \in X \subset \mathbb{N}^n \end{cases}$$

with $Q \in \mathbf{S}_n$ (space of symmetric matrices of order n), $c \in \mathbb{R}^n$ and X is defined as the set of integer solutions of a system of linear equalities and inequalities:

$$X = \left\{ x : \begin{array}{lll} Ax = b & & (1) \\ Dx \leq e & & (2) \\ x_i \leq u_i & i \in I & (3) \\ x_i \geq 0 & i \in I & (4) \\ x_i \in \mathbb{N} & i \in I & (5) \end{array} \right.$$

where $A \in \mathbf{M}_{m,n}$ (set of $m*n$ integer matrices), $b \in \mathbb{N}^m$, $D \in \mathbf{M}_{p,n}$, $e \in \mathbb{N}^p$, $u \in \mathbb{N}^n$, $I = \{i \,:\, i = 1,\ldots,n\}$. Without loss of generality, we shall suppose X non empty.
We denote $R = \{r \,:\, r = 1,\ldots,m\}$, $S = \{s \,:\, s = 1,\ldots,p\}$, $E = \{(i,k) \,:\, i = 1,\ldots,n,\ k = 0,\ldots \lfloor log(u_i)\rfloor\}$ and $N = |E| = \sum_{i=1}^{n}(\lfloor log(u_i)\rfloor + 1)$.

A lot of applications in operations research and industrial engineering involve discrete variables in their formulation. Some of these applications can be formulated as (QP). For instance, such a formulation is used in [1] for the chaotic mapping of complete multipartite graphs.

In the state-of-the-art, a majority of resolution methods of quadratic discrete problems are designed only for quadratic 0-1 programs. This is why a natural way to solve (QP) consists in replacing each integer variable by its binary decomposition. The number of additional variables is hence equal to N. Thereafter each integer product becomes an expression of binary products, that we standardly linearize. The idea of the standard 0-1 linearization [2] consists in adding a set of new variables and a family of inequalities that we substitute to the binary quadratic terms. The main drawback of this approach, that we call `BBL` (Binary Binary Linearization) is that the size of the obtained linear problem is $O(N^2)$. Possible improvements of the standard 0-1 linearization were introduced by Sherali and Adams [3] and consist in adding a family of valid inequalities. These improvements can be easily applied to the `BBL` approach, giving a reinforced linearization method that we call `BBLr`.

Our new approach, that we call `BIL` (Binary Integer Linearization), consists also in replacing each integer variable by its binary decomposition. Then, in each product of two different integer variables we replace only one of them by its binary decomposition. Thus, each integer product becomes an expression of products of a binary variable by an integer one. Finally, we linearize these new products by the standard binary-integer linearization [4]. The `BIL` approach hence leads to an integer linear program of size $O(nN)$ that is significantly smaller than the program of size $O(N^2)$ provided by the `BBL` method. Moreover, we improve this reformulation in term of integrality gap, by adding new valid inequalities. We denote by `BILr` the reinforced version of the `BIL` method.

Finally, we get 4 linear reformulations that we compare from the computational point of view. Our experimentations are carried out on the Integer Quadratic Knapsack Problem (IQKP).

The paper is organized as follows. In Section 2, we present the `BBL` approach and its reinforcement `BBLr`. In Section 3, we describe the `BIL` approach and its reinforcement `BILr`. Finally, in Section 4, we present our computational study of these different methods. Section 5 is a conclusion.

2 The BBL approach

Let $x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik}$ be the unique binary decomposition of x_i. We replace the x_i variables by the set of t_{ik} binary variables. Then each product $x_i x_j$ leads to an expression of products $t_{ik} t_{jl}$, that we linearize by adding new binary variables y_{ikjl}. We obtain the following program:

$$
(LP_{\texttt{BBL}}) \begin{cases}
Min & f_{\texttt{BBL}}(x,y) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ q_{ij} \neq 0}}^{n} q_{ij} \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^{k+l} y_{ikjl} + \sum_{i=1}^{n} c_i x_i \\
s.t. & (1)(2)(3) \\
& x_i = \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k t_{ik} \quad i \in I \quad (6) \\
& y_{ikjl} \leq t_{ik} \quad (i,k),(j,l) \in E,\ q_{ij} < 0 \quad (7) \\
& y_{ikjl} \leq t_{jl} \quad (i,k),(j,l) \in E,\ q_{ij} < 0 \quad (8) \\
& y_{ikjl} \geq t_{ik} + t_{jl} - 1 \quad (i,k),(j,l) \in E,\ q_{ij} > 0 \quad (9) \\
& y_{ikjl} \geq 0 \quad (i,k),(j,l) \in E,\ q_{ij} > 0 \quad (10) \\
& y_{ikjl} = y_{jlik} \quad (i,k),(j,l) \in E,\ i < j\ q_{ij} \neq 0 \quad (11) \\
& y_{ikik} = t_{ik} \quad (i,k) \in E,\ q_{ii} \neq 0 \quad (12) \\
& y_{ikil} = y_{ilik} \quad (i,k),(i,l) \in E,\ k < l,\ q_{ii} \neq 0 \quad (13) \\
& t_{ik} \in \{0,1\} \quad (i,k) \in E \quad (14)
\end{cases}
$$

Observe that for any optimal solution of $(LP_{\texttt{BBL}})$, as variables y_{ikjl} are present only in the objective function and in Constraints (7)-(13), the following properties are satisfied:

- If $q_{ij} < 0$ then $y_{ikjl} = \min(t_{ik}, t_{jl})$
- If $q_{ij} > 0$ then $y_{ikjl} = \max(0, t_{ik} + t_{jl} - 1)$

ensuring y_{ikjl} to be equal to the product $t_{ik} t_{jl}$ if Constraints (14) are satisfied. Constraints (11) and (13) follow from the equality $t_{ik} t_{jl} = t_{jl} t_{ik}$. Constraints (12) follow from the property that if $t_{ik} \in \{0,1\}$ then $t_{ik}^2 = t_{ik}$.

The size of $(LP_{\texttt{BBL}})$ is $O(N^2)$. As the y_{ikjl} variables and related constraints are not defined when $q_{ij} = 0$, the actual size depends on the density of matrix Q. In our computational results of Section 4, matrix Q is fully dense.

Improving the BBL approach

Here we improve the BBL approach by adding valid inequalities in $(LP_{\texttt{BBL}})$ following the same ideas as in [3]. We generate valid inequalities by multiplying the initial constraints (1) and (2) by the binary variables, then we linearize the obtained quadratic constraints. We obtain the following reinforced program:

$$
(LP_{\texttt{BBLr}})\begin{cases}
Min & f_{\texttt{BBLr}}(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} \sum_{k=0}^{\lfloor log(u_i)\rfloor}\sum_{l=0}^{\lfloor log(u_j)\rfloor} 2^{k+l} y_{ikjl} + \sum_{i=1}^{n} c_i x_i & & \\
s.t. & (1)(2)(3)(6)(14) & & \\
 & y_{ikjl} \leq t_{ik} & (i,k),(j,l) \in E & (7') \\
 & y_{ikjl} \leq t_{jl} & (i,k),(j,l) \in E & (8') \\
 & y_{ikjl} \geq t_{ik} + t_{jl} - 1 & (i,k),(j,l) \in E & (9') \\
 & y_{ikjl} \geq 0 & (i,k),(j,l) \in E & (10') \\
 & y_{ikjl} = y_{jlik} & (i,k),(j,l) \in E,\ i < j & (11') \\
 & y_{ikik} = t_{ik} & (i,k) \in E & (12') \\
 & y_{ikil} = y_{ilik} & (i,k),(i,l) \in E,\ k < l & (13') \\
 & \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k a_{ri} y_{ikjl} = b_r t_{jl} & (j,l) \in E,\ r \in R & (15) \\
 & \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k d_{si} y_{ikjl} \leq e_s t_{jl} & (j,l) \in E,\ s \in S & (16) \\
 & \sum_{i=1}^{n} d_{si} x_i - \sum_{i=1}^{n}\sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k d_{si} y_{ikjl} \leq e_s (1 - t_{jl}) & (j,l) \in E,\ s \in S & (17)
\end{cases}
$$

We multiply the equality Constraints (1) by variable t_{jl} to get Constraints (15). Similarly, we multiply the inequality Constraints (2) by t_{jl} (resp. $(1 - t_{jl})$) to get Constraints (16) (resp. (17)). Doing this introduces variables y_{ikjl} in the new constraints (15)-(17). Hence we need to define Constraints (7')-(13') independently from the sign of q_{ij}. Moreover, variables y_{ikjl} become needed even when $q_{ij} = 0$. The size of $(LP_{\texttt{BBLr}})$ does no longer depend on the density of matrix Q.

3 The `BIL` approach

Here again we use the unique binary decomposition $x_i = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik}$. We linearize the square terms x_i^2 by use of variables y_{ikil} that represent the product $t_{ik}t_{il}$ as in the `BBL` approach. However, for quadratic terms $x_i x_j$ with $i \neq j$, we use the equality $x_i x_j = \sum_{k=0}^{\lfloor log(u_i)\rfloor} 2^k t_{ik} x_j$, that we linearize by introducing new variables z_{ijk} to replace each quadratic term $t_{ik}x_j$. Then we add a set of inequalities that ensure z_{ijk} to be equal to $t_{ik}x_j$. We obtain the following program:

$$
(LP_{\texttt{BIL}}) \begin{cases}
Min & f_{\texttt{BIL}}(x,y,z) & & \\
s.t & (1)(2)(3)(6)(14) & & \\
 & z_{ijk} \leq u_j t_{ik} & (i,k) \in E,\ j \in I,\ q_{ij} < 0,\ i \neq j & (18) \\
 & z_{ijk} \leq x_j & (i,k) \in E,\ j \in I,\ q_{ij} < 0,\ i \neq j & (19) \\
 & z_{ijk} \geq x_j - u_j(1 - t_{ik}) & (i,k) \in E,\ j \in I,\ q_{ij} > 0,\ i \neq j & (20) \\
 & z_{ijk} \geq 0 & (i,k) \in E,\ j \in I,\ q_{ij} > 0,\ i \neq j & (21) \\
 & y_{ikik} = t_{ik} & (i,k) \in E,\ q_{ii} \neq 0 & (22) \\
 & y_{ikil} = y_{ilik} & (i,k),(i,l) \in E,\ k < l,\ q_{ii} \neq 0 & (23) \\
 & y_{ikil} \leq t_{ik} & (i,k),(i,l) \in E,\ q_{ii} < 0 & (24) \\
 & y_{ikil} \leq t_{il} & (i,k),(i,l) \in E,\ q_{ii} < 0 & (25) \\
 & y_{ikil} \geq t_{ik} + t_{il} - 1 & (i,k),(i,l) \in E,\ q_{ii} > 0 & (26) \\
 & y_{ikil} \geq 0 & (i,k),(i,l) \in E,\ q_{ii} > 0 & (27)
\end{cases}
$$

with

$$
f_{\texttt{BIL}}(x,y,z) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ q_{ij} \neq 0 \\ i \neq j}}^{n} q_{ij} \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} + \sum_{i=1}^{n} c_i x_i + \sum_{\substack{i=1 \\ q_{ii} \neq 0}}^{n} q_{ii} \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_i) \rfloor} 2^{k+l} y_{ikil}
$$

In any optimal solution of program $(LP_{\texttt{BIL}})$ we have:
- If $q_{ij} < 0$ then $z_{ijk} = \min(u_j t_{ik}, x_j)$
- If $q_{ij} > 0$ then $z_{ijk} = \max(0, u_j t_{ik} + x_j - u_j)$

it follows that, if $t_{ik} = 0$ then $z_{ijk} = 0$ and if $t_{ik} = 1$ then $z_{ijk} = x_j$. This proves that in any optimal integer solution, $z_{ijk} = t_{ik} x_j$. For the same reason as for program $(LP_{\texttt{BBL}})$ we also have $y_{ikil} = t_{ik} t_{il}$. Hence program $(LP_{\texttt{BIL}})$ is a mixed integer linear program that is equivalent to (QP).

The `BIL` approach produces program $(LP_{\texttt{BIL}})$ with $O(nN)$ variables and constraints. Here again, it is not necessary to define z_{ijk} when $q_{ij} = 0$. The actual size depends on the density of matrix Q.

Improving the `BIL` approach

We mainly add Constraints (28)-(35) and variables z_{iik} that represent $t_{ik} x_i$. We also need to transform Constraints (18)-(27) into Constraints (18')-(27'). All this give the following integer linear program $(LP_{\texttt{BILr}})$.

$$
(LP_{\text{BILr}})\left\{
\begin{array}{lllr}
Min & f_{\text{BILr}}(x,z) = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ i \neq j}}^{n} q_{ij} \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} + \sum_{i=1}^{n} q_{ii} \sum_{k=0}^{\lfloor log(u_i) \rfloor} \sum_{l=0}^{\lfloor log(u_i) \rfloor} 2^{k+l} y_{ikil} + \sum_{i=1}^{n} c_i x_i & & \\
s.t & (1)(2)(3)(6)(14) & & \\
 & z_{ijk} \leq u_j t_{ik} & (i,k) \in E, \ j \in I & (18') \\
 & z_{ijk} \leq x_j & (i,k) \in E, \ j \in I & (19') \\
 & z_{ijk} \geq x_j - u_j(1 - t_{ik}) & (i,k) \in E, \ j \in I & (20') \\
 & z_{ijk} \geq 0 & (i,k) \in E, \ j \in I & (21') \\
 & y_{ikik} = t_{ik} & (i,k) \in E & (22') \\
 & y_{ikil} = y_{ilik} & (i,k),(i,l) \in E, \ k < l & (23') \\
 & y_{ikil} \leq t_{ik} & (i,k),(i,l) \in E & (24') \\
 & y_{ikil} \leq t_{il} & (i,k),(i,l) \in E & (25') \\
 & y_{ikil} \geq t_{ik} + t_{il} - 1 & (i,k),(i,l) \in E & (26') \\
 & y_{ikil} \geq 0 & (i,k),(i,l) \in E & (27') \\
 & \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} = \sum_{l=0}^{\lfloor log(u_j) \rfloor} 2^l z_{jil} & i,j \in I & (28) \\
 & \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk} \geq x_i u_j + x_j u_i - u_i u_j & (i,k) \in E, \ j \in I & (29) \\
 & z_{iik} = \sum_{l=0}^{\lfloor log(u_i) \rfloor} 2^l y_{ikil} & (i,k) \in E & (30) \\
 & \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{iik} \geq x_i & i \in I & (31) \\
 & \sum_{i=1}^{n} a_{ri} z_{jil} = b_r t_{jl} & (j,l) \in E, \ r \in R & (32) \\
 & \sum_{i=1}^{n} d_{si} z_{jil} \leq e_s t_{jl} & (j,l) \in E, \ s \in S & (33) \\
 & \sum_{i=1}^{n} (d_{si} x_i - d_{si} z_{jil}) \leq e_s (1 - t_{jl}) & (j,l) \in E, \ s \in S & (34) \\
 & \sum_{i=1}^{n} (d_{si} x_i u_j - d_{si} \sum_{k=0}^{\lfloor log(u_i) \rfloor} 2^k z_{ijk}) \leq e_s (u_j - x_j) & j \in I, \ s \in S & (35)
\end{array}
\right.
$$

Here we describe how we get the above valid inequalities (28)-(35):

- Constraints (28) follow from the fact that in any product $x_i x_j$ either x_i or x_j can be replaced by its binary decomposition.
- Constraints (29) follow from the inequality $(x_i - u_i)(x_j - u_j) \geq 0$.
- Constraints (30) define variables z_{iik} that represent $t_{ik} x_i$ for an integer solution.
- Constraints (31) follow from inequality $x_i^2 \geq x_i$ that is satisfied by any integer x_i.
- Constraints (32) are obtained by multiplying the initial equality Constraints (1) by t_{jl}.
- Constraints (33) are obtained by multiplying the initial inequality Constraints (2) by t_{jl}.
- Constraints (34) are obtained by multiplying the initial inequality Constraints (2) by $(1 - t_{jl})$.

- Constraints (35) are obtained by multiplying the initial inequality Constraints (2) by $(u_j - x_j)$.

As in the BBLr method, the multiplication of Constraints (1) and (2) by the variables introduces variables z_{ijk} in the new constraints (32)-(35). This is why we need to define Constraints (18')-(27') independently from the sign of q_{ij}. Moreover, variables z_{ijk} become required even when $q_{ij} = 0$.

4 Computational results

We choose to perform numerical experiments on the Integer Quadratic Knapsack Problem $(IQKP)$ that consists in minimizing a quadratic function subject to a linear inequality constraint:

$$(IQKP) \begin{cases} Min & f(x) = x^T Q x + c^T x & \\ s.t & \sum_{i=1}^{n} d_i x_i \leq e & \\ & 0 \leq x_i \leq u_i & i \in I \\ & x_i \in \mathbb{N} & i \in I \end{cases}$$

We generate instances with 10, 20, and 30 variables. The coefficients are randomly generated as follows:

- the coefficients of Q and c are reals in the interval $[-100, 100]$
- the d_i coefficients are integers in the interval $[1, 50]$
- e is equal to $20 * \sum_{i=1}^{n} d_i$
- we generate a first class of instances, $(IQKP_1)$, with all $u_i = 50$, and a second class, $(IQKP_2)$, with all $u_i = 100$.

For any size $n = 10$, 20, or 30, we generate 5 instances in each class giving a total of 30 instances.

Our experiments are carried out on a Linux operating system based on an Intel core 2 duo processor, 2.8 GHz with 1024 MB of RAM. We use the modeler and the linear programs solver XPress-Mosel version 1.6.1 (2005) [5].
The results of the four formulations are presented in Tables 1 and 2, where each row corresponds to one instance.

Legenda of the tables:

- n: number of integer variables
- *gap*: $|\frac{b-l}{b}| * 100$ where b is the value of the best known solution and l is the optimal value of the LP relaxation at the root node (in %).
- *nodes*: number of nodes visited by the branch-and-bound algorithm

Table 1: Resolution of $(IQKP_1)$ $(u_i = 50)$

	$(LP_{\texttt{BBL}})$			$(LP_{\texttt{BBLr}})$			$(LP_{\texttt{BIL}})$			$(LP_{\texttt{BILr}})$		
n	gap	nodes	time	gap	nodes	time	gap	nodes	time	gap	nodes	time
10	69	1462	51	35	549	88	44	787	9	9	169	15
10	37	478	22	21	423	49	19	449	5	2	13	2
10	59	2316	83	29	577	92	41	886	14	6	66	7
10	41	573	19	31	301	30	19	389	4	0.4	75	5
10	41	403	17	20	129	30	22	319	3	0.3	45	5
20	37	13030	3303	24	863	*(3%)	16	1740	82	0.04	15	22
20	44	10000	*(7%)	26	956	*(7%)	25	3339	169	0.07	5	119
20	55	10000	*(7%)	35	866	*(13%)	33	9322	545	7	95	75
20	45	8218	*(11%)	28	758	*(9%)	23	4355	317	0	1	0
20	38	6435	*(3%)	29	485	*(10%)	23	6323	318	0.6	146	114
30	47	2632	*(36%)	36	270	*(27%)	25	10000	*(8%)	4	148	441
30	79	3288	*(55%)	51	159	*(42%)	52	4813	*(30%)	24	1086	*(11%)
30	45	5833	*(23%)	26	293	*(19%)	22	13739	*(3%)	0.05	81	193
30	84	195	*(60%)	58	171	*(43%)	60	10000	*(40%)	28	1103	*(15%)
30	48	2933	*(6%)	33	175	*(29%)	27	10000	*(11%)	3	568	2088

$*(g\%)$ means that the branch-and-bound is stopped after 1 hour with a MIP gap of $g\%$

- *time*: CPU time (in seconds) required by the branch-and-bound algorithm.
  This time is limited to 1 hour of CPU time.

Table 2: Resolution of $(IQKP_2)$ $(u_i = 100)$

	$(LP_{\texttt{BBL}})$			$(LP_{\texttt{BBLr}})$			$(LP_{\texttt{BIL}})$			$(LP_{\texttt{BILr}})$		
n	gap	nodes	time	gap	nodes	time	gap	nodes	time	gap	nodes	time
10	38	503	25	31	143	29	17	423	7	0.1	12	3
10	63	667	44	34	168	52	45	299	7	7	69	5
10	33	362	26	19	206	41	14	162	3	0.1	26	6
10	44	531	24	14	313	50	22	364	4	0.1	87	7
10	37	201	12	5	75	2s	15	251	3	0.1	9	1
20	43	5226	*(18%)	21	900	3524	24	2877	256	0.04	12	28
20	52	4617	996	20	708	*(4%)	29	83	1574	0	1	0
20	63	2900	*(20%)	39	484	*(22%)	38	19528	1130	8	118	123
20	64	6347	*(26%)	38	541	*(22%)	42	16630	1039	11	228	152
20	74	6307	*(22%)	30	385	2848	49	7910	920	4	74	74
30	46	1651	*(37%)	29	48	*(29%)	24	8548	*(1%)	0.4	60	488
30	61	99	*(47%)	23	124	*(29%)	37	1591	*(21%)	3	271	3278
30	44	2219	*(37%)	31	48	*(29%)	22	2956	*(4%)	0.6	255	1828
30	53	414	*(62%)	34	61	*(32%)	31	4452	*(17%)	5	499	3080
30	71	2327	*(39%)	56	108	*(49%)	40	5676	*(9%)	17	824	*(4%)

$*(g\%)$ means that the branch-and-bound is stopped after 1 hour with a MIP gap of $g\%$

Program ($LP_{\texttt{BIL}}$) has less variables and constraints than program ($LP_{\texttt{BBL}}$). For example, instances of class $IQKP_1$ with $n = 20$ lead to a program ($LP_{\texttt{BIL}}$) (resp. ($LP_{\texttt{BBL}}$)) with 2820 (resp. 7260) variables and 5061 (resp. 14421) constraints. Moreover, we can observe in Tables 1 and 2 that, for all the instances, the gap associated to ($LP_{\texttt{BIL}}$) is much smaller than the gap associated to ($LP_{\texttt{BBL}}$). Consequently, the BIL approach outperforms the BBL approach with regard to the number of nodes and the computational time.

For BBL and BIL the reinforced versions significantly improve the gap value. Consequently, the number of nodes decreases in these reinforced versions. However, for BBL, the gap improvement is not sufficient to compensate the increase of the size and finally the CPU time required by BBLr is larger than the CPU time required by BBL.

For BIL, the reinforced version leads to an important improvement of the gap, but in this case the improvement of the gap widely compensate the increase of the size and finally the CPU time required by BILr is generally significantly smaller than the CPU time required by BIL.

We can also observe in Tables 1 and 2 that the gap values associated with BBLr and BIL are quite similar. However, the size of BIL being much lower than that of BBLr, BIL outperforms BBLr from the computational time point of view.

As a conclusion, on these two classes of instances, BILr is the best approach for the three criteria : gap, nodes and time. However, the computational experiments have shown that this method was unable to solve instances with 40 variables or more within 1 hour of CPU time.

5 Concluding remarks

In this paper, we have presented several linear reformulations of linearly constrained quadratic integer programs. The BBL and BBLr methods that consist in using the standard linearization of quadratic 0-1 programs is not usable because the binary decomposition combined to this linearization leads to 0-1 quadratic programs with too many variables and constraints.

Then, we presented a new approach, BIL, using the standard linearization of the product of an integer variable by a binary one. This method reduces significantly the number of constraints and variables added, in comparison with the BBL approach. In our experiments, surprisingly, this size reduction comes along with a smaller integrality gap. Therefore, BIL is a better approach. Moreover, the valid inequalities added in BILr provide an important improvement. A further improvement would be to incorporate these valid inequalities into a branch-and-cut framework.

References

[1] Fu, H. L., Shiue, C. L., Cheng, X., Du, D. Z., Kim, J. M.: Quadratic Integer Programming with Application in the Chaotic Mappings of Complete Multipartite Graphs. J. Optim. Theory Appl. 110 (3), 545–556 (2001)

[2] Fortet, R.: Applications de l'Algèbre de Boole en Recherche Opérationelle. Revue Française De Recherche Opérationelle. 4, 17–25 (1960)

[3] Sherali, H.D., Adams, W.P.: A tight linearization and an algorithm for zero-one quadratic programming problems. Management Science. 32(10), 1274–90 (1986)

[4] McCormick, Garth P.: Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems. Mathematical Programming. 1(10), 147–175 (1976)

[5] Dash Optimization, Xpress-Mosel version 1.6.1.: Xpress-Mosel language Reference Manual 1.4., http://www.dashoptimization.com/ (2005)

[6] Körner, F.: A New Bound for the Quadratic Knapsack Problem and Its Use in a Branch and Bound Algorithm. Optimization. 17, 643–648 (1986)

[7] Körner, F.: An efficient branch and bound algorithm to solve the quadratic integer programming problem. Computing. 30, 253–260 (1983)