4OR manuscript No.
(will be inserted by the editor)

Compact linearization for binary quadratic problems

Leo Liberti[1]

LIX, École Polytechnique, F-91128 Palaiseau, France

Received: 25th April 2006 / Revised version: ?

Abstract We show that a well-known linearization technique initially proposed for quadratic assignment problems can be generalized to a broader class of quadratic 0-1 mixed-integer problems subject to assignment constraints. The resulting linearized formulation is more compact and tighter than that obtained with a more usual linearization technique. We discuss the application of the compact linearization to three classes of problems in the literature, among which the graph partitioning problem.

Keywords: binary quadratic problem – linearization – graph partitioning.

1 Introduction

Binary Quadratic Problems (BQP)s are **NP**-hard problems which are also practically difficult to solve (although polynomially solvable instance classes exist, see e.g. (Allemand et al., 2001)). However, because of the expressive power of 0-1 products, BQPs have a wealth of applications (Padberg and Rijal, 1996; Hammer and Rudeanu, 1968). BQPs can be reformulated exactly to MILPs with a large number of variables and constraints (Fortet, 1960; Beasley, 1998). A complete treatment of linearizations for some classes of BQPs can be found in (Padberg and Rijal, 1996), where a very detailed polyhedral study of the problem polytope is performed.

Although the main factor which improves Branch-and-Bound (BB) solution times of Integer Programming (IP) problems is usually the strength of formulation (i.e. the bound tightness at each node), formulation size is also important, since at every BB node a continuous relaxation of the problem is solved with Linear Programming (LP) techniques, whose solution times mainly depend on problem size. With respect to an existing formulation

with n_1 variables and m_1 constraints, a formulation with n_2 variables and m_2 constraints is *compact* if n_2 is smaller than $O(n_1)$ and m_2 is smaller than $O(m_1)$. We focus here on constraint-side compactness.

In this paper, we present a compact linearization (with respect to the usual linearization (Fortet, 1960)) for BQPs subject to various subsets of assignment constraints, and we discuss the strength of its continuous relaxation. The main idea behind this linearization is to multiply subsets of linear constraints by subsets of problem variables and then to linearize the products à la Reformulation-Linearization Technique (RLT) (Sherali and Adams, 1986). The same idea was previously used to propose a compact linearization of the Quadratic Assignment Problem (QAP) (Frieze and Yadegar, 1983), and eventually discussed in a more formal RLT framework (Sherali and Brown, 1994). A related line of argument was pursued in (Sherali and Lee, 1996), focusing on the relaxation, rather than on the linearization properties of the reformulation. A similar linearization was applied to problems subject to knapsack (Caprara et al., 1999) and packing (Caprara and Lancia, 2002) constraints, but its validity depends on the objective being the maximization a non-negatively weighted sum of the products. The original contribution of the present paper is to show that the same technique can be applied to linearize QPBs with assignment constraints whose form is considerably more general than the QAP; and furthermore, no restriction is imposed on the objective function.

In Section 2 we discuss the usual linearization employed for BQPs. In Section 3 we discuss the compact linearization, show that it is equivalent to the usual linearization and that its continuous relaxation is tighter. In Section 4 we use the compact linearization to reformulate three well-known classes of problems; among these, a detailed computational study is performed on the Graph Partitioning problem (GPP).

Throughout the paper, the following points hold: (a) if P is an optimization problem, $v(P)$ is the optimal objective function value of P; (b) all products x_ix_j are assumed ordered such that $i \leq j$.

2 Usual linearization

Let $x \in \{0,1\}^n$ be the problem variables, E the set of ordered index pairs (i,j) (with $1 \leq i \leq j \leq n$) for which the product x_ix_j is in the problem formulation (with $|E| = m$), and $w \in \mathbb{R}^m_+$ a set of non-negative linearization variables where w_{ij} is used to replace the products x_ix_j for all $(i,j) \in E$.

Let $N = \{1,\ldots,n\}$ and $\{I_k \mid k \in K\}$ a covering of N, where K is an index set whose cardinality is supposed bounded by a polynomial in n (for if it were not, the original formulation would have more than polynomially many constraints in n, and thus would likely be solved with an altogether different approach). A more realistic bound would be $|K| \leq n$, given that

$|K|$ is the size of a covering of $\{1, \ldots, n\}$. We consider the following problem P:

$$\min_{x,w} c^{\mathsf{T}} x + b^{\mathsf{T}} w \tag{1}$$

$$g(x, w) \leq 0 \tag{2}$$

$$\forall k \in K \qquad \sum_{i \in I_k} x_i = 1 \tag{3}$$

$$\forall (i,j) \in E \qquad w_{ij} = x_i x_j \tag{4}$$

$$\forall i \in N \qquad x_i \in \{0, 1\} \tag{5}$$

$$\forall (i,j) \in E \qquad w_{ij} \geq 0 \tag{6}$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and g is a vector of functions of x, w.

The linearization below, which we call *usual linearization*, was originally proposed in (Fortet, 1960) and discussed in (Hammer and Rudeanu, 1968), Sect. 7, Thm. 4. We reformulate problem P exactly to a MILP by replacing constraints (4) with the following linearization constraints:

$$\forall (i,j) \in E \quad (w_{ij} \leq x_i) \tag{7}$$

$$\forall (i,j) \in E \quad (w_{ij} \leq x_j) \tag{8}$$

$$\forall (i,j) \in E \quad (w_{ij} \geq x_i + x_j - 1). \tag{9}$$

If either x_i or x_j is zero, then by (6) and (7) or (8) we have $w_{ij} = 0$. If $x_i = x_j = 1$ then by (7) and (9) $w_{ij} = 1$. Hence this is an exact reformulation, which we indicate with $L_1(P)$. This linearization implies the addition of $3m$ constraints to the formulation.

For practical purposes, it is a good idea to also introduce the constraints $w_{ii} = x_i$ for all $i \leq n$, which are always valid if w_{ii} linearizes the quadratic term x_i^2 where $x_i \in \{0, 1\}$. Furthermore, for the linearization to yield a linear problem, it is required that g be a vector of linear functions of x, w.

3 Compact linearization

Let J_k be index sets such that $I_k \subseteq J_k$ for all $k \in K$, and $F(K, I, J)$ be defined by

$$\{(i,j) \mid \exists\, (h,l) \in \bigcup_{k \in K} (I_k \times J_k)\ (i = \min(h,l) \wedge j = \max(h,l))\}. \tag{10}$$

For ease of notation, let $F = F(K, I, J)$. We assume that the following covering condition holds:

$$E \subseteq F, \tag{11}$$

so that all products in the problem are considered. We multiply (3) by x_j (for $j \in J_k$) to form the following system:

$$\forall k \in K, j \in J_k \quad (\sum_{i \in I_k} x_i x_j = x_j),$$

then substitute $w_{ij} = x_i x_j$ to get the following linear system:

$$\forall k \in K, j \in J_k \quad (\sum_{\substack{i \in I_k \\ (i,j) \in F}} w_{ij} = x_j). \tag{12}$$

In order to simplify notation in the rest of the section, we are going to assume that w_{ij} are defined for $i > j$ too, and that $w_{ij} = w_{ji}$ holds in such cases; (12) may then be written as:

$$\forall k \in K, j \in J_k \quad (\sum_{i \in I_k} w_{ij} = x_j). \tag{13}$$

We then form the linearized problem $L_2(P)$ by replacing constraints (7)-(9) in $L_1(P)$ with (12).

Theorem 1. *Provided condition (11) holds, $L_2(P)$ is an exact reformulation of $L_1(P)$.*

Proof. By (12) we immediately have

$$\forall (i,j) \in F \quad (w_{ij} \leq x_j). \tag{14}$$

Since $J_k \supseteq I_k$ for all $k \in K$, we also have

$$\forall (i,j) \in F \quad (w_{ij} \leq x_i). \tag{15}$$

(14)-(15) imply (7)-(8) by (11). Any sum of (15) preserves the inequality sense, so:

$$\forall k \in K, (i,j) \in F \quad (\sum_{f \in I_k \setminus \{i\}} w_{fj} \leq \sum_{f \in I_k \setminus \{i\}} x_f). \tag{16}$$

We add and subtract w_{ij} from LHS of (16):

$$\forall k \in K, (i,j) \in F \quad (\sum_{f \in I_k} w_{fj} - w_{ij} \leq \sum_{f \in I_k \setminus \{i\}} x_f). \tag{17}$$

We substitute $x_j = \sum_{f \in I_k} w_{fj}$ (which holds by (13)) in (17):

$$\forall k \in K, (i,j) \in F \quad (x_j - w_{ij} \leq \sum_{f \in I_k \setminus \{i\}} x_f). \tag{18}$$

We add and subtract x_i from from RHS of (18):

$$\forall k \in K, (i,j) \in F \quad (x_j - w_{ij} \leq \sum_{f \in I_k} x_f - x_i). \tag{19}$$

Finally, we substitute $1 = \sum_{f \in I_k} x_f$ (which holds by (3)) in (18):

$$\forall k \in K, (i,j) \in F \quad (x_j - w_{ij} \leq 1 - x_i), \tag{20}$$

which, by condition (11), establishes (9). □

3.1 Choice of J_k

Since the size of the formulation depends on $|J_k|$ (for $k \in K$), we should choose J_k as small as possible such that they satisfy the required conditions $J_k \supseteq I_k$ and $E \subseteq F$. More precisely, the problem to be solved is the following.

$$\left.\begin{array}{rl} \min \sum_{k \in K} |J_k| & \\ \forall\, k \in K & J_k \supseteq I_k \\ & E \subseteq F(K, I, J). \end{array}\right\} \tag{21}$$

This covering problem can be modelled as an integer programming problem. For all $j \leq n, k \in K$ let $y_{jk} = 1$ if $j \in J_k$ and 0 otherwise. Then (21) is formulated as follows:

$$\left.\begin{array}{rl} \min & \sum_{k \in K} \sum_{j \leq n} y_{jk} \\ \forall\, k \in K, i \in I_k & y_{ik} = 1 \\ \forall\, (i,j) \in E & \sum_{k: i \in I_k} y_{jk} + \sum_{k: j \in I_k} y_{ik} \geq 1 \\ \forall\, k \in K, i \in I_k & y_{ik} \in \{0,1\}. \end{array}\right\} \tag{22}$$

Although finding the smallest possible J_k's yields a more compact linearization, there may be a trade-off between compactness and relaxation tightness. In other words, finding the optimal J_k's may not yield the fastest BB solution run.

3.2 Compactness

The number of added constraints (12) is $R = \sum_{k \in K} |J_k|$. The simplest worst case analysis yields a constraint set size of $O(n|K|)$, which, due to the size of $|K|$ (polynomial in n), may be asymptotically worse than the $3m$ constraints added by the usual linearization. In practice, however, assignment constraints turn up in problems where the variables have several indices, and the assignment constraint sums range over a proper subset of these

indices. A reasonable average case assumption is therefore that problem variables are arranged in a d-dimensional array, and that the assignment constraints are sums ranging over e-dimensional "slices" where $e < d$ and indexed across the remaining $d - e$ dimensions. This gives rise to a set K of size $O(n^{(d-e)/d})$ and sets I_k of size $O(n^{e/d})$. In most practical cases, $|J_k|$ is $O(|I_k|^c)$ with $1 \leq c < 2$, so the number of added constraints (12), in this scenario, is the product of the two sizes: $O(n^{(d-e)/d} n^{ce/d}) = O(n^{1+(c-1)e/d})$. Since $c - 1 < 1$ and $e < d$, we obtain $O(n^{1+c'})$ with $c' = e(c-1)/d < 1$, which compares favourably with the usual linearization ($3m$ is $O(n^2)$). In the GPP problem (see Section 4.1) and the QAP (see Section 4.3), $d = 2$ and $e = 1$. In the scheduling problem of Section 4.2, we have $d = 3$ and $e = 2$. The constant c is more difficult to estimate, but a reasonable approximation drawn from dimensionality considerations is $c \approx 1 + e/d$. For Graph Partitioning, the decrease in formulation size can be observed in the LC columns of Table 1.

The compact linearization may well have a larger set of linearization variables with respect to the usual linearization. Whereas in the usual linearization one new variable w_{ij} is introduced for each product $x_i x_j$ present in the original problem, in the compact linearization we introduce variables w_{ij} for each $(i, j) \in F$ (see Eq. (13)). It may seem, therefore, that the compact linearization is convenient only for *dense* quadratic problems (i.e. problems where most of the products are present). Notice however that the linearization variables w_{ij} are continuous, rather than binary, variables; so the additional time taken by a BB algorithm is limited to having to deal with larger linear relaxation solutions (no branching ever occurs on the w variables). Furthermore, many of the linearization variables can be set to 0 and eliminated with some preprocessing, as follows: for all $(i, j) \in F$, suppose there is a k such that both i and j are in I_k. Then by (3) we have $x_i + x_j \leq 1$, which implies $w_{ij} = 0$.

3.3 Relaxation strength

Denote by $R_i(P)$ the continuous relaxation of the linearization $L_i(P)$ for $i \in \{1, 2\}$.

Corollary 1. *The continuous relaxation of the compact linearization is at least as tight as that of the usual linearization, i.e.* $v(R_2(P)) \geq v(R_1(P))$.

Proof. Notice that the proof of Thm. 1 never uses the fact that x are binary variables. Therefore, we can use the same argument to prove that a point in the feasible region of $R_2(P)$ is also in the feasible region of $R_1(P)$, which immediately implies the result. □

That there are indeed cases when $R_2(P)$ is strictly tighter than $R_1(P)$ should appear clear from the BB column of Table 1 reporting the comparative number of BB iterations to solve the listed instances with the two different linearizations, which is almost always strictly smaller in favour of the compact linearization.

4 Applications

In this section we shall present some applications of the compact linearization described in this paper. We perform a computational study on the GPP, we report briefly on a computational study (Davidović et al., 2004) on the Multiprocessor Scheduling Problem with Communication Delays (MSPCD), and we provide an alternative proof of the Frieze-Yadegar formulation (Frieze and Yadegar, 1983) for the QAP.

4.1 Graph partitioning

The GPP, also called MIN-k-CUT, is a well-known **NP**-hard problem that has many applications in several fields (Hendrickson and Kolda, 2000; Battiti and Bertossi, 1999). Many different formulations of varying sizes where the clusters have maximal given cardinality have been proposed and compared in (Boulle, 2004). A problem variant with node capacities has been studied in (Ferreira et al., 1996). Very recently, a new formulation for a variant of the GRAPH BISECTION problem has been proposed in (Billionnet et al., 2006) (downloadable from `http://cedric.cnam.fr/PUBLIS/RC1003.pdf`). The latter shows considerable promise as its continuous relaxation is a convex NLP.

Given an undirected graph $G = (V, H)$ and an integer $k \leq |V|$, the problem consists of finding a partition of k subsets (clusters) of V minimizing the number of edges $\{i, j\}$ where i, j belong to different clusters. To each vertex $i \in V$ and for each cluster $h \leq k$, we associate a binary variable x_{ih} which is 1 if vertex i is in cluster h and 0 otherwise. We formulate the problem as follows:

$$\left.\begin{array}{rl} \min\limits_{x} \frac{1}{2} \sum\limits_{h \neq l \leq k} \sum\limits_{\{i,j\} \in H} x_{ih} x_{jl} & \\ \forall i \in V & \sum\limits_{h=1}^{k} x_{ih} = 1 \\ \forall h \leq k & \sum\limits_{i \in V} x_{ih} \geq 1 \\ \forall i \in V, h \leq k & x_{ih} \in \{0, 1\}. \end{array}\right\} \tag{23}$$

The usual linearization of this BQP involves the addition of $k^2|H|$ continuous variables $0 \leq w_{ijhl} \leq 1$ to the formulation, as well as the introduction

of $3|H|\frac{k(k-1)}{2}$ linearization constraints of the form (7)-(9) (one for each quadratic product $w_{ijhl} = x_{ih}x_{jl}$, for all $h \neq l, \{i,j\} \in H$ — keeping in mind that $w_{ijhl} = w_{jilh}$). In order to find the compact linearization, we define $E = \{((i,h),(j,l)) \mid \{i,j\} \in H \wedge i \leq j \wedge h \leq l \leq k\}$, $K = V$,

$$\forall\, i \in V\ (I_i = \{(i,h) \mid h \leq k\}) \tag{24}$$

$$\forall\, i \in V\ (J_i = I_i \cup \{(j,l) \mid \{i,j\} \in H, l \leq k\}), \tag{25}$$

and $F(K,I,J)$ defined as in (10). Since $J_i \supseteq I_i$ for all i and $E \subseteq F(K,I,J)$, Thm. 1 applies. We obtain the following $k|H|$ linearization constraints:

$$\forall \{i,j\} \in H,\, l \leq k\ (\sum_{h=1}^{k} w_{ijhl} = x_{jl}),$$

which replace the usual linearization constraints.

The sets J_i defined in (25) do not have minimal possible size (see Sect. 3.1). We define binary variables $y^i_{jl} = 1$ if $(j,l) \in J_i$, and 0 otherwise. Formulation (22) becomes:

$$\left.\begin{array}{rl} \min \sum\limits_{i \in V} \sum\limits_{\substack{j \in V \\ l \leq k}} y^i_{jl} & \\ \forall\, (i,h) \in I_i & y^i_{ih} = 1 \\ \forall\, (i,j) \in E, h \neq l \leq k & y^j_{ih} + y^i_{jl} \geq 1. \end{array}\right\} \tag{26}$$

It is possible to show that the constraint matrix of (26) is totally unimodular, so that finding optimal J_i's has polynomial time complexity. However, according to some exploratory computational tests, the GPP happens to be a case where enforcing maximum formulation compactness often yields a slacker linear relaxation, with slightly worse solution times. We therefore retained J_i's as defined in (25) in all subsequent computational tests.

4.1.1 Comparison between usual and compact linearization Table 1 reports computational results obtained over a few problem instances as a comparison between compact and usual linearizations. The `mesh` instances are $\sqrt{|V|} \times \sqrt{|V|}$ square grid graphs; `hypercube` instances are $(\log_2 |V|)$-dimensional hypercubic graphs; `random-0.5` is a randomly generated graph where each edge had a 0.5 generation probability. These results were obtained by solving the above formulations with CPLEX 8.1 (ILOG, 2002) on a 2.66GHz Pentium IV CPU with 1GB RAM running Linux. $|V|$ is the number of nodes and $|H|$ the number of edges in the graph; k is the number of subsets of the partition. The next three columns (labelled c) denote the number of linearization constraints LC, the number of BB nodes and user time of CPU required to reach the optimal solution with the compact linearization. The last three columns (labelled u) indicate the corresponding quantities for the usual linearization. The optimal objective function values were verified to be the same for both linearizations.

Instance	$\|V\|$	$\|H\|$	k	**c**: LC	BB	CPU	**u**: LC	BB	CPU
`mesh`	9	12	2	24	13	**0.03s**	36	13	0.1s
			5	60	2711	**3.99s**	360	19666	34.22s
			8	96	284094	**783.48s**	1008	317645	1498.2s
`hypercube`	16	32	2	64	45	**0.13s**	96	63	0.38s
			3	96	778	**2.18s**	288	1570	4.68s
			5	160	44874	**125.79s**	960	400020	2882.39s
`random-0.5`	15	64	2	128	33	**0.23s**	192	87	1.03s
			3	192	320	**2.91s**	576	535	8.03s
			5	320	11897	**218.2s**	1920	38034	1496.5s

Table 1. The effect of compact linearization on the GRAPH PARTITIONING problem. CPU times printed in boldface mark best results.

4.1.2 Performance comparison In order to test the compact linearization in a more realistic setting and against better formulations than that obtained with the usual linearization, we solved the GPP with an added maximum cardinality constraint

$$\forall h \leq k \ \left(\sum_{i \in V} x_{ih} \leq \left\lfloor \frac{|V|}{k} + \left(1 - \frac{1}{k}\right) \right\rfloor \right)$$

(when $k = 2$, this is also referred to as the BALANCED GRAPH BISECTION problem). Several instances are publically available (Johnson et al., 1989) at `http://public.research.att.com/~dsj/instances.html`. As this test suite was put together to test heuristic, rather than exact, methods, only the smallest instances in the set could be solved to optimality. Consequently, the result tables only report on a subset of the whole test suite. As the competing formulation, we chose the best-performing formulation in (Boulle, 2004) (formulation B2 for $k = 2$ and B for $k > 2$). The results were obtained by CPLEX 10.0 on an AMD Athlon 64bit 1.8GHz with 1GB RAM running Linux.

Table 2 gives the results for $k = 2$. The columns describe, in order: instance name, number of nodes $|V|$, number of edges $|H|$, and: optimal objective function value f^*, number of BB nodes (BB), user time in seconds (CPU) for the compact linearization (labelled *c*) and for Boulle's B2 formulation (labelled *B2*). Table 3 gives the results for $k = 3$, and Table 4 for $k = 4$. In the latter case, the test set was restricted to include only those instances which were solved to optimality by at least one formulation. The performance difference is more definite with $k > 2$ partly because Boulle's formulation B has a comparatively bigger size than formulation B2 (by contrast, formulation B is valid for all k, whereas formulation B2 is only valid for $k = 2$).

It is easy to see from the results in Tables 2-4 that in general the compact linearization outperforms Boulle's formulations B2/B. This is due to formulation size as well as formulation strength: Boulle's formulations are

Instance	$\|V\|$	$\|H\|$	**c**: f^*	BB	CPU	**B2**: f^*	BB	CPU
example1	6	8	2	2	0.00	2	3	0.00
example2	4	6	4	4	0.00	4	5	0.00
Breg100.04	100	150	4	14	**0.24**	4	48	0.56
Breg100.08	100	150	8	249	**0.57**	8	249	0.96
Breg100.20	100	150	16	8957	9.13	16	3669	**4.99**
Breg500.12	500	750	12	795	**15.96**	12	1546	29.26
Breg500.16	500	750	16	4715	**59.27**	16	7917	126.48
Breg500.20	500	750	20	47118	**549.56**	20	64065	906.73
Cat.0352	352	351	1	50	1.56	1	23	**1.35**
Cat.0702	702	701	1	100	6.05	1	28	**4.46**
Cat.1052	1052	1051	1	150	8.96	1	40	**1.47**
G124.02	124	149	13	322	**0.56**	13	155	0.79
G124.04	124	318	63	122369	**442.35**	63	268034	787.80
G124.08*	124	620	183	3.3e6	31.18%	182	5.1e6	33.36%
G250.01	250	331	29	8125	**29.25**	29	792579	1203.84
G500.005†	500	625	49	1046477	**8657.04**	50	2.1e6	8802.07
Grid100.10	100	180	10	259	**0.84**	10	107	0.86
Grid.900	900	1740	30	110533	**8974.17**	30	247843	18210.57
RCat.134	134	133	1	12	**0.24**	1	4	0.41
RCat.554	554	553	1	24	3.61	1	2	**2.49**
W-grid100.20	100	200	20	2146	**4.42**	20	2044	4.45

Table 2. Comparative results for $k = 2$ for the Balanced Graph Bisection problem. CPU times printed in boldface mark best results. * The instance G124.08 was not solved to optimality: both runs were interrupted after 10h CPU time; the CPU time column contains the integrality gap at termination. † The instance G500.05 was solved to optimality only with the compact linearization; the integrality gap at termination for formulation B2 was 25.6%.

based on a set of constraints whose form is similar to (9). This means that they are in fact "big M"-type constraints, which are generally known to often yield a comparatively poor LP relaxation bounds.

4.2 Multi-processor Scheduling with Communication Delays

The MSPCD arises in parallel computing. It involves scheduling dependent tasks with communication delays, which are due to data transfer, onto a homogeneous, arbitrarily connected multiprocessor architecture such that the total completion time is minimum. The problem is complicated by the fact that communication delays between tasks also depend on what processors the tasks are being executed on; namely, we assume that the connections among the processors do not form a complete graph, so transferring data between non-adjacent processors takes a time proportional to the distance between them. The MSPCD can be formulated as follows.

Instance	$\|V\|$	$\|H\|$	**c**: f^*	BB	CPU	**B**: f^*	BB	CPU
example1	6	8	5	16	0.01	5	4	0.01
example2	4	6	4	4	0.00	4	7	0.00
Breg100.04	100	150	16	11784	**59.49**	16	128948	802.07
Breg100.08	100	150	19	41345	**199.94**	19	297377	1872.68
Breg100.20†	100	150	23	632113	**2642.26**	27	973348	33.68%
Breg500.0*	500	750	62	105404	**77.4%**	79	68413	90.68%
Breg500.12*	500	750	72	37934	**82.77%**	88	32179	89.39%
Breg500.16*	500	750	70	34642	**80.91%**	87	33683	88.85%
Breg500.20*	500	750	78	48926	**83.25%**	89	34687	89.13%
Cat.0352	352	351	5	621122	2717.55	5	63806	**2113.89**
Cat.0702*	702	701	6	551272	**44.72%**	6	80775	48.23%
Cat.1052‡	1052	1051	-	-	-	2	204	**98.00**
G124.02	124	149	18	1911	**9.64**	18	8720	54.02
G124.04*	124	318	91	308390	**38.38%**	97	341905	54.28%
G250.01*	250	331	42	293064	**16.53%**	49	220903	41.26%
G500.005*	500	625	76	56753	**57.88%**	84	60406	76.48%
Grid100.10	100	180	18	4826	**33.29**	18	7066	78.94
Grid.900*	900	1740	63	12152	**74.6%**	64	4834	84.89%
RCat.134	134	133	2	78	**0.82**	2	6	1.51
RCat.554	554	553	2	87	**6.01**	2	127	14.66
W-grid100.20†	100	200	34	556582	**4432.88**	36	755551	27.91%

Table 3. Comparative results for $k = 3$ for the GRAPH PARTITIONING problem. CPU times printed in boldface mark best results. * Neither formulation found the optimum within 2 hours of user CPU time; the "winner" is the formulation which found the best solution and the smallest integrality gap. † The compact linearization found the optimum whereas Boulle's formulation B did not. ‡ The compact linearization formulation for Cat.1052 was too large to fit in the memory allocated to AMPL (Fourer and Gay, 2002).

Instance	$\|V\|$	$\|H\|$	**c**: f^*	BB	CPU	**B**: f^*	BB	CPU
example1	6	8	5	20	0.02	5	20	0.02
example2	4	6	6	6	0.01	6	0	**0.00**
Breg100.08*	100	150	23	524764	**6123.18**	29	329819	42.17%
Cat.0702	702	701	3	4530	**145.00**	3	2859	586.59
G124.02	124	149	23	62674	**537.86**	23	336911	3910.27
Grid100.10	100	180	20	5841	**139.70**	20	15824	524.33
RCat.134	134	133	3	248	**2.30**	3	26	2.63
RCat.554	554	553	3	417	**18.25**	3	104	24.24

Table 4. Comparative results for $k = 4$ for the GRAPH PARTITIONING problem. CPU times printed in boldface mark best results. * This instance was solved to optimality only with the compact linearization; the CPU time column for Boulle's formulation B contains the integrality gap after 2 hours of user CPU time.

$$
\left.
\begin{array}{rl}
\min\limits_{y,t} & \max\limits_{j\le n}\{t_j + L_j\} \\
\forall j \le n & \sum\limits_{k=1}^{p}\sum\limits_{s=1}^{n} y_{jk}^{s} = 1 \\
\forall k \le p & \sum\limits_{j=1}^{n} y_{jk}^{1} \le 1 \\
\forall k \le p, s \ge 2 & \sum\limits_{j=1}^{n} y_{jk}^{s} - \sum\limits_{j=1}^{n} y_{jk}^{s-1} \le 0 \\
\forall j \le n, i \in \delta^-(j) & t_i + L_i + \sum\limits_{k=1}^{p}\sum\limits_{s=1}^{n}\sum\limits_{l=1}^{p}\sum\limits_{r=1}^{n} \gamma_{ij}^{kl} y_{ik}^{s} y_{jl}^{r} \le t_j \\
\forall i,j \le n, k \le p, s \le n-1 & t_i + L_i - \alpha\left[2 - \left(y_{ik}^{s} + \sum\limits_{r=s+1}^{n} y_{jk}^{r}\right)\right] \le t_j \\
\forall j \le n & T^L - L_j \le t_j \\
\forall j, s \le n, k \le p & y_{jk}^{s} \in \{0,1\}\ ,\ t_j \ge 0.
\end{array}
\right\} \quad (27)
$$

Here, p is the number of processors, n is the number of tasks, $\delta^-(j)$ is the set of immediate predecessors of task j, L_j is the length of task j, γ_{ij}^{kl} is the communication delay between tasks i and j when they are executed on processors k and l, α is a sufficiently large penalty coefficient, y_{jk}^s is a binary variable set to 1 if task j is the s-th process to be executed on processor k and t_j is a continuous variable indicating the starting time of process j. T^L is a lower bound on the total completion time calculatd either with Load Balancing or with a Critical Path Method (CPM) applied to the task precedence graph where the arcs are weighted by the running times L_j. The usual linearization of this formulation is obtained by replacing each product $y_{jl}^r y_{ik}^s$ by a variable w_{ijkl}^{sr} and adding constraints (7)-(9) to the model.

The sets $\delta^-(j)$ define a digraph topology on the tasks given by $G = (V, A)$, where $(i, j) \in A$ if task i has to be completed before task j begins. The problem variables are indexed by triplets (i, s, k), where $i, s \leq n$ and $k \leq p$. We define E to be the set of pairs of triplets $\{(i, s, k), (j, r, l)\}$ such that there is an arc (i, j) in A. We define $K = \{1, \ldots n\}, , J_0 = \{(j, l, r) \mid l \leq p \wedge j, r \leq n\}$,

$$\forall\, k \in K\ (I_k = \{(k, h, s) \mid h \leq p \wedge s \leq n\})$$
$$\forall\, k \in K\ (J_k = J_0),$$

$F(K, I, J)$ as in (10), and note that $J_k \supseteq I_k$ for all k and condition (11) is satisfied. Thus, by Thm. 1, on multiplying the assignment constraints $\forall j \leq n(\sum_{k=1}^{p} \sum_{s=1}^{n} y_{jk}^s = 1)$ by all problem variables y_{il}^r, we obtain a set of $n^3 p$ constraints

$$\forall i, j, r \leq n, l \leq p\ (\sum_{k=1}^{p} \sum_{s=1}^{n} w_{ijlk}^{rs} = y_{il}^r)$$

which reformulate the $3n^4p^2$ constraints of the usual linearization. The comparative computational results, discussed in (Davidović et al., 2004) (downloadable from `http://www.gerad.ca/fichiers/cahiers/G-2004-99.pdf`), indicate user CPU time improvements of one to two orders of magnitude.

4.3 Quadratic Assignment problem

In this section we show that the application of the compact linearization to the QAP Koopmans-Beckmann formulation (Koopmans and Beckmann, 1957) yields the Frieze-Yadegar formulation (Frieze and Yadegar, 1983). This provides an alternative (and in our opinion, simpler) proof of validity of the Frieze-Yadegar formulation.

The QAP can be formulated as follows:

$$\left.\begin{array}{rl} \min\limits_{x} \sum\limits_{i,j,k,l}^{n} a_{ij}b_{kl}x_{ik}x_{jl} + \sum\limits_{i,j}^{n} c_{ij}x_{ij} & \\ \forall i \leq n & \sum\limits_{j=1}^{n} x_{ij} = 1 \\ \forall j \leq n & \sum\limits_{i=1}^{n} x_{ij} = 1 \\ \forall i,j \leq n & x_{ij} \in \{0,1\}, \end{array}\right\} \tag{28}$$

where (x_{ij}) is an $n \times n$ array of binary variables and $A = (a_{ij}), B = (b_{ij}), C = (c_{ij})$ are given $n \times n$ matrices. This is known as the Koopmans-Beckmann formulation (Koopmans and Beckmann, 1957). We multiply both sets of assignment constraints by each problem variable x_{kl} $(k, l \leq n)$. We then linearize the resulting products by requiring $\forall i, j, k, l \leq n$ $(w_{ijkl} = x_{ij}x_{kl})$. We define $E = \{((i,j),(k,l)) \mid i \leq k \wedge j \leq l\}$, $K = \{1, \ldots, n\}$, $I_k = \{(k,j) \mid j \leq n\}$ for all $k \in K$, $J_0 = \{(i,j) \mid i, j \leq n\}$, $J_k = J_0$ for all $k \in K$ and $F(K, I, J)$ as in (10). We note that $J_k \supseteq I_k$ and $E \subseteq F(K, I, J)$, and obtain the following linearization constraints:

$$\forall i,k,l \leq n \; (\sum_{j=1}^{n} w_{ijkl} = x_{kl})$$
$$\forall j,k,l \leq n \; (\sum_{i=1}^{n} w_{ijkl} = x_{kl}).$$

By Thm. 1, we can use the above constraints to replace the usual linearization constraints. The resulting linearization is the Frieze-Yadegar formulation (Frieze and Yadegar, 1983).

5 Conclusion

In this paper we derive a compact linearization for 0-1 mathematical programming problems involving products of binary variables subject to assignment constraints. Our result provides a substantial improvement over the usual linearization of 0-1 products, both in terms of size and relaxation tightness. We show the application of the compact linearization to: (a) the graph partitioning problem, where we provide a computational study showing that the compact linearization yields a formulation which is superior to other formulations in the literature; (b) the multiprocessor scheduling problem with communication delays, where we refer to a computational study that has already been made available as a technical report (Davidović et al., 2004); (c) the quadratic assignment problem, where we derive a new proof of the Frieze-Yadegar formulation.

Acknowledgments

I am very grateful to two anonymous referees for their extensive comments which led to substantial improvements.

References

K. Allemand, K. Fukuda, T.M. Liebling, and E. Steiner. A polynomial case of unconstrained zero-one quadratic optimization. *Mathematical Programming*, 91:49–52, 2001.

R. Battiti and A. Bertossi. Greedy, prohibition and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4):361–385, 1999.

J.E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. *Technical Report, Management School, Imperial College London*, 1998.

A. Billionnet, S. Elloumi, and M.-C. Plateau. Quadratic convex reformulation: a computational study of the graph bisection problem. Technical Report RC1003, Conservatoire National des Arts et Métiers, 2006.

M. Boulle. Compact mathematical formulation for graph partitioning. *Optimization and Engineering*, 5:315–333, 2004.

A. Caprara and G. Lancia. Structural alignment of large-size proteins via Lagrangian relaxation. In *Proceedings of 6th RECOMB*, pages 100–108. ACM Press, 2002.

A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.

T. Davidović, L. Liberti, N. Maculan, and N. Mladenović. Mathematical programming-based approach to scheduling of communicating tasks. Technical Report G-2004-99, Cahiers du GERAD, 2004.

C.E. Ferreira, A. Martin, C. Carvalho de Souza, R. Weismantel, and L.A. Wolsey. Formulations and valid inequalities for the node capacitated graph partitioning problem. *Mathematical Programming*, 74:247–266, 1996.

R. Fortet. Applications de l'algèbre de boole en recherche opérationelle. *Revue Française de Recherche Opérationelle*, 4:17–26, 1960.

R. Fourer and D. Gay. *The AMPL Book*. Duxbury Press, 2002.

A.M. Frieze and J. Yadegar. On the quadratic assignment problem. *Discrete Applied Mathematics*, 5:89–98, 1983.

P.L. Hammer and S. Rudeanu. *Boolean Methods in Operations Research and Related Areas*. Springer, Berlin, 1968.

B. Hendrickson and T. Kolda. Partitioning rectangular and structurally nonsymmetric sparse matrices for parallel processing. *SIAM Journal on Scientific Computing*, 21 (6):2048–2072, 2000.

ILOG. *ILOG CPLEX 8.0 User's Manual*. ILOG S.A., Gentilly, France, 2002.

D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning. *Operations Research*, 37:865–892, 1989.

T.C. Koopmans and M.J. Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957.

M.W. Padberg and M.P. Rijal. *Location, Scheduling, Design and Integer Programming*. Kluwer, Dordrecht, 1996.

H. Sherali and E. Brown. A quadratic partial assignment and packing model and algorithm for the airline gate assignment problem. In P. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and related problems*, pages 343–364. American Mathematical Society, 1994.

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.

H.D. Sherali and Y. Lee. Tighter representations for set partitioning problems. *Discrete Applied Mathematics*, 68:153–167, 1996.