An integer linear programming approach for a class of bilinear integer programs

Wuhua Hu[a,*], Wee Peng Tay[a]

[a] *School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore*

Abstract

We propose an Integer Linear Programming (ILP) approach for solving integer programs with bilinear objectives and linear constraints. Our approach is based on finding upper and lower bounds for the integer ensembles in the bilinear objective function, and using the bounds to obtain a tight ILP reformulation of the original problem, which can then be solved efficiently. Numerical experiments suggest that the proposed approach outperforms a latest iterative ILP approach, with notable reductions in the average solution time.

Keywords: Bilinear integer programming, integer linear programming, integer quadratic programming, binary binary linearization, binary integer linearization, bipartite graph

1. Introduction

We consider the following bilinear integer program (BIP),

$$\text{(P0)} \qquad \max_{\boldsymbol{x},\boldsymbol{y}} \ \left(\boldsymbol{\alpha}^T\boldsymbol{x}\right)\cdot\left(\boldsymbol{\beta}^T\boldsymbol{y}\right) \tag{1}$$
$$\text{subject to } \boldsymbol{a}_k^T\boldsymbol{x}+\boldsymbol{b}_k^T\boldsymbol{y}\le d_k,\ \forall\ k=1,\,2,\,\ldots,\,K,$$
$$\boldsymbol{x}\in\mathbb{N}^N,\ \boldsymbol{y}\in\mathbb{N}^M$$

where N, M, and K are positive integers, $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_N]^T$ and $\boldsymbol{x}=[x_1,x_2,\ldots,x_N]^T$ are $N\times 1$ vectors of non-negative integers, $\boldsymbol{\beta}=[\beta_1,\beta_2,\ldots,\beta_M]^T$ and $\boldsymbol{y}=[y_1,y_2,\ldots,y_M]^T$ are $M\times 1$ vectors of non-negative integers, and $\boldsymbol{a}_k$ and $\boldsymbol{b}_k$ are $N\times 1$ and $M\times 1$ real vectors respectively. We

*Corresponding author
Email addresses: hwh@ntu.edu.sg (Wuhua Hu), wptay@ntu.edu.sg (Wee Peng Tay)

also use the superscript ^{T} to denote vector transpose, and $\mathbb{N}$ the set of non-negative integers. This integer programming problem arises in several contexts, including bipartite graph matching and optimizing product bundle compositions [1].

By grouping the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ into a single vector, the problem (P0) can be viewed as an Integer Quadratic Programming (IQP) problem, which is in general *non-convex*. IQP problems can be solved using a series of convex or linear relaxations with valid inequalities [2, 3, 4]. Such IQP methods however do not exploit the special ensemble product structure of the objective function in (P0), and can lead to inefficient solvers for (P0).

Alternatively, if all integer variables are bounded, one can adopt the Binary Binary Linearization (BBL) approach [5, 6], in which every integer variable in $\boldsymbol{x}$ and $\boldsymbol{y}$ is replaced with its binary decomposition, and then linearizing the resulting binary products in the objective function via McCormick relaxations [7]. The resulting optimization program is a binary linear program, which can then be solved using standard Integer Linear Programming (ILP) methods. However, since the feasible region for $\boldsymbol{x}$ and $\boldsymbol{y}$ can be very large, this approach suffers from the curse of dimensionality problem.

Another way of solving (P0) is the Binary Integer Linearization (BIL) method [6, 8, 9]. By substituting $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]^T$ and $\boldsymbol{y} = [y_1, y_2, \ldots, y_M]^T$ into (1) and expanding, we see that the objective function of (P0) consists of bilinear terms of the form $c_{ij}x_iy_j$, where c_{ij} is a real number. In the BIL approach, an integer of each bilinear term $c_{ij}x_iy_j$, say x_i, is replaced with its binary decomposition while keeping the other integer unchanged [1]. This results in less binary variables and linearization constraints, which will hopefully lead to a more efficient solver for (P0). However, in this case, there are still $\Theta(M\sum_{i=1}^{N}\log_2 k_i)$ variables and constraints, where k_i is an upper bound for x_i for each i. This limits the BIL approach to problem sizes that are relatively small, and researchers are thus motivated to find a more efficient approach for solving (P0).

In the recent paper [1], the authors exploit a structural property of the objective function in (P0), and proposed novel *iterative* ILP approaches called LIN and LIN+ for solving (P0). In LIN, at each iteration an ILP with objective function $\boldsymbol{\alpha}^T\boldsymbol{x} + \boldsymbol{\beta}^T\boldsymbol{y}$ is solved together with appropriate lower bound constraints, to obtain a common lower bound for the integer ensembles $\boldsymbol{\alpha}^T\boldsymbol{x}$ and $\boldsymbol{\beta}^T\boldsymbol{y}$.

This new lower bound is then used as constraints in the next iteration. The algorithm converges if the ILP becomes infeasible, leading to an optimal solution of (P0), if there is any. LIN+ is an enhanced version of LIN, which under certain conditions, ensures that the objective function value improves at each iteration step. Both approaches are in contrast to the usual method of finding individual bounds for each integer variable, as is normally done in the IQP, BBL and BIL literature [2, 3, 4, 6]. Numerical experiments conducted by [1] showed that their iterative ILP approaches solve (P0) much more efficiently than the BIL approach, but the computation time depends on the number of iterations required before the algorithm converges. There are however no guarantees on the number of iterations required, which can be an issue in practical applications.

Motivated by the iterative ILP approach of [1] and the BIL method, we propose a hybrid approach that solves the problem (P0) even more efficiently, and in a fixed number of iterations. This is achieved by deriving a new upper bound for the ensembles $\boldsymbol{\alpha}^T\boldsymbol{x}$ and $\boldsymbol{\beta}^T\boldsymbol{y}$, in addition to the lower bound used in LIN and LIN+. The new upper bound leads to a tight binary representation of the ensemble $\boldsymbol{\alpha}^T\boldsymbol{x}$ or $\boldsymbol{\beta}^T\boldsymbol{y}$, which then allows us to apply the BIL approach to an *ensemble*, instead of to individual integer variables in $\boldsymbol{x}$ or $\boldsymbol{y}$. This significantly reduces the number of binary variables and linearization constraints in the reformulation, leading to a highly efficient solution for (P0). We present numerical experiments, which suggest that our approach is superior to LIN and LIN+, and consequently also the BIL method.

The rest of this letter is organized as follows. In Section 2, we present our new ILP approach for solving (P0), and review its connections to LIN and LIN+. In Section 3, we present numerical results based on a nonlinear bipartite matching problem to verify and compare the performances of our proposed method with those of LIN and LIN+. Finally, we conclude in Section 4.

2. A new ILP solution approach

In this section, we first present an elementary result that provides an upper bound to the *optimal* ensembles $\boldsymbol{\alpha}^T\boldsymbol{x}$ and $\boldsymbol{\beta}^T\boldsymbol{y}$ in (P0), which then allows us to propose enhanced versions of LIN and LIN+. These procedures are still iterative in nature, with minimal improvement (if any) in computation efficiency compared to LIN and LIN+. Therefore, we further propose a novel solution

approach based on our upper bound and the BIL method, and show how to reformulate (P0) into an ILP that can be solved efficiently.

The following result plays a central role in our proposed methods. This result is similar to Lemma 1 of [1], which only provides the lower bound for p_1^o and p_2^o.

Lemma 1. *Suppose that* $p_1, p_2, p_1^o, p_2^o > 0$ *and* $p_1 \le p_2$. *If* $p_1 + p_2 \ge p_1^o + p_2^o$ *and* $p_1 p_2 \le p_1^o p_2^o$, *then*

$$p_1 \le p_1^o \le p_2, \text{ and } p_1 \le p_2^o \le p_2. \tag{2}$$

Furthermore, if $p_1 p_2 < p_1^o p_2^o$, *then all the inequalities in* (2) *are strict.*

Proof. We have $p_1(p_1 + p_2) - p_1^2 = p_1 p_2 \le p_1^o p_2^o = p_1^o(p_1^o + p_2^o) - (p_1^o)^2 \le p_1^o(p_1 + p_2) - (p_1^o)^2$, and $(p_1^o - p_1)(p_1^o - p_2) \le 0$. Since $p_1 \le p_2$, we obtain $p_1 \le p_1^o \le p_2$. Similarly, with $p_2(p_1 + p_2) - p_2^2 = p_1 p_2 \le p_1^o p_2^o = p_2^o(p_1^o + p_2^o) - (p_2^o)^2 \le p_2^o(p_1 + p_2) - (p_2^o)^2$, it follows that $(p_2^o - p_1)(p_2^o - p_2) \le 0$ and hence $p_1 \le p_2^o \le p_2$. If $p_1 p_2 < p_1^o p_2^o$, then a similar argument as above shows that the inequalities in (2) are strict. The proof is now complete. □

A useful implication follows from the above lemma.

Corollary 1. *Suppose that the positive pairs* (p_1, p_2) *and* (p_1^o, p_2^o), *satisfying* $p_1 p_2 \ne p_1^o p_2^o$, *are optimal solutions for maximizing* $q_1 + q_2$ *and* $q_1 q_2$ *subject to the same constraints, respectively. Then, we have* $\underline{p} < p_1^o < \overline{p}$ *and* $\underline{p} < p_2^o < \overline{p}$, *where* $\underline{p} = \min\{p_1, p_2\}$ *and* $\overline{p} = \max\{p_1, p_2\}$.

Proof. Since we have $p_1 p_2 < p_1^o p_2^o$ and $p_1 + p_2 \ge p_1^o + p_2^o$, the result follows directly from Lemma 1. □

Suppose that $(\boldsymbol{x}^o, \boldsymbol{y}^o)$ is an optimal solution of (P0), and assume for the moment that the optimal objective value is non-zero. By interpreting $\boldsymbol{\alpha}^T \boldsymbol{x}^o$ as p_1^o and $\boldsymbol{\beta}^T \boldsymbol{y}^o$ as p_2^o in Corollary 1, we can try to obtain the optimal solution of (P0) by solving a series of ILPs iteratively, where at iteration $m \ge 0$, we solve the ILP:

$$\begin{aligned}
\text{(P1)} \qquad & \max_{\boldsymbol{x},\, \boldsymbol{y}} \ \boldsymbol{\alpha}^T \boldsymbol{x} + \boldsymbol{\beta}^T \boldsymbol{y} \\
& \text{subject to } \boldsymbol{a}_k^T \boldsymbol{x} + \boldsymbol{b}_k^T \boldsymbol{y} \le d_k, \ \forall\ k = 1, 2, ..., K, \\
& \qquad \underline{p}_m \le \boldsymbol{\alpha}^T \boldsymbol{x} \le \overline{p}_m, \qquad (3) \\
& \qquad \underline{p}_m \le \boldsymbol{\beta}^T \boldsymbol{y} \le \overline{p}_m, \qquad (4) \\
& \qquad \boldsymbol{x} \in \mathbb{N}^N, \ \boldsymbol{y} \in \mathbb{N}^M.
\end{aligned}$$

In (3) and (4) of the above program (P1), the bound $\underline{p}_m$ is set as 1 when $m = 0$, and updated as $\underline{p}_m = \min\{\boldsymbol{\alpha}^T \boldsymbol{x}^*, \boldsymbol{\beta}^T \boldsymbol{y}^*\} + 1$ at the mth iteration for $m \ge 1$, where $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is an optimal solution of

(P1) at the $(m-1)$-th iteration. Similarly, the bound $\overline{p}_m$ is set as ∞ when $m = 0$, and updated as $\overline{p}_m = \max\{\boldsymbol{\alpha}^T\boldsymbol{x}^*, \boldsymbol{\beta}^T\boldsymbol{y}^*\} - 1$ at the mth iteration for $m \geq 1$. We call this the eLIN algorithm. The only difference between eLIN and LIN of [1] lies in (3) and (4), where in LIN, only the lower bounds are used at each iteration. The iterations are repeated until (P1) becomes infeasible, whereupon we say that the algorithm has converged. If (P1) turns out to be unbounded when $m = 0$, then the original problem (P0) is also unbounded; and if (P1) is infeasible when $m = 0$, it is possible that (P0) has an optimal solution with objective value zero, in which case it can be easily verified by looking for a solution with either $x_i = 0$ for all i such that $\alpha_i > 0$, or $y_j = 0$ for all j such that $\beta_j > 0$ [1]. Otherwise, the best solution of all previous iterations gives the optimal solution of (P0), as shown in the following lemma.

Lemma 2. *Suppose eLIN is feasible at the first iteration. Then, the solution found by eLIN that has the maximal objective value (P0) amongst all feasible iterations is an optimal solution of (P0).*

Proof. We show this by contradiction. We note that eLIN always converges since the constraints (3)-(4) are tightened at each iteration. Assume that the optimal solution of (P0) is not amongst the sequence of feasible solutions found by eLIN. Then by Corollary 1, eLIN must be feasible at the last iteration, which contradicts the fact that eLIN always terminates at an infeasible iteration. The lemma is now proved. □

Numerical examples can be found to show that the last feasible solution returned by eLIN before it converges is not necessarily the optimal solution of (P0). This means that after reaching the optimal solution for (P0), the lower and upper bounds returned by eLIN are no longer meaningful bounds for the optimal solution of (P0), and eLIN wastes computation time searching for non-optimal solutions until the auxiliary problem (P1) becomes infeasible. The same remark also applies to LIN.

The above insight implies that at each iteration, eLIN may not improve the objective value of (P0). Suppose that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is the optimal solution of (P1) at a particular iteration. To ensure that the objective value of (P0) increases in the next iteration, an *objective improvement constraint* (OIC),

$$\left(\boldsymbol{\alpha}^T\boldsymbol{x}\right) \cdot \left(\boldsymbol{\beta}^T\boldsymbol{y}\right) \geq J_{best} \triangleq \left(\boldsymbol{\alpha}^T\boldsymbol{x}^*\right) \cdot \left(\boldsymbol{\beta}^T\boldsymbol{y}^*\right) + 1. \tag{5}$$

can be imposed. Since the constraint (5) is bilinear, one can try to approximate the bilinear OIC (5) by its convex and concave envelope [7], which however does not guarantee the objective value

improves in each iteration. The reference [1] thus proposes an alternative set of linear OICs at the mth iteration, given by

$$\begin{aligned}
\boldsymbol{\alpha}^T \boldsymbol{x} &\geq \sum_{t=0}^{L} s_t(\underline{p}_m + t), \\
\boldsymbol{\beta}^T \boldsymbol{y} &\geq \sum_{t=0}^{L} s_t(\underline{p}_m + t), \\
\boldsymbol{\alpha}^T \boldsymbol{x} + \boldsymbol{\beta}^T \boldsymbol{y} &\geq \sum_{t=0}^{L-1} s_t \left(\underline{p}_m + t + \left\lceil \frac{J_{best}}{\underline{p}_m + t} \right\rceil \right), \\
\sum_{t=0}^{L} s_t &= 1, \\
s_t &\in \{0,\, 1\},\ \forall t = 0,\, 1, \ldots, L,
\end{aligned} \tag{6}$$

where $\lceil \cdot \rceil$ is the smallest integer larger than its argument. The above constraints ensure that if $s_t = 1$ for some $t < L$, then the objective function value of (P0) will be at least J_{best}, an improvement over the best solution found so far; otherwise, $s_T = 1$ and there is *no* guarantee that the objective value improves at the current iteration. This algorithm is called LIN+. We refer the reader to [1] for details. If L is chosen to be sufficiently large, improvements are guaranteed at every iteration. However, a large L results in more binary variables $s_t, t = 0, 1, \ldots, L$, which may lead to computational inefficiency. On the other hand, if L is chosen too small, the objective function value of (P0) may not improve over each iteration, and the solution time will again be prolonged. To overcome this problem, we can appeal to Corollary 1 to choose the value of L adaptively at each iteration. It is clear that L is not bigger than the gap $\overline{p}_m - \underline{p}_m$. Therefore, by setting $L = \overline{p}_m - \underline{p}_m$, we can ensure that the OICs (6) guarantee improvement in the objective value (P0) at every iteration. We call this the eLIN+ algorithm. Summarizing the above discussion, we have the following Lemma 3.

Lemma 3. *In eLIN+, the objective value of (P0) improves at every iteration before convergence, while in LIN+ this happens only if the parameter L is sufficiently large.*

Lemma 3 implies that if eLIN+ is feasible for the initial iteration, then when it converges, the optimal solution of (P0) is given by the last feasible solution found by eLIN+, i.e., eLIN+ avoids redundant iterations, in sharp contrast to LIN, eLIN, and LIN+.

We have conducted extensive numerical experiments to compare the performances of LIN, LIN+, eLIN and eLIN+ in Section 3. Our results suggest that eLIN and eLIN+ perform only slightly better than LIN and LIN+ for small problem sizes, and worse when the problem size is large. Therefore, in the following, we propose a new reformulation of (P0) based on Corollary 1 and the BIL method, which leads to a significantly more efficient solution of (P0) compared to LIN and LIN+.

2.1. Hybrid LIN/eLIN+ and BIL: LINB

In this subsection, we describe a novel reformulation of (P0) making use of Corollary 1. Inspired by the BIL approach, we replace one ensemble, $\boldsymbol{\alpha}^T\boldsymbol{x}$ or $\boldsymbol{\beta}^T\boldsymbol{y}$, of the bilinear objective of (P0) with its binary decomposition by first estimating an upper bound for the ensembles using either LIN or eLIN+. We call this the LINB algorithm. The steps are as follows:

(a) Compute lower and upper bounds of $\boldsymbol{\alpha}^T\boldsymbol{x}$ and $\boldsymbol{\beta}^T\boldsymbol{y}$ by running LIN for a *single* iteration or eLIN+ for a fixed number m of iterations to obtain $\underline{q}_m = \min\{\boldsymbol{\alpha}^T\boldsymbol{x}^*, \boldsymbol{\beta}^T\boldsymbol{y}^*\}$ and $\overline{q}_m = \max\{\boldsymbol{\alpha}^T\boldsymbol{x}^*, \boldsymbol{\beta}^T\boldsymbol{y}^*\}$, where $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is the optimal solution of (P1) obtained by LIN at the first iteration or eLIN+ at the $(m-1)$-th iteration. If (P1) becomes infeasible during the iterations, the algorithm is stopped, and the optimal solution of (P0) is returned as per LIN or eLIN+.

(b) Let $L_1 = \lfloor \log_2 \overline{q}_m \rfloor + 1$ and $L_2 = \lfloor \log_2 \underline{q}_m \rfloor + 1$, which are the lengths of the binary decompositions of $\overline{q}_m$ and $\underline{q}_m$, respectively. We represent one of the ensembles, say, $\boldsymbol{\alpha}^T\boldsymbol{x}$, by its binary decomposition $\sum_{i=1}^{L_1} 2^{i-1} r_i$, where r_i are binary variables. Consequently the objective of (P0) can be rewritten as $\sum_{i=1}^{L_1} 2^{i-1} z_i$, with $z_i = r_i \boldsymbol{\beta}^T\boldsymbol{y}$ for $i = 1, 2, \ldots, L_1$. We then linearize these bilinear equality constraints via McCormick relaxations [7] (which are exact in this case), yielding an ILP reformulation (P2) of (P0) in the following, which can be solved using standard

ILP techniques.

$$\text{(P2)} \qquad \max_{\boldsymbol{x},\, \boldsymbol{y},\, \{z_i,\, r_i\}_{i=1}^{L_1}} \ \sum_{i=1}^{L_1} 2^{i-1} z_i$$

$$\text{subject to } \boldsymbol{a}_k^T \boldsymbol{x} + \boldsymbol{b}_k^T \boldsymbol{y} \leq d_k, \ \forall\ k = 1,\, 2,\, \ldots,\, K,$$

$$\boldsymbol{\alpha}^T \boldsymbol{x} = \sum_{i=1}^{L_1} 2^{i-1} r_i, \tag{7}$$

$$z_i \geq \underline{q}_m r_i, \ \forall i = 1,\, 2,\, \ldots,\, L_1, \tag{8}$$

$$z_i \geq \boldsymbol{\beta}^T \boldsymbol{y} + \overline{q}_m (r_i - 1), \ \forall i = 1,\, 2,\, \ldots,\, L_1, \tag{9}$$

$$z_i \leq \overline{q}_m r_i, \ \forall i = 1,\, 2,\, \ldots,\, L_1, \tag{10}$$

$$z_i \leq \boldsymbol{\beta}^T \boldsymbol{y} + \underline{q}_m (r_i - 1), \ \forall i = 1,\, 2,\, \ldots,\, L_1, \tag{11}$$

$$\underline{q}_m \leq \boldsymbol{\alpha}^T \boldsymbol{x} \leq \overline{q}_m, \tag{12}$$

$$\underline{q}_m \leq \boldsymbol{\beta}^T \boldsymbol{y} \leq \overline{q}_m, \tag{13}$$

$$\sum_{i=L_2}^{L_1} r_i \geq 1, \tag{14}$$

$$\boldsymbol{x} \in \mathbb{N}^N,\, \boldsymbol{y} \in \mathbb{N}^M,\, z_i \in \mathbb{N},\, r_i \in \{0,\, 1\},\, \forall\ i = 1,\, 2,\, \ldots,\, L_1.$$

In step (a), we either run LIN for one iteration (cf. Lemma 2) or eLIN+ for a fixed number of iterations (cf. Lemma 3) to ensure that the lower and the upper bounds obtained are valid for the optimal solution of (P0). In the formulation (P2) of step (b), the upper bound $\overline{q}_m$ on the ensemble $\boldsymbol{\alpha}^T \boldsymbol{x}$ also allows us to formulate a tight binary decomposition in the constraint (7). Note that this is different from the standard BIL approach, which decomposes each integer variable in either $\boldsymbol{x}$ or $\boldsymbol{y}$.

We also see that Corollary 1 results in the linear constraints (8)-(11) and the valid inequalities (12)-(14). In the formulation (P2), we do not use the tighter bounds $\underline{q}_m + 1$ and $\overline{q}_m - 1$ because it may happen that at the end of the first step (a) of LINB, the optimal solution of (P0) is given by $(\boldsymbol{x}^*, \boldsymbol{y}^*)$. The advantages of the valid inequalities (12)-(14) will be evaluated via numerical

experiments in Section 3. We also note that the formulation (P2) allows further addition of various well known valid inequalities that have been used to enhance bilinear or non-convex IQP problems [2, 6, 9]. This can potentially further speed up the solution process, which can be useful especially when the problem size is large.

3. Numerical results

In this section, we present numerical results to verify the performance of LINB. A nonlinear bipartite matching problem considered in [1] is used to compare the proposed ILP approach LINB with LIN and LIN+, and their modifications eLIN and eLIN+. The problem description is repeated here for completeness. We consider a complete bipartite graph $G = (U \cup V, U \times V)$, in which each vertex v is associated with a price p_v and a weight w_v, and each edge (u, v) is associated with a maximum weight M_{uv}. The problem is to select the quantities x_u, where $u \in U$, and y_v with $v \in V$ such that the objective function $(\sum_{u \in U} p_u x_u)(\sum_{v \in V} p_v y_v)$ is maximized, subject to the constraints that $w_u x_u + w_v y_v \leq M_{uv}$ for every edge $(u, v) \in U \times V$. The problem can be rewritten in the form of (P0) by stacking the variables and coefficients in appropriate vectors, and hence solvable by any of the ILP approaches LIN, LIN+, eLIN, eLIN+, and LINB.

As in [1], we use a bipartite graph consisting of 50 vertices in each of U and V, with the weight w_v and price p_v of each vertex $v \in U \cup V$ drawn from a Poisson distribution with mean 4. The maximum weight M_{uv} on each edge $(u, v) \in U \times V$ is generated from a Poisson distribution of mean λ, where λ is equal to 8, 16 and 32. The larger the value of λ, the larger the space of feasible solutions, resulting in a harder problem to solve. In order to allow easy comparisons with the results in [1], we use the same ten random instances for each value of λ as used in [1] (these are available from the website of the second author of the reference). In addition, we also perform tests on 50 new random instances for each case. We set $L = 50$ for LIN+ [1]. All the solvers are implemented using IBM ILOG CPLEX 12.6 (for Windows 64 bits) interfaced with YALMIP [10] in MATLAB (R2013a), and the algorithms are run on a PC with Intel Xeon CPU E3-1225 V2 @ 3.20 GHz and 4.0 GB of RAM. The main computational results are summarized in Table 1.

The average number of iterations used by the random instances in [1] are almost the same as

Table 1: Average solution time (seconds): shaded rows show results obtained with the ten instances used in [1], while unshaded rows show results from 50 new random instances. The columns (itr.) show the average number of feasible iterations used.

λ	LIN	(itr.)	eLIN	(itr.)	LIN+	(itr.)	eLIN+	(itr.)	LINB	LINB1	LINB2	LINB3
8	0.095	(4.30)	0.109	(4.10)	0.087	(2.30)	0.086	(2.30)	0.082	0.072	0.081	0.079
	0.181	(6.84)	0.211	(6.82)	0.150	(3.78)	0.183	(3.76)	0.113	0.106	0.111	0.104
16	5.01	(31.00)	4.61	(30.10)	3.56	(20.00)	3.61	(19.60)	0.46	0.44	0.45	0.43
	9.01	(36.92)	8.74	(36.22)	7.26	(25.88)	7.85	(25.92)	0.88	0.87	0.82	0.80
32	536.40	(88.00)	642.16	(84.20)	409.77	(65.70)	612.52	(62.20)	16.56	16.45	18.42	18.64
	740.47	(120.38)	829.85	(117.60)	692.35	(84.86)	1433.06	(85.44)	28.92	30.89	30.67	30.25

those reported in [1], where the small differences are due to different versions of the CPLEX solver used. We observe that the solution time of LINB is significantly reduced compared to LIN and LIN+. Compared to LIN+, the percentage reduction in the average solution time is up to 95% for λ equal to 32. The reductions are even larger compared with LIN. We also observe that the performance of eLIN and eLIN+ are comparable to their counterparts LIN and LIN+ when λ is equal to 8 or 16, but are worse when $\lambda = 32$. In the case of eLIN, this indicates that the computational burden introduced by the upper bounds on the ensembles outweighs the reduction in the search space when the feasible region is large. Nonetheless, the average number of iterations used by eLIN are less than those used by LIN in searching for the optimal solutions. In the case of eLIN+, the adaptive value of L can be too conservative to be computationally efficient and meanwhile additional processing time is required by our implementation to redefine the optimization problem at every iteration. However, the average number of iterations used by eLIN+ is less than those used by LIN and eLIN, which is not always the case when compared to LIN+ because of the complex branch and cut processes invoked by the integer solver. In summary, our experiments show that making use of the upper bounds in Corollary 1 to enhance LIN or LIN+ does not necessarily improve the solution times, and such an approach is inferior to the LINB method.

We also applied the IQP solver in CPLEX 12.6 to the BIP. We observe that the average solution time is much longer than those using ILP approaches like LINB. For example, the average solution times of the ten instances used in [1] for $\lambda = 8$ and $\lambda = 16$ are 0.168 seconds and 95.21 seconds, respectively, and *none* of the instances for $\lambda = 32$ can be solved within a time limit of 3 hours. These

results show that the IQP approach is too general for the BIP, and there is significant performance gain if we exploit the ensemble product property of the bilinear objective function in developing ILP approaches for solving (P0).

The numerical results also confirm that in general, LIN, LIN+ and eLIN do not terminate with optimal solutions for (P0) in the last feasible iteration. For example, let us consider the experiments on the random instances used in [1]. In only two out of the ten instances with $\lambda = 16$, the solutions found by the last feasible iterations of LIN and eLIN coincide with the optimal solutions, showing that much computation time is wasted even though the optimal solution of (P0) has already been found in a previous iteration; a fact that the algorithm is unfortunately unaware of. We also observe that if L in LIN+ is not large enough, the same phenomenon appears in LIN+. If we let $L = 20$, only half of the ten instances result in an optimal solution for (P0) at the last feasible iteration. In contrast, eLIN+ always terminates at an optimal solution for (P0) in the last feasible iteration. These observations verify the theoretical conclusions in Lemma 2 and Lemma 3.

Additionally, we investigate the effect of the valid inequalities (12)-(14) on the performance of LINB. We consider three cases: (i) LINB1 where (12)-(13) are excluded; (ii) LINB2 where (14) is excluded; and (iii) LINB3 where (12)-(14) are excluded. From Table 1, we see that the valid inequalities help to reduce the solution time when λ is large.

Finally, we investigate how the solution time of LINB vary with the number of iterations used in eLIN+ in the first step of LINB. Experiments were carried out for the case where $\lambda = 32$, and the results are shown in Figure 1. The average solution time increases almost monotonically as LINB uses more iterations of eLIN+. This happens if the ensemble bounds improve slowly with more iterations of eLIN+, resulting in only slightly tighter binary decompositions in (P2). The results thus suggest that LINB should use only a single iteration of eLIN+, which is equivalent to calling LIN for only one iteration.

4. Conclusion

In this paper, we propose a new ILP approach to solve a BIP, based on a hybrid of the method proposed in [1] and the BIL approach. Our approach makes use of an upper bound on the optimal

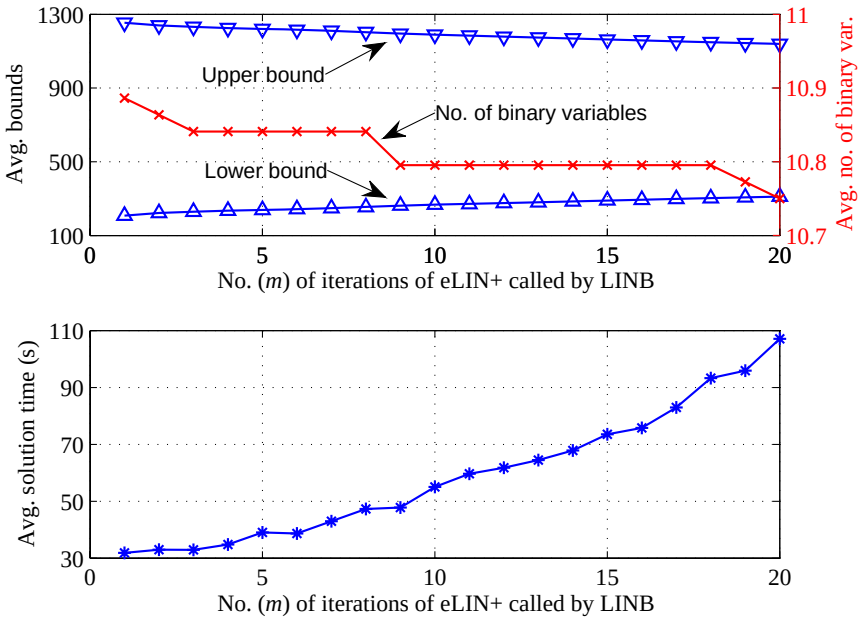

Figure 1: Numerical results averaged over 44 random instances.

solution for the BIP, which allows us to find tight binary decompositions for one of the ensembles in the BIP instead of decomposing every integer variable in the ensemble. Numerical experiments suggest that our proposed approach is much more efficient than existing methods, including the standalone BIL method, and those proposed in [1].

Acknowledgment

The first author thanks Professor E. Moreno for helpful discussions, and for sharing the CPLEX codes of LIN and LIN+, which have helped the author implement the algorithms efficiently. The authors are also grateful to the Associate Editor and anonymous reviewers for their valuable comments that helped to improve this article.

References

[1] A. S. Freire, E. Moreno, and J. P. Vielma, "An integer linear programming approach for bilinear integer programming," *Operations Research Letters*, vol. 40, no. 2, pp. 74–77, 2012.

[2] S. Burer and A. Saxena, "The MILP road to MIQCP," in *Mixed Integer Nonlinear Programming*. Springer, 2012, pp. 373–405.

[3] S. Burer and A. N. Letchford, "Non-convex mixed-integer nonlinear programming: a survey," *Surveys in Operations Research and Management Science*, vol. 17, no. 2, pp. 97–106, 2012.

[4] A. Qualizza, P. Belotti, and F. Margot, "Linear programming relaxations of quadratically constrained quadratic programs," in *Mixed Integer Nonlinear Programming*. Springer, 2012, pp. 407–426.

[5] W. P. Adams and H. D. Sherali, "A tight linearization and an algorithm for zero-one quadratic programming problems," *Management Science*, vol. 32, no. 10, pp. 1274–1290, 1986.

[6] A. Billionnet, S. Elloumi, and A. Lambert, "Linear reformulations of integer quadratic programs," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Springer, 2008, pp. 43–51.

[7] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I - Convex underestimating problems," *Mathematical Programming*, vol. 10, no. 1, pp. 147–175, 1976.

[8] I. Harjunkoski, R. Pörn, T. Westerlund, and H. Skrifvars, "Different strategies for solving bilinear integer non-linear programming problems with convex transformations," *Computers & Chemical Engineering*, vol. 21, pp. S487–S492, 1997.

[9] A. Gupte, S. Ahmed, M. S. Cheon, and S. S. Dey, "Solving mixed integer bilinear problems using MILP formulations," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 721–744, 2013.

[10] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB," in *IEEE International Symposium on Computer Aided Control Systems Design*, Taipei, Taiwan, 2004.