Operations Research Letters 42 (2014) 226–230

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

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

Wuhua Hu [*], Wee Peng Tay

School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 31 July 2013
Received in revised form
19 February 2014
Accepted 17 March 2014
Available online 22 March 2014

Keywords:
Bilinear integer programming
Integer linear programming
Integer quadratic programming
Binary binary linearization
Binary integer linearization
Bipartite graph

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.

1. Introduction

We consider the following bilinear integer program (BIP),

$$(\text{P0})\ \max_{\boldsymbol{x},\boldsymbol{y}}\ (\boldsymbol{\alpha}^T\boldsymbol{x})\cdot(\boldsymbol{\beta}^T\boldsymbol{y}) \qquad (1)$$

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

$$\boldsymbol{x}\in\mathbb{N}^N,\ \boldsymbol{y}\in\mathbb{N}^M$$

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 also use the superscript ^{T} to denote the 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 [5].

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 [4,3,10]. 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 [1,2], 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 [9]. 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 [2,7,6]. 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 [5]. 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 [5], the authors exploit a structural property of the objective function in (P0), and proposed novel *iterative* ILP

[*] Corresponding author. Tel.: +65 98514690.
E-mail addresses: w.hu920@gmail.com, hwh@ntu.edu.sg (W. Hu), wptay@ntu.edu.sg (W.P. Tay).

http://dx.doi.org/10.1016/j.orl.2014.03.002
0167-6377/© 2014 Elsevier B.V. All rights reserved.

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 [4,3,10,2]. Numerical experiments conducted by [5] 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 [5] 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 [5], 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 $p_1p_2 \le p_1^op_2^o$, then*

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

Furthermore, if $p_1p_2 < p_1^op_2^o$, then all the inequalities in (2) are strict.

Proof. We have $p_1(p_1+p_2) - p_1^2 = p_1p_2 \le p_1^op_2^o = p_1^o(p_1^o+p_2^o) - (p_1^o)^2 \le p_1^o(p_1+p_2) - (p_1^o)^2$, 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 $p_2(p_1+p_2)-p_2^2 = p_1p_2 \le p_1^op_2^o = p_2^o(p_1^o+p_2^o)-(p_2^o)^2 \le p_2^o(p_1+p_2)-(p_2^o)^2$, 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 $p_1p_2 < p_1^op_2^o$, 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 $p_1p_2 \ne p_1^op_2^o$, are optimal solutions for maximizing $q_1 + q_2$ and q_1q_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 $p_1p_2 < p_1^op_2^o$ 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:

$$\text{(P1)} \ \max_{\boldsymbol{x},\boldsymbol{y}} \boldsymbol{\alpha}^T\boldsymbol{x} + \boldsymbol{\beta}^T\boldsymbol{y}$$

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

$$\underline{p}_m \le \boldsymbol{\alpha}^T\boldsymbol{x} \le \overline{p}_m, \tag{3}$$

$$\underline{p}_m \le \boldsymbol{\beta}^T\boldsymbol{y} \le \overline{p}_m, \tag{4}$$

$$\boldsymbol{x} \in \mathbb{N}^N, \boldsymbol{y} \in \mathbb{N}^M.$$

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 $m \ge 1$. We call this the eLIN algorithm. The only difference between eLIN and LIN of [5] 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 the 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 $\beta_j > 0$ [5]. Otherwise, the best solution of all previous iterations gives the optimal solution of (P0), as shown in the following lemma.

Lemma 2. *Suppose that 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 the 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