An Integer Linear Programming Approach for Bilinear Integer Programming

Alexandre S. Freire[a], Eduardo Moreno[b], Juan Pablo Vielma[c]

[a] *IME - Universidade São Paulo*
[b] *Faculty of Engineering and Science, Universidad Adolfo Ibañez*
[c] *Department of Industrial Engineering, University of Pittsburgh*

Abstract

We introduce a new Integer Linear Programming (ILP) approach to solving Integer Programming (IP) problems with bilinear objectives and linear constraints. The approach relies on a series of ILP approximations of the bilinear IP. We compare this approach with standard linearization techniques on random instances and a set of real-world product bundling problems.

Key words: Bilinear programming, Integer programming, Product bundling

1. Introduction

We study the bilinear optimization problem

$$\max \left(\sum_i \alpha_i x_i \right) \left(\sum_j \beta_j y_j \right) \tag{1a}$$

$$\sum_{j=1}^{N} a_{k,j} x_j + \sum_{i=1}^{M} b_{k,i} y_i \leq d_k \quad \forall k = 1, \ldots, K \tag{1b}$$

$$x \in \mathbb{N}^N, \ y \in \mathbb{N}^M \tag{1c}$$

where coefficients α and β are nonnegative integers (through scaling we can also consider rational coefficients).

This integer programming (IP) problem with bilinear objective and linear constraints is a special case of non-convex quadratic IP problems and more generally of non-convex non-linear IP problems, both of which have received significant attention recently [1, 2, 3, 4]. If all variables are bounded this bilinear IP can be transformed to a Linear Integer Programming (ILP) Problem using well known linearization techniques [9, 6, 11, 10, 7, 8]. One such technique replaces x_j by its binary decomposition $\sum_{r=0}^{\lceil \log k_p \rceil} 2^r w_{j,r}$, where $w_{j,r}$ is

Email addresses: afreire@ime.usp.br (Alexandre S. Freire), eduardo.moreno@uai.cl (Eduardo Moreno), jvielma@pitt.edu (Juan Pablo Vielma)

a binary variable and k_j is an upper bound of x_j. It then adds auxiliary integer variables $z_{i,j,r}$ such that $z_{i,j,r} = w_{j,r} \cdot y_i$ by enforcing a standard linearization of this requirement to obtain the ILP formulation of (1) given by

$$\max \sum_{j=1}^{N} \sum_{i=1}^{M} \sum_{r=0}^{\lceil \log k_j \rceil} 2^r z_{i,j,r} \tag{2a}$$

$$\sum_{j=1}^{N} \sum_{r=0}^{\lceil \log k_j \rceil} 2^r a_{k,j} w_{j,r} + \sum_{i=1}^{M} b_{k,i} y_i \leq d_k \quad \forall k = 1, \ldots, K \tag{2b}$$

$$z_{i,j,r} \leq y_i \quad \forall\, i = 1, \ldots, M,\ j = 1, \ldots, N,\ r = 1, \ldots, \lceil \log k_j \rceil \tag{2c}$$

$$y_i \leq z_{i,j,r} + (1 - w_{j,r}) \cdot k_i \quad \forall\, i = 1, \ldots, M,\ j = 1, \ldots, N,\ r = 1, \ldots, \lceil \log k_j \rceil \tag{2d}$$

$$0 \leq z_{i,j,r} \leq w_{j,r} \cdot k_i \quad \forall\, i = 1, \ldots, M,\ j = 1, \ldots, N,\ r = 1, \ldots, \lceil \log k_j \rceil \tag{2e}$$

$$w_{j,r} \in \{0, 1\} \quad \forall\, j = 1, \ldots, N,\ r = 1, \ldots, \lceil \log k_j \rceil \tag{2f}$$

$$y \in \mathbb{N}^M \tag{2g}$$

where k_i is an upper bound on y_i. We can obtain an alternative formulation by also replacing x_i by its binary decomposition, but we would obtain a formulation that is even larger than the $\Theta(\log(k_p)|P||C|)$ variables and constraints of formulation (2). Because the size of formulation (2) can be prohibitively large, we propose a new linearization technique that leads to the solution of a series of ILP problems of the same size as the original problem.

Our proposed approach is detailed in Section 2. In Section 3 we present two family of problems that we use to test our methodology. One of them comes from a product bundling problem of a major food company. Finally, in Section 4 we show computational results of these models to compare our approaches with standard linearization techniques.

2. An ILP Solution Approach

In this section we present an ILP solution approach that is based on a linearization of the objective function. This approach is based in the following simple lemma.

Lemma 1. *Let a_1, a_2, b_1, b_2 be positive numbers. If $a_1 + a_2 \geq b_1 + b_2$ and $a_1 \cdot a_2 < b_1 \cdot b_2$ then $\min\{a_1, a_2\} < \min\{b_1, b_2\}$.*

Proof. Assume that $a_1 \leq a_2$ and $b_1 \leq b_2$. Let be $K_a = a_1 + a_2$, $K_b = b_1 + b_2$ and $f_K(x) = Kx - x^2$. Note that if $K_a \geq K_b$ then $f_{K_a}(x) \geq f_{K_b}(x)\ \forall x \in [0, K_b/2]$. Since $a_1 a_2 = f_{K_a}(a_1) < f_{K_b}(b_1) = b_1 b_2$ and due to $f_{K_a}(x)$ is strictly increasing in $[0, K_b/2]$ then $a_1 < b_1$. □

This lemma suggests the following idea to solve the problem: Instead of maximizing $(\sum_i \alpha_i x_i)(\sum_j \beta_j y_j)$ we can maximize $(\sum_i \alpha_i x_i) + (\sum_j \beta_j y_j)$. An optimal solution for this new objective function may be sub-optimal for the original objective function, but

if this is the case, we can use the lemma to obtain strict lower bounds for terms $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ in the optimal solution. We can use this idea to obtain an optimal solution to our original problem by solving a sequence of IP models with a linear objective as follows.

For a given lower bound LB for terms $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ we construct the ILP model with a linear objective given by

$$\max \sum_i \alpha_i x_i + \sum_j \beta_j y_j \tag{3a}$$

$$\sum_i \alpha_i x_i \geq LB \tag{3b}$$

$$\sum_j \beta_j y_j \geq LB \tag{3c}$$

$$Ax + By \leq d \tag{3d}$$

$$x_i, y_j \in \mathbb{N} \tag{3e}$$

We refer to this model as (LIN). In order to obtain the solution of our original problem, we start solving this IP with LB $= 1$. Next, we update this value by letting LB $= \min\{\sum_i \alpha_i x_i^*, \sum_j \beta_j y_j^*\} + 1$, where (x^*, y^*) is an optimal solution of the previous run, and we re-solve the ILP with the new parameter. We repeat this until the IP becomes infeasible. The best solution found (in terms of the original objective function $(\sum_i \alpha_i x_i)(\sum_j \beta_j y_j)$) is the optimal solution of our original problem.

Note that in each iteration, the original objective function might not improve. Hence, one way to enhance this approach is to modify the ILP model to ensures that new solution will always increase the original objective function. This is achieved as follows.

Let BEST and T be two parameters. The following formulation allows us to find a solution such that either (i) each term $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ are lower bounded by LB$+T$ or (ii) each term $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ is lower bounded by LB$+t$ for some $t = 1 \dots T$ and $(\sum_i \alpha_i x_i)(\sum_j \beta_j y_j) >$ BEST. If T is large enough, the original objective function

increases at each step.

$$\max \sum_i \alpha_i x_i + \sum_j \beta_j y_j \tag{4a}$$

$$\sum_i \alpha_i x_i \geq \sum_{t=0}^{T} s_t (LB + t) \tag{4b}$$

$$\sum_j \beta_j y_j \geq \sum_{t=0}^{T} s_t (LB + t) \tag{4c}$$

$$\sum_i \alpha_i x_i + \sum_j \beta_j y_j \geq \sum_{t=0}^{T-1} s_t \left((LB + t) + \left\lceil \frac{\text{BEST} + 1}{\text{LB} + t} \right\rceil \right) \tag{4d}$$

$$\sum_{t=0}^{T} s_t = 1 \tag{4e}$$

$$Ax + By \leq d \tag{4f}$$

$$x_i, y_j \in \mathbb{N} \tag{4g}$$

$$s_t \in \{0, 1\} \tag{4h}$$

Constraints (4b) and (4c) ensure that if $s_t = 1$ for some t then terms $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ are greater or equal than LB+t. Finally, constraints (4d) assure that the original objective function will increase. This is because if terms $\sum_i \alpha_i x_i$ and $\sum_j \beta_j y_j$ are greater or equal than LB $+ t$, and the sum of both terms is greater or equal than $(LB + t) + \lceil \frac{\text{BEST}+1}{\text{LB}+t} \rceil$, then by Lemma 1 the product $(\sum_i \alpha_i x_i)(\sum_j \beta_j y_j)$ should be greater or equal than $(LB + t) \cdot \lceil \frac{\text{BEST}+1}{\text{LB}+t} \rceil >$ BEST. We refer to this model as (LIN+)

As before, in order to obtain the solution of our original problem, we start solving this formulation with BEST $= 0$ and LB $= 1$. Next, we update these values with BEST $= (\sum_i \alpha_i x_i)(\sum_j \beta_j y_j)$ and LB $= \min\{\sum_i \alpha_i x_i, \sum_j \beta_j y_j\} + 1$, if a better solution has been found, or we update LB=LB+T, if not. Then, we re-solve the formulation with the new parameters until no feasible solution exists.

3. Testing instances

To compare our ILP approaches with linearization (2) we test them over two different problem: a nonlinear bipartite matching problem, and a real product bundling problem.

Bipartite graph problem

The first set of problems have been created to compare these different approaches. Given a complete bipartite graph $G = (L \cup R, L \times R)$, for each vertex v we set a price p_v and a weight w_v, and for each edge (u, v) we set a maximum weight M_{uv}. The problem is to select quantities x_u, $u \in L$ and y_v, $v \in R$ such that the objective function $(\sum_{u \in L} p_u x_u)(\sum_{v \in R} p_v y_v)$ is maximized, subject to $w_u x_u + w_v y_v \leq M_{uv}$ for each edge $uv \in L \times R$.

Product bundling problem

Our second set of problem comes from a major food company that is evaluating the possibility of delivering its products directly to small grocers, avoiding its current wholesalers and distributors. Because of distribution and storage logistics, the producer would like to select product bundles [13, 14] that maximizes the total sale of products through bundles, subject to constraints on client's demands.

Because bundles are more convenient for the producer, so we can expect the price of products obtained through the bundle to be equal or smaller that their individual prices. Hence, we can assume that clients will cover their demands by first buying as many units of the bundle as possible without exceeding their demand for an individual product and cover the remaining demand with individual purchases. Under this assumption, a mathematical programming model for obtaining the bundle design that maximizes the number of bundles sold can be constructed as follows.

Let P be a set of products. A bundle of products is an integer vector $x = (x_p)_{p \in P} \in \mathbb{N}^{|P|}$ where x_p indicates the number of units of product p in the bundle. Let C be a set of clients, $D_{c,p}$ be the demand of client $c \in C$ for product $p \in P$ and y_c be the number of bundles bought by client $c \in C$. A mathematical programming model for the optimal bundle design is given by

$$\max \sum_{c \in C} \sum_{p \in P} x_p \cdot y_c \tag{5a}$$

$$x_p \cdot y_c \leq D_{c,p} \quad \forall c \in C, p \in P \tag{5b}$$

$$x_p \in \mathbb{N} \quad \forall p \in P \tag{5c}$$

$$y_c \in \mathbb{N} \quad \forall c \in C \tag{5d}$$

Note that equation (5b) is not linear and hence our ILP approaches cannot be applied directly. However, using standard linearization techniques (e.g. [11]) it is possible to linearize such constraints in a compact ways that is not available for the objective function.

$$y_c \leq \left\lfloor \frac{D_{cp}}{i} \right\rfloor + \left(k_p - \left\lfloor \frac{D_{cp}}{i} \right\rfloor \right) \left(|b(i)| - \sum_{j : b(i)_j = 1} w_{p,j} \right) \quad \forall c \in C, \forall p \in P,\ i = 1 \dots k_p \tag{6a}$$

$$x_p = \sum_{i=0}^{\lceil \log k_p \rceil} 2^i w_{p,i} \quad \forall p \in P \tag{6b}$$

$$w_{p,i} \in \{0, 1\} \quad \forall p \in P, i = 1 \dots \lceil \log k_p \rceil \tag{6c}$$

where $b(i)$ is a vector containing the binary representation of the integer i (i.e. $i = \sum_j 2^{b(i)_j}$) and $|b(i)|$ is the number of non-zero bits in $b(i)$. In these constraints, note that $x_p = i$ if and only if the term $(|b(i)| - \sum_{j : b(i)_j = 1} w_{p,j})$ is equal to 0. Hence, these constraints assure that if $x_p = i$ then $y_c \leq \lfloor \frac{D_{cp}}{i} \rfloor$, so $x_p \cdot y_c \leq D_{cp}$. Note that this technique cannot be applied to original objective function or to constraints of the form $y_c \cdot x_p \leq z$ where z is a variable. However, because formulation (2) has already linearized terms $x_p \cdot y_p$ it can enforce constraints (5b) by adding linear constraints of the form $\sum_{r=0}^{\lceil \log k_p \rceil} 2^r z_{p,q,r} \leq D_{c,p}$ and hence does not require adding (6).

4. Computational results

We implement these formulations using IBM ILOG CPLEX 12.2 and we test them in a Intel Xeon server with 8 Gb of ram. In both problems, we compare our approach to the model obtained by replacing a set of variables by its binary decomposition, as explained in Section 1. We denote the models obtained by this decomposition by (BIN).

Bipartite graph problem

For the first problem, we use a complete bipartite graph of 50 vertices on each part, and we select random weights w_v and prices p_v following a Poisson probabilistic distribution of mean 4. The maximum weight on the arcs $M_{u,v}$ is also selected randomly following a Poisson distribution of mean λ, where λ is equal to 8, 16 and 32. Note that for higher values of λ, the space of feasible solutions is increased, obtaining a harder problem to solve. We compare solution times of ten random instances for each value of λ, and we report average times and number of iterations in Table 1. In the case of formulation (LIN+), a large value of parameter T leads us to fewer (but slower) iterations. In these experiments we use $T = 50$, which shows a good performance for these datasets.

	BIN	LIN		LIN+	
λ	time	time	(it.)	time	(it.)
8	6.0	0.56	(4.3)	2.99	(2.3)
16	4584.9	15.1	(31.2)	11.1	(19.3)
32	OOM	1637.6	(90.6)	1269.2	(63.2)

Table 1: Time results for our first problem using randomly generated data with $|L| = |R| = 50$

It can be seen that formulation (BIN) is significantly slower one in each case. Furthermore, in the larger problems, formulations (BIN) requires a large number of variables and constraints, resulting in an *Out of Memory (OOM)* error in the solver. Additionally, we see that in some instance formulation (LIN) requires a large number of iterations to finish, while formulation (LIN+) requires less iterations, but each iteration is harder to solve.

Product bundling problem

For the product bundling problem, we test the performance of our approaches on two set of instances. The first set of instances are randomly generated with $|C| = 10$ and $|P| = 30$. In order to generate these instances, we first fix $D_{c,p} = 0$ with uniform probability ρ. Secondly, among the values not fixed in zero, we fix $D_{c,p}$ according to a Poisson distribution of parameter λ. The results on these instances are shown in Table 2.

It can be seen that the problem became easier to solve for instances with a large number of zeros on the demand matrix. Also, with highly non-homogeneous matrices (p.e. $= 200$) problem became easier to solve for formulation (BIN), mainly because the LP relaxation of the problem is close to the optimal value. As before, formulation (BIN) is unable to solve large instances of this problem. In contrast, formulations (LIN) and

ρ	λ	BIN time	LIN time	(it.)	LIN+ time	(it.)
0.8	10	1	3	(9)	1	(2)
0.8	50	8	40	(19)	6	(2)
0.8	200	25	262	(35)	55	(4)
0.5	10	19	17	(11)	5	(3)
0.5	50	997	277	(28)	70	(4)
0.5	200	5511	27298	(77)	10874	(7)
0.2	10	287	71	(18)	20	(4)
0.2	50	6160	2156	(52)	2982	(9)
0.2	200	OOM	352889	(108)	114219	(12)

Table 2: Time results for product bundling problem on randomly generated data with $|C| = 10$ and $|P| = 30$

(LIN+) could deal with problems of these sizes, but they require a long time to solve them.

Our second set of instances for this problem is constructed using real data from a major food company, and we select subsets of clients and products of different sizes to compare the performance of our four formulations. Results are shown in Table 3. Results are similar to the obtained with randomly generated data. However, for instance *w10x60* (BIN) is significantly faster than (LIN) and (LIN+). One possible reason for this is that linearization (6) hurts the effectiveness of our ILP approaches. Another reason could be that, because demand values $D_{c,p}$ are very large for this instance, the solution of this problem includes 1 client that buy 29283 units. Both ILP approaches found this solution in the first iteration. However, it took many iterations to prove the optimality of this solution. Moreover, instance *w10x60d* has the same demand matrix than *w10x60* but in dozen of products, and now (LIN+) is faster than (BIN). As before, (BIN) is unable to solve our largest instance.

Name	$\|C\| \times \|P\|$	Max D_{cp}	BIN time	LIN time	(it.)	LIN+ time	(it.)
w5x41m	5 × 41	99	27	44	(31)	8	(5)
w5x41	5 × 41	433	92	127	(47)	33	(12)
w10x60	10× 60	2402	2393	199907	(129)	29213	(9)
w10x60d	10× 60	201	357	422	(35)	107	(3)
w32x311	32× 311	26	OOM	1175	(14)	193	(6)

Table 3: Time results for product bundling problem for instances on real data.

5. Impact on a real case

These models were used to construct bundles of food products for a major food company in Chile. We use the data of weekly demand on 497 products of 760 small retailers

with a total weekly demand of approximately 2 millions of SKU. Additionally to the original constraints, constraints were added to the size and final price of the bundle. To construct a bundle, a subset of 10-12 products was selected using hierarchical clustering with Euclidean distance, and solved using these formulations. After the selection of the first bundle, we update the demand and construct a new bundle using the same procedure. The IP model took approximately 5-10 hours per problem. Five bundles were constructed using this procedure, each one of them with more than 150 potential clients. A 29.75% of the total demand could be potentially delivered using these five bundles.

References

[1] P. Belotti, J. Lee, L. Liberti, F. Margot, A. Waechter, Branching and bound tightening techniques for non-convex MINLP, Optimization Methods and Software 24 (2009) 597–634.

[2] S. Leyffer, A. Sartenaer, E. Wanufelle, Branch-and-refine for mixed-integer nonconvex global optimization, Preprint ANL/MCS-P1547-0908, Argonne National Laboratory (September 2008).

[3] A. Saxena, P. Bonami, J. Lee, Convex relaxations of non-convex mixed integer quadratically constrained programs: Extended formulations, Mathematical Programming 124 (2010) 383–411.

[4] A. Saxena, P. Bonami, J. Lee, Convex relaxations of non-convex mixed integer quadratically constrained programs: Projected formulations, Mathematical Programming, Online First, DOI: 10.1007/s10107-010-0340-3.

[5] R. Jeroslow, There cannot be any algorithm for integer programming with quadratic constraints, Operations Research 21 (1973) 221–224.

[6] I. Harjunkoski, R. Pörn, T. Westerlund, H. Skrifvars, Different Strategies for Solving Bilinear Integer Non-Linear Programming Problems with Convex Transformations, Computers and Chemical Engineering 21 (1997) S487–S492.

[7] M. Padberg, The boolean quadric polytope: some characteristics, facets and relatives, Mathematical Programming 45 (1989) 139–172.

[8] H. Sherali, W. P. Adams, A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems, Kluwer Academic Publishers, 1999.

[9] F. Al-Khayyal, J. Falk, Jointly constrained biconvex programming, Mathematics of Operations Research 8 (1983) 273–286.

[10] G. McCormick, Nonlinear programming: theory, algorithms and applications, John Wiley & sons, 1983.

[11] T. Ibaraki, Integer programming formulation of combinatorial optimization problems, Discrete Mathematics 16 (1976) 39–52.

[12] R. Peeters, The maximum edge biclique problem is NP-complete, Discrete Applied Mathematics 131 (3) (2003) 651 – 654.

[13] K. F. McCardle, K. Rajaram, C. S. Tang, Bundling retail products: Model and analysis, European Journal of Operational Research 177 (2007) 1197–1217.

[14] S. Stremersch, G. J. Tellis, Strategic bundling of products and prices: a new synthesis for marketing, Journal of Marketing 66 (2002) 55–72.