ON A NEW CONSTRAINT REDUCTION HEURISTIC USING IMPROVED BISECTION METHOD FOR MIXED INTEGER LINEAR PROGRAMMING

HANDE GÜNAY AKDEMIR

Giresun University, Faculty of Arts and Sciences, Department of Mathematics, Giresun, Turkey

*Corresponding author: hande.akdemir@giresun.edu.tr

Abstract. In this study, we develop a surrogate relaxation-based procedure to reduce mixed-integer linear programming (MILP) problem sizes. This technique starts with one surrogate constraint which is a non-negative linear combination of multiple constraints of the problem. At this initial step, we calculate optimal Lagrangian multipliers from LP relaxation of the problem and use them as initial surrogate multipliers. We incorporate the improved bisection method (IBM) (B. Gavish, F. Glover, and H. Pirkul, Surrogate Constraints in Integer Programming, J. Inform. Optim. Sci. 12(2) (1991), 219–228.) into our algorithm. This simple heuristic algorithm is designed to iteratively generate a new surrogate cut that is to guarantee to satisfy the most violated two constraints of the corresponding iteration. The performance of the heuristic is tested using both some problems from the OR libraries and randomly generated ones.

1. Introduction

The objective here is to attempt to reduce the number of constraints of MILP type of problems. In the surrogate relaxation, a subset of constraints is substituted by a linear combination of these constraints, that is, we multiply these constraints with non-negative multipliers, and then aggregate them together to generate one combined constraint. With the optimal surrogate multipliers, this combined constraint serves as a proxy for the others and captures useful information. By doing that, we may obtain infeasible but near-optimal solutions while reducing processing time and complexity relative to the original model.
Besides, the use of surrogate constraints is known to provide better bounds relative to Lagrangian bounds. In the proposed heuristic, as initial surrogate multipliers, we simply use the optimal dual values (Lagrangian multipliers) of the LP relaxation to get the strongest initial surrogate constraint. We incorporate the IBM [12], which is an updated version of the bisection algorithm [13], into our heuristic. At each iteration, the procedure evaluates two infeasible constraints at a time, and uses the IBM to find the optimal surrogate multipliers while upper (or lower) bound decreases (or increases). Furthermore, we generate not one but several surrogate constraints and only relax inequality constraints. So, we solve a sequence of computationally easy relaxed and reduced problems to reach the optimal solution in a small amount of time. Our proposed heuristic also enables us to examine which ratio of the problem constraints are redundant and which ones are critical. The heuristic is implemented in the MATLAB environment. In order to test the algorithm, we use problems available for download from Beasley’s OR Library [4] and the Mixed Integer Programming Library [23]. We also conduct a set of testings using randomly generated instances. All of the computational experiments are done on a PC with an Intel Core i5-7400 CPU (3.00 GHz) and 4 GB of RAM.

Consider the following MILP problem:

\[
\begin{align*}
(P) \quad & \min \quad c^T x \\
\text{s.t.} \quad & Ax \leq b \\
& x \in X
\end{align*}
\]

where \( c, x \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n} \) and \( X \) is a discrete set that may be defined by some linear equalities and bound constraints of the decision variables.

By moving the constraints into the objective function, we generate Lagrange relaxation:

\[
\begin{align*}
\min \quad c^T x + \lambda^T (Ax - b) \\
\text{s.t.} \quad & x \in X
\end{align*}
\]

where Lagrange multipliers vector \( \lambda \geq 0 \).

Analogously, by assembling multiple constraints into a single new surrogate constraint, we generate surrogate relaxation:

\[
\begin{align*}
\min \quad c^T x \\
\text{s.t.} \quad & \mu^T (Ax - b) \leq 0 \\
& x \in X
\end{align*}
\]

where surrogate multipliers vector \( \mu \geq 0 \).

Both techniques enlarge the feasible region and provide a lower bound on the optimal objective value of Problem \((P)\). But, surrogate lower bound is tighter than the Lagrangian lower bound [14,17].
Relaxation-based search or dual algorithms, both exact and heuristics, have been extensively used in finding bounds for integer programming (IP). Let us now survey the related literature which investigates surrogate relaxation-based heuristics to solve combinatorial optimization problems. In [15], the author proposed a class of surrogate constraint heuristics that provide a variety of supplementing new alternatives and independent solution strategies. Lorena and Narciso [31, 39] proposed six heuristics based on both the surrogate and Lagrangian relaxations and a subgradient search algorithm for large scale generalized assignment problems. The authors showed that the use of procedures based on surrogate constraint analysis is effective for satisfiability problem in [30]. Applications of a classical combinatorial optimization problem called the set covering were given in [1, 11]. The former used the Surrogate constraint normalization technique to create appropriate weights for surrogate constraint relaxations, and the latter compared heuristics based on Lagrangian and surrogate relaxations for the Maximal Covering Location Problem. Karabati et al. [22] handled a class of discrete problems with min-max-sum objectives by using line search and surrogate procedures to obtain optimal surrogate multipliers. The combined use of surrogate and Lagrangian relaxation was considered for traveling salesman problem in [26, 32, 36]. The reduction is a preprocessing technique that is fundamental for developing efficient integer programming methods and based on dynamic programming and upper bounds. For a comprehensive literature review of reduction techniques, see [3, 10, 40]. Riberio and Lorena [46] examined an IP problem called cartographic label placement by using Lagrangian/surrogate heuristics. For a detailed review on solution techniques based on surrogate relaxations for $p$-median and facility location problems, see [43]. For a review of the approaches which combine metaheuristics with exact IP techniques, see [42]. In [27], a critical event tabu search heuristic was presented to solve the multidimensional knapsack problems (MKPs) with generalized upper bound constraints. Osorio et al. [41] considered a combined cutting and surrogate constraint analysis strategy for MKPs, see also [19]. The authors introduced problem-size reduction heuristics for MKPs in [21]. For heuristics and metaheuristics for MKPs and their variants, see recent works of [2, 5, 6, 8, 18, 20, 24, 25, 28, 29, 33, 35, 47–50], and references therein. For graph theory applications, such as graph coloring, weighted maximum clique, and shortest path problems see [9, 16, 37, 38]. Choi and Choi [7] proposed a redundancy identification method that is based on surrogate constraints. The relaxation adaptive memory programming approach based on surrogate constraints was proposed for combinatorial optimization problems in [44] and for capacitated minimum spanning tree problems in [45]. To review the scatter search method which was conceived as an extension of surrogate constraint relaxation, see [34].

The rest of this paper is organized as follows. Section 2 gives a brief version of the IBM, followed by the proposed heuristic is introduced in Section 3. Section 4 describes computational experiments. Section 5 concludes the paper.
2. Computing the optimal surrogate multipliers by IBM

Now, we consider problems with only two constraints. We start by determining which constraint is tighter, namely, it produces a lower (greater) objective value for maximization (minimization) problem considering only one constraint and ignoring the other one. Renumarate the tighter constraint as Constraint 1, and the other as Constraint 2. The surrogate multiplier of Constraint 1 is constant and equals to 1. Other multiplier is \( \mu \) which is going to be updated. The algorithm can be summarized as follows:

**Step 1.** Initial values of \( \mu_L \) and \( \mu_H \) are as following. The solution of the problem with multipliers \((1, \mu_L)\) and \((1, \mu_H)\) satisfies and does not satisfy Constraint 1, respectively.

**Step 2.** Let \( \mu = (\mu_L + \mu_H)/2 \). Solve the problem with the multipliers \((1, \mu)\). If (i) both constraints are satisfied stop, \((1, \mu)\) is the optimal multipliers, and the optimal solution of the problem is obtained.

(ii) only Constraint 1 is satisfied let \( \mu_L = f/g \) where \( f \) is the amount of oversatisfaction of Constraint 1 and \( g \) is the amount of undersatisfaction of Constraint 2.

(iii) only Constraint 2 is satisfied let \( \mu_H = f/g \) where \( f \) is the amount of undersatisfaction of Constraint 1 and \( g \) is the amount of oversatisfaction of Constraint 2.

**Step 3.** If \( \mu_L < \mu_H \) go to Step 2, otherwise Stop, \((1, \mu)\) is the optimal multipliers. Note that, we can also Stop if the objective function does not improve, or if the number of iterations reached an upper limit.

3. Constraint reduction heuristic

To determine the surrogate multipliers of the most violated two constraints of the current step, we apply the following iterative process which finds appropriate surrogate constraints.

**Initial solution:** We start by calculating optimal Lagrangian multipliers from LP relaxation of the problem and use them as initial surrogate multipliers. This surrogate constraint is our first surrogate constraint. Then, we solve our problem with this constraint and integrality restrictions.

**Adding a new surrogate constraint (generating a cut):** We first determine the most violated two constraints. Let

\[
g_t(x) = \sum_{j=1}^{n} a_{ij} x_j - b_i, \quad i \in I = \{1, 2, \ldots, m\},
\]

and

\[
\max_{i \in I} g_i(x) = g_t(x),
\]

\[
\max_{i \in I \setminus \{t\}} g_i(x) = g_k(x).
\]

Add a new surrogate constraint as \( \mu_t g_t(x) + \mu_k g_k(x) \leq 0 \) where \( \mu_t = \mu_k = 1 \). Solve the problem adding this new surrogate constraint to prior surrogate constraint(s). If \( g_t(x) \) and \( g_k(x) \) are non-positive while \( \mu_t = \mu_k = 1 \), add another surrogate constraint. Otherwise, apply the IBM supposing the positive one as
Constraint 1. If both constraints are satisfied then add a new surrogate constraint. Repeat these steps until all constraints are satisfied (within a tolerance value). Pseudo-code is given in Algorithm 3.1.

4. Computational results

This section presents the experimental results obtained with the proposed simple heuristic. Both test problems from the OR libraries [4, 23] and randomly generated ones are used to conduct the computational experiment. We only consider MILP problem cases which have non-negative Lagrange multipliers and at least one of them is non-zero. If this is not the case, we can assume that all of the initial surrogate multipliers are 1.

We denote the number of constraints by \( m \), and the number of variables by \( n \). For randomly generated 0-1 MKPs, we choose \( m \in \{3000, 4000, 5000\} \) and \( n \in \{150, 200\} \).

The constraint coefficients \( a_{ij} \) and objective function coefficients \( c_j \) are integers and randomly generated from the discrete uniform distributions \( U\{1, 2, \ldots, 500\} \) and \( U\{1, 2, \ldots, 100\} \), respectively. It is assumed that there are no correlations between the objective function and constraint coefficients. The right hand side values \( b_i \) are set to equal to \( s_i \sum_{j=1}^{n} a_{ij} \), where \( s_i \) is a slackness ratio and drawn from uniform distribution between 0.65 and 0.95. For each combination of \( (m, n) \), we generate 10 problems. Table 1 gives, for each combination of \( m \times n \), minimum, average (rounded off) and maximum computing times for both the intlinprog solver and the proposed heuristic. The heuristic performs better when the methods are compared by their average execution times which are less except for only one case. Table 1 indicates average execution times are reduced for almost every uncorrelated instance in which expected slackness ratio is 0.80.

In addition to the MKP instances, we test the proposed heuristic on a few instances from OR libraries [4, 23]. Refer to Table 2 for the results.

Note that, in all our calculations, the tolerance value are fixed to \( 10^{-6} \), the algorithm terminates if the upper bound does not improve 30 times, and for the IBM, the iteration upper limit is set to 10.

5. Conclusions

In this paper, we try to provide equivalent formulations with a fewer number of constraints for combinatorial optimization problems. To do that, we present a surrogate cut generation procedure based on the IBM. Experiments performed on problems with lots of redundant constraints have shown that the proposed heuristic gives reduced models which can be solved significantly faster than original models.
Algorithm 3.1 Constraint Reduction Heuristic

**INPUT:** Coefficient matrices and vectors of the MILP Problem \((P)\) \((A, b, c,\) and the set \(X)\)

**OUTPUT:** Optimal integer solution \(x\); number of constraints \(k\) after reduction; constraint coefficient matrix \(Sur\) after reduction; Right hand side vector \(SurRhs\) after reduction; Optimal surrogate weights matrix \(\mu\)

1: Solve the LP relaxation of Problem \((P)\) and calculate Lagrange multipliers, assign the corresponding Lagrange multipliers into the row vector \(\mu\) \((\mu(1, i) = \lambda(i)\) for \(i \in I = \{1, 2, \ldots, m\})

2: \(ConstNum \leftarrow\) the number of inequality constraints \((m), ToL \leftarrow 1e-6, IterUpLim \leftarrow 10, lowBound1 \leftarrow 0,\)
\(lowBound2 \leftarrow 0, w \leftarrow 0\) and \(k \leftarrow 1.\)

3: \textbf{function} SolveSur \((A, b, c, X, \mu)\)
4: \textbf{min} \(c^T x\)
5: \textbf{s.t.} \(\mu A x \leq \mu b\)
6: \(x \in X\)
7: \textbf{end function}

8: \textbf{function} lowBound \((k, c, x)\)
9: if \(k\) is odd then
10: \(lowBound1 = c^T x\)
11: else
12: \(lowBound2 = c^T x\)
13: end if
14: \textbf{Return} \(lowBound1, lowBound2\)
15: \textbf{end function}

16: SolveSur \((A, b, c, X, \mu)\)
17: lowBound \((k, c, x)\)
18: \textbf{while} \(\max(e) > ToL\) \textbf{and} \(k <= ConstNum\) \textbf{and} \(w <= 30\) \textbf{do}
19: if \(lowBound1 == lowBound2\) then
20: \(w + +\)
21: end if
22: \(r \leftarrow 0\)
23: \textbf{for} \(i \leftarrow 1, ConstNum\) \textbf{do}
24: \(f(i, 1) = e(i)\)
25: \(f(i, 2) = i\)
26: end for
27: Sort the rows of \(f\) in ascending order based on the elements in the first column and assign it to \(d\)
28: \(ind1 = d(ConstNum, 2)\)
29: \(ind2 = d(ConstNum - 1, 2)\)
30: \(mlow = 0\)
31: \(mhigh = 1\)
32: \(k + +\)
33: if \(e(ind1) > ToL\) \textbf{and} \(e(ind2) > ToL\) \textbf{then}
34: \(\mu(k, ind1) = 1\)
35: \(\mu(k, ind2) = 1\)
36: SolveSur \((A, b, c, X, \mu)\)
if $e(ind_1) > 0$ and $e(ind_2) < 0$ then
  if $e(ind_1) > TOL$ then
    $\mu(k, ind_2) = 0.5$
    \text{SOLVESUR}(A, b, c, X, \mu)$
  \end{if}
  \textbf{while} ($e(ind_1) > TOL$ or $e(ind_2) > TOL$) and $r <= IterUpLim$ \textbf{do}
  \textbf{if} $e(ind_1) > 0$ and $e(ind_2) \neq 0$ then
    $mhigh = |e(ind_1)/e(ind_2)|$
    \textbf{if} $mlow <= mhigh$ then
      $\mu(k, ind_2) = (mlow + mhigh)/2$
      \text{SOLVESUR}(A, b, c, X, \mu)$
      $r++$
    \end{if}
    \textbf{if} $mlow == mhigh$ then
      \textbf{break}
    \end{if}
  \end{if}
  \textbf{else}
    \textbf{break}
  \end{if}
\end{while}
\textbf{end if}
\else if $e(ind_2) > 0$ and $e(ind_2) \neq 0$ then
  $mlow = |e(ind_1)/e(ind_2)|$
  \textbf{if} $mlow <= mhigh$ then
    $\mu(k, ind_2) = (mlow + mhigh)/2$
    \text{SOLVESUR}(A, b, c, X, \mu)$
    $r++$
  \end{if}
  \textbf{if} $mlow == mhigh$ then
    \textbf{break}
  \end{if}
\textbf{else}
  \textbf{break}
\end{if}
\textbf{end if}
\textbf{end if}
\else if $e(ind_1) > 0$ and $e(ind_1) < 0$ then
  \textbf{if} $e(ind_2) > TOL$ then
    $\mu(k, ind_1) = 0.5$
    \text{SOLVESUR}(A, b, c, X, \mu)$
    \text{LOWBOUND}(k, c, x)$
  \end{if}
  \textbf{while} ($e(ind_1) > TOL$ or $e(ind_2) > TOL$) and $r <= IterUpLim$ \textbf{do}
  \textbf{if} $e(ind_1) > 0$ and $e(ind_1) \neq 0$ then
    $mlow = |e(ind_2)/e(ind_1)|$
    \textbf{if} $mlow <= mhigh$ then
      $\mu(k, ind_1) = (mlow + mhigh)/2$
      \text{SOLVESUR}(A, b, c, X, \mu)$
      $r++$
    \end{if}
    \textbf{if} $mlow == mhigh$ then
      \textbf{break}
    \end{if}
  \end{if}
\textbf{end if}
else
    Break
end if
else if \( e(\text{ind}2) > 0 \) and \( e(\text{ind}1) \neq 0 \) then
    \( m\text{high} = |e(\text{ind}2)/e(\text{ind}1)| \)
    if \( m\text{low} \leq m\text{high} \) then
        \( \mu(k, \text{ind}1) = (m\text{low} + m\text{high})/2 \)
        \text{SolveSur}(A, b, c, X, \mu)
        \( r + + \)
        if \( m\text{low} == m\text{high} \) then
            Break
        end if
    else
        Break
    end if
else
    Break
end if
end if
end while
\text{lowBound}(k, c, x)
end if
end if
else if \( e(\text{ind}1) > \text{ToL} \) and \( e(\text{ind}2) < \text{ToL} \) then
    \( \mu(k, \text{ind}1) = 1 \)
\text{SolveSur}(A, b, c, X, \mu)
\text{lowBound}(k, c, x)
end if
end while
\text{Sur} = \mu A
\text{SurRhs} = \mu b
Table 1. Computational results of randomly generated 0-1 MKPs.

<table>
<thead>
<tr>
<th>$m \times n$</th>
<th>intlinprog</th>
<th>Heuristic</th>
<th>Number of Const.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min ave max</td>
<td>min ave max</td>
<td>min ave max</td>
</tr>
<tr>
<td>3000 $\times$ 150</td>
<td>85 232 520</td>
<td>34 145 538</td>
<td>17 33 62</td>
</tr>
<tr>
<td>3000 $\times$ 200</td>
<td>425 1050 3203</td>
<td>168 960 3860</td>
<td>31 40 60</td>
</tr>
<tr>
<td>4000 $\times$ 150</td>
<td>59 257 626</td>
<td>31 207 805</td>
<td>18 36 76</td>
</tr>
<tr>
<td>4000 $\times$ 200</td>
<td>66 528 1391</td>
<td>30 443 2043</td>
<td>20 28 41</td>
</tr>
<tr>
<td>5000 $\times$ 150</td>
<td>149 861 2051</td>
<td>21 942 2761</td>
<td>26 45 83</td>
</tr>
<tr>
<td>5000 $\times$ 200</td>
<td>670 2000 2994</td>
<td>209 1390 4296</td>
<td>27 42 58</td>
</tr>
</tbody>
</table>

Table 2. Computational results for instances from OR libraries [4,23].

<table>
<thead>
<tr>
<th>Problem Key</th>
<th>Inequality Constraints</th>
<th>Equality Constraints</th>
<th>Variables</th>
<th>Ineq. Const. After Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST1</td>
<td>30</td>
<td>0</td>
<td>60</td>
<td>13</td>
</tr>
<tr>
<td>ST2</td>
<td>30</td>
<td>0</td>
<td>60</td>
<td>14</td>
</tr>
<tr>
<td>PB6</td>
<td>30</td>
<td>0</td>
<td>40</td>
<td>13</td>
</tr>
<tr>
<td>PB7</td>
<td>30</td>
<td>0</td>
<td>37</td>
<td>15</td>
</tr>
<tr>
<td>f2gap40400</td>
<td>40</td>
<td>0</td>
<td>400</td>
<td>21</td>
</tr>
<tr>
<td>f2gap401600</td>
<td>40</td>
<td>0</td>
<td>1600</td>
<td>27</td>
</tr>
<tr>
<td>f2gap801600</td>
<td>80</td>
<td>0</td>
<td>1600</td>
<td>47</td>
</tr>
<tr>
<td>app2-1</td>
<td>1038</td>
<td>0</td>
<td>3283</td>
<td>2</td>
</tr>
<tr>
<td>app2-2</td>
<td>335</td>
<td>0</td>
<td>1226</td>
<td>1</td>
</tr>
<tr>
<td>supportcase14</td>
<td>127</td>
<td>107</td>
<td>304</td>
<td>4</td>
</tr>
<tr>
<td>misc07</td>
<td>177</td>
<td>35</td>
<td>206</td>
<td>136</td>
</tr>
<tr>
<td>blend2</td>
<td>185</td>
<td>89</td>
<td>353</td>
<td>64</td>
</tr>
<tr>
<td>bppc8-02</td>
<td>39</td>
<td>20</td>
<td>232</td>
<td>33</td>
</tr>
<tr>
<td>BeasleyC1</td>
<td>1250</td>
<td>500</td>
<td>2500</td>
<td>628</td>
</tr>
</tbody>
</table>
Data Availability: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The author declares that there are no conflicts of interest regarding the publication of this paper.

References


