An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem

Arnaud Freville*,a, Gérard Plateaub

*aLaboratoire de modélisation mathématique et informatique, Institut des Sciences et Techniques de Valenciennes, Université de Valenciennes, BP 311, 59304 Valenciennes Cédex, France

bLaboratoire d'Informatique de Paris-Nord, CNRS URA 1507, Institut Galilée, Université de Paris-Nord, Avenue J.B. Clément, 93430 Villetaneuse, France

Received 22 February 1991; revised 15 January 1992

Abstract

The multidimensional 0-1 knapsack problem, defined as a knapsack with multiple resource constraints, is well known to be much more difficult than the single constraint version. This paper deals with the design of an efficient preprocessing procedure for large-scale instances. The algorithm provides sharp lower and upper bounds on the optimal value, and also a tighter equivalent representation by reducing the continuous feasible set and by eliminating constraints and variables. This scheme is shown to be very effective through a lot of computational experiments with test problems of the literature and large-scale randomly generated instances.

Key words: 0-1 multidimensional knapsack; Heuristics; Relaxation; Reduction

Notation

- \lfloor x \rfloor: the largest integer less than or equal to \( x \),
- \text{Co}(X): the convex hull of the set \( X \),
- \( U(a, b) \): the uniform distribution into the range \([a, b]\),
- \( (\cdot) \): the linear programming relaxation of a given linear integer problem \((\cdot)\),
- \( r(\cdot) \): the optimal value of a given problem \((\cdot)\),
- \( ||x|| \): euclidean norm of the vector \( x \),
- \( |X| \): cardinality of the finite set \( X \),
- \( A_i, A^T, A_{ij} \): respectively a row, a column and a coefficient of the matrix \( A \).

*Corresponding author.
0. Introduction

The multidimensional 0-1 knapsack problem is typically encountered in resource allocation models. Historically, the first example has been exhibited by Manne and Markowitz [49] as a capital budgeting model. Economic applications appeared again in [67], but other applications of the problem include project selection [54], cutting stock [29] and loading problems [3, 63]. More recently, this problem was identified as a subproblem in large models for allocating processors and databases in a distributed computer system [24, 25].

The multidimensional 0-1 knapsack problem is stated as

\[
(P) \quad \max \quad z = cx \\
\text{s.t.} \quad Ax \leq b, \\
\quad x \in B = \{x \in \mathbb{R}^n | x_j \in \{0, 1\}, j = 1, \ldots, n\},
\]

where \( A \) is an integer nonnegative \( m \) by \( n \) matrix, \( b \) and \( c \) are positive integer vectors of size \( m \) and \( n \) respectively, with \( m \leq n \) when the number of variables is growing.

This assumption with the nonnegativeness of the dense matrix \( A \) and the nonexistence of special constraints such as generalized upper bounds, special-ordered set and plant-location constraints, distinguish this problem from the general 0-1 linear programming problem. These hypotheses are fundamental because it has been shown that the existence of special constraints is essential to derive efficient methods for solving large-scale 0-1 linear problems [8, 36, 40].

Complexity theory gives an answer to understand the increase of difficulties due to the change of a single constraint by multiple constraints. As the single case, the multidimensional 0-1 knapsack problem is NP-complete but not strongly NP-complete. Polynomial approximation algorithms for single-dimensional knapsack problems have been extensively studied and a number of such algorithms generalized for \( m > 1 \) [21, 52]. Nevertheless, if fully polynomial approximation schemes have been given for \( m = 1 \) [37, 44, 48], finding fully polynomial approximation algorithms is NP-hard for \( m > 1 \) [43, 47].

Computational studies confirms this worst-case property. The 0-1 knapsack problem is well solved for many classes of instances, although hard problems have been identified [7, 53, 60, 64]. Instances with randomly generated data and \( n \) up to 250 000 may be solved by recent exact algorithms including implicit enumeration and hybrid strategies [2, 5, 12, 50, 51, 57]. On the other hand relatively few studies have addressed the multidimensional case. Moreover, the largest sizes reached up to now do not exceed 5 constraints and 200 variables [26, 27, 54, 61, 63, 65, 68].

Thus we present in this paper an effective preprocessing procedure for solving approximately (or possibly exactly when the surrogate dual solves the primal) multidimensional 0-1 knapsack problems of large size. Section 1 deals with the derivation of sharp lower bounds of the optimal value. The heuristic procedure presented in Section 1.1 is the final version of previous works [17, 18]. In Section 1.2, this algorithm is compared favorably with the heuristic and the early termination procedure developed by Gavish and Pirkul [26, 55]. Section 2.1 describes various strategies to compute upper bounds of the optimal value through the linear programming
relaxation and the surrogate dual. Section 2.1.1 is related to an exact procedure for solving the surrogate dual of the bidimensional 0-1 knapsack problem [20]. Section 2.1.2 describes a subgradient approach when \( m > 2 \) [16]. Computational results are presented in Section 2.2, which compare the above-mentioned procedures with classical dual techniques. A new sequential reduction scheme based on previous papers is presented in Section 3.1 [14, 15, 18, 19]. Section 3.2 details its computational efficiency in terms of size and continuous feasible set reduction, and outlines the quality of the bounds obtained for large-scale instances.

An experimental code has been implemented in Fortran 77 on a SUN station. A copy can be obtained upon request from one of the authors.

1. Lower bounds of the multidimensional 0-1 knapsack problem

The main ability of heuristic procedures is to produce quickly near-optimal solutions, especially for difficult optimization problems as large-scale multidimensional 0-1 knapsack problems. Yet heuristic procedures also take a prominent part in enumeration methods by providing good starting feasible solutions, and in reduction schemes by allowing to fix variables at their optimal value and to derive sharp bounds on the sum of variables fixed at 1 at the optimum.

A general categorized survey on heuristic methods and applications may be found in Zanakis et al. [69]. A detailed bibliography of papers devoted to heuristic procedures applied to the multidimensional 0-1 knapsack problem is presented in [17].

1.1. Outline of a new version of the heuristic procedure AGNES

The AGNES method is based initially on some heuristic proposals for integer programming developed in two papers of Glover [31, 32]. The main idea is to use different techniques from informations included in several surrogate constraints to generate a sequence of feasible solutions.

Starting with a dual multiplier \( u \in \mathbb{R}^n_+ = \{ u \in \mathbb{R}^n | u \geq 0 \} \), the first three phases are greedy algorithms:

- AGNES0 generates a first feasible solution by filling up the multidimensional 0-1 knapsack problem according to decreasing order of the \( c_j / u A_j \) ratios, as far as possible.
- AGNES1 uses an oscillating assignment technique from the optimal solution \( \tilde{x} \) of a perturbed continuous surrogate relaxation \( (\tilde{S}_c(u)) \): \( \max c x \) s.t. \( u A x \leq (1 + \varepsilon) u b \), \( x \in \text{Co}(B) \). The variables such that \( \tilde{x}_j = 1 \) are fixed at 1 temporarily, the others at 0. If the solution generated in this way is infeasible, a greedy algorithm, like the dual procedure described in [62], sets free a part of the variables fixed at 1 to move back into the feasible region. Then the variables remaining at 1 are definitively fixed, and the procedure restarts with the subproblem defined by the free variables. Several feasible solutions are generated according to different values of the parameter \( \varepsilon \) taken into the interval \([-0.2, 0.2]\).
- AGNES2 brings into play the concept of strongly determined variables. Given a rational \( x \in ]0, 1[ \), the procedure fixes \( \lfloor xn \rfloor \) variables according to the largest
reduced prices related to a continuous surrogate relaxation \( \bar{S}(u) \): max \( cx \) s.t. \( uAx \leq ub, x \in \text{Co}(B) \); if the reduced price is positive (respectively, negative), the corresponding variable is fixed at 1 (respectively, at 0). Then the procedure restarts with the incumbent problem defined by the free variables. Several feasible solutions are calculated from different values of the parameter \( \alpha = \frac{1}{3}, \frac{1}{4}, \frac{1}{5} \).

The last phase, called AGNES3, is a \( k \)-interchange heuristic which tries to find a better feasible solution than the incumbent \( \bar{x} \), by making only limited changes defined by the neighbourhoods \( \{ x \in B \mid \sum_{j \in I_k} |x_j - \bar{x}_j| = k \} \). For the procedure to be fast, we have limited \( k \) to value 1 in the numerical experiments.

The time complexity of AGNES0 is \( O(mn \log n) \), since \( O(n \log n) \) time is required to sort the \( c_j/uA^j \) ratios. We have proved in [14] that AGNES2 heuristic requires \( O(mn) \) time. However, the quadratic time complexity \( O(mn^2) \) of AGNES1 may limit its systematic utilization in enumeration procedures.

The AGNES heuristics is running successively with different starting multipliers \( u \in \mathbb{R}^m_+ \) and the best feasible solution is preserved. The employed multipliers are:
- structural coefficients defined as \( u_i = (\sum_{j=1}^n A_{ij} - b_j)/\sum_{j=1}^n A_{ij}, i = 1, \ldots, m \);
- the optimal dual solution of the continuous relaxation (\( \bar{P} \));
- the optimal solution of the surrogate dual (\( \bar{S} \)).

### 1.2. Performance of heuristic procedure AGNES

The heuristic procedure AGNES performs very well on the test problems of the literature (55 moderate size instances detailed in [19, 49, 54, 62, 63, 68]). The optimal solution is reached in all cases, except with Petersen’s problem 7 (\( m = 5, n = 50 \)) and Freville–Plateau’s 2 (\( m = 4, n = 34 \)), where the best obtained solution is 16 504 (respectively 3148) instead of 16 537 (respectively 3186). The two approximate values are better than the ones provided by the Balas and Martin’s Pivot and Complement method [1].

AGNES heuristic has been also compared with the MKHEUR heuristic [55] and the optimal solution method of Gavish–Pirkul, called GP-procedure [26]. A set of randomly generated instances has been considered, with matrix coefficients drawn from \( U(0, 1000) \). Each class contains 30 problems with the same number of variables and constraints. For each class, the right-hand side is determined by summing the columns of \( A \) and by multiplying this sum with 0.25 for ten problems, with 0.50 for ten others and with 0.75 for the last ones. The coefficients of the objective function correlate with the columns of \( A \) as follows:

\[
c_j = \left( \sum_{i=1}^m A_{ij} \right)/m + r_j \ast k, \quad j = 1, \ldots, n,
\]

where \( r_j \) is a real number drawn from \( U(0, 1) \) and \( k \) is a constant equal to 500 in the numerical experiments.

Table 1 clearly indicates the superiority of AGNES over MKHEUR by means of percent deviations; it is not surprising because the MKHEUR procedure is in fact equivalent to the two phases AGNES0 and AGNES3. This also explains the larger computing time of AGNES method. But it is important to point out that 90% of total
Table 1
Comparison of the methods AGNES, MKHEUR and GP-procedure (average performance over 30 instances for each size)

<table>
<thead>
<tr>
<th>Size</th>
<th>AGNES</th>
<th>MKHEUR</th>
<th>GP-procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>mean</td>
<td>max</td>
</tr>
<tr>
<td>5</td>
<td>0.48</td>
<td>1.15</td>
<td>2.60</td>
</tr>
<tr>
<td></td>
<td>0.24</td>
<td>0.31</td>
<td>0.39</td>
</tr>
<tr>
<td>10</td>
<td>1.82</td>
<td>3.14</td>
<td>5.77</td>
</tr>
<tr>
<td></td>
<td>0.48</td>
<td>0.96</td>
<td>1.19</td>
</tr>
<tr>
<td>30</td>
<td>3.12</td>
<td>6.86</td>
<td>13.93</td>
</tr>
<tr>
<td></td>
<td>1.31</td>
<td>2.44</td>
<td>3.70</td>
</tr>
<tr>
<td>500</td>
<td>0.51</td>
<td>1.65</td>
<td>3.33</td>
</tr>
</tbody>
</table>

min, mean, max: deviation percentage = $100 \times \frac{|v(P) - v(P)|}{v(P)}$.
time: average computing time in seconds on a SUN station 3/50.
a: the enumeration is stopped after scanning $10^6$ nodes.

Computing time is consumed by the quadratic time complexity AGNES1 phase. On the other hand, the performances of GP-procedure worsen at the same time the number of variables is increasing. Table 1 shows that AGNES heuristics is an effective alternative to GP-procedure for large-scale instances, the generated feasible solution being better than the best feasible solution identified after enumerating one million nodes. So the effectiveness of the three procedures is measured by the relative gap between the lower bound $v(P)$ and the optimal value $v(P)$ of the LP relaxation $(\tilde{P})$:

$$(\tilde{P}) \quad \max cx \text{ s.t. } Ax \leq b, \quad x \in \text{Co}(B).$$

2. Derivation of upper bounds and dual multipliers

A fundamental task in integer programming is the choice of relaxations allowing to obtain, quickly and at lowest time consuming, tight upper bounds of the current problem. The solving of the multidimensional 0-1 knapsack problem needs relaxations at several levels.

- **Heuristics**: feasible solutions are generated, either straightforwardly through optimal solutions of Lagrangean relaxations, or indirectly by using surrogate relaxations as in AGNES.
- **Reduction**: upper bounds are required for the elimination of redundant constraints, fixing variables, generation of logical relations and computation of bounds on the sum of variables fixed at 1 at the optimum.
- **Partial enumeration**: for pruning nodes of the enumeration tree.

Generally, a trade-off is required between the quality of the bound provided by the relaxation and the running time consumed by solving it. Still, a great part of the
above-mentioned applications needs only an answer to the following question:
"Does there exist an upper bound of the current problem less than a given threshold?"
Then it is particularly interesting to have at one's disposal dual procedures with
a monotonic behaviour, even if the convergence is near-optimal. The utilization of
such monotonic dual procedures leads clearly to decreased computing time.

2.1. Outlines of new constructive dual techniques

For a given multidimensional 0-1 knapsack problem \((P)\), we define now the
Lagrangean and the surrogate duals of the primal problem \((P)\), relative to the
constraints \(Ax \leq b\):

\[
\begin{align*}
(L) & \quad \min_{u \in \mathbb{R}^m} v(L(u)) \\
(S) & \quad \min_{u \in \mathbb{R}^m} v(S(u))
\end{align*}
\]

where

\[
\begin{align*}
v(L(u)) &= \max\{cx + u(b - Ax)|x \in \mathcal{B}\}, \\
v(S(u)) &= \max\{cx|uAx \leq ub, x \in \mathcal{B}\}.
\end{align*}
\]

We recall the main relations between all the optimal values of the above-mentioned
problems [28, 33]:

\[
v(P) \leq v(S) \leq v(L) = v(\bar{P}) = v(\bar{S}).
\]

2.1.1. Exact solution procedure for the surrogate dual when \(m = 2\)

We derive in the case \(m = 2\), a procedure, called SADE\(^2\), which provides in a finite
number of arithmetic operations, either an optimal surrogate dual solution or an
optimal primal solution. The framework of the method extends the previous results of
Glover [30].

The main characteristics of the procedure SADE\(^2\) are the following (see [20] for
a detailed description):

- The nice property of the surrogate dual function stated as

  \[
  \forall u \in \mathbb{R}^m, \quad \forall \lambda > 0 \quad v(S(\lambda u)) = v(S(u))
  \]

  allows the solving of dual \((S)\) in a one-dimensional search

  \[
  \begin{align*}
  (S) & \quad \min_{u \in [0, +\infty]} v(S(\mu)),
  \end{align*}
  \]

  by normalizing the multiplier \(u = (u_1, u_2) \in \mathbb{R}^2_+\) as follows:

  \[
  \mu = \begin{cases}
  \frac{u_2}{u_1} & \text{if } u_1 > 0, \\
  +\infty & \text{otherwise}.
  \end{cases}
  \]

- By exchanging the two constraints if necessary, the line search is carried on the
  compact interval \([0, 1]\), such that the first (respectively second) constraint is violated
  by the calculated optimal solution of the relaxation \((S(1))\) (respectively \((S(0))\).
The procedure SADE\textsuperscript{2} performs a modified dichotomic search by using the following property.

Let $\mu^o \in [0, 1]$, $x^*(\mu^o)$ be an optimal solution of the surrogate relaxation $(S(\mu^o))$ and $\alpha^o = (b_1 - A_1 x^*(\mu^o))/(A_2 x^*(\mu^o) - b_2)$. Then

(i) if $x^*(p')$ satisfies the first constraint and violates the second, then $\alpha^o \geq \mu^o$ and for any $\mu \in [0, \alpha^o]$, $v(S(\mu)) \geq cx^*(\mu^o) - v(S(\mu^o))$.

(ii) if $x^*(\mu^o)$ violates the first constraint and satisfies the second, then $\alpha^o \leq \mu^o$ and for any $\mu \in [\alpha^o, + \infty]$, $v(S(\mu)) \geq cx^*(\mu^o) - v(S(\mu^o))$.

(iii) if $x^*(\mu^o)$ satisfies the two constraints then $x^*(\mu^o)$ is obviously an optimal solution of the primal problem $(P)$ and $v(P) = v(S) = cx^*(\mu^o)$.

So the incumbent search interval $[\mu_l, \mu_u]$ is replaced by the tighter one $[\alpha_l, \alpha_u]$ at any iteration, and the procedure is stopped as soon as the criterion of dual optimality $\alpha_l \geq \alpha_u$ is verified. The finite termination has been proved, which is an important improvement over previous methods. Particularly, stopping criteria of the form $\mu_u - \mu_l \leq \epsilon$, where $\epsilon$ is a given precision threshold, are now obsolete.

Numerical experiments show the independence of the iteration number with the size of the instances. In nearly all the cases, this number is less than or equal to 10.

2.1.2. Efficient subgradient algorithms for the surrogate dual when $m > 2$

It is well known that slow convergence and nonmonotone behaviour are two main undesirable features of subgradient methods. Many alternative procedures have been proposed to obtain a monotonic behaviour by using subgradient-type algorithms for nonsmooth optimization, such as $\epsilon$-subgradient algorithms, bundle methods and subgradient methods with space dilatation [22, 41, 42, 45, 46, 64].

The aim of our contribution is centralized in controlling the step-size $t$ of the below algorithm $G$ for generating monotone decreasing sequences of surrogate and Lagrangean relaxation values [16].

Algorithm $G$

Let $u \in \mathbb{R}^m_+$ be any multiplier such that $u \neq 0$;

repeat

{Surrogate relaxation}
Solve the surrogate relaxation $(S(u))$ for an optimal solution $x^*$;

{Optimality}
If $Ax^* - b \leq 0$ then $x^*$ solves the primal problem $(P)$; stop;

{Subgradient step}
Compute $g_u = \sum_{j \in J_u} A_j - b$ where $J_u = \{j \in \{1, \ldots, n\} | x^*_j = 1\}$;

for an appropriate step-size $t > 0$ compute a new multiplier

$$u \leftarrow u + t \frac{g_u}{\|g_u\|^2};$$

{Projection}
Project the new multiplier \( u \) on \( \{u | u \in \mathbb{R}^n_+\} \) by setting
\[
{u}_i \leftarrow \max\{0, {u}_i\} \quad \text{for all } i;
\]

until termination criteria.

In the surrogate case, the algorithm \( G \) is indeed a quasi-subgradient algorithm as in [9]. Let \( u, \overline{u}, \bar{u} \) denote three consecutive multipliers generated by algorithm \( G \), and suppose that \( u \) and \( \overline{u} \) have been already calculated. Some theoretical results detailed in [16] suggest to select the step-size \( t \) such as
\[
t = |u_{\overline{u}}|,
\]
then
\[
\bar{u} = u + t \frac{g_u}{\|g_u\|^2}.
\]

Obviously, this step-size selection leads neither to a theoretical convergent procedure nor to a monotone decreasing procedure. However, the behaviour is in practice nearly monotonic, and for this reason, we use the number of iterations with no decrease of function values, as a termination criterion.

In the Lagrangean case, the surrogate relaxation \((S(u))\) is replaced by its linear programming relaxation \((\overline{S}(u))\). We use the following link between the Lagrangean dual \((L)\) and the continuous surrogate dual \((\overline{S})\):
\[
\forall u \in \mathbb{R}_+^n, \quad v(\overline{S}(u)) = v(L(w)) = \min_{\lambda \geq 0} v(L(\lambda w)),
\]
where \( w = \lambda_u \) with \( \lambda_u \) an optimal dual multiplier of the continuous knapsack \((\overline{S}(u))\).

So \( g_u \) is a subgradient of the continuous piecewise convex dual function \( v(L(w)) \) at the point \( w = \lambda_u \) and the multiplier \( u \) appears as the best one over all the Lagrangean multipliers lying in the one-dimensional space \( \{w | w = \lambda u, \lambda \geq 0\} \).

The step-size \( t \) is selected by
\[
t = \rho \frac{v(\overline{S}(u)) - v(P)}{\lambda_u},
\]
where \( \rho \) is a suitable positive constant, and target \( v(P) \) is a lower bound on \( v(P) \).

This choice leads to the move of a classical Lagrangean subgradient search into the continuous surrogate space. Thus algorithm \( G \) can be viewed as an accelerated subgradient method. Moreover, a monotone decreasing behaviour becomes patent in nearly all the cases when the normalization parameter \( \rho \) is suitable.

2.2. Comparison with other constructive dual techniques

For the solving of bi-dimensional 0-1 knapsack surrogate dual, Table 2 compares SAD\(2\) procedure with a standard dichotomic method, called DICHO, and a quasi-subgradient algorithm QSG developed by Dyer [9]. The range of the uniform distribution for generating \( c \) and \( A \) is, respectively, \([1, 1000]\) and \([0, 1000]\), and the
Table 2

Exact solution of the bi-knapsack surrogate dual (average performance over 10 instances for each size)

<table>
<thead>
<tr>
<th>Data generation method</th>
<th>SADE2</th>
<th>DICH0</th>
<th>QSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c, A, b )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x = 0.1 \beta = 0.8 \delta_i \in U(0, 1) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( a + b \delta_i \in [0.1, 0.9] )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>100</td>
<td>3.4</td>
<td>5.3</td>
<td>18.1</td>
</tr>
<tr>
<td>250</td>
<td>3.0</td>
<td>4.2</td>
<td>27.6</td>
</tr>
<tr>
<td>500</td>
<td>4.2</td>
<td>5.3</td>
<td>32.9</td>
</tr>
<tr>
<td>750</td>
<td>3.5</td>
<td>4.2</td>
<td>43.1</td>
</tr>
<tr>
<td>( x = 0.1 \beta = 0.2 \delta_i \in U(0, 1) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( a + b \delta_i \in [0.1, 0.3] )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>100</td>
<td>7.1</td>
<td>12.0</td>
<td>16.8</td>
</tr>
<tr>
<td>250</td>
<td>8.0</td>
<td>11.9</td>
<td>14.6</td>
</tr>
<tr>
<td>500</td>
<td>8.3</td>
<td>11.9</td>
<td>12.3</td>
</tr>
<tr>
<td>750</td>
<td>9.2</td>
<td>11.9</td>
<td>20.3</td>
</tr>
<tr>
<td>( x = 0.9 \beta = 0.2 \delta_i \in U(0, 1) )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( a + b \delta_i \in [0.7, 0.9] )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>100</td>
<td>3.3</td>
<td>7.0</td>
<td>9.7</td>
</tr>
<tr>
<td>250</td>
<td>3.0</td>
<td>4.2</td>
<td>17.5</td>
</tr>
<tr>
<td>500</td>
<td>3.1</td>
<td>4.2</td>
<td>23.7</td>
</tr>
<tr>
<td>750</td>
<td>3.9</td>
<td>5.3</td>
<td>29.3</td>
</tr>
</tbody>
</table>

**Legend:**
- **iter1:** total average number of iterations to get optimality.
- **time:** computing time average in seconds on a SUN station 3/50.
- **gap:** average of the percentage difference with the provably optimal value generated by SADE\(^2\).
- **iter2:** total average number of iterations related to the stopping criterion \( \mu_r - \mu_i \leq 10^{-3} \).
- **iter3:** iteration number of the best value generated by the QSG algorithm.
- **iter4:** total average number of iterations related to the stopping criterion \( ||u^{k+1} - u^k|| \leq 10^{-3} \).
right-hand side vector $b$ is varying as follows:

$$b_i = a + \beta \delta_i \quad \text{with } a, \beta \in [0, 1[ \quad \text{and} \quad \delta_i \in U(0, 1), \quad i = 1, \ldots, m,$$

while keeping $c$ and $A$ constant.

SADE$^2$ is superior to the two other schemes in all cases, both in accuracy and in computing time.

Tables 3 and 4 concern the effectiveness of the modified subgradient algorithm $G$, respectively, applied to the surrogate and the Lagrangean duals. The experiments have been carried on instances, whose coefficients of $A$ are drawn from $U(0, 1000)$ and coefficients of $c$ correlate with the columns of $A$ by the relations $c_j = (\sum_{i=1}^{m} A_{ij})/m + 500 r_j$, where $r_j \in U(0, 1)$.

Table 3 is related to the surrogate case, for which algorithm $G$ is compared with three procedures:

- a modified simplex algorithm for 0-1 variables (SIMPLEX, [23]) (which is only a reference of comparison in this case),
- a quasi-subgradient algorithm (QSG, [9, 14]),
- a descent dual procedure (DD [15, 26]).

SIMPLEX works well with moderate sizes. Algorithm $G$ and QSG are superior to the DD method when all the constraints are equally tightened ($r_i = r, \forall i = 1, \ldots, m$), with a slight advantage for algorithm $G$ when the size $n$ is growing. On the other hand, the descent dual method is the best when the coefficients $r_i$ are quite different. Moreover, the procedures $G$, QSG and DD give lower bounds than SIMPLEX in many cases.

For solving the Lagrangean dual, Table 4 compares algorithm $G$ with also three procedures:

- a modified simplex algorithm for 0-1 variables (SIMPLEX, [23]),
- a subgradient algorithm (SG, [6, 14, 35]),
- a descent dual procedure (DD, [15, 26]).

SIMPLEX is the best procedure when the size is small. The main advantage of algorithm $G$ is to provide good upper bounds within a significantly small number of iterations. SG has much smaller deviations than algorithm $G$ with not much larger computing times. As in the surrogate case, DD appears to be a promising alternative when the coefficients $r_i$ are quite different (see [16] for other comparative results with test problems of the literature).

3. Reduction

The main contribution of a reduction procedure is to decrease the size of the problem, and so to improve the efficiency of enumerative methods.

The following reduction scheme, called RAMBO ("Reduction algorithm for the multidimensional binary variables optimization problem"), is an improvement of our sequential implementation, called FPR83 and detailed in [14, 18, 19]. Other studies, carried on at the same time, exploit the same ideas in a parallel implementation denoted by PR$^2$ (Plateau and Roucairol's parallel resolution [58, 59]).
Table 3
Solving of the surrogate dual (average performance over 10 instances for each size)

<table>
<thead>
<tr>
<th>Data generation method</th>
<th>SIMPLEX</th>
<th>QSG</th>
<th>Algorithm G</th>
<th>DD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>n</td>
<td>time</td>
<td>iter.</td>
</tr>
<tr>
<td>( b_i = r \sum A_{ij} )</td>
<td>5</td>
<td>100</td>
<td>2.7</td>
<td>91</td>
</tr>
<tr>
<td>( r \in \mathbb{U}(0.2, 0.8) )</td>
<td>250</td>
<td>15.4</td>
<td>221</td>
<td>32.2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>57.8</td>
<td>432</td>
<td>67.0</td>
</tr>
<tr>
<td></td>
<td>750</td>
<td>148.1</td>
<td>661</td>
<td>97.5</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>4.5</td>
<td>88</td>
<td>33.6</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>28.1</td>
<td>227</td>
<td>67.6</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>107.1</td>
<td>445</td>
<td>108.2</td>
</tr>
<tr>
<td></td>
<td>750</td>
<td>241.2</td>
<td>672</td>
<td>167.1</td>
</tr>
<tr>
<td>( b_i = r_i \sum A_{ij} )</td>
<td>5</td>
<td>100</td>
<td>2.2</td>
<td>80</td>
</tr>
<tr>
<td>( \forall r_i \in \mathbb{U}(0.2, 0.8) )</td>
<td>250</td>
<td>11.1</td>
<td>182</td>
<td>25.3</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>54.7</td>
<td>427</td>
<td>56.4</td>
</tr>
<tr>
<td></td>
<td>750</td>
<td>106.5</td>
<td>626</td>
<td>84.1</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>4.2</td>
<td>75</td>
<td>18.2</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>24.1</td>
<td>186</td>
<td>42.9</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>87.9</td>
<td>386</td>
<td>83.9</td>
</tr>
<tr>
<td></td>
<td>750</td>
<td>220.6</td>
<td>567</td>
<td>127.0</td>
</tr>
</tbody>
</table>

- **time**: average computing in seconds on a SUN station 3/50.
- **dev**: deviation percentage = \( 100 \frac{|\hat{r}(P) - \hat{v}(P)|}{\hat{v}(P)} \), the negative sign indicates \( \hat{v}(P) < v(\hat{P}) \).
- **iter**: total average number of iterations.
- (a): the stopping criterion induces an early termination for at least one of the problems; in this case, where the initial value is not decreased after five iterations, algorithm \( G \) should be replaced by another dual technique.
Table 4
Solving of the Lagrangean dual (average performance over 10 instances for each size)

<table>
<thead>
<tr>
<th>Data generation method</th>
<th>SIMPLEX</th>
<th>SG</th>
<th>Algorithm G</th>
<th>DD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>c, A, b</td>
<td>m</td>
<td>n</td>
<td>time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$b_i = r \sum A_{ij}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$r \in U(0.2, 0.8)$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2.7</td>
<td>91</td>
<td>5.0</td>
</tr>
<tr>
<td>250</td>
<td>15.4</td>
<td>221</td>
<td>12.1</td>
<td>58</td>
</tr>
<tr>
<td>500</td>
<td>57.8</td>
<td>432</td>
<td>22.5</td>
<td>53</td>
</tr>
<tr>
<td>750</td>
<td>148.1</td>
<td>661</td>
<td>34.2</td>
<td>53</td>
</tr>
<tr>
<td>10</td>
<td>4.5</td>
<td>88</td>
<td>9.9</td>
<td>60</td>
</tr>
<tr>
<td>250</td>
<td>28.1</td>
<td>227</td>
<td>23.1</td>
<td>59</td>
</tr>
<tr>
<td>500</td>
<td>107.1</td>
<td>445</td>
<td>46.0</td>
<td>58</td>
</tr>
<tr>
<td>750</td>
<td>241.2</td>
<td>672</td>
<td>62.3</td>
<td>51</td>
</tr>
<tr>
<td>$b_i - \frac{r_i \sum A_{ij}}{\forall r_i \in U(0.2, 0.8)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>2.2</td>
<td>80</td>
<td>2.8</td>
</tr>
<tr>
<td>250</td>
<td>11.1</td>
<td>182</td>
<td>5.5</td>
<td>28</td>
</tr>
<tr>
<td>500</td>
<td>54.7</td>
<td>427</td>
<td>9.6</td>
<td>24</td>
</tr>
<tr>
<td>750</td>
<td>106.5</td>
<td>626</td>
<td>23.9</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>4.2</td>
<td>75</td>
<td>9.6</td>
<td>72</td>
</tr>
<tr>
<td>250</td>
<td>24.1</td>
<td>186</td>
<td>9.1</td>
<td>76</td>
</tr>
<tr>
<td>500</td>
<td>87.9</td>
<td>386</td>
<td>25.3</td>
<td>38</td>
</tr>
<tr>
<td>750</td>
<td>220.6</td>
<td>567</td>
<td>37.1</td>
<td>39</td>
</tr>
</tbody>
</table>

*time: average computing in seconds on a SUN station 3/50.*
*dev: deviation percentage = 100[\bar{v}(p) - v(\bar{p})]/v(\bar{p}).
*iter: total average number of iterations.*
3.1. Outlines of a new sequential reduction scheme

3.1.1. The well-stated problem

The first component of the algorithm RAMBO checks for inactive rows and to fix variables based on feasibility requirements. A well-stated formulation such that

\[ \forall i \in \{1, \ldots, m\} \quad \text{max} \quad A_{ij} \leq b_i < \sum_{j=1}^{n} A_{ij} \]

\[ \forall j \in \{1, \ldots, n\} \quad \text{column} A^j \neq \text{null vector of } \mathbb{R}^m \]

is obtained by performing the following tests:

[R1] \( \forall i \in \{1, \ldots, n\} \): if an index \( i \in \{1, \ldots, m\} \) exists such that \( A_{ij} > b_i \) then \( x_j \leftarrow 0 \);

[R2] \( \forall i \in \{1, \ldots, m\} \): if \( \sum_{j=1}^{n} A_{ij} \leq b_i \) then the constraint \( i \) can be eliminated;

[R3] \( \forall j \in \{1, \ldots, n\} \): if for any index \( i \in \{1, \ldots, m\} \) \( A_{ij} = 0 \) then \( x_j \leftarrow 1 \).

A second part tries to replace each constraint by a stronger valid inequality which has the same zero-one solution set but fewer real solutions in the unit-hypercube \( \text{Co}(B) \) [5]. This task is achieved by reducing coefficients within the matrix \( A \) and the right-hand side \( b \) with the substitution \( x_j \leftarrow 1 - x_j \) [8, 15]. The test requires \( O(mn) \) calculations but the constraint \( i \) is modified if and only if

\[ b_i > \sum_{j=1}^{n} A_{ij} - \max_{j=1, \ldots, n} \{ A_{ij} \} \]

[R4] \( \forall i \in \{1, \ldots, m\} \): let \( K_i \subseteq N = \{1,2, \ldots, n\} \) be a subset of indices \( k \) such that \( A_{ik} \geq \delta_i = \sum_{j=1}^{n} A_{ij} - b_i \).

If \( K_i \neq \emptyset \) then the constraint \( \sum_{j=1}^{n} A_{ij}x_j \leq b_i \) must be replaced by the following tighter constraint:

\[ \delta_i \sum_{j \in K_i} x_j + \sum_{j \in N - K_i} A_{ij}x_j \leq (|K_i| - 1)\delta_i + \sum_{j \in N - K_i} A_{ij} \]

3.1.2. Fixation of variables

The fixation of variables needs the knowledge of a good lower bound \( v(P) \) associated with feasible solution \( x_{\ast} \) and lies on the following basic property:

\[ \forall j \in \{1, \ldots, n\}, \forall \varepsilon \in \{0, 1\} : \text{if } v(P|x_j = \varepsilon) \leq v(P) \text{ then either } x_j = 1 - \varepsilon \text{ in any optimal solution of } (P), \text{ or } (P|x_j = 1 - \varepsilon, cx > v(P)) \text{ has no feasible solutions and so } x \text{ is optimal.} \]

Classically, a relaxation substitutes for problem \( (P) \). Given a multiplier \( u \in \mathbb{R}^n \), generated by the solving of the Lagrangean or surrogate dual of \( (P) \), we consider the associated surrogate relaxation \( (K) \):

\[ (K) \quad \max cx \text{ s.t. } ax \leq a_0, \quad x \in B \quad \text{with } a = uA \text{ and } a_0 = ub. \]

Then, by computing upper bounds on \( v(K|x_j = \varepsilon) \) with increasing complexity, we try to achieve the following inequality \( v(K|x_j = \varepsilon) \leq v(P) \) for any \( j \in \{1, \ldots, n\} \).
Continuous tests: All the continuous tests are related to Lagrangean relaxations \((LK(\lambda))\) of the knapsack \((K)\) and expressed in terms of reduced prices (an other version is given in [19], which exploits the property that the upper bound value must be integer). The optimal value \(v(LK(\lambda))\) may be stated as follows:

\[
v(LK(\lambda)) = \lambda a_0 + \sum_{j \in J_+(\lambda)} (c_j - \lambda a_j),
\]

where

\[
J_+(\lambda) = \{j | c_j - \lambda a_j > 0\} \quad \text{and} \quad J_-(\lambda) = \{j | c_j - \lambda a_j < 0\}.
\]

Then

\[
\forall j \in J_+(\lambda): v(LK(\lambda)|x_j = 1) = v(LK(\lambda)) \quad \text{and} \quad v(LK(\lambda)|x_j = 0) = v(LK(\lambda)) - (c_j - \lambda a_j);
\]

\[
\forall j \in J_-(\lambda): v(LK(\lambda)|x_j = 0) = v(LK(\lambda)) \quad \text{and} \quad v(LK(\lambda)|x_j = 1) = v(LK(\lambda)) + (c_j - \lambda a_j).
\]

An important result in [34] shows that it is sufficient to consider the set of preferred Lagrangean relaxations \(\{LK(\lambda) | \lambda \in \Lambda = \{c_j/a_j | i = 1, \ldots, n\}\}\) for the fixation of any variable \(x_j\).

The test \(V1\) takes into account only the optimal multiplier \(\lambda_\ast = c_{\ast}/a_{\ast}\), where \(i_\ast\) is the index of the basic variable at the optimum of \((\tilde{K})\), which allows to have an average linear time complexity by solving \((\tilde{K})\) [11, 12]:

\[
[V1] \forall j \in \{1, \ldots, n\}, j \neq i_\ast: \text{if } |c_j - \lambda_\ast a_j| \geq v(LK(\lambda_\ast)) - v(P), \text{ then } x_j \leftarrow \varepsilon, \text{ where } \varepsilon \text{ is equal to } 1 \text{ (respectively } 0 \text{) if } c_j - \lambda_\ast a_j > 0 \text{ (respectively } < 0).\]

The test \(V2\) considers all the preferred multipliers \(\lambda_i \in \Lambda\) and exploits for each variable \(x_j\) the comparison of the matrix \(RC(i, j) = c_j - \lambda_i a_j\) and the vector \(RV(i) = v(LK(\lambda_i)) - v(P)\). The set of fixed variables by using all the Lagrangean multipliers \(\Lambda\) is equal to the set that could be obtained by solving the 2n linear programming relaxations associated with the knapsack problems \((K|x_j = \varepsilon), j = 1, \ldots, n, \varepsilon \in \{0, 1\}\).

To reduce the time complexity \(O(n^2)\) of test \(V2\), we use a limited size subset of \(\Lambda\), corresponding to the preferred Lagrangean multipliers \(\lambda_i \in [\lambda_1, \lambda_2]\), where \(\lambda_1\) (respectively \(\lambda_2\)) is the optimal multiplier of \((K|x_i = 1)\) (respectively \((K|x_i = 0)\)).

\[
[V2] \forall j \in \{1, \ldots, n\}: \text{if an index } i \in \{1, \ldots, n\}, i \neq j, \text{ exists such that } |RC(i, j)| \geq RV(i) \text{ then } x_j \leftarrow \varepsilon, \text{ where } \varepsilon \text{ is equal to } 1 \text{ (respectively } 0 \text{) if } RC(i, j) > 0 \text{ (respectively } < 0).\]

The separation on the optimal basic fractional variable \(x_{i_\ast}\) allows to the strengthening of the upper bound \(v(K|x_j = \varepsilon)\) by max \(\{v(K|x_i = 1, x_j = \varepsilon), v(K|x_i = 0, x_j = \varepsilon)\}\). By denoting \(RV_0\) and \(RV_1\) two vectors such that

\[
RV_1(i) = v(LK(\lambda_i)|x_{i_\ast} = 1) - v(P)
\]

\[
= \begin{cases} V(i) - |RC(i, i_\ast)|, & \text{if } RC(i, i_\ast) < 0, \\ V(i), & \text{otherwise}, \end{cases} \quad i = 1, \ldots, n
\]
\[ RV_0(i) = v(I.K(\lambda_i) | x_i^* = 0) - v(P) \]
\[ = \begin{cases} V(i) - RC(i, i^*), & \text{if } RC(i, i^*) > 0, \\ V(i), & \text{otherwise} \end{cases} \]
\[ i = 1, \ldots, n \]

the test \( V_3 \) may be stated as follows:

\[ \text{[V3] } \forall j \in \{1, \ldots, n\}, j \neq i^*: \text{if two indices } i_0 \text{ and } i_1 \text{ exist such that } RC(i_0, j) \text{ and } RC(i_1, j) \text{ have the same sign, and satisfy } |RC(i_0, j)| \geq RV_0(i_0) \text{ and } |RC(i_1, j)| \geq RV_1(i_1), \text{ then } x_j \leftarrow \varepsilon, \text{ where } \varepsilon \text{ is equal to } 1 \text{ (respectively } 0 \text{) if the sign is positive (respectively negative).} \]

In practice, the test \( V_3 \) is carried on only for two indices \( i_0^* \), \( i_1^* \) associated, respectively, with the optimal basic variables \( x_{i_0}^*, x_{i_1}^* \) of the two relaxations \((K|x_i^* = 0, x_j = \varepsilon)\) and \((K|x_i^* = 1, x_j = \varepsilon)\).

Another way to improve the upper bound \( v(K|x_j = \varepsilon) \) is by using binary relations induced by the additivity of the reduced costs \([11, 38, 56]\). This property allows for easy computation of the sets \( X_{j^0} \) and \( X_{j^1} \) associated with conditional fixations under the hypothesis \( x_j = \varepsilon \), and then the tighter upper bound \( v(LK(\lambda_{i(j)}), x_j = \varepsilon, x_k = 0, \forall k \in X_{j^0}, x_k = 1, \forall k \in X_{j^1}) \), where \( \lambda_{i(j)} \) is a selected Lagrangean multiplier.

\[ \text{[V4] } \forall j \in \{1, \ldots, n\}: \]
- select a preferred multiplier \( \lambda_{i(j)} \) such that
  \[ RV(i(j)) - |RC(i(j), j)| = \min \{V(i) - |RC(i, j)| \mid i = 1, \ldots, n\}; \]
- hypothesis of fixation:
  \( x_j = \varepsilon \) with \( \varepsilon \) equal to 0 (respectively 1) if \( RC(i, j) > 0 \) (respectively \( < 0 \));
- conditional fixations:
  construct the sets \( X_{j^0}, X_{j^1} \) from a subset of the \( n - 1 \) other preferred Lagrangean relaxations \((LK(\lambda_i), i \neq i(j)):\)
  if \( v(LK(\lambda_i), x_j = \varepsilon, x_l = \eta, x_k = 0, \forall k \in X_{j^0}, x_k = 1, \forall k \in X_{j^1}) \leq v(P) \) then \( X_{j^0} = X_{j^0} \cup \{l\} \) if \( \eta = 1 \) or \( X_{j^1} = X_{j^1} \cup \{l\} \) if \( \eta = 0; \)
- feasibility:
  if \( (P|x_k = 0 \forall k \in X_{j^0}, x_k = 1 \forall k \in X_{j^1}) \) has no feasible solutions then \( x_j \leftarrow 1 - \varepsilon; \)
- additive penalties:
  if \( v(LK(\lambda_{i(j)}), x_j = \varepsilon, x_k = 0 \forall k \in X_{j^0}, x_k = 1 \forall k \in X_{j^1}) \leq v(P) \) then \( x_j \leftarrow 1 - \varepsilon. \)

\textbf{Integer test:} The last test \( V_5 \) consists in solving the 0-1 knapsack problems \((K|x_j = \varepsilon)\), but only for the variables \( x_j \) such that one of the two inequalities of test \( V_3 \) is satisfied.

\[ \text{[V5] Given } j \in \{1, \ldots, n\} \text{ and } \varepsilon \in \{0, 1\}: \text{if } v(K|x_j = \varepsilon) \leq v(P) \text{ then } x_j \leftarrow 1 - \varepsilon. \]
3.1.3. Elimination of constraints

All the tests are based on the following idea:

\[ \text{If } \max \{ A_k x | x \in E_k \} \leq b_k \text{ then the constraint } A_k x \leq b_k \text{ can be eliminated; it means that it is possible to display a subset } E_k \subseteq B, \text{ both including the feasible domain defined by all the other constraints and included in the half-space } \{ x \in B | A_k x \leq b_k \}. \]

The set \( E_k \) is usually chosen as the intersection of \( B \) with one half-space of \( \mathbb{R}^n \):

(i) \( E_k = \{ x \in B | A_p x \leq b_p, p \neq k \} \),

(ii) \( E_k = \{ x \in B | u A x \leq ub, u \in \mathbb{R}^n, u_k = 0 \} \),

where the multiplier \( u \) is generated by any dual technique and the index \( p \) is such that \( u_p = \max \{ u_i | i = 1, \ldots, m \} \).

So elimination of constraints needs the solving of the 0-1 knapsack problem (\( K_k \)): \[
(K_k) \quad \max A_k x \text{ s.t. } ax \leq a_0, x \in B \text{ where } ax \leq a_0 \text{ is defined as in (i) or (ii)}. \]

Continuous tests: The 0-1 knapsack problem (\( K_k \)) is replaced by linear programming relaxations. The order of implementation corresponds to tighter and tighter upper bounds and increasing complexity.

- [C1] If \( \lceil v(K_k) \rceil \leq b_k \) then the constraint \( k \) can be eliminated.

- [C2] Let \( x_{*,*} \) be the optimal basic variable of (\( K_k \));

\[ \text{if } \alpha = \max \{ \lceil v(K_k | x_{*,*} = 0) \rceil, \lceil v(K_k | x_{*,*} = 1) \rceil \} \leq b_k \text{ then the constraint } k \text{ can be eliminated}. \]

When \( \alpha > b_k \), but \( \lceil v(K_k | x_{*,*} = 1 - \varepsilon) \rceil \leq b_k \) with \( \varepsilon \in \{ 0, 1 \} \), the test C3 is performed:

- [C3] Let \( x_{*,(b)} \) be the optimal basic variable of (\( K_k | x_{*,*} = \varepsilon \));

\[ \text{if } \max \{ \lceil v(K_k | x_{*,*} = \varepsilon, x_{*,(b)} = 0) \rceil, \lceil v(K_k | x_{*,*} = \varepsilon, x_{*,(b)} = 1) \rceil \} \leq b_k \text{ then the constraint } k \text{ can be eliminated}. \]

In practice, the elimination procedure scans the constraints \( k \) such that \( u_k = 0 \). In case (i) the index \( p \) is selected such that \[ 10 \] :

\[ \frac{b_p}{\sum_{j=1}^{n} A_{pj} \cos^2(A_{kj}, A_{pk})} = \min \left\{ \frac{b_q}{\sum_{j=1}^{n} A_{qi} \cos^2(A_{kj}, A_{qk})} | q \neq k \right\}. \]

Integer test:

- [C4] If \( v(K_k) \leq b_k \) then the constraint \( k \) can be eliminated.

3.1.4. Description of the reduction procedure

All the above-mentioned tests are included in different components of a reduction procedure called RAMBO.

- WELLSTAT: the tests [R1 → 4] are performed in a sequential fashion to generate a well-stated problem. Moreover the incumbent problem is solved exactly by an implicit enumeration procedure if the number of variables is less than or equal to 10, or by the FPK79 method [12] if the number of constraints is reduced to one.

- KILLVAR1: performs the continuous tests [V1 → 4].
KILLVAR2: performs the integer test [V5].
KILLCONT1: performs the continuous tests [C1 → 3].
KILLCONT2: performs the integer test [C4].

The components are chained so as to favour continuous tests with regard to integer ones. The boolean variables reduction 1 and reduction 2 are such that RAMBO runs while reduction occurs. Procedures AGNES and MULTIPLIER provide lower and upper bounds and dual multipliers. The procedure MULTIPLIER switches RAMBO on to the best dual procedure according to the material presented in Section 2.

**Reduction algorithm RAMBO**

```
call WELLSTAT;
compute a structural multiplier:
    \[ u_i = \left( \frac{\sum_{j=1}^{n} A_{ij} - b_i}{\sum_{j=1}^{n} A_{ij}} \right) \]
    \( i = 1, \ldots, m \)
call AGNES(\( u \));
reduction 1 := false;
reduction 2 := false;
repeat
    repeat
        call MULTIPLIER(\( u \));
        call AGNES(\( u \));
        call KILLVAR1(\( u \), reduction1);
        call KILLCONT1(\( u \), reduction1);
    until reduction 1
    call KILLVAR2(\( u \), reduction2);
    call KILLCONT2(\( u \), reduction2);
until reduction2
```

3.2. **Computational results**

Tables 5 and 6 are related to test problems of the literature and detail the number of fixed variables and eliminated constraints. This new reduction scheme improves previous results obtained with the earlier version FPR83 [19].

The reduction algorithm RAMBO has been further tested on different groups of large-scale problems in order to identify its capabilities. Two types of probability distribution are used to generate the coefficients of matrix \( A \):
- the uniform \([0, \text{max}]\) distribution;
- the distribution associated with a continuous density function \( f_\alpha \), where \( f_\alpha \) is defined as follows:

\[
f_\alpha(x) = \begin{cases} 
\frac{\lambda - \theta}{\alpha} x & \text{if } 0 \leq x \leq \alpha, \\
0 & \text{if } \alpha < x < \text{max}, \\
\frac{\text{max} - \alpha}{\alpha} (\text{max} - x) & \text{if } \alpha \leq x \leq \text{max}, \\
0 & \text{otherwise.}
\end{cases}
\]
<table>
<thead>
<tr>
<th>Test problem</th>
<th>Initial size</th>
<th>Reduction tests</th>
<th>Reduced size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m</td>
<td>n</td>
<td>R1</td>
</tr>
<tr>
<td>Pet [54]</td>
<td>10</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>28</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>39</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>50</td>
<td>14</td>
</tr>
<tr>
<td>FP [18]</td>
<td>4</td>
<td>27</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>34</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>29</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>37</td>
<td>2</td>
</tr>
<tr>
<td>F [13, 18]</td>
<td>10</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>ST [62]</td>
<td>30</td>
<td>60</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>WN [68]</td>
<td>2</td>
<td>28</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>105</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>105</td>
<td></td>
</tr>
<tr>
<td>MM [49]</td>
<td>6</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

R1, 2, 3: well-stated problem tests.
R4: coefficient reduction.
V1, 2, 3, 4, 5: fixation of variables.
C1, 2, 3, 4: elimination of redundant constraints.
(a): the number of remaining variables is less than or equal to 10 or the reduced value of $m$ is 1; the reduced problem is then solved by an implicit enumeration procedure.

The parameters $\lambda$ and $\theta$ are chosen such that $\text{Prob}(x \leq \alpha) = p$, where probability $p \in [0, 1]$. This random distribution is simulated by the reject method.

In the two cases, the entries of the objective function $c$ and the right-hand side $b$ correlate with matrix $A$ as follows:

$$c_j = \left( \sum_{i=1}^{m} A_{ij} \right)/m + 500 \ast r_j, \text{ where } j = 1, \ldots, n \text{ and } r_j \in U(0, 1),$$

$$b_i = r_i \sum_{j=1}^{n} A_{ij}, \text{ where } i = 1, \ldots, m \text{ and } r_i \in U(0, 1).$$
Table 6
Reduction with test problems of the literature

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Initial size</th>
<th>Reduction tests</th>
<th>Reduced size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS [64] 50</td>
<td>5 30</td>
<td>1 17 3 (a)</td>
<td>1 6 19 1 (a)</td>
</tr>
<tr>
<td></td>
<td>1 14 20 (a)</td>
<td>1 17 29 2 (a)</td>
<td>1 6 30 (a)</td>
</tr>
<tr>
<td></td>
<td>3 26 (a)</td>
<td>1 17 38 1 (a)</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>5 27 (a)</td>
<td>4 1 38 6 (a)</td>
<td>5 6 2 4</td>
</tr>
<tr>
<td></td>
<td>1 7 27 2 1 1 (a)</td>
<td>2 36 2 4 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 8 27 2 1 1 (a)</td>
<td>5 2 46 5 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 47 3 (a)</td>
<td>2 11 51 (a)</td>
<td>4 79 49 (a)</td>
</tr>
<tr>
<td></td>
<td>1 2 16 1 4 (a)</td>
<td>2 8 51 2 4 1 3 12 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2 58 3 (a)</td>
<td>4 1 60 2 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 7 62 3 7 (a)</td>
<td>3 12 62 1 6 2 11 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 71 (a)</td>
<td>1 66 1 1 1 3 12 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 83 (a)</td>
<td>3 82 (a)</td>
<td>3 84 (a)</td>
</tr>
<tr>
<td></td>
<td>4 2 85 (a)</td>
<td>5 90</td>
<td>3 12 73 1 2 4 (a)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test problem</th>
<th>Initial size</th>
<th>Reduction tests</th>
<th>Reduced size</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS [64] 60</td>
<td>5 30</td>
<td>1 17 3 (a)</td>
<td>1 6 19 1 (a)</td>
</tr>
<tr>
<td></td>
<td>1 14 20 (a)</td>
<td>1 17 29 2 (a)</td>
<td>1 6 30 (a)</td>
</tr>
<tr>
<td></td>
<td>3 26 (a)</td>
<td>1 17 38 1 (a)</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td></td>
<td>5 27 (a)</td>
<td>4 1 38 6 (a)</td>
<td>5 6 2 4</td>
</tr>
<tr>
<td></td>
<td>1 7 27 2 1 1 (a)</td>
<td>2 36 2 4 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 8 27 2 1 1 (a)</td>
<td>5 2 46 5 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 47 3 (a)</td>
<td>2 11 51 (a)</td>
<td>4 79 49 (a)</td>
</tr>
<tr>
<td></td>
<td>1 2 16 1 4 (a)</td>
<td>2 8 51 2 4 1 3 12 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2 58 3 (a)</td>
<td>4 1 60 2 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 7 62 3 7 (a)</td>
<td>3 12 62 1 6 2 11 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 71 (a)</td>
<td>1 66 1 1 1 3 12 (a)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3 83 (a)</td>
<td>3 82 (a)</td>
<td>3 84 (a)</td>
</tr>
<tr>
<td></td>
<td>4 2 85 (a)</td>
<td>5 90</td>
<td>3 12 73 1 2 4 (a)</td>
</tr>
</tbody>
</table>

R1, 2, 3: well-stated problem tests.
R4: coefficient reduction.
V1, 2, 3, 4, 5: fixation of variables.
C1, 2, 3, 4: elimination of redundant constraints.
(a): the number of remaining variables is less than or equal to 10 or the reduced value of m is 1; the reduced problem is then solved by an implicit enumeration procedure.

Table 7 shows that the effectiveness of the reduction procedure is slightly decreasing when the matrix data are closer to one another, the best results being obtained with the uniform distribution.

Table 8 reports some indications about the gap $\bar{v}(P) - \bar{q}(P)$, where the upper bound $\bar{v}(P)$ is provided by the best linear programming relaxation. This gap could be tightened by solving the surrogate dual instead of the linear relaxation. These results confirm that uniform, randomly generated instances are among the easiest to solve.
Table 7
Reduction with randomly generated problems ((a)verage performance over 10 instances for each size)

<table>
<thead>
<tr>
<th>Data generation method</th>
<th>Performance average of reduction tests (%)</th>
<th>Size reduction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>c, A, b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U(0, 1000)</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>12</td>
</tr>
<tr>
<td>h(0, 1000)</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>x = 100</td>
<td>250</td>
<td>12</td>
</tr>
<tr>
<td>p = 0.9</td>
<td>500</td>
<td>250</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.1</td>
</tr>
<tr>
<td>f_0(0, 1000)</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>x = 50</td>
<td>250</td>
<td>0.2</td>
</tr>
<tr>
<td>p = 0.9</td>
<td>500</td>
<td>45.2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>250</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>21.4</td>
</tr>
</tbody>
</table>

R1, 2, 3: well-stated problem tests (fixation of variables or elimination of constraints).
R4: matrix coefficient reduction.
V1, 2, 3, 4, 5: fixation of variables.
C1, 2, 3, 4: elimination of redundant constraints.

4. Conclusion

Efficient tools as heuristics, new algorithms for obtaining surrogate bounds and rules for reducing problem size are developed and compared with previous ones. We discuss their use and implementation in a preprocessing phase. These tools are shown to be very effective through a lot of computational experiments. For the large-scale instances, we hope that the preprocessing procedure could alter significantly the effort taken to solve these problems.

Acknowledgement

The authors are indebted to Professor Pierre Hansen for providing the seminal ideas of the reduction development.
Table 8
Lower and upper bounds on the optimal value $v(P)$

<table>
<thead>
<tr>
<th>Data generation method</th>
<th>$\bar{v}(P)$</th>
<th>$v(P)$</th>
<th>$100(\bar{v}(P)/v(P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>min</td>
<td>mean</td>
<td>max</td>
</tr>
<tr>
<td>$c, A, b$</td>
<td>$m$</td>
<td>$n$</td>
<td></td>
</tr>
<tr>
<td>$U(0, 1000)$</td>
<td>5</td>
<td>100</td>
<td>96</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
<td>113</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>$f_3(0, 1000)$</td>
<td>5</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>$\alpha = 100$</td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>$\rho = 0.9$</td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
<td>269</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>$h(0, 1000)$</td>
<td>5</td>
<td>100</td>
<td>38</td>
</tr>
<tr>
<td>$\alpha = 50$</td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td>$\rho = 0.9$</td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>100</td>
<td>280</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>250</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
</tbody>
</table>

$\bar{v}(P)$: upper bound provided by SIMPLEX procedure.
$v(P)$: lower bound provided by AGNES procedure.

References


A. Fréville, Heuristiques et réduction pour les programmes mathématiques en variables 0-1 à contraintes d'inégalité, Thèse 3ème cycle, Université de Lille 1 (1983).


