# Mathematical programming approach to tighten a Big-$M$ formulation

Alejandro Crema
*Escuela de Computación, Facultad de Ciencias,*
*Universidad Central de Venezuela.*
*E-mail: alejandro.crema@ciens.ucv.ve*

August 2014

**Abstract**

In this paper we present a mathematical programming approach to tighten a Big-$M$ formulation ($P_M$) of a Mixed Integer Problem with Logical Implications ($P$). If $M_0$ is a valid vector (the optimal solutions of $P$ belong to the feasible solutions set of $P_{M_0}$) our procedures find a valid vector $M$ such that $M \leq M_0$. As a consequence, the upper bounds generated by using the linear relaxations of $P_M$ are stronger when $M \neq M_0$. As a result, an standard branch and cut algorithm can be used to try to solve $P$ by using $P_M$ with a high probability of obtaining better results than using $P_{M_0}$. Computational results are presented in order to show that our procedures are very promising.

**Keywords**: Integer Programming, Big-$M$ formulation

## 1 Introduction

Many real problems can be formulated as a Mixed Integer Programming (MIP) problem with Logical Implications (LI) as follows:

$$(P)\ \ max\ \ c^t x + d^t y + f^t z\ \ s.t.$$

$$(x, y, z) \in X \times \{0,1\}^n \times \Re^n$$

$$Ax + By + Cz \leq b$$

$$(LI)\ \ y_i = 0 \Rightarrow z_i \leq 0\ \ (i = 1, \cdots, n)$$

where $c, d, f$ and $b$ are vectors, $A, B$ and $C$ are matrices of conformable dimensions and either $X = \{0,1\}^p$ or $X = \Re^p$ according to the practical problem.

Some examples of problems of this type are: the maximun feasible subsystem problem ([1]), the mixed integer classification problem ([13]), the quadratic multidimensional knapsack problem ([14]), several scheduling problems ([5],[10]) and many planning and scheduling problems in the process industry ([9]). Another examples, the multidimensional knapsack problem with violations and its continous version will be presented in our computational experience.

A few words about our notation: If $\bullet$ is an optimization problem then $F(\bullet)$ is its feasible solutions set, $v(\bullet)$ is its optimal value (if it exists) and $Opt(\bullet)$ is its optimal solutions set.

In order to simplify the exposition we assume that $F(P)$ and $Opt(P)$ are nonempty.

Let $M \in \Re^n$. The Big-$M$ formulation of $P$ is defined as follows:

$$(P_M) \ \ max \ \ c^t x + d^t y + f^t z \ \ s.t.$$

$$(x, y, z) \in X \times \{0,1\}^n \times \Re^n$$

$$Ax + By + Cz \leq b$$

$$z_i \leq M_i y_i \ \ (i = 1, \cdots, n)$$

In practice $M$ is usually selected in such a manner that $F(P) = F(P_M)$ allowing us to try to solve the problem by using an standard branch and cut algorithm.

Large values for $M$ can easily lead to numerical problems and to weak linear relaxationes ([11],[3]). Some integer programming solvers , for example ILOG-Cplex 12.4 ([4]), have the option to avoid the Big-$M$ formulation by using the original formulation of $P$ managing the LI by using the branches $y_i = 1$ and $z_i \leq 0$. However, linear relaxations of the original formulation remove the LI and then may be potentially weaker than the less numerically stable Big-$M$ formulation ([11]). Also, if $P$ has an special structure we can avoid the use of the Big-$M$ formulation by using the Combinatorial Benders' Cuts ([6]).

For many problems an appropriate vector $M$ may be easily selected, in order to try to avoid numerical problems, either from a carefull inspection of the data or by solving appropriate auxiliary MIP problems. However, even if the numerical problems disappear the weakness of the linear relaxation could remain.

We say the $M$ is a valid vector for $P$ if and only if $Opt(P) \subseteq F(P_M)$. Again, if $M$ is a valid vector we can use an standard algorithm to try to solve $P$ by solving $P_M$.

In this paper we design algorithms based on a Mathematical Programming approach to tighten a Big-$M$ formulation. If $M_0$ is a valid vector our algorithm finds a valid vector $M$ such that $M \leq M_0$. Since $M \leq M_0$ then $F(P_M) \subseteq F(P_{M_0})$ and the upper bounds generated by using the new linear relaxations are stronger when $M \neq M_0$. Therefore, an standard branch and cut algorithm may be used to try to solve $P$ by using $P_M$ with a high probability of obtaining better performance than using $P_{M_0}$.

In Section 2 we present the theoretical results and the algorithms. Our computational experience will be presented in Section 3. A summary and further extensions are presented in Section 4.

# 2 Theoretical results and the algorithms

Two elementary reamarks are: (i) if $M^1$ is a valid vector and $M^2 \geq M^1$ then $M^2$ is a valid vector and (ii) $M$ may be a valid vector with $F(P_M) \subset F(P)$.

Let $k \in \{1, \cdots, n\}$. Let $P_k$ be a MIP problem defined as follows:

$$(P_k) \;\; max \;\; z_k \;\; s.t. \;\; (x, y, z) \in Opt(P)$$

**Lemma 1** *Let $M_k^* = v(P_k)\ \forall k$, then: (i) $M^*$ is a valid vector and (ii) $M$ is a valid vector if and only if $M \geq M^*$.*

**Proof:**

(i) Let $(x, y, z) \in Opt(P)$. From the construction of $P_k$ we have that $z_k \leq v(P_k)$. If $y_k = 1$ then $z_k \leq v(P_k)y_k$ and if $y_k = 0$ then $z_k \leq 0$ and $z_k \leq 0 = v(P_k)y_k$. Therefore $(x, y, z) \in F(P_{M^*})$ and $M^*$ is a valid vector.

(ii.1) If $M \geq M^*$ then $M$ is a valid vector because of $M^*$ is a valid vector.

(ii.2) Let $M$ be a valid vector. Since $M$ is a valid vector we have that $z_k \leq M_k y_k \leq M_k\ \forall k$, $\forall (x, y, z) \in Opt(P)$. Therefore $M_k^* = v(P_k) \leq M_k\ \forall k$ and $M^* \leq M$.•

For many problems a valid vector $M_0$ may be easily selected and the Big-$M_0$ formulation of $P_k$ is:

$$(P_k(M_0)) \;\; max \;\; z_k \;\; s.t. \;\; (x, y, z) \in F(P_{M_0}), \;\; c^t x + d^t y + f^t z \geq v(P)$$

We do not have $v(P)$ in advanced and then from the practical point of view $P_k(M_0)$ is not an useful problem to find new valid vectors. Therefore, we consider a relaxation. Let $w$ be a lower bound for $v(P)$ (possibly obtained with an

heuristic algorithm) and let $P_k(M_0, w)$ be an auxiliary MIP problem defined as follows:

$$(P_k(M_0,w))\ \ max\ \ z_k\ \ s.t.\ \ (x,y,z) \in F(P_{M_0}),\ \ c^t x + d^t y + f^t z \geq w$$

**Lemma 2** *Let $M_k = v(P_k(M_0, w))\ \forall k$ then $M$ is a valid vector with $M \leq M_0$.*

**Proof:**

Since $P_k(M_0, w)$ is a relaxation of $P_k(M_0)$ it follows that $v(P_k(M_0, w)) \geq v(P_k(M_0) = v(P_k) = M_k^*\ \forall k$ and then we have $M \geq M^*$. Therefore, $M$ is a valid vector. Let $(x, y, z)$ be an optimal solution for $P_k(M_0, w)$ then we have $v(P_k(M_0, w)) = z_k \leq M_{0k} y_k \leq M_{0k}$ and then $M_k \leq M_{0k}\ \forall k$ and $M \leq M_0$.•

In order to apply lemma 2 we need to solve $P_k(M_0, w)\ \forall k$ and this may be a hard computational task. Therefore, in our computational experience we consider a new relaxed value as follows: instead of solving $P_k(M_0, w)$ we use a partial execution of the standard branch and cut algorithm of ILOG-Cplex 12.4, with time limit (the Cplex's parameter *timelimit* is set to *tmax*) and then, if $\widehat{z}_k$ is the best upper bound obtained (corresponding to the field *bestobjval* of the Cplex.Solution Property) we have $v(P_k(M_0, w)) \leq \widehat{z}_k \leq M_{0k}$. Again, if $\widehat{M_k} = \widehat{z}_k\ \forall k$ then $\widehat{M}$ is a valid vector with $\widehat{M} \leq M_0$.

Let $M_0$ be a valid vector to formulate $P$. Below we present an algorithm based on lemmas 1 and 2 designed to obtain a valid vector $M$ with $M \leq M_0$.

**Algorithm with a mixed integer programming approach (AMIP)**:

Let $M^1 = M_0$ and let $timelimit = tmax$ ($tmax$ is the limit time to solve aproximately $P_k(M, w)\ \forall M\ \forall k$).

**for** $k = 1$ **until** $n$
  solve aproximately $P_k(M^k, w)$ to obtain $\widehat{z}_k$.
  let $M_k^k = \widehat{z}_k$.
  let $M^{k+1} = M^k$.
**endfor**

**AMIP** finds $M^1, \cdots, M^n$ such that $M^1 \geq \cdots \geq M^n$ and then we use $M = M^n$ to solve $P$. Note that instead of use $M_0$ for all $k$, we use $M^1 = M_0$ to obtain $M^2$, $M^2$ to obtain $M^3$ and so on.

In [9] a similar approach was presented without the constraint $c^t x + d^t y + f^t z \geq w$, therefore with that procedure the final valid vector is greater or equal than the final valid vector obtained with our algorithm. The procedure in [9] is presented as a sequence of tailor-made auxiliary problems in the process industry, however its generalization to any problem is very easy. Our approach is

general and we do not use the structure of the problem being solved.

In [8] the author presents, without computational experience, an effort to design a Branch and Bound algorithm including a dynamic updating of the $M$ vector. In [2] a similar approach to obtain $M$ from an initial $M_0$ by using linear problems was developed for the Quadratic Knapsack problem (QKP). In our computational experience we consider the use of $\overline{P_k(M_0, w)}$, the linear relaxation of $P_k(M_0, w)$, as follows:

**Algorithm with a linear programming approach (ALP)**:

**while** $\delta(s) > \epsilon$ and $s < smax$.
  let $\delta_0 = 0$.
  **for** $k = 1$ **until** $n$
    let $M = M^r$, $\bar{z}_k = v(\overline{P_k(M, w)})$, $\delta_0 = \delta_0 + M_k - \bar{z}_k$
    and $M_k = \bar{z}_k$.
    let $r = r + 1$ and $M^r = M$.
  **endfor**
  let $\delta(s) = \delta_0$ and $s = s + 1$
**endwhile**

Let $M^1 = M_0$, $\delta(0) = \infty$, $s = 0$ and $r = 1$. Let $\epsilon > 0$ and $smax \geq 1$ be parameters to stop the algorithm.

**ALP** generates $M^1, \cdots, M^r$ such that: $M^{i+1} \leq M^i \ \ \forall i$ and $M^i$ is a valid vector $\forall i$. Also, generates $\delta(0), \cdots, \delta(s)$ such that: $\delta(i) \leq \delta(i+1) \ \ \forall i$ and $\delta(s) \leq \epsilon$ if $s$ is large enought. If we define $smax = \infty$ the **ALP** stops if $\delta(s) \leq \epsilon$. **ALP** may be seen as a generalization of the procedure presented in [2] because we consider $P$ instead of an specific problem ($QKP$). Also, the procedure presented in [2] uses only one step ($smax = 1$).

# 3 Computational results

Our algorithms have been performed on a personal computer as follows: Intel(R)CoreTM2Duo CPU, T5900, with 4.00 GB Ram, 2.20 GHz and Windows 8.1 Operating System. All the instances have been processed trought the commercial ILOG-Cplex 12.4 solver ([4] ).

Realistically, you cannot expect our approach works well for all problems written as $P$. For some problems the final $M$ is very close to $M_0$ and sometimes even if the difference is large the impact in the performance of the branch and cut algorithm used to solve $P_M$ do not compensate the time required to find $M$. For example, this was the case, at least with the data used so far, for the maximum feasible subsystem problem ([1]). Nevertheless our approach proved

to have merits in handling some problems.

The problems considered in our experimentation are: the multidimensional knapsack problem with violations ($KPmv$),the continuous multidimensional knapsack problem with violations ($CKPmv$) and the quadratic multidimensional knapsack problem ($QKPm$).

The three selected problems may be rewritten as $P$ after some variables changes and algebraic manipulations. Instead of that we present in each case the corresponding problems $P_M$ and $P_k(M, w)$ by using the natural formulation. A very simple procedure to obtain $M_0$ with $F(P) = F(P_{M_0})$ is presented for each problem. An extended formulation for $KPmv$ and $CKPmv$ are presented avoiding the use of the Big-$M$.

For the problems considered in the computational experience the lower bound value for $v(P)$ that we need to define $P_k(M, w)$ was obtained either by using an appropriate heuristic for $P$ or by solving approximately $P_{M_0}$ with a time limit.

The experiments were designed to show that our approach is very promising. However, we do not claim that our algorithms are the best for the selected problems. The computational experience was designed in order to try to show that our algorithms works well for several problems as follows: we consider some problems and their Big-$M_0$ and extended formulations and then we apply differents approaches as follows:

(i) we solve $P_{M_0}$ directly (the 0-Strategy),

(ii) we obtain $w$ with an heuristic and then we apply **AMIP** to find $M$, a valid vector with $M \leq M_0$, and we solve $P_M$ (the **AMIP**-Strategy),

(iii) we obtain $w$ with an heuristic and then we apply **ALP** to find $M$, a valid vector with $M \leq M_0$, and we solve $P_M$ (the **ALP**-Strategy),

(iv) we solve the extended formulation directly (the **E**-Strategy)

and finally we compare the performance of the differents strategies.

In the tables presented in Appendix A *mipgap* and *timelimit* are CPLEX's parameters, *tmax* is the **AMIP**'s parameter, *smax* and $\epsilon$ are the **ALP**'s parameters and *preproctime* refers to the preprocessing time: the time used to obtain $M_0$ plus *heutime*, the time to find $w$ (the lower bound value for $v(P)$), plus the time to obtain the final $M$. Problems were solved with differents environments defined with the CPLEX's parameters (*timelimit* and *mipgap*) and our parameters (*tmax*, *smax*, $\epsilon$ and *heutime*). When a parameter has several options some cells in the tables have the corresponding outputs.

Tables in appendix A report:

P: an instance identifier,

gap($M_0$)(%), gapALP(%), gapAMIP(%) and gapE(%): the percentage gap obtained with the 0-strategy, the **ALP**-strategy, the **AMIP**-strategy and the $E$-strategy (the gap's values are obtained at the times presented in the tables),

iterALP: the number of iterations when we use the **ALP**-strategy,

DifALP(%) and DifAMIP(%): the average of the relative differences between $M_{0i}$ and $M_i$ when we use the **ALP**-strategy and the **AMIP**-strategy (the Dif's values are computed as follows in each case: Dif $= 100 \times \frac{1}{m} \sum_{i=1}^{m} (\frac{M_{0i} - M_i}{M_{0i}})$),

$\Delta$gapALP-$M_0$(%) and $\Delta$gapAMIP-$M_0$ (%): the relative difference between the gaps computed as follows:

$$\Delta\text{gapALP-}M_0(\%) = 100 \times \frac{(\text{gap}_{(M_0)} - \text{gapALP})}{\text{gap}_{(M_0)}}$$

$$\Delta\text{gapAMIP-}M_0(\%) = 100 \times \frac{(\text{gap}_{(M_0)} - \text{gapAMIP})}{\text{gap}_{(M_0)}}$$

(the $\Delta$ values are computed when the times are the same),

t($M_0$), totaltALP(sec.), totaltAMIP(sec.) and t($E$): the total time with the 0-strategy, the **ALP**-strategy and the **AMIP**-strategy (the preprocessing time plus the time solving the $P_M$ problem) and the time with the $E$-strategy,

$\Delta$timeALP-$M_0$(%) and $\Delta$timeAMIP-$M_0$(%): the relative difference between times when the corresponding gap values are the same, computed as follows:

$$\Delta\text{timeALP-}M_0(\%) = \frac{(\text{t}_{(M_0)} - \text{totaltALP(sec.)})}{\text{t}_{(M_0)}}$$

$$\Delta\text{timeAMIP-}M_0(\%) = \frac{(\text{t}_{(M_0)} - \text{totaltAMIP(sec.)})}{\text{t}_{(M_0)}},$$

nodes($M_0$) and nodesE: the nodes generated with the 0-strategy and the $E$-strategy.

When the times do not appear in a table is due to that the time limit was reached without checking the requested tolerance. On the other hand if the gaps are not listed in the tables is due to that the required tolerance was achieved.

### 3.1 Computational results for the multidimensional knapsack problem with violations and its continuous version

The multidimensional knapsack problem with violations ($KPmv$) is defined as follows:

$$(KPmv)\ \ max\ \sum_{j=1}^{n} c_j x_j\ \ s.t.$$

at least $k$ of the following constraints must be satisfied

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i\ \ (i = 1, \cdots, m)$$

$$x \in \{0,1\}^n$$

where $a_{ij} \geq 0$, $c_j \geq 0$, $1 \leq k < m$ and $b_i \geq 0$ for all $i, j$.

The multidimensional knapsack problem with violations ($KPmv$) is defined from the multidimensional knapsack problem ($KPm$) ([7]) following the same idea to define the $k$-violation linear programming problem from a linear programming problem involving a set of covering constraints ([12]).

The Big-$M$ formulation of $KPmv$ is defined as follows:

$$(KPmv_M)\ \ max\ \sum_{j=1}^{n} c_j x_j\ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i + (1 - y_i) M_i\ \ (i = 1, \cdots, m)$$

$$\sum_{i=1}^{m} y_i \geq k,\ \ x \in \{0,1\}^n,\ \ y \in \{0,1,\}^m$$

If $M_{0i} = \sum_{j=1}^{n} a_{ij} - b_i$ is very easy to see that $F(KPmv_{M_0}) = F(KPmv)$

Problem $KPmv_k(M, w)$ is defined as follows:

$$(KPmv_k(M,w))\ \ max\ \ \sum_{j=1}^{n} a_{kj} x_j - b_k\ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i + (1 - y_i) M_i\ \ (i = 1, \cdots, m)$$

$$\sum_{i=1}^{m} y_i \geq k$$

$$\sum_{j=1}^{n} c_j x_j \geq w$$

$$x \in \{0,1\}^n, \;\; y \in \{0,1\}^m$$

An extended formulation of $KPmv$, avoiding the Big-$M$ formulation, may be defined as follows:

$$(KPmvE) \;\; max \;\; \sum_{j=1}^{n} c_j x_j \;\; s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j^i \leq b_i y_i \;\; (i = 1, \cdots, m)$$

$$0 \leq x_j - x_j^i \leq 1 - y_i \;\; (i = 1, \cdots, m) \;\; (j = 1, \cdots, n)$$

$$\sum_{i=1}^{m} y_i \geq k$$

$$x \in \{0,1\}^n, \;\; x^i \in \{0,1\}^n, \;\; y \in \{0,1\}^m$$

The continuous multidimensional knapsack problem with violations ($CKPmv$) is defined as $KPmv$ with $X = [0,1]^p$ instead of $X = \{0,1\}^p$. The same change is used to obtain the Big-$M$ formulation ($CKPmv_M$), the problem $CKPmv_k(M,w)$ and the extended formulation $CKPmvE$.

Again, if $M_{0i} = \sum_{j=1}^{n} a_{ij} - b_i$ then $F(CKPmv_{M_0}) = F(CKPmv)$

The data were generated following an standard procedure widely used:

$a_{ij} \sim U(0,1000)$, $c_i = \frac{\sum_{i=1}^{m} a_{ij}}{m} + 500u$ where $u \sim U(0,1)$ and $b_i = 0.5 \sum_{j=1}^{n} a_{ij}$.

In order to generate $w$, a lower bound of $v(KPmv)$ (or $v(CKPmv)$), we solve aproximately $KPmv_{M_0}$ (or $CKPmv_{M_0}$) by using CPLEX with time limit set to *heutime*.

All instances are available and can be requested from the author.

Tables A.1.1 until A.1.7 refer to $KPmv$. The size of the problems is $(n,m,k)$ according the formulation presented.

The 0-strategy and the **ALP**-strategy were applied to problems I.1 until III.5. For the same problems we apply the **AMIP**-strategy with several options

to $tmax$. The **E** strategy was applied to problems II.1 until II.5 and finally the 0-strategy and the **E**-strategy were applied to problems EI.1 until EI.5.

By far the best strategy was the **AMIP**-strategy. The averages of the relative difference between the gaps $\Delta$gapAMIP-$M_0(\%)$ and $\Delta$gapALP-$M_0(\%)$ were: 100% and 19% for the problems I.1 until I.5, 36% and 13% for the problems II.1 until II.5 and 18% and 9% for the problems III.1 until III.5. Only one time the 0-strategy was better than the **AMIP**-strategy (problem III.3 with $tmax = 20sec.$). For the same problem the **ALP**-strategy was better than the 0-strategy.

The performance of the **E**-strategy was very poor for all instances considered (see tables A.1.3 and A.1.7).
Tables A.2.1 until A.2.8 refer to $CKPmv$. The size of the problems is $(n, m, k)$ according the formulation presented.

The 0-strategy and the **ALP**-strategy were applied to problems CI.1 until CIII.5. For the same problems we apply the **AMIP**-strategy with several options to $tmax$. The **E** strategy was applied to problems CIII.1 until CIII.5 and finally the 0-strategy and the **E**-strategy were applied to problems CEI.1 until CEII.5.

By far, again, the best strategy was the **AMIP**-strategy. The averages of the relative difference between the times $\Delta$timeAMIP-$M_0(\%)$ and $\Delta$timeALP-$M_0(\%)$ were: 62% and 50% for the problems CI.1 until CI.5 and 87% and 52% for the problems CII.1 until CII.5. The averages of the relative difference between the gaps $\Delta$gapAMIP-$M_0$ (%) and $\Delta$gapALP-$M_0$ (%) were: 37% and 29% for the problems CIII.1 until CIII.5.

The performance of the **E**-strategy was very poor for all instances considered (see tables A.2.5, A.2.7 and A.2.8).

## 3.2 An heuristic and computational results for the quadratic multidimensional knapsack problem

The quadratic multidimensional knapsack problem ($QKPm$) is defined as follows ([14]):

$$(QKPm)\ \ max\ \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n}(\sum_{j=1}^{n} Q_{ij} x_j)x_i\ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i\ \ (i = 1, \cdots, m)$$

$$x \in \{0,1\}^n$$

where $a_{ij} \geq 0$, $b_i \geq 0$, $Q_{ij} \geq 0$, $c_i \geq 0$, $Q_{ii} = 0$ and $Q_{ij} = Q_{ji}$ for all $i, j$.

$QKPm$ may be written as a MIP problem with LI. One of the most used formulation is defined as follows ([14]):

$$(QKPm) \ \ max \ \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} z_i \ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ (i = 1, \cdots, m)$$

$$z_i \leq \sum_{j=1}^{n} Q_{ij} x_j \ \ (i = 1, \cdots, n)$$

$$(LI) x_i = 0 \Rightarrow z_i = 0 \ \ (i = 1, \cdots, n), \ \ x \in \{0,1\}^n, \ \ z \geq 0$$

and the corresponding Big-$M$ formulation is:

$$(QKPm_M) \ \ max \ \sum_{i=1}^{n} c_i x_i + \sum z_i \ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ (i = 1, \cdots, m)$$

$$z_i \leq M_i x_i, \ \ z_i \leq \sum_{j=1}^{n} Q_{ij} x_j \ \ (i = 1, \cdots, n)$$

$$x \in \{0,1\}^n, \ \ z \geq 0$$

In order to define $M_0$ consider the followings $KPm$ problems:

$$(KPm_i) \ \ max \sum_{j=1}^{n} Q_{ij} x_j \ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \ \ (i = 1, \cdots, m)$$

$$x \in \{0,1\}^n, \ \ x_i = 1$$

If $M_{0i} = v(KPm_i) \ \ \forall i$ then $F(QKPm) = F(QKPm_{M_0})$.

If $w$ is a lower bound of $v(QKPm)$ then $QKPm_k(M, w)$ is defined as follows:

$$(QKPm_k(M,w))\ \ max\ z_k\ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i\ \ (i=1,\cdots,m)$$

$$z_i \leq M_i x_i,\ \ z_i \leq \sum_{j=1}^{n} Q_{ij}x_j\ \ (i=1,\cdots,m)$$

$$\sum_{i=1}^{n}(c_i x_i + z_i) \geq w$$

$$x \in \{0,1\}^n,\ \ z \geq 0$$

Note that if $\widehat{Q}_{ij} = 2Q_{ij}\ \ \forall (i,j)$ such that $j > i$ and $\widehat{Q}_{ij} = 0\ \ \forall (i,j)$ such that $j \leq i$ then we can rewrite $QKPm$, $QKPm_M$, $KPm(i)$ and $QKPm_k(M,w)$ by using $\widehat{Q}$ instead of $Q$. In our computational experience we consider the formulations using either $Q$ or $\widehat{Q}$.

### 3.2.1 An heuristic to solve aproximately the $QKPm$ problem

In order to generate $w$, a lower bound of $v(QKPm)$, we use a very simple heuristic. Let $R$ be defined as follows: let $R_{ij} = Q_{ij}$ for all $i,j$ and let $R_{ii} = c_i$ for all $i$. The $QKPm$ problem may be rewriten as follows:

$$(QKPm)\ \ max\ x^t R y\ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i,\ \ \sum_{j=1}^{n} a_{ij}y_j \leq b_i\ \ (i=1,\cdots,m)$$

$$x \in \{0,1\}^n,\ \ y \in \{0,1\}^n$$

$$x = y$$

From this point of view we have a 0-1 bilinear problem. If we delete the constraints $x = y$ and we fix the variables $x$ with $x = \hat{x}$ where $\sum_{j=1}^{n} a_{ij}\hat{x}_j \leq b_i\ \ \forall i$ and $\hat{x} \in \{0,1\}^n$ then the corresponding problem in the variables $y$ is a $KPm$ as follows:

$$(KPm(\hat{x}))\ \ max\ \hat{x}^t R y\ \ s.t.$$

$$\sum_{j=1}^{n} a_{ij}y_j \leq b_i,\ \ (i=1,\cdots,m)$$

$$y \in \{0,1\}^n$$

The heuristic is defined as follows:

Let $x^1$ where $\sum_{j=1}^{n} a_{ij}x_j^1 \leq b_i \ \ \forall i$ and $x^1 \in \{0,1\}^n$.

We solve $KPm(x^1)$. Let $y$ be an optimal solution of $KPm(x^1)$. Let $x^2 = y$. Next solve $KPm(x^2)$. Let $y$ be an optimal solution of $KPm(x^2)$. Let $x^3 = y$ and so on. Note that ${x^s}^t R x^{s+1} \leq {x^{s+1}}^t R x^{s+2} \quad \forall s$. The heuristic stops if $v(KPm(x^s)) = v(KPm(x^{s+1}))$. Let $x^1, \cdots, x^r$ be the generated solutions. If ${x^q}^t R x^q = max\{\ {x^s}^t R x^s \ \ (s = 1, \cdots, r)\ \}$ then the output of the heuristic is $x^q$ and its value is ${x^q}^t R x^q$.

We apply the heuristic several times and save the best value generated to obtain $w$, a lower bound for $v(QKPm)$. In order to define $x^1$ we use an optimal solution of $KPm_1$ and if the limit time for the heuristic has not been exceeded then we use an optimal solution of $KPm_2$ and so on.

In order to generate $w$, a lower bound of $v(QKPm)$, we use the heuristic presented with time limit fix to *heutime*.

### 3.2.2 Computational results for the $QKPm$ problem.

The heuristic presented works very well and with $heutime = 5sec.$ we obtain a near optimal solution for all instances. The details will be reported in the future.

Problems selected for experimentation are presented in [14] and data were obtained from the authors on request. We consider the medium problems from $m1$ until $m24$ (with $(n, m)$ from $(200, 2)$ until $(500, 10)$ and with the matrix $Q$ generated with low (25%), medium (50%) and high (75%) densities (the details can be seen in [14]). The problems $m1, m2, m4, m6, m7, m9, m11, m12, m14, m20$ and $m23$ may be solved with the 0-strategy in a few seconds (with $mipgap = 0.01$) and it is impossible to compensate the computational effort of the preprocessing phase to obtain the final valid vector $M$. The time of the preprocessing phase is equal to $heutime +$ time to obtain$M_0 + tmax \times n(sec.)$ and even with $tmax = 1sec.$ the 0-strategy is better than the **AMIP**-strategy for the above problems. The second point of the last section about the trade off between the preprocessing phase to obtain the final valid vector $M$ and the processing phase to solve $P_M$ is motivated by the previous comments.

The 0-strategy and the **AMIP**-strategy were applied to the rest of the problems. Table A.3.1 refers to $QKPm$ and table A.3.2 refers to $(\widehat{Q}KPm)$. With both strategies we use the CPLEX.mipstart property in order to use the heuristic solution obtained.

By far, again, the best strategy was the **AMIP**-strategy. The averages of the relative difference between the gaps ($\Delta$gapAMIP-$M_0$ (%)) were when we use $mipgap = 0.01$ and $mipgap = 0$: 33% and 47% for the problems written as $QKPm$ (table A.3.1) and 33% and 48% for the problems written as ($\widehat{Q}KPm$) (table A.3.2). Two times the 0-strategy was better than the **AMIP**-strategy: problem $m15$ written as $QKPm$ with $mipgap = 0.01$ (the tolerance required was reached in 415 seconds by using the 0-strategy and 470 seconds by using the **AMIP**-strategy) and problem $m13$ written as ($\widetilde{Q}KPm$) with the same $mipgap$ (the tolerance required was reached in 293 seconds by using the 0-strategy and 433 seconds by using the **AMIP**-strategy).

# 4 Summary and further extensions

Let $M_0$ a valid vector to define the Big-$M_0$ formulation of $P$, a MIP problem with LI. A mathematical programming approach to obtain $M$, a valid vector with $M \leq M_0$, was presented. The procedure do not use the structure of the problem being solved and can be seen as a generalization of the approaches presented in [9] and [2]. The computational experience presented shows that our procedure is very promising. For several problems the performance of a general branch and cut algorithm used to solve $P$ by solving $P_M$ compensate the effort required to find $M$.

The trade-off between the preprocessing phase to obtain the final valid vector $M$ and the processing phase to solve $P_M$ may be a very interesting topic to be researched. A better approach than the presented here may be something like: use an standard branch and cut algorithm to try to solve $P_{M_0}$ and if some conditions are detected then stops the processing to find a new valid vector $M$ to continue later by solving $P_M$ instead of $P_{M_0}$, without loose the information previously generated.

A sensitivity analysis may be designed to specific problems in order to learn to define the parameters of our procedures ($smax$, $\epsilon$ and $tmax$). The relation between the quality of $M$ and the $w$ value must be explored in order to fix the time to obtain $w$ if a general branch and cut algorithm is being used.

For some problems (the maximum feasible subsystem problem for example) our procedure was unable to generate a valid vector with a significative difference with respect to the initial valid vector (at least with the data used so far). We hope to understand this in the near future in order to modify our procedure or to decide not to use it when corresponding.

Far as we know this is the first paper that presents a computational experience that shows the benefits of having a good valid vector including in the analysis the computational effort to find it.

# A Tables

## A.1 Tables for $KPmv$

| **P** | **gap**($M_0$)(%) | **DifALP**(%) | **iterALP** | **gapALP**(%) | **ΔgapALP**-$M_0$ (%) |
|---|---|---|---|---|---|
| I.1 | 7.92 | 44 | 7 | 6.43 | 19 |
| I.2 | 6.85 | 44 | 7 | 4.94 | 28 |
| I.3 | 8.97 | 42 | 6 | 7.59 | 15 |
| I.4 | 7.72 | 42 | 7 | 5.52 | 28 |
| I.5 | 8.97 | 44 | 7 | 6.81 | 24 |

Table A.1.1 SET I. $KPmv$.
$(n, m, k) = (20, 80, 64)$, $heutime = 30(sec.)$, $mipgap = 0$, $timelimit = 3600 - heutime$(sec.), $smax = \infty$, $\epsilon = 1$

| **P** | **DifAMIP**(%) | **totaltAMIP(sec.)** | **gapAMIP**(%) | **t**($M_0$) | **gap**($M_0$)(%) |
|---|---|---|---|---|---|
| I.1 | 64 66 67 68 69 | 1727 1657 1396 1789 1653 | 0 0 0 0 0 | 1789 | 8.49 |
| I.2 | 64 66 67 68 69 | 1773 1835 1652 2106 1577 | 0 0 0 0 0 | 2106 | 7.31 |
| I.3 | 63 65 66 67 68 | 3600 3600 2978 2837 3557 | 4.92 3.25 0 0 0 | 3557 | 8.92 |
| I.4 | 63 65 66 67 67 | 1685 3388 2151 1626 1715 | 0 0 0 0 0 | 2151 | 7.92 |
| I.5 | 64 65 67 68 68 | 2168 3600 1817 3343 2517 | 0 3.98 0 0 0 | 3343 | 9.06 |

Table A.1.2 SET I $KPmv$.
$(n, m, k) = (20, 80, 64)$, $mipgap = 0$, $heutime = 30(sec.)$, $tmax \in \{5, 8, 10, 12, 15\}$(sec.), $preproctime = heutime + tmax \times m(sec.)$, $timelimit = 3600 - preproctime$(sec.)

| **P** | **gap**($M_0$)(%) | **DifALP**(%) | **iterALP** | **gapALP**(%) | **ΔgapALP**-$M_0$ (%) | **gapE**(%) |
|---|---|---|---|---|---|---|
| II.1 | 2.15 | 39 | 6 | 2.20 | **-2** | 12.03 |
| II.2 | 2.57 | 39 | 6 | 2.23 | 13 | 10.37 |
| II.3 | 2.38 | 39 | 6 | 2.01 | 16 | 11.02 |
| II.4 | 2.20 | 40 | 6 | 1.79 | 19 | 10.57 |
| II.5 | 2.15 | 39 | 6 | 1.72 | 20 | 10.91 |

Table A.1.3 SET II. $KPmv$.
$(n, m, k) = (125, 50, 40)$, $heutime = 30(sec.)$, $mipgap = 0$, $timelimit = 3600 - heutime$(sec.), $smax = 1$, $\epsilon = 1$

| **P** | **DifAMIP(%)** | **gapAMIP(%)** | **Δgap AMIP**-$M_0$ (%) |
|---|---|---|---|
| II.1 | 53 55 55 56 57 | 1.48 1.65 1.73 1.73 1.75 | 31 23 20 20 19 |
| II.2 | 52 54 56 56 57 | 1.57 1.27 1.35 1.56 1.40 | 39 50 47 39 45 |
| II.3 | 53 56 57 58 58 | 1.45 1.46 1.37 1.47 1.60 | 39 39 43 38 33 |
| II.4 | 53 55 56 56 57 | 1.48 1.47 1.48 1.38 1.57 | 33 33 33 37 29 |
| II.5 | 52 54 56 57 57 | 1.11 1.15 1.24 1.25 1.31 | 48 47 42 42 39 |

Table A.1.4 SET II $KPmv$.
$(n, m, k) = (125, 50, 40)$, $mipgap = 0$, $heutime = 30(sec.)$, $tmax \in \{5, 10, 15, 20, 25\}$(sec.), $preproctime \approx heutime + tmax \times m(sec.)$, $timelimit = 3600 - preproctime$(sec)

| **P** | **gap**($M_0$)(%) | **DifALP(%)** | **iterALP** | **gapALP**(%) | **$\Delta$gapALP-$M_0$** (%) |
|---|---|---|---|---|---|
| III.1 | 0.76 | 40 | 6 | 0.58 | 25 |
| III.2 | 0.51 | 40 | 6 | 0.45 | 12 |
| III.3 | 0.57 | 40 | 6 | 0.55 | 4 |
| III.4 | 0.65 | 39 | 6 | 0.64 | 2 |
| III.5 | 0.66 | 40 | 6 | 0.64 | 3 |

Table A.1.5 SET III. $KPmv$.

$(n, m, k) = (100, 40, 32)$, $heutime = 30(sec.)$, $mipgap = 0$, $timelimit = 3600 - heutime$(sec.), $smax = 1$, $\epsilon = 1$

| **P** | **DifAMIP**(%) | **gapAMIP**(%) | **$\Delta$gapAMIP-$M_0$** (%) |
|---|---|---|---|
| III.1 | 52 57 59 60 61 | 0.50 0.49 0.49 0.50 0.51 | 34 35 35 34 34 |
| III.2 | 52 57 60 61 62 | 0.41 0.40 0.41 0.41 0.42 | 20 23 20 19 18 |
| III.3 | 53 60 62 64 65 | 0.48 0.48 0.48 0.50 0.60 | 16 17 16 13 **-4** |
| III.4 | 52 59 61 63 63 | 0.58 0.59 0.59 0.60 0.63 | 11 9 10 8 3 |
| III.5 | 53 59 62 64 64 | 0.60 0.50 0.49 0.59 0.53 | 9 24 25 11 20 |

Table A.1.6 SET III $KPmv$.

$(n, m, k) = (100, 40, 32)$, $mipgap = 0$, $heutime = 30(sec.)$, $tmax \in \{2, 5, 10, 15, 20\}$(sec.), $preproctime \approx heutime + tmax \times m(sec.)$, $timelimit = 3600 - preproctime$(sec.)

| **P** | **t**($M_0$) | **gap**($M_0$)(%) | **tE** | **gapE**(%) |
|---|---|---|---|---|
| EI.1 | 23 | 0 | 630 | 0 |
| EI.2 | 32 | 0 | 1057 | 0 |
| EI.3 | 46 | 0 | 3600 | 2.02 |
| EI.4 | 4 | 0 | 287 | 0 |
| EI.5 | 62 | 0 | 3275 | 0 |

Table A.1.7 SET EI . $KPmv$.

$(n, m, k) = (80, 20, 16)$, $mipgap = 0$, $timelimit = 3600$(sec.)

## A.2 Tables for $CKPmv$

| **P** | **t**($M_0$) | **DifALP(%)** | **iterALP** | **totaltALP(sec.)** | **$\Delta$time ALP-Cx**(%) |
|---|---|---|---|---|---|
| CI.1 | 1576 | 40 | 6 | 734 | 53 |
| CI.2 | 1503 | 40 | 6 | 687 | 54 |
| CI.3 | 3871 | 39 | 6 | 1637 | 58 |
| CI.4 | 1374 | 39 | 6 | 965 | 30 |
| CI.5 | 2548 | 39 | 6 | 1216 | 52 |

Table A.2.1 SET CI. $CKPmv$.

$(n, m, k) = (200, 50, 40)$, $heutime = 30(sec.)$, $mipgap = 0.01$, $timelimit = \infty$, $\epsilon = 1$

| **P** | **DifAMIP**(%) | **totaltAMIP(sec.)** | **$\Delta$time AMIP-$M_0$**(%) |
|---|---|---|---|
| CI.1 | 47 53 53 54 58 | 878 575 530 642 951 | 44 64 66 59 40 |
| CI.2 | 46 52 52 53 57 | 570 437 557 542 762 | 62 71 63 64 49 |
| CI.3 | 45 50 50 51 54 | 1331 1310 1199 1209 1286 | 66 66 69 69 67 |
| CI.4 | 50 52 52 54 59 | 522 476 499 557 699 | 62 65 64 59 49 |
| CI.5 | 49 51 51 52 57 | 856 819 831 883 944 | 66 68 67 65 61 |

Table A.2.2 SET CI $CKPmv$.

$(n, m, k) = (200, 50, 40)$, $mipgap = 0.01$, $heutime = 30(sec.)$, $timelimit = \infty$, $tmax \in \{1, 2, 3, 5, 10\}$(sec.), $preproctime \approx heutime + tmax \times m(sec.)$

| **P** | **t**($M_0$) | **DifALP**(%) | **iterALP** | **totaltALP(sec.)** | **$\Delta$time ALP-$M_0$** (%) |
|---|---|---|---|---|---|
| CII.1 | 3678 | 40 | 6 | 1610 | 56 |
| CII.2 | 3720 | 39 | 7 | 1207 | 68 |
| CII.3 | 3677 | 39 | 6 | 2115 | 42 |
| CII.4 | 3662 | 39 | 6 | 2885 | 21 |
| CII.5 | 3699 | 39 | 6 | 1034 | 72 |

Table A.2.3 SET CII. $CKPmv$.

$(n, m, k) = (250, 50, 40)$, $heutime = 30(sec.)$, $mipgap = 0.01$, $timelimit = \infty$

| **P** | **DifAMIP**(%) | **totaltAMIP(sec.)** | **$\Delta$time AMIP-$M_0$**(%) |
|---|---|---|---|
| CII.1 | 47 50 51 51 56 | 841 1009 1050 961 1016 | 77 73 71 74 72 |
| CII.2 | 47 51 50 52 57 | 638 842 639 696 856 | 83 77 83 81 77 |
| CII.3 | 46 50 50 51 56 | 1082 1238 1004 1024 1055 | 71 66 73 72 71 |
| CII.4 | 46 48 49 50 54 | 2174 2254 1710 2099 1745 | 41 38 53 43 52 |
| CII.5 | 47 51 51 52 57 | 790 645 743 873 925 | 79 83 80 76 75 |

Table A.2.4 SET CII $CKPmv$.

$(n, m, k) = (250, 50, 40)$, $mipgap = 0.01$, $heutime = 30(sec.)$, $timelimit = \infty$, $tmax \in \{1, 2, 3, 5, 10\}$(sec.), $preproctime \approx heutime + tmax \times m(sec.)$

| **P** | **gap**($M_0$) | **DifALP**(%) | **iterALP** | **gapALP**(%) | **$\Delta$gapALP-$M_0$** (%) | **gapE** |
|---|---|---|---|---|---|---|
| CIII.1 | 2.86 | 39 | 7 | 1.99 | 31 | 11.72 |
| CIII.2 | 2.94 | 39 | 7 | 2.14 | 27 | 12.06 |
| CIII.3 | 2.89 | 39 | 7 | 2.11 | 27 | 12.20 |
| CIII.4 | 2.70 | 39 | 7 | 1.93 | 28 | 10.69 |
| CIII.5 | 3.01 | 39 | 7 | 2.12 | 30 | 11.36 |

Table A.2.5 SET CIII. $CKPmv$.

$(n, m, k) = (250, 65, 52)$, $heutime = 30(sec.)$, $mipgap = 0.01$, $timelimit = 3600 - heutime$(sec.), $timelimit = 3600$ for the **E**-strategy

| **P** | **DifAMIP**(%) | **gapAMIP**(%) | **Δgap AMIP**-$M_0$ (%) |
|---|---|---|---|
| CIII.1 | 44 48 50 51 | 1.84 1.71 1.68 1.66 | 36 40 41 42 |
| CIII.2 | 44 48 50 51 | 2.00 1.94 1.85 1.86 | 32 34 37 37 |
| CIII.3 | 44 48 51 51 | 1.97 1.89 1.84 1.75 | 32 35 36 39 |
| CIII.4 | 44 49 51 52 | 1.95 1.69 1.65 1.66 | 28 37 39 39 |
| CIII.5 | 44 48 50 51 | 1.90 1.82 1.86 1.77 | 37 39 38 41 |

Table A.2.6 SET CIII $CKPmv$.

$(n, m, k) = (250, 65, 52)$, $mipgap = 0.01$, $heutime = 30(sec.)$, $tmax \in \{1, 5, 10, 12\}$(sec.), $preproctime \approx heutime + tmax \times m(sec.)$, $timelimit = 3600 - preproctime$(sec.)

| **P** | **t**($M_0$)(sec.) | **nodes**($M_0$) | **tE** | **nodesE** |
|---|---|---|---|---|
| CEI.1 | 1.4 | 15810 | 139 | 5793 |
| CEI.2 | 0.6 | 7691 | 105 | 4257 |
| CEI.3 | 2.0 | 24639 | 368 | 20768 |
| CEI.4 | 1.4 | 15361 | 106 | 4977 |
| CEI.5 | 0.5 | 5663 | 83 | 3333 |

Table A.2.7 SET CEI. $CKPmv$.

$(n, m, k) = (100, 25, 20)$, $mipgap = 0$

| **P** | **t**($M_0$) | **gap**($M_0$)(%) | **gapE** (%) |
|---|---|---|---|
| CEII.1 | 66 | 0 | 7.00 |
| CEII.2 | 110 | 0 | 6.34 |
| CEII.3 | 30 | 0 | 7.54 |
| CEII.4 | 36 | 0 | 8.28 |
| CEII.5 | 90 | 0 | 6.45 |

Table A.2.8 SET CEII. $CKPmv$.

$(n, m, k) = (50, 50, 40)$, $mipgap = 0$

## A.3 Tables for $QKPm$

| **P** | **DifAMIP**(%) | **totaltAMIP** | **gapAMIP**(%) | **t**($M_0$) | **gap**($M_0$) | **Δ gapAMIP**-$M_0$ (%) |
|---|---|---|---|---|---|---|
| m3 | 12 | 231 3600 | 1.00 0.36 | 231 3600 | 1.55 1.19 | 35 70 |
| m5 | 25 | 3600 3600 | 5.75 5.74 | 3600 3600 | 13.46 13.48 | 57 57 |
| m8 | 9 | 460 3600 | 1.00 0.66 | 460 3600 | 1.48 1.27 | 32 48 |
| m10 | 27 | 3600 3600 | 13.56 15.25 | 3600 3600 | 22.12 22.17 | 39 31 |
| m13 | 34 | 462 3600 | 1.00 0.19 | 462 3600 | 1.18 0.90 | 15 78 |
| m15 | 15 | 470 3600 | 1.00 0.15 | **415** 3600 | 1.00 0.70 | - 79 |
| m16 | 45 | 397 419 | 1.00 0.00 | 397 419 | 5.10 4.66 | 80 100 |
| m17 | 21 | 3600 3600 | 4.57 4.71 | 3600 3600 | 6.61 6.63 | 31 29 |
| m18 | 13 | 3600 3600 | 4.20 4.14 | 3600 3600 | 7.72 7.73 | 46 46 |
| m19 | 23 | 3600 3600 | 22.93 22.83 | 3600 3600 | 29.67 29.63 | 23 23 |
| m21 | 12 | 3600 3600 | 1.36 1.55 | 3600 3600 | 2.18 2.18 | 38 29 |
| m22 | 29 | 3600 3600 | 29.31 28.46 | 3600 3600 | 41.10 41.10 | 29 31 |
| m24 | 15 | 598 3600 | 1.0 0.17 | 598 3600 | 1.49 1.23 | 33 86 |

Table A.3.1 $QKPmv$.

$mipgap \in \{0.01, 0\}$, $heutime = 5(sec.)$, $tmax = 1$, $preproctime \approx heutime +$ time to obtain$M_0 + tmax \times m(sec.)$, $timelimit = 3600 -$ time to obtain$M_0$(sec.) for the 0-strategy , $timelimit = 3600 - preproctime$(sec.) for the **AMIP**-strategy.

| **P** | **DifAMIP**(%) | **totaltAMIP** | **gapAMIP**(%) | **t**($M_0$) | **gap**($M_0$) | $\Delta$ **gapAMIP**-$M_0$ (%) |
|---|---|---|---|---|---|---|
| m3 | 13 | 219 3600 | 1.00 0.19 | 219 3600 | 1.40 0.79 | 29 76 |
| m5 | 24 | 3600 3600 | 6.45 7.34 | 3600 3600 | 16.97 16.97 | 62 57 |
| m8 | 11 | 464 3600 | 1.00 0.49 | 464 3600 | 1.62 1.23 | 38 60 |
| m10 | 26 | 3600 3600 | 14.56 14.93 | 3600 3600 | 18.30 18.29 | 20 18 |
| m13 | 37 | 433 931 | 1.00 0.00 | **293** 931 | 1.00 0.80 | - 100 |
| m15 | 16 | 508 3600 | 1.00 0.16 | 3600 3600 | 1.24 1.24 | 19 87 |
| m16 | 38 | 416 444 | 1.00 0.00 | 416 444 | 9.00 8.72 | 88 100 |
| m17 | 22 | 3600 3600 | 3.51 3.54 | 3600 3600 | 5.39 5.39 | 35 34 |
| m18 | 16 | 3600 3600 | 8.00 7.88 | 3600 3600 | 13.30 13.31 | 40 41 |
| m19 | 23 | 3600 3600 | 26.65 25.92 | 3600 3600 | 31.75 31.77 | 16 18 |
| m21 | 14 | 3600 3600 | 1.58 1.75 | 3600 3600 | 2.29 2.29 | 31 24 |
| m22 | 24 | 3600 3600 | 33.87 34.74 | 3600 3600 | 42.12 42.12 | 20 18 |
| m24 | 17 | 774 3600 | 1.00 0.27 | 774 3600 | 2.60 2.06 | 62 87 |

Table A.3.2 $\widehat{Q}KPmv$.

$mipgap \in \{0.01, 0\}$, $heutime = 5(sec.)$, $tmax = 1$, $preproctime \approx heutime +$ time to obtain$M_0 +$ $tmax \times m(sec.)$, $timelimit = 3600 -$ time to obtain$M_0$(sec.) for the 0-strategy , $timelimit = 3600 - preproctime$(sec.) for **AMIP**-strategy.

# References


[1] Edoardo Amaldi, Marc E. Pfetsch and Leslie E. Trotter, Jr. (2003): On the maximum feasible subsystem problem, IISs and IIS-Hypergraphs. *Mathematical programming, 95* (3) 533-554.

[2] A. Billionet and E. Soutif (2004): Using a mixed integer programming tool for solving the 0-1 quadratic knapsack problem. *INFORMS J. on Comp. 16* 188-197.

[3] Jeffry D. Camm, Amitbabh S. Raturi and Shigeru Tsubakitani (1990): Cutting Big M Down to Size.*INTERFACES 20* (5) 61-66.

[4] http://ibm-ilog-cplex-optimization-studio-acade.software.informer.com/12.4/

[5] Jen-Shiang Chen (2006): Using integer programming to solve the machine scheduling problem with a flexible maintenance activity. *Journal of Statistics & Management Science 9* (1), 87-104.

[6] Gianni Codato and Matteo Fischetti (2006): Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research 54* (4) 756-766.

[7] Arnaud Fréville (2004): The multidimensional 01 knapsack problem: An overview. *European Journal of Operational Research 155*, (1) 121.

[8] Eng Kiat Russell Gan (2012): Adaptive Branch and Bound for Efficient Solution of Mixed Integer Programs Formulated with Big-$M$. *Naval Postgraduate School, Master's Thesis* URL: http://hdl.handle.net/10945/17370

[9] Joseft Kallrath (2009): Combined strategic design and operational planning in the process industry. *Computers and Chemical Engineering 33* 1983-1993.

[10] Ahmet B. Keha, Ketan Khowala and John W. Fowler (2009): Mixed integer programming formulations for single machine scheduling problems. *Computers and Industrial Engineering 56* 357-367.

[11] Ed Klotz and Alexandra M. Newman (2013): Practical Guidelines for Solving Difficult Mixed Integer Linear Programs. *Surveys in Operations Research and Management Science, 18*, (1-2) 18-32.

[12] Feng Qiu, Shabbir Ahmed, Santanu S. Dey and Laurence A. Wolsey (2014): Covering Linear Programming with Violations. *Accepted for publication in INFORMS Journal on Computing, Articles in Advance April 11* http://dx.doi.org/10.1287/ijoc.2013.0582.

[13] P.A.Rubin (1997): Solving Mixed Integer Classification Problem by Decomposition. *Annals of Operations Research, 74*, 51-64.

[14] Haibo Wang and Fred Glover (2012): A computational study on the quadratic knapsack problem with multiple constraints. *Journal Computers and Operations Research, 39*, (1) 3-11.