

Comparing integer quadratic programming formulations for HA-assignment problems

Hugo José Lara Urdaneta (UFSC-Blumenau) hugo.lara.urdaneta@ufsc.br

Jinyun Yuan (UFPR- Curitiba) yuanjy@gmail.com

Abel Siqueira (UFPR- Curitiba) abel.s.siqueira@gmail.com

Resumo:

Problemas de alocação local-visitante são modelados de forma natural como programas quadráticos em variáveis binárias. Descrevemos uma formulação quadrática com restrições lineares, e oferecemos uma versão com restrições quadráticas. Logo comparamos formulações alternativas para dita classe de problemas. Primeiro, propomos uma outra formulação através da manipulação da sua estrutura especial, obtendo versões com 1/4 do tamanho original. Linearizando a função objetivo quadrática, obtemos outro par de modelos, que são comparados com os quadráticos. Os experimentos numéricos mostram as características de cada modelo proposto.

Palavras chave: Calendários esportivos, programação quadrática inteira.

Comparing integer quadratic programming formulations for HA-assignment problems

Abstract

Home-Away Assignment problems are naturally cast as quadratic programming models in binary variables. In this work we compare alternative formulations for this kind of problems. First, write a quadratic programming formulation with linear constraints, and then a quadratically constrained version. We also propose another formulation by manipulating their special structure to obtain versions with 1/4 of the original size. By linearizing the quadratic objective function, we get two more alternative models to be compared with the quadratic ones. Numerical experiments exhibit the characteristics of each model.

Key-words: Sports scheduling, Integer quadratic programming.

1. Introduction

Several problems in sport scheduling have been focus of attention in the Operational Research community. One of them, the Home-Away assignment problem (HA-Assignment) assign the label home (H) or away (A) to each match of a double round robin tournament, in a such way that some decision criteria is achieved (see Easton and Trick, 2001 and 2004). Some of these criteria involves minimizing the total traveling distance of the teams, or

minimizing the number of breaks (Easton and Trick, 2001, and Trick, 2006). Models dealing with HA-Assignment problems have been proposed, most of them as linear integer programs (Easton and Trick, 2001, 2004, or Trick, 2006), or as MIN-RES-CUT problems as Suzuka (2005), among others. We here propose integer linear and quadratic formulations based on cuts over a graph, but in a different flavor when compared with the MIN-RES-CUT formulation in Suzuka (2005).

1.1 Modeling HA-assignments

In the sequel we introduce, like in Suzuka (2005), the mathematical definition of the problem. Through this paper, we deal with a double round-robin tournament with a pair number $(2n)$ of teams. In a round robin tournament each team plays every other team twice, once at home and the other away. A slot, or round, is a date where all the teams plays against other team. The number of slots is $2(2n-1)$, and each team has its home and each match is held at the home of one of the two playing teams.

A *timetable* is a matrix $\mathcal{T}$ whose rows are indexed by a set of teams $T=\{1,2,..., 2n\}$, and columns are indexed by a set of slots (rounds) $S=\{1,2,...,4n-2\}$. Each entry of a timetable, say $\tau(t,s)$ in $T \times S$, shows the opponent of team t in slot s . A timetable $\mathcal{T}$ should satisfy the following conditions:

- for each team $t \in T$, the t -th row of $\mathcal{T}$ contains each element of $T \setminus \{t\}$ exactly twice;
- for any $(t,s) \in T \times S$, $\tau(\tau(t,s),s)=t$.

The first condition means that each team plays exactly twice with every other team, while the second one establishes that the team playing with $\tau(t,s)$ in slot s should be t . In figure 1 we show a timetable for $n=2$ (four teams). Generating timetables has been focus of attention of some works in sport scheduling (Easton, 2004). It is an easy task randomly generate timetables. If some matches are fixed in advance, the work becomes harder (Ribeiro, 2010).

$$\mathcal{T} = \begin{array}{|c|c|c|c|c|c|} \hline 2 & 3 & 2 & 4 & 3 & 4 \\ \hline 1 & 4 & 1 & 3 & 4 & 3 \\ \hline 4 & 1 & 4 & 2 & 1 & 2 \\ \hline 3 & 2 & 3 & 1 & 2 & 1 \\ \hline \end{array}$$

Fig 1: A timetable matrix for $n=2$.

A home-away assignment $\mathcal{A}$ (HA-assignment for short) is a matrix whose rows are indexed by T , and the columns by S . Each entry of the HA-assignment, say $a_{(t,s)}$ ($(t,s) \in T \times S$), is either 'H' (home) or 'A' (away), according to the status of team t at round s . Given a timetable $\mathcal{T}$, an HA assignment $\mathcal{A}=(a_{(t,s)})(t,s) \in \{T \times S\}$ is said to be *consistent* with $\mathcal{T}$ if it satisfy:

- C1 $\forall (t,s) \in T \times S$, $\{a_{(t,s)}, a_{(\tau(t,s),s)}\} = \{A,H\}$, and
- C2 $\forall t \in T$, if $\tau(t,s) = \tau(t,s')$ and $s \neq s'$ then $\{a_{(t,s)}, a_{(t,s')}\} = \{A,H\}$.

In figure \ref{table2} we present an HA-assignment, consistent with the timetable shown in figure \ref{table1}. A schedule of a round-robin tournament is defined as a pair $(\mathcal{T}, \mathcal{A})$ of a timetable and a HA-assignment consistent with the timetable.

$$A = \begin{bmatrix} H & H & A & A & A & H \\ A & H & H & H & A & A \\ A & A & H & A & H & H \\ H & A & A & H & H & A \end{bmatrix}$$

Figure 2: A HA-assignment consistent with $\mathfrak{T}$.

Our decision making is as follows: given a fixed timetable $\mathfrak{T}$, find a HA-assignment $\mathcal{A}$ consistent with $\mathfrak{T}$, according to some criteria. Break minimization and minimizing the total traveling distance, for instance, provide decision criteria based on a quadratic objective function in binary variables. We in consequence propose an unified framework, grouping these decision criteria.

1.2. Dealing with timetables

In Perdomo and Lara (2013) was proposed a procedure to build a partition in a timetable, in order to extract crucial information from it, and simplify the optimization models. We here formalize such procedure. Given a timetable $\mathfrak{T}$ and the index set $T \times S$, we construct a partition of the indices according to the following observation: For each (t,s) in $T \times S$, there exist unique indices (t',s') in $T \times S$, such that the indices in the set $\{(t,s), (t',s'), (t,s), (t',s')\}$ are related each other, and isolated from the rest of the indices. $t' = \tau(t,s)$ is the team which plays with t in slot s , and s' is the slot where the two teams plays again. Thus we can partition the index set $T \times S$ into $K=n(2n-1)$ subsets. Furthermore, we can order the elements in each group by appearance order in the table from left to right and from up to down. The following procedure assign the label $\gamma_{(t,s)} = (\gamma_{(t,s)}^1, \gamma_{(t,s)}^2)$ to each $(t,s) \in T \times S$:

Procedure:

Label $\gamma_{(t,s)} := [0, 0]$ for all $(t,s) \in T \times S$ and $k := 1$.

for $t=1$ to $2n$,

for $s=1$ to $4n-2$,

if $\gamma_{(t,s)} := [0, 0]$, then identify $t' > t, s' > s$ such that $\tau(t,s) = t'$, and $\tau(t,s) = \tau(t,s')$.

Label $\gamma_{(t,s)} := [k, 1], \gamma_{(t,s')} := [k, 2], \gamma_{(t',s)} := [k, 3], \gamma_{(t',s')} := [k, 4];$
 $k:=k+1,$

end

end

First let us define the set $K(k) = \{(t,s) \in T \times S : \gamma_{ts}^1 = k\}$ of indices labeled by k . Clearly $|K(k)| = 4$, because four labels with $\gamma_{ts}^1 = k$ where assigned in each step. The procedure pass through the timetable sequentially, by visiting each of the (t,s) components from left to right, and from up to down. Each component of $T \times S$ is labeled once, placing it into the $K(k)$ group, for some $k=1, \dots, n(2n-1)$. It is clear that for $k_1 \neq k_2$, $K(k_1) \cap K(k_2) = \emptyset$, since each component is labeled once. This last observation also explains that so we have our partition. In figure 3 we show labels assigned to each component of the timetable in figure 1. The

partition will be useful to construct the models later.

$$\gamma = \begin{bmatrix} [1, 1] & [2, 1] & [1, 2] & [3, 1] & [2, 2] & [3, 2] \\ [1, 3] & [4, 1] & [1, 4] & [5, 1] & [4, 2] & [5, 2] \\ [6, 1] & [2, 3] & [6, 2] & [5, 3] & [2, 4] & [5, 4] \\ [6, 3] & [4, 3] & [6, 4] & [3, 3] & [4, 4] & [3, 4] \end{bmatrix}$$

Figure 3: Labels assigned to each component of the timetable.

1.3. Organization

The remainder of this article is organized as follows. In the next section we describe the integer quadratic programming model, the first one with linear constraints, and the second with quadratic constraints; then we propose the procedure to reduce the problem size, and later, the linear versions are written. In section 3 we offer numerical results which compare the solver performance in each model. The last section is devoted to conclusion and final remarks.

2. Optimization models

2.1. The constraints

In this part we model the consistency **C1-C2** constraints of the HA-assignment problem as linear or quadratic equations in binary variables. Let us consider the undirected graph $G=(V,E)$ where the index set is $V = \{v_{ts} : (t,s) \in T \times S\}$, and the edges $E = \{(v_{ts-1}, v_{ts}) : t \in T \times S \setminus \{1\}\}$. Consider also the partition $\{K(k)\}_{k=1}^{n(2n-1)}$ defined in the section above. Associated to each entry in $K(k) = \{(t_k, s_k), (t_k, s'_k), (t'_k, s_k), (t'_k, s'_k)\}$ we denote the binary 0-1 variables $y_{t_k s_k}, y_{t_k s'_k}, y_{t'_k s_k}$, and $y_{t'_k s'_k}$. Each variable y_{ts} represents an entry on the timetable, and its value will construct the HA-assignment $\mathcal{A}$ in such way that if $y_{ts} = 0$, then $a_{ts} = H$ and the case $y_{ts} = 1$, then $a_{ts} = A$. In figure \ref{table4} we show the values for Y, associated to our example.

2.1.1. Linear constraints

The following linear constraints in binary variables model the conditions **C1-C2**: For each group $k, k=1, \dots, n(2n-1)$ we have

$$\begin{aligned} y_{t_k s_k} + y_{t'_k s_k} &= 1 \\ y_{t_k s_k} + y_{t_k s'_k} &= 1 \\ -y_{t_k s_k} + y_{t'_k s'_k} &= 0. \end{aligned} \tag{1}$$

The first equation means that only one of the teams plays at home in slot s_k . The second one establishes that team t_k should play alternatively at home and away in slots s_k and s'_k ; and the third one ensures that if the first team plays at home (away) at the first slot then the second team should be home (away) in the second match between them. Now, for $k=1, 2, \dots, n(2n-1)$, the equations (1) provide a system of equations of the form $Ay=b$ where A is a block diagonal matrix containing the coefficients above, and b the right hand side. It is worth to note here that the imposed order in the "block diagonal" components of A , determined by the $k=1, \dots, n(2n-1)$ groups is different from the natural order provided by the set $T \times S$.

2.1.2. Quadratic constraints

We also can enforce **C1-C2** conditions with the following quadratic constraints: For each k , $k=1,2,...,n(2n-1)$, we have

$$\begin{aligned} 2y_{t_{kS_k}}y_{t'_{kS_k}} - y_{t_{kS_k}} - y_{t'_{kS_k}} + 1 &= 0 \\ 2y_{t_{kS_k}}y_{t_{kS'_k}} - y_{t_{kS_k}} - y_{t_{kS'_k}} + 1 &= 0 \\ 2y_{t_{kS_k}}y_{t'_{kS'_k}} - y_{t_{kS_k}} - y_{t'_{kS'_k}} &= 0 \end{aligned} \quad (2)$$

We write the equations in matrix notation as

$$\begin{aligned} y^T Q_{k1} y + c_{k1}^T y + 1 &= 0 \\ y^T Q_{k2} y + c_{k1}^T y + 1 &= 0 \\ y^T Q_{k3} y + c_{k1}^T y + 1 &= 0 \end{aligned} \quad (3)$$

We use sparsity tools to deal with these highly sparse matrices.

2.2. The objective function

Some of the popular decision criteria that have been used to choose HA-assignments, like minimizing the total traveling distance (see Easton, 2001 and 2004), or minimizing the number of breaks as Elf (2003), and Miyashiro (2003), leads to quadratic functions on Y to be minimized. We will describe two different criteria which leads to quadratic functions.

2.2.1 The total traveling distance minimization problem

Instance: A timetable $\mathfrak{T}$ and distance matrix D which contains the distances between the venues of the teams.

Task: Find an HA-assignment $\mathcal{A}$ consistent with $\mathfrak{T}$ which minimizes the total traveling distance.

The traveling distance $l(t,s)$ of team t between rounds s and $s+1$ is defined as:

$$l(t,s) = \begin{cases} d(t,t) = 0 & \text{if } (a_{ts}, a_{t,s+1}) = (H, H) \\ d(\tau(t,s), \tau(t,s+1)) & \text{if } (a_{ts}, a_{t,s+1}) = (A, A) \\ d(t, \tau(t,s+1)) & \text{if } (a_{ts}, a_{t,s+1}) = (H, A) \\ d(\tau(t,s), t) & \text{if } (a_{ts}, a_{t,s+1}) = (A, H) \end{cases} \quad (4)$$

For each $(t,s) \in T \times S \setminus \{4n-2\}$, let us denote by $t' = \tau(t,s)$, and $t'' = \tau(t,s+1)$; and consider $y_{ts} \in \{0,1\}$, with value zero if $a_{ts} = H$ and one if $a_{ts} = A$, as formulated before. Then

$$l(t,s) = d(t',t'')y_{ts}y_{ts+1} + d(t,t'')(1-y_{ts})y_{ts+1} + d(t',t)y_{ts}(1-y_{ts+1}) \quad (5)$$

The total traveling distance for all the teams is given by

$$l(y) = \sum_{t=1}^{2n} \sum_{s=0}^{4n-2} l(t,s) \quad (6)$$

where $l(t,0) = d(t, \tau(t,1))y_{t1}$ and $l(t,4n-2) = d(\tau(t,4n-2), t)y_{t4n-2}$. For $y = (y_{ts})$, $((t,s) \in T \times S)$, this function can be written as $l(y) = c^T y + \frac{1}{2} y^T H y$,

a quadratic function in binary variables. Furthermore, the quadratic form H has a sparse configuration, because only two consecutive variables y_{ts} and y_{ts+1} are quadratically related.

2.2.2. Break minimization/maximization problems

Given an HA-assignment $\mathcal{A}$, it is said that the team $t \in T$ has a break at round $s \in S \setminus \{1\}$ if $a_{ts-1} = a_{ts} = A$ or $a_{ts-1} = a_{ts} = H$. The number of breaks $b(\mathcal{A})$ in a HA-assignment is defined as the number of breaks belonging to all teams. In practical sport scheduling, such as in \cite{Elf-03}, the number of breaks is required to be reduced.

Instance: a timetable $\mathfrak{T}$

Task: find an HA-assignment that is consistent with $\mathfrak{T}$ and minimizes/maximizes the number of breaks.

Our binary model about break minimization is based on the following observation: For any $(t, s) \in T \times S \setminus \{1\}$, $a_{ts-1} = a_{ts}$ if and only if

$$y_{\tau(ts-1)s-1} + y_{ts} = 1.$$

So, the expression

$$b_{ts}(y) = (1 - y_{\tau(ts-1)s-1})(1 - y_{ts}) + y_{\tau(ts-1)s-1}y_{ts} = 1 - y_{\tau(ts-1)s-1} - y_{ts} + 2y_{\tau(ts-1)s-1}y_{ts}$$

assumes the value 1 if there is a break, and zero otherwise. The total number of breaks $b(y)$ is given by

$$b(y) = \sum_{t \in T} \sum_{s \in S \setminus \{1\}} (1 - y_{\tau(ts-1)s-1} - y_{ts} + 2y_{\tau(ts-1)s-1}y_{ts})$$

which is a quadratic function in binary variables, like in problem (7). The associated quadratic form is also sparse, for similar reasons that the distance minimization case.

2.3 The basic quadratic model

Let us denote by $l(y) = c^T y + \frac{1}{2} y^T H y$ our objective function, where c and Q are chosen according to the specific chosen criteria. Our integer quadratic program with linear constraints (QLC) becomes

$$\begin{aligned} &\text{Minimize} && l(y) = c^T y + \frac{1}{2} y^T H y \\ &\text{subject to} && Ay = b \\ &&& y \in \{0, 1\}^{4n(2n-1)} \end{aligned} \tag{7}$$

This formulation shares the same problem size with the one in Suzuka (2005), namely $O(4n(2n-1))$ optimization variables.

Alternatively, we use the quadratic constraints (2) to define our quadratically constrained (QQC) model:

$$\begin{aligned}
 &\text{Minimize} && l(y) = c^T y + \frac{1}{2} y^T H y \\
 &\text{subject to} && y^T Q_{k1} y + c_{k1}^T y + 1 = 0 \\
 &&& y^T Q_{k2} y + c_{k2}^T y + 1 = 0 \\
 &&& y^T Q_{k3} y + c_{k3}^T y = 0 \\
 &&& y \in \{0, 1\}^{4n(2n-1)}.
 \end{aligned} \tag{8}$$

2.4. A simplified formulation

In Perdomo and Lara (2013), it is used observation that the value of each variable in the group $K(k)$ are determined by fixing one of such variables. This fact leads us to a simplified model with $\frac{1}{4}$ of the original size. In fact, by equations (1) we have

$$\begin{bmatrix} y_{t'_k s_k} \\ y_{t_k s'_k} \\ y_{t'_k s'_k} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} y_{t_k s_k} = b_k - A_{t_k s_k} y_{t_k s_k}.$$

Denoting by $z_k = y_{t_k s_k}$ we write all the variables y in function of $z \in \{0, 1\}^{n(2n-1)}$. For each k ($k=1, \dots, n(2n-1)$), we define the following subsets of the index set $T \times S$: $B(k) = \{(t_k, s'_k), (t'_k, s_k), (t'_k, s'_k)\}$ and $N(k) = \{(t_k, s_k)\}$.

Lemma: Consider the sets $B = \bigcup_{k=1}^{n(2n-1)} B(k)$ and $N = \bigcup_{k=1}^{n(2n-1)} N(k)$. Then $\{B, N\}$ is a partition of the index set $T \times S$.

Proof: In fact, first note that

$$B \cup N = \left(\bigcup_{k=1}^{n(2n-1)} B(k) \right) \cup \left(\bigcup_{k=1}^{n(2n-1)} N(k) \right) = \bigcup_{k=1}^{n(2n-1)} K(k) = T \times S$$

On the other hand, the fact that for $k \neq k'$, $K(k) \cap K(k') = \emptyset$, leads to $B(k) \cap N(k') = \emptyset$, for all $k \neq k'$. So,

$$B \cap N = \left(\bigcup_{k=1}^{n(2n-1)} B(k) \right) \cap \left(\bigcup_{k=1}^{n(2n-1)} N(k) \right) = \bigcup_{k=1}^{n(2n-1)} B(k) \cap N(k)$$

for any k , $B(k) \cap N(k') = \emptyset$, leading $B \cap N = \emptyset$. $\diamond$

By using the partition above, our variables can be rewritten as follows: $y_B = b - A_N z$, and $y_N = z$, which in turn define the linear transformation $Y : R^{n(2n-1)} \rightarrow R^{4n(2n-1)}$, by $Y(z) = y$. The objective function $l(y)$ becomes

$$\bar{l}(z) = l(Y(z)) = \bar{a} + \bar{c}^T z + \frac{1}{2} z^T \bar{H} z,$$

where $\bar{a} = c_B^T b + b^T H_{BB} b$; $\bar{c} = (c - A_N^T c_B - 2A_N^T H_{BB} b - 2H_{NB} b)^T$ and $\bar{H} = [A_N^T H_{BB} A_N + A_N^T H_{BN} + H_{BN} A_N]$. Our equivalent reduced model (QR) is:

$$\begin{aligned}
 &\text{Minimize} && \bar{l}(z) = \bar{a} + \bar{c}^T z + \frac{1}{2} z^T \bar{H} z \\
 &\text{subject to} && z \in \{0, 1\}^{n(2n-1)},
 \end{aligned} \tag{9}$$

which is a 0-1 quadratic programming with $\frac{1}{4}$ of the number of variables, compared with problem (7). The fact that this formulation avoid constraints other than the binary's, makes it

suitable for using some metaheuristics, like genetic algorithms, without the care of generating feasible solution populations. The price we pay for using the simplification is the fill effect of the matrix $\bar{H}$, in which the matrix loses the sparsity property.

2.5 Integer Linear Programming Formulations for HA-assignment problems

Motivated by the existence of robust solvers and the simplicity of linear programming, it is desirable whenever it is possible, to reformulate combinatorial programs into integer linear programs. When there are quadratic relationships between binary variables, this is always possible.

One suitable linear formulation for a quadratic model with binary variables replaces each quadratic term say $h_{ij}x_i x_j$, by a linear term $h_{ij}x_{ij}$, incorporating the following linear constraints into the original model

$$\begin{aligned} -x_{ij} + x_i + x_j &\leq 1 \\ x_{ij} - x_i &\leq 0 \\ x_{ij} - x_j &\leq 0 \\ x_{ij}, x_i, x_j &\in \{0, 1\}. \end{aligned}$$

The effect is to enforce x_{ij} to be 1 only if both x_i and x_j are one, otherwise x_{ij} should be 0. This kind of transformation adds to the model one extra binary variable for each quadratic term, that is if the problem has m variables, then potentially m^2 new binary variables should be incorporated. Fortunately, the quadratic relationships in our model only appear in consecutive slots, so, the number of new variables/constraints is $O(m)$. By using this kind of transformation, the problem (7) is equivalently written as a linear program with linear constraints (LLC):

$$\begin{aligned} &\text{Minimize} && l(y, w) = c^T y + r^T w \\ &\text{subject to} && Ay = b \\ &&& Cy + Dw \leq d \\ &&& (y, w) \in \{0, 1\}^{4n(2n-1)} \times \{0, 1\}^{2n(4n-3)}. \end{aligned} \quad (10)$$

The price we pay for linearization is to add extra variables and constraints to the model. A similar linearization is made in the simplified version (LR):

$$\begin{aligned} &\text{Minimize} && \bar{l}(z, w) = \bar{a} + \bar{c}^T z + \bar{r}^T w \\ &\text{subject to} && \bar{C}z + \bar{D}w \leq d \\ &&& (z, w) \in \{0, 1\}^{n(2n-1)} \times \{0, 1\}^{n^2(2n-1)^2}, \end{aligned} \quad (11)$$

which is a linear integer program with potentially more variables (paying the price of filling the quadratic form)

Other choices for linearizing are possible. For instance, in Easton (2004), a linear relaxation for the problem (7) was proposed, in the case of minimizing the total traveling distance in a single round robin tournament. They propose adding continuous variables, instead of binary variables, obtaining a relaxed model which approximates the underlined quadratic model.

2.6 SDP Formulations

In Suzuka (2005), a SDP relaxation was proposed for HA-assignment problems, based on a MIN-RES-CUT modeling. The problem (7) provides a quadratic programming model in binary variables which leads to a different SDP formulation, which we are addressing in an ongoing work.

3. Numerical Results

Five different integer programming formulations for the same problem are solved with simulated data: A quadratic programs with linear objective constraints QLC (7), a reduced quadratic program QR (9), and a quadratically constrained quadratic program QQC (8); also two integer linear programs (10) and (11). All computations were performed on a PC Intel(R) core(TM) i7-3632 QM, 2.20 GHZ, 64 bits.

At the first experiment we fix the objective function and solve problems with 10 different randomly generated timetables, for each problem size n , with the objective of exploring the performance of the solver in a variety of configurations (timetables). Even sharing the same objective function for each n , we solve 10 distinct problems, because different timetables leads to different instances. For the second experiment we fix a timetable, and then we solve the problem for 10 different objective functions, for each problem size (single configuration). We solve HA-assignment problems for a par number of teams, between 4 and 20. Since our objective is to compare the formulations for the problem, we use a non commercial solver which deal with both, linear and quadratic integer programs, namely SICP \cite{SCIP-09}.

$2n$	Fixing FO			Fixing timetable		
	QLC	QQC	Deviation	QLC	QQC	Deviation
4	835.5	957.7	0.12760	840.2	886,4	0.05212
6	2148.5	2395	0.10292	1845.3	2231	0.17288
8	3244.9	3727.9	0.12956	3332.5	3925,8	0.15113
10	5479.9	6205.8	0.11697	5364.0	6110,8	0.12221
12	7334.4	8730.1	0.15987	7672.3	9263,6	0.17178
14	10380.1	12089.2	0.14137	10757.9	12433,7	0.13478
16	13779.5	15912.5	0.13405	13799.5	15893,6	0.13176
18	16792.9	19349.2	0.13211	17531.8	20705,9	0.15329
20	21204.6	24875.6	0.14757	21869.0	25935,2	0.15678

Table 1: Objective function values for quadratic and linear models with linear constraints, with full and reduced size, vs quadratically constrained quadratic models

In table 1 we present the objective function values for both experiments. The second and fifth column show values for the quadratic model with linear constraints (QLC), while the third and sixth columns are for the model with quadratic constraints (QQC). The remained columns contain relative deviations of the values. The content of second and fifth columns are

shared by the remaining three models (QR, LLC, LR). The difference between linear constrained and quadratic constrained models is possibly because the solver only guarantee locally optimal solutions in problems with quadratic constraints.

2n	QLC fixing timetable				LLC fixing timetable			
	full		reduced		full		reduced	
	mean	SD	mean	SD	mean	SD	mean	SD
4	0.02768	0.01198	0.02247	0.01144	0.07063	0.00445	0.26672	0.62680
6	0.09818	0.01719	0.10523	0.03064	0.18222	0.09887	0.20801	0.09900
8	0.29231	0.07935	0.32632	0.12838	0.54525	0.28502	0.68075	0.42887
10	1.14851	0.29057	1.17092	0.30648	2.75070	0.82891	3.52782	1.07226
12	6.92692	0.74739	6.68803	1.08223	8.08235	2.47076	12.3484	1.84931
14	10.4368	0.99115	9.95315	0.89028	17.2353	3.84266	26.0670	9.45421
16	15.1239	1.65025	13.8831	1.45535	141.171	301.716	1890.36	2953.40
18	20.6103	1.86102	21.6154	7.18129				
20	50.1043	21.6128	60.6731	22.7746				

Table 2: Runtime for the quadratic and linear models, with linear constraints, fixing the timetable

In table 2 we present mean and standard deviation for the runtime, in the instances where the objective function is fixed. The first four columns show results for the full and reduced size quadratic models (7) and (9), while the last four columns are about the linear models (10) and (11). As can be expected, the running times for the reduced version in the quadratic models are consistently smaller. Studying the linear versions, we show results for n from 2 to 8 due to the time consuming of larger problems. Observe that while in small problems (n=2 to 6) the reduced version behaves better than the full size, for larger problems this advantage is lost. This fact is explained by the multiplication effect of the binary variables. Another comparison is between the quadratic models vs linearized models. We use the same solver for quadratic and linear binary programs with the intent of reducing the effect of using particular solvers when comparing the models.

2n	QLC fixing OF				LLC fixing OF			
	full		reduced		full		reduced	
	mean	SD	mean	SD	mean	SD	mean	SD
4	0.01881	0.00349	0.02049	0.00817	0.07151	0.00443	0.06999	0.00469
6	0.10556	0.03898	0.09833	0.02711	0.24210	0.10551	0.21468	0.08341
8	0.22068	0.06098	0.24719	0.07874	0.72274	0.30959	0.58631	0.40193
10	1.02054	0.30153	1.15223	0.54655	2.55786	1.35314	3.15127	1.08560
12	6.27446	1.73542	6.07395	1.71097	6.31574	1.45673	9.04688	2.50039
14	10.9502	1.57499	9.91357	0.59958	18.4189	4.30626	125.277	286.680
16	14.5375	1.38106	13.5211	1.30986	131.309	306.120	903.701	306.196
18	21.3364	1.86914	21.9286	6.80267				
20	60.8347	19.6424	41.4042	22.7594				

Table 3: Runtime for the quadratic and linear models with linear constraints, fixing the objective function

Now we shall study table 3, which provides runtime for our integer quadratic and linear programs, with full and reduced size configurations, but fixing the structure (fixing a timetable). In each case we solve for 10 different objective functions, for each problem size. There is no clear evidence of advantage in choosing the reduced version. Sometimes is faster and in other times slower. The comparison between the quadratic and linear models also suggest that quadratic models are stronger.

The Table 4 deal with runtime for both experiments, but for the quadratically constrained model (QQC). When compared with table 2 and table 3, we observe a huge difference in the performance. The price we pay is that global optimization is not guaranteed.

$2n$	Fixing OF		Fixing Timetable	
	mean	sd	mean	sd
4	0.02107	0.00155	0.02079	0.00139
6	0.03974	0.00286	0.03858	0.00298
8	0.06538	0.00474	0.06054	0.00273
10	0.10066	0.00444	0.10122	0.00608
12	0.15408	0.00958	0.14684	0.00681
14	0.22715	0.00974	0.21647	0.00427
16	0.30999	0.00893	0.30643	0.00908
18	0.41958	0.01458	0.42272	0.01283
20	0.56785	0.00973	0.57336	0.00966

Table 4: Runtime for the quadratically constrained model (QQC), fixing the objective function and the timetable

4. Conclusion

In this work we study the effect that different integer linear and quadratic equivalent formulations for the HA-assignment problem have over the solver's performance. By manipulating the special structure of the constraints, we propose a quadratic formulation with 1/4 of the original size. Then we linearize both of the above formulations, obtaining two more versions.

We compare the formulations by showing the runtime of the solver SCIP, in each version. For the set of problems with the same structure (timetable) we observe, as can be expected, that the reduced versions behave better for the quadratic formulations. We also note that the quadratic version behaves much faster than the linear ones. This shows that even the linear models are equivalent to the quadratic ones, they make the solver behave worst, due to the multiplication of the number of variables. This fact provides evidence that using the model in its natural form sometimes brings advantages.

The solver SICP behaves very fast for the model with quadratic constraints, but with the disadvantage of obtaining locally optimal solutions. Sometimes we need just this kind of solutions, so these cheap solutions are sometimes desirables.

In Suzuka (2005) a SDP relaxation was proposed for HA-assignment problems, based on a MIN-RES-CUT modeling. The problem (\ref{f3}) provides a quadratic programming model

in binary variables which lead to a different SDP formulation, which we are addressing in an ongoing work.

Acknowledgment

This work is supported in part by the Brazilian Government, through the Excellence Fellowship Program CAPES/IMPA of the first author while visiting the department of Mathematics at UFPR, Curitiba, Brazil.

Referências

- EASTON, N. G., and TRICK, M.** *The traveling tournament problem: Description and Bench-marks*. In Walsh, T. Editor. Principles and Practice of Constraint Programming, Vol 2239 of Lecture Notes in Computer Science. Springer, Berlin, 2001.
- EASTON, N. G., and TRICK, M.** *Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach*. In Leung, J. T, and Anderson, J. H. Editors. In Handbook of Scheduling: Algorithms, Models and Performance Analysis Chapman and Hall, 2004.
- ELF, M. JUNGER, M., and RINALDI, G.** *Minimizing Breaks by Maximizing Cuts*. Operations Research Letters. 31: 343-349.
- MIYASHIRO, R. and MATSUI, T.** *Semidefinite Programming Based Approaches to the Break Minimization Problem*. Computers and Operations Research, 33, 2007.
- PERDOMO, J. and LARA, H.** *A Combinatorial Optimization Formulation for the HA-Assignment Problem*. Publicaciones en Ciencias y Tecnología, 7(2):127-141, 2013.
- RIBEIRO, C.** *Sport Scheduling: A Tutorial on Fundamental Problems and Applications*. Technical report. Department of Computer Science, Universidade Federal Fluminense, RJ. Brazil, 2010.
- SUZUKA, M. R. and MATSUI, T.** *Semidefinite Programming Based Approaches for Home Away Assignment Problems in Sports Scheduling*. Mathematical Engineering Technical Reports, Department of Mathematical Informatics, The University of Tokyo, Japan, 2005.
- TRICK, M.** *A Schedule-Then-Break Approach to Sport Timetabling*. Technical Report, GSIA Carnegie Mellon, Pittsburg. URL <http://mat.gsia.cmu.edu>, 2006.