Grouping sport teams into round robin competitions

Túlio A. M. Toffolo · Jan Christiaens · Frits C. R. Spieksma · Greet Vanden Berghe

Abstract The sport teams grouping problem (STGP) concerns the assignment of teams to round-robin tournaments. The primary objective is to minimize the total travel distance of the participating teams while simultaneously respecting fairness constraints. Three integer programming formulations are presented: two compact formulations and another with an exponential number of variables. Additionally, meta-heuristic methods are applied to generate feasible high-quality solutions.

Keywords Sport teams grouping problem · Branch-and-price · Column generation · Decomposition strategies · Integer programming · Metaheuristic

1 Introduction

Both professional and recreational sports competitions require large organizational efforts. Despite the huge budgets involved, the actual impact of organizational decisions on players, supporters and potential game outcomes remain difficult to model and assess. In addition, the decisions to be made – the combinatorial problems requiring solutions –, vary considerably in nature.

The timetabling community has a history of welcoming efforts to investigate academic versions of real world sport scheduling problems. Examples include the traveling tournament (Easton et al., 2001; Nemhauser and Trick, 1998) and traveling umpire (Trick and Yildiz, 2007). Such initiatives have resulted in the formation of an international community (Kendall et al., 2010) defining new academic problems related to sports timetabling (Ribeiro and

Túlio A. M. Toffolo[1,3] · Jan Christaens[1] · Frits C.R. Spieksma[2] · Greet Vanden Berghe[1]
[1] KU Leuven - Belgium, Department of Computer Science, CODeS & iMinds-ITEC
[2] KU Leuven - Belgium, Faculty of Economics and Business, ORSTAT
[3] Federal University of Ouro Preto - Brazil, Department of Computing

Urrutia, 2007) and generating state of the art algorithms for the traveling tournament problem (Anagnostopoulos et al., 2006; Gaspero and Schaerf, 2007), and the traveling umpire problem (Toffolo et al., 2016; Trick and Yildiz, 2011; Trick et al., 2012; Xue et al., 2015).

The present paper's subject further contributes to this line of sports scheduling research by introducing an academic version of a relevant and challenging optimization problem which occurs in many regions and impacts a large number of youth players. The sport teams grouping problem (STGP) addressed in the present paper concerns the assignment of teams to round-robin tournaments, wherein the primary objective is to minimize the total travel distance of the teams while simultaneously respecting fairness constraints.

Prior to the start of each season, clubs enroll one or more youth teams to participate in leagues. Each team has a location (related to its club) and is assigned a certain level, given by $l \in \{1, 2, 3\}$. These levels denote a team's estimated strength, with level '1' used to denote the strongest teams. A league may contain teams from different levels, so long as the difference in level for each pair of teams in a league is at most 1. Furthermore, each league must comprise of between m^- and m^+ teams, where m^- and m^+ are problem parameters. It is undesirable for teams from the same club to play in the same league, however up to c^+ teams from the same club are permitted to do so, where c^+ is also a parameter – usually set to 2. Additionally, no team may travel more than a given time limit t^+ to another team's home venue.

The STGP is a challenge faced by numerous sport associations. The Royal Belgian Football Association (RBFA), for instance, organizes youth football leagues. Within the East Flanders province alone, these leagues comprise more than 500 youth teams playing approximately $5,000$ matches. Therefore, reducing the total travel time and distance is very desirable, both logistically and economically. Observe also that the RBFA faces multiple instances of the STGP: there are round-robin tournaments to be constructed for each age category of each province. In addition to being highly relevant in practice, the STGP is also challenging from a computational perspective since it generalizes clique partitioning with minimum clique size requirement and is, consequently, an NP-Hard problem.

Three integer programming formulations are presented: two compact formulations and one with an exponential number of variables derived from Ji and Mitchell (2007). The first two formulations are solved by an integer programming solver (Gurobi), while the last is solved by a tailor-made branch-and-price algorithm. Gurobi is employed to solve the master problem, whereas a specialized heuristic handles the column generation pricing problem. Additionally, a meta-heuristic method is applied to generate feasible high-quality solutions for large instances.

The present paper is organized as follows. The next section offers a more detailed description of the STGP and positions it in relation to other problems in the literature. Section 3 presents the two compact integer linear programming formulations considered. Section 5 details the formulation with an exponential number of variables, solved by column generation, and introduces the

heuristic employed to solve the pricing problem. Section 6 explains the developed branch-and-price approach. Computational experiments are presented in Section 9 and, finally, Section 10 summarizes the conclusions.

2 The sport teams grouping problem

Given a set T of teams and travel times between them, the STGP consists of distributing these teams within leagues while minimizing the total travel time. Each team $t \in T$ is assigned a level l_t and a club $k_t \in K$, where K is the set of different clubs. Each league must contain between m^- and m^+ teams. At most c^+ teams from the same club are allowed in a league, and teams within a league must not be further than t^+ time units away from each other.

Essentially, the STGP can be described by a graph $G = (V, E)$, where each vertex represents a team ($V = T$). Two vertices are connected by an edge $e \in E$ if the difference in level between teams does not exceed 1 and the time required to travel between them is not greater than t^+. The weight of an edge is given by the necessary time for a trip between the connected teams' locations. The objective is to partition the vertex set V into a set of cliques whose size respects both the minimum and maximum limits given by m^- and m^+ respectively, while minimizing the total weight of the cliques' edges.

Figure 1 shows a solution for a real-world STGP faced by the Royal Belgian Football Association. The problem depicted by the figure contains 521 teams distributed in 169 clubs and 4 different age categories. In this example, $m^- = 5$, $m^+ = 10$ and $c^+ = 2$.

The STGP is a generalization of the clique partitioning problem with minimum requirement (Ji and Mitchell, 2007; Labbé and Özsoy, 2010). The main difference between the STGP and the classical clique partitioning problem with minimum clique size requirement is that the STGP imposes a limit on the number of teams from the same club in a league (or clique), in addition to a maximum clique size requirement.

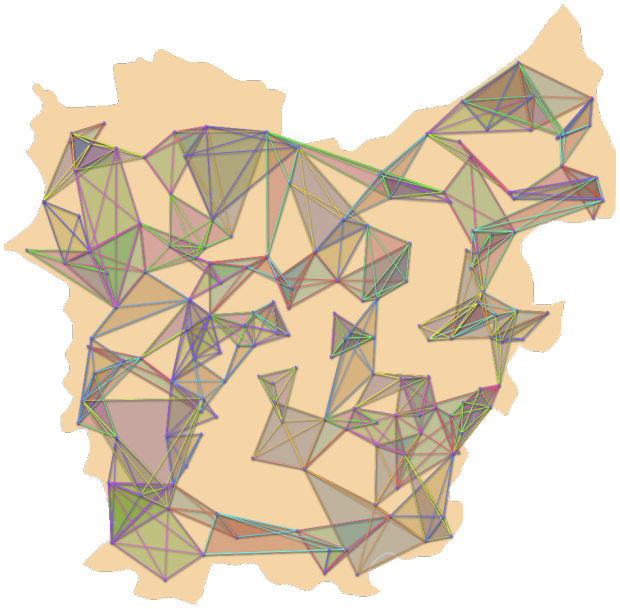

Fig. 1 Illustration of a solution for a STGP instance

3 Integer programming formulations

This section presents three integer (binary) linear programming formulations for the STGP: two compact formulations and one with an exponential number of variables. Before proceeding to detailed descriptions of each formulation, some additional input data is introduced:

L : set of leagues $L = \{1, 2, ..., |L|\}$, where $|L| = \lfloor |T|/m^{-} \rfloor$ is the maximum number of leagues;
T_k : subset of teams $T_k \subseteq T$ with all teams from club $k \in K$;
Φ : set of team pairs (i, j) that cannot be in the same league (due to level difference or travel time constraints);
t_{ij} : travel time between teams i and j.

3.1 Compact formulation 1

The first compact formulation considers three sets of variables:

x_{il} : binary variable that assumes value 1 if team i is assigned to league l and 0 otherwise;
y_{ij} : binary variable that assumes value 1 if teams i and j are assigned to the same league and 0 otherwise;
z_l : binary variable that assumes value 1 if league l is used and 0 otherwise.

Equations (1)-(7) present the problem formulation.

$$\text{minimize} \quad \sum_{i \in T} \sum_{j \in T} t_{ij} y_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{l \in L} x_{il} = 1 \qquad \forall\, i \in T \tag{2}$$

$$m^- z_l \leq \sum_{i \in T} x_{il} \leq m^+ z_l \qquad \forall\, l \in L \tag{3}$$

$$\sum_{i \in T_k} x_{il} \leq c^+ \qquad \forall\, l \in L, k \in K \tag{4}$$

$$x_{il} + x_{jl} \leq y_{ij} + 1 \qquad \forall\, i \in T, j \in T, l \in L \tag{5}$$

$$y_{ij} = 0 \qquad \forall\, (i,j) \in \Phi \tag{6}$$

$$x_{il}, y_{ij}, z_l \in \{0,1\} \qquad \forall\, i \in T, j \in T, l \in L \tag{7}$$

The objective function (1) minimizes the total travel time. Constraints (2) guarantee each team is assigned to exactly one league. Constraints (3) activate variables z while verifying that the number of teams in a selected league $l \in L$ respects lower and upper limits m^- and m^+, respectively. Note that these constraints also guarantee that if a league $l \in L$ is not used ($z_l = 0$), then $x_{il} = 0\ \forall i \in T$. Constraints (4) ensure that the number of teams from the same club in a league does not exceed limit c^+. Constraints (5) activate variables y, ascertaining that y_{ij} is 1 if teams i and j are assigned to the same league and 0 otherwise. Constraints (6) prevent allocations of two incompatible teams in the same league and, finally, Constraints (7) declare that variables x, y and z are binary.

The linear relaxation of formulation (1)-(7) provides trivial lower bound of value zero for most instances, since Constraints (5) are easily satisfied if $x_{il} = x_{jl} = 0.5$, consequently permitting $y_{ij} = 0$. To cut away this fractional solution, valid inequalities are introduced:

$$m^- - 1 \leq \sum_{j \in T} y_{ij} \leq m^+ - 1 \quad \forall\, i \in T \tag{8}$$

Constraints (8) certify that each team has at least $m^- - 1$ adversaries in a league (and at most $m^+ - 1$). This condition cuts away solutions with $y_{ij} = 0\ \forall i \in T$, considerably improving the quality of the lower bound provided by the linear relaxation.

Additionally, triangle inequalities may also be added (Constraints (9)). However, experiments revealed that they deteriorate the performance of the employed solver (Gurobi) for the presented formulation.

$$y_{ij} + y_{jh} \leq y_{ih} + 1 \quad \forall\, i \in T, j \in T, h \in T \tag{9}$$

3.2 Compact formulation 2

The second compact formulation presented for the STGP is quite similar to the first, except that different variable sets are considered:

x_{il} : binary variable that assumes value 1 if team i is assigned to league l and 0 otherwise;

y_{ijl} : binary variable that assumes value 1 if teams i and j are both assigned to league l and 0 otherwise;

Note that in contrast to the variables y employed in the first formulation, y defines not only if two teams are together in a league, but also the league to which they are assigned. This considerably increases the total number of variables but may impact the branching and overall performance of Gurobi.

$$\text{minimize} \quad \sum_{i\in T}\sum_{j\in T}\sum_{l\in L} t_{ij}y_{ijl} \tag{10}$$

$$\text{subject to} \quad \sum_{l\in L} x_{il} = 1 \qquad \forall\, i \in T \tag{11}$$

$$\sum_{j\in T} y_{ijl} \geq (m^- - 1)x_{il} \qquad \forall\, i \in T, l \in L \tag{12}$$

$$\sum_{j\in T} y_{ijl} \leq (m^+ - 1)x_{il} \qquad \forall\, i \in T, l \in L \tag{13}$$

$$\sum_{i\in T_k} x_{il} \leq c^+ \qquad \forall\, l \in L, k \in K \tag{14}$$

$$x_{il} + x_{jl} \leq y_{ijl} + 1 \qquad \forall\, i \in T, j \in T, l \in L \tag{15}$$

$$y_{ijl} = 0 \qquad \forall\, (i,j) \in \Phi, l \in L \tag{16}$$

$$x_{il}, y_{ijl} \in \{0,1\} \qquad \forall\, i \in T, j \in T, l \in L \tag{17}$$

Except for the different set of variables and a different constraint to ensure the leagues size, formulations (10)-(17) and (1)-(7) are rather similar. The objective function (10) minimizes the total travel time. Constraints (11) guarantee each team is assigned to exactly one league. Constraints (12) and (13) certify that each team is in a league with at least $m^- - 1$ and at most $m^+ - 1$ other teams, ensuring that league sizes are between m^- and m^+ teams. Constraints (14) limit the number of teams from the same club in a league to c^+. Constraints (15) activate variables y, applying a value of 1 to y_{ij} if teams i and j are assigned to the same league and 0 otherwise. Constraints (16) prevent allocations of two incompatible teams to the same league and, finally, Constraints (17) declare that variables x and y are binary.

3.3 Formulation with exponential number of variables

A formulation for the STGP that considers each feasible league (or clique) as a variable is presented by Equations (18)-(20). This formulation is a standard

set partitioning formulation, where Ω is the set of possible leagues (columns), t'_c the total travel time of league c and α_{ic} indicates whether league c contains team i ($\alpha_{ic} = 1$) or not ($\alpha_{ic} = 0$). Finally, λ_c is a binary variable that is 1 if league c is selected and 0 otherwise.

$$\text{minimize} \quad \sum_{c \in \Omega} t'_c \lambda_c \tag{18}$$

$$\text{subject to} \quad \sum_{c \in \Omega} \alpha_{ic} \lambda_c = 1 \quad \forall\, i \in T \tag{19}$$

$$\lambda_c \in \{0, 1\} \qquad \forall\, c \in \Omega \tag{20}$$

The objective function (18) minimizes the total travel time. Constraints (19) ensure each team is assigned to exactly one selected league, and Constraints (20) state that variables λ are binary.

When the number of teams $|T|$ is not a multiple of the minimum league size, m^-, the linear relaxation of (10)-(17) has a greater probability of producing fractional solutions, as observed by Ji and Mitchell (2007). Since larger leagues have a higher number of edges, they are generally associated with larger costs. In the linear relaxation these larger leagues can be replaced by a fractional combination of smaller leagues, which often yields a better objective value. Constraint (21) strengthens the formulation by limiting the total number of selected leagues to $\lfloor |T|/m^- \rfloor$, reducing the replacement of large leagues for fractional smaller leagues.

$$\sum_{c \in \Omega} \lambda_c \leq \left\lfloor \frac{|T|}{m^-} \right\rfloor \tag{21}$$

Note that, since m^- is the minimum size of a league, a solution for a problem with $|T|$ teams can have at most $\lfloor |T|/m^- \rfloor$ leagues. Section 9 shows the impact of adding this inequality to formulation (18)-(20).

4 Symmetry breaking

Formulations (1)-(7) and (10)-(17) contain many symmetric solutions. All leagues $l \in L$ are identically defined, with the subsequent permuting of leagues generating symmetric solutions. It is well known that symmetry has a very negative impact on branch-and-bound algorithms and should therefore be avoided.

To reduce symmetry, a set $\tilde{T}$ of teams which cannot coexist in the same league is fixed to different leagues. Such sets can be easily obtained. Let $G = (V, E)$ be the graph representation of a problem, where vertices represent teams and edges connect teams permitted within the same league. Any independent set of vertices in G is a valid set $\tilde{T}$ that may be used to reduce symmetry. Ideally, $\tilde{T}$ should be a maximum independent set of G.

5 Column generation approach

Explicitly implementing formulation (18)-(20) requires an exponential number of variables. To avoid this, a column generation scheme (Lübbecke and Desrosiers, 2005; Vanderbeck and Wolsey, 2010) is employed in which variables are iteratively generated based on dual costs.

The column generation approach is solved iteratively. A restricted master problem (RMP) is defined, consisting initially of formulation (18)-(21) with $\Omega = \emptyset$. Artificial variables with sufficiently high costs in the objective function are introduced to satisfy Constraints (19). The linear relaxation of the restricted master problem is subsequently solved. At each iteration, the pricing problem is solved to obtain columns (or leagues) with negative reduced cost considering the current dual values γ and τ, corresponding to Constraints (19) and Constraint (21) respectively. A column $c \in \Omega$ has negative reduced cost if $t'_c - \sum_{i \in T} \alpha_{ic}\gamma_i - \tau < 0$. If such columns are found, they are added to the restricted master problem, which is subsequently re-solved. The algorithm continues until no columns with negative reduced cost can be found, whereupon the linear relaxation of the master problem is solved.

5.1 Pricing problem

Given dual variables γ and τ, the pricing problem consists of finding a feasible league (column) such that $t'_c - \sum_{i \in T} \alpha_{ic}\gamma_i - \tau < 0$. This represents the problem of obtaining a league with an objective function value smaller than τ considering that a time-cost γ_i is associated with each team i.

The pricing problem can be formulated as an integer program. The input data introduced in Section 3 is considered in addition to dual values γ and τ related to Constraints (19) and (21), respectively. Two variable sets are defined:

x_i : binary variable that assumes value 1 if team i is selected and 0 otherwise;
y_{ij} : binary variable that assumes value 1 if teams i and j are both selected and 0 otherwise;

The formulation is given by Equations (22)-(27).

$$\text{minimize} \quad \sum_{i \in T} \sum_{j \in T} t_{ij} y_{ij} - \sum_{i \in T} \gamma_i x_i - \tau \tag{22}$$

$$\text{subject to} \quad m^- \leq \sum_{i \in T} x_i \leq m^+ \tag{23}$$

$$\sum_{i \in T_k} x_i \leq c^+ \qquad \forall\, k \in K \tag{24}$$

$$x_i + x_j \leq y_{ij} + 1 \qquad \forall\, i \in T, j \in T \tag{25}$$

$$y_{ij} = 0 \qquad \forall\ (i,j) \in \Phi \qquad (26)$$

$$x_i, y_{ij} \in \{0,1\} \qquad \forall\ i \in T, j \in T \qquad (27)$$

The objective function (22) minimizes the travel time while considering dual costs γ and τ. Note that a solution for this pricing problem is a column with negative reduced cost when its objective value is negative. Constraint (23) ensures that the size of the produced league is between m^- and m^+. Constraints (24) limit the number of teams from the same club in a single league to no more than c^+. Constraints (25) establish that y_{ij} should be 1 whenever both teams i and j are selected. Constraints (26) prevent the selection of two incompatible teams to the same league and, finally, Constraints (27) define variables x and y as binary.

Analogously to formulation (1)-(7), the linear relaxation of formulation (22)-(27) generates weak bounds, since it is possible that $y_{ij} = 0\ \forall i, j \in T$. Valid inequalities (28) are added to prevent this:

$$m^-(m^- - 1) \leq \sum_{i \in T} \sum_{j \in T} y_{ij} \leq m^+(m^+ - 1) \quad \forall\ i \in T \qquad (28)$$

Constraints (28) ensure at least $m^-(m^- - 1)$ and at most $m^+(m^+ - 1)$ games are played in a league. These constraints cut away the fractional solutions with $y_{ij} = 0\ \forall i, j \in T$.

6 Branch-and-price algorithm

The column generation algorithm (Section 5) only solves the linear relaxed version of the problem. The solutions produced may be fractional, and thus it may prove necessary to branch on the variables to find an integer solution. When this occurs, a branch-and-price algorithm (Barnhart et al., 1998; Vanderbeck, 2000) is applied. It is a variant of branch-and-bound where the relaxation is solved by column generation in each search tree node.

Once the linear relaxation of (18)-(20) is solved, branching begins. If the solution is fractional, there exists a fractional λ variable. Therefore, there exist two teams i and j such that the sum of the λ variables containing both i and j is (strictly) between 0 and 1. Clearly, in an optimal solution either these two teams are in the same league or not. If they are not, y_{ij} is fixed to 0. Otherwise, the pricing problem (22)-(27) is extended with $x_i = x_j$, which alters the problem but ensures that teams i and j are assigned to the same league. Figure 2 depicts the branching scheme.

The selection of the fractional variable to branch is a critical component of branch-and-bound algorithms (Achterberg et al., 2005). Ideally, the pair of teams i and j that incur the largest objective function increases for both branches ($y_{ij} = 0$ and $x_i = x_j$) should be selected. However, detecting such teams may be very time consuming. In the developed approach, a heuristic criterion is employed: the most distant teams i and j are selected. This criterion

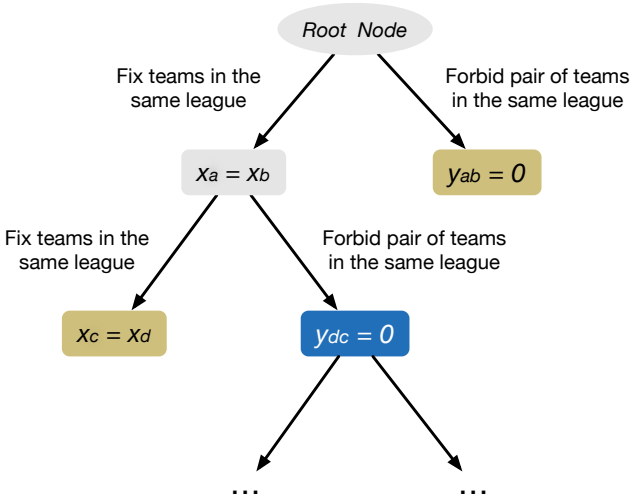

Fig. 2 Illustration of the branching scheme employed in the branch-and-price algorithm

yielded better results than, for instance, selecting the pair of teams with the most fractional sum of related λ.

A parallel branch-and-price algorithm was developed in which multiple threads analyze nodes concurrently. The different threads compete to update bounds and add new nodes to the heap, rendering the algorithm's execution non-deterministic.

7 Heuristic approach to the pricing problem

The pricing problem is either optimally solved, by a general integer programming solver for example, or addressed heuristically without convergence proof. Although heuristics lack convergence proof, it is worthwhile investigating whether they require less computational time to generate one or more columns (here leagues) with negative reduced cost than integer programming solvers.

The heuristic is provided with a partial solution in which a set of team-to-league assignments are fixed. The heuristic is initialized by first creating a new clique for every team. If two teams of the pricing problem are fixed to separate leagues, the cliques corresponding with these teams are marked as conflicting ones. Contrastingly, if two teams are fixed to the same league, the corresponding cliques are merged. The cost of a clique is defined as the sum, over all pairs of teams within the clique, of their corresponding edge weights. Similarly, the cost of merging two cliques or assigning them to the same league is defined as the sum, over all pairs of teams belonging to different cliques, of their corresponding edge weights. By monitoring these costs, the initialization procedure enables fast evaluation during the subsequent local search step where clique and team lists are updated.

The local search procedure begins by sorting all cliques in order of descending dual values, thus providing a sequence for selecting the first clique inside an empty league. This initial league is iteratively improved by adding the best neighboring solution of a randomly selected neighborhood. The set of local search neighborhoods includes:

1. *add teams neighborhood*: adds, if possible, a clique (set of teams) to a compatible league without exceeding the maximum league size.
2. *remove teams neighborhood*: removes, if possible, a clique (set of teams) without violating the minimum league size.
3. *swap teams neighborhood*: greedily replaces the clique (set of teams) adding the most to the objective function value with the clique that would incur the smallest increase.

The exploration of neighborhoods is randomized by rejecting, with a small probability α, better neighboring solutions than the best found so far. While iterating over the neighborhoods, all feasible leagues with negative cost are saved and ordered by increasing cost. The local search procedure ends when a given iteration limit is reached. If no league with negative reduced cost has been generated before the iteration limit, additional computation time is granted to the heuristic, thus potentially avoiding the necessity to call the integer programming solver, which requires a substantial amount of time to converge.

Algorithm 1 depicts the heuristic procedure. Initially, each *clique* consists of a single team. While the branching (of the branch-and-price) is executed, teams are fixed to the same league or forced to be in separate leagues. Fixing two teams to the same league is equivalent to merging these teams' *cliques* into a single *clique*. The algorithm requires four arguments: (*i*) a list with all cliques sorted by their dual cost, (*ii*) a function that calculates a league's cost considering the dual values, (*iii*) the probability α and (*iv*) a maximum number m of iterations per clique. The algorithm begins by initializing the list of leagues L and the pointer p to the current clique (lines 1-2). Next, the current league l is initialized with the teams of the selected clique (line 4) and the pointer p is updated (line 5). For a limited number of iterations, the algorithm performs local search using the neighborhoods presented. First, one of the neighborhoods is randomly selected (line 7). Second, a feasible neighbor is produced considering the current league l and the parameter α (line 8). Note that the acceptance of improving neighbors is prevented with probability $1-\alpha$. If a novel league (not in L) is generated, it is accepted (lines 9-10). In case the league also represents a negative reduced cost column, it is added to set L (lines 11-12). Finally, the set of produced columns is returned (line 13).

Whenever the heuristic fails to provide a column with negative reduced cost within its computation time limit, formulation (22)-(27) is solved to either find a column with negative reduced cost or prove the nonexistence of such column.

Algorithm 1: Pricing heuristic

```
Let C be a list with all cliques sorted by their dual cost
Let f(.) be a function that returns the cost of league (∞ if league is infeasible)
Let α be the probability of only accepting an improving neighbor
Let m be the maximum number of iterations per clique
PricingSolver(C, f(.), α, m):
1   L ← ∅
2   p ← 0
3   while stopping criterion not met do
4       l ← C_p
5       p ← p + 1
6       while limit of iterations m not met do
7           N ← random neighborhood
8           l' ← feasible solution in neighborhood N(l, α)
9           if l' ∉ L then
10              l ← l'
11              if f(l) < 0 then L ← L + {l};
12  return L
```

8 Metaheuristic approach

A simulated annealing heuristic was developed to quickly find good upper bounds. First an initial solution is constructed by assigning leagues to all teams based on a randomized best insertion method. The procedure iterates over all existing leagues which are compatible with the team. Whenever a league yields a lower insertion cost for a team, it is accepted with a probability of 80% as the best league assignment for a team. If no league is accepted, then a new league is created including only this team. This constructive method does not ensure feasibility of the constructed solution because some leagues may contain fewer teams than required. Next, it is iteratively improved by ruining and recreating the solution as follows. A random number of leagues in the current solution is selected. For each of these leagues a random number $n \in \{1, 2\}$) is chosen. To prevent many leagues from becoming infeasible, n is reduced, with a probability of 90%, to the maximum number of removed teams such that the minimum number of teams is retained. Thereafter, new separate leagues are created for a fraction $f = 1/m^-$ of all removed teams. This construction method tries to insert all remaining teams, considering only leagues with a shortage of teams while not creating new leagues. Finally the remaining teams are assigned to leagues by considering the full solution (again, no new leagues are created). If not all removed teams are reassigned, the neighbor solution is rejected and the original solution is returned. The neighbor solution created by the ruin and recreate procedure may be accepted by simulated annealing. The system is cooled down by simulated annealing from $t_0 = 100$ to $t_1 = 1$ in 80 million iterations. The cooling rate γ is thus given by Equation (29), where t_0 is the initial temperature, t_1 the final temperature and I the number

of iterations.

$$\gamma = \left(\frac{t_1}{t_0}\right)^{\frac{1}{I-1}} \tag{29}$$

Algorithm 2 presents the simulated annealing procedure. The procedure requires four arguments: (*i*) initial solution, (*ii*) initial temperature, (*iii*) final temperature and (*iv*) cooling rate. Additionally, let $f(.)$ be the function that returns the objective value of a solution and $g(.)$ the function that returns the number of infeasible leagues of a solution. Initially, the best solution S^* and current temperature t are initialized (lines 1-2). While the final temperature is not reached (line 3), neighboring solutions are generated (line 4) and evaluated. Note that solutions with higher number of infeasible leagues are always rejected (line 5). Otherwise, if the produced solution improves upon the previous one (lines 6-7), it is accepted (line 8). If it improves the best one, then the best solution is also updated (line 9). Worsening solutions may be accepted with a certain probability (lines 11-12). The temperature is updated (line 13) and the algorithm proceeds to the next iteration. Once the final temperature is reached, the best solution generated is returned (line 14).

Algorithm 2: Simulated Annealing

Let S be the initial solution
Let t_0 and t_1 be the initial and final temperatures, respectively
Let γ be the cooling rate, $\gamma \in [0, 1]$
SimulatedAnnealing(S**,** t_0**,** t_1**,** γ**):**

```
1  S* ← S
2  t ← t0
3  while t > t1 do
4      S' ← solution obtained by applying ruin-and-recreate on S
5      if g(S') ≤ g(S) then
6          Δ ← f(S') − f(S)
7          if Δ ≤ 0 then
8              S ← S'
9              if f(S') < f(S*) then S* ← S';
10         else
11             take a random x ∈ [0, 1]
12             if x < e^{−Δ/t} then S ← S';
13     t ← γ × t
14 return S*
```

9 Computational experiments

Computational experiments were conducted to evaluate the performance of the three formulations proposed. The running time for solving each instance

set was limited to 8 hours. The test system was an Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60GHz computer with 128GB of RAM memory running Ubuntu Linux 14.04 LTS.

Real-world instances were provided by Movetex[1]. In addition to the real-world problems, artificial instances were generated to evaluate the behavior of the different approaches on problems with both fewer and more teams. All instances are available online[2] so as to encourage future research on the STGP.

Table 1 shows the results obtained within 3 hours for the two compact formulations presented in Section 3. $|T|$, m^- and m^+ are the instance characteristics. For each formulation the value of the linear relaxation (LB^0), the best lower bound obtained (LB^*), the best solution (UB), the runtime and the resulting gap are presented. Best results are highlighted in bold. Note that the second formulation resulted in better bounds and solutions for most instances. It enables solving one instance more than the first formulation. For larger instances (u-13, u-15 and u-17), however, the first formulation resulted in better feasible solutions.

Table 1 Results obtained with the compact formulations

Inst.	$\|T\|$	m^-	m^+	1st Formulation					2nd Formulation				
				LB^0	LB^*	UB	Time	Gap	LB^0	LB^*	UB	Time	Gap
tiny	15	5	10	550.00	**opt**	**724**	1.0s	0.0%	550.00	**opt**	**724**	1.1s	0.0%
		6	10	936.09	**opt**	**1438**	0.2s	0.0%	1076.49	**opt**	**1438**	0.2s	0.0%
		7	10	1128.91	**opt**	**1438**	0.2s	0.0%	1288.09	**opt**	**1438**	0.2s	0.0%
small	50	5	10	996.00	1044.00	1202	10800.0s	13.1%	996.00	**1130.00**	**1196**	10800.0s	5.5%
		6	10	1382.00	1476.00	1768	10800.0s	16.5%	1382.00	**1602.00**	**1762**	10800.0s	9.1%
		7	10	1806.00	1914.00	2314	10800.0s	17.3%	1806.00	**2160.00**	**2312**	10800.0s	6.6%
		8	10	2258.00	2440.00	3120	10800.0s	21.8%	2258.00	**2810.00**	**3074**	10800.0s	8.6%
		9	10	2748.00	3182.00	**4190**	10800.0s	24.1%	2748.00	**3550.00**	**4190**	10800.0s	15.3%
		10	10	3276.00	3552.00	**4190**	10800.0s	15.2%	3276.00	**opt**	**4190**	3033.9s	0.0%
u-13	166	5	8	1841.00	**1842.00**	**4674**	10800.0s	60.6%	1841.00	**1842.00**	13016	10800.0s	85.8%
		6	8	2528.00	**2528.00**	19704	10800.0s	87.2%	2528.00	**2528.00**	**13016**	10800.0s	80.6%
		7	8	3290.00	**3290.00**	19168	10800.0s	82.8%	3290.00	**3290.00**	**13016**	10800.0s	74.7%
		8	8	-	**infeasible**		15.0s	-	4126.00	4140.00	-	10800.0s	-
u-15	167	5	8	1799.00	**1800.00**	**9256**	10800.0s	80.6%	1799.00	**1800.00**	14060	10800.0s	87.2%
		6	8	2507.00	**2508.00**	18984	10800.0s	86.8%	2507.00	**2508.00**	**13638**	10800.0s	81.6%
		7	8	3268.00	**3268.00**	**11552**	10800.0s	71.7%	3268.00	**3268.00**	19502	10800.0s	83.2%
		8	8	-	**infeasible**		11.4s	-	4090.00	4098.00	-	10800.0s	-
u-17	130	5	8	1637.00	**1638.00**	**5202**	10800.0s	68.5%	1637.00	**1638.00**	10910	10800.0s	85.0%
		6	8	2237.00	2238.00	**7188**	10800.0s	68.9%	2237.00	**2246.00**	12356	10800.0s	81.8%
		7	8	2889.00	2890.00	**8988**	10800.0s	67.8%	2889.00	**2892.00**	15228	10800.0s	81.0%
		8	8	-	**infeasible**		4.8s	-	3612.00	3616.00	-	10800.0s	-
u-21	58	5	8	1135.00	1148.00	1932	10800.0s	40.6%	1135.00	**1158.00**	**1478**	10800.0s	21.7%
		6	8	1546.00	1562.00	2832	10800.0s	44.8%	1546.00	**1628.00**	**1988**	10800.0s	18.1%
		7	8	2018.00	2088.00	2516	10800.0s	17.0%	2018.00	**2118.00**	**2616**	10800.0s	19.0%
		8	8	-	**infeasible**		0.5s	-	2556.00	2672.00	-	10800.0s	-

[1] http://movetex.be

[2] Instance and solution files are available at http://benchmark.gent.cs.kuleuven.be/

Table 2 Results obtained with the branch-and-price and simulated annealing algorithms

Inst.	$\|T\|$	m^-	m^+	Best Compact Form.			Branch-and-Price					SA	Gap
				LB^*	UB	Time	LB^a	LB^0	LB^*	UB	Time		
tiny	15	5	10	**0.00**	**724**	1.0s	704.00	704.00	**opt**	**724**	1.2s	1438	0.0%
		6	10	**0.00**	**1438**	0.2s	1019.00	**opt**	**opt**	**1438**	0.6s	**1438**	0.0%
		7	10	**0.00**	**1438**	0.2s	1306.29	**opt**	**opt**	**1438**	0.6s	**1438**	0.0%
small	50	5	10	1130.00	**1196**	10800.0s	1168.33	1168.33	**opt**	**1196**	19.8s	**1196**	0.0%
		6	10	1602.00	**1762**	10800.0s	1669.87	1740.72	**opt**	**1762**	21.1s	**1762**	0.0%
		7	10	2160.00	**2312**	10800.0s	2224.30	2278.13	**opt**	**2312**	25.4s	**2312**	0.0%
		8	10	2810.00	**3074**	10800.0s	2858.71	3024.33	**opt**	**3074**	14.7s	**3074**	0.0%
		9	10	3550.00	**4190**	10800.0s	3517.47	4180.00	**opt**	**4190**	5.3s	-	0.0%
		10	10	**3552.00**	**4190**	3033.9s	4172.67	4172.67	**opt**	**4190**	69.0s	-	0.0%
u-13	166	5	8	1842.00	4674	10800.0s	2126.28	2137.48	**2161.03**	-	10800.0s	**2194**	1.5%
		6	8	2528.00	13016	10800.0s	2934.03	3030.76	**3051.93**	-	10800.0s	**3064**	0.4%
		7	8	3290.00	13016	10800.0s	3859.64	4041.20	**opt**	**4054**	188.1s	**4054**	0.0%
		8	8		**infeasible**	15.0s	4938.37	-	**infeasible.**		16.2s	-	-
u-15	167	5	8	1800.00	9256	10800.0s	2077.36	2108.80	**2130.02**	-	10800.0s	**2168**	1.8%
		6	8	2508.00	13638	10800.0s	2935.13	3057.35	**opt**	**3084**	6006.1s	**3084**	0.0%
		7	8	3268.00	11552	10800.0s	3884.23	4098.56	**4142.77**	**4148**	10800.0s	4156	0.1%
		8	8		**infeasible**	11.4s	4935.82	-	**infeasible.**		16.3s	-	-
u-17	130	5	8	1638.00	5202	10800.0s	1926.13	1926.13	**1953.63**	-	10800.0s	**1998**	2.2%
		6	8	2246.00	7188	10800.0s	2669.58	2791.48	**opt**	**2838**	2942.3s	**2838**	0.0%
		7	8	2892.00	8988	10800.0s	3460.76	3603.07	**opt**	**3640**	127.5s	**3640**	0.0%
		8	8		**infeasible**	4.8s	4343.73	-	**infeasible.**		9.5s	-	-
u-21	58	5	8	1158.00	1478	10800.0s	1317.38	1417.00	**opt**	**1460**	365.1s	**1460**	0.0%
		6	8	1628.00	1988	10800.0s	1789.19	**opt**	**opt**	**1976**	2.2s	**1976**	0.0%
		7	8	2118.00	2516	10800.0s	2357.08	2474.91	**opt**	**2476**	7.2s	**2476**	0.0%
		8	8		**infeasible**	0.5s	2987.25	-	**infeasible.**		1.9s	-	-

Table 2 presents the results obtained with the branch-and-price approach described in Section 6. The best results obtained with the two compact formulations are presented, as in Table 1. Considering the branch-and-price: column LB^a shows the lower bounds obtained by the linear relaxation of formulation (19)-(20), while column LB^0 presents the linear relaxation value of (19)-(21). The best lower bound obtained (LB^*), best solution (UB) and runtime of the branch-and-price are also presented. In addition, the results obtained with the simulated annealing (SA) and the gaps between the best generated solution and the best bound are included. Note that Constraint (21) is essential whenever $|T|$ is indivisible by m^-, as expected. It enables considerably improving the quality of the linear relaxation's objective, leading to stronger formulations. The branch-and-price proved much more adequate than the presented compact formulations to solve the STGP. Although it was unable to provide feasible solutions for the larger instances (u-13, u-15 and u-17), high quality bounds were obtained. Additionally, simulated annealing generated high quality solutions, obtaining the optimal values for almost all solved instances. The resulting gap considering the best solution and the best bound are very low. The worst gap is only 2.2% for the largest considered instance, u-17.

Finally, it is noteworthy that the developed heuristic for the pricing problem performed considerably better than the integer programming approach. Therefore, results using only the solver for the pricing problem were omitted.

10 Conclusions

The present paper presented the Sport Teams Grouping Problem (STGP), a challenging and practical combinatorial optimization problem which is internationally applicable and relevant in recreational sports environments.

Despite its generality, the academic literature does not provide efficient algorithms for solving the STGP.

Three different integer programming formulations were developed and evaluated for the STGP, two of which are compact formulations and another with an exponential number of variables, being solved by a branch-and-price algorithm. A heuristic method is employed to solve the column generation's pricing problem within the developed branch-and-price. The compact formulations incurred much weaker results than the developed branch-and-price. The branch-and-price was able to solve most instances, except the large ones.

A simulated annealing approach was proposed to deal with the large instances. The method obtained optimal solutions for almost all instances.

Overall, good bounds and solutions were obtained. The largest optimality gap was only 2.2%.

Future research directions include further experimentation with larger instances, in addition to the development of cutting planes towards a branch-and-cut-and-price. Techniques such as strong branching should also be studied to improve the branch-and-price branching decisions.

Acknowledgments

Work supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX (http://comex.ulb.ac.be) and by the Leuven Mobility Research Centre (L-Mob). Editorial support provided by Luke Connolly, KU Leuven.

References

Achterberg, T., T. Koch, and A. Martin (2005). Branching rules revisited. *Operations Research Letters 33*(1), 42–54.

Anagnostopoulos, A., L. Michel, P. Van Hentenryck, and Y. Vergados (2006). A simulated annealing approach to the traveling tournament problem. *Journal of Scheduling 9*(2), 177–193.

Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance (1998). Branch-and-price: column generation for solving huge integer programs. *Operations Research 46*, 316–329.

Easton, K., G. Nemhauser, and M. Trick (2001). The traveling tournament problem description and benchmarks. In T. Walsh (Ed.), *Principles and Practice of Constraint Programming — CP 2001: 7th International Conference, CP 2001 Paphos, Cyprus, 2001 Proceedings*, pp. 580–584. Berlin, Heidelberg: Springer.

Gaspero, L. D. and A. Schaerf (2007). A composite-neighborhood tabu search approach to the traveling tournament problem. *Journal of Heuristics 13*(2), 189–207.

Ji, X. and J. E. Mitchell (2007). Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement. *Discrete Optimization 4*, 87–102.

Kendall, G., S. Knust, C. C. Ribeiro, and S. Urrutia (2010). Scheduling in sports: An annotated bibliography. *Computers & Operations Research 37*(1), 1–19.

Labbé, M. and F. A. Özsoy (2010). Size-constrained graph partitioning polytopes. *Discrete Mathematics 310*(24), 3473–3493.

Lübbecke, M. E. and J. Desrosiers (2005, December). Selected Topics in Column Generation. *Operations Research 53*(6), 1007–1023.

Nemhauser, G. L. and M. A. Trick (1998). Scheduling a major college basketball conference. *Operations research 46*(1), 1–8.

Ribeiro, C. C. and S. Urrutia (2007). Heuristics for the mirrored traveling tournament problem. *European Journal of Operational Research 179*(3), 775–787.

Toffolo, T. A. M., T. Wauters, S. Van Malderen, and G. Vanden Berghe (2016). Branch-and-bound with decomposition-based lower bounds for the traveling umpire problem. *European Journal of Operational Research 250*(3), 737–744.

Trick, M. A. and H. Yildiz (2007, January). Bender's cuts guided large neighborhood search for the traveling umpire problem. In P. Van Hentenryck and L. Wolsey (Eds.), *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, Number 4510 in Lecture Notes in Computer Science, pp. 332–345. Springer.

Trick, M. A. and H. Yildiz (2011). Benders' cuts guided large neighborhood search for the traveling umpire problem. *Naval Research Logistics (NRL) 58*(8), 771–781.

Trick, M. A., H. Yildiz, and T. Yunes (2012). Scheduling major league baseball umpires and the traveling umpire problem. *Interfaces 42*(3), 232–244.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research 48*(1), 111–128.

Vanderbeck, F. and L. Wolsey (2010). Reformulation and decomposition of integer programs. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey (Eds.), *50 Years of Integer Programming 1958-2008*, pp. 431–502. Springer.

Xue, L., Z. Luo, and A. Lim (2015). Two exact algorithms for the traveling umpire problem. *European Journal of Operational Research 243*(3), 932–943.