Solving the Sports League Scheduling Problem with Tabu Search

Jean-Philippe Hamiez[1] and Jin-Kao Hao[2]

[1] LGI2P / Ecole des Mines d'Alès – EERIE
Parc Scientifique Georges Besse – 30035 Nîmes Cedex 01 (France)
hamiez@site-eerie.ema.fr
[2] LERIA / Université d'Angers – 2, Bd. Lavoisier – 49045 Angers Cedex 01 (France)
Jin-Kao.Hao@univ-angers.fr

Abstract. In this paper we present a tabu approach for a version of the Sports League Scheduling Problem. The approach adopted is based on a formulation of the problem as a Constraint Satisfaction Problem (CSP). Tests were carried out on problem instances of up to 40 teams representing 780 integer variables with 780 values per variable. Experimental results show that this approach outperforms some existing methods and is one of the most promising methods for solving problems of this type.

1 Introduction

Many sports leagues (*e.g.* soccer, hockey, basketball) must deal with scheduling problems for tournaments. These scheduling problems contain in general many conflicting constraints to satisfy and different objectives to optimize, like minimization of traveling distance [2], only one match per team and per day, stadium unavailability at particular dates, minimum number of days between a home match and its corresponding away match, etc. Generating satisfactory schedules with respect to these conditions and objectives is therefore a very difficult problem to solve.

Many studies have been carried out to try to solve these problems with a variety of approaches and varying degrees of success: integer linear programming [7][12], constraint programming [10][17], local search (simulated annealing [19], tabu search [23], hybrid approach [4]).

This paper deals with a specific Sports League Scheduling Problem (SLSP) described by K. McAloon, C. Tretkoff and G. Wetzel in [11]. After having obtained poor results in integer linear programming tests, they experimented with constraint programming, an approach that produced better results. Finally, with a basic local search algorithm, they produced the same results as ILOG Solver does, but with less computing time.

C.P. Gomes, B. Selman and H.A. Kautz [9] obtained better results than those of McAloon et al. using constraint programming. With a randomized version of a deterministic complete search they solved problems involving a greater number of teams.

J.C. Régin proposed two approaches with constraint programming for the SLSP [15][16]. The first one, using powerful filtering algorithms [3][13][14], produced better results than those of McAloon et al. and those of Gomes et al. in terms of execution time and robustness, since it solved problem instances of a larger size. Using a second approach, he transformed the SLSP into an equivalent problem by adding an implicit constraint. With a new heuristic and specific filtering algorithms, he improved on his own results.

Finally, let us mention the work done by G. Wetzel and F. Zabatta [22]: using multiple threads on a 14 processor Sun system they obtained better results than the first approach of Régin. They were, however, not able to solve problems as large as those which Régin solved with his second approach.

The goal of this study is to propose an advanced local search approach based on tabu search (TS) [8] for the SLSP. References of the study are results presented in [11], [15] and [16].

The paper begins by formally describing the SLSP (§2). After modeling the problem as a constraint satisfaction problem (CSP) [20] (§3), we present our tabu algorithm (§4) and compare its results with those of [11], [15] and [16] (§5). Before concluding, we discuss some observations made during this work (§6).

2 Problem Description

In the rest of the paper, we will deal with the following constraints and definitions, the same as in [11]:

- There are |T| teams (|T| even), where T is the set of all teams. All teams play each other exactly once (half competition);
- The season lasts |T| - 1 weeks;
- Every team plays one game in every week of the season;
- There are |T| / 2 periods and, each week, one game is scheduled in every period;
- No team plays more than twice in the same period over the course of the season.

The problem then is to schedule the tournament with respect to all these constraints.

Table 1 below shows an example of a valid schedule for |T| = 8 teams labeled from 1 to 8; there are 7 weeks and 4 periods.

Table 1. Example of a valid schedule for 8 teams

		Weeks						
		1	2	3	4	5	6	7
Periods	1	1 vs 2	1 vs 3	5 vs 8	4 vs 7	4 vs 8	2 vs 6	3 vs 5
	2	3 vs 4	2 vs 8	1 vs 4	6 vs 8	2 vs 5	1 vs 7	6 vs 7
	3	5 vs 6	4 vs 6	2 vs 7	1 vs 5	3 vs 7	3 vs 8	1 vs 8
	4	7 vs 8	5 vs 7	3 vs 6	2 vs 3	1 vs 6	4 vs 5	2 vs 4

As shown in Table 1, a configuration may be represented as a two-dimensional array with weeks in columns and periods in rows. Each column satisfies the cardinality constraint: each team appears exactly once, *i.e.* all the teams are

different. In each row, no team appears more than twice. There is also a global constraint on the array: each match only appears once, *i.e.* all matches are different.

3 Problem Formulation

To represent the SLSP we follow the constraint programming approach: we consider it as a constraint satisfaction problem.

3.1 Constraint Satisfaction Problem - CSP

A constraint satisfaction problem [20] is defined by a triplet (X, D, C) with:

- A finite set X of n variables: $X = \{X_1, ... , X_n\}$;
- A set D of associated domains: $D = \{D_1, ... , D_n\}$. Each domain D_i specifies the finite set of possible values of the variable X_i;
- A finite set C of p constraints: $C = \{C_1, ... , C_p\}$. Each constraint is defined for a set of variables and specifies which combinations of values are compatible for these variables.

Given such a triplet, the problem is to generate a complete assignment of the values to the variables, which satisfies all the constraints: such an assignment is said to be consistent. Since the set of all assignments (not necessarily consistent) is defined by the Cartesian product $D_1 \times ... \times D_n$ of the domains, solving a CSP means to determine a particular assignment among a potentially huge number of possible assignments.

The CSP, as it has been formalized, is a powerful and general model. In fact, it can be used to conveniently model some well-known problems such as k-coloring and satisfiability, as well as many practical applications relating to resource assignment, planning or timetabling.

3.2 Formulation of the SLSP as a CSP

We will use the following notations to represent the SLSP as a constraint satisfaction problem:

- P: set of periods, $|P| = |T| / 2$;
- W: set of weeks, $|W| = |T| - 1$;
- t_n: team number n, $t_n \in T$, $1 \le n \le |T|$;
- $x(t_m, t_n)$: schedule of the match t_m vs. t_n. Values of this variable type are of $(p_{m,n}, w_{m,n})$ pattern, meaning that the match is scheduled in period $p_{m,n}$ and week $w_{m,n}$.

The set of variables is naturally $X = \{x(t_m, t_n), 1 \le m < n \le |T|\}$ and all domains are equal to $D = \{(p_{m,n}, w_{m,n}), p_{m,n} \in P, 1 \le p_{m,n} \le |P|, w_{m,n} \in W, 1 \le w_{m,n} \le |W|\}$; $\forall x \in X, D_x = D$. The set C of constraints contains the following three types of constraints:

- Uniqueness of all teams in each week. For each team t_n, $t_n \in T$, $1 \leq n \leq |T|$, we impose the constraint: $WEEK(t_n) \Leftrightarrow w_{m,n} \neq w_{q,n}$, $\forall (m, q) \in [1; |T|]^2$, $m \neq n$, $q \neq n$ and $m \neq q$;
- No more than two matches for each team in the same period. For each team t_n, $t_n \in T$, $1 \leq n \leq |T|$ and each period, we impose the constraint:
  $PERIOD(t_n) \Leftrightarrow |\{p_{m,n} = p_{q,n}, (m, q) \in [1; |T|]^2, m \neq n, q \neq n \text{ et } m \neq q\}| \leq 1$;
- All matches are different. For each match $< t_m, t_n >$, $(t_m, t_n) \in T^2$, $t_m \neq t_n$, we impose the constraint: $ALLDIFF(< t_m, t_n >) \Leftrightarrow < t_m, t_n > \neq < t_q, t_r >$, $\forall (t_q, t_r) \in T^2$, $t_q \neq t_r$.
  The WEEK and ALLDIFF constraints are always satisfied in our algorithm.

4 Solving the SLSP with Tabu Search

Tabu search is an advanced local search method using general mechanisms and rules as guidelines for smart search. Readers may find formal description of the method in [8]. We now define the components of our tabu algorithm for the SLSP, called TS-SLSP.

4.1 Search Space – Configuration

As represented in Table 1, a configuration is a complete assignment of $D = \{(p_{m,n}, w_{m,n}), p_{m,n} \in P, 1 \leq p_{m,n} \leq |P|, w_{m,n} \in W, 1 \leq w_{m,n} \leq |W|\}$ items to variables of $X = \{x(t_m, t_n), 1 \leq m < n \leq |T|\}$. Thus, a configuration is a $|W| * |P|$ sized table, whose items are integer couples (m, n), $1 \leq m < n \leq |T|$. For $|T| = 40$ teams, this represents a problem with 780 variables and 780 values per variable.

There are $|T| / 2 * (|T| - 1)$ matches to be scheduled. A schedule can be thought of as a permutation of these matches. So, for $|T|$ teams, the search space size is $[|T| / 2 * (|T| - 1)]!$ In other words, the search space size grows as the factorial of the square of $|T| / 2$.

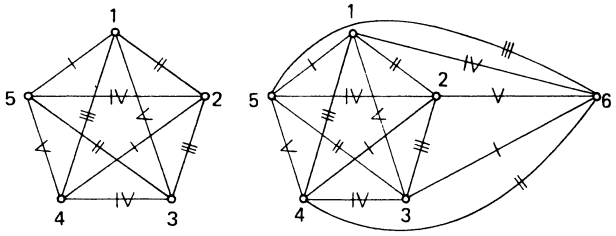

Fig. 1. Constructing initial configuration ($|T| = 6$)

4.2 Initial Configuration

In local search algorithms, the initial configuration specifies where the search begins in the search space. One can start with a random configuration by first creating all the |T| / 2 * (|T| - 1) matches with respect to the ALLDIFF constraint, then randomly assigning a match to each (w, p) couple, $w \in W$, $p \in P$.

We chose another way to build an initial configuration, inspired from [18]: construct a complete graph with the first |T| - 1 teams as vertices while placing the vertices in order to form a regular (|T| - 1)-gon (edges represent matches). Color the edges around the boundary using a different color for each edge. The remaining edges can be colored by assigning to each one the same color as that used for the boundary edge parallel to it, see left drawing in Fig. 1. At each vertex there will be exactly one color missing and these missing colors are different. The edge of the complete graph incident to the last vertex (*i.e.* the last team) can be colored using these missing colors, see right drawing in Fig. 1. Finally, fill in week "i", in the initial configuration, with edges colored "i".

This initial configuration has the property of satisfying the WEEK and ALLDIFF constraints. The algorithm will try to satisfy the PERIOD constraint.

4.3 Neighborhood

Let s be a configuration from the search space S. The neighborhood $N: S \rightarrow 2^S$ is an application such as for each $s \in S$, $s' \in N(s)$ if and only if s and s' only differ by values of a single couple of variables, with at least one in conflict (a variable is said to be conflicting if it is involved in an unsatisfied constraint).

A neighboring configuration of s can be obtained by making a single swap of current values of two arbitrary variables, with at least one conflicting in s. A move from configuration s to a neighboring configuration s' can then be described by a couple $< x(t_m, t_n), x(t_q, t_r) >$ *i.e.* swapping two matches (see Fig. 2). In addition, swaps are made in the same week to keep the WEEK constraint satisfied.

During the search the neighborhood size always evolves with the number of conflicts.

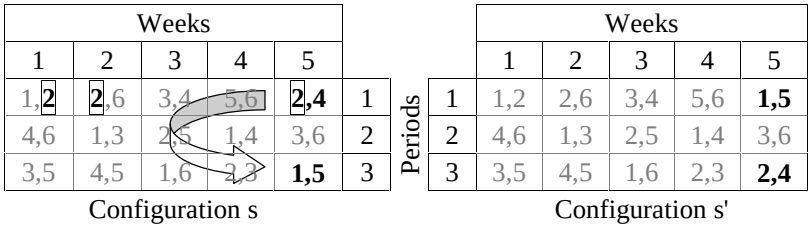

Fig. 2. Neighborhood illustration (6 teams)

4.4 Evaluation Function

To compare, in terms of quality, two configurations s and s' from S, we define an evaluation function which is an order relation for S.

Let $OccP_S(p, t_n)$ be the occurrence number of team t_n, in period p, in configuration s. The evaluation function f(s) is the total number of excess appearances of all teams in all periods (let us call it $f_p(s)$) *i.e.* the minimum number of variables to be changed to satisfy the PERIOD constraint:

$$f(s) = f_p(s) = \sum_{p=1}^{|P|} \sum_{n=1}^{|T|} \chi_P(s, n, p),$$
$$\chi_P(s, n, p) = \begin{cases} 0 & \text{if } OccP_S(p, t_n) \le 2, \\ OccP_S(p, t_n) - 2 & \text{otherwise}. \end{cases} \quad (1)$$

Solving the SLSP means finding a configuration $s^* \in S$ such as $f(s^*) = 0$.

4.5 Neighborhood Evaluation

The tabu algorithm considers in general at each step the whole neighborhood. So it is imperative to be able to quickly evaluate the cost of neighboring configurations. To do this, we used a technique inspired by [6].

Let δ be a |X| * |X| matrix. Each entry $\delta[x(t_m, t_n), x(t_q, t_r)]$ represents the effect of the chosen move (swapping matches $< t_m, t_n >$ and $< t_q, t_r >$) on the evaluation function. Thus, the cost of $s' \in N(s)$ is immediately obtained by adding the proper entry of δ to f(s) (in $O(1)$). To get the best neighbor, one search only a subset of the δ matrix in $O(|N(s)|)$ time. After a move, the δ matrix is updated in $O(|X|\sqrt{|X|})$ time in the worst case.

4.6 Tabu List Management

A fundamental component of TS is tabu list: a special, short-term memory that maintains a selective history, composed of previously encountered configurations or more generally pertinent attributes of such configurations. The aim of a tabu list is to avoid cycling and go beyond local optima.

Entries in our tabu list are paired matches. Indeed, we make match swaps. After swapping, the matched couple is classified tabu for the next k iterations (k, called tabu tenure, is problem dependent), which means that reverse swap is forbidden during that period.

Tabu tenure plays a very important role for the performance of a tabu algorithm. If overestimated, it will perhaps incorrectly prohibit the visiting of unexplored configurations and the method's capacity to explore the neighborhood will be reduced. If underestimated, the method may get trapped in local optima.

After having tried various tabu tenures (randomized and bounded, dynamic, static etc.), we chose a randomized one. For $s \in S$, we define the tabu tenure k_s by: $k_s = rand(g)$ where rand(g) is a random integer from [1; g], $g \in [4; 100]$.

To implement the tabu list, we use a |X| * |X| matrix with entries corresponding to moves. Each entry stores the current iteration number plus the tabu tenure. With this data structure, a single comparison with the current iteration number is sufficient to know if a move is tabu.

Nevertheless, note that a tabu move leading to a configuration better than the best configuration found so far is always accepted (aspiration criterion).

```
Input: |T|; {number of teams}
Output: a valid schedule or "No valid schedule found";
var  f, f*; {evaluation function and its best value
             encountered so far}
     s, s*; {current configuration and the best
             configuration encountered so far}
begin
  initialize the tabu list to empty;
  generate initial configuration s; {§4.2}
  s* := s;
  f* := f(s);
  while not Stop Condition do
    make a best move m in the same week such that m is
    not tabu or satisfies the aspiration criterion;
    introduce m in the tabu list;
    if (f(s) < f*) then
      s* := s;
      f* := f(s);
    else if f* has not been improved for a sufficient
    time then
      s = s*; {intensification}
      set the tabu list to empty; {diversification}
  if (f* = 0) then {valid schedule found}
    return s*;
  return "No valid schedule found";
end
```

Fig. 3. The TS-SLSP algorithm

4.7 Diversification – Intensification

Intensification and diversification are useful techniques for improving the search power of a tabu algorithm.

Intensification stores interesting properties of the best configurations found so far and to be used later. It may initiate a return to attractive regions to search them more thoroughly. Diversification tries to direct the search to unexplored configurations.

Our TS algorithm includes a single intensification: after a certain number of non-improving iterations, we return to the best configuration found so far. The

difficulty then is to formally determine when to go back to this best configuration, because an underestimated interval might inadvertently stop a possibly promising search. Values of this parameter were determined empirically in [30; 1 000].

We make a diversification immediately after an intensification process by removing all tabu statuses to enable previously classified tabu moves.

4.8 General Algorithm

The TS-SLSP algorithm (see Fig. 3) begins with an initial configuration in a search space S built with respect to WEEK and ALLDIFF constraints.

Then it proceeds iteratively to visit a series of locally best configurations following the neighborhood. At each iteration, a best move m (in the same week to keep the WEEK constraint satisfied) is applied to the current configuration, even if it does not improve the current configuration in terms of the value of the evaluation function. Intensification / diversification steps are performed after a sufficient number of non-improving moves.

Stop Condition. The algorithm stops if f(s) = 0 (a valid schedule is found) or if a given limit is reached concerning the number of moves.

5 Computational Results

In this section, we compare results of our TS algorithm with those of [11], [15] and [16]. Our tests were carried out on a Sun Sparc Ultra 1 (256 RAM, 143 MHz). TS-SLSP[1] is implemented in C (CC compiler with -O5 option). Tests were carried out on problem instances including 6 to 40 teams (only those greater than 14 are shown). The TS-SLSP algorithm was allowed to run until a maximum number of 10 million iterations.

5.1 Comparison Criteria

We used three main criteria to do the comparative study. These criteria are given here in decreasing order of importance:

- Number |T| of teams: one algorithm is said to be more efficient than another one, if it solves the SLSP with a higher |T|;
- Number of moves: the algorithm's effort to find a solution, machine independent;
- Running time: the CPU time spent by an algorithm to carry out a given number of moves, machine dependent.

---

[1] For the source code of TS-SLSP, contact the authors.

5.2 Comparative Results

McAloon et al. [11] recall that Integer Linear Programming (ILP) with CPLEX [5] was not able to solve the SLSP for |T| = 14 teams, even when avoiding symmetries. Nevertheless, the method is able to provide solutions for $|T| \leq 12$. They also propose a constraint programming algorithm, implemented under ILOG Solver, which solved the |T| = 14 problem in 45 minutes (Ultra Sparc), and was the first result obtained with constraint programming. However, their Constraint Logic Programming approach failed for |T| = 14. Finally, with a basic local search algorithm, they solved the |T| = 14 problem in 10 minutes.

Gomes et al. [9] used constraint programming to solve the SLSP up to 18 teams (in more than 48 hours) with a deterministic complete search. By including randomness in their algorithm, they reached the same result more quickly: approximately 22 hours for 18 teams.

In 1998, Régin presented improved results with a more elaborate approach based on constraint propagation [15] (let us call the approach CP1). This approach integrates advanced techniques including symmetries elimination and new powerful filtering algorithms based on arc consistency [3][13][14]. CP1 solves much larger problem instances since it is able to produce valid schedules for |T| = 24 in 12 hours.

In yet another study, by using multiple threads on a 14 processor Sun system, Wetzel and Zabatta [22] obtain better results, since schedules were generated for 26 and 28 teams.

Recently, Régin proposed another improved approach [16] (let us call it CP2) integrating among others, a heuristic based on Euler's theorem for Latin squares. The SLSP is also transformed into an equivalent problem by adding a dummy week in order to quickly deduce some inconsistencies. This later approach gives the best-known results for the SLSP, since it produces solutions for 40 teams in 6 hours.

Table 2. Comparative results of TS-SLSP and well-known methods. A "-" sign means no result is available

	CP1		CP2		TS-SLSP		
|T|	Time	Backtracks	Time (s)	Backtracks	Success Ratio (%)	Time	Moves
16	4.2 s	1 112	0.6	182	100	0.5 s	4 313
18	36 s	8 756	0.9	263	100	0.3 s	2 711
20	< 6 min	72 095	1.2	226	100	23.9 s	149 356
22	10 h	6 172 672	1.2	157	100	34.3 s	163 717
24	12 h	6 391 470	10.5	2 702	90	5 min	1 205 171
26	-	-	26.4	5 683	50	10.7 min	2 219 711
28	-	-	316	32 324	50	12.5 min	2 043 353
30	-	-	138	11 895	10	22 min	3 034 073
32	-	-	-	-	10	49 min	6 387 470
34	-	-	-	-	30	25 min	2 917 279
36	-	-	-	-	10	1.5 h	9 307 524
40	-	-	6 h	2 834 754	10	54.3 s	68 746

Table 2 shows the results of TS-SLSP together with those of CP1 and CP2. Columns 2-5 give respectively, for CP1 and CP2, execution times and numbers of backtracks. The last three columns give respectively the success ratio (number of successful runs / total number of runs), mean times and average numbers of moves for successful runs of TS-SLSP.

From the Table 2, we may make several remarks. First, we observe that TS-SLSP is much more powerful than CP1 both in terms of the size of problem instances solved and computing efforts. Compared with CP2, TS-SLSP also manages to find valid schedules for problem instances going up to 40 teams. We notice however that the resolution becomes much more difficult for $|T| > 28$. We notice also that the running time of the TS-SLSP algorithm for $|T| = 40$ is not consistent with other results. This is simply explained by the stochastic nature of the TS-SLSP algorithm. Globally, the results of TS-SLSP are consistent and its running times remain reasonable.

6 Discussions

How to handle the different constraints of a constrained problem is an important issue that must be answered by any neighborhood algorithm. For the SLSP, recall that there are three types of constraints (see §3.2): WEEK, PERIOD and ALLDIFF.

As we saw previously in §4, TS-SLSP works with a limited search space whose configurations satisfy already the WEEK and ALLDIFF constraints. The goal of TS-SLSP is then to satisfy the PERIOD constraint.

We also experimented with another constraint handling technique, in which the search space is defined by all the configurations satisfying the ALLDIFF constraint, and the search starts with a randomized initial configuration (see §4.2). Then the algorithm tries to satisfy the WEEK and PERIOD constraints. In this case, the evaluation function $f'(s)$ used to guide the search is defined by a weighted[2] summation function of the violations of the WEEK constraint $f_W(s)$, formally described by formula 2 below[3], and PERIOD constraint $f_P(s)$ (see formula 1 in §4.4): $\forall s \in S, f'(s) = 2 * f_W(s) + f_P(s)$.

$$f_W(s) = \sum_{w=1}^{|W|} \sum_{n=1}^{|T|} \chi_W(s, n, w),$$
$$\chi_W(s, n, w) = \begin{cases} 0 & \text{if } OccW_S(w, t_n) \leq 1, \\ OccW_S(w, t_n) - 1 & \text{otherwise.} \end{cases} \quad (2)$$

[2] It seems that the best weight values are 1 for PERIOD and 2 for WEEK.

[3] $OccW_S(w, t_n)$ is the occurrence number of team t_n on week w in configuration s.

The neighborhood is extended to all possible exchanges of matches (with at least one in conflict) in the same week or in the same period or in different periods and weeks, see Fig. 4.

The tabu tenure is weighted and dynamic: $k'_s = \alpha * [f_W(s) + f_P(s)] + rand(g)$, $\alpha \in [0.1; 1]$. Intensification and diversification processes are used in the same way as in TS-SLSP.

		Weeks						
		1	2	3	4	5	6	7
Periods	1	1 vs 3	1 vs 2	5 vs 8	1 vs 5	4 vs 8	2 vs 6	3 vs 5
	2	3 vs 4	2 vs 8	1 vs 4	6 vs 8	2 vs 5	1 vs 7	6 vs 7
	3	5 vs 6	4 vs 6	2 vs 7	4 vs 7	3 vs 7	3 vs 8	1 vs 8
	4	7 vs 8	5 vs 7	3 vs 6	2 vs 3	1 vs 6	4 vs 5	2 vs 4

Fig. 4. Extended neighborhood illustration

Experimentation using this technique produces results similar to those of CP1. Although it fails to solve the $|T| > 22$ problem, this technique is very fast for 22 teams: it provides solutions in less than 28 minutes (34 146 moves). When only dealing with the PERIOD constraint, TS-SLSP outperforms results of this technique, since it solves the SLSP up to 40 teams. The main difference between the two approaches is the structure of the initial configuration. The technique starting with an extra constraint to verify (the WEEK constraint) must explore a larger search space than the other technique.

7 Conclusion

In this paper we presented a tabu algorithm (TS-SLSP) for the sports league scheduling problem. The algorithm is based on a CSP formulation of the SLSP and includes the following main features:

- a simple swap neighborhood;
- efficient data structures for fast neighborhood search;
- a dynamic tabu tenure;
- a simple intensification-diversification process.

TS-SLSP was tested on problem instances going up to 40 teams. The experimental results show that TS-SLSP largely outperforms some previously developed approaches (ILP [11], basic LS [11] and constraint programming [9][15]). Indeed, while these approaches are limited to instances of 24 teams, TS-SLSP is able to find a schedule for instances going up to 40 teams. This result compares well with the best-known approach, which combines well-elaborated constraint propagation algorithms and a non-trivial formulation of the initial problem [16].

At the same time, the computing times required by TS-SLSP are much greater than those obtained using the most efficient algorithm. Nevertheless, there is certainly plenty of room for improvement in TS-SLSP. One possible enhancement would be to integrate more elaborate intensification and diversification mechanisms, another would be to have a close study of configuration structures in order to devise

clever neighborhoods and constraint handling techniques. It would also be interesting to envisage a combination of some advanced constraint propagation techniques with tabu search.

This work underlines once again the importance of efficient data structures for the high performance of a TS algorithm. It also confirms that parameter setting is crucial for obtaining quality solutions.

The TS-SLSP may also be adaptable to other sports scheduling problems *e.g.* the minimization of breaks (two consecutive home or away matches) [21] and those studied in [17]. Let us also mention that TS has already proven efficient for solving some sports scheduling problems, with constraints of various types. A hybrid tabu search has been proposed [4] to respond to the National Hockey League's scheduling problem, which underlines the effectiveness of a TS / genetic algorithm combination; the skilful mix of elements of the two methods produces better results than those of separately considered methods. All these studies suggest that a metaheuristic approach (*e.g.* TS, simulated annealing, genetic hybrid) has good potential for solving various planning and scheduling problems related to sporting events.[4]

Acknowledgments

We would like to thank J.C. Régin for having sent us his latest results and the referees for their relevant remarks.

References

1. Ackley, D.H.: *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer Academic Publishers (1987)
2. Bean, J.C., Birge, J.R.: Reducing Travelling Costs and Player Fatigue in the National Basketball Association. *Interfaces* 10 / 3 (1980) 98–102
3. Bessière, C., Régin, J.C.: Arc Consistency for General Constraint Networks: Preliminary Results. In: *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*. Nagoya, Aichi, Japan: International Joint Conference on Artificial Intelligence, Inc. (1997) 398–404
4. Costa, D.: An Evolutionary Tabu Search Algorithm and the NHL Scheduling Problem. *INFOR* 33 / 3 (1995) 161–178
5. CPLEX Optimization: Using the CPLEX©Callable Library. Tahoe, NV, Inc. (1995)
6. Ferland J.A., Fleurent, C.: Genetic and Hybrid Algorithms for Graph Coloring. *Annals of Operations Research: Metaheuristics in Combinatorial Optimization* 63 / 1. Hammer, P.L. et al. (Eds) (1996) 437–461
7. Ferland J.A., Fleurent, C.: Allocating Games for the NHL Using Integer Programming. *Operations Research* 41 / 4 (1993) 649–654
8. Glover, F., Laguna, M.: *Tabu Search*. Boston, MA: Kluwer Academic Publishers (1997)

[4] Additional studies are reported in [4][19][23].

9. Gomes C.P., Selman B., Kautz H.A.: Boosting Combinatorial Search Through Randomization. In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI Press / The MIT Press, Madison, WI (1998) 431–437
10. Henz, M.: Scheduling a Major College Basketball Conference – Revisited. To appear in: *Operations Research* 49 / 1 (2001)
11. McAloon, K., Tretkoff, C., Wetzel, G.: Sports League Scheduling. In: *Proceedings of the Third ILOG Optimization Suite International Users' Conference*. Paris, France (1997)
12. Nemhauser, G.L., Trick, M.A.: Scheduling a Major College Basketball Conference. *Operations Research* 46 / 1 (1998) 1–8
13. Régin, J.C.: A Filtering Algorithm for Constraints of Difference in CSPs. In: *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Vol. 1. Seattle, Washington: AAAI Press (1994) 362–367
14. Régin, J.C.: Generalized Arc Consistency for Global Cardinality Constraint. In: *Proceedings of the Thirteenth National Conference on Artificial Intelligence / Eighth Conference on Innovative Applications of Artificial Intelligence*, Vol. 1. Portland, Oregon: AAAI Press (1996) 209–215
15. Régin, J.C.: Modeling and Solving Sports League Scheduling with Constraint Programming. *First Congress of the French Operational Research Society* (*ROADEF*). Paris, France (1998)
16. Régin, J.C.: Sports Scheduling and Constraint Programming. *INFORMS*. Cincinnati, Ohio (1999)
17. Schaerf, A.: Scheduling Sport Tournaments Using Constraint Logic Programming. *Constraints* 4 / 1 (1999) 43–65
18. Schreuder, J.A.M.: Constructing Timetables for Sport Competitions. *Mathematical Programming Study* 13 (1980) 58–67
19. Terril, B.J., Willis, R.J.: Scheduling the Australian State Cricket Season Using Simulated Annealing. *Journal of the Operational Research Society* 45 / 3 (1994) 276–280
20. Tsang, E.P.K.: Foundations of Constraint Satisfaction. London and San Diego: Academic Press (1993)
21. Werra (de), D.: Geography, Games and Graphs. *Discrete Applied Mathematics* 2 (1980) 327–337
22. Wetzel G., Zabatta F.: CUNY Graduate Center CS Technical Report (1998)
23. Wright, M.: Timetabling County Cricket Fixtures Using a Form of Tabu Search. *Journal of the Operational Research Society* 45 / 7 (1994) 758–770

Appendix:

The table in the next page gives a solution to the Sports League Scheduling Problem found by TS-SLSP for 40 teams.

Periods	Weeks																																								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
1	1 2	2 3	3 4	4 5	5 6	6 7	7 8	8 9	9 10	10 11	11 12	32 40	27 39	14 15	15 16	16 17	17 18	18 19	19 20	20 21	21 22	22 23	23 24	24 25	25 26	1 13	27 28	28 29	29 30	30 31	12 40	32 33	33 34	34 35	35 36	36 37	37 38	38 39	14 26
2	3 39	1 4	2 5	3 6	19 31	5 8	6 9	22 34	8 11	9 12	10 13	11 14	12 15	13 16	14 17	15 18	7 28	38 40	18 21	10 31	20 23	21 24	22 25	23 26	24 27	25 28	26 29	27 30	4 16	29 32	30 33	7 19	32 35	33 36	34 37	17 40	36 39	1 37	2 38
3	8 34	5 39	10 36	11 37	3 8	4 9	23 31	6 11	7 12	4 17	9 14	2 23	7 20	12 17	13 18	14 19	15 20	16 21	13 26	1 40	19 24	16 29	21 26	22 27	15 36	24 29	25 30	22 35	27 32	28 33	25 38	30 35	10 18	2 28	33 38	34 39	5 31	6 32	3 37
4	5 37	6 38	7 39	1 8	25 40	3 10	4 11	5 12	6 13	7 14	8 15	9 16	10 17	11 18	12 19	13 20	14 21	15 22	16 23	17 24	18 25	19 26	20 27	21 28	22 29	23 30	24 31	9 40	26 33	27 34	28 35	29 36	30 37	31 38	32 39	1 33	2 34	3 35	4 36
5	9 33	7 37	8 38	9 39	1 10	2 11	3 12	4 13	5 14	6 15	7 16	8 17	30 36	10 19	11 20	12 21	13 22	14 23	15 24	16 25	17 26	18 27	19 28	20 29	6 40	22 31	23 32	24 33	25 34	26 35	27 36	28 37	29 38	30 39	1 31	2 32	18 40	4 34	5 35
6	7 35	8 36	9 37	10 38	11 39	1 12	2 13	3 14	4 15	5 16	6 17	7 18	8 19	9 20	10 21	11 22	12 23	13 24	14 25	15 26	2 40	17 28	18 29	19 30	20 31	21 32	22 33	23 34	24 35	25 36	26 37	27 38	28 39	1 29	16 40	3 31	4 32	5 33	6 34
7	4 38	9 35	1 6	2 7	12 38	13 39	27 40	2 15	3 16	8 13	5 18	6 19	11 16	8 21	9 22	10 23	11 24	12 25	17 22	14 27	15 28	20 25	17 30	18 31	19 32	20 33	21 34	26 31	23 36	24 37	29 34	26 39	14 40	32 37	3 29	4 30	1 35	10 28	7 33
8	18 24	10 34	11 35	12 36	13 37	14 38	24 30	1 16	2 17	30 40	4 19	5 20	33 40	7 22	8 23	33 39	10 25	11 26	12 27	13 28	14 29	6 39	16 31	17 32	3 9	19 34	20 35	21 36	22 37	23 38	9 15	1 25	2 26	3 27	4 28	5 29	15 21	7 31	8 32
9	10 32	11 33	12 34	17 31	14 36	26 40	16 38	21 35	1 18	23 37	3 20	25 39	5 22	2 27	7 24	36 40	9 26	6 31	11 28	12 29	13 30	10 35	15 32	16 33	17 34	14 39	1 15	20 37	21 38	4 18	5 19	2 24	3 25	8 22	9 23	6 28	7 29	8 30	13 27
10	11 31	12 32	13 33	14 34	15 35	16 36	17 37	18 38	19 39	1 20	2 21	3 22	4 23	34 40	6 25	7 26	8 27	9 28	10 29	11 30	12 31	13 32	14 33	5 40	16 35	17 36	18 37	19 38	20 39	1 21	2 22	3 23	4 24	5 25	6 26	7 27	8 28	9 29	10 30
11	12 30	13 31	14 32	15 33	16 34	17 35	18 36	28 40	20 38	21 39	1 22	10 15	3 24	4 25	5 26	6 27	4 31	8 29	9 30	19 22	11 32	12 33	10 37	14 35	23 28	16 37	17 38	18 39	1 19	2 20	3 21	13 40	5 23	6 24	7 25	8 26	9 27	2 36	11 29
12	13 29	14 30	15 31	16 32	17 33	18 34	19 35	20 36	21 37	22 38	31 40	1 24	2 25	3 26	4 27	5 28	6 29	7 30	39 40	9 32	10 33	11 34	12 35	13 36	14 37	15 38	16 39	1 17	2 18	3 19	4 20	5 21	6 22	7 23	8 24	9 25	10 26	11 27	12 28
13	14 28	15 29	16 30	13 35	18 32	19 33	20 34	17 39	22 36	2 19	24 38	4 21	1 26	6 23	3 28	4 29	5 30	10 27	7 32	8 33	9 34	14 31	11 36	12 37	13 38	18 35	8 40	2 16	3 17	22 39	1 23	6 20	7 21	15 40	5 27	10 24	11 25	12 26	9 31
14	15 27	22 40	17 29	18 30	4 7	20 32	21 33	19 37	23 35	24 36	25 37	26 38	13 14	1 28	2 29	3 30	16 19	5 32	6 33	7 34	8 35	9 36	4 40	11 38	12 39	26 27	2 14	3 15	28 31	5 17	6 18	31 34	8 20	9 21	10 22	11 23	12 24	13 25	1 39
15	16 26	17 27	18 28	24 40	20 30	21 31	22 32	23 33	24 24	25 35	26 36	27 37	28 38	29 39	1 30	2 31	3 32	4 33	5 34	6 35	7 36	8 37	9 38	10 39	1 11	2 12	3 13	4 14	5 15	6 16	7 17	8 18	9 19	10 20	11 21	12 22	13 23	19 40	15 25
16	17 25	18 26	19 27	20 28	21 29	22 30	5 10	24 32	25 33	26 34	27 35	28 36	29 37	30 38	31 39	1 32	2 33	3 34	4 35	18 23	6 37	3 40	8 39	1 9	2 10	7 40	4 12	5 13	6 14	7 15	8 16	9 17	31 36	11 19	12 20	13 21	14 22	15 23	16 24
17	6 36	19 25	23 40	21 27	22 28	23 29	15 39	25 31	26 32	27 33	28 34	29 35	9 18	31 37	32 38	9 24	1 34	2 35	3 36	4 37	5 38	15 30	1 7	2 8	21 30	4 10	5 11	6 12	7 13	8 14	24 39	10 16	11 17	12 18	13 19	14 20	3 33	16 22	20 40
18	19 23	20 24	21 25	22 26	23 27	24 28	25 29	26 30	27 31	28 32	29 33	30 34	31 35	32 36	35 40	34 38	37 40	1 36	2 37	3 38	4 39	1 5	2 6	3 7	4 8	5 9	6 10	7 11	8 12	9 13	10 14	11 15	12 16	13 17	14 18	15 19	16 20	17 21	18 22
19	20 22	21 23	22 24	23 25	24 26	25 27	26 28	27 29	28 30	29 31	30 32	31 33	32 34	33 35	34 36	35 37	36 38	37 39	1 38	2 39	1 3	2 4	3 5	4 6	5 7	6 8	7 9	8 10	10 40	11 40	11 13	12 14	13 15	14 16	15 17	16 18	17 19	18 20	19 21
20	21 41	16 28	20 26	19 29	2 9	15 37	1 14	7 10	29 40	3 18	23 39	12 13	6 21	5 24	33 37	8 25	35 39	17 20	8 31	5 36	16 27	7 38	13 34	15 34	18 33	3 11	19 36	25 32	9 11	10 12	31 32	4 22	1 27	4 26	2 30	35 38	6 30	14 24	17 23