

Metaheuristic methods based on Tabu search for assigning judges to competitions

Amina Lamghari · Jacques A. Ferland

© Springer Science+Business Media, LLC 2008

Abstract Two metaheuristic methods based on Tabu search are introduced to assign judges to individual competitions in a tournament. The complexity of the mathematical formulation accounting for the assignment rules, leads us to use such an approach. The first metaheuristic includes two different Tabu searches that are combined with a diversification strategy. The second metaheuristic is applied to a penalized version of the original model formulated as an assignment problem. This metaheuristic is also based on a Tabu search procedure including a diversification strategy driven by the constraints violated. Numerical results are provided to indicate the efficiency of the methods to generate very good solutions.

Keywords Tabu search · Metaheuristic · Diversification · Assignment

1 Introduction

Whenever competitions take place, judges have to be selected to evaluate the performance of the competitors and to identify a winner. According to the competition context, specific rules must be followed in assigning the judges. In general, it makes sense to have an odd number of judges having complementary expertises to cover as exhaustively as possible all the expertises required to evaluate the performances of the competitors. Furthermore, conflicts of interest should be avoided. In this paper, we analyze the judge assignment problem for the John Molson International Case Competition that takes place every year at Concordia University in Montreal (Canada) for the last 25 years. Even if the solution techniques are introduced for this specific context, they should nevertheless be easily adapted to other contexts by making proper minor adjustments to deal with slightly different specific rules.

The John Molson International Case Competition involves 30 teams of business students coming from top international universities. This set of teams is partitioned into 5 groups,

A. Lamghari · J.A. Ferland (✉)
Department of Computer Science and Operations Research, University of Montreal, Montreal, Canada
e-mail: ferland@iro.umontreal.ca

A. Lamghari
e-mail: lamghara@iro.umontreal.ca

each including 6 teams. The first part of the competition consists of a round-robin tournament including 5 rounds where each team competes against each of the other 5 teams of its group. Thus, each round includes 15 individual competitions where a pair of teams debate and propose solutions for a specified business case. The best teams move to the finals in the second part of the competition. In this paper we consider assigning the judges during one round of the first part. Note that the same approach can be used to assign the judges during the second part of the competition, but currently, the organizing committee prefers to proceed manually.

For each individual competition of a round, 3 or 5 judges are assigned according to the number of judges available for that round. For each round, two sets of judges are available:

- the set of *lead* judges
- the set of *other* judges.

Six (6) different fields of expertise for the judges are considered, and each judge has one of these expertises.

The following rules must be followed in assigning the judges to the individual competitions of a specific round:

- Hard rules or constraints that must be satisfied:
 - 3 or 5 judges must be assigned to each individual competition
 - A judge cannot be assigned to an individual competition involving a team coming from a University where he received a degree
 - A judge cannot be assigned to an individual competition involving a team coming from a University where he is a faculty member
 - At least one of the judges belongs to the set of *lead* judges.
- Soft rules or constraints (or objective) to be satisfied as much as possible:
 - The expertises of the judges assigned should be as different as possible to cover as many of the 6 fields of expertise
 - The number of individual competitions having 5 judges assigned should be maximized.

Finally, note that, for a specific round, once a *lead* judge has been assigned to each individual competition, the rest of the *lead* judges (if any are left) are available for assignment as *other* judges.

This judge assignment problem has some similarity with several other applications. One of these is the problem of forming maximally diverse groups (FMDG). This problem consists of partitioning a number of entities into a fixed number of groups having the same size where the objective is to maximize diversity within groups. Our problem would reduce to a (FMDG) if all judges were admissible for all competitions, if there was only one type of judges, and if the same number of judges was to be assigned to each competition. In this case the problem is simplified greatly. The (FMDG) has been shown to be NP-Complete by Kuo et al. (1993), and heuristic procedures have been proposed for the problem specified in different contexts by several authors, see Weitz and Lakshminarayanan (1998). Bhadury et al. (2000) use a network flow formulation (the dining problem) to solve efficiently the (FMDG).

In Landa-Silva and Burke (2007), the authors propose an asynchronous cooperative local search approach to deal with the office-space-allocation problem where a set of entities of different sizes have to be located in a set of rooms having different sizes. When the room capacities are exceeded, the violation is penalized. The violations of other soft constraints are also included in the objective function. The solution approach consists of applying several local searches starting with different initial solutions. Each local search generates a solution

that is improved with an intensification strategy using shared information between the different local searches and related to attractive and unattractive assignments of some entities to some rooms. Furthermore, each local search also provides additional information to be shared with the others. A diversification strategy is also used by each local search to strongly perturb their current solution.

Another application closely related to our problem is the selection of reviewers to referee journal papers, conference papers, and grant applications. Here we have to account for several hard and soft constraints related to the number of papers sent to the same referee, the expertise of the referees, the conflicts of interest, etc. Several authors have proposed different solution approaches to deal with different variants of the problem accounting for different constraints and objective functions. Goldsmith and Sloan (2007), Hartvigsen et al. (1999), and Sun et al. (2008) use network flow approaches. Sampson (2006) refers to a goal programming model to formulate a minimum cost flow problem. Dumais and Nielsen (1992) introduce an assignment procedure while Schirrer et al. (2007) use a local search method.

The assignment of referees to sport competitions is the most closely related application to ours. The assignment rules and the objectives differ with the specific contexts. For instance, some constraints are related to the number of referees required, to their level of experience, and the sequence of assignments of a given referee. The objective function might be to minimize the total travelled distance of the referees, or to balance their workload, or to minimize a weighted sum of violations of the soft constraints, for instance. These applications induce combinatorial optimization problems usually solved with heuristic or metaheuristic methods. Among others, we mention the following references: Duarte et al. (2006) and (2007a), (2007b), and Yavuz et al. (2008). Evans et al. (1984) use a minimum cost flow formulation. Also, Evans (1988) and Wright (1991) develop decision support systems for baseball and cricket leagues, respectively.

Since our problem is also of combinatorial nature, in Lamghari and Ferland (2005), we introduce heuristic procedures to construct good feasible solutions for this problem. But for some instances, these heuristic procedures may fail to generate feasible solutions. For this reason, in this paper we use more powerful metaheuristic methods based on the Tabu search principle (Glover and Laguna 1998; Hansen 1986) to generate better solutions. Note that this paper is an extended version of Lamghari and Ferland (2007) including only the first metaheuristic method based on the structured neighborhood Tabu search.

In Sect. 2 we introduce a mathematical formulation of the problem. Section 3 includes a first metaheuristic method to deal with the problem. First we introduce the modified mathematical model to be solved, and then we describe the different stages of the method. In the first stage, we use a structured neighborhood Tabu search to increase the number of individual competitions having 5 judges assigned. Another Tabu search is completed in the next stage to improve the diversity of the fields of expertise of the judges assigned to the same individual competition. Finally, a diversification strategy is applied in the third stage in order to reinitialize the procedure. In Sect. 4 we introduce a second metaheuristic method to solve the problem formulated as an assignment problem where the violations of the constraints are penalized in the objective function. This method is also based on a Tabu search procedure including a diversification strategy driven by the constraints violated. Numerical results are given in Sect. 5. Four different variants of each metaheuristic method are compared numerically. These variants are specified according to the process for generating the initial solution, and to the strategies for selecting the solution in the neighborhood of the current solution. For the last three editions of the John Molson International Case Competition, the number of judges available allowed to assign 5 judges to most competitions. In order to test the efficiency of our approaches to generate very good solutions for more difficult problems, we choose to generate problems simulating tighter situations as far as the number of

judges, their expertise and their admissibility are concerned. The numerical results indicate the efficiency of these approaches to generate very good solutions.

2 Mathematical formulation

Referring to the assignment rules, the problem for a round can be formulated as a linear binary programming problem. The hard rules are used to specify the constraints of the problem. The objective function is specified in terms of the soft rules to reduce the number of judges assigned to an individual competition sharing the same field of expertise, and to maximize the number of individual competitions having 5 judges assigned.

We use the following notation to formulate the problem:

N : the total number of judges

M : the number of individual competitions

K : the number of fields of expertise

i : judge index, $i = 1, 2, \dots, N$

j : individual competition index, $j = 1, 2, \dots, M$

$A = [a_{ij}]$ where

$$a_{ij} = \begin{cases} 1 & \text{if judge } i \text{ is admissible for individual competition } j \\ 0 & \text{otherwise.} \end{cases}$$

Note that judge i is not admissible for individual competition j if i received a degree or if he is a faculty member of either team university involved in competition j .

$$l_i = \begin{cases} 1 & \text{if judge } i \text{ is a lead judge} \\ 0 & \text{otherwise} \end{cases}$$

$$e_{ik} = \begin{cases} 1 & \text{if judge } i \text{ has expertise } k \\ 0 & \text{otherwise.} \end{cases}$$

The mathematical model can be summarized as follows:

$$\text{Min } \sum_{j=1}^M \sum_{k=1}^K \max \left\{ \left(\sum_{i=1}^N e_{ik} x_{ij} \right) - 1, 0 \right\} - 5M \sum_{j=1}^M y_5^j \quad (1)$$

$$\text{Subject to } \sum_{j=1}^M x_{ij} \leq 1 \quad i = 1, \dots, N \quad (2)$$

$$\sum_{j=1}^M (1 - a_{ij}) x_{ij} = 0 \quad i = 1, \dots, N \quad (3)$$

$$\sum_{i=1}^N l_i x_{ij} \geq 1 \quad j = 1, \dots, M \quad (4)$$

$$\sum_{i=1}^N x_{ij} = 3y_3^j + 5y_5^j \quad j = 1, \dots, M \quad (5)$$

$$y_3^j + y_5^j = 1 \quad j = 1, \dots, M \quad (6)$$

$$x_{ij} = 0 \text{ or } 1 \quad i = 1, \dots, N, \quad j = 1, \dots, M \quad (7)$$

$$y_3^j, y_5^j = 0 \text{ or } 1 \quad j = 1, \dots, M \quad (8)$$

where for $i = 1, \dots, N$ and $j = 1, \dots, M$, the variables

$$x_{ij} = \begin{cases} 1 & \text{if judge } i \text{ is assigned to individual competition } j \\ 0 & \text{otherwise} \end{cases}$$

and for all $j = 1, \dots, M$, the variables

$$y_3^j = \begin{cases} 1 & \text{if 3 judges are assigned to individual competition } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_5^j = \begin{cases} 1 & \text{if 5 judges are assigned to individual competition } j \\ 0 & \text{otherwise.} \end{cases}$$

In the objective function (1), we minimize the number of judges assigned to an individual competition sharing the same field of expertise, and we maximize the number of individual competitions with 5 judges. The coefficient $5M$ for the second term of the objective function guarantees that an additional pair of judges would be added to some competition even if it would induce that the 5 judges in each competition share the same field of expertise. The constraints (2) indicate that a judge cannot be assigned to more than one individual competition, and the constraints (3) do not allow inadmissible judge to be assigned to an individual competition. At least one *lead* judge is assigned to each individual competition according to constraints (4). Constraints (5) and (6) guarantee that 3 or 5 judges are assigned to each individual competition.

3 Structured neighborhood metaheuristic method (SN)

The complexity of the model leads us to introduce a modified version of the original model where we allow a competition to have only one judge assigned at the expense of a large penalty cost in the objective function. Then a metaheuristic including three main stages is used to solve the problem (Lamghari and Ferland 2007).

3.1 Modified model

In the modified model, the surplus of judges (if any) is assigned to a fictitious individual competition ($M + 1$) requiring no *lead* judge and for which all judges are admissible. Furthermore, since individual competitions with only one judge assigned are feasible but incurring a very high cost $C \gg 5M$, the following variables are introduced for all $j = 1, \dots, M$

$$y_1^j = \begin{cases} 1 & \text{if 1 judge is assigned to } j \\ 0 & \text{otherwise.} \end{cases}$$

To simplify the notation, the number of judges with field of expertise k assigned to individual competition j is denoted

$$d_{kj}(x) = \sum_{i=1}^N e_{ik} x_{ij}.$$

The new mathematical model can be summarized as follows:

$$\text{Min } f(x) = \sum_{j=1}^M \sum_{k=1}^K \max\{d_{kj}(x) - 1, 0\} - 5M \sum_{j=1}^M y_5^j + C \sum_{j=1}^M y_1^j \quad (9)$$

$$\text{Subject to } \sum_{j=1}^{M+1} x_{ij} = 1 \quad i = 1, \dots, N \quad (10)$$

$$\sum_{j=1}^M (1 - a_{ij})x_{ij} = 0 \quad i = 1, \dots, N \quad (11)$$

$$\sum_{i=1}^N l_i x_{ij} \geq 1 \quad j = 1, \dots, M \quad (12)$$

$$\sum_{i=1}^N x_{ij} = y_1^j + 3y_3^j + 5y_5^j \quad j = 1, \dots, M \quad (13)$$

$$y_1^j + y_3^j + y_5^j = 1 \quad j = 1, \dots, M \quad (14)$$

$$x_{ij} = 0 \text{ or } 1 \quad i = 1, \dots, N, \quad j = 1, \dots, M \quad (15)$$

$$y_1^j, y_3^j, y_5^j = 0 \text{ or } 1 \quad j = 1, \dots, M. \quad (16)$$

For any feasible solution x , we denote

$I(j, x) = \{i : \text{judge } i \text{ is assigned to individual competition } j \text{ in solution } x\}$

$M_1(x) = \text{set of individual competitions } j \neq M + 1 \text{ with 1 (lead) judge assigned}$

$M_3(x) = \text{set of individual competitions } j \neq M + 1 \text{ with 3 judges assigned}$

$M_5(x) = \text{set of individual competitions } j \neq M + 1 \text{ with 5 judges assigned.}$

The solution procedure is initiated with a feasible solution of the model. In a first stage, we use a structured neighborhood Tabu search to improve the quality of the solution by reducing the number of individual competitions with 1 or 3 judges assigned (i.e., to optimize the last two terms of the objective function (9)). Then, a second Tabu search is used to improve the diversity of the fields of expertise of the judges assigned to the same individual competition (i.e., to optimize the first term of the objective function (9)). Finally, a diversification strategy is used to generate a new initial solution to reinitialize the procedure.

The procedure terminates whenever an optimal solution is generated or whenever the time elapsed reaches *tempsmax*. Note that a solution is optimal if all individual competitions have 5 judges assigned, or if all individual competitions have 3 or 5 judges assigned and $I(M + 1, x) = 0$ or 1, and if the value of $\sum_{j=1}^M \sum_{k=1}^K \max\{d_{kj}(x) - 1, 0\}$ is equal to 0 or to a lower bound b_{inf} known a priori for the problem.

3.2 Initial solution

In this paper we consider two different processes to generate an initial solution. In the first one denoted **Random_bias**, an initial solution x^0 where each individual competition has only one *lead* judge assigned (i.e., $|M_1(x^0)| = M$ and $|M_3(x^0)| = |M_5(x^0)| = 0$), is generated randomly. Here, *lead* judges are assigned sequentially to the individual competitions with a bias to deal with those with fewer *lead* judges admissible first.

Initialization

$L = \{i : l_i = 1\}$, the set of *lead* judges available, and $O = \{i : l_i = 0\}$, the set of *other* judges available

$E^k = \{i \in L \cup O : e_{ik} = 1\}$, the set of judges with expertise k available

$M = \{1, \dots, M\}$, the set of competitions where no *lead* judge has been assigned yet

Algorithm

While ($M \neq \emptyset$) **do**

 LEAD JUDGE SELECTION

$i' := \infty, K = \{1, \dots, K\}$

While ($i' = \infty$) **do**

$k' := \arg \max_{k \in K} |E^k|$

$L^{k'} := \left\{ i \in L \cap E^{k'} : \sum_{j \in M} a_{ij} \geq 1 \right\}$

If ($L^{k'} \neq \emptyset$) **Then**

$i' := \arg \min_{i \in L^{k'}} \sum_{j \in M} a_{ij}$

Else

$K := K - \{k'\}$

 COMPETITION SELECTION

$\bar{J} := \left\{ \bar{j} \in M : a_{i'\bar{j}} = 1 \text{ and } \bar{j} = \arg \min_{j \in M} \sum_{i \in L} a_{ij} \right\}$

$j' := \arg \min_{\bar{j} \in \bar{J}} \sum_{i \in O \cap E^{k'}} a_{i\bar{j}}$

 UPDATE

$x_{i'j'} := 1, E^{k'} := E^{k'} - \{i'\}$

$L := L - \{i'\}, M := M - \{j'\}$

Fig. 1 HLA procedure

The second process is the **HLA-HOA** heuristic method introduced in Lamghari and Ferland (2005). The *heuristic for the lead judge assignment (HLA)* is used to assign a *lead* judge to each individual competition according to the following strategy: *lead* judges having the most common expertise k' and being admissible for fewer individual competitions are assigned first to individual competitions having the smallest number of admissible *other* judges having the expertise k' among those with the smallest number of admissible *lead* judges still available. These look ahead features make further assignments easier. The **(HLA)** method is summarized in Fig. 1.

The *heuristic for the other judge assignment (HOA)* is a two phase heuristic method using a similar strategy to assign additional pairs of judges to individual competitions accounting for the diversity requirement of the fields of expertise of the judges assigned to each individual competition. In the first phase, we try to assign an additional pair of judges to each individual competition, and in the second one, another pair is assigned to as many individual competitions as possible.

3.3 Structured neighborhood Tabu search for stage 1

Recall that during the first stage, the objective is to reduce the number of individual competitions with 1 or 3 judges assigned by reassigning pairs of judges. Accordingly, the neighborhood of a feasible solution x is generated by reassigning a pair of judges (i, r) currently assigned to some individual competition j to another individual competition l . The new solution generated is denoted

$$x \oplus (i, r, j, l).$$

The reassignment is feasible, and the solution generated belongs to the neighborhood of x if and only if it is feasible; i.e., if and only if

- (i) judges i and r are admissible for the individual competition l : $a_{il} = a_{rl} = 1$
- (ii) there exists a *lead* judge among those left assigned to the individual competition j :

$$\exists \bar{i} \in I(j, x) - \{i, r\} \text{ such that } l_{\bar{i}} = 1.$$

A safeguard against cycling is provided by the short term Tabu status of recently used reassignments. We use a Tabu matrix $LT = [LT_{ij}]$ where

LT_{ij} = the iteration after which the judge i can be assigned to the individual competition j .

If we move from x to $x \oplus (i, r, j, l)$, then two elements of the Tabu matrix are modified as follows:

$$LT_{ij} = \text{curiter} + t_1 \quad \text{and} \quad LT_{rj} = \text{curiter} + t_2$$

where *curiter* denotes the index of the current iteration, and t_1 and t_2 are random integer numbers in $[t_{\min}, t_{\max}]$. Note that this variable length Tabu list approach introduced in Tallard (1991) led to a robust Tabu search for the quadratic assignment problem. Also, referring to the current Tabu matrix LT , a neighbor solution $x \oplus (i, r, j, l)$ is Tabu at the iteration *iter* if

$$LT_{il} \geq \text{iter} \quad \text{and} \quad LT_{rl} \geq \text{iter}.$$

In our implementation, we use the classic aspiration criterion to override the Tabu status of a solution when its value is better than the current best solution found so far.

We take advantage of the problem structure in order to partition the set of reassignments leading to different neighborhood structures. To be more specific, for a solution x , denote by $V(x)$ the set of feasible reassignments. Given any pair of subsets of individual competitions $M_{\text{out}}, M_{\text{in}}$, denote by $\{M_{\text{out}}, M_{\text{in}}\}$ the subset of feasible reassignments from $j \in M_{\text{out}}$ to $l \in M_{\text{in}}, j \neq l$. Accordingly, we consider the following partition:

$$\bigcup_{p=1}^8 V_p(x) \subseteq V(x)$$

where the subsets $V_p(x)$ are specified in Table 1.

We use different neighborhood structures $V_p(x), p = 1, \dots, 8$, as in a variable neighborhood search proposed by Hansen and Mladenovic (2001), but here the strategy for moving from one neighborhood structure to another is different and strongly dependent on the partition and on the potential improvement associated with the reassignments in the different

Table 1 Partition of $V(x)$

p	$V_p(x)$	$\Delta_p(x)$
1	$\{M + 1, M_1(x)\}$	$-C$
2	$\{M_5(x), M_1(x)\}$	$-C + 5M - 2$
3	$\{M + 1, M_3(x)\}$	$-5M$
4	$\{M_5(x), M_3(x)\}$	-2
5	$\{M_3(x), M_1(x)\}$	-1
6	$\{M_5(x), M + 1\}$	$5M - 2$
7	$\{M_3(x), M_3(x)\}$	$-5M + C - 1$
8	$\{M_3(x), M + 1\}$	$C - 2$

subsets. For any reassignment leading to the neighbor solution $x \oplus (i, r, j, l)$, denote by $\Delta(i, r, j, l)$ the modification induced on the objective function

$$\Delta(i, r, j, l) = f(x \oplus (i, r, j, l)) - f(x).$$

Also, let $\Delta_p(x) = \min_{(i,r,j,l) \in V_p(x)} \Delta(i, r, j, l)$ be the best modification that can be induced by any reassignment $(i, r, j, l) \in V_p(x)$. Referring to the values $\Delta_p(x)$ given in the third column of Table 1, it follows that the subsets are ordered in increasing order of these values. Furthermore, the reassignments in the first 3 subsets are *improving reassignments* ($\Delta_p(x) < 0$, $p = 1, 2, 3$), those in $V_4(x)$ and $V_5(x)$ can be *slightly improving or deteriorating*, and those in the last 3 subsets are *deteriorating reassignments* ($\Delta_p(x) > 0$, $p = 6, 7, 8$). This partition leads to the following search strategy of the neighborhood. The Tabu search is initiated by using sequentially the subsets $V_p(x)$ for $p = 1, 2, 3$. Furthermore, the search in any of these subsets is *exhaustive* (in the sense that no more feasible non Tabu reassignments are available) before moving to the next. Once the exhaustive search of $V_3(x)$ is completed, then we use sequentially the other subsets. Now since the reassignments included in any of these subsets may be deteriorating, we are not completing an exhaustive search before moving to the next subset. Instead, after completing each reassignment in $V_4(x)$, we return to the second subset $V_2(x)$ initializing a new exhaustive search using the subsets $V_2(x)$ and $V_3(x)$. Indeed, any reassignment in $V_4(x)$ implies that new individual competitions in $M_3(x)$ and $M_5(x)$ are created, and consequently, new feasible improving reassignments may become available in $V_2(x)$ and $V_3(x)$. Similarly, after completing any reassignment in $\bigcup_{p=5}^8 V_p(x)$, we return to the first subset $V_1(x)$ because any such reassignment implies that a new individual competition in $M_1(x)$ is created or new judges are moved to the fictitious individual competition $M + 1$.

We also consider two different strategies for selecting the solution in the neighborhood of the current solution x (generated by any subset of reassignments $V_p(x)$). The *best improving strategy* is to select one of the best non Tabu neighbor solutions or one of the best Tabu neighbor solutions satisfying the aspiration criterion. Even though in general this strategy require generating all neighbor solutions, we can interrupt the generation whenever a solution inducing a modification $\Delta_p(x)$ of the objective function is reached. The *first improving strategy* is to select the first non Tabu solution improving the value of the current solution or the first Tabu solution satisfying the aspiration criterion. If no such solution exists, then the best non Tabu solution in the neighborhood is selected.

Finally, the stopping criterion for the method is specified in terms of a maximal number *nitermax* of successive iterations where the objective function does not improve. The procedure also terminates whenever all individual competitions have 5 judges assigned, or when all individual competitions have 3 or 5 judges assigned and $I(M + 1, x) = 0$ or 1.

The structured neighborhood Tabu search procedure for stage 1 where the *best improving strategy* is used to select the next current solution is summarized in Fig. 2.

3.4 Tabu search for stage 2

The solution generated in the first stage is used to initialize the Tabu search of the second stage where the objective is to improve the diversity of the fields of expertise of the judges assigned to each individual competition. Accordingly, the neighborhood of a feasible solution x is generated by exchanging two judges i and r currently assigned to different individual competitions j and l , respectively. The new solution generated is denoted

$$x \oplus (i, j, r, l).$$

The exchange is feasible, and the solution generated belongs to the neighborhood of x if and only if it is feasible; i.e., if and only if

- (i) judges i and r are admissible for individual competitions l and j ($a_{il} = a_{rj} = 1$), respectively
- (ii) there exists a *lead* judge in individual competitions j and l ; i.e.,

$$\exists \bar{i} \in I(j, x) - \{i\} \cup \{r\} \quad \text{such that } l_{\bar{i}} = 1$$

$$\exists \bar{i} \in I(l, x) - \{r\} \cup \{i\} \quad \text{such that } l_{\bar{i}} = 1.$$

As in stage 1, we use a Tabu matrix $LT = [LT_{ij}]$ where

LT_{ij} = the iteration after which the judge i can be assigned to the individual competition j .

If we move from x to $x \oplus (i, j, r, l)$, then two elements of the Tabu matrix are modified as follows:

$$LT_{ij} = \text{curiter} + t_1 \quad \text{and} \quad LT_{rl} = \text{curiter} + t_2$$

where *curiter* denotes the index of the current iteration, and t_1 and t_2 are random integer numbers in $[t_{\min}, t_{\max}]$. Also, referring to the current Tabu matrix LT , a neighbor solution $x \oplus (i, j, r, l)$ is Tabu at the iteration *iter* if

$$LT_{il} \geq \text{iter} \quad \text{and} \quad LT_{rj} \geq \text{iter}.$$

Furthermore, we use the same aspiration criterion, and we consider the same strategies for selecting the solution in the neighborhood of the current solution x . Here, in the *best improving strategy* we can interrupt the generation whenever a solution inducing a modification $\Delta(x) = -2$ of the objective function is reached.

Finally, the stopping criterion for the method is specified in terms of a maximal number *nitermax* of successive iterations where the objective function does not improve. The procedure also terminates whenever the value of $\sum_{j=1}^M \sum_{k=1}^K \max\{d_{kj}(x) - 1, 0\}$ is equal to 0 or to a lower bound b_{\inf} known a priori for the problem. The pseudo-code for this procedure where the *best improving strategy* is used to select the next current solution is given in Fig. 3.

Initialization

x^0 , an initial solution, and $x := x^0$, the current solution

x^* := x^0 , the best solution generated during stage 1

Γ , the set of the ρ best solutions generated so far

$LT := \emptyset$, the Tabu list

$niter := 0$, the number of successive iterations where the objective function does not improve

end := false

$p := 1$, the index of the subset of reassignments currently used

Algorithm

While ($end = false$) **do**

$niter := niter + 1$, $Z := \emptyset$, $s := 1$

GENERATING THE NEIGHBOR SOLUTIONS

While ($s \leq |V_p(x)|$) **do**

$x' \in V_p(x)$ (sequentially generated)

If (x' is not Tabu **or** satisfies the aspiration criterion) **Then**

If ($f(x') - f(x) = \Delta_p$) **Then**

$Z := \{x'\}$, $s := |V_p(x)| + 1$

Else

$Z := Z \cup \{x'\}$, $s := s + 1$

Else

$s := s + 1$

REPLACING THE CURRENT SOLUTION

If ($Z \neq \emptyset$) **Then**

$x := \arg \min_{z \in Z} \{f(z)\}$

SELECTING THE NEXT SUBSET OF REASSIGNMENTS TO BE USED

If ($Z \neq \emptyset$) **Then**

If ($p = 4$) **Then**

$p := 2$

If ($p \geq 5$) **Then**

$p := 1$

Else

If ($p < 8$) **Then**

$p := p + 1$

Else

$p := 1$

UPDATE

If ($Z \neq \emptyset$) **Then**

Update the Tabu list LT and the set Γ

If ($f(x) < f(x^*)$) **Then**

$x^* := x$, $niter := 0$

VERIFYING THE STOPPING CRITERIA

If ($(|M_5(x)| = M)$ **or** $(|M_1(x)| = 0)$ **and** $I(M + 1, x) = 0$ **or** 1) **or** ($niter = nitermax$) **Then**

end := true

x^* is the best solution generated during stage 1

Fig. 2 Structured neighborhood Tabu search for stage 1 using the *best improving strategy*

Initialization

x^* , the best solution generated in stage 1, and $x := x^*$, the current solution

$x^{**} := x^*$, the best solution generated during stage 2

Γ , the set of the ρ best solutions generated so far

$LT := \emptyset$, the Tabu list

$niter := 0$, the number of successive iterations where the objective function does not improve

$end := \text{false}$

Algorithm

While ($end = \text{false}$) **do**

$niter := niter + 1$, $Z := \emptyset$, $s := 1$

 GENERATING THE NEIGHBOR SOLUTIONS

While ($s \leq |V(x)|$) **do**

$x' := x \oplus (i, j, r, l)$ (sequentially generated)

If (x' is not Tabu **or** satisfies the aspiration criterion) **Then**

If ($f(x') - f(x) = -2$) **Then**

$Z := \{x'\}$, $s := |V(x)| + 1$

Else

$Z := Z \cup \{x'\}$, $s := s + 1$

Else

$s := s + 1$

 REPLACING THE CURRENT SOLUTION

If ($Z \neq \emptyset$) **Then**

$x := \arg \min_{z \in Z} \{f(z)\}$

 UPDATE

If ($Z \neq \emptyset$) **Then**

 Update the Tabu list LT and the set Γ

If ($f(x) < f(x^{**})$) **Then**

$x^{**} := x$, $niter := 0$

 VERIFYING THE STOPPING CRITERIA

If $\left(\left(\sum_{j=1}^M \sum_{k=1}^K \max\{d_{kj}(x) - 1, 0\} = 0 \text{ or } b_{\text{inf}} \right) \text{ or } (niter = nitermax) \right)$ **Then**

$end := \text{true}$

x^{**} is the best solution generated during stage 2

Fig. 3 Tabu search for stage 2 using the *best improving strategy*

3.5 Diversification strategy in stage 3

Once the first two stages of the solution procedure are completed, and the best solution generated is not optimal, then we use the following diversification strategy to search more extensively the feasible domain. A new initial solution is generated to reinitialize the first stage of the solution procedure.

Denote by Γ the set of the ρ best solutions generated so far during the procedure, and by $xbest \in \Gamma$ the best solution in Γ . First a new assignment x^0 of the judges (not necessarily feasible) is generated by using a variant of the uniform crossover operator (see Syswerda 1989) commonly used to implement genetic algorithms (see Goldberg 1989). Thus the pair

of solutions x_{best} and $\bar{x} \neq x_{best}$ selected randomly in Γ are combined according to the following process. Select randomly a subset of $\lceil \frac{\beta M}{countiter+1} \rceil$ individual competitions $M_{best} \subset \{1, \dots, M+1\}$, where *countiter* denotes the number of recent successive iterations where the value of the best solution generated by the procedure is not improved. For each individual competition $j = 1, \dots, M+1$,

$$I(j, x^0) = \begin{cases} I(j, x_{best}) & \text{if } j \in M_{best} \\ I(j, \bar{x}) & \text{otherwise.} \end{cases}$$

Note that in general, the number of elements selected from each solution is not specified a priori when the classical uniform crossover operator is used.

Now x^0 may not be feasible because the same judge i is assigned to two different individual competitions $j_1 \in M_{best}$ and $j_2 \notin M_{best}$; i.e., $i \in I(j_1, x^0) \cap I(j_2, x^0)$. Such a judge i is eliminated from j_1 or j_2 as follows. In order to favor the presence of the assignments found in x_{best} , we eliminate i from j_2 unless $j_1 = M+1$ or i is the only *lead* judge assigned to j_2 and there is more than one *lead* judge assigned to j_1 , in which case i is eliminated from j_1 .

But x^0 may still be infeasible because some individual competitions have no *lead* judge assigned or have 2 or 4 judges assigned. In this case, we apply the following *repair process* including two phases. In the first one, a *lead* judge is assigned to each individual competition. Denote $U = \{j : j \neq M+1, \text{ and } j \text{ has no lead judge assigned}\}$.

At each iteration of the first phase, select randomly an individual competition $l \in U$. Permute randomly the set of *lead* judges i admissible for l (i.e., $a_{il} = 1$). Consider sequentially the *lead* judges i :

- If i is assigned to $M+1$, then assign i to l
- If i is assigned to an individual competition j having several *lead* judges assigned, then assign i to l
- If i is the only *lead* judge assigned to j , and if r is another *lead* judge admissible for j but currently assigned to another individual competition \bar{j} having several *lead* judges assigned or currently assigned to $M+1$, then assign r to j and i to l .

(Note that it is always possible to assign a *lead* judge to each individual competition by assumption.)

Once every individual competition has a *lead* judge assigned, we proceed to phase 2 to deal with the individual competitions having 2 or 4 judges assigned. Denote

$$\xi = \{j : j \neq M+1, j \text{ having 2 or 4 judges assigned}\}.$$

At each iteration, select randomly $l \in \xi$. If there exists $i \in I(M+1, x^0)$ such that $a_{il} = 1$, then assign i to l . Otherwise, select randomly $r \in I(l, x^0)$ that is not the only *lead* judge in l , and assign r to $M+1$. At the end of phase 2, x^0 is feasible and can be used to reinitialize the stage 1 of the procedure.

The diversification strategy used in stage 3 is summarized in the pseudo-code presented in Fig. 4.

4 Tabu search method (TS) applied to a penalized version of the original model

In this section, we first introduce a penalized version of the original model in Sect. 2. The violations of the constraints (3) to (6) are penalized in a new objective function. Furthermore, fictitious judges are introduced, if necessary, to transform the new model into an assignment problem. Then a Tabu search procedure is used to deal with the new formulation of the problem.

Initialization

Γ , the set of the ρ best solutions generated so far

$xbest \in \Gamma$, the best solution found so far

$x^0 := \emptyset$, the new initial solution to generate

$M_{best} := \emptyset$, the set of competitions where the judges assigned are those in $xbest$

$L(j, x)$, the set of *lead* judges assigned to the competition j in the solution x

Algorithm

GENERATING A NEW INITIAL SOLUTION

Select randomly a solution $\bar{x} \neq xbest$ in Γ

For $j = 1$ **to** $\lceil \frac{\beta M}{\text{counter}+1} \rceil$ **do**

Select randomly a competition $l \notin M_{best}$

$M_{best} := M_{best} \cup \{l\}$

For $j = 1$ **to** $M + 1$ **do**

If ($j \in M_{best}$) **Then**

$I(j, x^0) := I(j, xbest)$

Else

$I(j, x^0) := I(j, \bar{x})$

For each $i \in I(j_1, xbest) \cap I(j_2, \bar{x})$ **do**

If ($(j_1 = M + 1)$ **or** ($j_1 \neq M + 1$ **and** $|L(j_1, x^0)| \geq 2$ **and** $|L(j_2, x^0)| = 1$))

Then

$I(j_1, x^0) := I(j_1, x^0) - \{i\}$

Else

$I(j_2, x^0) := I(j_2, x^0) - \{i\}$

REPAIR PROCESS

PHASE 1

$U = \{j : j \neq M + 1, \text{ and } |L(j, x^0)| = 0\}$

While ($U \neq \emptyset$) **do**

Select randomly $l \in U$

$end := \text{false}$

While ($end = \text{false}$) **do**

For each $i \in L(j, x^0)$ such that $a_{il} = 1$ (the competitions j are considered sequentially according to a random permutation) **do**

If ($(j = M + 1)$ **or** ($j \neq M + 1$ **and** $|L(j, x^0)| \geq 2$)) **Then**

$I(j, x^0) := I(j, x^0) - \{i\}, I(l, x^0) := I(l, x^0) \cup \{i\}$

$end := \text{true}$

If ($j \neq M + 1$ **and** $|L(j, x^0)| = 1$) **Then**

If ($(\exists r \in L(\bar{j}, x^0))$ **and** ($a_{rj} = 1$) **and** ($\bar{j} = M + 1$ **or**

$|L(\bar{j}, x^0)| \geq 2$)) **Then**

$I(\bar{j}, x^0) := I(\bar{j}, x^0) - \{r\}$

$I(j, x^0) := I(j, x^0) - \{i\} \cup \{r\}$

$I(l, x^0) := I(l, x^0) \cup \{i\}$

$end := \text{true}$

$U := U - \{l\}$

Fig. 4 Diversification strategy in stage 3

PHASE 2

$$\xi = \{j : j \neq M + 1, \text{ and } |I(j, x^0)| = 2 \text{ or } 4\}$$
While ($\xi \neq \emptyset$) **do**

 Select randomly $l \in \xi$
 $end := \text{false}$
 $s := 1$
While ($end = \text{false}$) **do**
 $i \in I(M + 1, x^0)$ some judge not assigned (considered sequentially)

If ($a_{il} = 1$) **Then**
 $I(M + 1, x^0) := I(M + 1, x^0) - \{i\}$
 $I(l, x^0) := I(l, x^0) \cup \{i\}$
 $end := \text{true}$
Else
 $s := s + 1$
If ($end = \text{false}$ and $s > |I(M + 1, x^0)|$) **Then**

 Select randomly $r \in I(l, x^0)$ such that $|I(l, x^0) - \{r\}| \geq 1$
 $I(l, x^0) := I(l, x^0) - \{r\}$
 $I(M + 1, x^0) := I(M + 1, x^0) \cup \{r\}$
 $end := \text{true}$
 $\xi := \xi - \{l\}$
 x^0 is the new initial solution generated to reinitialize stage 1

Fig. 4 (Continued)

4.1 Assignment problem with penalized violations of the constraints

Since the purpose is to have as many individual competitions as possible with 5 judges assigned, we modify the structure of the problem to associate 5 destination nodes with each match $j : j, M + j, 2M + j, 3M + j, \text{ and } 4M + j$. If the number of judges $N > 5M$, then $(N - 5M)$ fictitious destination nodes are associated with the fictitious individual competition $(M + 1)$.

Now if $N < 5M$, then once 3 judges are assigned to each individual competition, only $\lfloor \frac{N-3M}{2} \rfloor$ individual competitions can have an additional pair of judges assigned. Also, there may be a judge in surplus that cannot be assigned to any individual competition if the difference $(N - 3M)$ is an odd number; i.e., $\tau = (N - 3M) \bmod 2 = 1$. In this case we introduce a fictitious destination node associated with the fictitious individual competition $(M + 1)$. Furthermore, we introduce $(5M + \tau - N)$ source nodes associated with fictitious judges in order to have 5 judges assigned to each individual competition.

In any case, we generate an assignment problem having \overline{N} source nodes and \overline{N} destination nodes where

$$\overline{N} = \begin{cases} 5M + \tau & \text{if } N < 5M \\ N & \text{otherwise.} \end{cases}$$

The objective function includes 4 different terms $ad_j(x)$, $lj_j(x)$, $nj_j(x)$, and $dv_j(x)$ associated with the admissibility constraints (3), the *lead* judge constraints (4), the requirement that 3 or 5 judges must be assigned to each individual competition (constraints (5) and (6)), and the diversity of the assigned judges fields of expertise, respectively. The values of these terms are specified as follows:

(i) Judge admissibility constraints (3):

$$ad_j(x) = \sum_{t=0}^4 \sum_{i=1}^N (1 - a_{ij}) x_{i(tM+j)} \quad j = 1, \dots, M.$$

(ii) Lead judge constraints (4):

If we denote by $L_j(x) = \sum_{t=0}^4 \sum_{i=1}^N l_i x_{i(tM+j)}$ the number of lead judges assigned to the individual competition j , then the value of $lj_j(x)$ should be equal to 0 if and only if $L_j(x) \geq 1$; i.e.,

$$lj_j(x) = \max(1 - L_j(x), 0) \quad j = 1, \dots, M.$$

(iii) 3 or 5 judges assigned to each individual competition j (5) and (6):

Denote by $F_j(x) = \sum_{t=0}^4 \sum_{i=N+1}^{\bar{N}} x_{i(tM+j)}$ the number of fictitious judges assigned to the individual competition j . Then the value of $nj_j(x)$ should be equal to 0 if and only if $F_j(x) = 0$ or 2. It follows that

$$nj_j(x) = \max\left(\left\lfloor \frac{F_j(x)}{2} \right\rfloor - 1, F_j(x) - 2 \left\lfloor \frac{F_j(x)}{2} \right\rfloor\right) \quad j = 1, \dots, M.$$

(iv) Diversity of the fields of expertise:

$$dv_j(x) = \sum_{k=1}^K \max(d_{kj}(x) - 1, 0) \quad j = 1, \dots, M.$$

The new model can be formulated as the following assignment problem:

$$\text{Min } C \sum_{j=1}^M (ad_j(x) + lj_j(x) + nj_j(x)) + \sum_{j=1}^M dv_j(x) \quad (17)$$

$$\text{Subject to } \sum_{\mu=1}^{\bar{N}} x_{i\mu} = 1 \quad i = 1, \dots, \bar{N} \quad (18)$$

$$\sum_{i=1}^{\bar{N}} x_{i\mu} = 1 \quad \mu = 1, \dots, \bar{N} \quad (19)$$

$$x_{i\mu} = 0 \text{ or } 1 \quad i = 1, \dots, \bar{N}, \mu = 1, \dots, \bar{N} \quad (20)$$

Note that the destination nodes of the model are denoted $\mu = 1, \dots, \bar{N}$ while the individual competitions are denoted $j = 1, \dots, M$, and exactly 5 destination nodes are associated with each individual competition.

4.2 Tabu search procedure

This assignment problem is solved using a Tabu search procedure. As in Sect. 3.2, we consider two different processes to generate an initial solution. In the **Random** process, the \bar{N} judges (source nodes) are assigned randomly to the \bar{N} destination nodes associated with the individual competitions. The second process generates an assignment from the solution obtained with the **HLA-HOA** heuristic procedure introduced in Sect. 3.2.

The neighborhood of a feasible solution x of the model in Sect. 4.1 is generated by exchanging two judges i and r currently assigned to two different destination nodes associated with two different individual competitions j and l , respectively. The new solution generated is denoted

$$x \oplus (i, j, r, l).$$

Note that if the two judges i and r are fictitious, then their exchange does not modify the value of the objective function. Hence we only consider *interesting* modifications (i, j, r, l) where $i \leq N$ or $r \leq N$.

The short term Tabu status of recently used reassignments is specified as follows. If we move from x to $x \oplus (i, j, r, l)$, then two elements of the Tabu matrix are modified as follows:

$$LT_{ij} = \text{curiter} + t_1 \quad \text{and} \quad LT_{rl} = \text{curiter} + t_2$$

where *curiter* denotes the index of the current iteration, and t_1 and t_2 are random integer numbers in $[t_{\min}, t_{\max}]$. Also, referring to the current Tabu matrix LT , a neighbor solution $x \oplus (i, j, r, l)$ is Tabu at the iteration *iter* if

$$LT_{il} \geq \text{iter} \quad \text{or} \quad LT_{rj} \geq \text{iter}.$$

We use the same aspiration criterion, and we consider the same strategies (*first improving* and *best improving strategies*) for selecting the solution in the neighborhood of the current solution x as in the procedure in Sect. 3.3. But in order to break ties among the best candidate solutions, we rely on the following secondary frequency based criterion using a frequency matrix $\text{Freq} = [\text{Freq}_{ij}]$ where each entry Freq_{ij} is associated with the pair of judge i and individual competition j . This frequency matrix is updated at each iteration in order to keep track of the number of times that each pair (i, j) has been involved in modifying the current solution; i.e., whenever the solution $x \oplus (i, j, r, l)$ becomes the current solution, then the values of the two entries Freq_{il} and Freq_{rj} are increased by 1. Now among the best candidate solutions $x \oplus (i, j, r, l)$, we select the one having the smallest frequency value

$$P(i, j, r, l) = \sqrt{\text{Freq}_{il}} + \sqrt{\text{Freq}_{rj}}.$$

This secondary selection criterion can be seen as a diversification strategy since it has the advantage of selecting modifications less frequently used up to now, and hence allowing a more extensive search of the feasible domain. Furthermore, if we have to choose between two candidate solutions $x \oplus (i, j, r, l)$ and $x \oplus (i', j', r', l')$ where $(\text{Freq}_{il} + \text{Freq}_{rj}) = (\text{Freq}_{i'l'} + \text{Freq}_{r'j'})$, then the criterion based on the expression of $P(i, j, r, l)$ above indicates to select $x \oplus (i, j, r, l)$ if $\min\{\text{Freq}_{il}, \text{Freq}_{rj}\} \leq \min\{\text{Freq}_{i'l'}, \text{Freq}_{r'j'}\}$.

Finally, the stopping criterion for the method is specified in terms of a maximal number *nitermax* of successive iterations where the objective function does not improve. In Fig. 5, we summarized this Tabu search procedure where the best improving strategy is used to select the next current solution.

4.3 Diversification strategy

When the Tabu search procedure terminates, let *xbest* be the best solution generated so far. If *xbest* is not optimal and if the time elapsed is smaller than *tempsmax*, then we use the following diversification strategy generating a new initial solution used to reinitialize the

Initialization

x^0 , an initial solution and $x := x^0$, the current solution

$x^* := x^0$, the best solution generated by the procedure

LT , the Tabu list

$Freq$, the frequency matrix

$niter := 0$, the number of successive iterations where the objective function does not improve

$end := \text{false}$

Algorithm

While ($end = \text{false}$) **do**

$niter := niter + 1$, $E := \emptyset$, $\Delta_{\min} := \infty$

 GENERATING THE NEIGHBOR SOLUTIONS

For each *interesting* modification (i, j, r, l) **do**

If ($x \oplus (i, j, r, l)$ is not Tabu **or** verifies the aspiration criterion) **Then**

If ($f(x \oplus (i, j, r, l)) - f(x) < \Delta_{\min}$) **Then**

$\Delta_{\min} := \Delta_{ir}$, $E := \emptyset$, $E := E \cup (i, j, r, l)$

Else

If ($\Delta_{ir} = \Delta_{\min}$) **Then**

$E := E \cup (i, j, r, l)$

 REPLACING THE CURRENT SOLUTION

If ($E \neq \emptyset$) **Then**

$(i', j', r', l') := \arg \min_{e \in E} \{P(e)\}$

$x := x \oplus (i', j', r', l')$

 UPDATE

If ($E \neq \emptyset$) **Then**

 Update the Tabu list LT and the frequency matrix $Freq$

If ($f(x) < f(x^*)$) **Then**

$x^* := x$, $niter := 0$

 VERIFYING THE STOPPING CRITERIA

If ($(f(x) = 0$ or $b_{\inf})$ **or** ($niter = niter_{\max}$)) **Then**

$end := \text{true}$

x^* is the best solution generated

Fig. 5 Tabu search procedure using the *best improving strategy*

solution procedure. The purpose of this strategy is to search more extensively the feasible domain. In order to select the modifications to be applied to the solution $xbest$ and to generate a new initial solution x^0 , we partition the set of judges (including the fictitious judges) into 2 sets W and G . Denote by $I(j, xbest)$ the set of the 5 judges assigned to the individual competition j . Then judge $i \in I(j, xbest)$ belongs to the set W if one of the following conditions is satisfied:

- $n_{j_j}(xbest) > 0$ and i is a fictitious judge
- $ad_j(xbest) > 0$ and i is not admissible for the individual competition j .

Also,

- if $lj_j(xbest) > 0$, then include one of the judges $i \in I(j, xbest)$ selected randomly

Initialization

$xbest$, the best solution generated so far and $x^0 := xbest$, the new initial solution to generate

L , the list of modifications in increasing order of their $P(i, j, r, l)$ values

$LT := \emptyset$, the Tabu list, and $Freq$, the frequency matrix

Algorithm

While ($L \neq \emptyset$) **do**

 SELECTING A MODIFICATION

$e_1 :=$ first element in L

$E := \{e_1\}$

$s := 2$

While ($P(e_s) = P(e_1)$ and $s \leq |L|$) **do**

$E := E \cup \{e_s\}$

$s := s + 1$

$Z := \{x^0 \oplus (i, j, r, l) \quad : \quad (i, j, r, l) \in E\}$

$E' := \{(i, j, r, l) \quad : \quad f(x^0 \oplus (i, j, r, l)) = \min_{z \in Z} f(z)\}$

 Select randomly $(i', j', r', l') \in E'$

 UPDATE

$x^0 := x^0 \oplus (i', j', r', l')$

$L := L - \{(i, j, r, l) \quad : \quad i = i' \text{ or } r = r'\}$

 Update the Tabu list LT and the frequency matrix $Freq$

x^0 is the new initial solution generated to reinitialize the Tabu search procedure

Fig. 6 Diversification strategy

- if $dv_j(xbest) > 0$, then include one of the judges $i \in I(j, xbest)$ selected randomly for each pair of judges sharing the same expertise.

All the other judges belong to the set G .

Consider the list L of modifications (i, j, r, l) satisfying the 3 following conditions:

- $i \in W$ and $r \in G$
- i and r are assigned to two different individual competitions
- i and r are not both fictitious judges.

We use the following sequential process summarized in the pseudo-code given in Fig. 6, to generate a new initial solution x^0 . Start with $x^0 = xbest$. At each iteration, select the modification in L having the smallest frequency value $P(i, j, r, l)$. If several modifications in L have the same frequency value, then select the one inducing the best improvement of the objective function. Then set $x^0 = x^0 \oplus (i, j, r, l)$, and eliminate from L all the modifications involving judges i or r . The Tabu list matrix and the frequency matrix are also updated accordingly. The process terminates when the list L is empty. Then the solution x^0 is used to reinitialize the Tabu search procedure.

5 Numerical results

Four different variants for each metaheuristic method are compared numerically. These variants are specified according to the process generating the initial solution (**HLA-HOA** denoted **H** and **Random_bias** or **Random** denoted **R**), and to the strategies for selecting the

solution in the neighborhood of the current solution x (*best improving strategy* and *first improving strategy* denoted **Best** and **First**, respectively). **SN-H-Best**, **SN-H-First**, **SN-R-Best**, and **SN-R-First** denote the 4 variants of the structured neighborhood method introduced in Sect. 3, and **TS-H-Best**, **TS-H-First**, **TS-R-Best**, and **TS-R-First** those of the Tabu search method in Sect. 4. Furthermore, since (1)–(8) is a linear binary model, we try to solve it using CPLEX 9.13 for the sake of comparison with the four variants of the two methods.

We were involved in generating the judge schedules for the last three editions of the John Molson International Case Competition. This collaboration with the organizing committee allowed us to take hold of the structure of the problem. In their specific context, the committee is fortunate enough to have a number of judges available allowing assigning 5 judges to most competitions, and several of these judges are admissible to be assigned to most of the competitions without any conflict of interest. As a consequence, the problems were easily solved with any of our heuristic and metaheuristic methods. Nevertheless, the committee was fully satisfied since the judge schedules had to be generated during a very short period of time between the evening where the draw takes place to determine the groups and the next morning when the competitions start.

Being aware of the structure of the problem, we choose to generate problems having a similar structure but including a larger number of individual competitions than in the John Molson International Case Competition in order to test more extensively the efficiency of the methods. Furthermore, different types of problems were generated randomly allowing to simulating tighter situations as far as the number of judges, their expertise and their admissibility are concerned. For these reasons, the numerical tests are completed using 3 different sets of problems P_1 , P_2 , and P_3 where each set includes 4 different subsets (each subset including 10 different problems) with 15, 50, 150, and 500 individual competitions, respectively. The problems are generated such that for problems in P_1 , there exists a solution where no pair of judges assigned to the same competition share the same expertise (i.e., the first term of the objective function (1) is equal to 0), and for problems in P_2 and P_3 , no such solution exists. Furthermore, for problems in P_2 , there always exists a solution where all competitions have 5 judges assigned, and for problems in P_1 and P_3 , some competitions have only 3 judges assigned. The data files of these problems are available from the authors upon request.

All the problems are randomly generated. For each team of each individual competition, its University corresponds to a random number in the intervals [1, 10], [1, 10], [1, 30], and [1, 100] for the problems with 15, 50, 150, and 500 individual competitions, respectively. The number of *lead* judges available is equal to a random number in the intervals [15, 50], [50, 60], [150, 160], and [500, 510] for the problems with 15, 50, 150, and 500 individual competitions, respectively. Similarly, the number of *other* judges available is equal to a random number in the intervals [60, 300], [100, 220], [300, 620], and [1000, 2020] for the problems with 15, 50, 150, and 500 individual competitions, respectively. For each judge available, the Universities where he received his degree and where he is a faculty member correspond to random numbers in the intervals [1, 10], [1, 10], [1, 30], and [1, 100] for the problems with 15, 50, 150, and 500 individual competitions, respectively. Finally, the field of expertise of each judge is a random number in the interval [1, 6]. Note that for problems P_2 and P_3 , the number of judges having the same field of expertise is fixed a priori for each field in order to compute a lower bound of the optimal value.

All the variants are implemented in Java, and the tests are completed on a 1993 MHz processor AMD Opteron 246 with 4 GB of memory working under a Linux operating system.

The penalty C used in both models is fixed to the value $50M^2$. For the values of the other parameters common to both models, we consider the 18 different combinations generated with the two values for the interval $[t_{\min}, t_{\max}]$ ($[0.8N]$, $[1.2N]$) and $[0.9N]$, $[1.1N]$), the three values for $nitermax$ (N , $5N$ and $10N$), and the three values for $tempsmax$ ($2N$, $3N$ and $4N$ seconds). To fix the values of the parameters, some preliminary numerical experimentations are completed using the larger problems having 500 individual competitions and the 4 variants of the metaheuristic method introduced in Sect. 4. The best results are obtained with the following combination:

- $[t_{\min}, t_{\max}] = [[0.8N], [1.2N]]$
- $nitermax = N$
- $tempsmax = 3N$ seconds.

The additional parameter ρ (the number of the best solutions generated so far during the procedure) in the first metaheuristic method in Sect. 3 is fixed to the value $(M + 1)$. Finally, three values are considered for the parameter β (0.55, 0.65, and 0.75), and preliminary experimentations indicate that the best results are obtained with the value 0.65. Hence we complete the rest of the numerical tests using these values for the parameters.

Each problem is solved 10 times using different initial solutions to compare the efficiency of the eight variants. Note that each problem is solved only once with CPLEX 9.13 allowing a time limit of 10 hours. Moreover, in the variants **SN-H-First** and **SN-R-First** (**TS-H-First** and **TS-R-First**), for each resolution, we use different orders in which the judges and the individual competitions (the judges and the destination nodes) are considered. The efficiency of the different solution procedures is compared with respect to three different criteria:

- (i) *Ave dev*: the average deviation of the values of the solutions generated from the optimal value or from the lower bound
- (ii) *%Opt*: the percentage of runs achieving the optimal value or the lower bound for the different variants or the percentage of problems solved for CPLEX
- (iii) *Ave CPU (sec)*: the average CPU time over all runs for the different variants or the average CPU time over all problems solved in less than 10 hours (i.e., not reaching the time limit) for CPLEX.

Note that the first criterion does not apply for CPLEX as indicated by NA in Tables 2 and 3.

If we consider only the subsets of problems where CPLEX is able to solve the 10 problems (subsets of P_1 and P_3 with 15 individual competitions, and the subsets in P_2), the results in Table 2 indicate that the *Ave dev* is rather small for the eight variants. Furthermore, for these problems, there is always at least one variant having an *Ave CPU* smaller than CPLEX. Consequently, it seems to be worth using the metaheuristic approaches since they can generate solutions of very good quality requiring an *Ave CPU* smaller than CPLEX, in general.

Now, considering all the variants, we observe that they generate results of excellent quality since the *Ave dev* is always smaller than 1 (except the variant **SN-R-Best** for problems P_3 having 500 individual competitions). This means that on average, the solution generated for each problem includes at most one individual competition where one judge has the same field of expertise as another judge assigned to it.

5.1 Comparing the variants for the structured neighborhood metaheuristic method of Sect. 3

On the one hand, the variants **SN-H-Best** and **SN-H-First** dominate the variants **SN-R-Best** and **SN-R-First**, respectively, showing that it is worthwhile of initializing the solution ap-

Table 2 Comparing efficiency for problem sets

	Size	SN-H-		SN-R-		SN-H-		SN-R-		TS-H-		TS-R-		TS-H-		TS-R-		CPLEX
		Best	First	Best	First	Best	First	Best	First	Best	First	Best	First	Best	First			
<i>Ave dev</i>	P_1	15	0	0.03	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
P_2		15	0	0.17	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		150	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		500	0.03	0.01	0.04	0.03	0.04	0.12	0	0.13	0	0	0	0	0	0	0	NA
P_3		15	0	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		150	0	0.04	0	0	0	0	0	0	0	0	0	0	0	0	0	NA
		500	0	1.12	0.25	0.01	0.25	0.03	0	0.04	0	0	0	0	0	0	0	NA
<i>%Opt</i>	P_1	15	100	97	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		50	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	60
		150	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	50
		500	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	50
P_2		15	100	85	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		50	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		150	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
		500	97	99	97	96	91	91	100	90	100	100	100	100	100	100	100	100

Table 2 (Continued)

	Size	SN-H-		SN-R-		SN-H-		SN-R-		TS-H-		TS-R-		CPLEX
		Best	First	Best	First	Best	First	Best	First	Best	First	Best	First	
P_3	15	100	100	94	100	100	100	100	100	100	100	100	100	100
	50	100	100	100	100	100	100	100	100	100	100	100	100	60
	150	100	100	96	100	100	100	100	100	100	100	100	100	50
	500	100	100	70	99	89	89	97	96	96	96	100	100	40
Ave CPU (sec)	15	0.04	0.03	6.11	0.06	0.06	0.04	0.04	0.07	0.05	0.09	0.09	0.09	28.74
	50	0.15	0.18	0.17	0.23	0.23	0.17	0.17	0.47	0.17	0.37	0.37	0.37	2.31
	150	0.46	0.48	0.32	1.07	1.07	0.46	0.46	9.11	0.45	3.54	3.54	3.54	96.38
	500	11.26	12.23	1.30	26.98	26.98	12.34	12.34	411.99	11.40	85.98	85.98	85.98	12159.96
P_2	15	0.08	0.10	59.63	0.15	0.15	0.08	0.13	0.13	0.10	0.15	0.15	0.15	0.69
	50	1.25	1.01	1.82	1.13	1.13	1.05	1.05	1.05	1.17	0.97	0.97	0.97	1.93
	150	60.08	29.90	145.71	31.55	31.55	35.76	35.76	15.42	34.41	16.38	16.38	16.38	32.41
	500	1036.40	1354.39	1477.29	1246.47	1246.47	3189.12	3189.12	969.25	3327.33	667.98	667.98	667.98	21299.74
P_3	15	0.04	0.04	12.42	0.06	0.06	0.04	0.04	0.07	0.05	0.08	0.08	0.08	13.12
	50	0.21	0.21	0.30	0.33	0.33	0.19	0.19	0.48	0.17	0.44	0.44	0.44	8.85
	150	3.34	13.25	217.94	132.70	132.70	2.37	2.37	20.02	2.32	17.77	17.77	17.77	276.17
	500	436.64	770.37	3291.74	1655.56	1655.56	994.93	994.93	899.32	940.77	740.20	740.20	740.20	23824.47

Table 3 Comparing efficiency for problem sizes

Size	SN-H-	SN-R-	SN-H-	SN-R-	TS-H-	TS-R-	TS-H-	TS-R-	CPLEX
	Best	Best	First	First	Best	Best	First	First	
<i>Ave dev</i>									
15	0.00	0.09	0.00	0.00	0.00	0.00	0.00	0.00	NA
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	NA
150	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	NA
500	0.01	0.38	0.01	0.10	0.05	0.00	0.06	0.00	NA
<i>%Opt</i>									
15	100	92	100	100	100	100	100	100	100
50	100	100	100	100	100	100	100	100	73.33
150	100	98.67	100	100	100	100	100	100	66.67
500	99	89.67	98.67	95	96	100	95.33	100	63.33
<i>Ave CPU (sec)</i>									
15	0.05	26.06	0.06	0.09	0.05	0.09	0.06	0.11	14.18
50	0.54	0.76	0.47	0.56	0.47	0.67	0.50	0.60	4.36
150	21.29	121.32	14.55	55.10	12.86	14.85	12.39	12.56	134.99
500	494.76	1590.11	712.33	976.34	1398.80	760.19	1426.50	498.06	19094.72

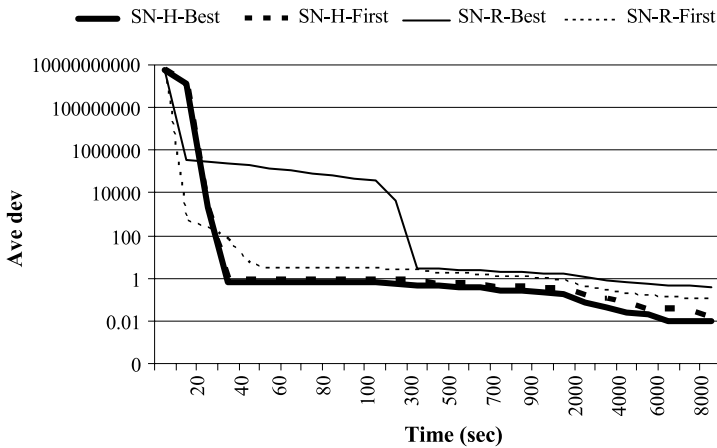


Fig. 7 Ave dev evolution of SN variants for problems of size 500

proach with a better initial solution. In particular, for the problems P_3 having 500 individual competitions, the $\%Opt$ is improved by more than 20%. On the other hand, when comparing the strategies for selecting the solution in the neighborhood of the current solution x , it is more difficult to verify that one is dominating the other.

To further analyze the relationship among the four variants of the structured neighborhood metaheuristic, we compare their performance according to the problem sizes in Table 3. For each pair of variants, we also carry out matched-pairs signed rank Wilcoxon tests (see the website <http://www.R-project.org>) with a 5% level of confidence over their numerical results. With regards to the *Ave dev*, the Wilcoxon tests confirm the results in Table 3 indicating that **SN-H-Best** and **SN-H-First** are not statistically different, but that they surpass the other two variants **SN-R-Best** and **SN-R-First**. Furthermore, the *Ave CPU* required by **SN-H-Best** is statistically smaller than **SN-H-First**, and the other two variants (having *Ave CPU* that are not statistically different).

Figure 7 illustrates the behavior of the *Ave dev* during the resolution. Each curve is associated with a different variant, and it shows the average values of the *Ave dev* over the 10 resolutions of the problems with 500 individual competitions calculated at different times of the resolution. During the first 30 seconds, the look of the curves indicates that the variants **SN-R-Best** and **SN-R-First** perform better than **SN-H-Best** and **SN-H-First**.

This is due to the fact that the heuristic procedure **HLA-HOA** is more time consuming to generate the initial solution of **SN-H-Best** and **SN-H-First** than the structured neighborhood Tabu search of phase 1 used to improve the initial solution generated almost instantaneously with the **Random_bias** procedure. Furthermore, during this period **SN-R-First** performs better than **SN-R-Best** probably because the greediness of this last variant when using the neighborhood structure $\{M + 1, M_1(x)\}$ reduces the impact when using the neighborhood structure $\{M + 1, M_3(x)\}$. Nevertheless, the CPU time required by **HLA-HOA** to generate the initial solution is worthwhile since when completed (i.e., after 30 seconds), the solutions of **SN-H-Best** and **SN-H-First** are better than **SN-R-Best** and **SN-R-First**.

After the first 30 seconds, the initial solution for the variants **SN-H-Best** and **SN-H-First** is so good that the structured neighborhood Tabu search of phase 1 has no impact. Hence the curves are levelling, and the solution is slightly improved by the Tabu search of phase 2 mostly after applying the diversification strategy. For the other two variants,

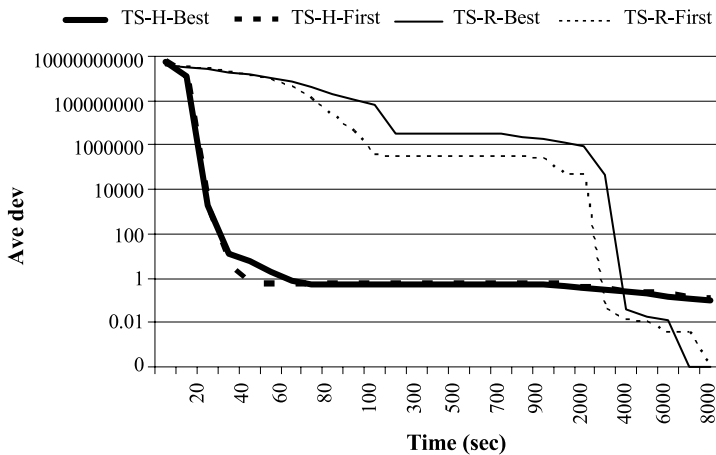


Fig. 8 Ave dev evolution of TS variants for problems of size 500

the curves indicate that larger improvements are obtained with the structured neighborhood Tabu search of phase 1 after applying the diversification strategy during the period from 30 to 300 seconds. After that time, their curves are also levelling, but at higher values of the *Ave dev* than **SN-H-Best** and **SN-H-First**.

In conclusion, the numerical results indicate a neat dominance of **SN-H-Best** over the three others that can be ranked as follows: **SN-H-First**, **SN-R-First**, and **SN-R-Best**.

5.2 Comparing the variants for the Tabu search method applied to a penalized version of the original model of Sect. 4

The results in Table 2 indicate a slight dominance of the variants **TS-R-Best** and **TS-R-First** (where the initial solutions are generated randomly) as far as the *Ave dev* and the *%Opt* are concerned since they always generate optimal solutions. We also refer to Table 3 to compare the different variants according to the problem sizes, and we use matched-pairs sign rank Wilcoxon tests to complete the following analysis. With regards to the *Ave dev*, the variants **TS-R-Best** and **TS-R-First** are not statistically different, and neither are the variants **TS-H-Best** and **TS-H-First**. But **TS-R-Best** and **TS-R-First** are statistically better than **TS-H-Best** and **TS-H-First**, respectively. Furthermore, the variant **TS-R-First** surpasses the other variants since its *Ave CPU* is statistically smaller than the others.

The curves in Fig. 8 are generated as in Fig. 7. The curves associated with the variants **TS-H-Best** and **TS-H-First** are quite similar to those of **SN-H-Best** and **SN-H-First**, but they are levelling at higher values of the *Ave dev*. Note that during the period between 30 and 60 seconds, **TS-H-First** performs better than **TS-H-Best**. This is due to the fact that, since only the diversity can be improved by this time, the Tabu search can identify very rapidly a modification inducing a gain of 1 with the *first improving strategy* while the best improvement that can be identified with the *best improving strategy* is only equal to 2.

The behavior of the curves associated with **TS-R-Best** and **TS-R-First** indicates that several applications of the diversification strategy are required before reaching the state where only the diversity can be improved. They surpass the other two variants since they can reduce the value of *Ave dev* to 0.

In conclusion, the numerical results indicate that the different variants should be globally ranked as follows: **TS-R-First**, **TS-R-Best**, **TS-H-First**, and **TS-H-Best**.

Table 4 Impact of the frequency

	Size	TS-H-Best	TS-R-Best	TS-H-First	TS-R-First
<i>G dev</i>	15	0.00	0.00	0.00	0.00
	50	0.00	0.00	0.00	0.00
	150	0.00	0.00	0.00	0.01
	500	0.02	0.07	0.14	0.08
<i>G CPU</i>	15	1.09	0.91	0.69	0.65
	50	0.85	1.89	2.49	2.03
	150	1.15	2.65	5.67	5.17
	500	0.73	1.46	1.64	2.94

Table 5 Comparing the best variants

Size	CPLEX		SN-H-Best		TS-R-First	
	% Opt	Ave CPU	% Opt	Ave CPU	% Opt	Ave CPU
15	100	14.18	100	0.05	100	0.11
50	73.33	4.36	100	0.54	100	0.60
150	66.67	134.99	100	21.29	100	12.56
500	63.33	19094.72	99	494.76	100	498.06

In order to evaluate the impact of the frequency $P(i, j, r, l)$ as a secondary selection criterion of the new current solution and in the diversification strategy, we solve the same problems without using it. Table 4 includes results comparing the two approaches where

- $G dev = (\text{value of Ave dev without frequency}) - (\text{value of Ave dev with frequency})$
- $G CPU = (\text{value of Ave CPU without frequency}) / (\text{value of Ave CPU with frequency})$.

The numerical results in Table 4 indicate that the frequency has a positive impact to improve the quality of the solutions for the largest problems, and even to reduce the *Ave CPU*, in general. Note that using the frequency has a negative impact on the *Ave CPU* for smaller problems. This is due to the fact that the *Ave CPU* for the smaller problem is so small that they fail to benefit of using the frequency in the long run.

5.3 Comparing the best variants of the methods with CPLEX

To compare the variants **SN-H-Best**, **TS-R-First**, and CPLEX, we refer to the results summarized in Table 5 specifying the %Opt and the *Ave dev* for the different problem sizes (with 15, 50, 150, and 500 individual competitions). These results clearly indicate the advantage of using the metaheuristic methods instead of CPLEX, but it is not so easy to decide between the two variants of the metaheuristic methods. Indeed, the matched-pairs sign rank Wilcoxon tests indicate that the two variants are not statistically different with respect to their *Ave dev*, but that **SN-H-Best** requires an *Ave CPU* statistically smaller than **TS-R-First**.

In summary, both variants outperform CPLEX and generate solutions of excellent quality. But with respect to the solution time required, it seems that according to the Wilcoxon tests, the variant **SN-H-Best** is slightly dominating.

6 Conclusion

In this paper we introduce two metaheuristic approaches for assigning judges to individual competitions in the context of a round of the John Molson International Case Competition. The structured neighborhood approach depends more strongly on the problem structure (to determine the different neighborhood structures used in the first stage) than the second one where a Tabu search is applied to a penalized version of the model where the constraint violations are included in the objective function. On the one hand, the diversification strategy in the first approach uses an adaptive memory of the best solutions found so far where two solutions are selected to be combined using a uniform crossover operator in order to generate a new initial solution to reinitialize the method. On the other hand, the diversification strategy in the second approach strongly depends on the constraints violations. Hence its efficiency is reduced when the number of constraints violated decreases. Furthermore, the frequency based secondary selection criterion used in the second approach to modify the current solution is beneficial but not as powerful as expected to diversify the search.

These approaches can be adapted to other contexts by making proper adjustments to deal with slightly different specific rules of assignment. In particular, it seems that the second approach would be more easily extended than the first one since it does not depend as strongly on the problem structure. For instance, additional soft rules can be dualized by introducing associated penalty terms in the objective function.

We are currently extending the approaches to solve the problem associated with the 5 rounds of the John Molson International Case Competition. Adjustments are required to account for additional constraints connecting the round sub problems in order to reduce the number of rounds where a judge evaluates the same team and to reduce the number of rounds where the same pair of judges works together, for instance. These results should be included in a forthcoming publication.

References

- Bhadury, J., Mighty, E. J., & Damar, H. (2000). Maximizing workforce diversity in project teams: a network flow approach. *Omega*, 28, 143–153.
- Duarte, A. R., Ribeiro, C. C., & Urrutia, S. (2006). Referee assignment in sports tournaments. In *Proc. of the 6th international conference on the practice and theory of automated timetabling PATAT*, Brno, Czech Republic, pp. 394–397.
- Duarte, A. R., Ribeiro, C. C., & Urrutia, S. (2007a). A hybrid ILS heuristic to the referee assignment problem with an embedded MIP strategy. *Lecture Notes in Computer Science*, 4771, 82–95.
- Duarte, A. R., Ribeiro, C. C., Urrutia, S., & Haeusler, E. H. (2007b). Referee assignment in sports leagues. *Lecture Notes in Computer Science*, 3867, 158–173.
- Dumais, S. T., & Nielsen, J. (1992). Automating the assignment of submitted manuscripts to reviewers. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, Copenhagen, Denmark, pp. 233–244.
- Evans, J. R. (1988). A microcomputer-based decision support system for scheduling umpires in the American baseball league. *Interfaces*, 18(6), 42–51.
- Evans, J. R., Hebert, J. E., & Deckro, R. F. (1984). Play ball! – the scheduling of sports officials. *Perspectives in Computing*, 4(1), 18–29.
- Glover, F., & Laguna, M. (1998). *Tabu search*. Boston: Kluwer.
- Goldberg, D. E. (1989). *Genetic algorithm in search, optimization, and machine learning*. Reading: Addison Wesley.
- Goldsmith J., & Sloan, R. H. (2007, to appear). The AI conference paper assignment problem. In *Proc. AAAI 2007 workshop on preference handling for artificial intelligence*.
- Hansen, P. (1986). *The steepest ascent mildest descent heuristic for combinatorial programming*. Presented at Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy.

- Hansen, P., & Mladenovic, N. (2001). Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130, 449–467.
- Hartvigsen, D., Wei, J. C., & Czuchlewski, R. (1999). The conference paper-reviewer assignment problem. *Decision Sciences*, 30(3), 865–876.
- Kuo, C. C., Glover, F., & Dhir, K. S. (1993). Analysis and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, 24, 1171–1185.
- Lamghari, A., & Ferland, J. A. (2007). Structured neighborhood Tabu search for assigning judges to competitions. In D. Srinivasan (Ed.), *Proc. of IEEE symposium on computational intelligence in scheduling (CI-Sched 2007)* (pp. 238–245). Honolulu.
- Lamghari, A., & Ferland, J. A. (2005). Heuristic techniques to assign judges in competitions. In V. Prahlad, W. W. Tan, A. P. Loh (Eds.), *Proc. of the third international conference on computational intelligence, robotics and autonomous System (CIRAS)*, Singapore.
- Landa-Silva, D., & Burke, E. K. (2007). Asynchronous cooperative local search for the office space allocation problem. *INFORMS Journal on Computing*, 19(4), 575–587.
- Sampson, S. E. (2006). Optimization of volunteer labor assignments. *Journal of Operations Management*, 24, 363–377.
- Schirrer, A., Doerner, K. F., & Hartl, R. F. (2007). Reviewer assignment for scientific articles using memetic algorithms. In K. F. Doerner, M. Gendreau, P. Greistorfer, W. Gutjahr, R. F. Hartl, & M. Reimann (eds.), *Metaheuristics, progress in complex optimization systems* (pp. 113–134). New York: Springer.
- Sun, Y., Ma, J., Fan, Z., & Wang, J. (2008). A hybrid knowledge and model approach for reviewer assignment. *Expert Systems with Applications*, 34(2), 817–824.
- Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J.D. (Ed.) *Proc. of the third international conference on genetic algorithms* (pp. 2–9). San Mateo: Morgan Kaufmann.
- Taillard, E. (1991). Robust Tabu search for the quadratic assignment problem. *Parallel Computing* 17, 443–455.
- Weitz, R. R., & Lakshminarayanan, S. (1998). An empirical comparison of heuristic methods for creating maximally diverse groups. *Journal of Operational Research Society*, 49, 635–646.
- Wright, M. B. (1991). Scheduling english cricket umpires. *Journal of the Operational Research Society*, 42(6), 447–452.
- Yavuz, M., Inan, U. H., & Figlali, A. (2008). Fair referee assignments for professional football leagues. *Computers and Operations Research* 35, 2937–2951.