Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling (CI-Sched 2007)

Structured Neighborhood Tabu Search for Assigning Judges to Competitions

Amina Lamghari and Jacques A. Ferland

Abstract— **A metaheuristic approach including three different stages is introduced to assign the judges for the John Molson International Case Competition. The complexity of the mathematical formulation accounting for the rules to be followed in assigning the judges, leads us to use such an approach. The two different Tabu search methods in the first two stages are combined with a diversification strategy. Numerical results are provided to indicate the efficiency of the approach to generate very good solutions.**

I. 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.

This competition involves 30 teams of business students coming from top international universities. This set of teams is partitioned into 5 groups, each including 6 teams. The first part of the competition consists in 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 three best teams move to the finals in the second part of the competition.

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 his 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 as possible
  - 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 the problem of forming maximally diverse groups (FMDG). This problem consists on 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 [1], and heuristic procedures have been proposed for the problem specified in different contexts by several authors, see [2]. More recently, in [3], the authors use a network flow formulation (the dining problem) to solve efficiently the (FMDG). Since our problem is even more complex, in [4], we introduce heuristic procedures to construct good feasible solutions for this problem. In this paper we use a more powerful metaheuristic based on the Tabu Search principle [5] [6] to generate better solutions.

Manuscript received October 13, 2006. This work was supported by NSERC Grant (OGP0008312) from Canada

Amina Lamghari is with the Département d'informatique et de recherche opérationnelle of the Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, CANADA H3C 3J7 (e-mail: lamghara@iro.umontreal.ca).

Jacques A. Ferland is with the Département d'informatique et de recherche opérationnelle of the Université de Montréal, C.P. 6128, Succ. Centre-Ville, Montréal, CANADA H3C 3J7 (Corresponding Author. Phone: 514-343-5687; fax: 514-343-5834; e-mail: ferland@iro.umontreal.ca).

In Section II we introduce a mathematical formulation and a metaheuristic procedure including three different stages. Section III includes a description of the methods used in the different stages. 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. Numerical results are given in Section IV. Four different variants are compared numerically. These variants are specified according to the process generating the initial solution, and to the strategies for selecting the solution in the neighborhood of the current solution. The numerical results indicate the efficiency of the approach to generate very good solutions.

II. MODEL AND SOLUTION APPROACH

Referring to the rules for assigning the judges in a specific round, 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 times that several judges assigned to an individual competition share 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, $i = 1, 2, \ldots, N$

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

$A = [a_{ij}]$ 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 admissible for individual competition j if i did not receive a degree or is not a faculty member of neither team universities

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

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

The mathematical model can be summarized as follows:

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

$$Subject\ to \sum_{j=1}^{M} x_{ij} \le 1 \qquad i = 1,\ldots,N \qquad (2)$$

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

$$\sum_{i=1}^{N} l_i x_{ij} \ge 1 \qquad j = 1,\ldots,M \qquad (4)$$

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

$$y_3^j + y_5^j = 1 \qquad j = 1,\ldots,M \qquad (6)$$

$$x_{ij} = 0 \text{ or } 1 \qquad \begin{matrix} i = 1,\ldots,N \\ j = 1,\ldots,M \end{matrix} \qquad (7)$$

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

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

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

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

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

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

In the objective function (1), we minimize the number of times that several judges assigned to an individual competition share 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).The constraints (5) and (6) guarantee that 3 or 5 judges are assigned to each individual competition.

The complexity of the model leads us to use a metaheuristic approach solving an equivalent model where the surplus of judges (if any) is assigned to a dummy individual competition (M+1) requiring no *lead* judge and for which all judges are admissible. Furthermore, in this new model, individual competitions with only one judge assigned are feasible but incur a very high cost $R >> 5M$. To simplify the notation, the number of judges with field of expertise k assigned to individual competition j is denoted

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

Furthermore, for all $j = 1,\ldots, M$, the variables

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

The new mathematical model can be summarized as follows:

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

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

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

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

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

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

$$x_{ij} = 0 \text{ or } 1 \qquad \begin{matrix} i = 1,...,N \\ j = 1,...,M \end{matrix} \quad (15)$$

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

For any feasible solution x, we denote

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

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

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

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

The solution procedure is a metaheuristic including three main stages. It is initiated with a feasible solution for the problem. 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). 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). 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 best solution generated is not improved during *itermax* successive iterations. 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 $\sum_{j=1}^{M}\sum_{k=1}^{K}\max\{d_{kj}-1,0\}$ is equal to 0 or to a lower bound known a priori for the problem.

III. Solution Methods for the Different Stages

In this section, we describe the solution methods used in the different stages.

A. Initial Solution

In this paper we consider two different processes to generate an initial solution. In the first one denoted **Random,** 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.

The second process is the **HLA-HOA** heuristic method introduced in [4]. The **HLA** method 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 **HOA** is a two phases 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, a second pair is assigned to as many individual competitions as possible.

B. Structured 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: $\exists \bar{i} \in I(j,x) - \{i,r\}$ 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:

$$LT_{ij} = curiter + t_1$$
$$LT_{rj} = curiter + t_2$$

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 $x \oplus (i, r, j, l)$ is Tabu at the iteration *iter* if

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

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_{out}, M_{in}, denote by $\{M_{out}, M_{in}\}$ the subset of feasible reassignments from $j \in M_{out}$ to $l \in M_{in}, j \neq l$. 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 I.

TABLE I
PARTITION OF *V*(*X*)

p	$V_p(x)$	$\Delta_p(x)$
1	$\{M+1, M_1(x)\}$	$-R$
2	$\{M_5(x), M_1(x)\}$	$-R+5M-2$
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	$\{M_3(x), M_3(x)\}$	$-5M+R-1$
8	$\{M_3(x), M+1\}$	$R-2$

We use different neighborhood structures as in a variable neighborhood search [7], 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 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 $\Delta_p = \min_{(i,r,j,l) \in V_p} \Delta(i, r, j, l)$ be the best modification that can be induced by any reassignment $(i, r, j, l) \in V_p(x)$. Referring to the values Δ_p given in the third column of Table I, it follows that the subsets are ordered in increasing order of these values. Furthermore, the reassignments in the first 3 subsets are *improving reassignments* ($\Delta_p < 0$, 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* ($\Delta_p > 0$, p = 6, 7, 8). This partition leads to the following search strategy of the neighborhood. The Tabu search in 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 dummy 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 tough 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.

C. 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) judge i is admissible for individual competitio n l ($a_{il} = 1$), and judge r for individual competitio n j ($a_{rj} = 1$);

ii) there exists a *lead* judge in individual competitio ns j and l

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

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

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:

$$LT_{ij} = curiter + t_1$$

$$LT_{rl} = curiter + t_2$$

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

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

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 $\sum_{j=1}^{M} \sum_{k=1}^{K} \max\{d_{kj} - 1, 0\}$ is equal to 0 or to a lower bound known a priori for the problem.

D. Diversification Strategy in Stage 3

Once the first two stages of the solution procedure are completed, and the best solution generated is not optimal, 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 applying a uniform crossover operator to the pair of solutions *xbest* and $\bar{x} \neq xbest$ selected randomly in Γ. More specifically, select randomly a subset of $\left\lceil \frac{\beta M}{countiter+1} \right\rceil$ individual competitions $M_{best} \subset \{1, \cdots, M+1\}$, 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 $j = 1, \cdots, M+1$,

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

Now x^0 may not be feasible because the same judge i may be assigned to two different individual competitions $j_1 \in M_{best}$ and $j_2 \notin M_{best}$; i.e., $i \in I(j_1, x^0) \bigcap I(j_2, x^0)$. Each such judge i is eliminated from j_1 or j_2 as follows. In order to favor the presence of the assignments found in *xbest*, 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 some 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 $U = \{j : j \neq M+1, \text{and } j \text{ has no } lead \text{ judge assigned}\}$.

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 assigned to another individual competition $\bar{j}$ having several *lead* judges assigned or assigned to $M + 1$, and if $a_{rj} = 1$, then assign r to $\bar{j}$ and i to l.

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

In phase 2, denote

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

At each iteration, select randomly $l \in W$. 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.

IV. NUMERICAL RESULTS

Four different variants are compared numerically. These variants are specified according to the process generating the initial solution (**HLA-HOA** and **Random** denoted **H** and **R**,

respectively), 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): **H-Best**, **H-First**, **R-Best**, and **R-First**. Furthermore, since (1) – (8) is a linear binary model, we try to solve it using CPLEX 9.0 for the sake of comparison with the four variants.

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.

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 [100, 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.

The four variants are implemented in Java, and the tests are completed on a 2 GHz processor AMD Athlon 3200$^+$ with 2 GB of memory working under a Linux operating system.

The constant R (used in the model to penalize the number of individual competitions having only one judge assigned) and the parameter ρ (the number of the best solutions generated so far during the procedure) are fixed to the values $50M^2$ and (M +1), respectively. To fix the other parameters, we consider the 18 different combinations generated with the two values for the interval $[t_{\min}, t_{\max}]$ ($[\lfloor 0.8N \rfloor, \lceil 1.2N \rceil]$ and $[\lfloor 0.9N \rfloor, \lceil 1.1N \rceil]$), the three pairs of values for (*nitermax*, *itermax*) ((*N*, 15), (5*N*, 10), and (10*N*, 5)), and the three values for β (0.55, 0.65, and 0.75). For each combination, each problem is solved once with the variant **R-First**. The best results are obtained with the following combination:

- $[t_{\min}, t_{\max}] = [\lfloor 0.8N \rfloor, \lceil 1.2N \rceil]$
- (*nitermax*, *itermax*) = (*N*, 15)
- $\beta = 0.65$.

Hence we complete the rest of the numerical tests using these values for the parameters.

Afterward, each problem is solved 5 times using different initial solutions to compare the efficiency of the four variants. Note that each problem is solved only once with CPLEX. Moreover, in the two variants **R-First** and **H-First**, for each resolution, we use different orders in which the judges and the individual competitions are considered. The efficiency of the four variants is compared with respect to three different criteria:

i. Average deviation:

Ave dev: the average deviation of the values of the solutions generated from the optimal value or from the lower bound.

ii. Number of problems where the optimal value or the lower bound is achieved. For each set of problems we compute:

NB_1: the number of problems where the optimal value or the lower bound is achieved for at least one of the 5 solutions generated

NB_5: the number of problems where the optimal value or the lower bound is achieved for each of the 5 solutions generated

$$Opt_1(\%) = \frac{NB_1}{10} \times 100$$

$$Opt_5(\%) = \frac{NB_5}{10} \times 100.$$

iii. Average solution time (CPU)

For each set of problems we compute the average CPU time *Ave CPU* (sec.) over all the resolutions.

Note that the first two criteria do not apply to CPLEX as indicated by NA in Tables II and III.

The numerical results for these criteria are summarized in Tables II and III where a column is associated with each method. In Table II, the results are given for each subset of problems. These results are used to compute the values of the different criteria for each problem size (15, 50, 150, and 500) given in Table III.

Consider the numerical results generated with CPLEX. Referring to Table II, we observe that CPLEX can solve all the problems in the set P_2 (for which there always exists a solution where all competitions have 5 judges assigned), but that it fails to solve several problems in the sets P_1 and P_3 due to running out of memory. Furthermore, the failure rate is non decreasing with the size of the problems.

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 II indicate that the *Ave dev* is rather small for the

four variants. Furthermore, the *Ave CPU* of CPLEX is smaller than that of the four variants for problems in the subset of P_2 with 50 individual competitions (except for the variant **R-First)** and in the subset of P_2 with 150 individual competitions, but it is larger for the other subsets. To verify if the *Ave CPU* of some method *A* is significantly smaller than that of some other method *B,* we can carry out a statistical analysis based on the nonparametric method of the Wilcoxon signed-rank test [8] on the numerical results. In our case, this test indicates with a 5% level of confidence that the CPU of CPLEX is not different than that of each variant in the subsets of P_2 with 50 and 150 individual competitions, and that it is different in the other subsets except for the variant **R-Best** in the subset of P_2 with 15 individual competitions.

TABLE II
COMPARING EFFICIENCY FOR PROBLEM SETS

		Size	H-Best	R-Best	H-First	R-First	CPLEX
Ave dev	P_1	15	0	0.04	0	0	NA
		50	0	0	0	0	NA
		150	0	0	0	0	NA
		500	0	0	0	0	NA
	P_2	15	0	0.14	0	0	NA
		50	0.08	0	0.04	0	NA
		150	0.14	0.4	0	0	NA
		500	0.14	0.04	0.02	0.02	NA
	P_3	15	0	0.02	0	0	NA
		50	0	0.04	0	0	NA
		150	0	0.56	0.02	0.22	NA
		500	0.02	0.92	0.06	0.12	NA
Opt5%	P_1	15	100	80	100	100	NA
		50	100	100	100	100	NA
		150	100	100	100	100	NA
		500	100	100	100	100	NA
	P_2	15	100	70	100	100	NA
		50	80	100	80	100	NA
		150	80	40	100	100	NA
		500	60	80	90	90	NA
	P_3	15	100	90	100	100	NA
		50	100	90	100	100	NA
		150	100	70	90	70	NA
		500	90	30	90	90	NA
Ave CPU (sec)	P_1	15	0.03	0.04	0.02	0.03	93.80
		50	0.24	0.33	0.24	0.28	3.27[(1)]
		150	0.62	0.61	0.59	0.57	85.57[(2)]
		500	10.71	1.37	10.33	6.64	7714[(3)]
	P_2	15	0.05	0.24	0.04	0.08	0.65
		50	2.14	2.36	2.34	1.62	1.95
		150	43.61	85.01	68.63	61.22	31.24
		500	1651.40	1457.04	4009.32	3395.30	15696.50
	P_3	15	0.02	0.04	0.03	0.03	7.34
		50	0.29	0.77	0.26	0.57	10389.48[(4)]
		150	2.80	64.45	21.17	103.13	314.97[(5)]
		500	660.99	3831.36	4777.32	7391.73	13698.85[(6)]

[(1)] Only 6 of the 10 problems were solved
[(2)] Only 2 of the 10 problems were solved
[(3)] Only 2 of the 10 problems were solved
[(4)] Only 7 of the 10 problems were solved
[(5)] Only 5 of the 10 problems were solved
[(6)] Only 2 of the 10 problems were solved

Consequently, it seems to be worth using the metaheuristic approach since it can generate solutions of very good quality and since it is faster than CPLEX, in general.

Now, considering the four variants, we observe that they generate results of excellent quality since the *Ave dev* is always smaller than 1. This means that on the 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. Furthermore, the three variants **H-Best**, **H-First**, and **R-first** are very *robust* in the sense that for each problem, the value of $Opt_1(\%)$ is equal to 100%, indicating that the optimal value or the lower bound is achieved for at least one of the 5 solutions generated. The variant **R-Best** is also quite *robust* since this is also true for all problems except for 2 instances in the subset of problems P_3 with 150 individual competitions and for 1 in the subset of problems P_3 with 500 individual competitions. (Note that the values of $Opt_1(\%)$ are not reported in Tables II and III for this reason.)

To further clarify the relationship among the four variants, we can apply the Friedman test [8] to the set of solutions generated by the four variants for each problem size. This test shows that with a 5% level of confidence, a statistically significant difference exists among the results except for the problems of size 50. This is in line with the results in Table III indicating that for problems of size 50, the four variants are quite competitive in terms of solution quality and solution time. Hence we refer to the problems of size 15, 150 and 500 for comparing the variants. Note that this is the reason why we elected to use also problems of larger dimension than those found in the specific application of the John Molson International Case Competition to complete the tests.

TABLE III
COMPARING EFFICIENCY FOR PROBLEM SIZES

		H-Best	R-Best	H-First	R-First	CPLEX
Ave dev	15	0.000	0.067	0.000	0.000	NA
	50	0.027	0.013	0.013	0.000	NA
	150	0.047	0.320	0.007	0.073	NA
	500	0.053	0.320	0.027	0.047	NA
Opt5 %	15	100	80	100	100	NA
	50	93.3	96.7	93.3	100	NA
	150	93.3	70	96.7	90	NA
	500	83.3	70	93.3	93.3	NA
Ave CPU (sec)	15	0.03	0.11	0.03	0.05	33.93
	50	0.89	1.15	0.95	0.82	3464.90[(7)]
	150	15.68	50.02	30.13	54.97	143.93[(7)]
	500	774.37	1763.26	2932.32	3597.89	12369.78[(7)]

[(7)] The average reported is taken over the problems solved

On the one hand, the variants **H-Best** and **H-First** dominate the variants **R-Best** and **R-First**, respectively,

showing that it is worthy of initializing the solution approach with better initial solution. 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. Indeed, **H-First** and **R-First** generate better solutions but require more solution time than **H-Best** and **R-Best**, respectively.

Consider the variants **H-First** and **H-Best**. For problems of size 150, the variant **H-First** can improve by a factor of 7 the *Ave dev* of the variant **H-Best**, but the solution time increases by a factor of 2. Similarly, for problems of size 500, the *Ave dev* is improved by a factor of only 2, but the solution time is increased by a factor of 4. Hence the improvement of the solution quality induced by the variant **H-First** seems to be more expensive in solution time as the problem size increases. Furthermore, considering the facts that the *Ave dev* of the variant **H-Best** is always smaller than 1 and that this variant is very robust, it seems more interesting to use this variant than the variant **H-First** as the problem size increases. Similar conclusion can be drawn when comparing **R-First** and **R-Best**.

Note that all the conclusions above rely on average values (*Ave dev* or *Ave CPU*). But whenever necessary we can carry out a matched-pairs signed-rank Wilcoxon test to verify that the performance (*Ave dev* or *Ave CPU*) of two different variants are statistically different or not. Now in each case discussed above, the result of the test indicates that the hypothesis is verified with a 5% level of confidence.

In summary, all the variants generate solutions of excellent quality, but considering the solution time required, it seems that the variant **H-Best** is slightly dominating the others.

V. CONCLUSION

In this paper we introduce a metaheuristic approach for assigning judges to individual competitions in the context of a round of the John Molson International Case Competition. This approach can be adapted to other contexts by making proper adjustments to deal with slightly different specific rules of assignment. For instance, additional soft rules can be dualized by introducing associated penalty terms in the objective function. All the variants of the approach are very efficient, but the variant **H-Best** is slightly dominating the others.

We are currently extending the approach 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. We are also testing the approach using real data obtained from the John Molson International Case Competition organization, and the people seem to be fully satisfied by the results produced. These results should be included in a forthcoming publication.

REFERENCES

[1] C.C. Kuo, F. Glover, K.S. Dhir, "Analysis and modeling the maximum diversity problem by zero-one programming," *Decision Sciences*, vol. 24, pp. 1171 – 1185, Nov./Dec. 1993.

[2] R.R. Weitz, S. Lakshminarayanan, "An empirical comparison of heuristic methods for creating maximally diverse groups," *Journal of Operational Research Society*, vol. 49, pp. 635 – 646, June 1998.

[3] J. Bhadury, E.J. Mighty, H. Damar, "Maximizing workforce diversity in project teams: a network flow approach," *Omega*, vol. 28, pp. 143 – 153, April 2000.

[4] A. Lamghari, J.A. Ferland, "Heuristic techniques to assign judges in competitions," in *Proc. of the Third International Conference on Computational Intelligence, Robotics and Autonomous System* (CIRAS), Singhapore, December 2005.

[5] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, June 1998.

[6] P. Hansen, "The steepest ascent mildest descent heuristic for combinatorial programming," presented at *Congress on Numerical Methods in Combinatorial Optimization*, Capri, Italy, 1986.

[7] P. Hansen, N. Mladenovic, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, pp. 449 – 467, May 2001.

[8] R Development Core Team, "R: A language and environment for statistical computing". Available: http://www.R-project.org