Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 174 (2006) 1459–1478
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A simulated annealing and hill-climbing algorithm for the traveling tournament problem

A. Lim [a], B. Rodrigues [b,*], X. Zhang [a]

[a] Department of IEEM, Hong Kong University of Science and Technology, Clearwater Bay, Kowloon, Hong Kong
[b] School of Business, Singapore Management University, 469 Bukit Timah Road, Singapore 259756, Singapore

Received 6 June 2003; accepted 22 February 2005
Available online 16 June 2005

Abstract

The Traveling Tournament Problem (TTP) [E. Easton, G. Nemhauser, M. Trick, The traveling tournament problem description and benchmarks, in: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, CP 2001, 2001, pp. 580–584; M. Trick, Challenge Traveling Tournament Problems, 2004] schedules a double round-robin tournament to minimize the total distance traveled by competing teams. It involves issues of feasibility and optimality and is a challenge to constraint and integer programming. In this work, we divide the search space and use simulated annealing (SA) to search a timetable space and hill-climbing to explore a team assignment space. The SA component mutates timetables using conditional local jumps to find timetables which lead to better schedules while hill-climbing is enhanced by pre-computation and dynamic cost updating to provide fast and efficient search. Computational experiments using this hybrid approach on benchmark sets give results comparable to or better than current best known solutions.

Keywords: Tournament; Scheduling; Heuristics

1. Introduction

The problem of scheduling a tournament in which all participants compete against each other one-on-one in a series of rounds goes back, as does competitive activity itself, to antiquity. Such round robin tournament scheduling is common and comes with varying rules in a number of sports [5,6]; in particular, a double round robin Traveling Tournament Problem (TTP) proposed by Easton et al. [4] has recently attracted research interest aimed at developing good schedules for sports such as minor league baseball, college basketball and professional football. A comprehensive review

* Corresponding author. Fax: +65 68220777.
E-mail address: br@smu.edu.sg (B. Rodrigues).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.02.065

of such problems and of research conducted on the problem is given in [4].

The TTP was developed for requirements of Major League Basketball in the United States. On one hand, the objective of minimizing distances traveled by leagues has to be achieved. Teams travel on road trips as they visit opponents and then return home. On the other hand, feasibility constraints imposed by rules applied to the sport make the resulting combinatorial optimization problem a very difficult one. Indeed, while most solution schemes are able to deal with the constraints separately, the problem is considerably more demanding when all constraints are involved for which significant effort is required even for small sized problems.

The problem has generated much research interest in recent years and challenge instances have been published [15]. Benoist et al. [3] combined Lagrangian relaxation and constraint programming for the problem, while Easton et al. [4] used an integer linear programming approach in which they modified a three-phased approach used by Nemhauser and Trick [11] in scheduling the Atlantic Coast Conference basketball league. Recently, Anagnostopoulos et al. [2] achieved good results for National League benchmark instances (NL8-16) given in [15] with a simulated annealing method enhanced by strategic oscillation and reheats. Their approach explores both feasible and infeasible schedules, and uses complex moves and large neighborhoods.

In this work, a simulated annealing with hill-climbing algorithm for the TTP is proposed. The method we develop was motivated by the three-phased approach of Nemhauser and Trick in which team assignments are handled after the timetable is fixed. We adapted the approach for TTP by parallelizing components which search for better timetables and better team assignments, instead of using these sequentially. The good performance of the hill-climbing component and the framework show that decomposing the problem into these components is effective. Computational experiments on benchmark sets provided by Trick [15] were conducted. These include the National League set (NL4-16) and the set described in [15] as Circular Distances instances (CIRC4-20). Results obtained are comparable to those currently available and are significantly better for the CIRC benchmarks.

2. Problem description

Recently, Easton et al. [4] introduced and defined the TTP: "Given n teams, n even, a round-robin tournament is a tournament among the teams so that every team plays every other team. Such a tournament has $n-1$ slots during which $n/2$ games are played. For each game, one team is denoted the home team and its opponent is the away team. As suggested by the name, the game is held at the venue of the home team (this differs form other situations where all teams travel to a single venue). A double round-robin tournament has $2(n-1)$ slots and has every pair of teams played twice, once at home and once away for each team."

For the problem, distances between team sites are given by an $n \times n$ distance matrix. Assuming equality of distances to and from sites, this matrix is symmetric. Each team begins the tournament at its home site to which it must return at the end of the tournament. Also, when a team plays an away game, it is assumed to travel from its home site to the away venue, and when playing consecutive away games, a team travels from one away venue to the next directly. The cost to each team is the total distance traveled starting from its home site and ending back there on completion of its scheduled games. The constraints for TTP are as follows:

1. *Double round-robin constraints*: Each pair of teams, A and B, say, play exactly twice-once at A's home site (denoted B@A) and once at B's home site (denoted A@B). Thus, there is a total $2(n-1)$ rounds, and in each round, $n/2$ games are played.
2. *Consecutive constraints*: For each team, no more than three consecutive home or three consecutive away games is allowed.
3. *No repeater constraints*: For any pair of teams, A and B, say, A@B cannot immediately follow B@A in the next round.

A schedule for all teams is a solution to the TTP when the above are satisfied, the cost of such a solution being the sum of the costs of every team. The objective is to find a schedule with minimum cost satisfying the constraints. We notice that although games played at home contribute nothing to the cost for any team, any length of repetitions of such games is checked by the consecutive constraint. We illustrate a schedule solution to the NL6 (6 teams) instance from [4] in Table 1: where ATL, NYM, PHI, MON, FLA and PIT are teams/sites and 0–9 represent rounds. In this schedule, ATL, for example, plays FLA, NYM, and PIT at home, then PHI, MON, and PIT away and then PHI and MON at home and NYM and FLA away.

3. Outline of solution approach

The strategy used here is to divide the search space into a timetable space and a team assignment space. The timetable space is explored by a simulated annealing (SA) algorithm, while the team assignment space is explored by a hill-climbing algorithm. Fig. 1 illustrates the framework of the approach and the interactions between its components.

A Controller fixes team assignments and calls on the SA component to generate better timetables. The timetable with best schedules is passed on to the hill-climbing component which searches it for better team assignments. Team assignments that give best schedules for the given timetable are then passed back to the SA component. The process continues until there is no improvement for a specified fixed number of consecutive cycles or when a time limit is reached. Here, the underlining idea is to look for better team assignments only for timetables with a higher chance of giving better schedules and to search for better timetables only for team assignments that have a higher chance of giving better schedules.

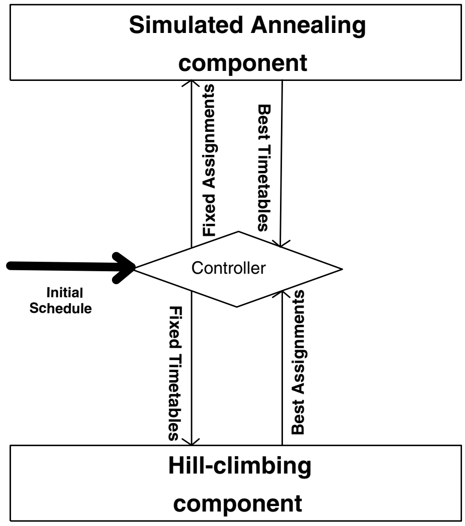

Fig. 1. Algorithm components.

With fixed team assignments, the SA component changes a given timetable with a local move to search for timetables with smaller total distances for the given team assignment. On the other hand, with fixed timetables, the hill-climbing component generates and improves team assignments searching for team assignments resulting in smaller total distances for the given timetable.

3.1. Initial solutions

Initial solutions were generated by the modified three-phase approach originally used to schedule the ACC league [11]. In this three-phase approach,

Table 1
A schedule for a league of 6 teams

Slot	0	1	2	3	4	5	6	7	8	9
ATL	FLA	NYM	PIT	@PHI	@MON	@PIT	PHI	MON	@NYM	@FLA
NYM	@PIT	@ATL	@FLA	MON	FLA	@PHI	@MON	PIT	ATL	PHI
PHI	@MON	FLA	MON	ATL	@PIT	NYM	@ATL	@FLA	PIT	@NYM
MON	PHI	@PIT	@PHI	@NYM	ATL	FLA	NYM	@ATL	@FLA	PIT
FLA	@ATL	@PHI	NYM	PIT	@NYM	@MON	@PIT	PHI	MON	ATL
PIT	NYM	MON	@ATL	@FLA	PHI	ATL	FLA	@NYM	@PHI	@MON

a *pattern* is a string consisting of H's (Home) and A's (Away), of length equal to $2 \times (n - 1)$, for a league of n teams. A set of n such patterns forms a pattern set. For example, HHHAAA is a pattern for the league of four teams, and {HHHAAA, HAAAHH, AAAHHH, AHHHAA} is a pattern set for the four teams.

Given a pattern set, games are assigned to the pattern set consistent with H, A letters resulting in a timetable. For example, for the pattern set {P0 = HHHAAA, P1 = HAAAHH, P2 = AAAHHH, P3 = AHHHAA}, a possible timetable could be:

P0: 3 2 1 3 2 1
P1: 2 3 0 2 3 0
P2: 1 0 3 1 0 3
P3: 0 1 2 0 1 2

where the entry x in column j of row i indicates that pattern i plays against pattern x in round j.

Given a timetable, the teams are assigned to patterns. The result from this stage is a schedule. For example, for the above pattern set and timetable, we could assign team 0 to P0, team 3 to P1, team 1 to P2, and team 2 to P3. This results in the following complete schedule:

0: 0 0 0 2 1 3
3: 3 2 0 1 3 3
1: 3 0 2 1 1 1
2: 0 2 2 2 3 1

where the entry x in column j of the row t indicates that team t plays at the home site of team x in round j.

In order to generate feasible solutions expediently for large n, we further modified the three-phase approach. In this modification, a double round robin tournament must have a round robin tournament in the first $(n - 1)$ slots and then have the same tournament with venues reversed in the second $(n - 1)$ slots. For example, if team A plays team B at A's home site in the Kth $(1 \leqslant K \leqslant n - 1)$ round, then team A must play team B at B's home site in the $(K + n - 1)$th round. The mirrored tournament necessitates that patterns are mirrored. That is, if a pattern has a H at the Kth$(1 \leqslant K \leqslant n - 1)$ position, then it must have an A at the $(K + n - 1)$th position, and vise versa. This modification greatly reduces the number of available patterns. For example, the number is reduced from 1972 without the mirrored constraints to 72 with the mirrored constraints for $n = 8$, and from 24857864 to 9328 for $n = 16$. For a given pattern set, this modification reduces the computation time required to generate feasible timetables from it. This is because we only need to assign games between patterns for the first $(n - 1)$ rounds, and then mirror these for the remaining $(n - 1)$ rounds.

The modified three-phase approach above generates initial solutions fast. However, the quality of resulting schedules is not guaranteed. In order to obtain schedules of better quality, we applied beam search with a look-ahead procedure to the TTP. Beam search [10,14] expands only the p most promising nodes at each depth level of breadth first search (BFS) and thus avoids the combinatorial explosion problem of BFS. In the TTP context, each of the $2(n - 1)$ rounds corresponds to a depth level of beam search. For the first round, the beam search procedure generates p game plans which are taken as initial parents. For every subsequent round, the procedure expands at most b children from each parent of the previous round. At this step, the b children (similarly, the p initial parents of the first round) are simply those game plans with least total travel distances (TTD) satisfying double round-robin, consecutive and no repeater constraints.

These children of the current round, denoted as the cth round, are then sorted according to their evaluation cost, and only the best p ones are retained and, in turn, serve as parent nodes for the $(c + 1)$th round. In view of the tight constraints of the TTP, a look-ahead procedure is necessary both to ensure the feasibility of resulting solutions and to improve their quality. More precisely, when calculating the evaluation cost of an expanded child, the procedure greedily constructs $d(d \leqslant d_{\max}$ and $c + d \leqslant 2n - 2$, where $d_{\max}$ is a pre-defined parameter) more rounds forward from the cth round. Let TTD denote the total travel distance from the 1st to cth rounds, and $ITD_i (1 \leqslant i \leqslant d)$ denote the travel distance between the $(c+i-1)$th and $(c+i)$th rounds. If $c+d=2n-2$, the travel distance necessary for away teams to return home after the $(2n-2)$th round is added to ITD_d. The evaluation cost, E_{val} in Eq. (1), sums up weighted distances so that TTD contributes the most while ITD_i has a higher weight than ITD_j if $i<j$. The underlining rationale of E_{val} (or equivalently, of the look-ahead procedure) is to take into account estimated future costs of current choices, instead of entirely depending on the TTD. In experiments, the parameters p, b and $d_{\max}$ were set as 1000, 10, and 6, respectively.

$$E_{\text{val}} = \text{TTD} + \sum_{i=1}^{d} \frac{ITD_i}{i+1} \tag{1}$$

Take the instance CIRC4 as an example and let $p=b=d_{\max}=2$. In Fig. 2, in the 2nd round, two child nodes are generated from each parent node inherited from the 1st round. The look-ahead procedure is then applied to all four children and computes their evaluation costs. With the evaluation cost array [7,8], the algorithm selects the 1st and 3rd child for the 2nd round, which results in the beam search tree as illustrated in Fig. 3. The same process is repeated for the remaining rounds until it reaches complete schedules.

3.2. The simulated annealing component

In implementing SA [8], moves that lead to a solution worse than the current solution are accepted with probability $P = \exp^{-\text{cost}(S')-\text{cost}(S)/T} = \exp^{-\Delta/T}$, where cost($S$) denotes the cost of the current solution S, cost(S') denotes the cost of the next solution S', Δ is the difference of these costs and the temperature T is a control parameter in the same units as the cost function. For more on SA, see [1,7,9,12].

In the algorithm, the SA component takes feasible timetables generated by the modified three-phase approach or a beam search procedure as the initial solution. The initial temperature (T)

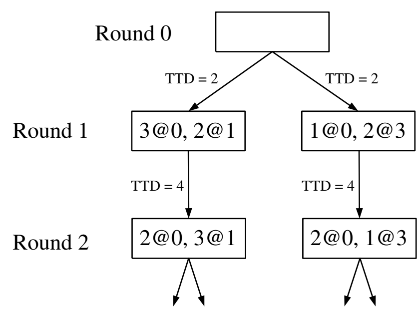

Fig. 3. The beam search tree.

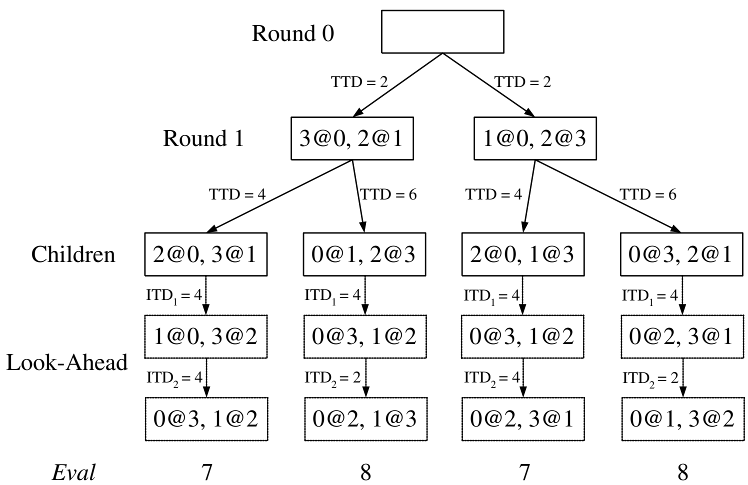

Fig. 2. The look-ahead procedure.

varies for the NL and CIRC sets. Preliminary experiments with fewer test cases and shorter running times showed that this should be approximately 400 for NL4-16 set and 3 for CIRC4-20 set for best results. The stop criterion of SA terminates iterations when the cost function is unchanged for a specified number (40) of consecutive temperatures. The inner loop criterion of SA executes the loop a constant number (500) of times and the new temperature is calculated by $T' = rT$ with $r = 0.99$.

The remaining components for the SA algorithm are the neighborhood and the cost evaluation function. The neighborhood of a timetable consists of timetables generated by performing a conditional local jump on it. With fixed team assignments, the cost of a timetable is the total distance traveled, plus total penalties in case the schedule is infeasible.

Given a timetable, $P_\alpha@P_\beta$ is a *match* of the ith round if P_α plays against P_β in the ith round. A *match set* of the ith round is a set of one or more such matches of the ith round. For a match set, its *projected pattern set* is the set of patterns that participate in its matches. Two match sets are equal if and only if their projected pattern sets are identical.

Local jump is the operation of exchanging two equal match sets between two rounds. For example, if $S1 = \{P1@P2, P3@P4\}$ is a match set of round 1 and $S2 = \{P1@P3, P2@P4\}$ is a match set of round 2, then one possible local jump is to exchange S1 and S2, i.e. to move $S1 = \{P1@P2, P3@P4\}$ to round 2 and move $S2 = \{P1@P3, P2@P4\}$ to round 1. It is clear that local jumps do not violate double round-robin constraints.

Conditional local jump (CLJ) is a local jump that satis.es the consecutive and no repeater constraints and provides a way to mutate timetables. The neighborhood of a timetable comprises those timetables generated by applying a CLJ to it. The next timetable S' is selected randomly from the neighborhood of the current timetable S. Moreover, the feasibility of a CLJ is established by ensuring that the constraints of those teams involved in the move are not violated. The entire resulting schedule is checked for feasibility. From the efficiency point of view, only those constraints concerning the teams in the move and the neighboring rounds are checked.

In initial implementation, after receiving a timetable from the Controller, the SA component generates all possible match sets for each round. Generally, for n teams, there are $(2^{n/2} - 1)$ match sets for each round and $(2n - 2) \times (2^{n/2} - 1)$ match sets altogether for a given timetable. For computational and storage efficiency, each match set is hashed to an integer k, where $0 < k < 2^n$. The neighborhood, i.e., the set of possible conditional local jumps, is built from these hashed match sets. If a move is accepted, only the match sets of the two rounds concerned with the conditional local jump are rebuilt. The neighborhood is then updated with the newly rebuilt match sets and the unchanged match sets. If the move is not accepted, then the neighborhood is unchanged.

In a more advanced implementation, the moves are generated without the need to explicitly enumerate all match sets of a given timetable. The algorithm randomly selects two rounds, r_i and r_j, and computes all the connected components in the associated graph as suggested in [2]. More precisely, in the graph associated with r_i and r_j, the vertices are the patterns, and there will be an edge connecting P_α and P_β for any match $P_\alpha@P_\beta$ in r_i or r_j. A connected component is a *valid* one if and only if the move to exchange the matches concerning the patterns in the connected component is a valid conditional local jump. For each valid connected component, the algorithm randomly assigns a Boolean flag to it to indicate whether it is to be included in the move. Thus, the final move is to exchange the matches concerning all the patterns in the selected components. For given r_i and r_j, the time complexity to generate a move is $O(n)$ with the fact that the size of the edge set in the associated graph is $O(n)$. In the worse case, the algorithm needs to examine all possible choices of (r_i, r_j). Therefore, the worse case complexity to generate a conditional local jump is $O(n^3)$, although the size of the neighborhood is $O(n^2 \times 2^{n/2})$ which is generally much larger than $O(n^3)$.

In an improved version of the algorithm, consecutive constraints are relaxed, and conditional local jump is a local jump that satisfies the no repeater constraints. For each team, if the consecutive constraint is violated, a penalty is added to the cost function. The penalty value is taken to be the estimated repair cost of violations. More precisely, given a Home/Away pattern, the algorithm identifies all maximum consecutive sequences of H's or A's in the pattern. For a consecutive sequence of length $L (L > 3)$, the number of violations is estimated by $\lfloor L \rfloor$. For each violation, the penalty is set as 1.5 times the maximum distance between the team and any other team. For example, if the maximum distance from team t to any other team is d and the pattern of team t is HHAAAAHHHHHAAA, the penalty is taken to be $1.5d + 1.5d = 3d$. The sequence, AAAA, contributes one violation and thus leads to the penalty of $1.5d$. Similarly, HHHHH gives the additional penalty of $1.5d$. The sum of penalties from all teams is added to the total distance traveled to form the new cost function when consecutive constraints are relaxed. By temporarily relaxing the consecutive constraints, the algorithm is allowed explore a wider solution space and therefore has a higher chance of escaping local optima.

In the implementation, we use a simple reheating scheme for the SA algorithm. That is, after it freezes the SA algorithm takes the best schedules found so far as new initial solutions and restarts the search process with initial parameter settings, as long as the number of reheating is no more than R which was set as 200 in our implementation.

3.3. The hill-climbing component

In the final phase of the three-phase approach, a team is assigned to each pattern for a given timetable, thus completing the schedule. A straightforward implementation by enumerating all possible permutations results in $n!$ schedules. For speed and efficiency, a hill-climbing algorithm with K random restarts, for a given K, is used. With fixed timetables, the hill-climbing component improves team assignments by reducing travel distances. The hill-climbing algorithm is applied only to the feasible timetables, i.e., those satisfying the three sets of constraints.

After receiving a timetable from the Controller, the hill-climbing component randomly assigns a team to each pattern to generate the initial schedule, and evaluates the cost of the schedule. An assignment is a function σ from teams to patterns. The local move for the hill-climbing component, which we call a local exchange, is an exchange between assignments of two teams. For example, a local exchange (A, B) produces a new assignment σ' identical to σ except for A and B that are switched, i.e., $\sigma'(A) = \sigma(B)$ and $\sigma'(B) = \sigma(A)$.

The hill-climbing algorithm changes assignments of teams to patterns by means of local exchanges, and moves in the direction of decreasing cost where only local exchanges that lead to better schedules are accepted. The hill-climbing component uses a first-accept strategy, i.e., it accepts a move immediately as long as it leads to an improvement. The hill-climbing algorithm continues to reduce cost until it reaches a local optimum where no further reduction is possible. It then restarts with a new randomly generated initial team assignment. After K such random restarts, the hill-climbing model terminates and returns the best schedule it found to the SA model. In experiments, the parameter K was set to be 20.

The local moves in the SA component are similar to [2]. More specifically, the neighborhood of conditional local jump contains the neighborhoods of SwapHomes, SwapRounds and PartialSwapRounds of [2]. However, conditional local jumps allow additional moves that exchange two or more connected components between two rounds. The local exchange operator is essentially the same as the SwapTeams operator in [2] while the PartialSwapTeams operator is not used here.

3.3.1. Precomputation

To make the hill-climbing model more efficient, pre-computation and dynamic cost updates are used. Given a timetable, the hill-climbing component pre-computes and stores the travel information in an $n \times n$ integer matrix $\mathbf{M}$ before performing any local exchange. Each entry, $\mathbf{M}_{ij}$, of $\mathbf{M}$ indicates that teams need to travel between the home site of pattern i and the home site of pattern j $\mathbf{M}_{ij}$ number of times according to the given timetable, regardless of how assignments from teams to patterns are made. For example, for the pattern set in Table 2(a) (P0, P1, P2 and P3) and the timetable in Table 2(b) (where a number "x"

Table 2
A timetable and its travel matrix

Patterns	Rounds					
	0	**1**	**2**	**3**	**4**	**5**
(a) Pattern set						
0	H	H	H	A	A	A
1	H	A	A	A	H	H
2	A	A	A	H	H	H
3	A	H	H	H	A	A
(b) Timetable						
0	3	2	1	@3	@2	@1
1	2	@3	@0	@2	3	0
2	@1	@0	@3	1	0	3
3	@0	1	2	0	@1	@2

	Patterns			
	0	**1**	**2**	**3**
(c) Travel matrix				
0	0	**2**	**1**	5
1	2	**0**	**4**	2
2	1	**4**	**0**	3
3	5	**2**	**3**	0

for pattern i in round j indicates that pattern i plays against pattern x in round j and that the game venue is the home site of pattern i and "@x" indicates that pattern i plays against pattern x at the home site of pattern x), the travel matrix **M** is given in Table 2(c). The trips for the 4 patterns are given in Fig. 4(a)–(d) where each arc represents a move required by the timetable. For example, the arc from P0 to P3 in Fig. 4(a) with the label "$R_2 \to R_3$" indicates that the team needs to travel from the home site of P0 to that of P3 between Round 2 and Round 3. The number of edges connecting each pair of patterns is given in Fig. 4(e). Thus, each entry, $\mathbf{M}_{ij}$, of **M** represents the number of edges between P_i and P_j.

Each time the hill-climbing component randomly generates the initial schedule, the cost is computed from the original timetable once. With **M**, checking whether a local exchange is downhill or uphill only requires computing the cost change due to the local exchange. With the cost change, the new cost is just the old cost plus the cost change. More precisely, let $A_1, A_2, A_3, \ldots, A_n$ be the teams assigned to P1, P2, P3, $\ldots$, P$_n$, respectively. The cost increase, Δ, incurred by exchanging the team assignments P$_x$ and P$_y$($1 \leqslant x \leqslant y \leqslant n$) is defined in Eqs. (2) and (3), where **D** is the distance matrix and C is the travel distance of A_x or A_y before the operation. C' is computed by the same formula used for C but with A_x and A_y swapped, and hence is the travel distance of A_x or A_y after the exchange operation. The difference between C and C', Δ, is therefore the increase in distance resulting from the local exchange. With regard to time complexity, for given x and y, $O(n)$ time is required to check if the corresponding local exchange is an uphill or downhill one. Moreover, the time complexity to determine if the hill-climbing process has reach a local optimum is $O(n^3)$.

$$\Delta = C' - C, \tag{2}$$

$$C = \sum_{k=1}^{n}(\mathbf{M}_{k,x} \times \mathbf{D}_{A_k,A_x}) + \sum_{k=1}^{n}(\mathbf{M}_{k,y} \times \mathbf{D}_{A_k,A_y}) \tag{3}$$

Using the travel matrix in Table 2(c) as an example, in a local exchange between the team assignment of P1 and that of P2, only the bold entries in Table 2(c) affect the cost change. The remaining entries have no effect on the change, and are not accessed by the algorithm. Generally, only 2 of

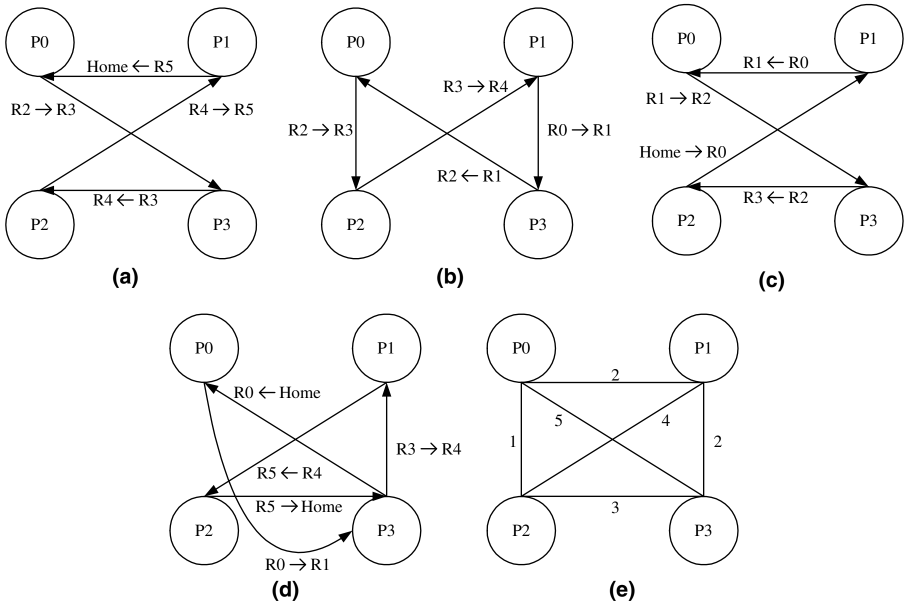

Fig. 4. Trips of Patterns: (a) The Trip of P0; (b) The Trip of P1; (c) The Trip of P2; (d) The Trip of P3; (e) All Trips.

the n columns of the matrix elements are accessed for each local exchange and, as n increases, the percentage of columns not accessed becomes larger, which leads to more significant time savings.

We note that generating an optimal team assignment for a timetable is a Quadratic Assignment Problem (QAP). Here, the travel matrix corresponds to the flow matrix, while the distance matrix remains as in a QAP formulation. Although it was possible to use algorithms available for the QAP, preliminary testing on standard QAP instances (from QAPLIB) showed that the hill-climbing component with a local exchange operator developed here was adequately effective. Also, it was computationally efficient and hence suited for use in the framework where it is invoked a number of times.

4. Computational experiments

The design and implementation of the approach given here has evolved from [16]. Better ways of generating initial schedules by a beam search with a look-ahead procedure, improving of the implementation of neighborhood structure and the relaxation of the consecutive constraints have been developed. We ran the algorithm on the two benchmark data sets available from [15]. Each of the 16 benchmark instances is run on a 2.53-GHz Pentium 4 PC with 512 Mb of RAM. These include the National League set NL4-16 and another set described in [15] as Circular Distances instances and denoted CIRC4-20. These arise from the embedded aspects of traveling salesman problem in the TTP. In the computational experiments, the algorithm was tested with multiple initial solutions generated by beam search. The experiment for each test instance was divided into stages with equal amounts of time allocated to each stage. In the initial stage, 128 initial solutions were used and resulting schedules sorted according to total travel distances, and the best half retained for the next stage. This was repeated in the subsequent stages, where altogether there were 8 stages with 128, 64, 32, 16, 8, 4, 2 and 1 solution(s) to be improved in each stage.

4.1. Initial temperatures

To investigate how initial temperature (T) influences solution quality, experiments using NL8-16 and CIRC8-20 were conducted with T equal to 0.5, 1, 2, 3, 4, 5, 10, 20, 40, 60, 80, 100, 200, 400, 600, 800, 1000, 2000. The values 0.5, 1, 2, 3, 4, 5, 10 were chosen because the average distance between two team sites for CIRC4-20 ranged from 1.00 to 5.00. Similarly, the values 100, 200, 400, 600, 800, 1000, 2000 were selected since the average distances between two team sites for NL4-16 ranged from 392.00 to 1119.98. In addition, the the values 20, 40, 60, 80 were included to represent intermediate temperatures. These 18 values provided a reasonable base to investigate the effect of initial temperatures. The average gaps of the NL and CIRC data sets for these values are reported in Fig. 5. From Fig. 5, the best results occurred at $T_b = 400$ for the NL set, and $T_b = 3$ for the CIRC set. For both sets of data, the average gap increased as T decreased when $T < T_b$, possibly because the initial temperatures here reduced the chances of algorithm getting out of local optima. When $T > T_b$, the performance was deteriorated as T increased. Here, it is conceivable that the algorithm moved into bad regions with the relatively high initial temperatures and failed to return to good regions when the temperature was frozen. As Fig. 5 shows, the average distance between two team sites must be taken into consideration when determining the initial temperature for an instance, which justifies setting distinct initial temperatures, even if they are far apart, for the NL and CIRC data sets.

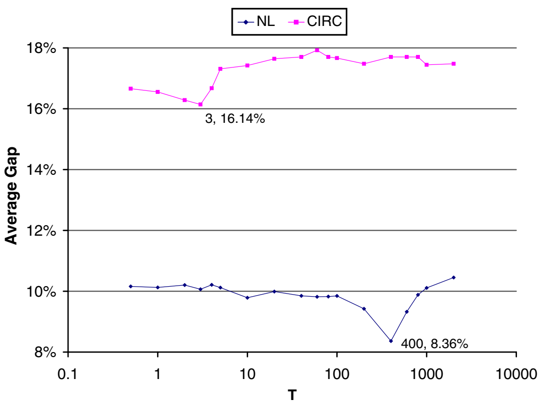

Fig. 5. The effect of initial temperatures on the solution quality.

4.2. Initial schedules

The distance and running time from 10 runs of the beam search algorithm are given in Table 3, where LB are lower bounds from [15]. When compared to results given at [15], the quality of the initial solutions is satisfactory. For example, the algorithm was able to find a schedule that is better than best-known results for the instance CIRC10. More importantly, they provide good starting points for the SA and hill-climbing components. Meanwhile, the required running time is reasonable and does not produce a significant overhead for the overall algorithm.

In Fig. 6, the two procedures for finding initial schedules, the modified three-phase (MTP) approach and the beam search (BS) approach, are compared using NL8-16 and CIRC8-20. The average gaps of the initial solutions (Stage 0) and the average gaps at the end of each improvement stage are given in Fig. 6 for both approaches. From Fig. 6, it is clear that the BS approach produced much better initial solutions than MTP. With the same running time, the average gap was reduced from 21.20% to 11.79% for the NL set, and from 36.38% to 18.77% for the CIRC set. Meanwhile, the final solution quality was improved by using the BS procedure. The average gap was reduced from 7.28% to 6.59% for the NL set, and the improvement, from 22.73% to 14.91%, was more significant for the CIRC set. As Fig. 6 shows, for both NL and CIRC sets, the graphs for BS are always below those for MTP, which indicates that with the same running time the algorithm with BS produced better solutions. In addition, the algorithm with BS required less running time to produce comparable results. For example, at the end of Stage 4 it was able to reduce the average gap to 7.20%, close to the final solution quality (7.28%) produced by the algorithm with MTP. As Fig. 6 shows, the BS procedure improved both initial and final solution quality.

4.3. Analysis and comparisons

Given its randomized nature, the algorithm was run 10 times with initial solutions in Table 3. The

Table 3
Analysis of initial schedules

Instance	Distance					Time (seconds)			
	LB	Min	Avg	Max	SD	Min	Avg	Max	SD
NL4	8276	8276	8310.8	8559	83.51	1	1.6	2	0.49
NL6	23 916	24 579	24 802.8	25 146	156.24	576	628.5	669	24.50
NL8	39 479	41 265	41 852.6	42 334	317.12	3211	3345.0	3443	65.84
NL10	57 500	63 337	64 544.2	65 856	750.41	5801	6179.6	6299	138.75
NL12	107 483	118 047	120 528.8	123 770	1977.56	8009	8594.0	8730	200.94
NL14	182 797	202 916	205,929.3	208 797	1912.35	10 332	10 806.4	10 947	173.87
NL16	248 852	283 795	289 235.2	295 864	3591.61	12 855	13 631.2	13 889	275.81
CIRC4	20	20	20.0	20	0.00	1	1.8	2	0.40
CIRC6	64	64	65.0	66	1.00	594	647.8	682	33.60
CIRC8	128	136	139.0	142	1.84	3177	3272.1	3339	48.74
CIRC10	220	**252**	261.0	266	3.71	5880	5939.1	6022	42.86
CIRC12	384	430	441.2	446	4.21	7987	8147.7	8287	110.94
CIRC14	588	680	686.8	698	5.74	10 798	10 877.2	10 952	43.49
CIRC16	832	986	996.0	1010	6.99	12 708	13 066.0	13 251	133.37
CIRC18	1188	1390	1404.4	1430	10.91	15 315	16 091.9	16 400	274.29
CIRC20	1400	1886	1903.6	1924	11.13	18 262	19 187.3	19 505	321.80

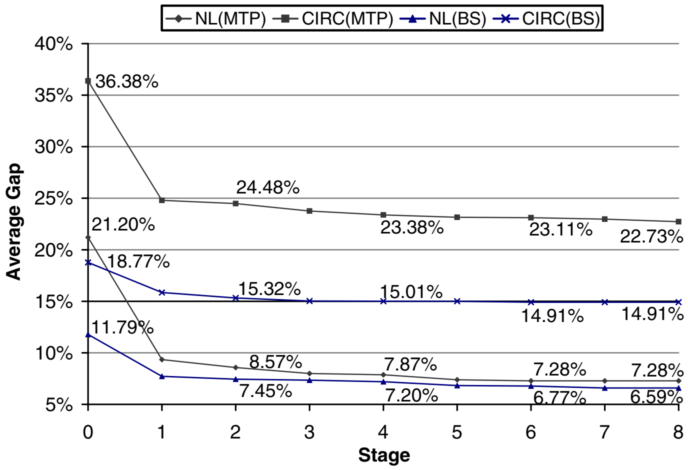

Fig. 6. The effect of initial schedules on the final solution quality.

resulting distances from the 10 runs are analyzed in Table 4 where the gap percentage is computed by $(\frac{\text{Distance}}{\text{Lower bound}} - 1)$ for each instance. The quality of the schedules is comparable to those results given in [15] at the time of writing (June 28, 2004). The difference between the results in the "Min" column and the best results in [15] is in the range from -3.64% to 3.61%, with an average of 1.51% for the NL set and -1.55% for the CIRC set. The performance of the algorithm is relatively stable since the difference between maximum and minimum distances for each instance ranged from 0% to 5.21% with an average of 1.93%. The running times of the 10 runs are given in Table 5. In Table 5, "Total Time" indicates the total time of each run including the time to generate initial solutions and "Best Time" denotes the time when the algorithm found the best solutions. In general,

Table 4
Analysis of schedules from 10 runs

Instance	LB	Min		Avg		Max		SD
NL4	8276	**8276**	0%	8276.0	0%	8276	0%	0
NL6	23 916	**23 916**	0%	23 916.0	0%	23 916	0%	0
NL8	39 479	**39 721**	0.61%	39 721.0	0.61%	39 721	0.61%	0
NL10	57 500	59 821	4.04%	60 375.0	5.00%	61 426	6.83%	552.72
NL12	107 483	115 089	7.08%	116 792.3	8.66%	118 677	10.41%	1069.59
NL14	182 797	196 363	7.42%	197 769.9	8.19%	199 137	8.94%	731.52
NL16	248 852	274 673	10.38%	278 477.9	11.91%	281 401	13.08%	1885.53
CIRC4	20	**20**	0%	20.0	0%	20	0%	0
CIRC6	64	**64**	0%	64.0	0%	64	0%	0
CIRC8	128	**132**	3.13%	132.0	3.13%	132	3.13%	0
CIRC10	220	**246**	11.82%	252.0	14.55%	256	16.36%	2.68
CIRC12	384	**408**	6.25%	418.0	8.85%	428	11.46%	6.69
CIRC14	588	**664**	12.93%	670.4	14.01%	676	14.97%	3.77
CIRC16	832	**954**	14.66%	970.0	16.59%	986	18.51%	8.39
CIRC18	1188	**1356**	14.14%	1370.4	15.35%	1388	16.84%	11.62
CIRC20	1400	**1842**	31.57%	1859.2	32.80%	1872	33.71%	9.93

Table 5
Analysis of running times of 10 runs

Instance	Total time (seconds)				Best time (seconds)			
	Min	Avg	Max	SD	Min	Avg	Max	SD
NL4	24 577	24 577.6	24 578	0.49	1	1.7	3	0.64
NL6	37 440	37 492.5	37 533	24.50	577	821.2	1416	224.81
NL8	52 363	52 497.0	52 595	65.84	3224	4106.8	6278	862.08
NL10	67 241	67 619.6	67 739	138.75	8557	40 289.4	61 083	17 761.24
NL12	81 737	82 322.0	82 458	200.94	10 355	54 026.5	79 262	21 954.35
NL14	96 348	96 822.4	96 963	173.87	21 978	59 019.7	92 901	22 794.11
NL16	111 159	111 935.2	112 193	275.81	50 767	83 287.0	110 342	18 645.08
CIRC4	24 577	24 577.8	24 578	0.40	1	1.8	2	0.40
CIRC6	37 458	37 511.8	37 546	33.60	594	648.3	682	33.27
CIRC8	52 329	52 424.1	52 491	48.74	3340	3729.9	4213	299.90
CIRC10	67 320	67 379.1	67 462	42.86	5886	23 022.8	64 195	19 612.50
CIRC12	81 715	81 875.7	82 015	110.94	8709	37 947.1	73 962	22 257.57
CIRC14	96 814	96 893.2	96 968	43.49	11 022	53 751.0	88 988	25 744.46
CIRC16	111 012	111 370.0	111 555	133.37	13 261	56 036.9	109 719	34 834.93
CIRC18	125 907	126 683.9	126 992	274.29	16 199	63 872.2	108 351	29 782.67
CIRC20	141 142	142 067.3	142 385	321.80	50 303	68 807.0	130 226	22 513.15

the running time of the algorithm is comparable to algorithms that produced the best known upper bounds. According to [15], the authors of [2,13] reported the majority of online best known solutions. In [2], the authors reported the average running time as 233578.35 seconds for NL14 and 192086.55 seconds for NL16 on an AMD Athlon machine at 1544 MHz. In Table 5, the corresponding time is 96822.4 seconds for NL14 and 111935.2 seconds for NL16. Taking into account machine difference, our computation effort is comparable to [2]. In [13], the authors reported the time to find the best solution as 22078.5 seconds for NL14 and 42290.8 seconds for NL16 on a 2.0-GHz Pentium 4 machine with 512 MB of RAM. In Table 5, the average running time required to find the best solution is 59019.7 seconds for NL14 and 83287.0 seconds for NL16. Consid-

Table 6
Comparisons of best results

Instance	LB	Previous best	New best	Gap (%)	Difference (%)
NL4	8276	8276	*8276*	0	0
NL6	23916	23916	*23916*	0	0
NL8	39479	39721	*39721*	0.61	0
NL10	57500	59583	59821	4.04	0.41
NL12	107483	111248	115089	7.08	3.57
NL14	182797	189766	196363	7.42	3.61
NL16	248852	267194	274673	10.38	3.01
CIRC4	20	20	*20*	0	0
CIRC6	64	64	*64*	0	0
CIRC8	128	132	*132*	3.13	0
CIRC10	220	254	**246**	1.82	−3.64
CIRC12	384	420	**408**	6.25	−3.13
CIRC14	588	670	**654**	11.22	−2.72
CIRC16	832	976	**928**	11.54	−5.77
CIRC18	1188	1364	**1356**	14.14	−0.67
CIRC20	1400	1882	**1842**	31.57	−2.86

ering the machine difference, our computation effort to find the best solution is approximately 3 times of that of [13].

The best results achieved for the benchmark data, where for each instance the percentage gap is computed by $(\frac{\text{New best}}{\text{Lower bound}} - 1)$ and the difference is calculated by $(\frac{\text{New best}-\text{Previous best}}{\text{Lower bound}})$, is shown in Table 6. The average gap for the 16 instances is 7.45%. These results show that the algorithm is at least comparable to current and previous results in [15]: they are better than the best-known results for 6 instances, equal for 6 instances and worse off for 4 instances. The difference is in the range from −5.77% to 3.61%, while the average value is −0.51%. New best schedules found are given in the Appendix A. For the CIRC data set, our results have improved, with an average improvement of −3.13%, or equaled previous best results.

5. Conclusion

The Traveling Tournament Problem schedules a double round-robin tournament to minimize the total distance traveled by competing teams. It involves issues of feasibility and optimality and is a very difficult problem for both constraint programming and integer programming. To search for schedules with smaller total distances, we divided the search space into two subspaces. A SA component searches the timetable space, while a hill-climbing component searches the team assignment space.

A hybrid approach combines SA and hill-climbing components with random restarts. The modified three-phased approach and the beam search procedure with look-ahead provide initial solutions for SA algorithm which changes timetables using conditional local jumps to search for timetables which lead to schedules with less total travel distance. The hill-climbing algorithm, enhanced by pre-computation and dynamic cost updating, provides a fast and efficient way to search in the team assignment space. Experiments have shown that the hybrid approach gives results which are comparable to or better than best solutions for benchmark sets.

Techniques develop here complement those found by other researchers, which together improve understanding of the effectiveness of the various approaches to this very difficult combinatorial optimization problem. Although one meta-heuristic is developed here, it is possible that other meta-heuristics or hybrids thereof, can be employed for this problem in the future.

Appendix A

The appendix contains new best-known schedules to TTP instances at [15] at the time of writing (June 28, 2004) (see Tables 7–13).

Table 7
CIRC8 distance: 132

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI
@CHI	PHI	@NYM	@CIN	@PIT	FLA	MON	ATL
NYM	@ATL	@CHI	@PIT	CIN	MON	@FLA	PHI
PHI	CHI	@ATL	CIN	PIT	@FLA	@MON	@NYM
@NYM	ATL	CHI	@FLA	MON	CIN	@PIT	@PHI
@PHI	@PIT	ATL	CHI	@CIN	NYM	FLA	@MON
@MON	@CIN	@PIT	ATL	@CHI	PHI	NYM	FLA
FLA	@CHI	@CIN	PIT	@ATL	@MON	PHI	NYM
CIN	PIT	@FLA	@CHI	PHI	@NYM	@ATL	MON
MON	CIN	PIT	@ATL	CHI	@PHI	@NYM	@FLA
@FLA	MON	CIN	@NYM	ATL	CHI	@PHI	@PIT
@PIT	@PHI	NYM	FLA	@MON	ATL	CHI	@CIN
@CIN	@FLA	@MON	PHI	NYM	@CHI	ATL	PIT
PIT	@MON	FLA	NYM	@PHI	@ATL	@CHI	CIN
CHI	FLA	MON	@PHI	@NYM	@CIN	PIT	@ATL

Table 8
CIRC10 distance: 246

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL
@MIL	@STL	@MON	PHI	CIN	CHI	@FLA	@PIT	NYM	ATL
@STL	@MIL	@PIT	CIN	CHI	PHI	@MON	@FLA	ATL	NYM
NYM	@ATL	@FLA	CHI	PHI	CIN	@PIT	@MON	@MIL	STL
STL	PIT	MON	@PHI	@CIN	@NYM	FLA	MIL	@ATL	@CHI
MON	STL	PIT	@ATL	@CHI	@PHI	MIL	FLA	@NYM	@CIN
@PHI	MON	ATL	@NYM	@PIT	FLA	CHI	@CIN	MIL	@STL
@MON	@FLA	@STL	ATL	NYM	@CHI	@MIL	PIT	PHI	CIN
@FLA	@MON	@CHI	NYM	ATL	@MIL	@STL	PHI	CIN	PIT
PIT	MIL	@CIN	@FLA	MON	@ATL	PHI	STL	@CHI	@NYM
FLA	CHI	MIL	@PIT	@ATL	MON	STL	@NYM	@CIN	@PHI
@NYM	ATL	CHI	@CIN	@MIL	STL	MON	@PHI	@PIT	FLA
CIN	@PHI	NYM	PIT	@STL	@MON	@ATL	@MIL	FLA	CHI
@CHI	CIN	@MIL	STL	PIT	@FLA	@NYM	ATL	@MON	PHI
@PIT	@CHI	CIN	MIL	STL	ATL	@PHI	NYM	@FLA	@MON
@CIN	@PIT	STL	@CHI	MIL	NYM	ATL	MON	@PHI	@FLA
CHI	@CIN	FLA	@STL	@PHI	MIL	NYM	@ATL	MON	@PIT
PHI	FLA	@ATL	@MIL	@NYM	@CIN	PIT	@STL	CHI	MON
MIL	PHI	@NYM	FLA	@MON	@STL	@CHI	CIN	PIT	@ATL

Table 9
CIRC12 distance: 408

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL	HOU	COL
@COL	@PHI	NYM	FLA	@MON	@CHI	STL	PIT	@CIN	@HOU	MIL	ATL
@HOU	@MON	FLA	NYM	@PHI	STL	@CHI	CIN	@PIT	@COL	ATL	MIL
COL	PHI	@NYM	@CIN	PIT	@FLA	MON	@STL	CHI	HOU	@MIL	@ATL
PHI	COL	@ATL	@PIT	CIN	MON	@FLA	@MIL	HOU	CHI	@STL	@NYM
@NYM	ATL	COL	@FLA	MON	CIN	@PIT	HOU	@MIL	STL	@CHI	@PHI
@MON	@HOU	PIT	ATL	@CIN	@PHI	FLA	MIL	@COL	@CHI	NYM	STL
@PHI	@COL	ATL	PIT	@CHI	@MON	MIL	FLA	@HOU	@CIN	STL	NYM
NYM	@ATL	@MON	PHI	@STL	MIL	CHI	@CIN	FLA	@PIT	COL	@HOU
MON	HOU	@PIT	@ATL	CHI	PHI	@MIL	@FLA	COL	CIN	@NYM	@STL
HOU	MON	@FLA	@NYM	PHI	CHI	@STL	@PIT	CIN	COL	@ATL	@MIL
@MIL	@FLA	MON	@PHI	NYM	@CIN	PIT	STL	@CHI	ATL	@COL	HOU
@CHI	@CIN	@HOU	COL	@PIT	FLA	NYM	ATL	MIL	@STL	PHI	@MON
@STL	@PIT	@MIL	@CHI	COL	NYM	HOU	MON	ATL	PHI	@CIN	@FLA
CHI	CIN	@COL	@STL	HOU	@MIL	@NYM	@ATL	MON	PIT	@FLA	PHI
PIT	MIL	CIN	HOU	STL	@ATL	@PHI	@COL	@FLA	@NYM	@MON	CHI
MIL	PIT	STL	CIN	@COL	@NYM	@MON	@HOU	@PHI	@ATL	CHI	FLA
@PIT	@MIL	@CIN	STL	@HOU	ATL	PHI	COL	@MON	NYM	FLA	@CHI
@CIN	@CHI	@STL	@HOU	@MIL	COL	ATL	NYM	PHI	FLA	MON	@PIT
@FLA	@STL	@CHI	@MIL	ATL	HOU	COL	PHI	NYM	MON	@PIT	@CIN
CIN	CHI	HOU	@COL	MIL	@STL	@ATL	@NYM	PIT	@FLA	@PHI	MON
FLA	STL	CHI	MIL	@ATL	@HOU	@COL	@PHI	@NYM	@MON	PIT	CIN
STL	FLA	MIL	CHI	@NYM	@COL	@HOU	@MON	@ATL	@PHI	CIN	PIT

Table 10
CIRC14 distance: 654

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL	HOU	COL	SF	SD
SD	@PHI	NYM	@FLA	MON	@MIL	@CHI	CIN	HOU	PIT	@STL	@SF	COL	@ATL
COL	@MON	@FLA	NYM	PHI	@STL	@MIL	HOU	PIT	CIN	@CHI	@ATL	SD	@SF
@SF	@FLA	@MON	PHI	NYM	@CHI	@STL	PIT	CIN	HOU	@MIL	@SD	ATL	COL
@COL	MON	FLA	@NYM	@PHI	MIL	CHI	@CIN	@HOU	@PIT	STL	ATL	@SD	SF
@SD	FLA	MON	@PHI	@NYM	CHI	MIL	@PIT	@SF	@CIN	COL	@HOU	STL	ATL
SF	PHI	@NYM	FLA	@MON	CIN	@PIT	MIL	@COL	@CHI	SD	STL	@ATL	@HOU
NYM	@ATL	SF	CIN	PIT	@FLA	@MON	@STL	CHI	SD	@COL	HOU	@PHI	@MIL
PHI	SF	@ATL	PIT	CIN	@MON	@FLA	@HOU	@MIL	STL	CHI	SD	@NYM	@COL
@NYM	ATL	@SD	@STL	@PIT	FLA	COL	@MIL	MON	CHI	@SF	@CIN	HOU	PHI
HOU	@SD	@SF	@CHI	@CIN	COL	FLA	MON	MIL	@STL	@ATL	@PIT	PHI	NYM
@PHI	@SF	ATL	@MIL	@CHI	@CIN	PIT	FLA	COL	MON	@SD	@STL	NYM	HOU
@FLA	@COL	CIN	SD	ATL	@HOU	@PHI	STL	@CHI	SF	PIT	NYM	@MIL	@MON
@MON	CIN	SD	ATL	CHI	@COL	@NYM	@FLA	SF	@HOU	MIL	PIT	@STL	@PHI
CIN	SD	@PIT	CHI	STL	PHI	@ATL	@MON	@FLA	@COL	SF	MIL	@HOU	@NYM
MON	COL	CHI	@ATL	@SD	STL	HOU	@PHI	@PIT	@SF	@CIN	@NYM	MIL	FLA
FLA	@MIL	COL	@SD	@ATL	HOU	STL	SF	@CIN	NYM	@PIT	@PHI	@CHI	MON
@MIL	@STL	PIT	COL	HOU	@PHI	SF	SD	NYM	ATL	@FLA	@MON	@CIN	@CHI
@STL	@HOU	@CHI	SF	MIL	SD	@COL	PHI	ATL	@FLA	NYM	CIN	@MON	@PIT
@HOU	MIL	@STL	@PIT	SD	MON	@SF	@COL	PHI	@NYM	ATL	CHI	CIN	@FLA
MIL	PIT	@HOU	STL	@COL	@NYM	@SD	@SF	@MON	@ATL	PHI	FLA	CHI	CIN
PIT	HOU	STL	@CIN	@SF	@ATL	MON	@SD	@PHI	COL	@NYM	@MIL	FLA	CHI
STL	@CHI	HOU	MIL	COL	SF	SD	NYM	@ATL	@MON	@PHI	@FLA	@PIT	@CIN
@PIT	@CIN	MIL	HOU	SF	ATL	NYM	COL	SD	@PHI	@MON	@CHI	@FLA	@STL
@CHI	@PIT	@COL	@SF	@STL	NYM	@HOU	ATL	FLA	@SD	CIN	PHI	MON	MIL
@CIN	CHI	@MIL	@COL	@HOU	@SF	ATL	@NYM	@SD	PHI	FLA	MON	PIT	STL
CHI	STL	@CIN	@HOU	@MIL	@SD	PHI	@ATL	@NYM	FLA	MON	SF	@COL	PIT

Table 11
CIRC16 distance: 928

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL	HOU	COL	SF	SD	LA	ARI
NYM	@ATL	MON	@PHI	PIT	@FLA	CHI	@CIN	MIL	@STL	COL	@HOU	SD	@SF	ARI	@LA
MON	PHI	@NYM	@ATL	CHI	LA	@SF	@FLA	COL	HOU	@MIL	@STL	CIN	ARI	@PIT	@ST
PHI	MON	@ATL	@MYM	LA	CHI	@SD	@PIT	HOU	COL	@STL	@MIL	ARI	CIN	@FLA	@SF
@NYM	ATL	@ARI	LA	@PIT	FLA	@COL	HOU	@MIL	STL	@CHI	CIN	@ST	SF	@MON	PHI
@MON	@PHI	NYM	ATL	@CHI	@CIN	PIT	FLA	@COL	@HOU	MIL	STL	@ARI	@LA	SD	SF
@PHI	MON	ATL	NYM	@CIN	@CHI	FLA	PIT	@HOU	@COL	STL	MIL	@LA	@ARI	SF	SD
SD	@FLA	ARI	@CIN	NYM	@STL	MON	@HOU	PIT	@SF	CHI	@LA	MIL	@ATL	CON	@PHI
HOU	ARI	@MIL	@PIT	CIN	MON	@FLA	STL	@CHI	PHI	@ATL	@SD	LA	COL	@SF	@NYM
ARI	HOU	@STL	FLA	@MON	CIN	@PIT	@MIL	PHI	CHI	@NYM	SF	@COL	LA	@SD	@ATL
@SD	FLA	@CHI	CIN	@LYM	STL	@MOL	PHI	@PIT	SF	@COL	HOU	@MIL	ATL	@ARI	LA
@LA	@ARI	FLA	PIT	@PHI	@MON	STL	MIL	@CIN	@CHI	SF	SD	HOU	@COL	ATL	NYM
@SF	@LA	PIT	@CHI	STL	@PHI	MIL	MON	@FLA	@CIN	SD	@ARI	ATL	@HOU	NYM	COL
COL	PIT	CIN	@STL	ARI	@NYM	@PHI	@SF	MON	SD	LA	@ATL	CHI	@MIL	@HOU	@FLA
CIN	COL	@SF	@MIL	HOU	ARI	@ATL	@SD	LA	MON	@FLA	@NYM	PHI	CHI	@STL	@PIT
SF	CIN	@SD	ARI	@STL	HOU	@NYM	@COL	FLA	LA	@PIT	CHI	@ATL	PHI	@MIL	@MON
@CHI	@PIT	@LA	SF	@COL	NYM	HOU	ATL	@ARI	@SD	@CIN	FLA	@MON	MIL	PHI	STL
@CIN	@CHI	SF	SD	@HOU	@COL	ATL	NYM	@LA	@ARI	FLA	PIT	@PHI	@MON	STL	MIL
@FLA	@CIN	SD	@ARI	ATL	@HOU	NYM	COL	@SF	@LA	PIT	@CHI	STL	@PHI	MIL	MON
CHI	SD	LA	@SF	COL	@MIL	@HOU	@ATL	ARI	PIT	CIN	@FLA	MON	@NYM	@PHI	@STL
LA	CHI	@FLA	@SD	PHI	COL	@STL	@NYM	CIN	ARI	@SF	@PIT	HOU	MON	@ATL	@MIL
FLA	LA	@PIT	CHI	@ATL	PHI	@MIL	@MON	SF	CIN	@SD	ARI	@STL	HOU	@NYM	@COL
@COL	@SD	@CIN	STL	@ARI	MIL	PHI	SF	@MON	@PIT	@LA	ATL	@CHI	NYM	SOU	FLA
@HOU	@COL	STL	MIL	@LA	@ARI	SF	SD	@PHI	@MON	ATL	NYM	@CIN	@CHI	FLA	PIT
@ARI	@HOU	MIL	@FLA	MON	@LA	SD	@STL	CHI	@PHI	NYM	@SF	COL	@CIN	PIT	ATL
PIT	SF	@MON	PHI	@MIL	@ATL	@CHI	CIN	@SD	FLA	ARI	LA	@NYM	STL	@COL	@HOU
STL	MIL	COL	HOU	SD	SF	LA	ARI	@ATL	@NYM	@MON	@PHI	@PIT	@FLA	@CIN	@CHI
MIL	STL	HOU	COL	SF	SD	ARI	LA	@NYM	@ATL	@PHI	@MON	@FLA	@PIT	@CHI	@CIN
@PIT	@SF	CHI	@LA	MIL	ATL	COL	@PHI	SD	@FLA	@ARI	@CIN	NYM	@STL	MON	HOU
@STL	@MIL	@HOU	@COL	@SF	@SD	@LA	@ARI	ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI
@MIL	@STL	@COL	@HOU	@SD	@SF	@ARI	@LA	NYM	ATL	MON	PHI	PIT	FLA	CHI	CIN

Table 12
CIRC18 distance: 1356

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL	HOU	COL	SF	SD	LA	ARI	T17	T18
@NYM	ATL	MON	@PHI	@PIT	FLA	CHI	@CIN	@HOU	SD	STL	SF	@COL	@MIL	ARI	@LA	T18	@T17
@PHI	MON	ATL	@NYM	@CIN	CHI	FLA	@PIT	@COL	SF	SD	STL	@MIL	@HOU	@T17	T18	LA	@ARI
@MON	PHI	@NYM	ATL	@CHI	CIN	@PIT	FLA	@MIL	STL	SF	SD	@HOU	@COL	@T18	T17	@ARI	LA
NYM	@ATL	@T18	CIN	PIT	@FLA	@MON	MIL	HOU	@CHI	@STL	@SF	COL	T17	@ARI	LA	@SD	PHI
PHI	@T18	@ATL	PIT	CIN	@MON	@FLA	HOU	MIL	@STL	@CHI	ARI	SD	@SF	T17	@COL	@LA	NYM
T18	@T17	PIT	@FLA	MON	@PHI	HOU	@STL	CHI	@COL	@CIN	MIL	ARI	LA	@SD	@SF	NYM	@ATL
@T17	T18	FLA	@PIT	@PHI	MON	STL	@MIL	@CIN	CHI	COL	@HOU	@LA	ARI	SF	@SD	ATL	@NYM
@ARI	FLA	T18	@CIN	@NYM	STL	MON	@HOU	@PIT	COL	CHI	@MIL	@T17	@LA	SD	ATL	SF	@PHI
@LA	T17	@PIT	FLA	@MON	PHI	@CHI	CIN	COL	@HOU	MIL	@STL	@T18	@ARI	ATL	SD	@NYM	SF
T17	@MON	@FLA	NYM	PHI	@CIN	PIT	STL	@CHI	@SD	@COL	HOU	LA	MIL	@SF	@T18	@ATL	ARI
ARI	@PIT	@MON	PHI	CHI	NYM	@STL	@FLA	CIN	@SF	@SD	LA	MIL	HOU	@COL	@ATL	@T18	T17
MON	@FLA	CHI	@ATL	NYM	@STL	@MIL	@PHI	PIT	CIN	@SF	@SD	HOU	COL	T18	@T17	ARI	@LA
@T18	CHI	SD	@T17	@LA	@MIL	@HOU	@NYM	SF	PIT	CIN	@ARI	@STL	@PHI	FLA	COL	MON	ATL
SD	@PHI	NYM	@T18	@ARI	@CHI	SF	PIT	@T17	HOU	@MIL	@LA	@CIN	@ATL	COL	FLA	STL	MON
FLA	SD	@CHI	LA	@ATL	SF	MIL	PHI	@ARI	@CIN	T18	T17	@PIT	@NYM	@MON	STL	@COL	@HOU
@SD	ARI	@CIN	MIL	LA	COL	PHI	T18	@SF	@MON	T17	@PIT	STL	ATL	@FLA	@NYM	@HOU	@CHI
@HOU	CIN	MIL	COL	ARI	T18	@NYM	@SD	LA	@PHI	ATL	@MON	T17	CHI	@STL	@FLA	@SF	@PIT
@MIL	@SD	ARI	@STL	COL	@T17	@T18	@SF	MON	ATL	@LA	@FLA	CHI	NYM	HOU	@PHI	PIT	CIN
PIT	@LA	CIN	@MIL	@STL	@ATL	@PHI	@COL	FLA	MON	@T18	CHI	@ARI	@T17	NYM	SF	SD	HOU
STL	@ARI	@LA	@CHI	@MIL	@T18	COL	MON	@ATL	FLA	@T17	@CIN	@SD	SF	PHI	NYM	HOU	PIT
LA	STL	@ARI	T17	@HOU	MIL	@SF	COL	@NYM	@PIT	FLA	@CHI	CIN	T18	@ATL	PHI	@MON	@SD
@PIT	LA	T17	STL	MIL	ATL	@COL	SD	@MON	@FLA	ARI	CIN	T18	@CHI	@NYM	@HOU	@PHI	@SF
@FLA	SF	LA	CHI	ATL	HOU	T17	@MON	SD	ARI	@PIT	T18	@NYM	@STL	@PHI	@MIL	@CIN	@COL
SF	@STL	@T17	@LA	HOU	@COL	@SD	ARI	NYM	@T18	@FLA	PIT	@ATL	CIN	MON	@CHI	PHI	MIL
COL	@CHI	HOU	@SF	@T18	@SD	@T17	NYM	@LA	@ARI	@PHI	@ATL	MON	PIT	STL	MIN	CIL	FLA
CIN	@HOU	COL	@ARI	@T17	@SF	@ATL	@T18	@SD	@LA	NYM	@PHI	PIT	STL	MIL	MON	FLA	CHI
@SF	COL	@STL	HOU	SD	LA	T18	@ARI	PHI	T17	@MON	@NYM	ATL	@FLA	@PIT	CHI	@MIL	@CIN
@COL	HOU	@MIL	ARI	SF	SD	LA	T17	T18	PHI	@NYM	ATL	@FLA	@PIT	@CIN	@MON	@CHI	@STL
HOU	@CIN	@SD	SF	@COL	@ARI	NYM	LA	T17	T18	@ATL	FLA	@MON	PHI	@CHI	PIT	@STL	@MIL
CHI	MIL	SF	@HOU	@SD	@LA	@ARI	@ATL	@T18	@NYM	MON	@T17	@PHI	FLA	PIT	CIN	COL	STL
MIL	PIT	STL	@SD	@SF	@NYM	@LA	@T17	@PHI	@ATL	@ARI	@T18	FLA	MON	CIN	HOU	CHI	COL
@CIN	@SF	@HOU	@COL	STL	ARI	ATL	@LA	@FLA	@T17	PHI	MON	NYM	@T18	CHI	@PIT	MIL	SD
@STL	@MIL	@COL	SD	T18	T17	ARI	SF	ATL	NYM	LA	PHI	@CHI	@MON	@HOU	@CIN	@PIT	@FLA
@CHI	@COL	@SF	T18	T17	@HOU	SD	ATL	ARI	LA	PIT	NYM	PHI	@CIN	@MIL	@STL	@FLA	@MON

Table 13
CIRC20 distance: 1842

ATL	NYM	PHI	MON	FLA	PIT	CIN	CHI	STL	MIL	HOU	COL	SF	SD	LA	ARI	T17	T18	T19	T20
@T20	@FLA	@MON	PHI	NYM	CIN	@PIT	@STL	CHI	HOU	@MIL	@SF	COL	T17	ARI	@LA	@SD	T19	@T18	ATL
@T18	@PIT	@FLA	CIN	PHI	NYM	@MON	@MIL	HOU	CHI	@STL	@LA	T17	ARI	COL	@SD	@SF	ATL	T20	@T19
@T19	@MON	@PIT	NYM	CIN	PHI	@FLA	@HOU	@MIL	STL	CHI	@SD	ARI	COL	T17	@SF	@LA	T20	ATL	@T18
T20	FLA	MON	@PHI	@NYM	@CIN	PIT	MIL	@HOU	@CHI	STL	SF	@COL	@T17	@ARI	LA	SD	@T19	T18	@ATL
T19	MON	FLA	@NYM	@PHI	@CHI	MIL	PIT	@COL	@CIN	SF	STL	@HOU	@ARI	@T17	SD	LA	@T20	@ATL	T18
T18	@PHI	NYM	FLA	@MON	@STL	CHI	@CIN	PIT	SF	@COL	HOU	@MIL	@LA	SD	@T17	ARI	@ATL	@T20	T19
@NYM	ATL	@T20	@PIT	CHI	MON	@STL	@FLA	CIN	COL	@SD	@MIL	LA	HOU	@SF	@T19	@T18	T17	ARI	PHI
T17	T20	@T19	@FLA	MON	CHI	@COL	@PIT	MIL	@STL	@SF	CIN	HOU	LA	@SD	@T18	@ATL	ARI	PHI	@NYM
PHI	T17	@ATL	T20	@PIT	FLA	@MIL	STL	@CHI	CIN	COL	@HOU	SD	@SF	T18	T19	@NYM	@LA	@ARI	@MON
NYM	@ATL	T20	PIT	@CHI	@MON	STL	FLA	@CIN	@HOU	MIL	SD	@LA	@COL	SF	T18	T19	@ARI	@T17	@PHI
@MON	@T20	PIT	ATL	@STL	@PHI	@CHI	CIN	FLA	@COL	SD	MIL	@ARI	@HOU	T19	SF	T18	@T17	@LA	NYM
@PHI	@T19	ATL	@STL	PIT	@FLA	HOU	@COL	MON	@SD	@CIN	CHI	@T17	MIL	@T18	@T20	SF	LA	NYM	ARI
ARI	PHI	@NYM	@MIL	@CIN	HOU	FLA	@SF	COL	MON	@PIT	@STL	CHI	@T18	@T20	@ATL	@T19	SD	T17	LA
LA	ARI	@T18	@CIN	@MIL	COL	MON	HOU	@SF	FLA	@CHI	@PIT	STL	@T19	@ATL	@NYM	T20	PHI	SD	@T17
SD	CIN	@T17	T19	@HOU	@MIL	@NYM	COL	@LA	PIT	FLA	@CHI	T18	@ATL	STL	T20	PHI	@SF	@MON	@ARI
@T17	@CHI	CIN	MIL	T19	@HOU	@PHI	NYM	@ARI	@MON	PIT	T18	@SD	SF	T20	STL	ATL	@COL	@FLA	@LA
@SF	@STL	T19	CHI	MIL	@COL	@T20	@MON	NYM	@FLA	@LA	PIT	ATL	T18	HOU	T17	@ARI	@SD	@PHI	CIN
@SD	@CIN	CHI	@T17	T20	MIL	NYM	@PHI	SF	@PIT	@T18	ARI	@STL	ATL	@T19	@COL	MON	HOU	LA	@FLA
FLA	LA	@CIN	@T18	@ATL	T20	PHI	SF	SD	ARI	@T19	@T17	@CHI	@STL	@NYM	@MIL	COL	MON	HOU	@PIT
MON	@T17	LA	@ATL	@T19	STL	SF	T20	@PIT	SD	ARI	@T18	@CIN	@MIL	@PHI	@HOU	NYM	COL	FLA	@CHI
PIT	@LA	STL	T18	@T20	@ATL	@HOU	SD	@PHI	T17	CIN	@ARI	T19	@CHI	NYM	COL	@MIL	@MON	@SF	FLA
@FLA	@ARI	T18	STL	ATL	@T20	@SD	@LA	@MON	@SF	T17	T19	MIL	CIN	CHI	NYM	@HOU	@PHI	@COL	PIT
@MIL	T18	@STL	@T20	HOU	@LA	@SF	@ARI	PHI	ATL	@FLA	T17	CIN	T19	PIT	CHI	@COL	@NYM	@SD	MON
@STL	HOU	@MIL	@T19	@T18	SF	ARI	@T17	ATL	PHI	@NYM	LA	@PIT	T20	@COL	@CIN	CHI	FLA	MON	@SD
SF	@T18	@CHI	HOU	@T17	ARI	SD	PHI	T20	LA	@MON	@T19	@ATL	@CIN	@MIL	@PIT	FLA	NYM	COL	@STL
COL	T19	SF	T17	STL	SD	T20	ARI	@FLA	@T18	LA	@ATL	@PHI	@PIT	@HOU	@CHI	@MON	MIL	@NYM	@CIN
@ARI	STL	T17	@CHI	SD	@SF	@LA	MON	@NYM	@T19	T18	@T20	PIT	@FLA	CIN	ATL	@PHI	@HOU	MIL	COL
@LA	CHI	MIL	@HOU	@COL	@SD	@ARI	@NYM	T18	@PHI	MON	FLA	@T19	PIT	ATL	CIN	@T20	@STL	SF	T17
CHI	@SD	@SF	@COL	@LA	@T19	@T17	@ATL	@T20	T18	@ARI	MON	PHI	NYM	FLA	HOU	CIN	@MIL	PIT	STL
STL	@MIL	@COL	ARI	@SD	LA	T19	@T18	@ATL	NYM	@T17	PHI	T20	FLA	@PIT	@MON	HOU	CHI	@CIN	@SF
MIL	COL	@HOU	SF	ARI	T18	LA	T19	T17	@ATL	PHI	@NYM	@MON	@T20	@CIN	@FLA	@STL	@PIT	@CHI	SD
@HOU	SF	ARI	@SD	COL	T19	T17	T18	LA	@T20	ATL	@FLA	@NYM	MON	@STL	@PHI	@CIN	@CHI	@PIT	MIL
@COL	MIL	@SD	@LA	SF	@ARI	@T18	T17	@T19	@NYM	T20	ATL	@FLA	PHI	MON	PIT	@CHI	CIN	STL	@HOU
HOU	@COL	@LA	@SF	@ARI	@T17	@T19	@SD	@T18	T20	@ATL	NYM	MON	CHI	PHI	FLA	PIT	STL	CIN	@MIL
CIN	@HOU	@ARI	SD	@SF	@T18	@ATL	@T19	@T17	@LA	NYM	T20	FLA	@MON	MIL	PHI	STL	PIT	CHI	@COL
@PIT	@SF	SD	LA	T18	ATL	COL	@T20	ARI	@T17	T19	@CIN	NYM	@PHI	@MON	@STL	MIL	@FLA	@HOU	CHI
@CHI	SD	HOU	COL	LA	T17	T18	ATL	T19	@ARI	@PHI	@MON	@T20	@NYM	@FLA	MIL	@PIT	@CIN	@STL	SF
@CIN	PIT	COL	@ARI	T17	@NYM	ATL	LA	@SD	T19	@T20	@PHI	@T18	STL	@CHI	MON	@FLA	SF	@MIL	HOU

References

[1] E. Aarts, J. Korst, Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, Chichester, 1989.

[2] A. Anagnostopoulos, L. Michel, P. Van Hentenryck, Y. Vergados, A simulated annealing approach to the traveling tournament problem, in: Proceedings of CPAIOR'03, 2003.

[3] T. Benoist, F. Laburthe, B. Rottembourg, Lagrange relaxation and constraint programming collaborative schemes for traveling tournament problems, in: Proceedings of CPAIOR'01. Wye College, UK, 2001, pp. 15–26.

[4] K. Easton, G. Nemhauser, M. Trick, The traveling tournament problem description and benchmarks, in: Proceedings of the 7th International Conference on Principles and Practice of Constraint Programming, CP 2001, Paphos, Cyprus, 2001, pp. 580–584.

[5] M. Henz, Constraint-based round robin tournament planning, in: Proceedings of the 1999 International Conference on Logic Programming, 1999. Las Cruces, NM, USA.

[6] M. Henz, Scheduling a major college basketball conference: Revisited, Operations Research 49 (2000) 163–168.

[7] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning, Operations Research 37 (1989) 865–892.

[8] S. Kirkpatrick Jr., C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[9] P.J.M. Laarhoven, E.H.L. Aarts, Simulated Annealing: Theory and Applications, Reidel, Dordrecht, 1987.

[10] B.T. Lowerre, The HARPY Speech Recognition System, PhD thesis, Carnegie-Mellon University, USA, April 1976.

[11] G. Nemhauser, M. Trick, Scheduling a major college basketball conference, Operations Research 46 (1998) 1–8.

[12] K.J. Nurmela, P.R.J. Ostergard, Constructing Covering Designs by Simulated Annealing. Technical report, Digital Systems Laboratory, Helsinki University of Technology, Otaniemi, Finland, 1993.

[13] C.C. Ribeiro, S. Urrutia. Heuristics for the mirrored traveling tournament problem, in: Proceedings PATAT'04, 2004. Pittsburgh, USA.

[14] S. Rubin, The ARGOS Image Understanding System. PhD thesis, Carnegie-Mellon University, USA, April 1978.

[15] M. Trick, Challenge Traveling Tournament Problems, 2004 http://mat.gsia.cmu.edu/TTP/.

[16] X. Zhang, Constraint Programming, Simulated Annealing and Hill-Climbing Algorithm for Traveling Tournament Problems. Technical report, School of Computing, National University of Singapore, 2002.