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

An Ant Based Hyper-heuristic for the Travelling Tournament Problem

Pai-Chun Chen, Graham Kendall and Greet Vanden Berghe

Abstract: **The Travelling Tournament Problem is a challenging sports timetabling problem which is widely believed to be NP-Hard. The objective is to establish a feasible double round robin tournament schedule, with minimum travel distances. This paper investigates the application of an ant based hyper-heuristic algorithm for this problem. Ant algorithms, a well known meta-heuristic, have been successfully applied to various problems. Whilst hyper-heuristics are an emerging technology, which operate at a higher level of abstraction than meta-heuristics. This paper presents a framework which employs ant algorithms as a hyper-heuristic. We show that this approach produces good quality solutions for the traveling tournament problem when compared with results from the literature.**

I. Introduction

Across the world sports leagues receive significant income from television and radio. For example, the UK based football team, Manchester United, has a market value of over 400 million pounds, largely due to its TV exposure. In the United States, television networks pay over 400 million dollars annually for nationally televised baseball games and that much again for local presentations [1].

An attractive sports schedule can generate large incomes, whilst poor schedules can decrease revenue. Furthermore, there are many smaller leagues that would benefit from better schedules so as to increase their revenue.

There have been various studies of sports schedules such as those for Minor League Baseball [2] and college basketball [3]. This paper tackles the Travelling Tournament Problem (TTP), which is an abstract instance of Major League Baseball (MLB). There have been many approaches to tackling the TTP (see table I).

In this paper we propose a new approach; an ant based hyper-heuristic. The novelty of this approach is that we can tackle all problem instances using the same algorithm, and using the same parameters.

Manuscript received January 30, 2007.

Pai-Chun. Chen is with the School of Computer Science at the University of Nottingham, UK. (e-mail: pzc@cs.nott.ac.uk)

Graham Kendall is with the School of Computer Science at the University of Nottingham, UK. (+44 115 846 6514, e-mail: gxk@cs.nott.ac.uk)

Greet Vanden Berghe is with KaHo Sint-Lieven, Belgium (e-mail: greetvb@kahosl.be)

By studying table I, it is apparent that different approaches are able to produce the best known solutions on just some of the problems. Whilst we are not aiming to be the best on any one instance, we aim to show that our approach is able to produce good quality solutions across a range of problem instances.

This paper is organised as follows. In section II, we describe the TTP in more detail and also introduce hyper-heuristics and ant algorithms. Section III provides a description of our proposed algorithm. Results and conclusions are presented in sections IV and V.

II. The Travelling Tournament Problem

The formal definition of the TTP is as follows:

Input: n teams (n is even).
D, an nxn symmetric integer distance matrix.

Output: a schedule for a double round-robin tournament with n teams

The aim is to minimise the total travel distance, whilst satisfying the following constraints.

- The schedule must represent a valid double round-robin tournament. That is, each team i plays one home game and one away game with every other team, j ($i \neq j$).
- Repeaters (team i at team j, followed immediately by team j at team i) are not allowed.
- No team is allowed to play more than three consecutive home games or three consecutive away games.

These definitions refer to the rules described by [1,4,5].

A double round robin tournament has exactly $2n$-2 rounds and n/2 games are played in every round. For example, a tournament of six teams has ten rounds and in each round three games are held.

As stated above, the main objective is to minimise the total travel distance. The total travel distance is defined as the sum of the travel distance of all teams. Each team starts at home, travels to the match venues according to the schedule and returns back home at the end of the schedule. For example (refer to figure 1), team ATL plays against teams FLA (home), NYM (home), PIT (home), PHI (away), MON (away), PIT (away), PHI (home), MON (home), NYM (away) and FLA (away). The distance matrix, D, figure 2, allows us to calculate.

$$d_{ATL,PHI} + d_{PHI,MON} + d_{MON,PIT} + d_{PIT,ATL} + d_{ATL,NYM} + d_{NYM,FLA} + d_{FLA,ATL}$$

$$= 665+380+408+521+745+1090+605 = 4414$$

where $d_{xxx,yyy}$, is the distance between *xxx* and *yyy*.

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

Figure 1. An example TTP schedule. @ represents a team playing away

	ATL	NYM	PHI	MON	FLA	PIT
ATL	0	745	665	929	605	521
NYM	745	0	80	337	1090	315
PHI	665	80	0	380	1020	257
MON	929	337	380	0	1380	408
FLA	605	1090	1020	1380	0	1010
PIT	521	315	257	408	1010	0

Figure 2. Distance matrix for a 6x6 TTP

Summing all the distances for each team, leads to the total distance for a given schedule.

Many different algorithmic techniques have been applied to the Travelling Tournament Problem. Easton et al. [1,4] introduced the description and the benchmarks of the TTP and presented the first solution approaches; Integer and Constraint Programming. They did not present results for all the instances. One remarkable fact is that their approach necessitated approximately 4 days computation for the NL8 instance (the instances provided by [1,4] are described by NL*n*, where NL means National League and *n* indicates how many teams are involved) using parallel programming on 20 CPU processors compared with a few minutes computation time for NL6.

Benoist et al. [6] used a combination of Constraint Programming and Lagrange relaxation to reach optimal solutions for NL4 and NL6. They also obtained feasible solutions for the larger instances, but did not beat the previous solution for NL16.

Cardemil [7] also achieves good results for NL4 to NL16, employing tabu search with the results being significantly better than [1,4,6].

Lim et al. [8] combined simulated annealing and hill-climbing. They divide the solution space into two sub-spaces; a timetable space and a team assignment space. Simulated annealing is used to search the feasible timetable space, while hill-climbing generates and improves team assignment for the given timetable in the team assignment space.

Crauwels and Oudheusden [9] investigate ant colony optimisation for the TTP. Compared with other studies, the results make no significant improvement, but the authors emphasise that their aim was to investigate an implementation of an ACO algorithm which might be used to solve other hard combinatorial optimisation problems.

Shen and Zhang [10] present a meta-heuristic named "greedy big step" which combines local search, branch-and-bound and constraint satisfaction. Compared with previous studies, their results are satisfactory.

Langford's [11] results are shown on Michael Trick's web site (http://www.tsp.gatech.edu/). The result for NL10 (59436) is the best known solution for NL10. Zhang [12] also presents his results on the TTP website. The results are competitive for the larger instances.

Gaspero and Schaerf [13] propose a family of tabu search algorithms. Their results show that the new method is competitive with those already in the literature.

In existing studies, the best approach to date is a simulated annealing method, proposed in [5]. They have recently presented the best results for many of the instances.

A summary of methods, and best results, are shown in table I. This table represents the best results as at 15th August 2006.

A. Hyper-heuristics

Hyper-heuristic approaches "*broadly describe the process of using meta-heuristics to choose meta-heuristics to solve the problem in hand*" [14]. A hyper-heuristic does not operate on the problem directly. Instead it utilises low level (meta-)heuristics and, as such, does not have domain knowledge of the problem over which it operates. The broad aim of hyper-heuristics is to raise the level of generality at which search algorithms can operate. That is, they aim to be applicable to a wide range of problem instances; and even problem domains.

Ross [16] says, "*The broad aim (of hyper-heuristics) is to discover some algorithm for solving a whole range of problems that is fast, reasonably comprehensible, trustable in terms of quality and repeatability and with good worst-case behaviour across that range of problems.*"

Hyper-heuristics have been applied to a variety of problems, using a variety of solution methodologies. Burke et al. [17] used a tabu search based algorithm as a hyper-heuristic and it was shown to operate well across three different problem domains. Ross et al. [18] used a learning classifier system (XCS) in order to solve a large set of one-dimensional bin packing problems. They show that individual heuristics are not able to find optimal solutions to any of the problems, but using a hyper-heuristic, to combine heuristics, they were able to find optimal solutions for 78% of the cases. Other hyper-heuristic examples can be found in [19], [20,21] (timetabling), [22] (determining shipper sizes),

[23] (presentation scheduling), [24] (shelf space layout), [25,26] (channel assignment) and [27] (component placement).

Introductions to hyper-heuristics can be found in [16,28].

B. *Ant Algorithms*

Ant Colony Optimisation algorithms (ACOs) were inspired by the observation of real ant colony behaviour. Dorigo et al. [29] proposed the first ACO algorithm, Ant System (AS), which has a set of artificial ants which exchange information using a pheromone trail. There are differences between artificial ants and natural ants. Artificial ants are not completely blind. As well as using a pheromone trail, they also use heuristic information to *look ahead*. Artificial ants can also recall complete routes they have previously taken. They also move in discrete time steps [29].

AS algorithms have been applied to the Travelling Salesman Problem and have generally achieved good results, giving a good solution, for example, to a 75–city problem [29]. To improve the performance of the AS algorithm, Dorigo and Gambardella [30] introduced the Ant Colony System (ACS) and compared it to other meta-heuristic methods such as genetic algorithms, evolutionary programming, and simulated annealing. They achieved impressive results for a 100-city problem. In addition, new improved versions of ACO algorithms are continually being proposed, for example: Max-Min Ant System, MMAS [31], Approximate Nondeterministic Tree-Search [32] and The Rank-based Version of Ant System, ASrank [33] etc.

Apart from the Travelling Salesman Problem, ant algorithms have been applied to many other combinatorial optimisation problems such as the Job-Shop Scheduling Problem [34], the Quadratic Assignment Problem [32,35,36], Vehicle Routing [37,38], Vehicle Routing with time windows [39], Graph Coloring [40] and Sequential Ordering [41].

III. Proposed Algorithm

One of the basic premises of an ant algorithm is to model the problem as a (normally) fully connected graph. The ants use the amount of pheromone on the edges to decide which vertex they should visit from the current vertex. The vertices represent some aspect of the problem. In the TSP, for example, each edge is a city. In our representation, each vertex represents a heuristic to be applied to the current solution and return a new solution.

A previous ant-algorithm-hyper-heuristic [42] uses a similar idea. There is a network in which every vertex represents a low-level heuristic. A number of ants, each of which represents a hyper-heuristic agent, are located uniformly among the vertices of the network and carry initial solutions. Each ant traverse particular edges and reach the next vertex. Once an ant arrives at a new vertex it applies the low-level heuristic at that node.

Unlike the TSP, when solved by ant algorithms, the ants in our representation are allowed to visit the same vertex many times. Indeed, they can cycle back to the same vertex; so that the same heuristic is repeatedly applied to the current solution. This is unlike the TSP, where each vertex can only be visited once, in any one tour, due to the constraints that must be respected for that problem.

In addition to the pheromone trails, the TSP ant algorithm provides ants with additional information. Visibility (as it is known) provides the ants with information as to how far each unvisited city is away from its current destination. By combining the pheromone information (which indicates how many ants have used that route before), with the visibility (the distance to the potential next cities), the ants probabilistically choose which city to visit next.

In the proposed algorithm, we still use the idea of pheromone trails. Visibility is also used, but this represents how quickly the heuristic at a potential vertex takes to compute. This is on the assumption that short, good quality heuristics are to be preferred to good quality heuristics that take a long time to compute (or even computationally expensive, yet poor, heuristics).

More formally, we can say that an ant k, located at vertex i, applies the low-level heuristic i to its own solution. The next destination, j, will be probabilistically selected using visibility, η_j, and the level of pheromone, τ_{ij}. η_j is information about heuristic j that indicates how well the heuristic performs. The pheromone value τ_{ij} is a probability proportional to the intensity of the pheromone trail laid on the edge from vertex i to vertex j (see figure 3).

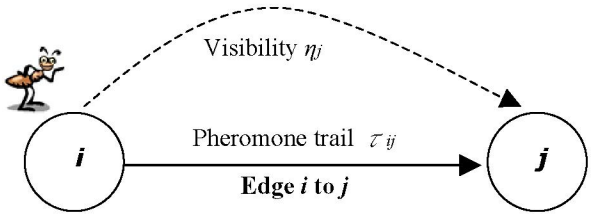

Figure 3. Design outline of proposed ant algorithm hyper-heuristic [23]

After visiting vertex j and implementing the low-level heuristic at that vertex, the ant uses the evaluation (objective) function to calculate the level of improvement (positive or negative) and deposits an amount of pheromone, which is proportional to the change in the evaluation. Like the TSP, the pheromone must not grow unbounded and there is a need for an *evaporation* function so that pheromone levels decay over time.

However, it is notable that since ants immediately judge the degree of improvement of their solution implemented by low-level heuristics, this judgement always leads to more successful low-level heuristics being rewarded with higher levels of pheromone than less successful heuristics.

Therefore, no ants will visit these less-successful heuristics, which could decrease the quality of the final solution. At first sight this might seem reasonable but poorly performing low-level heuristics may enable escape from local optima, and allow the *good* heuristics to find even better solutions. This is why [42] suggests that ants should not make judgements immediately after each move, but only after a complete *journey* has been completed. For example, an ant with a journey length of 4 will visit four vertices (four low-level heuristics) then judge the quality of its journey. It is more important that the overall sequence of steps consisting of "good" and "bad" moves generates an appreciable improvement than to find individually "good" moves. Indeed, this is the idea behind being able to combine a number of heuristics in order to find a *good set* of heuristics [18]. As regards to the length of the journey, Burke et al. [42] experimented with various journey lengths. Results for experiments with a length equal to the number of vertices (or low-level heuristics) are generally good, so we adopt the same for our experiments.

A. Low Level Heuristics

As stated above, each vertex in the fully connected graph represents a heuristic. In this work we have 10 heuristics (H01 – H10). In the descriptions below, we refer to a team (a single team in a TTP) and a round (a slot when all teams are playing).

For the low-level heuristics H01 to H05, the choice of team or round is randomly selected. For H06 to H10, the behaviour is the same as for H01 to H05, but these heuristics explore all possible neighbourhood moves and selects the one which minimizes the objective function. Hence, H06 to H10 are computationally more expensive.

H1: Partial Swap Rounds: This low-level heuristic is a partial swap function which swaps any two random rounds for a random team. This heuristic carries out a repair chain in order to maintain the feasibility of the schedule.

H2: Swap Teams: This low-level heuristic swaps any two random teams

H3: Swap Home-Away: This low-level heuristic swaps the home/away matches of teams i and j. If team i plays a home game against team j in round k (and plays an away game at round l), rounds k and l will be exchanged. Team j's match will also change to correspond to the change in team i's match.

H4: Swap Rounds: The following heuristic simply swaps rounds i and j and there is no repair chain.

H5: Shift Move: Randomly choose rounds j and k ($j < k$; $j \neq k$), remove round j and insert it in the position of round k, the intermediate rounds are shifted one slot forward.

The low-level heuristics described above are inspired by [5] and [9]. As Anagnostopoulos et al. [5] points out, H02 (Swap Teams), H03 (Swap Home-Away) and H04 (Swap Rounds) are "*not sufficient for exploring the entire search space and, as a consequence, they lead to suboptimal solutions for large instances*". Therefore, to improve these results they consider two methods: Partial-Swap Rounds and Partial-Swap Teams, which significantly expands the neighbourhood, creating a more densely linked search space. The advantage of this is that although these two moves are not as *global* as the macro moves H02 to H05, they may accomplish a better trade-off between feasibility and optimality by improving feasibility in one part of the TTP schedule, while not undermining feasibility in another. They are also more *global* than the micro-moves [5]. H05 (Shift-Move) is taken from [9], which can also be considered a kind of micro-move.

B. The Algorithm

Below we present the algorithm and, following, some explanations

$t = 0$ // t is a counter
$l = 0$ // l is the counter of the journey length
Sb = null // best solution
S_k = null // each ant's solution
For each vertex j: set $\eta_j = 1$
For each edge(i, j): set $\tau_{ij} = 1$

// Provide each ant with an initial solution
For each ant k: S_k = RandomInitSolution()

Put m ants uniformly on the n vertices.
For each ant k: apply low-level heuristic from ant k's current location to S_k.

While (NOT Stopping-condition())
  While ($k = 1$ to m)
    //Choose next vertex based on pheromone and visibility
    Ant k chooses the next vertex j
    Move ant k to vertex j
    Apply heuristic j to S_k to produce new solution S'_k
    if (eval(S'_k) < eval(Sb)) $Sb = S'_k$
  End while
  $t = t + 1$
  $l = l + 1$

Update Visibility :
 While ($j = 1$ to n) do
   Update η_j
 End while

Update Pheromone :
  if ($l = = n$) then // we have completed a journey
    $l = 0$
    Update $\tau_{ij}(t)$
    if (any new best solution found in this journey) then
        For all ants : $S_k = Sb$
    End if
  End if

Escape Local Optimal:
  if(no new best solution found for x iterations) then

```
    Adjust-Objective-Function()
  End if
End while
```

We used a random initial solution generator from [5] to generate a double round robin schedule. They state that "*this procedure is very simple and can be improved considerably, but it appears sufficient to find schedules satisfying the hard constraints reasonably fast*".

Each ant has to choose its next heuristic (vertex). This *transition probability* uses the same formula as the original ant system formulation [29].

As stated in the algorithm, the objective function is adaptive (see the end of this section for more discussion on this point) and is defined as follows:

$$Objective(S) = \begin{cases} \text{total-travel-distance}(S) & \text{if S is feasible timetable} \\ \text{total-travel-distance}(S) + \text{penaltyFunction(S)} & \text{if S is infeasible timetable} \end{cases}$$

$$penaltyFunction(S) = pv(S) \ * \ numv(S)$$

Infeasible solutions are penalised according to how many times they violate constraints. *numv*(*x*) denotes the number of violations. *pv*(*x*) is the penalty value, which is a fixed number depending on the number of teams. For example, if the NL8 instance violates the *NoRepeat* constraint twice and "*Greater Than Three Consecutive Home/Aways*" five times, the penalty value is 7 * *pv*(8). *pv*(*x*) was found empirically and is just the best known solution (see table I) divided by 100.

The visibility updating formula is given by:

$$\eta ij(t) = \rho * \eta ij(t-1) + \sum_{k}^{m} \frac{\lambda^{Ikj(t)}}{Tkj(t) * num(i,j)}$$

ρ is a constant between 0 and 1 to provide decay. $I_{kj}(t)$ is the improvement of the k^{th}'s ant current solution, S_k. This value could be negative (heuristic *j* decreases the quality of the solution) so we provide a monotonic conversion λ (λ is a static number; 1.0001) to convert the negative value to a small positive value and so still give poorly performing heuristics some probability of being selected. $T_{kj}(t)$ is the amount of time heuristic *j* uses for the k^{th}'s ant current solution at time *t*. *num*(*i*, *j*) is the number of edges the ants have used from the start until now. This stops the ants from depending on a particular edge. Burke et al. (2003b) suggested, that visibility should be defined as 1 divided by the amount of CPU time taken to implement heuristic *j*. However, after experimentation we found that the above formula leads to better results.

Pheromone updates are carried out according to the following formulae (in fact they closely mimic the original ant algorithm formulation):

$$\tau ij(t) = \rho \cdot \tau ij(t\text{-}n) + \sum_{k=1}^{m} \Delta \tau_{ij}^{k}$$

Where ρ=0.5 is an evaporation parameter and $\Delta \tau_{ij}^{k}$ is the quantity of pheromone associated with edge(*i*, *j*) to be laid by ant *k* during this journey and is given by.

$$\Delta \tau_{ij}^{k} = \begin{cases} \frac{I_k}{L_k} + Q & \text{If } I_K > 0 \\ 0 & \text{Otherwise} \end{cases}$$

Where Q is a constant which increases pheromone value by resisting evaporation. L_k is the length of ant *k*'s journey. I_k is the improvement contributed by the low-level heuristics (vertices) that ant *k* used (visited) during this journey, and only if improvement occurs is edge(*i*, *j*) rewarded with pheromone.

Anagnostopoulos et al. [5] demonstrated the importance of balance in exploring both feasible and unfeasible areas. They use a Strategic Oscillation strategy to search both areas equally and achieved good results. For our experiments, we also use this idea of searching both feasible and infeasible regions. An early experiment showed that when the objective function was used without a penalty function, the majority of solutions found were unfeasible. By contrast, only poor feasible solutions were found when the objective function used a high penalty value. The penalty value is adjusted according to the progress of the solution search. If no new best solution is found within a certain number of iterations, the search may be trapped in a local optima; so, the penalty value, *pv*(*x*), of the objective function is reduced by half at each iteration until a new best solution is found, the penalty value then returns to its normal level.

IV. Results

All algorithms were run on a laptop Intel(R) Pentium(R) M processor 1.73 GHz with 512M RAM running under Microsoft Windows XP Home Edition version 2002 service Pack 2.

Table I shows the comparison of existing literature results, compared to our best results (see last row). The hyper-heuristic reached the optimal solution for NL4 (8276) in every run (these results report the best result from 30 runs). For NL6, which we also find the optimal solution, only Zhang [12] fails to find the optimal solution. Our result produced the optimal solution in about 45 seconds. NL8 is quite challenging. The best result (39721) was found by Anagnostopoulos et al. [5]. Lim et al. [8] also found the same result. Although our result cannot beat this, we still obtain a good quality solution (the difference between the

best and ours is about 16%) and we also beat some previous work such as [1,4,6,7,9]. One notable performance is that finding this solution using a hyper-heuristic only took about 481 seconds (250 iterations). It is quite fast in comparison with result of Easton et al. [1,4] which spent approximately 4 days and used 20 processors. For the even larger instances (NL10 to NL16) our approach does not beat the best known results. However it still produces good quality solutions and also beats some existing studies (e.g. NL12: [1,4,6,7,9].

There is another notable comparison that we can make. We have used an ant colony algorithm, and when compared against the only other method using this approach [9], we observe that all our results are superior.

V. Conclusions and Discussion

This paper has developed an ant algorithm based hyper-heuristic that used the Travelling Tournament Problem as an experimental testbed. The results show that the hyper-heuristic can find optimal solutions for small instances and is able to produce good quality solutions for larger instances. The hyper-heuristic approach is able to beat some of the results reported in the literature [1,4,7,9].

Although the proposed approach cannot beat the best results from the literature, it is the first time that a hyper-heuristic has been tested on this problem.

The hyper-heuristic does not use complex low-level (meta-)heuristics, but utilises five simple, straightforward neighbouhood algorithms in order to obtain high quality, feasible solutions. Similar to previous studies of hyper-heuristics, this paper has demonstrated that hyper-heuristics can easily be applied to new problems utilising a flexible framework: simple low-level heuristics, an objective function and a high level manager which intelligently manages low-level heuristics to search the solution space.

An encouraging faeture is that our ant based approach is able to beat the results of the only other ant algorithm reported for the TTP [9].

There are still many other research directions that can be followed in the future. It would be nice to extend this work so that we are more competitive with other results for instances NL8-NL16. We would also like to test the proposed method on some larger instances which have recently been introduced.

We would also like to investigate other hyper-heuristic approaches on this problem; from the many that now appear in the literature.

This has been a challenging investigation. We have introduced a new hyper-heuristic and tested it on a challenging sports scheduling problem. The results are very encouraging and we hope it motivates other researchers to continue to investigate this area.

References

[1] Easton, K., G.L. Nemhauser, and M.A. Trick (2001). The travelling tournament problem: description and benchmarks. Principles and Practice of Constraint Programming CP 2001, Springer, LNCS 2239, pp. 580-585

[2] Russell, R.A. and Leung, (1994). Devising a cost effective schedule for a baseball league. Operations Research 42 (4), pp. 614-625

[3] Nemhauser, G. and Trick, M. A. (1998). Scheduling a major college basketball. Conference George Operations Research, 46(1) , pp. 1-8

[4] Easton, K., G.L. Nemhauser, and M.A. Trick (2002). Solving the travelling tournament problem: a combined integer programming and constraint programming approach Proceedings of the 4th international conference on the Practice and Theory of Automated Timetabling, Gent, Belgium, pp. 319-330

[5] Anagnostopoulos, A., Michel, L., Hentenryck, P., Vergados, Y. (2006). A simulated annealing approach to the travelling tournament problem. Journal of Scheduling, 9(2), pp. 177-193

[6] Benoist, T., Laburthe, F. and Rottembourg, B. (2001). LagrangeRelaxation and Constraint Programming Collaborative Schemes for Travelling Tournament Problems. In CP-AI-OR'2001,Wye College (Imperial College), Ashford, Kent UK

[7] Cardemil, A. (2002). Optimizacion de fixtures deportivos: Estado del arte y un algoritmo tabu search para el travelling tournament problem. tesis de Licenciatura, Departamento de ComputaciÓn Facultad de Ciencias Exactas y Naturales Universidad de Buenos Aires avaliable at :http://www.dc.uba.ar/people/profesores/willy/tesistas.html

[8] Lim A., Rodrigues B. and Zhang X. (2006). A simulated annealing and hill-climbing algorithm for the traveling tournament problem, EJOR 174:3, pp 1459-1478

[9] Crauwels, H., Oudheusden, D.V. (2003). Ant colony optimization and local improvement. The Third Workshop on Real-Life Applications of Metaheuristics, Antwerp

[10] Shen, H., Zhang, H. (2004). Greedy big steps as a meta-heuristic for combinatorial search. The University of Iowa AR Reading Group, Spring 2004 Readings. http://Goedl.cs.uiowa.edu/classes/AR-group/04spring.html

[11] Langford (2004). In Challenging Travelling Tournament Instances http://mat.gsia.cmu.edu/TOURN/ [Accessed 15/08/06]

[12] Zhang (2002). In Challenging Travelling Tournament Instances http://mat.gsia.cmu.edu/TOURN/ [Accessed 15/08/06]

[13] Gaspero, L. D. and Schaerf, A. (2006). A composite-neighborhood tabu search approach to the travelling tournament problem. Journal of Heuristics, to appear

[14] Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P. and Schulenburg, S. (2003a). Hyper-heuristics: An emerging direction in modern searchtechnology. In: Handbook of Meta-Heuristics, F. Glover and G. Kochenberger (Eds.), Kluwer, pp. 457–474

[16] Ross, P. (2005). Hyper-heuristic search methodologies. In: Burke, Edmund K.; Kendall, Graham (Eds.) Introductory Tutorials in Optimization and Decision Support Techniques, Springer, pp.529-556 (chapter 17)

[17] Burke E.K., Kendall G. and Soubeiga E. (2003). A Tabu-Search Hyper-Heuristic for Timetabling and Rostering. Journal of Heuristics, 9(6), 451-470, 200

[18] Ross P., Marín-Blázquez J. G., Schulenburg S. and Hart E. (2003). Learning a Procedure That Can Solve Hard Bin-Packing Problems: A New GA-Based Approach to Hyper-heuristics. LNCS 2724, pp 1295-1306 (Proceedings of GECCO 2003)

[19] Burke E.K., McCollum B., Meisels A., Petrovic S. and Qu R. (2007). A Graph-Based Hyper-Heuristic for Timetabling Problems, EJOR 176:1, 177-192

[20] Kendall G. and Hussin M. (2005). A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology. In Revised Selected Papers of the 5th International Conference of Practice and Theory of Automated Timetabling V (PATAT 2004), Burke E. and Trick T. (eds), LNCS 3616, pp 270-293, 2005a

[21] Kendall G. and Mohd Hussin N. (2005). An Investigation of a Tabu-Search-Based Hyper-heuristic for Examination Timetabling, Multidisciplinary Scheduling; Theory and Applications. Kendall G., Burke E., Petrovic S. and Gendreau M (eds), Springer, 309-328, (selected volume from the conference)

[22] Dowsland K., Soubeiga E. and Burke E.K. (2006). A Simulated Annealing Hyper-heuristic for Determining Shipper Sizes, EJOR, accepted

[23] Burke E.K., Kendall G., Landa Silva D., O'Brien R. and Soubeiga. (2005). An Ant Algorithm Hyperheuristic for the Project Presentation Scheduling Problem. In Proceedings of 2005 IEEE Congress on Evolutionary Computation (CEC'05), 2-5 September, Edinburgh, Scotland, pp 2263-2270

[24] Bai, R. and Kendall, G. (2005). An Investigation of Automated Planograms Using a Simulated Annealing Based Hyper-heuristics. In: Ibaraki, T., Nonobe, K., and Yagiura, M. (Eds.), Metaheuristics: Progress as Real Problem Solvers - (Operations Research/Computer Science Interfaces Series, Vol. 32), Berlin, Heidelberg, New York, Springer, pp. 87-108, ISBN: 0-387-25382-3

[25] Kendall G. and Mohamad M. (2004). Channel Assignment Optimisation Using a Hyper-heuristic. Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems (CIS2004), pp. 790-795

[26] Kendall G. and Mohamad M. (2005). Channel Assignment in Cellular Communication Using a Great Deluge Hyper-heuristic. In proceedings of the 2004 12th IEEE International conference on networks (ICON 2004), pp 769-773

[27] Ayob, M. and Kendall, G. (2003). A Monte Carlo Hyper-Heuristic to Optimise Component Placement Sequencing For Multi Head Placement Machine. In Proceedings of the International Conference on Intelligent Technologies, InTech'03, pp 132-141, Chiang Mai, Thailand, Dec 17-19, pp 132-141

[28] Burke E., Hart E., Kendall G., Newall J., Ross P. and Schulenburg S. (2003). Hyper-Heuristics: An Emerging Direction in Modern Search Technology. Handbook of Meta-Heuristics (Glover F., ed), pp 457 – 474, Kluwer

[29] Dorigo M., Maniezzo, V. and Colorni, A. (1996). Ant system: optimisation by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26:1, pp. 29-41

[30] Dorigo, M., Gambardella, L.M. (1997). Ant colony system: a cooperative learning approach to the travelling salesman problem. IEEE Transactions on Evolutionary Computation,1:1, 53-66

[31] Stützle, T. and Hoos, H. (1997). Improvements on the ant system: introducing max–min ant system. In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer Verlag, pp. 245–249

[32] Maniezzo, V. (1998). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Technical Report CSR 98-1, C. L. In Scienzedell'Informazione, Universit´a di Bologna, sede di Cesena, Italy

[33] Bullnheimer, B., Hartl, R. F. and Strauss, C. (1997). A new rank-based version of the ant system: a computational study. Technical Report POM-03/97, Institute of Management Science, University of Vienna

[34] Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M. (1994). Ant system for job-shop scheduling. Belgian Journal of Operations Research Statistics and Computer Science (JORBEL), 34, pp.39–53

[35] Maniezzo, V., Colorni, A. and Dorigo., M. (1994). The ant system applied to the quadratic assignment problem. Technical Report IRIDIA/94-28, Universit´e Libre de Bruxelles, Belgium

[36] Gambardella, L. M., Taillard, E. and Dorigo, M. (1997). Ant colonies for the QAP. Technical Report 4-97, IDSIA, Lugano

[37] Bullnheimer, B., Hartl, R. F. and Strauss, C. (1997). An improved ant system algorithm for the vehicle routing problem. Technical Report POM-10/97, Institute of Management Science, University of Vienna, 1997

[38] Bullnheimer, B., Hartl, R. F. and Strauss, C. (1998). Applying the ant system to the vehicle routing problem. In I. H. Osman S. Voss, S. Martello and C. Roucairol(Eds.) Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academics, pp. 109–120

[39] Gambardella, L. M., Taillard, E. and Agazzi., G. (1999). Ant colonies for vehicle routing problems. In: Corne, D., Dorigo, M. and Glover, F. (Eds), New Ideas in Optimization. McGraw-Hill

[40] Costa, D. and Hertz, A. (1997). Ants can colour graphs. Journal of the Operational Research Society, 48, pp. 295–305

[41] Gambardella, L. M. and Dorigo, M. (1997). HAS-SOP: An hybrid ant system for the sequential ordering problem. Technical Report 11-97, IDSIA, Lugano

[42] Burke, E. K. Kendall, G., O'Brien, R. F. J., Redrup, D., Soubeiga, E. (2003b). An ant algorithm hyper-heuristic. In Proceedings of The Fifth Meta-heuristics International Conference (MIC2003)

TABLE I
TTP SOLUTION METHODS. THE BEST RESULTS ARE IN BOLD

Author(s)	Method	NL4	NL6	NL8	NL10	NL12	NL14	NL16
Easton et al. [1,4]	Linear Programming (LP)	**8276**	**23916**	41113				312623
Benoist et al. [6]	A combination of constraint programming and lagrange relaxation	**8276**	**23916**	42517	68691	143655	301113	437273
Cardemil [7]	Tabu search	**8276**	**23916**	40416	66037	125803	205894	308413
Zhang [12]	Unknown (data from TTP Website)	**8276**	24073	39947	61608	119012	207075	293175
Shen and Zhang [10]	"Greedy big step" Meta-Heuristic			39776	61679	117888	206274	281660
Lim et al. [8]	Simulated Annealing and Hill-climbing	**8276**	**23916**	**39721**	59821	115089	196363	274673
Langford [11]	Unknown (data from TTP Website)				**59436**	112298	190056	272902
Crauwels and Oudheusden [9]	Ant Colony Optimization with Local Improvement	**8276**	**23916**	40797	67640	128909	238507	346530
Anagnostopoulos et al. [5]	Simulated Annealing	**8276**	**23916**	**39721**	59583	**111248**	**188728**	**263772**
Gaspero and Schaerf [13]	Composite-Neighbourhood Tabu Search Approach				59583	111483	190174	270063
This Paper	Ant Algorithm Hyper-heuristic	**8276**	**23916**	40361	65168	123752	225169	321037